Expected Complexity and Gradients of Deep Maxout Neural Networks and Implications to Parameter Initialization

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

DISSERTATION

zur Erlangung des akademischen Grades

DOCTOR RERUM NATURALIUM
(Dr.rer.nat.)

im Fachgebiet

Informatik

vorgelegt

von M.Sc. Hanna Tseran
geboren am 15.09.1992 in Minsk (Belarus)

Die Annahme der Dissertation wurde empfohlen von:

1. Prof. Dr. Guido Montúfar gemeinsam mit Prof. Dr. Bernd Sturmfels (Max-Planck-Institut für Mathematik in den Naturwissenschaften)
2. Prof. Dr. Gitta Kutyniok (LMU München)

Die Verleihung des akademischen Grades erfolgt mit Bestehen der Verteidigung am 08.11.2023 mit dem Gesamtprädikat magna cum laude.

Abstract

Learning with neural networks depends on the particular parametrization of the functions represented by the network, that is, the assignment of parameters to functions. It also depends on the identity of the functions, which get assigned typical parameters at initialization, and, later, the parameters that arise during training. The choice of the activation function is a critical aspect of the network design that influences these function properties and requires investigation. This thesis focuses on analyzing the expected behavior of networks with maxout (multi-argument) activation functions. On top of enhancing the practical applicability of maxout networks, these findings add to the theoretical exploration of activation functions beyond the common choices. We believe this work can advance the study of activation functions and complicated neural network architectures.

We begin by taking the number of activation regions as a complexity measure and showing that the practical complexity of deep networks with maxout activation functions is often far from the theoretical maximum. This analysis extends the previous results that were valid for deep neural networks with single-argument activation functions such as ReLU. Additionally, we demonstrate that a similar phenomenon occurs when considering the decision boundaries in classification tasks. We also show that the parameter space has a multitude of full-dimensional regions with widely different complexity and obtain nontrivial lower bounds on the expected complexity. Finally, we investigate different parameter initialization procedures and show that they can increase the speed of the gradient descent convergence in training.

Further, continuing the investigation of the expected behavior, we study the gradients of a maxout network with respect to inputs and parameters and obtain bounds for the moments depending on the architecture and the parameter distribution. We observe that the distribution of the input-output Jacobian depends on the input, which complicates a stable parameter initialization. Based on the moments of the gradients, we formulate parameter initialization strategies that avoid vanishing and exploding gradients in wide networks. Experiments with deep fully-connected and convolutional networks show that this strategy improves SGD and Adam training of deep maxout networks. In addition, we obtain refined bounds on the expected number of linear regions, results on the expected curve length distortion, and results on the NTK. As the result of the research in this thesis, we develop multiple experiments and helpful components and make the code for them publicly available.

Acknowledgements

I am grateful to the Max-Planck Institute for Mathematics in the Sciences (MPI MiS) and the Mathematical Machine Learning group within it for an opportunity to pursue my research interests. I am also thankful for the additional possibilities provided for me as a student at the IMPRS MiS and as a participating researcher in the Combinatorial and Implicit Approaches to Deep Learning project within the Priority Programme "Theoretical Foundations of Deep Learning" (SPP 2298).

I want to continue by expressing my gratitude to everyone who made this thesis possible. Firstly, I want to thank my supervisor Guido Montúfar for being the most supportive supervisor I could ever hope for. Thank you for helping me find and pursue my interests, helping me navigate through the fundamentals of deep learning theory, and helping me to learn how to communicate my research. But first and foremost, thank you for being a kind mentor. I also want to thank members of my thesis advisory committee, Jürgen Jost and Bernd Sturmfels, for their insightful comments on the machine learning problems during our meetings.

I am also grateful to my colleagues from the Mathematical Machine Learning group, Max-Planck Institute for Mathematics in the Sciences, and UCLA, whom I have met along the way. Thank you, Johannes Müller, Pierre Bréchet, Pradeep Banerjee, Katerina Papagiannouli, Michael Murray, Kedar Karhadkar, Rishi Sonthalia, Marie Brandenburg, and Jiayi Li, for all the exciting discussions we had together and all the projects we worked on. I enjoyed a lot communicating with you all.

I also want to express my sincere gratitude to all the amazing staff of MPI MiS who made my life in Leipzig so much easier, especially to Heike Rackwitz and Katharina Matschke. I also extend heartfelt thanks to the institute ombudsman, Jörg Lehnert, for being there whenever I had questions.

Additionally, I want to thank the great Amazon Berlin Machine Learning team people I met during my internship there: Cheng Wang, Gyuri Szarvas, Patrick Ernst, Georges Balazs, Pavel Danchenko, and Lahari Poddar. Thank you for teaching me the applied approach to natural language processing and helping me through my internship. The results we obtained are mentioned only in the introduction, but they have definitely broadened my machine-learning knowledge.

I want to end these acknowledgments with sincere thanks to my family, my husband Kirill, and my friends Toma and Masha for being there for me whenever I needed them, always believing in me, supporting my decisions even when they did not understand them fully and staying with me through the tough times the world and my home country, Belarus, has been experiencing while I was working towards this thesis. Special thanks go to Kirill for being so close to me despite the 10,000 kilometers separating us for the past two years.

Funding Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement n° 757983).

Contents

List of figures 7

List of tables 12

1 Introduction 14

1.1 Maxout networks . . . 15
1.1.1 Definition . . . 15
1.1.2 Significance of maxout network research . . . 16
1.2 Contributions and thesis outline . . . 19
1.3 Overview of the computational results . . . 21
1.4 Discussion . . . 22
1.5 Other projects developed during the Ph.D. studies . . . 23

2 Background 26

2.1 Network expressivity . . . 26
2.1.1 Connection to approximation . . . 26
2.1.2 Connection to memorization . . . 28
2.1.3 Complexity measures . . . 29
2.1.4 Complexity of maxout networks . . . 31
2.2 Parameter initialization in neural networks . . . 32
2.3 Tropical perspective on neural networks with piece-wise linear activation functions 33
2.3.1 Basic tropical geometry definitions . . . 34
2.3.2 Tropical geometry of neural networks . . . 35
2.3.3 Difference between linear regions of ReLU and maxout networks . . . 35
2.4 NTK and implicit bias . . . 36

3 On the expected complexity of maxout networks 38

3.1 Introduction . . . 38
3.2 Activation regions of maxout networks . . . 41
3.3 Numbers of regions attained with positive probability . . . 42
3.4 Expected number and volume of activation regions . . . 44
3.5 Expected number of pieces and volume of the decision boundary . . . 46
3.6 Experiments . . . 47
3.7 Discussion . . . 49

Proofs and experiment details **53**
3.A Proofs related to activation patterns and activation regions 53
3.A.1 Number of activation patterns 53
3.A.2 Generic correspondence between activation regions and linear regions . . . 54
3.A.3 Partial activation regions 55
3.B Proofs related to the generic numbers of regions 58
3.B.1 Number of regions and Newton polytopes 58
3.B.2 Bounds on the maximum number of linear regions 60
3.B.3 Numbers of regions attained over positive measure subsets of parameters . 60
3.B.4 Minimum number of activation regions 62
3.C Expected number of activation regions of a single maxout unit 65
3.D Proofs related to the expected volume 67
3.E Proofs related to the expected number of regions 71
3.F Upper bounding the constants 75
3.G Expected number of regions for networks without bias 80
3.H Proofs related to the decision boundary 82
3.H.1 Simple upper bound on the number of pieces of the decision boundary . . . 82
3.H.2 Lower bound on the maximum number of pieces of the decision boundary 82
3.H.3 Upper bound on the expected volume of the decision boundary 83
3.H.4 Upper bound on the expected number of pieces of the decision boundary . 84
3.H.5 Lower bound on the expected distance to the decision boundary 85
3.I Counting algorithms 86
3.I.1 Approximate counting of the activation regions 86
3.I.2 Exact counting of the activation regions 87
3.I.3 Exact counting of linear pieces in the decision boundary 87
3.I.4 Algorithm discussion 88
3.J Parameter initialization 90
3.J.1 He initialization 90
3.J.2 He-like initialization for maxout (Maxout-He) 90
3.J.3 Sphere initialization 93
3.J.4 Many regions initialization 93
3.J.5 Steinwart-like initialization for maxout 94
3.K Experiment details and additional experiments 94

4 Expected gradients of maxout networks and consequences to parameter initialization 103
4.1 Introduction 103
4.2 Preliminaries 106
4.3 Results 107
4.3.1 Bounds on the input-output Jacobian 107
4.3.2 Distribution of the input-output Jacobian 109
4.3.3 Activation length 110
4.4 Implications to initialization and network architecture 111
4.5 Implications to expressivity and NTK 112
4.5.1 Expected number of linear regions of maxout networks 112
4.5.2 Expected curve length distortion 113
4.5.3 On-diagonal NTK 113

4.6 Experiments . . . 114
4.7 Discussion . . . 116

Proofs and experiment details **118**
4.A Notation . . . 118
4.A.1 Variables . . . 118
4.A.2 Symbols . . . 120
4.B Basics . . . 121
4.B.1 Basics on maxout networks . . . 121
4.B.2 Basic notions of probability . . . 123
4.B.3 Details on the chain rule . . . 123
4.C Bounds for the input-output Jacobian norm $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$ . . . 123
4.C.1 Preliminaries . . . 123
4.C.2 Expression for $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$ . . . 124
4.C.3 Stochastic ordering for $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$ . . . 126
4.D Moments of the input-output Jacobian norm $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$ . . . 127
4.E Equality in distribution for the input-output Jacobian norm and wide network results 131
4.F Activation length . . . 140
4.G Expected number of linear regions . . . 144
4.H Expected curve length distortion . . . 146
4.I NTK . . . 148
4.J Experiment details and additional experiments . . . 150
4.J.1 Experiments with SGD and Adam from Section 4.6 . . . 150
4.J.2 Ablation analysis . . . 154
4.J.3 Batch normalization . . . 155
4.J.4 Comparison of maxout and ReLU networks in terms of the number of parameters . . . 155
4.J.5 Comparison of maxout and ReLU networks with dropout . . . 156
4.J.6 Gradient values during training . . . 157

Bibliography **160**

List of figures

1.1 Illustration of a simple maxout network with two input units, one hidden layer consisting of two maxout units of rank 3, and an affine output layer with a single output unit. 17

1.2 Example of a situation where the training is unsuccessful for a ReLU network because all neurons in the first layer are dead, while a maxout network trains successfully on the same dataset. We indicate the positions of the breakpoints in the first layer with the green × symbol. Notice that the breakpoints do not move during the training of a ReLU network but change their positions in a maxout network. 18

2.1 Left: Shown is a piecewise linear function $\mathbb{R}^2 \to \mathbb{R}$ represented by a network with a layer of two rank-3 maxout units for a choice of the parameters. The input space is subdivided into linear regions. Right: Shown is the number of linear regions of a 3-layer maxout network over a portion of the input space as a function of a 2D affine subspace of parameter values $\theta(\xi_1, \xi_2)$. Shown are also two examples of the input-space subdivisions of functions represented by the network for different parameter values. As the figure illustrates, the function taking parameters to the number of regions is rather intricate. In this work, we characterize values attained with positive probability and upper bound the expected value given a parameter distribution. . 30

2.2 Linear regions of ReLU networks vs. linear regions of maxout networks. Observe that regions of a maxout network are created by the intersection of more complex objects. 36

3.1 Left: Shown is a piecewise linear function $\mathbb{R}^2 \to \mathbb{R}$ represented by a network with a layer of two rank-3 maxout units for a choice of the parameters. The input space is subdivided into activation regions $\mathcal{R}(J; \theta)$ with linear regions separated by $\mathcal{X}_1(\theta)$. Right: Shown is the number of linear regions of a 3-layer maxout network over a portion of the input space as a function of a 2D affine subspace of parameter values $\theta(\xi_1, \xi_2)$. Shown are also two examples of the input-space subdivisions of functions represented by the network for different parameter values. More details about this figure are given in Section 3.K. As the figure illustrates, the function taking parameters to the number of regions is rather intricate. In this work, we characterize values attained with positive probability and upper bound the expected value given a parameter distribution. 43

3.2 Shown are means and stds for 30 maxout-He normal initializations for networks with $K = 2$ and $n_0 = 2$. Left: Comparison of the theoretically predicted growth $O(N^{n_0}/n_0!)$ and the experimentally obtained number of regions for networks with different architectures. Right: Comparison of the theoretically predicted growth $O(N)$ and the experimentally obtained number of linear pieces of the decision boundary for networks with different architectures. 49

3.3 Effect of the depth and number of neurons on the number of activation regions at initialization for networks with $n_0 = 2$. Shown are means and stds for 30 maxout-He normal initializations. 50

3.4 Left: Plotted is #regions$^{1/N}$ for a shallow network with $N = 5$. The multiplicative contribution per unit increases with the input dimension until the trivial upper bound K is reached. Right: Number of regions of 3 layer networks with $n_0 = 2$ depending on K. Shown are means and stds for 30 ReLU-He normal initializations. 50

3.5 Evolution of the linear regions and the decision boundary during training on the MNIST dataset in a slice determined by three random points from different classes. The network had 100 maxout units of rank $K = 2$, and was initialized using maxout-He normal initialization. The right panel is for the 3 layer network. As expected, for the shallow rank-2 network, the number of regions is approximately constant. For deep networks, we observe a moderate increase in the number of regions as training progresses, especially around the training data. However, the number of regions remains far from the theoretical maximum. This is consistent with previous observations for ReLU networks. There is also a slight increase in the number of linear pieces in the decision boundary, and at the end of training the decision boundary clearly separates the three reference points. 51

3.6 Comparison of training on MNIST with different initializations. All networks had 200 units, 10 layers, and maxout networks had rank $K = 5$. Shown are averages and std (barely noticeable) over 30 repetitions. The type of initialization has a significant impact on the training time of maxout networks, with maxout-He, sphere, and many regions giving better results for deep networks and larger maxout rank (more details on this in Section 3.K). 52

3.7 The linear regions of a function $f(x) = \max_j\{\langle w_j, x\rangle + b_j\}$ correspond to the lower vertices of the polytope $P'_f = \mathrm{conv}_j\{(w_j, -b_j)\} \subseteq \mathbb{R}^{n_0+1}$, or, equivalently, the upper vertices of the lifted Newton polytope $P_f = \mathrm{conv}_j\{(w_j, b_j)\} \subseteq \mathbb{R}^{n_0+1}$. The linear regions of f can also be described as the intersection of the normal fan $N_{P'_f}$, consisting of outer normal cones of faces of P'_f, with the affine space $\mathbb{R}^{n_0} \times \{-1\}$. 59

3.8 A layer of maxout units of rank $K \geq 3$ attains several different numbers of linear regions with positive probability over the parameters. For a layer with two rank-3 maxout units, some neighborhoods of parameters give 6 linear regions and others 9, with nonlinear loci given by perturbations of the red-pink and red-darkred lines. 60

3.9 Linear regions of a 3 layer network with 100 units and the maxout rank $K = 2$. The network was initialized with the maxout-He distribution. Activation regions of a maxout network with zero biases are convex cones. Small biases are initialized as the biases sampled from the maxout-He distribution multiplied by 0.1. The majority of linear regions of a network with small biases are cones, and the ones that are not are small and concentrated around zero. 80

3.10 Shown are normal (top) and uniform (bottom) input distributions, as well as the corresponding response distributions for ReLU, maxout of rank $K = 2$, and maxout of rank $K = 5$. The expectation of the square response for maxouts of rank $K > 2$ depends not only on the variance but also on the particular shape of the input distribution. 92

3.11 A few functions represented by a maxout network for different parameter values in a 2D slice of parameter space. For each function, we plot regions of the input space with different gradient values using different colors. 96

3.12 Comparison to the formulas with and without the constants for the number of activation regions and linear pieces in the decision boundary from Theorem 3.9 and Theorem 3.11 respectively. Networks had 100 units and maxout rank $K = 2$. The settings are similar to those in Figure 3.2. 97

3.13 Comparison to the formula from Theorem 3.9 for maxout ranks $K = 3$ and $K = 5$. The networks were initialized with maxout-He normal initialization. We observe the increase in the number of activation regions as the maxout rank increases, and for networks with higher maxout rank deeper networks tend to have less regions than less deep networks with the same rank. 98

3.14 Difference between the effects of depth and number of neurons on the number of activation regions. These plots are additional to Figure 3.3 and have similar settings. ReLU and maxout networks with $K = 2$ have a similar number of linear regions. For maxout rank $K > 2$ deeper networks tend to have less regions than less deep networks with the same rank. For $K = 3$ the gaps between different depths are smaller than for $K = 5$. 99

3.15 Effect of the Steinwart initialization approach on the convergence speed during training on the MNIST dataset for a network with 200 units and 5 layers. Maxout rank was $K = 5$. In this experiment, for various initialization procedures, the addition or omission of a random shift of the non-linear regions of different units led to similar training curves. 100

3.16 Effect of the initialization on the convergence speed during training on the MNIST dataset of networks with 200 units depending on the network depth and the maxout rank. Maxout-He, sphere, and many regions initializations behave similarly, and the improvement in the convergence speed becomes more noticeable for larger network depth and maxout rank. 100

3.17 Evolution of the linear regions and the decision boundary during training on MNIST in a 2D slice determined by three random input points from the dataset. The network had 3 layers, a total of 100 maxout units of rank $K = 2$, and was initialized with the maxout-He initialization. 101

3.18 Change in the number of linear regions and the decision boundary pieces during 100 training epochs given different initializations. Networks had 100 neurons and for maxout networks $K = 2$. Both the number of linear regions and linear pieces of the decision boundary increase during training for all initializations but remain much smaller than the theoretical maximum. The settings were the same as in Figure 3.5. 102

4.1 Expectation of the directional derivative of the input-output map $\mathbb{E}[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2]$ for width-2 fully-connected networks with inputs in $\mathbb{R}^2$. For maxout networks, this expectation depends on the input, while for ReLU networks, it does not. Input points $\boldsymbol{x}$ were generated as a grid of 100×100 points in $[-10^3, 10^3]^2$, and $\boldsymbol{u}$ was a fixed vector sampled from the unit sphere. The expectation was estimated based on $10{,}000$ initializations with weights and biases sampled from $N(0, 1)$. 106

4.2 Expected value and interquartile range of the squared gradients $n_0(\partial\mathcal{N}/\partial W_{i,k',j})^2$ as a function of depth. Weights are sampled from $N(0, c/\text{fan-in})$ in fully-connected networks and $N(0, c/(k^2 \cdot \text{fan-in}))$, where k is the kernel size, in CNNs. Biases are zero, and the maxout rank K is 5. The gradient is stable in wide fully-connected and convolutional networks with $c = 0.55555$ (red line), the value suggested in Section 4.4. The dark and light blue lines represent the bounds from Corollary 4.2, and equal $1/\mathcal{L} = 0.36$ and $1/\mathcal{S} = 12$. The yellow line corresponds to the ReLU-He initialization. We compute the mean and quartiles from 100 network initializations and a fixed input. The same color lines that are close to each other correspond to 3 different unit-norm network inputs. 108

4.3 Illustration of a simple maxout network with two input units, one hidden layer consisting of two maxout units of rank 3, and an affine output layer with a single output unit. 121

4.4 Example of a situation where the training is unsuccessful for a ReLU network because all neurons in the first layer are dead, while a maxout network trains successfully on the same dataset. We plot the breakpoints in the first layer. Notice that they do not move during the training of a ReLU network but change their positions in a maxout network. 122

4.5 The plots show that $|\cos\gamma_{\mathfrak{r}^{(l)},\mathfrak{u}^{(l)}}|$ grows with the network depth and eventually converges to 1 for wide networks and maxout rank $K > 2$. The results were averaged over 1000 parameter initializations, and both weights and biases were sampled from $N(0, c/\text{fan-in})$, $c = 1/\mathbb{E}[(\Xi(N(0,1), K))^2]$, as discussed in Section 4.4. Vectors $\boldsymbol{x}$ and $\boldsymbol{u}$ were sampled from $N(0, I)$. 135

4.6 The plots show that $|\cos\gamma_{\mathfrak{r}^{(l)},\mathfrak{u}^{(l)}}|$ does not converge to 1 for $c < 1/\mathbb{E}[\Xi^2]$ and converges for $c \geq 1/\mathbb{E}[\Xi^2]$. The network had 100 neurons at each layer, and both weights and biases were sampled from $N(0, c/\text{fan-in})$. The results were averaged over 1000 parameter initializations. Vectors $\boldsymbol{x}$ and $\boldsymbol{u}$ were sampled from $N(0, I)$. . . 135

4.7 Second moment of $\Xi(N(0,1), K)$ for different sample sizes K. It increases with K for any $K, 2 \leq K \leq 100$, and $\mathbb{E}[(\Xi(N(0,1), K))^2] > 1$ for $K > 2$. 136

4.8 Shown is the expectation value of the square norm of the directional derivative of the input-output map of a maxout network for a fixed random direction with respect to the weights, plotted as a function of the input. Weights and biases are sampled from $N(0, 1/\text{fan-in})$, and biases are zero. Inputs are standard Gaussian vectors. Vector $\boldsymbol{u}$ is a one-hot vector with 1 at a random position, and it is the same for one setup. We sampled 1000 inputs and 1000 initializations for each input. The left end corresponds to the second moment of the Gaussian distribution, and the right end to the second moment of the largest order statistic. Observe that for wide and deep networks, the mean is closer to the second moment of the largest order statistic. 137

4.9 Comparison of the normalized activation length with the equality in distribution result from Corollary 4.21 and the formula for the mean from Corollary 4.5. Plotted are means and stds estimated with respect to the distribution of parameters / random variables $\Xi_{l,i}(N(0,1), K)$ averaged over $100,000$ initializations, and a numerically evaluated formula for the mean from Corollary 4.5. All layers had 10 neurons. The lines for the mean and areas for the std overlap. Note that there is no std for the formula in the plot. 141

4.10 Expected value and interquartile range of the squared gradients $(\partial\mathcal{N}/\partial W_{i,k',j})^2$ of a fully-connected maxout network during training on the MNIST dataset. Weights are sampled from $N(0, c/\text{fan-in})$, and biases are initialized to zero. The maxout rank K is 5. We compute the mean and quartiles for 4 training runs using one random input that is fixed at the start of the training. The gradient values increase at the beginning of the training and then remain stable during training for all plotted initializations. Note that red and green lines corresponding to the values of $c = 0.55555$ and $c = 1$, respectively, overlap. Similarly, blue and orange lines corresponding to $c = 0.01$ and $c = 0.1$ overlap. Results for $c = 2$ and $c = 10$ are not shown in the plot since their gradients explode and go to NaN after the training starts. 159

4.11 Expected value and interquartile range of the squared gradients $(\partial\mathcal{N}/\partial W_{i,k',j})^2$ of maxout networks during training. Weights are sampled from $N(0, c/\text{fan-in})$, where $c = 0.55555$, and biases are initialized to zero. The maxout rank K is 5. We use SGD with momentum and compute the mean and quartiles for 4 training runs using one random input that is fixed at the start of the training. All other experiment parameters are as described in Section 4.J.1. The gradient values increase at the beginning of the training and then remain stable during training for all datasets. 159

List of tables

1.1 Accuracy on the MNIST dataset of fully-connected networks trained with dropout with a rate of 0.5 and of average predictions of several networks in which half of the weights were masked. All results were averaged over 4 runs. Maxout rank $K = 5$. Networks had 3 layers with 128, 64, and 32 neurons. Maxout networks were initialized using the initialization suggested in Chapter 4, and ReLU networks using He initialization with Gaussian distribution (He et al., 2015). ReLU networks with dropout give results closer to a single model, whereas maxout networks with dropout give results closer to the average of a larger number of models. This observation indicates that maxout units are more effective for obtaining better model averaging using dropout. 19

3.1 Standard deviation of the weight distribution for maxout-He initialization. 48
3.2 Comparison of the activation region results for maxout and ReLU networks. 76

4.1 Recommended values for the constant c for different maxout ranks K based on Section 4.4. 111
4.2 Accuracy on the test set for networks trained using SGD with Nesterov momentum. Observe that maxout networks initialized with the maxout or max-pooling initialization perform significantly better than the ones initialized with other initializations and better or comparably to ReLU networks. 114
4.3 Accuracy on the test set for the networks trained with Adam. Observe that maxout networks initialized with the maxout or max-pooling initialization perform better or comparably to ReLU networks, while maxout networks initialized with ReLU-He converge slower and perform worse. 115
4.9 Constants $\mathcal{L}$ and $\mathcal{S}$ denote the means of the largest and the smallest order statistics in a sample of size K of chi-squared random variables with 1 degree of freedom. Constant $\mathcal{M}$ denotes the second moment of the largest order statistic in a sample of size K of standard Gaussian random variables. See Remark 4.14 for the explanation of how these constants are computed. 131
4.10 Ablation study of the value of c. Reported is accuracy on the test set for maxout networks with maxout rank $K = 5$ trained using SGD with Nesterov momentum. Observe that the optimal value of c is close to $c = 0.55555$ which is suggested in Section 4.4. 154
4.11 Accuracy on the test set for maxout networks with maxout rank $K = 5$ that use batch normalization trained using SGD with Nesterov momentum. Observe that the optimal value of c is close to $c = 0.55555$ which is suggested in Section 4.4. . . 155

4.12 Accuracy on the test set for networks trained using SGD with Nesterov momentum. Fully-connected ReLU networks are 5 times wider than fully-connected maxout networks. Convolutional ReLU networks are 4 times wider than convolutional maxout networks for the MNIST and Fashion MNIST datasets and 3 times wider for the CIFAR-10 and CIFAR-100 datasets. All networks have the same number of layers. 157
4.13 Accuracy on the test set for networks trained using SGD with Nesterov momentum. ReLU networks are 5 times deeper than maxout networks but have the same width. 158
4.14 Accuracy on the MNIST dataset of fully-connected networks trained with dropout with a rate of 0.5 and of average predictions of several networks in which half of the weights were masked. All results were averaged over 4 runs. Maxout rank $K = 5$. Networks had 3 layers with 128, 64, and 32 neurons. Maxout networks were initialized using the initialization suggested in Section 4.4, and ReLU networks using He initialization with Gaussian distribution (He et al., 2015). ReLU networks with dropout give results closer to a single model, whereas maxout networks with dropout give results closer to the average of a larger number of models. This indicates that maxout units are more effective for obtaining better model averaging using dropout. 158

Chapter 1

Introduction

In this thesis, we advance the line of analysis proposed by Hanin and Rolnick (2019a,b), where the focus is on the expected behavior of the deep neural networks. We consider feedforward neural networks with n_0 inputs, L layers of widths $n_1, \dots, n_L$, which implement functions of the form $f = \psi \circ \phi_{L-1} \circ \cdots \circ \phi_1$. The l-th hidden layer implements a function $\phi_l \colon \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ with output coordinates, i.e., units, given by trainable affine functions followed by a fixed real-valued activation function, and $\psi \colon \mathbb{R}^{n_{L-1}} \to \mathbb{R}^{n_L}$ is a linear output layer. Whereas prior works focus on single-argument activation functions, we obtain novel results for the previously little-studied multi-argument maxout activation function (Goodfellow et al., 2013). Concretely, maxout units compute parametric affine functions followed by a fixed multi-argument activation function of the form $(s_1, \dots, s_K) \mapsto \max\{s_1, \dots, s_K\}$ and can be regarded as a natural generalization of ReLUs, which have a single-argument activation function $s \mapsto \max\{0, s\}$. On top of enhancing the practical applicability of maxout networks, these findings add to the theoretical exploration of activation functions beyond the standard choices, such as ReLU. We believe this work can pave the way for the investigation of multi-argument activation functions and complicated neural network architectures.

For any choice of parameters, maxout networks subdivide their input space into linear regions, maximal connected subsets of the input space $\mathbb{R}^{n_0}$ on which the function f computed by a network has a constant gradient. We take the number of linear regions as a measure of the complexity of the function that a network computes and observe that maxout networks can assume widely different numbers of linear regions with positive probability. We then compute an upper bound on the expected number of regions and volume given properties of the parameter distribution, covering the case of zero biases. Further, taking the classification standpoint, we obtain corresponding results for the decision boundary of maxout (and ReLU) networks, along with bounds on the expected distance to the decision boundary. Experiments show that the theoretical bounds capture the general behavior. We present algorithms for enumerating the regions of maxout networks and propose parameter initialization strategies with two types of motivations, one to increase the number of regions, and second, to normalize the variance of the activations similar to Glorot and Bengio (2010) and He et al. (2015), but now for maxout. We observe experimentally that this could improve training in maxout networks.

We continue with the study of the gradients of maxout networks with respect to the parameters and the inputs by analyzing a directional derivative of the input-output map. We observe that the distribution of the input-output Jacobian of maxout networks depends on the network input (in contrast to ReLU networks), which can complicate the stable initialization of maxout networks. Nevertheless, based on bounds on the moments, we derive an initialization that avoids vanishing and exploding gradients in wide networks. Experimentally, we show that, compared to other initializations, the suggested approach leads to better performance for fully connected and convolutional deep networks of standard width trained with SGD or Adam and better or similar performance than ReLU networks. Additionally, we refine the previous upper bounds on the expected number of linear regions. We also derive results for the other measure of the complexity of a network, the expected curve length distortion, observing that it does not grow exponentially with the depth in wide networks. Furthermore, we obtain bounds on the maxout neural tangent kernel (NTK), which describes the evolution of neural networks during the training by gradient descent, suggesting that it might not converge to a constant when both the width and depth are large.

1.1 Maxout networks

1.1.1 Definition

As stated above, we consider a feedforward neural network $\mathcal{N}$ with n_0 inputs and L layers of widths $n_1, \dots, n_L$, which implements a map $\mathcal{N} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ given by a composition of maps

$$\mathcal{N} := \psi \circ \phi_{L-1} \circ \cdots \circ \phi_1.$$

The l-th hidden layer, $l = 1, \dots, L-1$ implements a function $\phi_l \colon \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$. Its output coordinates, i.e., units, are given by trainable affine functions called *pre-activation functions* $\zeta^{(l)}(x) : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ followed by fixed real-valued *activation functions* $\sigma^{(l)} : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$. Pre-activation functions and parametrized by a weight matrix $W^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ and a bias vector $\boldsymbol{b}^{(l)} \in \mathbb{R}^{n_l}$:

$$\zeta^{(l)}(x) := W^{(l)}\boldsymbol{x} + \boldsymbol{b}^{(l)},$$

where $\boldsymbol{x} \in \mathbb{R}^{n_{l-1}}$ is a layer input, with $\boldsymbol{x} \in \mathbb{R}^{n_0}$ being a network input. The output layer is given by a linear function $\psi \colon \mathbb{R}^{n_{L-1}} \to \mathbb{R}^{n_L}$. We denote the total number of hidden units by $N = n_1 + \cdots + n_{L-1}$. The collection of all trainable parameters is denoted by $\boldsymbol{\Theta} = \{W, \boldsymbol{b}\}$.

In this thesis, we are interested in the functions parametrized by artificial feedforward neural networks with maxout units, which can be regarded as a natural generalization of ReLUs.

Definition 1.1 (ReLU activation)**.** Given an input $\boldsymbol{x} \in \mathbb{R}^n$, where $\boldsymbol{x}$ may be an input or a hidden layer state, a rectified linear unit (ReLU) implements a function

$$\mathbb{R}^n \to \mathbb{R}; \quad \boldsymbol{x} \mapsto \max\{\langle W, \boldsymbol{x}\rangle + b, 0\},$$

where $W \in \mathbb{R}^n$ and $b \in \mathbb{R}$ are trainable weights and biases.

Definition 1.2 (Maxout activation)**.** A rank-K maxout unit, introduced by Goodfellow et al. (2013), computes the maximum of K real-valued parametric affine functions. Concretely, a rank-K maxout unit with n inputs implements a function

$$\mathbb{R}^n \to \mathbb{R}; \quad \boldsymbol{x} \mapsto \max_{k \in [K]}\{\langle W_k, \boldsymbol{x}\rangle + b_k\},$$

where $W_k \in \mathbb{R}^n$ and $b_k \in \mathbb{R}$, $k \in [K] := \{1, \ldots, K\}$, are trainable weights and biases. The K arguments of the maximum are called the *pre-activation features* of the maxout unit.

A rank-K maxout unit can be regarded as a composition of an affine map with K outputs and a maximum gate. A layer corresponds to the parallel computation of several such units. For instance, a layer with n inputs and m maxout units computes functions of the form

$$\mathbb{R}^n \to \mathbb{R}^m; \quad \boldsymbol{x} \mapsto \begin{bmatrix} \max_{k \in [K]}\{\langle W^{(1)}_{1,k}, \boldsymbol{x}\rangle + \mathbf{b}^{(1)}_{1,k}\} \\ \vdots \\ \max_{k \in [K]}\{\langle W^{(1)}_{m,k}, \boldsymbol{x}\rangle + \mathbf{b}^{(1)}_{m,k}\} \end{bmatrix},$$

where now $W^{(1)}_{i,k}$ and $\mathbf{b}^{(1)}_{i,k}$ are the weights and biases of the kth pre-activation feature of the ith maxout unit in the first layer. The situation is illustrated in Figure 1.1 for the case of a network with two inputs, one layer with two maxout units of rank three, and one output layer with a single output unit.

1.1.2 Significance of maxout network research

Below we outline multiple reasons that motivate the study of maxout networks.

Maxout unit activation functions are multi-argument A maxout unit may be considered a multi-argument generalization of a ReLU, which computes the maximum of a real-valued affine function and zero. Since understanding the behavior of multi-argument activation functions is interesting from the theoretical point of view and can facilitate the design of novel activation functions, maxout unit activation function analysis can serve as a platform for such investigation.

Potential extension to complicated architectures Some of the modern architectures use multi-argument activation functions. For instance, graph neural networks (GNNs) employ a maximum

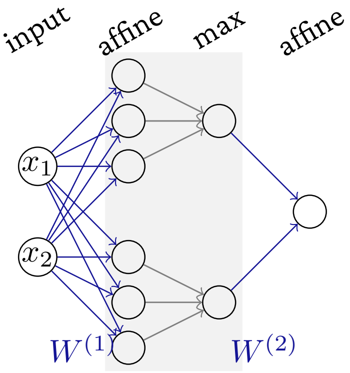


Figure 1.1: Illustration of a simple maxout network with two input units, one hidden layer consisting of two maxout units of rank 3, and an affine output layer with a single output unit.

aggregation function (Hamilton, 2020), essentially a maxout activation function. Therefore, we believe that developing the theory and implementation aspects of maxout networks can serve as an interesting platform for architecture design.

Maxout units are frequently used Even though maxout networks are used less than other popular activation function choices, such as ReLU activations, they are still frequently used. For instance, there were $2,010$ references mentioning maxout networks since 2022 on Google Scholar as of 11.06.2023 (Google Scholar, 2023). Moreover, Goodfellow et al. (2013) demonstrated that maxout networks could perform better than ReLU networks under similar circumstances. Additionally, maxout networks have been shown to be useful for combating catastrophic forgetting in neural networks (Goodfellow et al., 2015). On the other hand, Castaneda et al. (2019) evaluated the performance of maxout networks in a big data setting and observed that increasing the width of ReLU networks is more effective in improving performance than replacing ReLUs with maxout units and that ReLU networks converge faster than maxout networks. We observe that maxout networks, in general, and proper initialization strategies for maxout networks, in particular, have not been studied in the same level of detail as for ReLU networks and that this might resolve some of the problems encountered in previous maxout network applications.

Maxout networks solve the dying neurons problem in ReLU networks One of the motivations for introducing maxout networks in Goodfellow et al. (2013) was to provide an alternative to ReLU networks with the potential to improve issues with dying neurons. The dying neurons problem in ReLU networks refers to ReLU neurons being inactive on a dataset and never getting updated during optimization. It can lead to a situation when the training cannot commence if all neurons in one layer are dead. This problem never occurs in maxout networks since maxout units are always active. Furthermore, the absence of the zeroed paths, as in ReLU networks, has unclear effects on the function complexity.

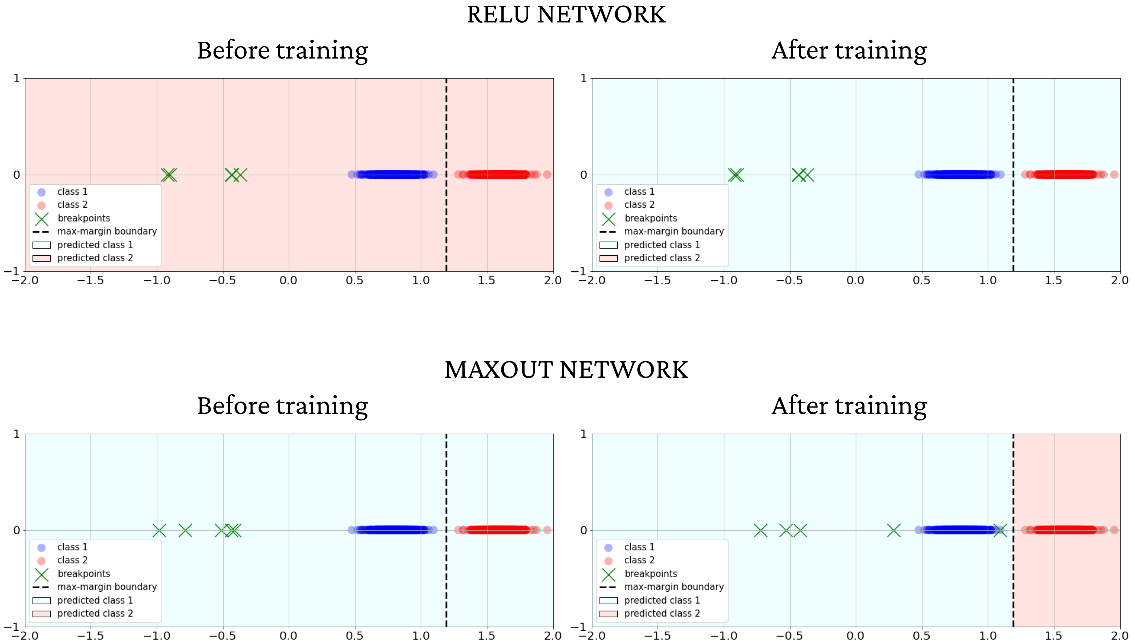


Figure 1.2: Example of a situation where the training is unsuccessful for a ReLU network because all neurons in the first layer are dead, while a maxout network trains successfully on the same dataset. We indicate the positions of the breakpoints in the first layer with the green $\times$ symbol. Notice that the breakpoints do not move during the training of a ReLU network but change their positions in a maxout network.

We design a simple experiment to illustrate the issue of dying neurons. We consider a binary classification task on a dataset sampled from a Gaussian mixture of two univariate Gaussians $N(0.8, 0.1)$ and $N(1.6, 0.1)$. We sample 600 training, 200 validation, and 200 test points. We construct maxout and ReLU networks with 5 layers and 5 units per layer. Maxout units rank equals 2. We set weights and biases in the first layer so that the breakpoints are left of the data. For ReLU, we also ensure that the weights are negative to guarantee that the neurons in the first layer are inactive. Hence, all the units in the first layer of the ReLU network are dead. Then we train the network for 20 epochs using SGD with a learning rate of 0.5 and batch size of 32. For the ReLU networks, since all units in the first layer are dead, the training is unsuccessful, and the accuracy on the test set is 50%. In contrast, for the maxout network, the test set accuracy is 100%. Figure 1.2 illustrates this example.

Maxout networks are more compatible with dropout than ReLU networks Maxout networks were proposed by Goodfellow et al. (2013) as an alternative to ReLU networks with the potential to attain better model averaging when used with dropout (Hinton et al., 2012). The original paper (Goodfellow et al., 2013) conducted experiments comparing maxout and tanh. In Table 1.1, we show the results of an experiment demonstrating that in terms of allowing for a better approximation

Table 1.1: Accuracy on the MNIST dataset of fully-connected networks trained with dropout with a rate of 0.5 and of average predictions of several networks in which half of the weights were masked. All results were averaged over 4 runs. Maxout rank $K = 5$. Networks had 3 layers with $128, 64$, and 32 neurons. Maxout networks were initialized using the initialization suggested in Chapter 4, and ReLU networks using He initialization with Gaussian distribution (He et al., 2015). ReLU networks with dropout give results closer to a single model, whereas maxout networks with dropout give results closer to the average of a larger number of models. This observation indicates that maxout units are more effective for obtaining better model averaging using dropout.

	DROPOUT	1 MODEL	AVERAGE OF 2 MODELS	AVERAGE OF 3 MODELS	AVERAGE OF 4 MODELS
ReLU	$97.04^{\pm 0.14}$	$97.09^{\pm 0.17}$	$97.73^{\pm 0.08}$	$97.87^{\pm 0.04}$	$97.94^{\pm 0.08}$
Maxout	$98.37^{\pm 0.09}$	$97.66^{\pm 0.04}$	$98.03^{\pm 0.05}$	$98.15^{\pm 0.08}$	$98.19^{\pm 0.06}$

of model averaging based on dropout, maxout networks compare favorably against ReLU. This observation indicates that maxout units can indeed be more suitable for training with dropout when properly initialized. We point out that several contemporary architectures often rely on dropout, such as transformers (Vaswani et al., 2017).

1.2 Contributions and thesis outline

We start by providing the background for the results presented in this thesis in Chapter 2. We review the historical development of expressivity and network complexity research; stable initialization of neural networks; the connection between neural networks with piece-wise linear activations and tropical geometry; and conclude with a brief overview of the neural tangent kernel.

We proceed with Chapter 3, based on Tseran and Montúfar (2021). In this part of the thesis, we analyze the expected complexity of maxout networks. For any choice of parameters, maxout networks subdivide their input space into linear regions, maximal connected subsets of the input space $\mathbb{R}^{n_0}$ on which the function f computed by a network has a constant gradient. We formalize the notion of linear regions using the concept of activation regions, defined as subsets of the input space where different pre-activation features attain maximum. Hence, to characterize the network complexity, we are concerned with the expected number of activation regions and their volume given probability distributions of parameters and corresponding properties for the decision boundaries in classification tasks. We obtain the following results for the complexity of maxout networks.

- There are widely different numbers of linear regions that are attained with positive probability over the parameters (Theorem 3.7). There is a non-trivial lower bound on the number of linear regions that holds for almost every choice of the parameters (Theorem 3.8). These results advance the maximum complexity analysis of Montúfar et al. (2022) from the perspective of generic parameters.

- For common parameter distributions, the expected number of activation regions is polynomial in the number of units (Theorem 3.9). Moreover, the expected volume of activation regions of different dimensions is polynomial in the number of units (Theorem 3.10). These results correspond to maxout versions of results from Hanin and Rolnick (2019b) and Hanin and Rolnick (2019a).

- For multi-class classifiers, we obtain an upper bound on the expected number of linear pieces (Theorem 3.11) and the expected volume (Theorem 3.12) of the decision boundary, along with a lower bound on the expected distance between input points and decision boundaries (Corollary 3.13).

- We provide an algorithm and implementation for counting the number of linear regions of maxout networks (Algorithm 3.1).

- We present parameter initialization procedures for maxout networks maximizing the number of regions or normalizing the mean activations across layers (similar to Glorot and Bengio 2010; He et al. 2015), and observe experimentally that these can lead to faster convergence of training.

We continue with Chapter 4 based on Tseran and Montúfar (2023), in which we study the gradients of maxout networks. The analysis is based on the input-output Jacobian. We discover that, in contrast to ReLU networks, when initialized with a zero-mean Gaussian distribution, the distribution of the input-output Jacobian of a maxout network depends on the network input, which may lead to unstable gradients and training difficulties. Nevertheless, we compute bounds on the moments of the gradients of maxout networks depending on the parameter distribution and the network architecture and derive a rigorous parameter initialization strategy for wide networks and several implications for stability and expressivity. Our results can be summarized as follows.

- For expected gradients, we derive stochastic order bounds for the directional derivative of the input-output map of a deep fully-connected maxout network (Theorem 4.1) as well as bounds for the moments (Corollary 4.2). Additionally, we derive equality in distribution for the directional derivatives (Theorem 4.3), based on which we also discuss the moments (Remark 4.4) in wide networks. We further derive the moments of the activation length of a fully-connected maxout network (Corollary 4.5).

- We rigorously derive parameter initialization guidelines for wide maxout networks preventing vanishing and exploding gradients and formulate architecture recommendations. We experimentally demonstrate that they make it possible to train standard-width deep fully-connected and convolutional maxout networks using simple procedures (such as SGD with momentum and Adam), yielding higher accuracy than other initializations or ReLU networks on image classification tasks.

- We derive several implications refining previous bounds on the expected number of linear regions (Corollary 4.6), and new results on length distortion (Corollary 4.7) and the NTK (Corollary 4.9).

1.3 Overview of the computational results

As a result of the research in this thesis, we develop multiple experiments and helpful components. The code for them is public. The implementations relevant for Chapter 3 are available at `https://github.com/hanna-tseran/maxout_complexity` and for Chapter 4, in `https://github.com/hanna-tseran/maxout_expected_gradients`.

Specifically, `https://github.com/hanna-tseran/maxout_complexity` contains the following routines implemented in Python using PyTorch library (Paszke et al., 2019) that can be used for maxout and ReLU networks, where applicable:

- Implementations of the maxout fully-connected networks;
- Approximate and exact computation of the number of linear regions;
- Exact computation of the number of linear pieces in the decision boundary;
- Computation of the formulas for the upper bounds on the expected number of linear regions and pieces in the decision boundary for maxout networks from Chapter 3;
- Plots of the linear regions and decision boundary in a 2D slice of the input space determined by three data points;
- Computation of the routines above during the network training on the MNIST dataset (LeCun and Cortes, 2010);
- Implementations of the various maxout network parameter initialization procedures described in Chapter 3.

The second repository, `https://github.com/hanna-tseran/maxout_expected_gradients`, contains the following components in Python implemented using Tensorflow library (Martín Abadi et al., 2015) that can, similar to the tools above, be used for maxout and ReLU networks, where applicable:

- Implementations of the maxout fully-connected and convolutional neural networks;
- Implementations of the maxout and max-pooling initializations;
- Experiments training networks on MNIST (LeCun and Cortes, 2010), Iris (Fisher, 1936), Fashion MNIST (Xiao et al., 2017), SVHN (Netzer et al., 2011), CIFAR-10 and CIFAR-100 datasets

(Krizhevsky et al., 2009) using SGD with Nesterov momentum and Adam optimizer Kingma and Ba (2015);

- Expectation of the directional derivative of the input-output map for width-2 fully-connected networks with inputs in $\mathbb{R}^2$;
- Estimation of the mean and interquartile range of the squared gradients with respect to the network weights;
- Estimation of the activation length and its comparison to the formula from Chapter 4;
- Estimation of the value of the cosine appearing in the statement of Theorem 4.3 in Chapter 4 depending on the network initialization, and the network width and depth;
- Plots of the square norm of the directional derivative of the input-output map of a maxout network for a fixed random direction with respect to the weights, plotted as a function of the input;
- Plots for the second moment of the largest order statistic in a sample of K standard Gaussians $\Xi(N(0,1),K)$ for different sample sizes K.

1.4 Discussion

Conclusion

In this thesis, we have contributed to the understanding of the expected behavior of deep maxout neural networks and extended the previous line of work investigating the expected behavior of networks with single-argument activation functions (Hanin and Rolnick, 2019b,a, 2018), particularly ReLU networks, to the multi-argument case. For maxout networks, we observe that their practical complexity is far from the theoretical maximum, derive an initialization procedure provably improving the optimization with gradient descent, and obtain several implications for the expressivity and neural tangent kernel. We support our findings with multiple experiments. We believe this line of work can advance the research of neural network activation functions beyond the common choices and serve as a platform for the analysis of more complicated neural network architectures, such as graph neural networks and transformers.

Limitations

- In our theory, we have considered only fully connected networks. However, our experiments indicate that the subset of our results on gradient moments and network initialization also holds for CNNs, though a theoretical analysis of CNNs is yet to be conducted.

- By design, the results on the expected complexity of maxout networks focus on parameter distributions that have a density.

- Even though our proposed initialization of a maxout network is optimal in the sense of the criteria specified in Chapter 4, our results are applicable only when the weights are sampled from $N(0, c/\text{fan-in})$ for some c.

Further directions

Multiple exciting further directions can build on the work in this thesis. They include the following topics:

- Extension of the results to architectures that are more involved than feedforward fully-connected maxout networks and employ multi-argument functions. One example is graph neural networks, which use multi-argument aggregation functions, including the maximum aggregation function;

- Exploration of the benefits of maxout units to architectures that use dropout, such as transformers;

- Extension of the results on the expected number of linear regions to a fine-grained description of the distribution of activation regions over the input space depending on the parameter distribution;

- Analysis of the relationship between the expected complexity of the networks at initialization and the speed of convergence and implicit biases in gradient descent;

- Extension of the presented results on the expected complexity to specific types of parameter distributions, including those that do not have a density or those one might obtain after training;

- Investigation of the effects of the initialization strategies stabilizing the initial gradients during later stages of training.

1.5 Other projects developed during the Ph.D. studies

In addition to the topics discussed in this thesis, during the Ph.D., I have worked on analyzing the loss landscape of mildly overparametrized ReLU networks and designing a memory-augmented efficient transformer-based model. These projects are absent in the main discussion since they are not directly related to the main subject of the thesis, which is the expected behavior of maxout networks. However, these studies address related questions from complementary angles.

The project on the loss landscape focuses on the behavior of mildly-overparametrized neural networks. It further advances the line of work on the loss landscape analysis to practical scenarios, while the previous studies often focused on the overparametrized case or had to introduce assumptions that might not hold in practice. Hence, it shares goals with the investigation of the expected network behavior. Correspondingly, the Turing Machine Transformer project studies how to improve a transformer architecture by making the attention mechanism more efficient. Similar to the maxout network analysis, such research helps understand and improve neural network architectures.

Below I outline the summary of these two projects.

Mildly overparametrized ReLU networks have a favorable loss landscape[1] In Karhadkar et al. (2023), we consider the loss landscape of mildly overparametrized ReLU networks. In general, the optimization landscape of neural networks has been a topic of enormous interest over the years. A particularly puzzling question is why bad local minima do not seem to be a problem for training. We study the loss landscape of two-layer mildly overparametrized ReLU neural networks on a generic finite input dataset for the squared error loss. Our approach involves bounding the dimension of the sets of local and global minima using the rank of the Jacobian of the parameterization map.

In contrast to previous related works, we can formulate our results for ReLU activations rather than LeakyReLU or other smooth activation functions. Unlike, for instance, Soudry and Carmon (2016), we do not assume dropout noise on the network outputs; contrary to Safran and Shamir (2018), we do not assume any particular distribution on our datasets. Using results on random binary matrices, we show that most activation patterns correspond to parameter regions with no bad differentiable local minima. Furthermore, for one-dimensional input data, we show that most activation regions realizable by the network contain a high dimensional set of global minima and no bad local minima. We experimentally confirm these results by finding a phase transition from most regions having full rank Jacobian to many regions having deficient rank depending on the amount of overparametrization.

Turing Machine Transformer for unbounded sequence processing[2] Multi-head self-attention is crucial to the transformer architecture (Vaswani et al., 2017). However, the amount of computation it performs quadratically depends on the input length, and besides, standard transformers can work only with inputs of fixed length. Furthermore, theoretical limitations make it challenging for these models to represent hierarchical structures (Hahn, 2020). These issues make applying

---

[1] Karhadkar et al. (2023), under review. This is joint work with Kedar Karhadkar, Michael Murray, and Guido Montúfar. My main contribution is to experiments. Specifically, I conducted experiments on the estimation of the percentage of randomly sampled activation regions containing a global minimum of the loss, participated in the experiments estimating the probability of the Jacobian being of full rank, and helped with the preprint preparation.

[2] Under review. This is joint work with Cheng Wang. The project was carried out during the internship at Amazon at Berlin Machine Learning team between 15.11.2022 – 15.03.2023. I contributed to the model design, implemented the experiments, and wrote the paper jointly with Cheng Wang.

transformer-based models to applications such as long chatbot conversations or document embedding problematic.

In this work, following the idea of augmenting transformers with external memory, introduced to allow for extended contexts, we develop a novel approach based on the Neural Turing Machine (Graves et al., 2014). Previously, Ebrahimi et al. (2020) observed that a transformer might work as a pushdown automaton (PDA) which uses a stack as its memory. Since the Turing machine is provably more powerful than the PDA in terms of the set of languages it can recognize, and an extension based on the Neural Turing Machine is Turing complete, we expected it to be more efficient than similar memory-augmented transformers. Consequently, we present a modification of the transformer architecture based on the Neural Turing Machine. It reduces the space and time complexity of self-attention, allows processing sequences of any length, and is better rooted in theory than similar models. We perform a series of experiments on several datasets comparing the memory capacity of the introduced model to a vanilla transformer.

Chapter 2

Background

2.1 Network expressivity

The *expressive power* or *capacity* of a given network corresponds to the richness of the class of functions it can represent. A substantial number of works has considered this question, for example, Pascanu et al. (2013); Montufar et al. (2014); Bianchini and Scarselli (2014); Poole et al. (2016); Raghu et al. (2017); Serra et al. (2018); Kileel et al. (2019); Xiong et al. (2020); Bodnar et al. (2021). Particularly upper bounds on the expressivity measures have been studied a lot. The common conclusion for different complexity measures is that their upper bounds grow exponentially with the network depth, which has been proposed as an explanation of the effectiveness of deeper networks.

However, a question one might be particularly interested in is the expected complexity of neural networks that one can observe in practice and how far it is from the maximum theoretically possible complexity. In this thesis, we continue the line of work started by Hanin and Rolnick (2019a,b), who analyzed the expected behavior of ReLU networks. Advancing this course of study, we investigate the expected behavior of maxout networks, including the question of the expected complexity at initialization time, where the expectation is with respect to the distribution of the network parameters.

2.1.1 Connection to approximation

In this section, we expand on the connection between the analysis of neural network expressivity and approximation properties following, to a certain extent, the discussion by Telgarsky (2021). Consider the following setting. Given a dataset (X, y), $X \in \mathbb{R}^{n\times n_0}, y \in \mathbb{R}^n$, a model computing function $f(x)$, and a loss function $\ell : \mathbb{R}^{n\times n_0} \times \mathbb{R}^n \to \mathbb{R}$, we suffer empirical risk $\hat{R}(f) = 1/n \sum_{i=1}^{n} \ell(f(x_i), y_i)$ on a training set. For the test data, consider population risk $R(f) = \mathbb{E}[\ell(f(x), y)]$. Additionally, consider an optimization algorithm choice $\hat{f} \in \mathcal{F}$, where $\mathcal{F}$ is a hypothesis class, and $f^* \in \mathcal{F}$ is the solution with the minimum risk from the hypothesis class. Following a classical point of view, the population risk for $\hat{f} \in \mathcal{F}$ can be decomposed into the following terms: generalization, concentration/generalization, optimization, and approximation errors. Specifically,

$$R(\hat{f}) = \underbrace{R(\hat{f}) - \hat{R}(\hat{f})}_{\text{generalization error}} + \underbrace{\hat{R}(\hat{f}) - \hat{R}(f^*)}_{\text{optimization error}} + \underbrace{\hat{R}(f^*) - R(f^*)}_{\text{concentration/generalization error}} + \underbrace{R(f^*)}_{\text{approximation error}} .$$

Out of these terms, approximation error is the one that is the most related to expressivity. The study of approximation errors of neural networks and expressivity are closely connected, and sometimes the terms are used interchangeably (Gühring et al., 2020). However, in contrast to the approximation error analysis, expressivity usually refers to describing the functions that neural networks can represent exactly, and this is the expressivity definition we will be using in this thesis.

The classical results on approximation in neural networks are the universality theorems due to Hornik et al. (1989); Cybenko (1989); Funahashi (1989). These works discuss similar statements, and, for instance, omitting the details, Hornik et al. (1989) proved that there is a single hidden layer feedforward network that approximates any measurable function to any desired degree of accuracy on some compact set K.

However, in the results above, as approximation accuracy tends to zero, the network width tends to infinity, which does not describe the practical scenarios. Furthermore, in practice, it is observed that deep networks perform better than their shallow counterparts. Hence, the effects of network depth on the complexity of functions computed by networks and approximation errors have received special attention. Below we expand on the type of results termed "depth separation results" that are particularly relevant to the discussion of network expressivity.

Depth separation results

The line of work concentrated on studying if there are functions that cannot be approximated by reasonably wide shallow networks but can be arbitrarily well approximated by a finitely wide deeper network is often referred to as "depth separation results" in the literature. Further, we review several of the results of this type.

One of the earliest results of this type considers sum-product networks. A sum-product network is a network composed of units that either compute the product of their inputs or a weighted sum of their inputs (where weights are strictly positive) Delalleau and Bengio (2011). Consider $\mathcal{F}$ – a family of functions computed by a sum-product network (deep for $i \geq 2$) composed of alternating product and sum layers. Denote with $n = 4^i$ the input size. We set the network depth to $2i$. Delalleau and Bengio (2011) have shown that any shallow sum-product network computing $f \in \mathcal{F}$ must have at least $2^{\sqrt{n}-1}$ hidden units. Note that a multi-layer sum-product network is a polynomial, and this separation result does not imply a ReLU separation.

Telgarsky (2015, 2016) discovered the earliest proof showing that a deep network can not be approximated by a reasonably-sized wide shallow network. Consider the following function Δ:

$$\Delta(x) = \begin{cases} 2x & x \in [0, 1/2), \\ 2 - 2x & x \in [1/2, 1), \\ 0 & \text{otherwise.} \end{cases}$$

Telgarsky (2015, 2016) demonstrate that for any $L \geq 2$, $f = \Delta^{L^2+2}$ is a ReLU network with $3L^2 + 6$ nodes and $4L^2 + 4$ layers, but any ReLU network g with $\leq 2^L$ nodes and $\leq L$ layers can not approximate it:

$$\int_{[0,1]} |f(x) - g(x)| dx \geq \frac{1}{32}.$$

The proof idea behind this statement is to upper bound the number of linear regions in a ReLU network. Since the number of linear regions grows polynomially in width but exponentially in depth (Montufar et al., 2014), we can use the region counting argument to show that reasonably sized shallow networks cannot approximate the goal function. Notice that this result is naturally connected to the analysis of the expressivity of neural networks in terms of the number of linear regions.

2.1.2 Connection to memorization

The results mentioned above considered continuous function approximation. Another related field of study is the investigation of the memorization capabilities of neural networks. Memorization refers to the phenomenon that a large enough network can memorize an entire dataset, meaning that if given N data points, the network can learn the function they represent. These works address whether deeper networks require fewer parameters to memorize the training data, which is related to understanding the effects of the network depth on its expressivity. Below we summarize the results from Baldi and Vershynin (2019) of this type.

Baldi and Vershynin (2019) considered threshold functions as activations. They define the capacity of a neural architecture $A(n_1, n_2, \ldots, n_L)$ as the binary logarithm of the number of different functions $f : H^{n_1} \to H^{n_L}$ it can compute. They show that for a neural architecture $A(n_1, \ldots, n_L)$ with $L \geq 2$ layers, assuming that the number of nodes in each layer satisfies $n_j > 18 \log_2(Ln_k)$ for any pair j, k such that $1 \leq j < k \leq L$, its capacity is

$$C(n_1, \ldots, n_L) \asymp \min(n_1, \ldots, n_L) \sum_{k=1}^{L-1} n_k n_{k+1}.$$

Here the notation $a \asymp b$ means that there exist two positive absolute constants c_1, c_2, such that $c_1 b \leq a \leq c_2 b$. Baldi and Vershynin (2019) conclude that while shallow networks compute more functions than deep networks, the functions computed by deep networks are more regular, in terms of avoiding overfitting.

2.1.3 Complexity measures

Previous works have introduced various measures of network expressivity. For instance, the number of linear regions, distortion of the length of the curve as it passes through the network, and topological complexity of the input space. In this thesis, we consider the number of regions of maxout networks in Chapters 3 and 4, and the curve length distortion for maxout networks in Chapter 4. Below we expand on several of the measures in more detail.

Number of linear regions

Consider the networks with piece-wise linear activation functions, such as ReLU and maxout unit activation functions. Then the functions computed by such networks are piece-wise linear, and we can define the notion of a linear region. Specifically, let $f: \mathbb{R}^{n_0} \to \mathbb{R}$ be a piecewise linear function. A *linear region* of f is a maximal connected subset of $\mathbb{R}^{n_0}$ on which f has a constant gradient. See Figure 2.1 for the illustration of linear regions of a network.

For neural networks with piece-wise linear activation functions, the number of activation regions serves as a complexity measure and summary description, which has proven useful in the investigation of approximation errors, Lipschitz constants, speed of convergence, implicit biases of parameter optimization, and robustness against adversarial attacks. In particular, Pascanu et al. (2014); Montufar et al. (2014); Telgarsky (2015, 2016) obtained depth separation results showing that deep networks can represent functions with many more linear regions than any of the functions that shallow networks with the same number of units or parameters can represent. This implies that certain tasks require enormous shallow networks but can be solved with small deep networks. The geometry of the boundaries between linear regions has been used to study function-preserving transformations of the network weights (Phuong and Lampert, 2019; Serra et al., 2020) and robustness (Croce et al., 2019; Lee et al., 2019a). Steinwart (2019) demonstrated empirically that the distribution of regions at initialization could be related to the speed of convergence of gradient descent, and Williams et al. (2019); Jin and Montúfar (2023) related the density of breakpoints at initialization to the curvature of the solutions after training. The properties of linear regions concerning training have been recently studied by Zhang and Wu (2020), and the number of linear regions of a shallow univariate ReLU network after optimization has been analyzed in Safran et al. (2022). The number of regions has also been utilized to study the eigenvalues of the neural tangent kernel and Lipschitz constants (Nguyen et al., 2020).

Especially the maximum number of linear regions has been studied intensively. In particular, Montúfar (2017); Serra et al. (2018) improved the upper bounds from Montufar et al. (2014) by accounting for output dimension bottlenecks across layers. Hinz and Van de Geer (2019) introduced a histogram framework for a fine-grained analysis of such dimensions in ReLU networks. Based on this, Xie et al. (2020); Hinz (2021) obtained still tighter upper bounds for ReLU networks. Sharp upper bounds on the number of linear regions of fully-connected maxout networks were obtained in

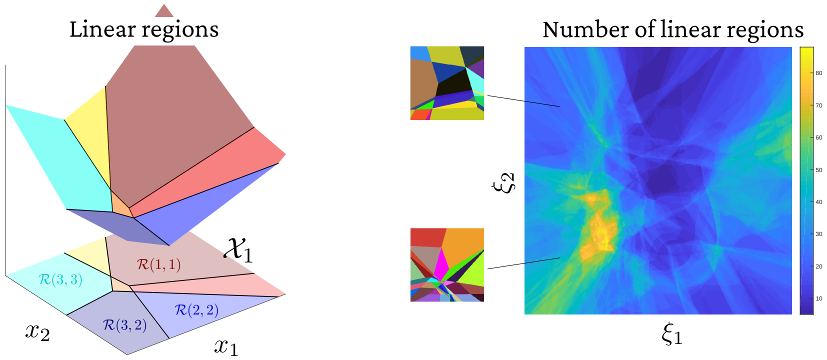


Figure 2.1: Left: Shown is a piecewise linear function $\mathbb{R}^2 \to \mathbb{R}$ represented by a network with a layer of two rank-3 maxout units for a choice of the parameters. The input space is subdivided into linear regions. Right: Shown is the number of linear regions of a 3-layer maxout network over a portion of the input space as a function of a 2D affine subspace of parameter values $\theta(\xi_1, \xi_2)$. Shown are also two examples of the input-space subdivisions of functions represented by the network for different parameter values. As the figure illustrates, the function taking parameters to the number of regions is rather intricate. In this work, we characterize values attained with positive probability and upper bound the expected value given a parameter distribution.

Montúfar et al. (2022). The maximum number of regions has been studied not only for fully connected networks but also convolutional neural networks (Xiong et al., 2020), graph neural networks (GNNs), and message passing simplicial networks (MPSN) (Bodnar et al., 2021).

Expected number of linear regions Although the maximum possible number of regions gives useful complexity bounds and insights into different architectures, in practice, one may be more interested in the expected behavior for typical choices of the parameters. The first results on the expected number of regions were obtained by Hanin and Rolnick (2019a,b) for the case of ReLU networks or single-argument piecewise linear activations. They show that if one has a distribution of parameters such that the conditional densities of bias values are bounded, and the expected gradients of activation values are bounded, then the expected number of linear regions can be much smaller than the maximum theoretically possible number. Specifically, while upper bounds on the maximum number of linear regions grow exponentially with the network depth, upper bounds on their expected number do not depend on the network depth and grow polynomially with the total number of neurons in the network. Moreover, they obtain bounds for the expected number and volume of lower dimensional linear pieces of the represented functions. These results do not directly apply to the case of maxout units, and in Chapter 3, we adapt the proofs to obtain corresponding results for maxout networks, refining the bounds further in Chapter 4.

Curve distortion

Another useful measure of the complexity of the function computed by a neural network is the distortion of the length of an input curve as it passes through the network. Poole et al. (2016) studied the propagation of Riemannian curvature through wide neural networks using a mean-field approach, and later, a related notion of “trajectory length” was considered by Raghu et al. (2017). It was demonstrated that these measures can grow exponentially with the network depth, which was linked to the ability of deep networks to “disentangle” complex representations. Based on these notions, Murray et al. (2022) studies how to avoid rapid convergence of pairwise input correlations, vanishing, and exploding gradients.

Expected curve distortion In contrast to the results above, Hanin et al. (2021) proved that for a ReLU network with He initialization, the length of the curve does not grow with depth and even shrinks slightly. Similar results are established for maxout networks in Chapter 4.

Topological complexity

Another interesting network complexity measure is the topological complexity introduced in Bianchini and Scarselli (2014). Let $f_{\mathcal{N}} : \mathbb{R}^n \to \mathbb{R}$ be the function implemented by a feedforward neural network $\mathcal{N}$, with n inputs and a single output. Complexity of the function $f_{\mathcal{N}}$ is then measured by the topological complexity of the set $S_{\mathcal{N}} = \{x \in \mathbb{R}^n | f_{\mathcal{N}}(x) \geq 0\}$.

Bianchini and Scarselli (2014) prove the following statements. First, for network architectures with a single hidden layer, the sum of the Betti numbers, $B(S_{\mathcal{N}})$, grows at most polynomially with respect to the number of the hidden units h, i.e., $B(S_{\mathcal{N}}) \in O(h^n)$, where n is the input dimension. Second, for deep networks, $B(S_{\mathcal{N}})$ can grow exponentially in the number of the hidden units, i.e., $B(S_{\mathcal{N}}) \in \Omega(2^h)$.

2.1.4 Complexity of maxout networks

Most previous works investigating neural network expressivity have focused on ReLUs or single-argument activation functions.

Firstly, consider the number of linear regions as the measure of the network complexity of ReLU networks. The linear regions of individual layers are described by hyperplane arrangements, which have been investigated since the 19th century (Steiner, 1826; Buck, 1943; Zaslavsky, 1975). Hence, the main challenge in these works is the description of compositions of several layers. In contrast, the linear regions of maxout layers are described by complex arrangements that are not so well understood yet. We describe this problem in more detail in Section 2.3.3.

Consequently, the study of maxout networks poses significant challenges already at the level of individual layers and, in fact, single units. For maxout networks, the maximum possible number of regions has been studied by Pascanu et al. (2014); Montufar et al. (2014); Serra et al. (2018). Recently, Montúfar et al. (2022) obtained counting formulas and sharp (asymptotic) upper bounds for the number of regions of shallow (deep) maxout networks. However, they focused on the maximum possible value rather than the generic behavior we investigate in this thesis.

Secondly, there are no works we know of investigating the complexity of maxout networks from the angle of curve distortion, as we do in Chapter 4.

2.2 Parameter initialization in neural networks

Neural network initialization can have different purposes besides ensuring the training is possible. Both in Chapter 3 and 4, our main goal concerning initialization is to obtain a stable approach to the parameter initialization in maxout networks. We consider a network initialization stable if it avoids vanishing and exploding gradients. The vanishing and exploding gradient problem has been known since the work of Hochreiter (1991). It refers to gradient updates approaching zero or becoming extremely large. Exploding or vanishing gradients can prevent the optimization from starting, or, if the training can begin, makes choosing an appropriate learning rate harder and slows training (Sun, 2019).

Common approaches to address this difficulty besides using an appropriate initialization include the choice of specific architectures, e.g., LSTMs (Hochreiter, 1991) or ResNets (He et al., 2016), and normalization methods such as batch normalization (Ioffe and Szegedy, 2015) or explicit control of the gradient magnitude with gradient clipping (Pascanu et al., 2013). These techniques can be combined with initialization to achieve even better performance. For instance, in Section 4.J.3, we observe that the initialization strategy proposed for maxout networks in Chapter 4 is still beneficial when training with batch normalization.

In this thesis, we focus on approaches based on parameter initialization that control the activation length and parameter gradients (LeCun et al., 2012; Glorot and Bengio, 2010; He et al., 2015; Gurbuzbalaban and Hu, 2021; Zhang et al., 2019; Bachlechner et al., 2021). Two of the most well-known examples of such initialization are Glorot (Xavier) initialization for tanh networks (Glorot and Bengio, 2010) and He (Kaiming) initialization for ReLU networks (He et al., 2015). Glorot and Bengio (2010) studied forward and backward passes to obtain initialization recommendations for tanh activation function. Specifically, they suggest initializing weights in the lth layer as i.i.d. samples from a Gaussian distribution $N(0, 2/n_{l-1} + n_l)$ or from a uniform distribution $U[-a, a], a = \sqrt{6/(n_{l-1} + n_l)}$. He et al. (2015), in a similar fashion, studied forward and backward passes of a ReLU network. They suggest initializing weights in the lth layer of a ReLU neural network, as i.i.d. samples from a Gaussian distribution $N(0, 2/n_{l-1})$ or from a uniform distribution $U[-a, a], a = \sqrt{6/n_{l-1}}$. Hanin and Rolnick (2018); Hanin (2018) performed a more rigorous analysis of the gradients of ReLU networks. They also considered higher-order moments, confirmed the recommendation from He et al. (2015), and derived recommendations on the network architecture.

Stable initialization of maxout networks For the first time, a stable initialization specific to maxout networks was mentioned in Sun et al. (2018), who derived an initialization strategy inspired by Glorot and Bengio (2010); He et al. (2015) for rank $K = 2$ maxout networks. Taking a similar approach, we derive the first stable initialization for maxout networks with units of higher ranks in Chapter 3. There we consider balancing the forward pass, assuming Gaussian or uniform distribution on the pre-activation features of each layer. However, this assumption is not fully justified. Continuing this line of work, in Chapter 4, we analyze maxout network gradients, including the higher order moments, following the ideas in Hanin and Rolnick (2018); Hanin (2018), and provide a rigorous justification for the initialization suggested in Chapter 3.

Other types of initialization As mentioned above, the neural network initialization methods can be designed for different purposes. One is to ensure that the complexity of the function represented by a neural network is as high as possible. To achieve this, one can ensure the network has the maximum number of regions in the input space at initialization. Such a description of parameter choices maximizing the number of regions for a layer of maxout units has been given by Montúfar et al. (2022, Proposition 3.4). Another technique, suggested by Steinwart (2019) for ReLU networks, is to initialize parameters in such a way that the nonlinear locus of different units of a network is evenly spaced over the input space at initialization, which could lead to faster convergence of training. We consider several initializations of these other types for maxout networks in Chapter 3.

2.3 Tropical perspective on neural networks with piece-wise linear activation functions

Tropical geometry can be utilized to understand neural networks with piece-wise linear activation functions, including maxout and ReLU activations. This approach was first formalized in Zhang et al. (2018). The tropical geometry interpretation allows us to understand better the complexity of maxout units and their difference from ReLUs in Section 2.3.3 below. Additionally, we use statements, which are based on the tropical geometry approach, to prove a generic lower bound on the number of linear regions (Theorem 3.8) in Chapter 3. Outside this work, the tropical geometry was the basis for deriving sharp upper bounds on the number of linear regions of maxout networks (Montúfar et al., 2022), which bounds the expressivity of maxout networks and is closely related to this thesis topic.

Further, in Sections 2.3.1 and 2.3.2, we briefly recap the basics of tropical geometry and its application to neural networks, omitting many details, and follow Zhang et al. (2018). A detailed introduction to tropical geometry can be found in Maclagan and Sturmfels (2015).

2.3.1 Basic tropical geometry definitions

The most fundamental component of tropical algebraic geometry is the *tropical semiring* $T := (\mathbb{R} \cup \{-\infty\}, \oplus, \odot)$. The two operations $\oplus$ and $\odot$, called *tropical addition* and *tropical multiplication*, respectively, are defined as follows. For $x, y \in \mathbb{R}$, their tropical sum is

$$x \oplus y := \max\{x, y\},$$

and their tropical product is

$$x \odot y := x + y.$$

Additionally, the *tropical quotient* of x over y is defined as

$$x \oslash y := x - y.$$

Let $\mathbb{N} = \{n \in \mathbb{Z} : n \geq 0\}$. For an integer $a \in \mathbb{N}$, raising $x \in \mathbb{R}$ to the ath power is the same as multiplying x to itself a times. When standard multiplication is replaced by tropical multiplication, this gives us *tropical power*,

$$x^{\odot a} := x \odot \cdots \odot x = a \cdot x,$$

where the last $\cdot$ denotes the standard product of real numbers. We will write x^a instead of $x^{\odot a}$ for notation simplicity.

Then, a *tropical monomial* in d variables $x_1, \ldots, x_d$ is an expression of the form

$$c \odot x_1^{a_1} \odot x_2^{a_2} \odot \cdots \odot x_d^{a_d},$$

where $c \in \mathbb{R} \cup \{-\infty\}$ and $a_1, \ldots, a_d \in \mathbb{N}$. As a shorthand, we will write a tropical monomial as cx^α where $\alpha = (a_1, \ldots, a_d) \in \mathbb{N}^d$ and $x = (x_1, \ldots, x_d)$.

Subsequently, using the notation above, a *tropical polynomial* is defined as

$$f(x) = \sum_{i=0}^{r} c_i x^{\alpha_i},$$

where the sum is the tropical sum, $\alpha_i = (a_{i1}, \ldots, a_{id}) \in \mathbb{N}^d$ and $c_i \in \mathbb{R} \cup \{-\infty\}, i = 1, \ldots, r$.

Finally, a *tropical rational function* is the tropical quotient of two tropical polynomials:

$$f(x) \oslash g(x) = f(x) - g(x).$$

Tropical hypersurfaces prove to be useful for analyzing linear regions of neural networks. A *trop-*

ical hypersurface of a tropical polynomial $f : \mathbb{R}^d \to \mathbb{R}$ is defined as

$$\text{Trop}(f) := \{x \in \mathbb{R}^d : c_i x^{\alpha_i} = c_j x^{\alpha_j} = f(x) \text{ for } i \neq j \text{ two distinct monomials}\}.$$

2.3.2 Tropical geometry of neural networks

Zhang et al. (2018, Theorem 5.2) established that a feedforward neural network under assumptions that weight matrices are integer-valued and ReLU is used as an activation function is a function $\nu : \mathbb{R}^d \to \mathbb{R}^p$ whose coordinates are tropical rational functions of the input, i.e.,

$$\nu(x) = F(x) \oslash G(x),$$

where F and G are tropical polynomial maps. Thus ν is a tropical rational map. A similar statement holds for a maxout network, which is used in Montúfar et al. (2022).

One of the questions considered in this thesis is the complexity of the function computed by a neural network in terms of the number of linear regions in the input space. Using the statement above stating that a neural network can be regarded as a tropical rational map, we can apply tropical geometry to investigate this issue in the following way.

For a tropical polynomial $f(x) = \sum_{i=0}^{r} c_i x^{\alpha_i}$ define its Newton polytope as the convex hull of $\alpha_1, \dots, \alpha_r \in \mathbb{N}^d$. Specifically,

$$\Delta(f) := \text{conv}\{\alpha_i \in \mathbb{R}^d : c_i \neq -\infty, i = 1, \dots, r\}.$$

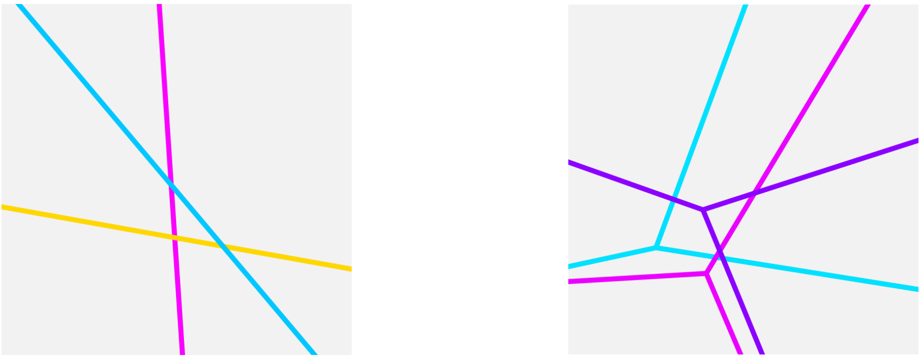
Then the lifted Newton polytope is defined as $\mathcal{P}(f) := \text{conv}\{(\alpha_i, c_i) \in \mathbb{R}^d \times \mathbb{R} : i = 1, \dots, r\}$. It is known that the number of vertices in $\mathcal{P}(f)$ provides an upper bound on the number of linear regions of f, and hence, an upper bound on the number of linear regions of a tropical rational function $f \oslash g$ (Zhang et al., 2018, Section 3).

2.3.3 Difference between linear regions of ReLU and maxout networks

Tropical geometry also provides a helpful framework for understanding the difference in complexity of the linear regions corresponding to ReLU and maxout networks. A layer of ReLUs is the following map:

$$\mathbb{R}^n \to \mathbb{R}^m; \quad \boldsymbol{x} \mapsto \begin{bmatrix} \max\{0, W_1\boldsymbol{x} + \boldsymbol{b}_1\} \\ \vdots \\ \max\{0, W_m\boldsymbol{x} + \boldsymbol{b}_m\} \end{bmatrix}.$$

It has linear regions separated by a hyperplane arrangement, which has been studied since the 19th century (Steiner, 1826; Buck, 1943; Zaslavsky, 1975), and the number of linear regions corresponding



(a) One layer of ReLUs subdivides the input space into linear regions by slicing it with *hyperplanes*. The number of linear regions is upper bounded by the number of vertices in a zonotope (Minkowski sum of line segments).

(b) One layer of maxout units subdivides the input space into linear regions by slicing it with *tropical hypersurfaces*. The number of linear regions is upper bounded by the number of vertices in a Minkowski sum of polytopes.

Figure 2.2: Linear regions of ReLU networks vs. linear regions of maxout networks. Observe that regions of a maxout network are created by the intersection of more complex objects.

to a single layer of ReLUs is upper bounded by the number of vertices in zonotopes (Minkowski sum of line segments). Therefore, the main challenge for ReLU networks is in studying the composition of layers. In contrast, a layer of maxout units is given by the following map

$$\mathbb{R}^n \to \mathbb{R}^m; \quad \boldsymbol{x} \mapsto \begin{bmatrix} \max\{W_{11}\boldsymbol{x} + \boldsymbol{b}_{11}, \dots, W_{1K}\boldsymbol{x} + \boldsymbol{b}_{1K}\} \\ \vdots \\ \max\{W_{m1}\boldsymbol{x} + \boldsymbol{b}_{m1}, \dots, W_{mK}\boldsymbol{x} + \boldsymbol{b}_{mK}\} \end{bmatrix},$$

and has regions separated by a tropical hypersurface arrangement. As a result, for a maxout network, the number of linear regions corresponding to one layer is upper bounded by the number of vertices in a Minkowski sum of polytopes. These objects are less studied than zonotopes, so understanding even one layer of maxout networks is challenging. We illustrate this issue in Figure 2.2.

2.4 NTK and implicit bias

Neural tangent kernel (NTK) was introduced in Jacot et al. (2018). Consider SGD update to one of the network parameters θ_p:

$$\Delta\theta_p = -\eta \frac{\partial \mathcal{L}}{\partial \theta_p}, \quad p = 1, \dots, P,$$

where $\eta \in \mathbb{R}$ is the learning rate, $\mathcal{L} : \mathbb{R}^P \to \mathbb{R}$ is an empirical loss, and $P = |\boldsymbol{\Theta}|$ is the total number of parameters in a neural network. The loss $\mathcal{L}$ is defined as $1/B \sum_{j=1}^{B} \ell(\mathcal{N}(\boldsymbol{x}_j; \Theta), y_j)$, where $\ell : \mathbb{R}^{n_0} \times \mathbb{R}^{n_L} \to \mathbb{R}$ is a per-sample cost function, and B is a batch size. Using Taylor expansion around the network parameters at initialization and the chain rule, we get that the update to the function computed by the network $\mathcal{N}$ is

$$\Delta\mathcal{N}(\boldsymbol{x}) = -\eta\,\langle K_{\mathcal{N}}(\boldsymbol{x},\cdot), \nabla\mathcal{L}(\cdot)\rangle = -\frac{\eta}{B}\sum_{j=1}^{B} K_{\mathcal{N}}(\boldsymbol{x},\boldsymbol{x}_j)\frac{\partial\ell}{\partial\mathcal{N}}(\boldsymbol{x}_j, y_j).$$

Here $K_{\mathcal{N}}(\boldsymbol{x},\boldsymbol{x}')$ is the NTK defined as

$$K_{\mathcal{N}}(\boldsymbol{x},\boldsymbol{x}') = \sum_{p=1}^{P}\frac{\partial\mathcal{N}}{\partial\theta_p}(\boldsymbol{x})\frac{\partial\mathcal{N}}{\partial\theta_p}(\boldsymbol{x}').$$

Consider a fully connected network with a Lipschitz, twice differentiable nonlinearity function $\sigma : \mathbb{R} \to \mathbb{R}$, with bounded second derivative. Assume that the NTK parametrization is applied to the pre-activations, meaning that they are normalized in the following way: $1/\sqrt{n_l}W^{(l)}\boldsymbol{x} + \beta\boldsymbol{b}^{(l)}$, where the scalar $\beta > 0$ is a parameter which allows tuning the influence of the bias on the training. Additionally, assume that the parameters are initialized as i.i.d. Gaussians $N(0, 1)$. Then, Jacot et al. (2018) observed that when the network depth is fixed, and the width tends to infinity, the NTK $K_{\mathcal{N}}$ simplifies and stays frozen during training at the infinite width limit of its average $\mathbb{E}[K_{\mathcal{N}}]$ at initialization, where the expectation is with respect to the distribution of the network parameters. Hence, the NTK of a finite network can be approximated by its expectation. NTK has proven useful for studying neural network optimization and its implicit biases, particularly for overparametrized networks. Numerous works analyzed NTK or used it in their studies, such as Chizat et al. (2019); Arora et al. (2019); Lee et al. (2019b); Williams et al. (2019); Dukler et al. (2020); Bowman and Montufar (2022); Jin and Montúfar (2023).

However, Hanin and Nica (2020a) showed that if both the depth and width of a ReLU network tend to infinity, the NTK does not converge to a constant in probability. By studying the expectation of the gradients of a maxout network in Chapter 4, we show in Corollary 4.9 that similarly to ReLU, the NTK of maxout networks does not converge to a constant when both width and depth are sent to infinity.

Chapter 3

On the expected complexity of maxout networks

3.1 Introduction

We are interested in the functions parametrized by artificial feedforward neural networks with maxout units. Maxout units compute parametric affine functions followed by a fixed multi-argument activation function of the form $(s_1, \ldots, s_K) \mapsto \max\{s_1, \ldots, s_K\}$ and can be regarded as a natural generalization of ReLUs, which have a single-argument activation function $s \mapsto \max\{0, s\}$. For any choice of parameters, these networks subdivide their input space into activation regions where different pre-activation features attain the maximum and the computed function is (affine) linear. We are concerned with the expected number of activation regions and their volume given probability distributions of parameters, as well as corresponding properties for the decision boundaries in classification tasks. We show that different architectures can attain very different numbers of regions with positive probability, but for parameter distributions for which the conditional densities of bias values and the expected gradients of activation values are bounded, the expected number of regions is at most polynomial in the rank K, and the total number of units.

Activation regions of neural networks For neural networks with piecewise linear activation functions, the number of activation regions serves as a complexity measure and summary description, which has proven useful in the investigation of approximation errors, Lipschitz constants, speed of convergence, implicit biases of parameter optimization, and robustness against adversarial attacks. In particular, Pascanu et al. (2014); Montufar et al. (2014); Telgarsky (2015, 2016) obtained depth separation results showing that deep networks can represent functions with many more linear regions than any of the functions that can be represented by shallow networks with the same number of units or parameters. This implies that certain tasks require enormous shallow networks but can be solved with small deep networks. The geometry of the boundaries between linear regions has been used to study function-preserving transformations of the network weights (Phuong and Lampert, 2019; Serra et al., 2020) and robustness (Croce et al., 2019; Lee et al., 2019a). Steinwart (2019) demonstrated empirically that the distribution of regions at initialization can be related to the speed of convergence of gradient descent, and Williams et al. (2019); Jin and Montúfar (2023) related the density of breakpoints at initialization to the curvature of the solutions after training. The properties of linear regions in relation to training have been recently studied by Zhang and Wu (2020). The number of regions has also been utilized to study the eigenvalues of the neural tangent kernel and Lipschitz constants (Nguyen et al., 2020).

Maximum number of regions Especially the maximum number of linear regions has been studied intensively. In particular, Montúfar (2017); Serra et al. (2018) improved the upper bounds from Montufar et al. (2014) by accounting for output dimension bottlenecks across layers. Hinz and Van de Geer (2019) introduced a histogram framework for a fine grained analysis of such dimensions in ReLU networks. Based on this, Xie et al. (2020); Hinz (2021) obtained still tighter upper bounds for ReLU networks. The maximum number of regions has been studied not only for fully connected networks, but also convolutional neural networks (Xiong et al., 2020), graph neural networks (GNNs) and message passing simplicial networks (MPSN) (Bodnar et al., 2021).

Expected number of regions Although the maximum possible number of regions gives useful complexity bounds and insights into different architectures, in practice one may be more interested in the expected behavior for typical choices of the parameters. The first results on the expected number of regions were obtained by Hanin and Rolnick (2019a,b) for the case of ReLU networks or single-argument piecewise linear activations. They show that if one has a distribution of parameters such that the conditional densities of bias values are bounded and the expected gradients of activation values are bounded, then the expected number of linear regions can be much smaller than the maximum theoretically possible number. Moreover, they obtain bounds for the expected number and volume of lower dimensional linear pieces of the represented functions. These results do not directly apply to the case of maxout units, but we will adapt the proofs to obtain corresponding results.

Regions of maxout networks Most previous works focus on ReLUs or single-argument activation functions. In this case, the linear regions of individual layers are described by hyperplane arrangements, which have been investigated since the 19th century (Steiner, 1826; Buck, 1943; Zaslavsky, 1975). Hence, the main challenge in these works is the description of compositions of several layers. In contrast, the linear regions of maxout layers are described by complex arrangements that are not so well understood yet. The study of maxout networks poses significant challenges already at the level of individual layers and in fact single units. For maxout networks, the maximum possible number of regions has been studied by Pascanu et al. (2014); Montufar et al. (2014); Serra et al. (2018). Recently, Montúfar et al. (2022) obtained counting formulas and sharp (asymptotic) upper bounds for the number of regions of shallow (deep) maxout networks. However, their focus was on the maximum possible value, and not on the generic behavior, which we investigate here.

Related notions The activation regions of neural networks can be approached from several perspectives. In particular, the functions represented by networks with piecewise linear activations correspond to so-called tropical rational functions and can be studied from the perspective of tropical geometry (Zhang et al., 2018; Charisopoulos and Maragos, 2018). In the case of piecewise affine convex nonlinearities, these can be studied in terms of so-called max-affine splines (Balestriero et al., 2019). A related but complementary notion of network expressivity is trajectory length, proposed by Raghu et al. (2017), which measures transitions between activation patterns along one-dimensional paths on the input space, which also leads to depth separation results. Recent work (Hanin et al., 2021) shows that ReLU networks preserve expected length.

Contributions We obtain the following results for maxout networks.

- There are widely different numbers of linear regions that are attained with positive probability over the parameters (Theorem 3.7). There is a non-trivial lower bound on the number of linear regions that holds for almost every choice of the parameters (Theorem 3.8). These results advance the maximum complexity analysis of Montúfar et al. (2022) from the perspective of generic parameters.

- For common parameter distributions, the expected number of activation regions is polynomial in the number of units (Theorem 3.9). Moreover, the expected volume of activation regions of different dimensions is polynomial in the number of units (Theorem 3.10). These results correspond to maxout versions of results from Hanin and Rolnick (2019b) and Hanin and Rolnick (2019a).

- For multi-class classifiers, we obtain an upper bound on the expected number of linear pieces (Theorem 3.11) and the expected volume (Theorem 3.12) of the decision boundary, along with a lower bound on the expected distance between input points and decision boundaries (Corollary 3.13).

- We provide an algorithm and implementation for counting the number of linear regions of maxout networks (Algorithm 3.1).

- We present parameter initialization procedures for maxout networks maximizing the number of regions or normalizing the mean activations across layers (similar to Glorot and Bengio 2010; He et al. 2015), and observe experimentally that these can lead to faster convergence of training.

3.2 Activation regions of maxout networks

We consider feedforward neural networks with n_0 inputs, L hidden layers of widths $n_1, \dots, n_L$, and no skip connections, which implement functions of the form $f = \psi \circ \phi_L \circ \cdots \circ \phi_1$. The l-th hidden layer implements a function $\phi_l \colon \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ with output coordinates, i.e. units, given by trainable affine functions followed by a fixed real-valued activation function, and $\psi \colon \mathbb{R}^{n_L} \to \mathbb{R}^{n_{L+1}}$ is a linear output layer. We denote the total number of hidden units by $N = n_1 + \cdots + n_L$, and index them by $z \in [N] := \{1, \dots, N\}$. The collection of all trainable parameters is denoted by θ.

We consider networks with maxout units, introduced by Goodfellow et al. (2013). A rank-K maxout unit with n inputs implements a function $\mathbb{R}^n \to \mathbb{R}$; $x \mapsto \max_{k\in[K]}\{w_k \cdot x + b_k\}$, where $w_k \in \mathbb{R}^n$ and $b_k \in \mathbb{R}$, $k \in [K]$, are trainable weights and biases. The activation function $(s_1, \dots, s_K) \mapsto \max\{s_1, \dots, s_K\}$ can be regarded as a multi-argument generalization of the rectified linear unit (ReLU) activation function $s \mapsto \max\{0, s\}$. The K arguments of the maximum are called the pre-activation features of the maxout unit. For unit z in a maxout network, we denote $\zeta_{z,k}(x; \theta)$ its k-th pre-activation feature, considered as a function of the input to the network.

For any choice of the trainable parameters, the function represented by a maxout network is piecewise linear, meaning it splits the input space into countably many regions over each of which it is linear.

Definition 3.1 (Linear regions)**.** Let $f \colon \mathbb{R}^{n_0} \to \mathbb{R}$ be a piecewise linear function. A linear region of f is a maximal connected subset of $\mathbb{R}^{n_0}$ on which f has a constant gradient.

We will relate the linear regions of the represented functions to activation regions defined next.

Definition 3.2 (Activation patterns)**.** An activation pattern of a network with N rank-K maxout units is an assignment of a non-empty set $J_z \subseteq [K]$ to each unit $z \in [N]$. An activation pattern $J = (J_z)_{z\in[N]}$ with $\sum_{z\in[N]}(|J_z| - 1) = r$ is called an r-partial activation pattern. The set of all possible activation patterns is denoted $\mathcal{P}$, and the set of r-partial activation patterns is denoted $\mathcal{P}_r$. An activation sub-pattern is a pattern where we disregard all J_z with $|J_z| = 1$. The set of all possible activation sub-patterns is denoted $\mathcal{S}$, and the set of r-partial activation sub-patterns is denoted $\mathcal{S}_r$.

Definition 3.3 (Activation regions)**.** Consider a network $\mathcal{N}$ with N maxout units. For any parameter value θ and any activation pattern J, the corresponding activation region is

$$\mathcal{R}(J, \theta) := \Big\{x \in \mathbb{R}^{n_0} \;\Big|\; \operatorname*{argmax}_{k\in[K]} \zeta_{z,k}(x; \theta) = J_z \text{ for each } z \in [N]\Big\}.$$

For any $r \in \{0, \dots, n_0\}$ we denote the union of r-partial activation regions by

$$\mathcal{X}_{\mathcal{N},r}(\theta) := \bigcup_{J\in\mathcal{P}_r} \mathcal{R}(J; \theta).$$

By these definitions, we have a decomposition of the input space as a disjoint union of activation regions, $\mathbb{R}^{n_0} = \sqcup_{J\in\mathcal{P}}\mathcal{R}(J,\theta)$. See Figure 3.1. Next we observe that for almost every choice of θ, r-partial activation regions are either empty or relatively open convex polyhedra of co-dimension r. In particular, for almost every choice of the parameters, if r is larger than n_0, the r-partial activation regions are empty. Therefore, in our discussion we only need to consider r up to n_0.

Lemma 3.4 (r-partial activation regions are relatively open convex polyhedra)**.** *Consider a maxout network $\mathcal{N}$. Let $r \in \{0,\dots,n_0\}$ and $J \in \mathcal{P}_r$. Then for any θ, $\mathcal{R}(J,\theta)$ is a relatively open convex polyhedron in $\mathbb{R}^{n_0}$. For almost every θ, it is either empty or has co-dimension r.*

The proof of Lemma 3.4 is given in Section 3.A. Next we show that for almost every choice of θ, 0-partial activation regions and linear regions correspond to each other.

Lemma 3.5 (Activation regions vs linear regions)**.** *Consider a maxout network $\mathcal{N}$. The set of parameter values θ for which the represented function has the same gradient on two distinct activation regions is a null set. In particular, for almost every θ, linear regions and activation regions correspond to each other.*

The proof of Lemma 3.5 is given in Section 3.A. We note that for specific parameters, linear regions can be the union of several activation regions and can be non-convex. Such a situation is more common in ReLU networks, whose units can more readily output zero, thereby hiding the activation pattern of the units in the previous layers.

To summarize the above observations, for almost every θ, the 0-partial activation regions are n_0-dimensional open convex polyhedra which agree with the linear regions of the represented function, and for $r = 1,\dots,n_0$ the r-partial activation regions are co-dimension-r polyhedral pieces of the boundary between linear regions. Next, we investigate the number of non-empty r-partial activation regions and their volume within given subsets of the input space. We are concerned with their generic numbers, where we use "generic" in the standard sense to refer to a positive Lebesgue measure event.

3.3 Numbers of regions attained with positive probability

We start with a simple upper bound.

Lemma 3.6 (Simple upper bound on the number of r-partial activation patterns)**.** *Let $r \in \mathbb{N}_0$. The number of r-partial activation patterns and sub-patterns in a network with a total of N rank-K maxout units are upper bounded by $|\mathcal{P}_r| \leq \binom{rK}{2r}\binom{N}{r}K^{N-r}$ and $|\mathcal{S}_r| \leq \binom{rK}{2r}\binom{N}{r}$ respectively.*

The upper bound has asymptotic order $O(N^r K^{N+r})$ in K and N. The proof of Lemma 3.6 is given in Section 3.A, where we also provide an exact but unhandy counting formula.

By definition, the number of r-partial activation patterns is a trivial upper bound on the number of non-empty r-partial activation regions for any choice of parameters. Depending on the network

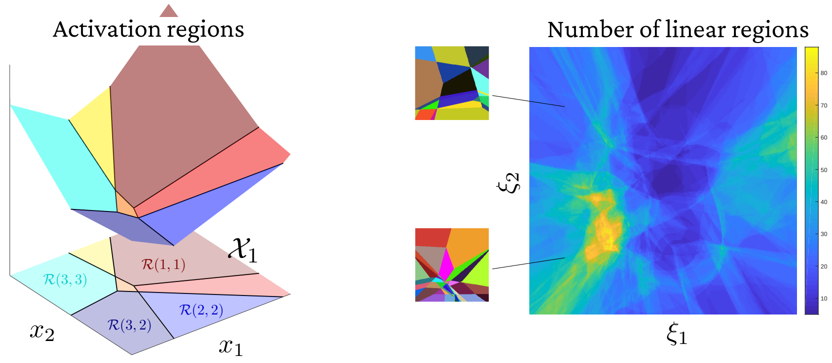


Figure 3.1: Left: Shown is a piecewise linear function $\mathbb{R}^2 \to \mathbb{R}$ represented by a network with a layer of two rank-3 maxout units for a choice of the parameters. The input space is subdivided into activation regions $\mathcal{R}(J;\theta)$ with linear regions separated by $\mathcal{X}_1(\theta)$. Right: Shown is the number of linear regions of a 3-layer maxout network over a portion of the input space as a function of a 2D affine subspace of parameter values $\theta(\xi_1,\xi_2)$. Shown are also two examples of the input-space subdivisions of functions represented by the network for different parameter values. More details about this figure are given in Section 3.K. As the figure illustrates, the function taking parameters to the number of regions is rather intricate. In this work, we characterize values attained with positive probability and upper bound the expected value given a parameter distribution.

architecture, this bound may not be attainable for any choice of parameters. Montúfar et al. (2022, Theorems 3.7 and 3.12) obtained bounds for the maximum number of linear regions. For a shallow network with n_0 inputs and a single layer of n_1 rank-K maxout units it has order $\Theta((n_1K)^{n_0})$ in K and n_1, and for a deep network with n_0 inputs and L layers of $n_1,\ldots,n_L$ rank-K maxout units it has order $\Theta(\prod_{l=1}^{L}(n_lK)^{n_0})$ in K and $n_1,\ldots,n_L$. Hence the maximum number of non-empty activation regions can be very large, especially for deep networks.

Intuitively, linear regions have a non-zero volume and cannot 'disappear' under small perturbations of parameters. This raises the question about which numbers of linear regions are attained with positive probability, i.e. over positive Lebesgue measure subsets of parameter values. Figure 3.1 shows that the number of linear regions of a maxout network is a very intricate function of the parameter values.

For a network with n_0 inputs and a single layer of n_1 ReLUs, the maximum number of linear regions is $\sum_{j=0}^{n_0}\binom{n_1}{j}$, and is attained for almost all parameter values. This is a consequence of the generic behavior of hyperplane arrangements (see Buck, 1943; Zaslavsky, 1975; Montufar et al., 2014). In contrast, shallow maxout networks can attain different numbers of linear regions with positive probability. The intuitive reason is that the nonlinear locus of maxout units is described not only by linear equations $\langle w_i, x\rangle + b_i = \langle w_j, x\rangle + b_j$ but also linear inequalities $\langle w_i, x\rangle + b_i \geq \langle w_k, x\rangle + b_k$. See Figure 3.1 for an example. We obtain the following result.

Theorem 3.7 (Numbers of linear regions)**.**

- *Consider a rank-K maxout unit with n_0 inputs. This corresponds to a network with an input layer of size n_0 and single maxout layer with a single maxout unit. For each $1 \leq k \leq K$, there is a set of parameter values for which the number of linear regions is k. For $\min\{K, n_0+1\} \leq k \leq K$, the corresponding set has positive measure, and else it is a null set.*
- *Consider a layer of n_1 rank-K maxout units with n_0 inputs. This corresponds to a network with a single maxout layer, $L = 1$, and $n_L = n_1$. For each choice of $1 \leq k_1, \ldots, k_{n_1} \leq K$, there are parameters for which the number of linear regions is $\sum_{j=0}^{n_0} \sum_{S \in \binom{[n_1]}{j}} \prod_{i \in S}(k_i - 1)$. For $\min\{K, n_0+1\} \leq k_1, \ldots, k_{n_1} \leq K$, the corresponding set has positive measure. Here $S \in \binom{[n_1]}{j}$ means that S is a subset of $[n_1] := \{1, \ldots, n_1\}$ of cardinality $|S| = j$.*
- *Consider a network with n_0 inputs and L layers of $n_1, \ldots, n_L$ rank-K maxout units, $K \geq 2$, $\frac{n_l}{n_0}$ even. Then, for each choice of $1 \leq k_{li} \leq K$, $i = 1, \ldots, n_0$, $l = 1, \ldots, L$, there are parameters for which the number of linear regions is $\prod_{l=1}^{L} \prod_{i=1}^{n_0} (\frac{n_l}{n_0}(k_{li} - 1) + 1)$. There is a positive measure subset of parameters for which the latter is the number of linear regions over $(0,1)^{n_0}$.*

The proof is provided in Section 3.B. The result shows that maxout networks have a multitude of positive measure subsets of parameters over which they attain widely different numbers of linear regions. In the last statement of the theorem we consider inputs from a cube, but qualitatively similar statements can be formulated for the entire input space.

There are specific parameter values for which the network represents functions with very few linear regions (e.g., setting the weights and biases of the last layer to zero). However, the smallest numbers of regions are only attained over null sets of parameters:

Theorem 3.8 (Generic lower bound on the number of linear regions)**.** *Consider a rank-K maxout network, $K \geq 2$, with n_0 inputs, n_1 units in the first layer, and any number of additional nonzero width layers. Then, for almost every choice of the parameters, the number of linear regions is at least $\sum_{j=0}^{n_0} \binom{n_1}{j}$ and the number of bounded linear regions is at least $\binom{n_1 - 1}{n_0}$.*

This lower bound has asymptotic order $\Omega(n_1^{n_0})$ in K and $n_1, \ldots, n_L$. The proof is provided in Section 3.B. To our knowledge, this is the first non-trivial probability-one lower bound for a maxout network. Note that this statement does not apply to ReLU networks unless they have a single layer of ReLUs. In the next section we investigate the expected number of activation regions for given probability distributions over the parameter space.

3.4 Expected number and volume of activation regions

For the expected number of activation regions we obtain the following upper bound, which corresponds to a maxout version of (Hanin and Rolnick, 2019b, Theorem 10).

Theorem 3.9 (Upper bound on the expected number of partial activation regions)**.** *Let $\mathcal{N}$ be a fully-connected feed-forward maxout network with n_0 inputs and a total of N rank K maxout units. Suppose we have a probability distribution over the parameters so that:*

1. *The distribution of all weights has a density with respect to the Lebesgue measure on $\mathbb{R}^{\#\text{weights}}$.*
2. *Every collection of biases has a conditional density with respect to Lebesgue measure given the values of all other weights and biases.*
3. *There exists $C_{\text{grad}} > 0$ so that for any $t \in \mathbb{N}$ and any pre-activation feature $\zeta_{z,k}$,*

$$\sup_{x \in \mathbb{R}^{n_0}} \mathbb{E}[\|\nabla \zeta_{z,k}(x)\|^t] \leq C_{\text{grad}}^t.$$

4. *There exists $C_{\text{bias}} > 0$ so that for any pre-activation features $\zeta_1, \dots, \zeta_t$ from any neurons, the conditional density of their biases $\rho_{b_1,\dots,b_t}$ given all the other weights and biases satisfies*

$$\sup_{b_1,\dots,b_t \in \mathbb{R}} \rho_{b_1,\dots,b_t}(b_1, \dots, b_t) \leq C_{\text{bias}}^t.$$

Fix $r \in \{0, \dots, n_0\}$ and let $T = 2^5 C_{\text{grad}} C_{\text{bias}}$. Then, there exists $\delta_0 \leq 1/(2C_{\text{grad}} C_{\text{bias}})$ such that for all cubes $C \subseteq \mathbb{R}^{n_0}$ with side length $\delta > \delta_0$ we have

$$\frac{\mathbb{E}[\#\ r\text{-partial activation regions of } \mathcal{N} \text{ in } C]}{\text{vol}(C)} \leq \begin{cases} \binom{rK}{2r}\binom{N}{r} K^{N-r}, & N \leq n_0 \\ \frac{(TKN)^{n_0}\binom{n_0 K}{2n_0}}{(2K)^r n_0!}, & N \geq n_0 \end{cases}.$$

Here the expectation is taken with respect to the distribution of weights and biases in $\mathcal{N}$. Of particular interest is the case $r = 0$, which corresponds to the number of linear regions.

The proof of Theorem 3.9 is given in Section 3.E. The upper bound has asymptotic order $O(N^{n_0} K^{3n_0 - r})$ in K and N, which is polynomial. In contrast, Montúfar et al. (2022) shows that the maximum number of linear regions of a deep network of width n is $\Theta((nK)^{\frac{n_0}{n} N})$, which for constant width is exponential in N; see Section 3.B. We present an analog of Theorem 3.9 for networks without biases in Section 3.G.

When the rank is $K = 2$, the formula coincides with the result obtained previously by Hanin and Rolnick (2019b, Theorem 10) for ReLU networks, up to a factor K^r. For some settings, we expect that the result can be further improved. For instance, for iid Gaussian weights and biases, one can show that the expected number of regions of a rank K maxout unit grows only like log K, as we discuss in Section 3.C.

We note that the constants C_{bias} and C_{grad} only need to be evaluated over the inputs in the region C. Intuitively, the bound on the conditional density of bias values corresponds to a bound on the density of non-linear locations over the input. The bound on the expected gradient norm of the pre-activation features is determined by the distribution of weights. We provide more details in Section 3.F.

For the expected volume of the r-dimensional part of the non-linear locus, we obtain the following upper bound, which corresponds to a maxout version of (Hanin and Rolnick, 2019a, Corollary 7).

Theorem 3.10 (Upper bound on the expected volume of the non-linear locus)**.** *Consider a bounded measurable set $S \subset \mathbb{R}^{n_0}$ and the settings of Theorem 3.9 with constants C_{grad} and C_{bias} evaluated over S. Then, for any $r \in \{1, \dots, n_0\}$,*

$$\frac{\mathbb{E}[\mathrm{vol}_{n_0-r}(\mathcal{X}_{\mathcal{N},r} \cap S)]}{\mathrm{vol}_{n_0}(S)} \leq (2C_{\mathrm{grad}}C_{\mathrm{bias}})^r \binom{rK}{2r}\binom{N}{r}.$$

The proof of Theorem 3.10 is given in Section 3.D. When the rank is $K = 2$, the formula coincides with the result obtained previously by Hanin and Rolnick (2019a, Corollary 7) for ReLU networks. A table comparing the results for maxout and ReLU networks is given in Section 3.E.

3.5 Expected number of pieces and volume of the decision boundary

In the case of classification problems, we are primarily interested in the decision boundary rather than the overall function. We define an M-class classifier by appending an argmax gate to a network with M outputs. The decision boundary is then a union of certain r-partial activation regions for the network with a maxout unit as the output layer. For simplicity, here we present the results for the $n_0 - 1$-dimensional regions, which we call 'pieces', and present the results for arbitrary values of r in Section 3.H. The number of pieces of the decision boundary is at most equal to the number of activation regions in the original network times $\binom{M}{2}$. A related statement appeared in Alfarra et al. (2020). For specific choices of the network parameters, the decision boundary does intersect most activation regions and can have as many as $\Omega(M^2 \prod_{l=1}^{L}(n_l K)^{n_0})$ pieces (see Section 3.H). However, in general, this upper bound can be improved. For the expected number of pieces and volume of the decision boundary, we obtain the following results. We write $\mathcal{X}_{\mathrm{DB}}$ for the decision boundary, and $\mathcal{X}_{\mathrm{DB},r}$ for the union of r-partial activation regions which include equations from the decision boundary (generically these are the co-dimension-r pieces of the decision boundary).

Theorem 3.11 (Upper bound on the expected number of linear pieces of the decision boundary)**.** *Let $\mathcal{N}$ be a fully-connected feedforward maxout network, with n_0 inputs, a total of N rank-K maxout units, and M linear output units used for multi-class classification. Under the assumptions of Theorem 3.9, there* *exists* $\delta_0 \leq 1/(2C_{\text{grad}}C_{\text{bias}})$ *such that for all cubes* $C \subseteq \mathbb{R}^{n_0}$ *with side length* $\delta > \delta_0$*,*

$$\frac{\mathbb{E}\left[\substack{\#\text{ linear pieces in the}\\ \text{decision boundary of }\mathcal{N}\text{ in }C}\right]}{\text{vol}(C)} \leq \begin{cases} \binom{M}{2}K^N, & N \leq n_0 \\ \frac{(2^4 C_{\text{grad}}C_{\text{bias}})^{n_0}(2KN)^{n_0-1}}{(n_0-1)!}\binom{M}{2}\binom{K(n_0-1)}{2(n_0-1)}, & N \geq n_0 \end{cases}.$$

Here the expectation is taken with respect to the distribution of weights and biases in $\mathcal{N}$*.*

For binary classification, $M = 2$, this bound has asymptotic order $O((K^3N)^{n_0-1})$ in K and N. For the expected volume, we have the following.

Theorem 3.12 (Upper bound on the volume of the $(n_0 - r)$-skeleton of the decision boundary)**.** *Consider a bounded measurable set* $S \subset \mathbb{R}^{n_0}$*. Consider the notation and assumptions of Theorem 3.9, whereby the constants* C_{grad} *and* C_{bias} *are over* S*. Then, for any* $r \in \{1, \ldots, n_0\}$ *we have*

$$\frac{\mathbb{E}[\text{vol}_{n_0-r}(\mathcal{X}_{\text{DB},r} \cap S)]}{\text{vol}_{n_0}(S)} \leq (2C_{\text{grad}}C_{\text{bias}})^r \sum_{i=1}^{\min\{M-1,r\}} \binom{M}{i+1}\binom{K(r-i)}{2(r-i)}\binom{N}{r-i}.$$

Moreover, the expected distance to the decision boundary can be bounded as follows.

Corollary 3.13 (Distance to the decision boundary)**.** *Suppose* $\mathcal{N}$ *is as in Theorem 3.9. For any compact set* $S \subset \mathbb{R}^{n_0}$ *let* x *be a uniform point in* S*. There exists* $c > 0$ *independent of* S *so that*

$$\mathbb{E}[\text{distance}(x, \mathcal{X}_{\text{DB}})] \geq \frac{c}{2C_{\text{grad}}C_{\text{bias}}M^{m+1}m},$$

where $m := \min\{M - 1, n_0\}$*.*

The proofs are presented in Section 3.H, where we also extend Theorem 3.11 to address the expected number of co-dimension-r pieces of the decision boundary. A corresponding result applies for the case of ReLU networks (see details in Section 3.H).

3.6 Experiments

In the experiments we used fully-connected networks. We describe the network architecture in terms of the depth and total number of units, with units evenly distributed across the layers with larger lower layers if needed. For instance, a network of depth 3 with 110 units has 3 hidden layers of widths $37, 37, 36$. Details and additional experiments are presented in Section 3.K. The computer implementation of the key functions is available on GitHub at `https://github.com/hanna-tseran/maxout_complexity`.

Initialization procedures We consider several initialization procedures detailed in Section 3.J: 1) ReLU-He initializes the parameters as iid samples from the distribution proposed by He et al. (2015)

Table 3.1: Standard deviation of the weight distribution for maxout-He initialization.

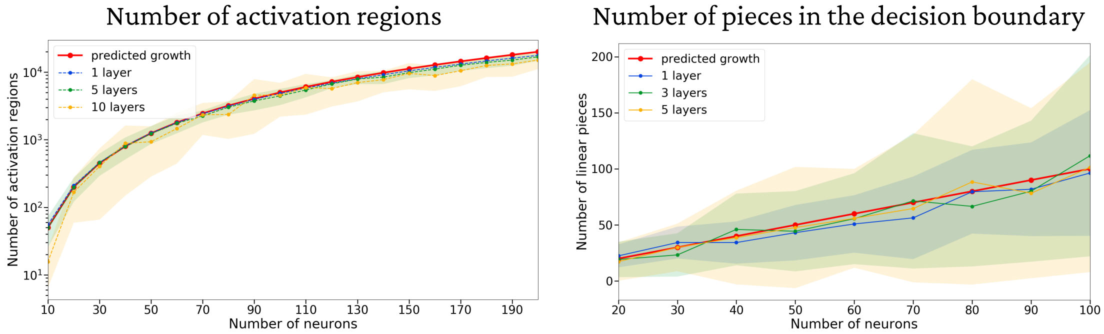
MAXOUT RANK	STANDARD DEVIATION
2	$\sqrt{1/n_l}$
3	$\sqrt{2\pi/((\sqrt{3}+2\pi)n_l)}$
4	$\sqrt{\pi/((\sqrt{3}+\pi)n_l)}$
5	$\sqrt{0.5555/n_l}$
ReLU	$\sqrt{2/n_l}$

for ReLUs. 2) Maxout-He follows a similar reasoning to normalize the expected norm of activation vectors across layers but for the case of maxout networks. The weight distribution has standard deviation depending on K and the assumed type of data distribution, as shown in Table 3.1. 3) "Sphere" ensures each unit has the maximum number of regions. 4) "Many regions" ensures each layer has the maximum number of regions.

Algorithm for counting activation regions Several approaches for counting linear regions of ReLU networks have been considered (e.g., Serra et al., 2018; Hanin and Rolnick, 2019b; Serra and Ramalingam, 2020; Xiong et al., 2020). For maxout networks, we count the activation regions and pieces of the decision boundary by iterative addition of linear inequality constraints and feasibility verification using linear programming. Pseudocode and complexity analysis are provided in Section 3.I.

Number of regions and decision boundary for different networks Figure 3.2 shows a close agreement, up to constants, of the theoretical upper bounds on the expected number of activation regions and on the expected number of linear pieces of the decision boundaries with the empirically obtained values for different networks. Further comparisons with constants and different values of K are provided in Section 3.K. Figure 3.3 shows that for common parameter distributions, the growth of the expected number of activation regions is more significantly determined by the total number of neurons than by the network's depth. In fact, we observe that for high rank units and certain types of distributions, deeper networks may have fewer activation regions. We attribute this to the fact that higher rank units tend to have smaller images (since they compute the max of more pre-activation features). Figure 3.4 shows how n_0 and K affect the number of activation regions. For small input dimension, the number of regions per unit tends to be smaller than K. Indeed, for iid Gaussian parameters the number of regions per unit scales as log K (see Section 3.C).

Number of regions during training We consider the 10-class classification task with the MNIST dataset (LeCun and Cortes, 2010) and optimization with Adam (Kingma and Ba, 2015) using different initialization strategies. Notice that for deep skinny fully connected networks the task is non-trivial. Figure 3.5 shows how the number of activation regions evolves during training. Shown are also the linear regions and decision boundaries over a 2D slice of the input space through 3 training data points. Figure 3.6 shows the training loss and accuracy curves for the different initializations. We observe that maxout networks with maxout-He, sphere, and many regions converge faster than with naive He initialization.


Number of activation regions

Number of pieces in the decision boundary

Figure 3.2: Shown are means and stds for 30 maxout-He normal initializations for networks with $K = 2$ and $n_0 = 2$. Left: Comparison of the theoretically predicted growth $O(N^{n_0}/n_0!)$ and the experimentally obtained number of regions for networks with different architectures. Right: Comparison of the theoretically predicted growth $O(N)$ and the experimentally obtained number of linear pieces of the decision boundary for networks with different architectures.

3.7 Discussion

We advance a line of analysis recently proposed by Hanin and Rolnick (2019a,b), where the focus lies on the expected complexity of the functions represented by neural networks rather than worst case bounds. Whereas previous works focus on single-argument activations, our results apply to networks with multi-argument maxout units. We observe that maxout networks can assume widely different numbers of linear regions with positive probability and then computed an upper bound on the expected number of regions and volume given properties of the parameter distribution, covering the case of zero biases. Further, taking the standpoint of classification, we obtained corresponding results for the decision boundary of maxout (and ReLU) networks, along with bounds on the expected distance to the decision boundary.

Experiments show that the theoretical bounds capture the general behavior. We present algorithms for enumerating the regions of maxout networks and proposed parameter initialization strategies with two types of motivations, one to increase the number of regions, and second, to normalize the variance of the activations similar to Glorot and Bengio (2010) and He et al. (2015), but now for maxout. We observed experimentally that this can improve training in maxout networks.

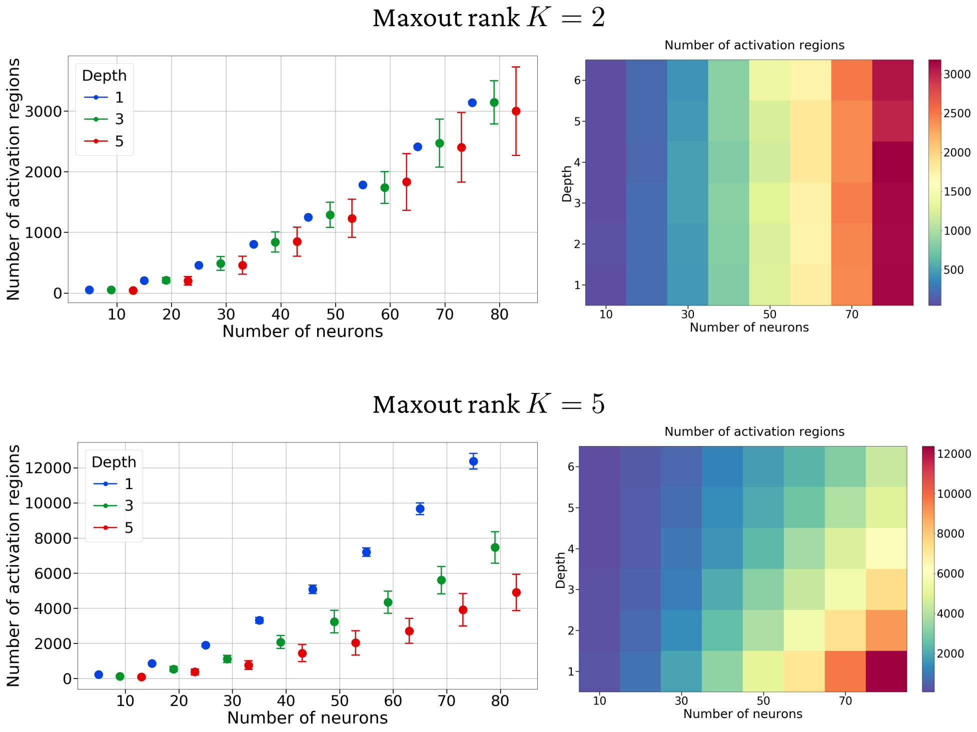


Figure 3.3: Effect of the depth and number of neurons on the number of activation regions at initialization for networks with $n_0 = 2$. Shown are means and stds for 30 maxout-He normal initializations.

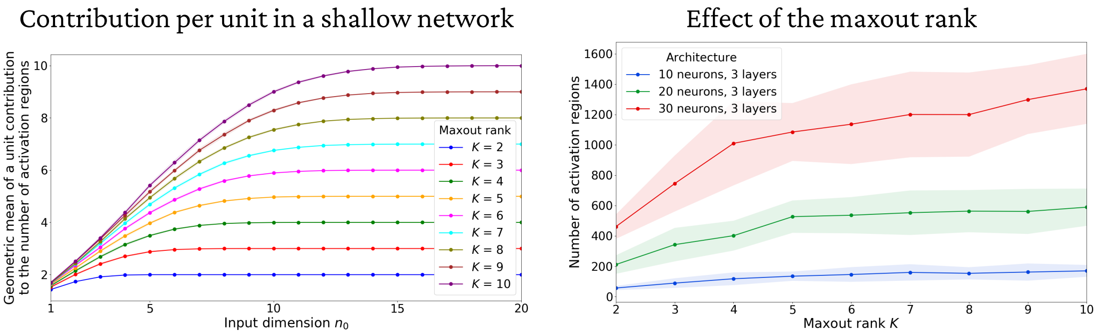


Figure 3.4: Left: Plotted is #regions$^{1/N}$ for a shallow network with $N = 5$. The multiplicative contribution per unit increases with the input dimension until the trivial upper bound K is reached. Right: Number of regions of 3 layer networks with $n_0 = 2$ depending on K. Shown are means and stds for 30 ReLU-He normal initializations.

Limitations In our theory and experiments, we have considered only fully connected networks. The analysis and implementation of other architectures for experiments with more diverse datasets are interesting extensions. By design, the results focus on parameter distributions that have a density.

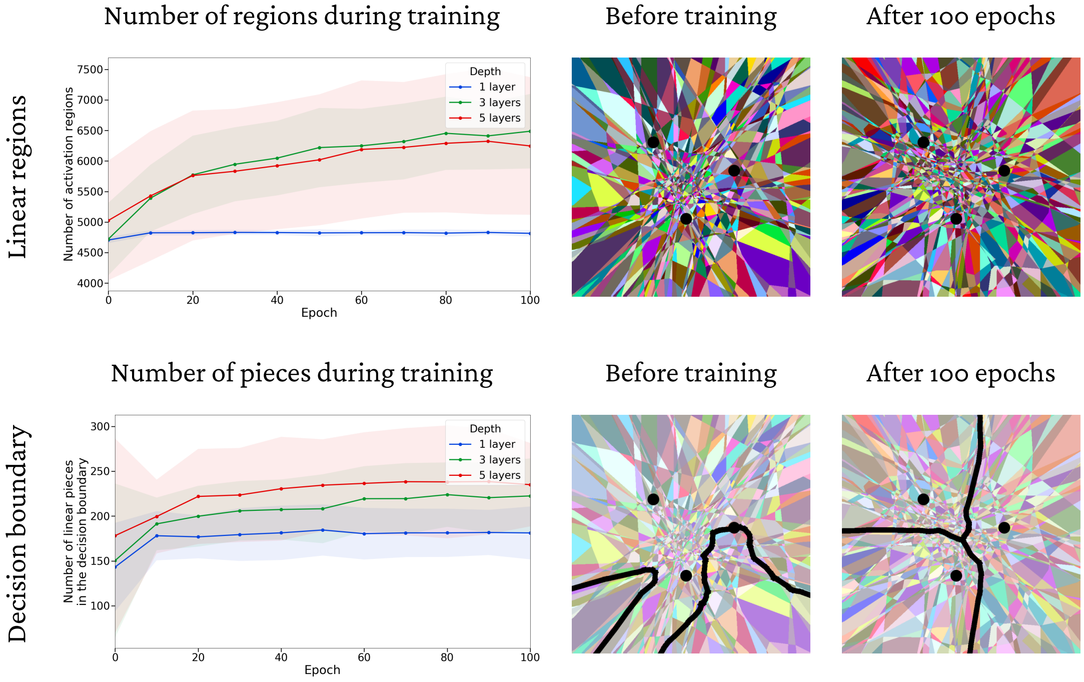


Figure 3.5: Evolution of the linear regions and the decision boundary during training on the MNIST dataset in a slice determined by three random points from different classes. The network had 100 maxout units of rank $K = 2$, and was initialized using maxout-He normal initialization. The right panel is for the 3 layer network. As expected, for the shallow rank-2 network, the number of regions is approximately constant. For deep networks, we observe a moderate increase in the number of regions as training progresses, especially around the training data. However, the number of regions remains far from the theoretical maximum. This is consistent with previous observations for ReLU networks. There is also a slight increase in the number of linear pieces in the decision boundary, and at the end of training the decision boundary clearly separates the three reference points.

Future work In future work, we would like to obtain a fine-grained description of the distribution of activation regions over the input space depending on the parameter distribution and explore the relations to the speed of convergence and implicit biases in gradient descent. Of significant interest would be an extension of the presented results to specific types of parameter distributions, including such which do not have a density or those one might obtain after training.

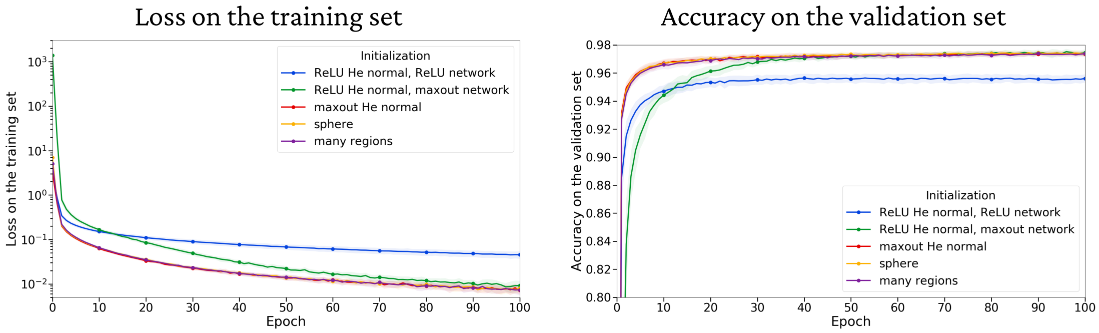


Figure 3.6: Comparison of training on MNIST with different initializations. All networks had 200 units, 10 layers, and maxout networks had rank $K = 5$. Shown are averages and std (barely noticeable) over 30 repetitions. The type of initialization has a significant impact on the training time of maxout networks, with maxout-He, sphere, and many regions giving better results for deep networks and larger maxout rank (more details on this in Section 3.K).

Proofs and experiment details

Proofs and experiment details are organized as follows.

- 3.A Proofs related to activation patterns and activation regions.
- 3.B Proofs related to the numbers of regions attained with positive probability.
- 3.C Expected number of regions for large rank.
- 3.D Proofs related to the expected volume of activation regions.
- 3.E Proofs related to the expected number of activation regions.
- 3.F Upper bounding the constants.
- 3.G Proofs related to the expected number of regions for networks with zero bias.
- 3.H Proofs related to the decision boundary.
- 3.I Algorithm for counting regions and pieces of the decision boundary.
- 3.J Initialization procedures.
- 3.K Details on the experiments and additional experiments.

3.A Proofs related to activation patterns and activation regions

3.A.1 Number of activation patterns

Lemma 3.6 (Simple upper bound on the number of r-partial activation patterns)**.** *Let* $r \in \mathbb{N}_0$*. The number of* r*-partial activation patterns and sub-patterns in a network with a total of* N *rank-*K *maxout units are upper bounded by* $|\mathcal{P}_r| \leq \binom{rK}{2r}\binom{N}{r}K^{N-r}$ *and* $|\mathcal{S}_r| \leq \binom{rK}{2r}\binom{N}{r}$ *respectively.*

Proof of Lemma 3.6. To get an r-partial activation pattern one needs at most r neurons. The number of ways to choose them is $\binom{N}{r}$. The number of ways to choose a pre-activation feature that attains a maximum in the rest of neurons is K^{N-r}. The r chosen neurons have in total rK pre-activation features. Out of them, we need to choose r features that attain maximum, and r additional features to construct the pre-activation pattern, so $2r$ features in total. We ignore the restriction that there needs to be at least one feature from each neuron, which gives us an upper-bound $r\binom{K}{2r}$. Notice that this way we also count r-partial patterns that require less than r neurons. Combining everything, we get the desired result. For the sub-patters, we simply ignore the term K^{n-r}. $\square$

We will use the above upper bound in our calculations due to its simplicity. For completeness, we note that the exact number of partial activation patterns can be given as follows.

Proposition 3.14 (Number of r-partial activation patterns)**.** *For a network with a total of N rank-K maxout units the number of distinct r-partial activation patterns is*

$$|\mathcal{P}_r| = \sum_{\substack{(N_0,\ldots,N_{K-1})\in\mathbb{N}_0^K:\\ \sum_{j=0}^{K-1} N_j = N, \sum_{j=0}^{K-1} jN_j = r}} \binom{N}{N_0,\ldots,N_{K-1}} \prod_{j=0}^{K-1} \binom{K}{1+j}^{N_j}.$$

If $K = 2$ then the summation index takes only one value $(N_0, N_1) = (N - r, r)$ and the expression simplifies to $\binom{N}{N-r}2^{N-r}$.

Proof. We have N neurons. For a given activation pattern, for $j = 0, \ldots, K-1$, denote N_j the number of neurons with $(1+j)$ pre-activation features attaining the maximum. Since every neuron has indecision in the range $0, \ldots, K-1$, we have $\sum_{j=0}^{K-1} N_j = N$. The r-partial activation patterns are precisely those for which $\sum_j jN_j = r$. The number of distinct ways in which we can partition the set of N neurons into K sets of cardinalities $N_0, \ldots, N_{K-1}$ is precisely $\binom{N}{N_0,\ldots,N_{K-1}}$. For each j, the number of ways in which a given neuron can have $(1+j)$ pre-activation features attaining the maximum is $\binom{K}{1+j}$. $\square$

3.A.2 Generic correspondence between activation regions and linear regions

For a fixed activation pattern J, a *computation path* γ is a path in the computation graph of the network $\mathcal{N}$ that goes from input to the output through one of the units in each layer, where $\gamma = (\gamma_0, \gamma_1, \ldots, \gamma_L)$, $\gamma_l \in [n_l] \times [K]$ specifies a unit and a corresponding pre-activation feature in layer l. For any input x in the activation region $\mathcal{R}(J,\theta)$, the gradient with respect to x can be expressed through the computation paths as

$$\nabla\mathcal{N}(x,\theta) = W_x^{(L+1)}W_x^{(L)}\cdots W_x^{(1)}, \qquad \frac{\partial}{\partial x_i}\mathcal{N}(x,\theta) = \sum_{\substack{\text{paths } \gamma\\ \text{starting at } i}} \prod_{l=1}^{L+1} w_\gamma^{(l)},$$

where in $W_x^{(l)} \in \mathbb{R}^{n_l\times n_{l-1}}$ is a piecewise constant matrix valued function of the input x with rows corresponding to the pre-activation features that attain the maximum according to the pattern J, and $w_\gamma^{(l)} \in \mathbb{R}$ are corresponding weights on the edge of γ between the layer $(l-1)$ and l, again depending on J. For a simple example of when one linear region is a union of several activation regions in a maxout network, consider a network with one of the weights in the single linear output unit set to zero. Such a situation can happen, for instance, at initialization, though with probability 0. Then, switching between the maximums in the unit in the previous layer to which this weight connects will not be visible when we compute the gradient, and several activation regions created by the transitions between maximums in this unit will become a part of the same linear region.

Lemma 3.5 (Activation regions vs linear regions)**.** *Consider a maxout network $\mathcal{N}$. The set of parameter values θ for which the represented function has the same gradient on two distinct activation regions is a null set. In particular, for almost every θ, linear regions and activation regions correspond to each other.*

Proof of the Lemma 3.5. Consider two different non-empty activation regions corresponding to activation patterns J_1 and J_2 for which $\nabla\mathcal{N}(x;\theta)$ has the same value. This means that n_0 equations of the form

$$\sum_{\text{paths } \gamma\in\Gamma_{1,i}} \prod_{l=1}^{L+1} w_\gamma^{(l)} = \sum_{\text{paths } \gamma\in\Gamma_{2,i}} \prod_{l=1}^{L+1} w_\gamma^{(l)}$$

are satisfied, where $\Gamma_{1,i}, \Gamma_{2,i}$ are collections of paths starting at i corresponding to the activation patterns J_1 and J_2 respectively. For different values of i the sets of paths differ only at the input layer.

Based on this equation, there exists $c_{\gamma,i} \in \{\pm 1\}$ and a non-empty collection of paths Γ_i (the symmetric difference of $\Gamma_{1,i}$ and $\Gamma_{2,i}$) so that

$$\sum_{\text{paths } \gamma\in\Gamma_i} c_{\gamma,i} \prod_{l=1}^{L+1} w_\gamma^{(l)} = 0.$$

This is a polynomial equation in the weights of the network. Each monomial occurs either with coefficient 1 or -1. In particular, this polynomial is not identically zero. The zero set of a polynomial is of measure zero on $\mathbb{R}^{\#\text{weights}}$ unless it is identically zero, see e.g. Caron and Traynor (2005). We have a system of n_0 such equations (one for each i). The intersection of the solution sets is again a set of measure zero. The total number of pairs of activation regions is finite, upper bounded by $\binom{K^N}{2}$. A countable union of measure zero sets is of measure zero, thus the set of weights for which two activation regions have the same gradient values has measure zero with respect to the Lebesgue measure on $\mathbb{R}^{\#\text{weights}}$. $\square$

3.A.3 Partial activation regions

Now we introduce several objects that are needed to discuss r-partial activation regions.

Definition 3.15. Fix a value θ of the trainable parameters. For a neuron z in $\mathcal{N}$ and a set $J_z \subseteq [K]$, the J_z**-activation region** of a unit z is

$$\mathcal{H}(J_z;\theta) := \{x_0 \in \mathbb{R}^{n_0} \mid \operatorname*{argmax}_{k\in[K]} \zeta_{z,k}(x_{l(z)-1};\theta) = J_z\}.$$

More generally, for a set of neurons $\mathcal{Z} = \{z\}$ and a corresponding list of sets $J_{\mathcal{Z}} = (J_z)_{z\in\mathcal{Z}}$, the corresponding $J_{\mathcal{Z}}$-activation region is

$$\mathcal{H}(J_{\mathcal{Z}};\theta) := \bigcap_{z\in\mathcal{Z}} \mathcal{H}(J_z;\theta). \tag{3.1}$$

If we specify an activation pattern for every neuron, $J_{[N]}$, so that $\mathcal{Z} = [N]$, then we write

$$\mathcal{R}(J_{[N]};\theta) = \mathcal{H}(J_{[N]};\theta).$$

Recall that an activation pattern $J_{[N]}$ with with the property that $\sum_z(|J_z|-1) = r$ is called an r-partial activation pattern. To distinguish such patterns, we denote them by $J^r \in \mathcal{P}_r$. The union of all corresponding activation regions is denoted

$$\mathcal{X}_{\mathcal{N},r}(\theta) = \bigcup_{J^r\in\mathcal{P}_r} \mathcal{R}(J^r;\theta).$$

Lemma 3.4 (r-partial activation regions are relatively open convex polyhedra)**.** *Consider a maxout network $\mathcal{N}$. Let $r \in \{0,\dots,n_0\}$ and $J \in \mathcal{P}_r$. Then for any θ, $\mathcal{R}(J,\theta)$ is a relatively open convex polyhedron in $\mathbb{R}^{n_0}$. For almost every θ, it is either empty or has co-dimension r.*

Proof of Lemma 3.4. Fix an r-partial activation pattern $J^r \in \mathcal{P}_r$. Over the activation region $\mathcal{R}(J;\theta)$, the k-th pre-activation feature of each neuron z is a linear function of the input to the network, namely

$$w^*_{z,k}\cdot x + b^*_{z,k} = w^{(l(z))}_{z,k}(w^{(l(z)-1)}\cdots(w^{(1)}\cdot x + b^{(1)})\cdots + b^{(l(z)-1)}) + b^{(l(z))}_{z,k},$$

where $w^*_{z,k}$ and $b^*_{z,k}$, $k \in [K]$ denote the weights and biases of this linear function, which depend on the weights and biases and activation values of the units up to unit z. For each z specify a fixed element $j_0 \in J_z$. The activation region can be written as

$$\bigcap_{z\in[N]} \{x\in\mathbb{R}^{n_0} \mid w^*_{z,j_0}\cdot x + b^*_{z,j_0} = w^*_{z,j}\cdot x + b^*_{z,j}, \quad \forall j \in J_z\setminus\{j_0\};$$
$$w^*_{z,j_0}\cdot x + b^*_{z,j_0} > w^*_{z,i}\cdot x + b^*_{z,i}, \quad \forall i \in [K]\setminus J_z\}.$$

This means that an r-partial activation region is determined by a set of strict linear inequalities and r linear equations. The equations are represented by vectors $v_{z,j} = (w^*_{z,j_0}, b^*_{z j_0}) - (w^*_{z,j}, b^*_{z,j})$ for all $j \in J_z \setminus \{j_0\}$ for all z for which $|J_z| > 1$. For generic parameters these equations are linearly independent. Indeed, the vectors being linearly dependent means that there is a matrix $V^\top V$, where V has rows $v_{z,j}$, with vanishing determinant. By similar arguments as in the proof of Lemma 3.5, the set of parameters solving a polynomial system has measure zero. Hence, for generic choices of parameters, the r linear equations are independent and the polyhedron will have a co-dimension r (or otherwise be empty). □

The same result can be obtained for r-partial activation regions of ReLU networks since ReLU activation regions can be similarly written as a system of linear equations and inequalities.

We can make a statement about the shape of r-partial activation regions of maxout networks. Recall that a *convex polyhedron* is the closure of the solution set to finite system of linear inequalities. If it is bounded, it is called a convex polytope. The dimension of a polyhedron is the dimension of the smallest affine space containing it.

The next statement follows immediately from Lemma 3.4.

Lemma 3.16 ($\mathcal{X}_{\mathcal{N},r}$ consists of $(n_0 - r)$-dimensional pieces)**.** *With probability* 1 *with respect to the distribution of the network parameters* θ*, for any* $x \in \mathcal{X}_{\mathcal{N},r}$ *there exists* $\varepsilon > 0$ *(depending on* x *and* θ*) s.t.* $\mathcal{X}_{\mathcal{N},r}$ *intersected with the* ε *ball* $B_\varepsilon(x)$ *is equal to the intersection of this ball with an* $(n_0 - r)$*-dimensional affine subspace of* $\mathbb{R}^{n_0}$.

Corollary 3.17 (r-partial activation regions are relatively open convex polyhedra)**.** *Recall that an an* r*-partial activation sub-pattern* $\hat{J} \in \mathcal{S}_r$ *is a list* $\hat{J} = (J_z)_{z \in Z}$ *of sets* $J_z \subseteq [K]$, $z \in Z \subseteq [N]$ *with* $|J_z| > 1$ *and* $\sum_{z \in Z}(|J_z| - 1) = r$. *For almost all choices of the parameter (i.e., except for a null set with respect to the Lebesgue measure),*

$$\mathrm{vol}_{n_0-r}\left(\mathcal{X}_{\mathcal{N},r}(\theta)\right) = \sum_{\hat{J} \in \mathcal{S}_r} \mathrm{vol}_{n_0-r}(\mathcal{H}(\hat{J};\theta)).$$

Proof of Corollary 3.17. Given $\hat{J} \in \mathcal{S}_r$, we denote $Z \subseteq [N]$ the corresponding list of neurons. Using the notion of indecision loci from Definition 3.15, we can re-write $\mathcal{X}_{\mathcal{N},r}(\theta)$ as

$$\begin{aligned}\mathcal{X}_{\mathcal{N},r}(\theta) &= \bigcup_{J\in\mathcal{P}_r}\mathcal{R}(J;\theta) = \bigcup_{J\in\mathcal{P}_r}\mathcal{H}(J;\theta) = \bigcup_{J\in\mathcal{P}_r}\bigcap_{z\in[N]}\mathcal{H}(J_z;\theta)\\ &= \bigcup_{J\in\mathcal{P}_r}\left[\bigcap_{z\in Z}\mathcal{H}(J_z;\theta)\cap\bigcap_{z\in[N]\setminus Z}\mathcal{H}(J_z;\theta)\right]\\ &= \bigcup_{\hat{J}\in\mathcal{S}_r}\left[\bigcap_{z\in Z}\mathcal{H}(J_z;\theta)\cap\bigcup_{J_z\in[K],z\in[N]\setminus Z}\ \bigcap_{z\in[N]\setminus Z}\mathcal{H}(J_z;\theta)\right]\\ &= \bigcup_{\hat{J}\in\mathcal{S}_r}\left[\bigcap_{z\in Z}\mathcal{H}(J_z;\theta)\cap\bigcap_{z\notin Z}\bigcup_{k\in[K]}\mathcal{H}(J_z=\{k\};\theta)\right].\end{aligned}$$

Therefore,

$$\mathrm{vol}_{n_0-r}\left(\mathcal{X}_{\mathcal{N},r}(\theta)\right) = \sum_{\hat{J}\in\mathcal{S}_r}\mathrm{vol}_{n_0-r}\left(\bigcap_{z\in Z}\mathcal{H}(J_z;\theta)\cap\bigcap_{z\notin Z}\bigcup_{k\in[K]}\mathcal{H}(J_z=\{k\};\theta)\right).$$

Notice that $\left(\bigcap_{z\notin Z}\bigcup_{k\in[K]}\mathcal{H}(J_z=\{k\};\theta)\right)^c$ is a zero measure set in $\mathcal{X}_{\mathcal{N},r}(\theta)$, because over that set, by Lemma 3.16 the co-dimension of the corresponding activation regions is larger than r. Therefore, for any given $\hat{J}=(J_z)_{z\in Z}\in\mathcal{S}_r$,

$$\mathrm{vol}_{n_0-r}\left(\bigcap_{z\in Z}\mathcal{H}(J_z;\theta)\cap\bigcap_{z\notin Z}\bigcup_{k\in[K]}\mathcal{H}(J_z=\{k\};\theta)\right) = \mathrm{vol}_{n_0-r}\left(\bigcap_{z\in Z}\mathcal{H}(J_z;\theta)\right).$$

This completes the proof. □

3.B Proofs related to the generic numbers of regions

3.B.1 Number of regions and Newton polytopes

We start with the observation that the linear regions of a maxout unit correspond to the upper vertices of a polytope constructed from its parameters.

Definition 3.18. Consider a function of the form $f\colon\mathbb{R}^n\to\mathbb{R};\ f(x)=\max\{w_j\cdot x+b_j\}$, where $w_j\in\mathbb{R}^n$ and $b_j\in\mathbb{R}$, $j=1,\dots,M$. The *lifted Newton polytope* of f is defined as $P_f:=\mathrm{conv}\{(w_j,b_j)\in\mathbb{R}^{n+1}\colon j=1,\dots,M\}$.

Definition 3.19. Let P be a polytope in $\mathbb{R}^{n+1}$ and let F be a face of P. An outer normal vector of F is a vector $v\in\mathbb{R}^{n+1}$ with $\langle v,p-q\rangle>0$ for all $p\in F,q\in P\setminus F$ and $\langle v,p-q\rangle=0$ for all $p,q\in F$.

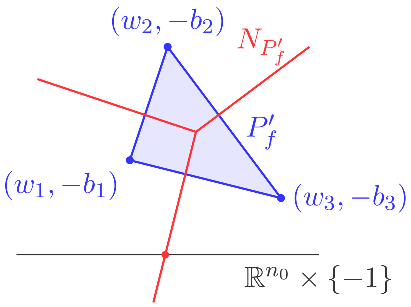


Figure 3.7: The linear regions of a function $f(x) = \max_j\{\langle w_j, x\rangle + b_j\}$ correspond to the lower vertices of the polytope $P'_f = \text{conv}_j\{(w_j, -b_j)\} \subseteq \mathbb{R}^{n_0+1}$, or, equivalently, the upper vertices of the lifted Newton polytope $P_f = \text{conv}_j\{(w_j, b_j)\} \subseteq \mathbb{R}^{n_0+1}$. The linear regions of f can also be described as the intersection of the normal fan $N_{P'_f}$, consisting of outer normal cones of faces of P'_f, with the affine space $\mathbb{R}^{n_0} \times \{-1\}$.

The face F is an *upper face* of P if it has an outer normal vector v whose last coordinate is positive, $v_{n+1} > 0$. It is a *strict upper face* if each of its outer normal vectors has a positive last coordinate.

The Newton polytope is a fundamental object in the study of polynomials. The naming in the context of piecewise linear functions stems from the fact that piecewise linear functions can be regarded as differences of so-called tropical polynomials. The connections between such polynomials and neural networks with piecewise linear activation functions have been discussed in several recent works (Zhang et al., 2018; Charisopoulos and Maragos, 2018; Alfarra et al., 2020). For details on tropical geometry, see (Maclagan and Sturmfels, 2015; Joswig, 2022). Although in the context of (tropical) polynomials the coefficients are integers, such a restriction is not needed in our discussion.

A convex analysis interpretation of the Newton polytope can be given as follows. Consider a piecewise linear convex function $f\colon \mathbb{R}^n \to \mathbb{R};\ x \mapsto \max_j\{w_j \cdot x + b_j\}$. Then the upper faces of its lifted Newton polytope P_f correspond to the graph $\{(x^*, -f^*(x^*))\colon x^* \in \mathbb{R}^n \cap \text{dom}(f^*)\}$ of the negated convex conjugate $f^*\colon \mathbb{R}^n \to \mathbb{R};\ x^* \mapsto \sup_{x\in\mathbb{R}^n}\langle x, x^*\rangle - f(x)$, which is a convex piecewise linear function. This implies that the upper vertices of P_f are the points $(w_j, b_j) \in \mathbb{R}^{n+1}$ for which $f(x) = w_j \cdot x + b_j$ over a neighborhood of inputs. Hence the upper vertices of the Newton polytope correspond to the linear regions of f. This relationship holds more generally for boundaries between linear regions and other lower dimensional linear features of the graph of the function. We will use the following result, which is well known in tropical geometry (see Joswig, 2022).

Proposition 3.20 (Regions correspond to upper faces)**.** *The r-partial activation regions of a function* $f(x) = \max_j\{w_j \cdot x + b_j\}$ *correspond to the r-dimensional upper faces of its lifted Newton polytope P_f. Moreover, the bounded activation regions correspond to the strict upper faces of P_f.*

The situation is illustrated in Figure 3.7.

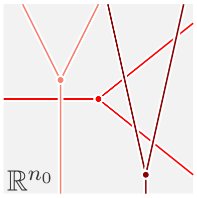


Figure 3.8: A layer of maxout units of rank $K \geq 3$ attains several different numbers of linear regions with positive probability over the parameters. For a layer with two rank-3 maxout units, some neighborhoods of parameters give 6 linear regions and others 9, with nonlinear loci given by perturbations of the red-pink and red-darkred lines.

3.B.2 Bounds on the maximum number of linear regions

For reference, we briefly recall results providing upper bounds on the maximum number of linear regions of maxout networks. The maximum number of regions of maxout networks was studied by Pascanu et al. (2014); Montufar et al. (2014), showing that deep networks can represent functions with many more linear regions than any of the functions that can be represented by a shallow network with the same number of units or parameters. Serra et al. (2018) obtained an upper bound for deep maxout networks based on multiplying upper bounds for individual layers. These bounds were recently improved by Montúfar et al. (2022), who obtained the following result, here stated in a simplified form.

Theorem 3.21 (Maximum number of linear regions, Montúfar et al. 2022)**.**

- *For a network with n_0 inputs and a single layer of n_1 rank-K maxout units, the maximum number of linear regions is $\sum_{j=0}^{n_0} \binom{n_1}{j}(K-1)^j$.*
- *For a network with n_0 inputs and L layers of $n_1, \ldots, n_L$ rank-K maxout units, if $n \leq n_0$, $\frac{n_l}{n}$ even, and $e_l = \min\{n_0, \ldots, n_{l-1}\}$, the maximum number of linear regions is lower bounded by $\prod_{l=1}^{L}(\frac{n_l}{n}(K-1)+1)^n$ and upper bounded by $\prod_{l=1}^{L}\sum_{j=0}^{e_l}\binom{n_l}{j}(K-1)^j$.*

3.B.3 Numbers of regions attained over positive measure subsets of parameters

A layer of maxout units can attain several different numbers of linear regions with positive probability over the parameters. This is illustrated in Figure 3.8. We obtain the following result, describing numbers of linear regions that can be attained by maxout units, layers, and deep maxout networks with positive probability over the parameters.

Theorem 3.7 (Numbers of linear regions)**.**

- *Consider a rank-K maxout unit with n_0 inputs. This corresponds to a network with an input layer of size n_0 and single maxout layer with a single maxout unit. For each $1 \leq k \leq K$, there is a set of parameter values for which the number of linear regions is k. For $\min\{K, n_0+1\} \leq k \leq K$, the corresponding set has positive measure, and else it is a null set.*

- *Consider a layer of n_1 rank-K maxout units with n_0 inputs. This corresponds to a network with a single maxout layer, $L = 1$, and $n_L = n_1$. For each choice of $1 \leq k_1, \dots, k_{n_1} \leq K$, there are parameters for which the number of linear regions is $\sum_{j=0}^{n_0} \sum_{S \in \binom{[n_1]}{j}} \prod_{i \in S}(k_i - 1)$. For $\min\{K, n_0+1\} \leq k_1, \dots, k_{n_1} \leq K$, the corresponding set has positive measure. Here $S \in \binom{[n_1]}{j}$ means that S is a subset of $[n_1] := \{1, \dots, n_1\}$ of cardinality $|S| = j$.*
- *Consider a network with n_0 inputs and L layers of $n_1, \dots, n_L$ rank-K maxout units, $K \geq 2$, $\frac{n_l}{n_0}$ even. Then, for each choice of $1 \leq k_{li} \leq K$, $i = 1, \dots, n_0$, $l = 1, \dots, L$, there are parameters for which the number of linear regions is $\prod_{l=1}^{L} \prod_{i=1}^{n_0} (\frac{n_l}{n_0}(k_{li} - 1) + 1)$. There is a positive measure subset of parameters for which the latter is the number of linear regions over $(0, 1)^{n_0}$.*

The strategy of the proof is as follows. We first show that there are parameters such that individual rank-K maxout units behave as rank-k maxout units, for any $1 \leq k \leq K$, and there are positive measure subsets of the parameters for which they behave as rank-k maxout units, for any $n+1 \leq k \leq K$. Further, there are positive measure subsets of the parameters of individual rank-K maxout units for which, over the positive orthant $\mathbb{R}^n_{\geq 0}$, they behave as rank-k maxout units, for any $1 \leq k \leq K$. Then we use a similar strategy as Montúfar et al. (2022) to construct parameters of a network with units of pre-specified ranks which attain a particular number of linear regions.

Proposition 3.22. *Consider a rank-K maxout unit with n inputs restricted to $\mathbb{R}^n_{\geq 0}$. For any $1 \leq k \leq K$, there is a positive measure subset of parameters for which the behaves as a rank-k maxout unit. Moreover, this set can be made to contain parameters representing any desired function that can be computed by a rank-k maxout unit.*

Proof. We need to show that for any choices of (w_i, b_i), $i \in [k]$, there are generic choices of (w_j, b_j), $j \in [K] \setminus [k]$, so that for each $J \subseteq [K]$ with $J \nsubseteq [k]$, the corresponding activation region $\mathcal{R}(J, \theta)$ does not intersect $\mathbb{R}^n_{\geq 0}$. Notice that, if $j \in J \setminus [k]$, then the corresponding activation region $\mathcal{R}(J, \theta)$ is contained in the arrangement consisting of hyperplanes $H_{ji} = \{x \colon (w_j - w_i) \cdot x + (b_j - b_i) = 0\}$, $i \in J \setminus \{j\}$. For each $j \in [K] \setminus [k]$, we choose $w_j = jc(-1, \dots, -1) + \epsilon_j$, $b_j = -jc' + \epsilon'_j$ for some $c > 2 \max\{\|w_i\|_\infty \colon i \in [k]\}$, $c' > 2 \max\{b_i \colon i \in [k]\}$ and small $\epsilon_j \in \mathbb{R}^n$, $\epsilon'_j \in \mathbb{R}$. Then, for each $j \in [K] \setminus [k]$ and $i \in [K]$, $j < j$, the hyperplane H_{ji} has a normal vector $(w_j - w_i) \in \mathbb{R}_{<0}$ and an intercept $b_j - b_i < 0$, and hence it does not intersect $\mathbb{R}^n_{\geq 0}$. □

We are now ready to prove the theorem.

Proof of Theorem 3.7. Single unit. Consider a maxout unit $\max_{j \in [K]}\{w_j \cdot x + b_j\}$. To have this behave as a rank-k maxout unit, $1 \leq k \leq K$, we simply set $(w_j, b_j) = (w_1, b_1 - 1)$, $j \in [K] \setminus [k]$. This is a non-generic choice of parameters. Consider now a rank-k maxout unit with $n + 1 \leq k$ and generic parameters (w_i, b_i), $i \in [k]$. We want to show that there are generic choices of (w_j, b_j), $j \in [K] \setminus [k]$ so that $\max_{j \in [K]}\{w_j \cdot x + b_j\} = \max_{i \in [k]}\{w_i \cdot x + b_i\}$ for all $x \in \mathbb{R}^n$. In view of Proposition 3.20, this is equivalent to (w_j, b_j), $j \in [K] \setminus [k]$ not being upper vertices of the lifted

Newton polytope $P = \operatorname{conv}\{(w_j, b_j) \colon j \in [K]\}$. Since any generic $n+1$ points in $\mathbb{R}^n$ are affinely independent, we have that the convex hull $\operatorname{conv}\{w_i \in \mathbb{R}^n \colon i \in [k]\}$ has full dimension n. Hence, any $w_j = \frac{1}{k}\sum_{i\in[k]} w_i + \epsilon_j$ and $b_j = \min_{i\in[k]}\{b_i\} - 1 + \epsilon'_j$ with sufficiently small $\epsilon_j \in \mathbb{R}^n$, $\epsilon'_j \in \mathbb{R}$, $j \in [K] \setminus [k]$ are strictly below $\operatorname{conv}\{(w_i, b_i) \colon i \in [k]\}$ and are not upper vertices of P.

Single layer. We use the previous item to obtain n_1 maxout units of ranks $k_1, \dots, k_{n_1}$, either in the non-generic or in the generic cases. Then we apply the construction of parameters and the region counting argument from Montúfar et al. (2022, Proposition 3.4) to this layer, to obtain a function with $\sum_{j=0}^{n_0} \sum_{S \subseteq \binom{[n_1]}{j}} \prod_{i\in S}(k_i - 1)$ linear regions. For each of the units $i = 1, \dots, n_1$, one may choose a generic vector $v_i \in \mathbb{R}^n$ and define the weights and biases of the pre-activation features as $w_{ij} = \frac{j}{k_i} v_i$ and $b_{ij} = -g(\frac{j}{k_i} + \epsilon_i)$, $j = 1, \dots, k_i$, where $g \colon \mathbb{R} \to \mathbb{R}$ is any strictly convex function and ϵ_i is chosen generically. Then the non-linear locus of each unit consists of $k_i - 1$ parallel hyperplanes with a generic shift ϵ_i, and the normal vectors v_i of different units are in general position. The number of regions defined by such an arrangement of hyperplanes in $\mathbb{R}^n$ can be computed using Zaslavsky's theorem, giving the indicated result. It remains to show that, for $n_0 + 1 \leq k_1, \dots, k_{n_1}$, there are positive measure perturbations of these parameters that do change the number of regions. By the lower semi-continuity discussed in Section 3.3, the number of regions does not decrease for sufficiently small generic perturbations of the parameters. To show that it does not increase, we note that, by Theorem 3.21 this number of regions is the maximum that can be attained by a layer of n_1 maxout units of ranks $k_1, \dots, k_n$.

Deep network. For the first statement, we use the first item to obtain maxout units of any desired ranks $1 \leq k_{li} \leq K, l = 1, \dots, L, i = 1, \dots, n_l$, and then apply the construction of parameters from Montúfar et al. (2022, Proposition 3.11) to this network, to obtain the indicated number of regions.

For the second statement, we use Proposition 3.22 to have the units behave as maxout units of any desired ranks over $[0,1]^{n_0}$. For the l-th layer, we divide the n_l units into n_0 blocks $x_{ij}^{(l)}$, $i = 1, \dots, n_0$, $j = 1, \dots, \frac{n_l}{n_0}$. For $i = 1, \dots, n_0$, the i-th block consists of $\frac{n_l}{n_0}$ maxout units of rank k_{li}. We can choose the weights and biases so that over $[0,1]^{n_0}$, the nonlinear locus of the i-th block $(x_{i,1}^{(1)}, \dots, x_{i,\frac{n_1}{n_0}}^{(1)})$ consists of $\frac{n_l}{n_0}(k_{li} - 1)$ parallel hyperplanes with normal e_i, and the alternating sum $\sum_{j=1}^{n_l/n_0}(-1)^j x_{ij}^{(l)}$ is a zig-zag function along the direction e_i which maps $(0,1)^{n_0}$ to $(0,1)$, and maps any point in $\mathbb{R}^{n_0} \setminus [0,1]^{n_0}$ to a point in $\mathbb{R} \setminus [0,1]$. In this way, the l-th layer, followed by a linear layer $\mathbb{R}^{n_l} \to \mathbb{R}^{n_0}$, maps $(0,1)^{n_0}$ onto $(0,1)^{n_0}$ in a $\prod_{i=1}^{n_0}(\frac{n_l}{n_0}(k_{li} - 1) + 1)$ to one manner. Sufficiently small perturbations of the parameters do not affect this general behavior. The composition of L such layers gives the desired number of regions over $(0,1)^{n_0}$. ☐

3.B.4 Minimum number of activation regions

One can easily construct parameters so that the represented function is identically zero. However, these are very special parameters. Moreover, it can be shown that the number of linear regions of a maxout network is a lower semi-continuous function of the parameters, in the sense that sufficiently small generic perturbations of the parameters do not decrease the number of linear regions (Montúfar et al., 2022, Proposition 3.2). Hence, the question arises: What is the smallest number of linear regions that will occur with positive probability over the parameter space (i.e. for all parameters except for a null set). For example, in the case of shallow ReLU networks, it is known that the number of regions for generic parameters is equal to the maximum. For maxout networks we saw in Theorem 3.7 that several numbers of linear regions can happen with positive probability. We prove the following lower bound on the number of regions for maxout networks with generic parameters.

Theorem 3.8 (Generic lower bound on the number of linear regions)**.** *Consider a rank-K maxout network, $K \geq 2$, with n_0 inputs, n_1 units in the first layer, and any number of additional nonzero width layers. Then, for almost every choice of the parameters, the number of linear regions is at least $\sum_{j=0}^{n_0} \binom{n_1}{j}$ and the number of bounded linear regions is at least $\binom{n_1-1}{n_0}$.*

First we observe that for generic parameters, the number of linear regions of the function represented by a network is bounded below by the number of linear regions of the network restricted to the first layer. This is not trivial, since the deeper layers could in principle map the values from the first layer to a constant value, resulting in a function with a single linear region. However, for maxout networks this only happens for a null set of parameters.

Proposition 3.23. *The number of activation regions of a maxout network is at least as large as the number of regions of the first layer. Moreover, for generic parameters the number of linear regions is equal to the number of activation regions.*

Proof. The number of regions never reduces as we pass through the network. The region is either kept as it is or split into parts by a neuron. The fact that for generic parameters activation regions correspond to linear regions is Lemma 3.5. □

In order to lower bound the number of regions of a single layer, we use the correspondence between linear regions and the upper vertices of the corresponding lifted Newton polytope, Proposition 3.20. We first observe that the Newton polytope of a shallow maxout units is the Minkowski sum of the Newton polytopes of the individual units. Recall that the Minkowski sum of two sets A and B is the set $A + B = \{a + b \colon a \in A, b \in B\}$.

Proposition 3.24. *Consider a layer of maxout units, $f \colon \mathbb{R}^n \to \mathbb{R}^m;\ f_i(x) = \max\{w_{ir} \cdot x + b_{ir} \colon r = 1, \dots, k\}$. Let $f(x) = \sum_{i=1}^m f_i(x)$. Then the lifted Newton polytope of f is the Minkowski sum of the lifted Newton polytopes of $f_1, \dots, f_m$, $P_f = \sum_{i=1}^m P_{f_i}$.*

Proof. This follows from direct calculation. Details can be found in the works of Zhang et al. (2018) and Montúfar et al. (2022). □

Next, a family of polytopes $P_i = \operatorname{conv}\{(w_{i,r}, b_{i,r}) \in \mathbb{R}^{n_0+1} \colon r = 1, \dots, K\}$ with generic $(w_{i,r}, b_{i,r}), r = 1, \dots, K, i = 1, \dots, n_1$, is in general orientation. For such a family, the Minkowski sum $P = P_1 + \cdots + P_{n_1}$ has at least as many vertices as a Minkowski sum of n_1 line segments in general orientation:

Proposition 3.25 (Adiprasito 2017, Corollary 8.2)**.** *The number of vertices of a Minkowski sum of m polytopes in general orientation is lower bounded by the number of vertices of a sum of m line segments in general orientations.*

From this, we derive a lower bound on the number of upper vertices of a Minkowski sum of polytopes in general orientations.

Proposition 3.26. *The number of upper vertices of a Minkowski sum of n_1 polytopes in $\mathbb{R}^{n_0+1}$ in general orientation is at least $\sum_{j=0}^{n_0} \binom{n_1}{j}$, and the number of strict upper vertices is at least $\binom{n_1-1}{n_0}$.*

Proof. Consider the sum $P = P_1 + \cdots + P_{n_1}$ of polytopes $P_i = \{(w_{i,r}, b_{i,r})\colon r = 1, \ldots, k\}$, $i = 1, \ldots, n_1$. The set of upper vertices consists of 1) strict upper vertices and 2) vertices which are both upper and lower. The number of strict upper vertices of a Minkowski sum of n_1 positive dimensional polytopes in general orientations in $\mathbb{R}^{n_0+1}$ is at least $\binom{n_1-1}{n_0}$ (Montúfar et al., 2022, Corollary 3.8).

Now note that the vertices which are upper and lower are precisely the vertices of the Minkowski sum $Q = Q_1 + \cdots + Q_{n_1}$ of the poltyopes $Q_i = \operatorname{conv}\{w_{i,r} \in \mathbb{R}^{n_0}\colon r = 1, \ldots, k\}, i = 1, \ldots, n_1$. By Proposition 3.25 the total number of vertices of a Minkowski sum is at least equal to the number of vertices of a Minkowski sum of line segments. The latter is the same as the number of regions of a central hyperplane arrangement in n_0 dimensions, which is $\binom{n_1-1}{n_0-1} + \sum_{j=0}^{n_0-1} \binom{n_1}{j}$.

Hence for any generic Minkowski sum of n_1 positive-dimensional polytopes in $n_0 + 1$ dimensions, the number of upper vertices is at least

$$\binom{n_1-1}{n_0} + \binom{n_1-1}{n_0-1} + \sum_{j=0}^{n_0-1} \binom{n_1}{j} = \binom{n_1}{n_0} + \sum_{j=0}^{n_0-1} \binom{n_1}{j} = \sum_{j=0}^{n_0} \binom{n_1}{j}.$$

This concludes the proof. □

Now we have all tools we need to prove the theorem.

Proof of Theorem 3.8. By Proposition 3.23, the number of regions is lower bounded by the number of regions of the first layer. We now derive a lower bound for the number of regions of a single layer with n_0 inputs and n_1 maxout units. In view of Propositions 3.20 and 3.24, we need to lower bound the number of upper vertices of a generic Minkowksi sum. The bounded regions correspond to the strict upper vertices. The result follows from Proposition 3.26. □

Remark 3.27. The statement of Theorem 3.8 does not apply to ReLU networks unless they have a single layer of ReLUs. Indeed, for a network with 2 layers of ReLUs there exists a positive measure subset of parameters for which the represented functions have only 1 linear region. To see this, consider a ReLU network with pre-activation features of the units in the second layer always being non-positive. A subset of parameters required to achieve this is defined as a solution to a set of inequalities (for instance, when the input weights and biases of the second layer are non-positive) and has a positive measure. For such pre-activation features, the ReLUs in the second layer always output 0 and there is a single linear region for the network.

3.C Expected number of activation regions of a single maxout unit

We discuss a single maxout unit with n inputs. In this case, the r-partial activation patterns correspond to the r-dimensional upper faces of a polytope given as the convex hull of the points $(w_k, b_k) \in \mathbb{R}^{n+1}$, $k = 1, \dots, K$. The statistics of faces of random polytopes have been studied in the literature (Hug et al., 2004; Hug and Reitzner, 2005; Bárány and Vu, 2007). We will use the following result.

Theorem 3.28 (Hug et al. 2004, Theorem 1.1)**.** *If $v_1, \dots, v_K$ are sampled iid according to the standard normal distribution in $\mathbb{R}^d$, then, the number of s-faces of the convex hull $P_K = \mathrm{conv}\{v_1, \dots, v_K\}$, denoted $f_s(P_K)$, has expected value*

$$\mathbb{E} f_s(P_K) \sim \bar{c}(s,d)(\log K)^{\frac{d-1}{2}}, \tag{3.2}$$

and the union s-faces of P_K, denoted $\mathrm{skel}_s(P_K)$, has expected volume

$$\mathbb{E}\,\mathrm{vol}_s(\mathrm{skel}_s(P_K)) \sim c(s,d)(\log K)^{\frac{d-1}{2}}, \tag{3.3}$$

where $\bar{c}(s,d)$ and $c(s,d)$ are constants depending only on s and d.

Based on this, we obtain the following upper bound for the expected number of linear regions of a maxout unit with iid Gaussian weights and biases.

Proposition 3.29 (Expected number of regions of a large-rank Gaussian maxout unit)**.** *Consider a rank-K maxout unit with n_0 inputs. If the weights and biases are sampled iid from a standard normal distribution, then for large K the expected number of non-empty r-partial activation regions satisfies*

$$\mathbb{E}[\#\, r\text{-partial activation regions}] \leq \bar{c}(r, n_0)(\log K)^{\frac{n_0}{2}}.$$

where $\bar{c}(r, n_0)$ is a constants depending solely on r and n_0.

Proof of Proposition 3.29. We use the correspondence between r-partial activation regions and the upper r-dimensional faces of the lifted Newton polytope (Proposition 3.20). The total number of s-dimensional faces of a polytope is an upper bound on the number of upper s-dimensional faces. Now we just apply Theorem 3.28. □

We can use the above result to upper bound the expectation value of the number of regions of a maxout network with iid Gaussian weights and biases. In particular, for a shallow maxout network we have the following.

Proposition 3.30 (Expected number of linear regions of a large-rank Gaussian maxout layer)**.** *Consider network $\mathcal{N}$ with n_0 inputs and a single layer of n_1 rank-K maxout units. Suppose the weights and biases are sampled iid from a standard normal distribution. Then, for sufficiently large K, the expected number of linear regions is bounded as*

$$\mathbb{E}[\#\text{ linear regions}] \leq \sum_{j=0}^{n_0} \binom{n_1}{j} (\bar{c}(n_0)(\log K)^{\frac{n_0}{2}} - 1)^j,$$

where $\bar{c}(n_0)$ is a constant depending solely on n_0. This upper bound behaves as $O(n_1^{n_0}(\log K)^{\frac{1}{2}n_0^2})$ in n_1 and K.

Proof. By Montúfar et al. (2022, Theorem 3.6), the maximum number of regions of a layer with n_0 inputs and n_1 maxout units of ranks $k_1, \dots, k_{n_1}$ is

$$\max[\#\text{ linear regions}] = \sum_{j=0}^{n_0} \sum_{S \in \binom{[n_1]}{j}} \prod_{i \in S} (k_i - 1).$$

Consider now our network with n_1 maxout units of rank K. For a given probability distribution over the parameter space, denote $\Pr(k_1, \dots, k_{n_1})$ the probability of the event that the i-th unit has k_i linear regions, $i = 1, \dots, n_1$. If the parameters of the different units are independent, we have

$$\begin{aligned}\mathbb{E}[\#\text{ linear regions}] &\leq \sum_{1 \leq k_1, \dots, k_{n_1} \leq K} \Pr(k_1, \dots, k_{n_1}) \sum_{j=0}^{n_0} \sum_{S \in \binom{[n_1]}{j}} \prod_{i \in S} (k_i - 1) \\ &= \sum_{j=0}^{n_0} \sum_{S \in \binom{[n_1]}{j}} \prod_{i \in S} (\mathbb{E}[k_i] - 1).\end{aligned}$$

If the weights and biases of each unit are iid normal, Proposition 3.29 allows us to upper bound the latter expression by

$$\leq \sum_{j=0}^{n_0} \binom{n_1}{j} (\bar{c}(n_0)(\log K)^{\frac{n_0}{2}} - 1)^j.$$

This concludes the proof. □

3.D Proofs related to the expected volume

The following is a maxout version of a result obtained by Hanin and Rolnick (2019a, Theorem 6) for the case of networks with single-argument piecewise linear activation functions.

Lemma 3.31 (Upper bound on the expected volume of $\mathcal{X}_{\mathcal{N},r}$). *Consider a rank-$K$ maxout network $\mathcal{N}$ with input dimension n_0, output dimension 1, and random weights and biases satisfying:*

1. *The distribution of all weights has a density with respect to the Lebesgue measure.*
2. *Every collection of biases has a conditional density with respect to Lebesgue measure given the values of all weights and other biases.*

Then, for any bounded measurable set $S \subset \mathbb{R}^{n_0}$ and any $r \in \{1, \dots, n_0\}$, the expectation value of the $(n_0 - r)$-dimensional volume of $\mathcal{X}_{\mathcal{N},r}$ inside S is upper bounded as

$$\begin{aligned}&\mathbb{E}[\mathrm{vol}_{n_0-r}(\mathcal{X}_{\mathcal{N},r} \cap S)] \\ &\leq \sum_{J\in\mathcal{S}_r} \int_S \mathbb{E}\left[\rho_{\boldsymbol{b}^r}((\boldsymbol{w}^m - \boldsymbol{w}^r)\cdot \boldsymbol{x}_{-1}^m + \boldsymbol{b}^m)\ \|\boldsymbol{J}((\boldsymbol{w}^m - \boldsymbol{w}^r)\cdot \boldsymbol{x}_{-1}^m + \boldsymbol{b}^m)\|\right]\ dx,\end{aligned}$$

where, for any given r-partial activation sub-pattern $J = (J_z)_{z\in\mathcal{Z}} \in \mathcal{S}_r$, for any given J_z we denote its smallest element by j_0, we let $\rho_{\boldsymbol{b}^r}$ denote the joint conditional density of the biases of pre-activation features $j \in J_z \setminus \{j_0\}$ of the neurons $z \in \mathcal{Z}$, given all other network parameters, we let $g\colon \mathbb{R}^{n_0} \to \mathbb{R}^r$; $x \mapsto (\boldsymbol{w}^m - \boldsymbol{w}^r)\cdot \boldsymbol{x}_{-1}^m + \boldsymbol{b}^m := ((w_{z,j_0} - w_{z,j})\cdot x_{l(z)-1} + b_{z,j_0})_{z\in\mathcal{Z}, j\in J_z\setminus\{j_0\}} \in \mathbb{R}^r$, denote $\boldsymbol{J}g$ the $r \times n_0$ Jacobian of g, and $\|\boldsymbol{J}g(x)\| = \det\big((\boldsymbol{J}g(x))(\boldsymbol{J}g(x))^\top\big)^{\frac{1}{2}}$, and the inner expectation is with respect to all parameters aside these biases.

Proof of Lemma 3.31. The proof follows the arguments of Hanin and Rolnick (2019a, Theorem 6). The main difference is that maxout units are generically active and the activation regions of maxout units may involve several pre-activation features and additional inequalities. To obtain the upper bound, we will discard certain inequalities, and separate one distinguished pre-activation feature j_0 for each neuron participating in a sub-pattern, which allows us to relate inputs in the corresponding activation regions to bias values and apply the co-area formula.

Recall that an r-partial activation sub-pattern $J \in \mathcal{S}_r$ is a list of patterns $J_z \subseteq [K]$ of cardinality at least 2 for a collection of participating neurons $z \in \mathcal{Z}$, with $\sum_{z\in\mathcal{Z}}(|J_z| - 1) = r$. Further, for any given J_z we denote j_0 its smallest element. When discussing a particular sub-pattern, we will write $m = |\mathcal{Z}|$ for the number of participating neurons. Finally, recall that $\mathcal{H}(J,\theta) = \bigcap_{z\in\mathcal{Z}} \mathcal{H}(J_z,\theta)$.

By Corollary 3.17, with probability 1 with respect to θ,

$$\mathrm{vol}_{n_0-r}(\mathcal{X}_{\mathcal{N},r}(\theta)) = \sum_{J\in\mathcal{S}_r} \mathrm{vol}_{n_0-r}(\mathcal{H}(J,\theta)).$$

Fix $J \in \mathcal{S}_r$. In the following, we prove that

$$\begin{aligned}&\mathbb{E}[\mathrm{vol}_{n_0-r}(\mathcal{H}(J,\theta)\cap S)] \\ &\leq \int_S \mathbb{E}\left[\rho_{\boldsymbol{b}^r}((\boldsymbol{w}^m-\boldsymbol{w}^r)\cdot\boldsymbol{x}_{-1}^m+\boldsymbol{b}^m)\ \|\boldsymbol{J}((\boldsymbol{w}^m-\boldsymbol{w}^r)\cdot\boldsymbol{x}_{-1}^m+\boldsymbol{b}^m)\|\right]\ dx.\end{aligned}$$

We first note that

$$\mathrm{vol}_{n_0-r}(\mathcal{H}(J,\theta)\cap S) = \int\limits_{\mathcal{H}(J,\theta)\cap S} d\,\mathrm{vol}_{n_0-r}(x). \tag{3.4}$$

For each $z \in \mathcal{Z}$ and J_z we can pick an element $j_0 \in J_z$ and express $\mathcal{H}(J_z,\theta)$ in terms of $(|J_z|-1)$ equations and $(K-|J_z|)$ inequalities (not necessarily linear),

$$\begin{aligned}\mathcal{H}(J_z,\theta) = \{x\in\mathbb{R}^{n_0} \mid w_{z,j_0}\cdot x_{l(z)-1}+b_{z,j_0} &= w_{z,j}\cdot x_{l(z)-1}+b_{z,j}, \quad \forall j\in J_z\setminus\{j_0\}; \\ w_{z,j_0}\cdot x_{l(z)-1}+b_{z,j_0} &> w_{z,i}\cdot x_{l(z)-1}+b_{z,i}, \quad \forall i\in[K]\setminus J_z\}.\end{aligned} \tag{3.5}$$

Here, $x_{l(z)-1}$ are the activation values of the units in the layer preceding unit z, depending on the input x. Since $\sum_z(|J_z|-1)=r$, the set $\mathcal{H}(J,\theta)$ is defined by r equations (in addition to inequalities). We will denote with $\boldsymbol{b}^r \in \mathbb{R}^r$ the vector of biases $b_{z,j}$ that are involved in these r equations, with subscripts (z,j) with $j\in J_z\setminus\{j_0\}$ and $z\in\mathcal{Z}$.

We take the expectation of (3.4) with respect to the conditional distribution of $\boldsymbol{b}^r$ given the values of all the other network parameters. We have assumed that this has a density. Denoting the conditional density of $\boldsymbol{b}^r$ by $\rho_{\boldsymbol{b}^r}$, this is given by

$$\int\limits_{\mathbb{R}^r}\ \int\limits_{\mathcal{H}(J,\theta)\cap S} d\,\mathrm{vol}_{n_0-r}(x)\rho_{\boldsymbol{b}^r}(\boldsymbol{b}^r)d\boldsymbol{b}^r. \tag{3.6}$$

The equations in (3.5) imply that $b_{z,j}=(w_{z,j_0}-w_{z,j})\cdot x_{l(z)-1}+b_{z,j_0}$ for any $x\in\mathcal{H}(J,\theta)$. We write all these equations concisely as $\boldsymbol{b}^r=(\boldsymbol{w}^m-\boldsymbol{w}^r)\cdot\boldsymbol{x}_{-1}^m+\boldsymbol{b}^m$. Then (3.6) becomes

$$\int\limits_{\mathbb{R}^r}\ \int\limits_{\mathcal{H}(J,\theta)\cap S} \rho_{\boldsymbol{b}^r}((\boldsymbol{w}^m-\boldsymbol{w}^r)\cdot\boldsymbol{x}_{-1}^m+\boldsymbol{b}^m)\ d\,\mathrm{vol}_{n_0-r}(x)d\boldsymbol{b}^r. \tag{3.7}$$

We will upper bound the volume of $\mathcal{H}(J,\theta)$ by the volume of the corresponding set without the inequalities,

$$\mathcal{H}'(J,\theta) := \bigcap_{z\in\mathcal{Z}}\left\{x\in\mathbb{R}^{n_0} \mid w_{z,j_0}\cdot x_{l(z)-1}+b_{z,j_0}=w_{z,j}\cdot x_{l(z)-1}+b_{z,j}, \quad \forall j\in J_z\setminus\{j_0\}\right\}.$$

Since $\mathcal{H}(J,\theta) \subseteq \mathcal{H}'(J,\theta)$, we can upper bound (3.7) by

$$\int\limits_{\mathbb{R}^r} \int\limits_{\mathcal{H}'(J,\theta)\cap S} \rho_{\boldsymbol{b}^r}\left((\boldsymbol{w}^m - \boldsymbol{w}^r)\cdot \boldsymbol{x}_{-1}^m + \boldsymbol{b}^m\right) \; d\,\mathrm{vol}_{n_0-r}(x) d\boldsymbol{b}^r. \tag{3.8}$$

Now we will use the co-area formula to express (3.8) as an integral over S alone. Recall that the co-area formula says that if $\psi \in L^1(\mathbb{R}^n)$ and $g : \mathbb{R}^n \to \mathbb{R}^r$ with $r \le n$ is Lipschitz, then

$$\int_{\mathbb{R}^r} \int_{g^{-1}(t)} \psi(x) d\,\mathrm{vol}_{n-r}(x) dt = \int_{\mathbb{R}^n} \psi(x) \|\boldsymbol{J}g(x)\| d\,\mathrm{vol}_n(x),$$

where $\boldsymbol{J}g$ is the $r \times n$ Jacobian of g and $\|\boldsymbol{J}g(x)\| = \det((\boldsymbol{J}g(x))(\boldsymbol{J}g(x))^\top)^{\frac{1}{2}}$.

In our case $r = r, n = n_0$, which satisfy $r \le n_0$. Further, $\boldsymbol{b}^r \in \mathbb{R}^r$ plays the role of $t \in \mathbb{R}^r$, and $\mathbb{R}^{n_0} \to \mathbb{R}^r; x \mapsto \rho_{\boldsymbol{b}^r}((\boldsymbol{w}^m - \boldsymbol{w}^r)\cdot \boldsymbol{x}_{-1}^m + \boldsymbol{b}^m)$ plays the role of ψ. Since $(\boldsymbol{w}^m - \boldsymbol{w}^r)\cdot \boldsymbol{x}_{-1}^m + \boldsymbol{b}^m$ is continuous and S is bounded, assuming $\rho_{\boldsymbol{b}^r}$ is continuous, this is in $L^1(S)$. Finally, we set $g\colon S \to \mathbb{R}^r; x \mapsto ((\boldsymbol{w}^m - \boldsymbol{w}^r)\cdot \boldsymbol{x}_{-1}^m + \boldsymbol{b}^m)$, which is Lipschitz.

Hence (3.8) can be expressed as

$$\int\limits_S \rho_{\boldsymbol{b}^r}\left((\boldsymbol{w}^m - \boldsymbol{w}^r)\cdot \boldsymbol{x}_{-1}^m + \boldsymbol{b}^m\right) \; \|\boldsymbol{J}((\boldsymbol{w}^m - \boldsymbol{w}^r)\cdot \boldsymbol{x}_{-1}^m + \boldsymbol{b}^m)\| \; dx.$$

Taking expectation with respect to all other weights and biases, and interchanging the integral over S with the expectation (according to Fubini's theorem, since the integral is non-negative),

$$\int\limits_S \mathbb{E}\left[\rho_{\boldsymbol{b}^r}\left((\boldsymbol{w}^m - \boldsymbol{w}^r)\cdot \boldsymbol{x}_{-1}^m + \boldsymbol{b}^m\right) \; \|\boldsymbol{J}((\boldsymbol{w}^m - \boldsymbol{w}^r)\cdot \boldsymbol{x}_{-1}^m + \boldsymbol{b}^m)\|\right] \; dx.$$

Summing over all r-partial activation sub-patterns $J \in \mathcal{S}_r$ gives the desired result. ☐

Based on the preceding Lemma 3.31, now we derive a more explicit upper bound expressed in terms of properties of the probability distribution of the network parameters.

Theorem 3.10 (Upper bound on the expected volume of the non-linear locus). *Consider a bounded measurable set $S \subset \mathbb{R}^{n_0}$ and the settings of Theorem 3.9 with constants C_{grad} and C_{bias} evaluated over S. Then, for any $r \in \{1, \dots, n_0\}$,*

$$\frac{\mathbb{E}[\mathrm{vol}_{n_0-r}(\mathcal{X}_{\mathcal{N},r} \cap S)]}{\mathrm{vol}_{n_0}(S)} \le (2C_{\text{grad}}C_{\text{bias}})^r \binom{rK}{2r}\binom{N}{r}.$$

Proof of Theorem 3.10. By Lemma 3.31, $\mathbb{E}\left[\mathrm{vol}_{n_0-r}(\mathcal{X}_{\mathcal{N},r} \cap S)\right]$ is upper bounded by

$$\sum_{J\in\mathcal{S}_r} \int\limits_S \mathbb{E}\left[\rho_{\boldsymbol{b}^r}\left((\boldsymbol{w}^m - \boldsymbol{w}^r)\cdot \boldsymbol{x}_{-1}^m + \boldsymbol{b}^m\right) \; \|\boldsymbol{J}((\boldsymbol{w}^m - \boldsymbol{w}^r)\cdot \boldsymbol{x}_{-1}^m + \boldsymbol{b}^m)\|\right] \; dx.$$

Since we have assumed that for any collection of t biases the conditional density given all weights and the other biases can be upper-bounded with C^t_{bias}, we have $\rho_{\boldsymbol{b}^r}((\boldsymbol{w}^m - \boldsymbol{w}^r) \cdot \boldsymbol{x}^m_{-1} + \boldsymbol{b}^m) \leq C^r_{\text{bias}}$.

As for the term $\mathbb{E}[\|\boldsymbol{J}((\boldsymbol{w}^m - \boldsymbol{w}^r) \cdot \boldsymbol{x}^m_{-1} + \boldsymbol{b}^m)\|]$, note that

$$\begin{aligned}
&\|\boldsymbol{J}((\boldsymbol{w}^m - \boldsymbol{w}^r) \cdot \boldsymbol{x}^m_{-1} + \boldsymbol{b}^m)\| \\
&= \det\big(\boldsymbol{J}((\boldsymbol{w}^m - \boldsymbol{w}^r) \cdot \boldsymbol{x}^m_{-1} + \boldsymbol{b}^m)^T \boldsymbol{J}((\boldsymbol{w}^m - \boldsymbol{w}^r) \cdot \boldsymbol{x}^m_{-1} + \boldsymbol{b}^m)\big)^{1/2} \\
&= \det\big(\text{Gram}\big(\nabla((w_{z_1,j_0} - w_{z_1,j_1}) \cdot x_{l(z_1)-1} + b_{z_1,j_0}), \ldots, \\
&\qquad\qquad \nabla((w_{z_m,j_0} - w_{z_m,j_{r_m}}) \cdot x_{l(z_m)-1} + b_{z_m,j_0})\big)\big)^{1/2},
\end{aligned} \tag{3.9}$$

where for any $v_1, \ldots, v_r \in \mathbb{R}^n$, $(\text{Gram}(v_1, \ldots, v_r))_{i,j} = \langle v_i, v_j \rangle$ is the associated Gram matrix.

It is known that the Gram determinant can also be expressed in terms of the exterior product of vectors, meaning that (3.9) can be written as

$$\|\nabla((w_{z_1,j_0} - w_{z_1,j_1}) \cdot x_{l(z_1)-1} + b_{z_1,j_0}) \wedge \cdots \wedge \nabla((w_{z_m,j_0} - w_{z_m,j_{r_m}}) \cdot x_{l(z_m)-1} + b_{z_m,j_0})\|,$$

which is the the r-dimensional volume of the parallelepiped in $\mathbb{R}^{n_0}$ spanned by r elements. Therefore, for $J \in \mathcal{S}_r$ with participating neurons Z, we can upper bound this expression by (see Gover and Krikorian, 2010)

$$\begin{aligned}
&\prod_{z \in Z} \prod_{j \in J_z \setminus \{j_0\}} \|\nabla((w_{z,j_0} - w_{z_i,j}) \cdot x_{l(z)-1} + b_{z,j_0})\| \\
&\leq \prod_{z \in Z} \prod_{j \in J_z \setminus \{j_0\}} 2 \max\left\{\|\nabla(w_{z,j_0} \cdot x_{l(z)-1})\|, \|\nabla(w_{z,j} \cdot x_{l(z)-1})\|\right\} \\
&\leq 2^r \max_{z \in Z, j \in J_z} \left\{\|\nabla(w_{z,j} \cdot x_{l(z)-1})\|\right\}^r.
\end{aligned}$$

In the second line we use the triangle inequality. Considering the assumption that we have made on the gradients, for the expectation we obtain the upper bound $(2C_{\text{grad}})^r$.

By Lemma 3.6, we can upper-bound the number of entries of the sum $\sum_{J \in \mathcal{S}_r}$ with $\binom{rK}{2r}\binom{N}{r}$. Combining everything, we get the final upper bound

$$(2C_{\text{grad}} C_{\text{bias}})^r \binom{rK}{2r} \binom{N}{r} \operatorname{vol}_{n_0}(S).$$

This concludes the proof. $\square$

3.E Proofs related to the expected number of regions

Theorem 3.9 (Upper bound on the expected number of partial activation regions)**.** *Let $\mathcal{N}$ be a fully-connected feed-forward maxout network with n_0 inputs and a total of N rank K maxout units. Suppose we have a probability distribution over the parameters so that:*

1. *The distribution of all weights has a density with respect to the Lebesgue measure on $\mathbb{R}^{\#\text{weights}}$.*
2. *Every collection of biases has a conditional density with respect to Lebesgue measure given the values of all other weights and biases.*
3. *There exists $C_{\text{grad}} > 0$ so that for any $t \in \mathbb{N}$ and any pre-activation feature $\zeta_{z,k}$,*
$$\sup_{x \in \mathbb{R}^{n_0}} \mathbb{E}[\|\nabla \zeta_{z,k}(x)\|^t] \leq C_{\text{grad}}^t.$$
4. *There exists $C_{\text{bias}} > 0$ so that for any pre-activation features $\zeta_1, \ldots, \zeta_t$ from any neurons, the conditional density of their biases $\rho_{b_1,\ldots,b_t}$ given all the other weights and biases satisfies*
$$\sup_{b_1,\ldots,b_t \in \mathbb{R}} \rho_{b_1,\ldots,b_t}(b_1, \ldots, b_t) \leq C_{\text{bias}}^t.$$

Fix $r \in \{0, \ldots, n_0\}$ and let $T = 2^5 C_{\text{grad}} C_{\text{bias}}$. Then, there exists $\delta_0 \leq 1/(2C_{\text{grad}} C_{\text{bias}})$ such that for all cubes $C \subseteq \mathbb{R}^{n_0}$ with side length $\delta > \delta_0$ we have

$$\frac{\mathbb{E}[\#\, r\text{-partial activation regions of } \mathcal{N} \text{ in } C]}{\operatorname{vol}(C)} \leq \begin{cases} \binom{rK}{2r}\binom{N}{r} K^{N-r}, & N \leq n_0 \\ \frac{(TKN)^{n_0}\binom{n_0 K}{2n_0}}{(2K)^r n_0!}, & N \geq n_0 \end{cases}.$$

Here the expectation is taken with respect to the distribution of weights and biases in $\mathcal{N}$. Of particular interest is the case $r = 0$, which corresponds to the number of linear regions.

Proof of Theorem 3.9. The proof follows closely the arguments of Hanin and Rolnick (2019b, Proof of Theorem 10), whereby we use appropriate supporting results for maxout networks and need to accommodate the combinatorics depending on K. Fix a network $\mathcal{N}$ with rank-K maxout units, input dimension n_0 and output dimension 1. Let $0 \leq r \leq n_0$. For $N \leq n_0$, the statement follows direction from the simple upper bound on the number of distinct r-partial activation patterns given in Lemma 3.6.

Consider now the case $N \geq n_0$. Fix a closed cube $C \subseteq \mathbb{R}^{n_0}$ of sidelength $\delta > 0$. For any $t \in \{0, \ldots, n_0\}$ let

$$C_t := t\text{-skeleton of } C$$

denote the union of t-dimensional faces of C. For example, C_0 is the set of 2^{n_0} vertices of C, C_{n_0-1} is the set of $2n_0$ facets of C, and C_{n_0} is C. In general, C_t consists of $\binom{n_0}{t}2^{n_0-t}$ faces of dimension t, each with t-volume δ^t. Hence,

$$\mathrm{vol}_t(C_t) = \binom{n_0}{t} 2^{n_0-t}\delta^t. \tag{3.10}$$

For any choice of θ let

$$\mathcal{V}_t(\theta) := \mathcal{X}_{\mathcal{N},t}(\theta) \cap C_t.$$

By Lemma 3.33 below, for any t and almost every choice of θ, the set $\mathcal{V}_t(\theta)$ is a finite set of points. For each $t \in \{0, \dots, n_0\}$, we also define

$$C_{t,\varepsilon} := \{x \in \mathbb{R}^{n_0} \mid \mathrm{dist}(x, C_t) \leq \varepsilon\},$$

the ε-thickening of C_t. For almost every θ, Lemma 3.34 ensures the existence of an $\varepsilon > 0$ such that for all $v \in \mathcal{V}_t(\theta)$, the radius-$\varepsilon$ balls $B_\varepsilon(v)$ are contained in $C_{t,\varepsilon}$ and are disjoint. Hence, writing ω_{n_0-t} for the (n_0-t)-volume of the (n_0-t)-dimensional ball with unit radius,

$$\mathrm{vol}_{n_0-t}(X_{\mathcal{N},t} \cap C_{t,\varepsilon}) \geq \sum_{v \in \mathcal{V}_t} \varepsilon^{n_0-t}\omega_{n_0-t} = \#\mathcal{V}_t \cdot \varepsilon^{n_0-t}\omega_{n_0-t}.$$

Therefore, for all but a measure 0 set of $\theta \in \mathbb{R}^{\#\text{params}}$, there exists $\varepsilon > 0$ so that

$$\frac{\mathrm{vol}_{n_0-t}(X_{\mathcal{N},t} \cap C_{t,\varepsilon})}{\varepsilon^{n_0-t}\omega_{n_0-t}} \geq \#\mathcal{V}_t.$$

Thus taking the limit $\varepsilon \to 0$ and taking expectation with respect to the parameter θ, and applying Fatou's lemma to upper bound the result by the expression with exchanged limit and expectation,

$$\mathbb{E}\left[\#\mathcal{V}_t\right] \leq \mathbb{E}\left[\lim_{\varepsilon\to 0} \frac{\mathrm{vol}_{n_0-t}(\mathcal{X}_{\mathcal{N},t} \cap C_{t,\varepsilon})}{\varepsilon^{n_0-t}\omega_{n_0-t}}\right] \leq \lim_{\varepsilon\to 0} \mathbb{E}\left[\frac{\mathrm{vol}_{n_0-t}(\mathcal{X}_{\mathcal{N},t} \cap C_{t,\varepsilon})}{\varepsilon^{n_0-t}\omega_{n_0-t}}\right].$$

Then,

$$\begin{aligned}\mathbb{E}\left[\#\mathcal{V}_t\right] &\leq \lim_{\varepsilon\to 0} \mathbb{E}\left[\frac{\mathrm{vol}_{n_0-t}(\mathcal{X}_{\mathcal{N},t} \cap C_{t,\varepsilon})}{\mathrm{vol}_{n_0}(C_{t,\varepsilon})} \cdot \frac{\mathrm{vol}_{n_0}(C_{t,\varepsilon})}{\varepsilon^{n_0-t}\omega_{n_0-t}}\right] \\ &= \lim_{\varepsilon\to 0} \mathbb{E}\left[\frac{\mathrm{vol}_{n_0-t}(\mathcal{X}_{\mathcal{N},t} \cap C_{t,\varepsilon})}{\mathrm{vol}_{n_0}(C_{t,\varepsilon})}\right] \cdot \lim_{\varepsilon\to 0} \frac{\mathrm{vol}_{n_0}(C_{t,\varepsilon})}{\varepsilon^{n_0-t}\omega_{n_0-t}} \\ &\leq (2C_{\text{grad}}C_{\text{bias}})^t \binom{tK}{2t}\binom{N}{t} \mathrm{vol}_t(C_t).\end{aligned}$$

To obtain the last line, the first term is upper bounded using Theorem 3.10, and the second term is evaluated using

$$\lim_{\varepsilon\to 0}\frac{\mathrm{vol}_{n_0}(C_{t,\varepsilon})}{\varepsilon^{n_0-t}\omega_{n_0-t}}=\mathrm{vol}_t(C_t).$$

Combining this with Lemma 3.33 and the formula (3.10) for $\mathrm{vol}_t(C_t)$, we find

$$\begin{aligned}&\mathbb{E}\left[\#\{r\text{-partial activation regions with }\mathcal{R}(J^r;\theta)\cap C\neq\emptyset\}\right]\\ &\leq \quad \sum_{t=r}^{n_0}\binom{t}{r}K^{t-r}(2C_{\text{grad}}C_{\text{bias}})^t\binom{tK}{2t}\binom{N}{t}\binom{n_0}{t}2^{n_0-t}\delta^t\\ &\overset{\delta\geq 1/(2C_{\text{grad}}C_{\text{bias}})}{\leq}\quad (2\delta C_{\text{grad}}C_{\text{bias}})^{n_0}\binom{n_0K}{2n_0}(2K)^{n_0-r}\sum_{t=r}^{n_0}\binom{t}{r}\binom{N}{t}\binom{n_0}{t}.\end{aligned}\tag{3.11}$$

The last line uses the assumption that $\delta\geq 1/(2C_{\text{grad}}C_{\text{bias}})$ and Lemma 3.32, which states that $\binom{tK}{2t}\leq\binom{nK}{2n}$ for $t\leq n$.

In the following we simplify (3.11). Note that $\binom{t}{r}\leq\sum_{r=0}^{t}\binom{t}{r}=2^t\leq 2^{n_0}$. Hence (3.11) can be upper bounded by

$$\begin{aligned}&(4\delta C_{\text{grad}}C_{\text{bias}})^{n_0}\binom{n_0K}{2n_0}(2K)^{n_0-r}\sum_{t=r}^{n_0}\binom{N}{t}\binom{n_0}{t}\\ &=(4\delta C_{\text{grad}}C_{\text{bias}})^{n_0}\binom{n_0K}{2n_0}(2K)^{n_0-r}\sum_{t=r}^{n_0}\binom{n_0}{t}^2\frac{\binom{N}{t}}{\binom{n_0}{t}}.\end{aligned}$$

Using $n_0\leq N$, observe that

$$\begin{aligned}\frac{\binom{N}{t}}{\binom{n_0}{t}}&=\frac{N!\cdot(n_0-t)!}{(N-t)!\cdot n_0!}\leq N^t\cdot\frac{(n_0-t)!}{n_0!}=\frac{N^{n_0}}{N^{n_0-t}}\cdot\frac{(n_0-t)!}{n_0!}=\frac{N^{n_0}}{n_0!}\cdot\frac{(n_0-t)!}{N^{n_0-t}}\\ &\leq\frac{N^{n_0}}{n_0!}\cdot\frac{(n_0-t)^{n_0-t}}{N^{n_0-t}}\leq\frac{N^{n_0}}{n_0!}.\end{aligned}$$

Also, using Vandermonde's identity, observe that

$$\sum_{t=0}^{n_0}\binom{n_0}{t}^2=\binom{2n_0}{n_0}\leq 4^{n_0}.$$

Combing everything, (3.11) is upper bounded by

$$(16\delta C_{\text{grad}}C_{\text{bias}})^{n_0}\binom{n_0K}{2n_0}(2K)^{n_0-r}\frac{N^{n_0}}{n_0!}=(32KC_{\text{grad}}C_{\text{bias}})^{n_0}\binom{n_0K}{2n_0}\frac{N^{n_0}}{(2K)^r n_0!}\,\mathrm{vol}(C).$$

Setting $T = 2^5 C_{\text{grad}} C_{\text{bias}}$, we get

$$\frac{(TKN)^{n_0}\binom{n_0 K}{2n_0}}{(2K)^r n_0!} \operatorname{vol}(C).$$

This concludes the proof. □

We state and prove lemmas used in the proof of Theorem 3.9.

Lemma 3.32. *For any* $t \leq n$, $\binom{tK}{2t} \leq \binom{nK}{2n}$.

Proof. To see this, note that $\binom{tK}{2t} \leq \binom{nK}{2n}$ is equivalent to the following:

$$\begin{aligned} \frac{(Kr)!}{(2r)!(Kr-2r)!} &\leq \frac{(Kn)!}{(2n)!(Kn-2n)!} \\ \frac{(2n)!}{(2r)!}\frac{(Kn-2n)!}{(Kr-2r)!} &\leq \frac{(Kn)!}{(Kr)!} \\ \prod_{i=1}^{2n-2r}(2r+i) \prod_{j=1}^{(K-2)n-(K-2)r}(Kr-2r+j) &\leq \prod_{k=1}^{Kn-Kr}(Kr+k). \end{aligned}$$

Since $\prod_{i=1}^{2n-2r}(2r+i) \leq \prod_{k=1}^{2n-2r}(Kr+k)$ and $\prod_{j=1}^{(K-2)n-(K-2)r}(Kr-2r+j) \leq \prod_{k=2n-2r+1}^{Kn-Kr}(Kr+k)$ the inequality holds. □

Lemma 3.33. *For almost every θ, for each $t \in \{0, \dots, n_0\}$, the set $\mathcal{V}_t(\theta) = \mathcal{X}_{\mathcal{N},t}(\theta) \cap C_t$ consists of finitely many points and*

$$\#\{r\text{-partial activation regions } \mathcal{R}(J^r, \theta) \text{ with } \mathcal{R}(J^r, \theta) \cap C \neq \emptyset\} \leq \sum_{t=r}^{n_0} \binom{t}{r} K^{t-r} \# \mathcal{V}_t(\theta), \quad (3.12)$$

where $\#\mathcal{V}_t(\theta)$ is the number of points in $\mathcal{V}_t(\theta)$.

Proof. The proof is similar to the proof of (Hanin and Rolnick, 2019b, Lemma 12). The difference lies in the types of equations that appear in the partial activation regions of maxout networks. The dimension of $\mathcal{V}_t(\theta)$ is 0 with probability 1, because the set C_t has dimension t and, by Lemma 3.16, with probability 1 the set $\mathcal{X}_{\mathcal{N},t}$ coincides locally with a subspace of codimension t. The intersection of two generic affine spaces of complementary dimension has dimension 0.

Now we prove (3.12). If J^r is an r-partial activation pattern and $\mathcal{R}(J^r, \theta) \cap C \neq \emptyset$, then the closure $\operatorname{cl} \mathcal{R}(J^r, \theta) \cap C$ is a non-empty polytope. The intersection is bounded because C is bounded, and, by Lemma 3.4, the closure of $\mathcal{R}(J^r, \theta)$ is a polyhedron. As a non-empty polytope, this set has at least one vertex. Generically, if a vertex is in an $(n_0 - t)$-face of $\operatorname{cl} \mathcal{R}(J^r, \theta)$, then it is in a t-face of C. Hence, with probability 1,

$$\mathcal{R}(J^r, \theta) \cap C \neq \emptyset \quad \Rightarrow \quad \exists\, t \in \{r, \dots, n_0\} \quad \text{s.t.} \quad \operatorname{cl} \mathcal{R}(J^r, \theta) \cap \mathcal{V}_t \neq \emptyset.$$

Thus, with probability 1,

$$\#\{r\text{-partial activation regions with } \mathcal{R}(J^r, \theta) \cap C \neq \emptyset\} \leq \sum_{t=r}^{n_0} T_t \# \mathcal{V}_t,$$

where T_t is the maximum over all $v \in \mathcal{V}_t$ of the number of r-partial activation regions whose closure contains v.

To complete the proof, it remains to check that, with probability 1,

$$T_t \leq \binom{t}{r} K^{t-r}.$$

By the definition of $\mathcal{X}_{\mathcal{N},t}$, each $v \in \mathcal{V}_t$ is an element of exactly one t-partial activation region defined by t equations. To upper bound the number of r-partial activation regions that contain v, we upper bound the number of ways in which one can get an r-partial region from this t-partial region. We have $\binom{t}{r}$ options to pick r equations that will remain satisfied. In each case, there are at most $t - r$ neurons for which we need to specify a pre-activation feature attaining the maximum, for a total of at most K^{t-r} options. This concludes the proof. □

Lemma 3.34. *Fix $t \in \{0, \dots n_0\}$. For almost every choice of θ, there exists $\varepsilon > 0$ (depending on θ) so that the balls $\mathcal{B}_\varepsilon(v)$ of radius ε centered at $v \in \mathcal{V}_t$ are disjoint and*

$$\mathrm{vol}_{n_0-t}(\mathcal{X}_{\mathcal{N},t} \cap \mathcal{B}_\varepsilon(v)) = \varepsilon^{n_0-t}\omega_{n_0-t},$$

where ω_t is the volume of a unit ball in $\mathbb{R}^t$.

Proof. The proof is similar to the proof of Hanin and Rolnick (2019b, Lemma 13), whereby we use Lemma 3.33 and the results for maxout networks obtained in Section 3.A. By Lemma 3.33, with probability 1 over θ, each $\mathcal{V}_t$ is a finite set of points. Hence, we may choose $\varepsilon > 0$ sufficiently small so that the balls $B_\varepsilon(v)$ are disjoint. Moreover, by Lemma 3.16, in a sufficiently small neighborhood of $v \in \mathcal{V}_t$, the set $\mathcal{X}_{\mathcal{N},t}$ coincides with a $(n_0 - t)$-dimensional subspace. The $(n_0 - t)$-dimensional volume of this subspace in $B_\varepsilon(v)$ is the volume of $(n_0 - t)$-dimensional ball of radius ε, which equals $\varepsilon^{n_0-t}\omega_{n_0-t}$, completing the proof. □

To conclude this section, we compare the results on the numbers of activation regions of maxout and ReLU networks in Table 3.2.

3.F Upper bounding the constants

We briefly discuss the constants C_{bias} and C_{bias} in the hypothesis of Theorem 3.9. The constant C_{bias} can be evaluated at initialization using the definition since we know the distribution of biases. Recall

Table 3.2: Comparison of the activation region results for maxout and ReLU networks.

	RELU NETWORK	MAXOUT NETWORK
Generic lower bound on the number of linear regions for a deep network	1, Remark 3.27	$\sum_{j=0}^{n_0}\binom{n_1}{j}$, Theorem 3.8
Trivial upper-bound on the number of r-partial activation regions	$\binom{N}{r}2^{N-r}$, (Hanin and Rolnick, 2019b, Theorem 10)	$\binom{rK}{2r}\binom{N}{r}K^{N-r}$, Lemma 3.6, see also Proposition 3.14
Upper-bound on the expected number of r-partial activation regions, $N \geq n_0$	$\frac{(TN)^{n_0}}{2^r n_0!}$, $T = 2^5 C_{\text{grad}} C_{\text{bias}}$, (Hanin and Rolnick, 2019b, Theorem 10)	$\frac{(TKN)^{n_0}\binom{n_0K}{2n_0}}{(2K)^r n_0!}$, $T = 2^5 C_{\text{grad}} C_{\text{bias}}$, Theorem 3.9
Upper bound on the expected $(n_0 - r)$-dimensional volume of the non-linear locus	$(2C_{\text{grad}}C_{\text{bias}})^r\binom{N}{r}$, (Hanin and Rolnick, 2019a, Corollary 7)	$(2C_{\text{grad}}C_{\text{bias}})^r\binom{rK}{2r}\binom{N}{r}$, Theorem 3.10

that we defined C_{bias} as an upper bound on

$$\left(\sup_{b_1,\ldots,b_t\in\mathbb{R}} \rho_{b_1,\ldots,b_t}(b_1,\ldots,b_t)\right)^{1/t},$$

where $\rho_{b_1,\ldots,b_t}$ is the conditional distribution of any collection of biases given all the other weights and biases in $\mathcal{N}$ and $t \in \mathbb{N}$. If the biases are sampled independently, independently of the weights, this equals $\sup_{b\in\mathbb{R}} \rho_b(b)$. Then, for instance, for a normal distribution with standard deviation $\sqrt{C/n_l}$, the constant C_{bias} can be chosen as

$$\max_{l\in\{0,\ldots,L-1\}} \sqrt{\frac{n_l}{2\pi C}}.$$

The constant C_{grad} was defined as an upper bound on

$$\left(\sup_{x\in\mathbb{R}^{n_0}} \mathbb{E}[\|\nabla\zeta_{z,k}(x)\|^t]\right)^{1/t}.$$

Therefore we need to upper-bound $\mathbb{E}\left[\|\nabla\zeta_{z,k}(x)\|^t\right]$. This expression stands for the t-th moment of the L2 norm of the gradient of a pre-activation feature $\zeta_{z,k}$ in a network, with respect to the input to the network.

One possible calculation is as follows. We consider $J_x = [\nabla_x\mathcal{N}_1(x;\theta),\ldots,\nabla_x\mathcal{N}_{n_L}(x;\theta)]^\top$ the Jacobian of the output vector with respect to the input, for a given parameter θ and input x. Note that the gradient $\nabla\zeta_{z,k}(x)$ for a pre-activation feature of a unit in the l-th layer of a network is a row in the Jacobian matrix of an l-layer network. Therefore, $\|\nabla\zeta_{z,k}(x)\|$ can be upper-bounded by the spectral norm $\|J_x\|$ of the Jacobian, and the moments of the Jacobian norm can be used as an upper-bound on the t-th moments of the gradient norm, $t \geq 1$.

Proposition 3.35 (Upper bound on the moments of the Jacobian matrix norm). *Let $\mathcal{N}$ be a fully-connected feed-forward network with maxout units of rank K and a linear last layer. Let the network have L layers of widths $n_1, \ldots, n_L$ and n_0 inputs. Assume that the weights and biases of the units in the l-th layer are sampled iid from a Gaussian distribution with mean 0 and variance c/n_{l-1}, $l = 1, \ldots, L$ and c is some constant $c \in \mathbb{R}, c > 0$. Then*

$$\mathbb{E}[\|J_x\|^t] \leq c^{t/2} n_0^{-t/2} \mathbb{E}[\chi_{n_L}^t] \prod_{l=1}^{L-1} \mathbb{E}\left[\left(\frac{c}{n_l} \sum_{i=1}^{n_l} m_{n_{l-1},i}^{(K)}\right)^{t/2}\right],$$

where J_x is the Jacobian as defined above, $x \in \mathbb{R}^{n_0}$; $t \geq 1, t \in \mathbb{N}$; $m_{n_{l-1},i}^{(K)}$ is the largest order statistic in a sample of size K of $\chi_{n_{l-1}}^2$ variables. Recall that the largest order statistic is a random variable defined as the maximum of a random sample and that a sum of squares of n independent Gaussian variables has a chi-squared distribution χ_n^2.

Proof. Our first goal will be to upper-bound $\|J_x\| = \sup_{\|u\|=1} \|J_x u\|$. The Jacobian J_x of $\mathcal{N}(x) \colon \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ can be written as a product of matrices $\overline{W}^{(l)}$, $l = 1, \ldots, L$ depending on the activation region of the input x. The matrix $\overline{W}^{(l)}$ consists of rows $\overline{W}_i^{(l)} = W_{i,k_i}^{(l)} \in \mathbb{R}^{n_{l-1}}$, where $k_i = \operatorname{argmax}_{k \in [K]} \{W_{i,k}^{(l)} x^{(l-1)} + b_{i,k}^{(l)}\}$ for $i = 1, \ldots, n_l$, and $x^{(l-1)}$ is the l-th layer's input. For the last layer, which is linear, we have $\overline{W}^{(L)} = W^{(L)}$. Thus for any given $u \in \mathbb{R}^{n_0}$ we have

$$\|J_x u\| = \|W^{(L)} \overline{W}^{(L-1)} \cdots \overline{W}^{(1)} u\|.$$

Consider some $u^{(0)}$ with $\|u^{(0)}\| = 1$ and assume $\|\overline{W}^{(1)} u^{(0)}\| \neq 0$. Note that for fixed $u^{(0)}$, the probability of $\overline{W}^{(1)}$ being such that $\|\overline{W}^{(1)} u^{(0)}\| = 0$ is 0. Multiplying and dividing by $\|\overline{W}^{(1)} u^{(0)}\|$ we get

$$\begin{aligned}
&\|W^{(L)} \overline{W}^{(L-1)} \cdots \overline{W}^{(1)} u^{(0)}\| \frac{\|\overline{W}^{(1)} u^{(0)}\|}{\|\overline{W}^{(1)} u^{(0)}\|} \\
&= \left\| W^{(L)} \overline{W}^{(L-1)} \cdots \overline{W}^{(2)} \frac{\overline{W}^{(1)} u^{(0)}}{\|\overline{W}^{(1)} u^{(0)}\|} \right\| \|\overline{W}^{(1)} u^{(0)}\| \\
&= \left\| W^{(L)} \overline{W}^{(L-1)} \cdots \overline{W}^{(2)} u^{(1)} \right\| \|\overline{W}^{(1)} u^{(0)}\|,
\end{aligned}$$

where $u^{(1)} = \frac{\overline{W}^{(1)} u^{(0)}}{\|\overline{W}^{(1)} u^{(0)}\|}$. Notice, $\|u^{(1)}\| = 1$. Repeating this procedure layer-by-layer, we get

$$\|W^{(L)} u^{(L-1)}\| \|\overline{W}^{(L-1)} u^{(L-2)}\| \cdots \|\overline{W}^{(2)} u^{(1)}\| \|\overline{W}^{(1)} u^{(0)}\|.$$

Now consider one of the factors, $\|\overline{W}^{(l)}u^{(l-1)}\|$. We have

$$\|\overline{W}^{(l)}u^{(l-1)}\|^2 = \sum_{i=1}^{n_l}\langle \overline{W}_i^{(l)}, u^{(l-1)}\rangle^2 \overset{\substack{\text{Cauchy–Schwarz}\\ \|u^{(l-1)}\|=1}}{\leq} \sum_{i=1}^{n_l}\|\overline{W}_i^{(l)}\|^2 \leq \sum_{i=1}^{n_l}\max_{k\in[K]}\left\{\|W_{i,k}^{(l)}\|^2\right\}.$$

Notice that this upper bound only depends on $W^{(l)}$ and is independent of all other weight matrices and of the input vector.

According to our assumptions, $W_{i,k}^{(l)} \overset{d}{=} \sqrt{\frac{c}{n_{l-1}}}v$, where v is a standard Gaussian random vector in $\mathbb{R}^{n_{l-1}}$. Therefore, $\|W_{i,k}^{(l)}\|^2 \overset{d}{=} \frac{c}{n_{l-1}}\chi^2_{n_{l-1}}$ has the distribution of a chi-squared random variable scaled by c/n_{l-1}. Moreover, since the vectors $W_{i,1}^{(l)}, \dots, W_{i,K}^{(l)}$ consist of the same number of separate iid entries, the variables $\|W_{i,1}^{(l)}\|^2, \dots, \|W_{i,K}^{(l)}\|^2$ are iid. In turn, $\max_{k\in[K]}\left\{\|W_{i,k}^{(l)}\|^2\right\} \overset{d}{=} \frac{c}{n_{l-1}}m^{(K)}_{n_{l-1},i}$, where $m^{(K)}_{n_{l-1},i}$ is the largest order statistic in a sample of size K of $\chi^2_{n_{l-1}}$ variables.

Notice that $\|W^{(L)}u^{(L-1)}\|^2 \overset{d}{=} \frac{c}{n_{L-1}}\chi^2_{n_L}$. To see this, recall that if u is a fixed vector and w is a Gaussian random vector with mean μ and covariance matrix Σ, then the product $u^\top w$ is Gaussian with mean $u^\top\mu$ and variance $u^\top\Sigma u$. Hence, since $W_i^{(L)}$ is a Gaussian vector with mean zero and covariance matrix $\Sigma = \frac{c}{n_{L-1}}I$, $W_i^{(L)}u^{(L-1)}$ is Gaussian with mean zero and variance $\frac{c}{n_{L-1}}\|u^{(L-1)}\|^2 = \frac{c}{n_{L-1}}$.

Combining everything, we get

$$\begin{aligned}\|J_x\| = \sup_{\|u\|=1}\|J_x u\| &\leq \left(\frac{c}{n_{L-1}}\chi^2_{n_L}\right)^{1/2}\left(\frac{c}{n_{L-2}}\sum_{i=1}^{n_{L-1}}m^{(K)}_{n_{L-2}}\right)^{1/2}\cdots\left(\frac{c}{n_0}\sum_{i=1}^{n_1}m^{(K)}_{n_0}\right)^{1/2}\\ =&c^{L/2}\chi_{n_L}\left(\prod_{l=0}^{L-1}n_l^{-1/2}\right)\prod_{l=1}^{L-1}\left(\sum_{i=1}^{n_l}m^{(K)}_{n_{l-1},i}\right)^{1/2}.\end{aligned}$$

Now using the monotonicity of the expectation, the moments of the right hand side upper-bound those of the left hand side. Moreover, using the independence of the individual factors, the expectation factorizes. For the t-th moment we get

$$\begin{aligned}\mathbb{E}[\|J_x\|^t] &\leq \mathbb{E}\left[c^{tL/2}\chi_{n_L}\left(\prod_{l=0}^{L-1}n_l^{-1/2}\right)\prod_{l=1}^{L-1}\left(\sum_{i=1}^{n_l}m^{(K)}_{n_{l-1},i}\right)^{t/2}\right]\\ &= c^{t/2}n_0^{-t/2}\mathbb{E}[\chi^t_{n_L}]\prod_{l=1}^{L-1}\mathbb{E}\left[\left(\frac{c}{n_l}\sum_{i=1}^{n_l}m^{(K)}_{n_{l-1},i}\right)^{t/2}\right].\end{aligned}$$

□

Corollary 3.36 (Upper bound on C_{grad})**.** *Under the same assumptions as in Proposition 3.35, assuming that c is set according to He initialization, meaning $c = 2$, or maxout-He initialization (see Table 3.1 for* *specific values of* c *for various* K*), the following expression can be used as the value for* C_{grad}*:*

$$\left(\frac{c}{n_0}\right)^{1/2}\left(n_L(n_L+t)^{\frac{t}{2}-1}\right)^{1/t}\prod_{l=1}^{L-1}\left(\mathbb{E}\left[\left(\frac{c}{n_l}\sum_{i=1}^{n_l}m_{n_{l-1},i}^{(K)}\right)^{t/2}\right]\right)^{1/t},$$

where $m_{n_{l-1},i}^{(K)}$ *is the largest order statistic in a sample of size* K *of* $\chi^2_{n_{l-1}}$ *variables.*

Proof. The constant C_{grad} was defined as an upper bound on

$$\left(\sup_{x\in\mathbb{R}^{n_0}}\mathbb{E}[\|\nabla\zeta_{z,k}(x)\|^t]\right)^{1/t}.$$

Therefore, using the upper-bound on the moments of the Jacobian norm from Proposition 3.35, an upper-bound on the following expression can be used as a value for C_{grad}:

$$c^{1/2}n_0^{-1/2}\left(\mathbb{E}[\chi_{n_L}^t]\right)^{1/t}\prod_{l=1}^{L-1}\left(\mathbb{E}\left[\left(\frac{c}{n_l}\sum_{i=1}^{n_l}m_{n_{l-1},i}^{(K)}\right)^{t/2}\right]\right)^{1/t}.$$

The moments of the chi distribution are

$$\mathbb{E}\left[\chi_{n_L}^t\right]=2^{t/2}\frac{\Gamma((n_L+t)/2)}{\Gamma(n_L/2)}.$$

Using an upper-bound on a Gamma function ratio (see Jameson, 2013, Equation 12), this can be upper-bounded with

$$n_L(n_L+t)^{\frac{t}{2}-1}.$$

The factor involving $m_{n_{l-1}}^{(K)}$ can be upper-bounded by considering the explicit expression for the moments of the largest order statistic of chi-squared variables. The closed form for these moments is available (see Nadarajah, 2008), but they have complicated form and we will keep the factor involving $m_{n_{l-1}}^{(K)}$ as it is. Then the total upper bound is

$$\left(\frac{c}{n_0}\right)^{1/2}\left(n_L(n_L+t)^{\frac{t}{2}-1}\right)^{1/t}\prod_{l=1}^{L-1}\left(\mathbb{E}\left[\left(\frac{c}{n_l}\sum_{i=1}^{n_l}m_{n_{l-1},i}^{(K)}\right)^{t/2}\right]\right)^{1/t}.$$

□

Estimating the moments of the gradient of maxout networks is a challenging topic, as can be seen from the above discussion, and is worthy of a separate investigation. It might be possible to obtain tighter upper bounds on it and on C_{grad}, a question that we leave for future work.

3.G Expected number of regions for networks without bias

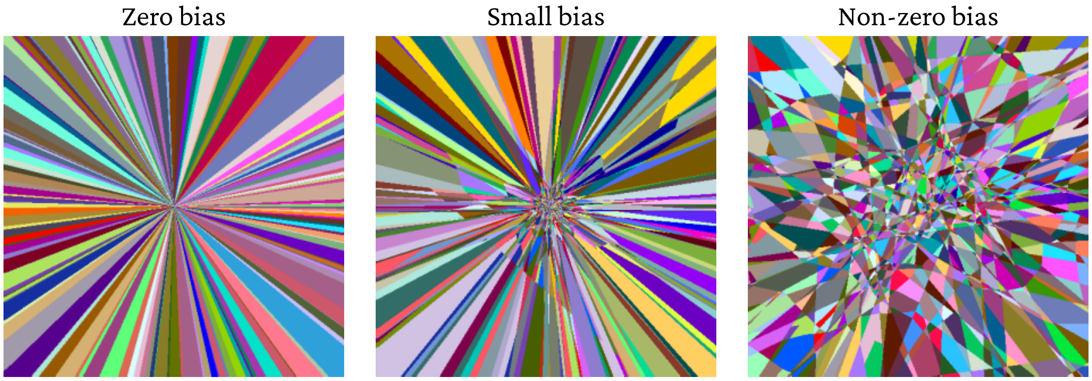


Figure 3.9: Linear regions of a 3 layer network with 100 units and the maxout rank $K = 2$. The network was initialized with the maxout-He distribution. Activation regions of a maxout network with zero biases are convex cones. Small biases are initialized as the biases sampled from the maxout-He distribution multiplied by 0.1. The majority of linear regions of a network with small biases are cones, and the ones that are not are small and concentrated around zero.

Zero biases of ReLU networks were discussed in Hanin and Rolnick (2019b) and studied in detail in Steinwart (2019). There is no distribution on the biases in the zero bias case, meaning that conditions on the biases from Theorem 3.9 are not satisfied. We closely follow the proofs in Hanin and Rolnick (2019b) and show that the arguments similar to those regarding the zero bias case in the ReLU networks also apply to the maxout networks. According to Lemma 3.37, activation regions of zero-bias maxout networks are convex cones, see Figure 3.9 for the illustration. In Corollary 3.39, we come to a conclusion that the number of activation regions in expectation in a network with zero biases grows as $O(n_0 (KN)^{n_0-1} \binom{K(n_0-1)}{2(n_0-1)})$.

Lemma 3.37. *Let $\mathcal{N}$ be a maxout network with biases set to zero. Then,*

(a) $\mathcal{N}$ is nonnegative homogeneous : $\mathcal{N}(cx) = c\mathcal{N}(x)$ for each $c \geq 0$.

(b) For every activation region $\mathcal{R}$ of $\mathcal{N}$, and every point x in $\mathcal{R}$, all points cx are also in $\mathcal{R}$ for $c > 0$ and $\mathcal{R}$ is a convex polyhedral cone.

Proof of Lemma 3.37. Each neuron of the network computes a function of the form $z(x_1, \dots, x_n) = \max_{k\in[K]} \{\sum_{i=1}^n w_{i,k} \cdot x_i\}$. Note that for any $c \geq 0$:

$$z(cx_1, \dots, cx_n) = \max_{k\in[K]} \left\{ c \sum_{i=1}^n w_{i,k} \cdot x_i \right\} = c \cdot \max_{k\in[K]} \left\{ \sum_{i=1}^n w_{i,k} \cdot x_i \right\} = c \cdot z(x_1, \dots, x_n).$$

Therefore, each neuron is equivariant under multiplication by a nonnegative constant c, and thus the overall network as well, proving (a). If $c > 0$, the activation patterns for x and cx are also identical, since for any inequality in the activation region definition we have

$$\sum_{i=1}^{n} w_{i,j} \cdot cx_i > \sum_{i=1}^{n} w_{i,j'} \cdot cx_i \iff \sum_{i=1}^{n} w_{i,j} \cdot x_i > \sum_{i=1}^{n} w_{i,j'} \cdot x_i, \quad j, j' \in [K].$$

This implies that x and cx lie in the same activation region, and that $\mathcal{R}$ is a convex polyhedral cone, see e.g. Chandru and Hooker (2011). This proves (b). ☐

Proposition 3.38 (Networks without biases do not have more regions)**.** *Suppose that $\mathcal{N}$ is a maxout network with biases and conditions from Theorem 3.9 are satisfied. Let $\mathcal{N}_0$ be the same network with all biases set to 0. Then, the total number of activation regions (in all the input space) for $\mathcal{N}_0$ is no more than that for $\mathcal{N}$.*

Proof of Proposition 3.38. We define an injective mapping from activation regions of $\mathcal{N}_0$ to regions of $\mathcal{N}$. For each region $\mathcal{R}$ of $\mathcal{N}_0$, pick a point $x_{\mathcal{R}} \in \mathcal{R}$. By Lemma 3.37, $cx_{\mathcal{R}} \in \mathcal{R}$ for each $c > 0$. Let $\mathcal{N}_{1/c}$ be the network obtained from $\mathcal{N}$ by dividing all biases by c, and observe that $\mathcal{N}(cx_{\mathcal{R}}) = c\mathcal{N}_{1/c}(x_{\mathcal{R}})$, with the same activation pattern between the two networks.

By picking c sufficiently large, $\mathcal{N}_{1/c}$ becomes arbitrarily close to $\mathcal{N}_0$. Therefore, for some sufficiently large c, $\mathcal{N}_0(cx_{\mathcal{R}})$ and $\mathcal{N}(cx_{\mathcal{R}})$ have the same pattern of activations. Regions of $\mathcal{N}$ in which $cx_{\mathcal{R}}$ lies are distinct for all distinct $\mathcal{R}$. Thus, the number of regions of $\mathcal{N}$ is at least as large as the number of regions of $\mathcal{N}_0$. ☐

We obtain following corollary of Theorem 3.9 for the zero-bias case.

Corollary 3.39 (Expected number of activation regions of zero-bias networks)**.** *Suppose that $\mathcal{N}_0$ is a fully-connected feed-forward maxout network with zero biases, n_0 inputs, a total of N rank K maxout units. Also, suppose that all conditions from Theorem 3.9, except for the conditions on the biases, are satisfied. Then there exists a constant T' depending on C_{grad} so that*

$$\mathbb{E}[\#\text{activation regions of } \mathcal{N}_0] \leq \begin{cases} K^N, & N \leq n_0 \\ 2n_0 \frac{(T'KN)^{n_0-1}\binom{K(n_0-1)}{2(n_0-1)}}{(n_0-1)!}, & N \geq n_0 \end{cases}.$$

The expectation is taken with respect to the distribution of weights in $\mathcal{N}_0$.

Proof of Corollary 3.39. Based on Proposition 3.38 we can use the same upper bound as for the networks with biases, thus for the case $N \leq n_0$, the expectation is upper bounded with K^N.

Now consider the case $N \geq n_0$. We will add biases to $\mathcal{N}_0$ in such a way that the bias conditions of Theorem 3.9 are satisfied with some C'_{bias}. Denote the resulting network with $\mathcal{N}$. Then, by Proposition 3.38, $\mathcal{N}$ has a region corresponding to each region of $\mathcal{N}_0$. All the corresponding regions in $\mathcal{N}$ are unbounded because according to Proposition 3.38 for any $x_{\mathcal{R}}$ from a region of $\mathcal{N}_0$ there exists a constant $c > 0$ so that $cx_{\mathcal{R}}$ belongs to a region in $\mathcal{N}$. Since all regions in $\mathcal{N}_0$ are unbounded, all corresponding regions in $\mathcal{N}$ are unbounded under such a mapping.

Therefore, to obtain the result, it is enough to upper-bound the number of unbounded activation regions of $\mathcal{N}$. Similarly to the proof of Theorem 3.9, consider a hypercube with a side length $\delta > \delta_0$, large enough to interest all the unbounded regions. Then the total number of unbounded activation regions of $\mathcal{N}$ is upper bounded by the sum of the numbers of activation regions intersecting each of the hypercube $2n_0$ facets, each of dimension $(n_0 - 1)$. By Theorem 3.9, the expected number of activation regions of $\mathcal{N}$ in $\mathbb{R}^{n_0-1}$ is upper bounded with $(\delta 2^5 C_{\text{grad}} C'_{\text{bias}} KN)^{n_0-1}\binom{K(n_0-1)}{2(n_0-1)}/(n_0-1)!$. Denoting $\delta 2^5 C_{\text{grad}} C'_{\text{bias}}$ with T' and combining everything we get the desired result. $\square$

3.H Proofs related to the decision boundary

3.H.1 Simple upper bound on the number of pieces of the decision boundary

A network used for multi-class classification into $M \in \mathbb{N}, M \geq 2$ classes can be seen as a network with a rank M maxout unit on top. Therefore, to discuss the decision boundary, we consider r-partial activation regions, $r \geq 1$, with at least one equation in the last unit. With J^r_{DB}, we denote the r-partial activation patterns corresponding to such regions and with $\mathcal{X}_{\text{DB},r} := \bigcup_{J^r_{\text{DB}} \in \mathcal{P}_{\text{DB},r}} \mathcal{R}(J^r_{\text{DB}}; \theta)$ their union. All decision boundary is then written as $\mathcal{X}_{DB}$.

Lemma 3.40 (Simple upper bound on the number of r-partial activation patterns of the decision boundary)**.** *Let $r \in \mathbb{N}_+$. The number of r-partial activation patterns in the decision boundary of a network with a total of N rank-K maxout units is upper bounded by $|\mathcal{P}_{DB,r}| \leq \sum_{i=1}^{\min\{M-1,r\}} \binom{M}{i+1}\binom{K(r-i)}{2(r-i)}\binom{N}{r-i} K^{N-r+i}$. The number of r-partial activation sub-patterns is upper bounded by $|\mathcal{S}_{DB,r}| \leq \sum_{i=1}^{\min\{M-1,r\}} \binom{M}{i+1}\binom{K(r-i)}{2(r-i)}\binom{N}{r-i}$.*

Proof of Lemma 3.40. Activation patterns for the decision boundary regions should have at least one equality in the upper unit. At the same time, the maximum possible number of equations in the last unit is $\min\{M-1, r\}$. To get all suitable activation patterns we need to sum over all these options.

Now consider a fixed number of equations $i \in \{1, \dots, \min\{M-1, r\}\}$. The number of ways to choose them is $\binom{M}{i+1}$ and the number of options for the all other units in the network is given by Lemma 3.6 for $r - i$. Combining everything, we get the claimed statement. $\square$

3.H.2 Lower bound on the maximum number of pieces of the decision boundary

The lower bound in the second item of Theorem 3.21 is based on a construction of parameters for which the network maps an n-cube in the input space to an n-cube in the output space in many-to-one fashion. This means that any feature implemented over the last layer will replicate multiple times over the input layer. We infer the following lower bound on the maximum number of pieces of the decision boundary of a maxout network.

Proposition 3.41 (Lower bound on the maximum number of pieces of the decision boundary)**.** *Consider a network $\mathcal{N}$ with n_0 inputs and L layers of $n_1, \dots, n_L$ rank-K maxout units followed by an M-class classifier. Suppose $n \leq n_0$, $\frac{n_l}{n}$ even, and $e_l = \min\{n_0, \dots, n_{l-1}\}$. Denote by $N(M, n)$ the maximum number of boundary pieces implemented by an M-class classifier over an n-cube. Then the maximum number of linear pieces of the decision boundary of $\mathcal{N}$ is lower bounded by $N(M, n) \prod_{l=1}^{L} (\frac{n_l}{n}(K-1)+1)^n$. If $n \geq M$ or $n \geq 4$, $N(M, n) = \binom{M}{2}$.*

The asymptotic order of this bound is $\Omega(M^2 \prod_{l=1}^{L} (n_l K)^{n_0})$.

Proof. We use the construction of parameters from Montúfar et al. (2022, Proposition 3.11) refining a previous construction for ReLU networks (Montufar et al., 2014) to have the network represent a many-to-one map. There are $\prod_{l=1}^{L} (\frac{n_l}{n}(K-1)+1)^n$ distinct linear regions whose image in the output space of the last layer contains an n-cube. The linear pieces of the decision boundary of an M-class classifier over an n-cube at the L-th layer will have a corresponding multiplicity over the input space. An M-class classifier is implemented as $\mathbb{R}^M \to [M]; y = (y_1, \dots, y_M) \mapsto \operatorname{argmax}_{r \in [M]} y_r$. This has $\binom{M}{2}$ boundaries, one between any two classes. If $n \geq M$, then the image of the preceding layers intersects all of these boundaries. More generally, the number of boundary pieces of an M-class classifier over n-dimensional space can be seen to correspond to the number of edges of a polytope with M vertices in n-dimensional space. The trivial upper bound $N(M, n) \leq \binom{M}{2}$ is attained if $1 < \lfloor \frac{n}{2} \rfloor$. This follows form the celebrated Upper Bound Theorem for the maximum number of faces of convex polytopes (McMullen, 1970). □

3.H.3 Upper bound on the expected volume of the decision boundary

Theorem 3.12 (Upper bound on the volume of the $(n_0 - r)$-skeleton of the decision boundary)**.** *Consider a bounded measurable set $S \subset \mathbb{R}^{n_0}$. Consider the notation and assumptions of Theorem 3.9, whereby the constants C_{grad} and C_{bias} are over S. Then, for any $r \in \{1, \dots, n_0\}$ we have*

$$\frac{\mathbb{E}[\mathrm{vol}_{n_0 - r}(\mathcal{X}_{\mathrm{DB}, r} \cap S)]}{\mathrm{vol}_{n_0}(S)} \leq (2 C_{\mathrm{grad}} C_{\mathrm{bias}})^r \sum_{i=1}^{\min\{M-1, r\}} \binom{M}{i+1} \binom{K(r-i)}{2(r-i)} \binom{N}{r-i}.$$

Proof of Theorem 3.12. Using Lemma 3.31, but considering only r-partial activation patterns that belong to the decision boundary, volume of the $(n_0 - r)$-skeleton of the decision boundary can be upper-bounded with

$$\sum_{\hat{J}^r_{\mathrm{DB}}} \int_S \mathbb{E}\left[\rho_{\boldsymbol{b}^r}((\boldsymbol{w}^m - \boldsymbol{w}^r) \cdot \boldsymbol{x}^m_{-1} + \boldsymbol{b}^m) \ \|\boldsymbol{J}((\boldsymbol{w}^m - \boldsymbol{w}^r) \cdot \boldsymbol{x}^m_{-1} + \boldsymbol{b}^m)\|\right] \ dx.$$

Upper-bounding the integral as in Theorem 3.10, but using Lemma 3.40 to count the number of entries in the sum, we get the final upper-bound

$$(2C_{\text{grad}}C_{\text{bias}})^r \sum_{i=1}^{\min\{M-1,r\}} \binom{N}{r-i}\binom{K(r-i)}{2(r-i)}\binom{M}{i+1}\text{vol}_{n_0}(S).$$

□

3.H.4 Upper bound on the expected number of pieces of the decision boundary

Lemma 3.42 (Upper bound on the expected number of r-partial activation regions of the decision boundary)**.** *Let $\mathcal{N}$ be a fully-connected feed-forward maxout network, with n_0 inputs, a total of N rank K maxout units, and M linear output units used for multi-classification. Fix $r \in \{1,\ldots,n_0\}$. Then, under the assumptions of Theorem 3.9, there exists $\delta_0 \leq 1/(2C_{\text{grad}}C_{\text{bias}})$ such that for all cubes $\mathcal{C} \subseteq \mathbb{R}^{n_0}$ with side length $\delta > \delta_0$,*

$$\frac{\mathbb{E}\left[\begin{smallmatrix}\#\, r\text{-partial activation regions in}\\ \text{the decision boundary of } \mathcal{N} \text{ in } \mathcal{C}\end{smallmatrix}\right]}{\text{vol}(\mathcal{C})} \leq \begin{cases} \sum_{i=1}^{\min\{M-1,r\}} \binom{M}{i+1}\binom{K(r-i)}{2(r-i)}\binom{N}{r-i}K^{N-r+i}, & N \leq n_0 \\ \frac{(2^4 C_{\text{grad}}C_{\text{bias}}N)^{n_0}(2K)^{n_0-1}}{n_0!} & \\ \times \sum_{i=1}^{\min\{M-1,n_0\}} \binom{M}{i+1}\binom{K(n_0-i)}{2(n_0-i)} \frac{\prod_{j=1}^{i}(n_0-j+1)}{\prod_{j=1}^{i}(N-1+j)}, & N \geq n_0 \end{cases}.$$

Here the expectation is taken with respect to the distribution of weights and biases in $\mathcal{N}$.

Proof of Lemma 3.42. Result for the case $N \leq n_0$ arises from Lemma 3.40. Consider $N \geq n_0$. The proof closely follows the proof of Theorem 3.10, and we only highlight the differences. Based on Lemma 3.12,

$$\mathbb{E}\left[\#\mathcal{V}_t\right] \leq (2C_{\text{grad}}C_{\text{bias}})^t \sum_{i=1}^{\min\{M-1,t\}} \binom{N}{t-i}\binom{K(t-i)}{2(t-i)}\binom{M}{i+1}\text{vol}_t(\mathcal{C}_t).$$

Therefore, the upper bound on the expected number of r-partial activation regions in the decision boundary is

$$\begin{aligned} &\sum_{t=r}^{n_0}\binom{t}{r}K^{t-r}(2C_{\text{grad}}C_{\text{bias}})^t \sum_{i=1}^{\min\{M-1,t\}} \binom{N}{t-i}\binom{K(t-i)}{2(t-i)}\binom{M}{i+1}\binom{n_0}{t}2^{n_0-t}\delta^t \\ &\leq (4\delta C_{\text{grad}}C_{\text{bias}})^{n_0}(2K)^{n_0-r} \sum_{i=1}^{\min\{M-1,n_0\}} \binom{M}{i+1}\binom{K(n_0-i)}{2(n_0-i)} \sum_{t=r}^{n_0}\binom{N}{t-i}\binom{n_0}{t} \end{aligned}$$

Re-writing $\binom{N}{t-i}\binom{n_0}{t}$ as $\binom{n_0}{t}^2\frac{\binom{N}{t-i}}{\binom{n_0}{t}}$ we can upper-bound it with

$$4^{n_0}\frac{\prod_{j=1}^{i}(t-j+1)}{\prod_{j=1}^{i}(N-t+j)}\frac{N^{n_0}}{n_0!} \leq 4^{n_0}\frac{\prod_{j=1}^{i}(n_0-j+1)}{\prod_{j=1}^{i}(N-r+j)}\frac{N^{n_0}}{n_0!}.$$

The final upper bound is then

$$\frac{(2^5 C_{\text{grad}} C_{\text{bias}} KN)^{n_0}}{(2K)^r n_0!}\sum_{i=1}^{\min\{M-1,n_0\}}\binom{M}{i+1}\binom{K(n_0-i)}{2(n_0-i)}\frac{\prod_{j=1}^{i}(n_0-j+1)}{\prod_{j=1}^{i}(N-r+j)}\operatorname{vol}(C).$$

Dividing this expression by $\operatorname{vol}(C)$ we get the desired result. □

The next theorem follows immediately from Lemma 3.42 if r is set to 1.

Theorem 3.11 (Upper bound on the expected number of linear pieces of the decision boundary)**.** *Let $\mathcal{N}$ be a fully-connected feedforward maxout network, with n_0 inputs, a total of N rank-K maxout units, and M linear output units used for multi-class classification. Under the assumptions of Theorem 3.9, there exists $\delta_0 \leq 1/(2C_{\text{grad}}C_{\text{bias}})$ such that for all cubes $C \subseteq \mathbb{R}^{n_0}$ with side length $\delta > \delta_0$,*

$$\frac{\mathbb{E}\left[\substack{\text{\# linear pieces in the}\\ \text{decision boundary of } \mathcal{N} \text{ in } C}\right]}{\operatorname{vol}(C)} \leq \begin{cases}\binom{M}{2}K^N, & N \leq n_0\\ \frac{(2^4 C_{\text{grad}}C_{\text{bias}})^{n_0}(2KN)^{n_0-1}}{(n_0-1)!}\binom{M}{2}\binom{K(n_0-1)}{2(n_0-1)}, & N \geq n_0\end{cases}.$$

Here the expectation is taken with respect to the distribution of weights and biases in $\mathcal{N}$.

3.H.5 Lower bound on the expected distance to the decision boundary

Now, using an approach similar to Hanin and Rolnick (2019a, Corollary 5), who provided a lower bound on the expected distance to the boundary of linear regions, we discuss a lower bound on the distance to the decision boundary. We will use the following result from that work.

Lemma 3.43 (Hanin and Rolnick 2019a, Lemma 12)**.** *Fix a positive integer $n \geq 1$, and let $Q \subseteq \mathbb{R}^n$ be a compact continuous piecewise linear submanifold with finitely many pieces. Define $Q_0 = \emptyset$ and let Q_t be the union of the interiors of all k-dimensional pieces of $Q \setminus (Q_0 \cup \cdots \cup Q_{t-1})$. Denote by $T_\varepsilon(X)$ the ε-tubular neighborhood of any $X \subset \mathbb{R}^n$. We have $\operatorname{vol}_n(T_\varepsilon(Q)) \leq \sum_{t=0}^{n}\omega_{n-t}\varepsilon^{n-t}\operatorname{vol}_k(Q_t)$,where $\omega_d :=$ volume of ball of radius 1 in $\mathbb{R}^d$.*

We will prove the following.

Corollary 3.13 (Distance to the decision boundary)**.** *Suppose $\mathcal{N}$ is as in Theorem 3.9. For any compact set $S \subset \mathbb{R}^{n_0}$ let x be a uniform point in S. There exists $c > 0$ independent of S so that*

$$\mathbb{E}[\text{distance}(x, \mathcal{X}_{\text{DB}})] \geq \frac{c}{2C_{\text{grad}}C_{\text{bias}}M^{m+1}m},$$

where $m := \min\{M-1, n_0\}$.

Proof of Corollary 3.13. Let $x \in K$ be uniformly chosen. Then, for any $\varepsilon > 0$, using Markov's inequality and Lemma 3.43, we have

$$\begin{aligned}\mathbb{E}[\text{distance}(x, \mathcal{X}_{DB})] &\geq \varepsilon P(\text{distance}(x, \mathcal{X}_{DB}) > \varepsilon) = \varepsilon(1 - P(\text{distance}(x, \mathcal{X}_{DB}) \leq \varepsilon)) \\ &= \varepsilon \left(1 - \mathbb{E}\left[\text{vol}_{n_0}(T_\varepsilon(\mathcal{X}_{DB}))\right]\right) \geq \varepsilon \left(1 - \sum_{t=1}^{n_0} \omega_{n_0-t}\varepsilon^{n_0-t}\mathbb{E}\left[\text{vol}_{n_0-t}(\mathcal{X}_{DB})\right]\right)\end{aligned}$$

The upper bound from Theorem 3.12 can be upper bounded further with

$$\begin{aligned}\mathbb{E}[\text{vol}_{n_0-t}(\mathcal{X}_{\text{DB},t} \cap S)] \leq &(2C_{\text{grad}}C_{\text{bias}})^t \sum_{i=1}^{\min\{M-1,t\}} \binom{M}{i+1}\binom{K(t-i)}{2(t-i)}\binom{N}{t-i} \text{vol}_{n_0}(S) \\ \leq &(2C_{\text{grad}}C_{\text{bias}})^t (4K^2N)^{t-1} M^{m^*+1} m^* \,\text{vol}_{n_0}(S),\end{aligned}$$

where $m^* := \min\{M-1, t\}$. Then the expectation of the distance can be lower bounded with

$$\varepsilon \left(1 - \sum_{t=1}^{n_0} (2C_{\text{grad}}C_{\text{bias}}\varepsilon)^t (4\varepsilon K^2 N)^{t-1} M^{m^*+1} m^*\right) \geq \varepsilon \left(1 - 2C_{\text{grad}}C_{\text{bias}}M^{m+1}m\varepsilon\right),$$

where $m := \min\{M-1, n_0\}$. Taking ε to be a small constant c times $1/(2C_{\text{grad}}C_{\text{bias}}M^{m+1}m)$ completes the proof. □

Remark 3.44 (Decision boundary of ReLU networks)**.** All proofs consider the indecision locus of the last unit on top of the network and reuse results on the volume of the boundary and the number of activation regions. If one sets K to 2, these results differ only in 2^{-r} from those for the ReLU networks. Therefore, the decision boundary analysis should also apply to the ReLU networks if one sets K to 2 with a difference only in the constant.

3.I Counting algorithms

3.I.1 Approximate counting of the activation regions

First, we describe an approximate method for counting linear regions that is useful for quickly estimating the number of linear regions or plotting them.

We generate a grid of inputs in an n_0-dimensional cube, compute the gradients with respect to the input, which is simply a product of weights on the path that corresponds to a given input, and then sum the gradient values for each input dimension of one input. Then, we compute the number of unique sums and use it as the number of linear regions.

The method is not exact because it works by computing network gradients on a grid, so it is possible to miss a small region. Also, it does not distinguish between regions with the same gradient value, which is one more reason it might miss some linear regions and why it counts linear regions, not activation regions. However, from what we have seen, if the grid has many points, the difference between the exact and approximate method is not that big.

3.I.2 Exact counting of the activation regions

The algorithm starts with a cube in which we want to count the activation regions defined with a set of linear inequalities in $\mathbb{R}^{n_0}$. We go through the network layer by layer, unit by unit, and for each unit, we determine if its pre-activation features attain a maximum on the regions obtained so far by checking the feasibility of the corresponding linear inequalities systems. For this, we use linear programming. More specifically, an interior-point method implementation from `scipy.optimize.linprog`. The use of linear programming is justified since, according to Lemma 3.4, the activation regions are convex.

The input to the simplex method becomes the combined system of inequalities for the region and the pre-activation feature. We set the objective to zero, meaning that any x can satisfy it. One has to use non-strict inequalities in linear programming methods, implying the boundary of activation regions is also included. We also add a small $\varepsilon = 1\text{e}{-6}$ to avoid zero solutions in a zero bias case. The inequalities for a pre-activation feature of some neuron z have the form

$$\{x \in \mathbb{R}^{n_0} \mid a_{z,j_0}(x;\theta) + b_{z,j_0} \geq a_{z,i}(x;\theta) + b_{z,i} + \varepsilon, \quad \forall i \in [K] \backslash [j_0]\}.$$

As a result, we get a new list of activation regions and pass it to the next unit.

To correctly estimate inequalities corresponding to a pre-activation feature on a specific region, one has to keep track of the function computed on this region, which has the form: $w_J^{(l)} \dots (w_J^{(0)} \cdot x + b_J^{(0)}) + \dots + b_J^{(l)}$, where J is an activation pattern of the region.

The pseudocode for the algorithm is in Algorithm 3.1, and the pseudocode for a check for one pre-activation feature is in Algorithm 3.2.

3.I.3 Exact counting of linear pieces in the decision boundary

We define an algorithm for exactly counting linear pieces in the decision boundary based on the algorithm from Section 3.I.2. Consider a classification problem with M classes, and to describe the decision boundary, add a maxout unit of rank M on top of the network. To count the number of linear pieces in the decision boundary, for each pair of classes, go through all the activation regions of the network. Construct a linear program for which the set of inequalities is given by a union of the region inequalities and inequalities which determine if the given classes attain maximum. Also, add the equality between these two classes. If the problem is feasible, there is a piece in the deci-

Algorithm 3.1 Exactly Count the Number of Activation Regions in a Maxout Network

```
function CountActivationRegions
    activation_regions = [starting_cube]
    for layer in $\{1, \dots, L\}$ do
        for unit in layer do
            new_activation_regions = []
            for region in activation_regions do
                for feature in unit do
                    ▷ See Algorithm 3.2
                    if NewRegionCheck(unit.features, feature, region) then
                        new_activation_regions.append(new_region)
            activation_regions = new_activation_regions
        for region in activation_regions do
            region.function = region.next_layer_function
            region.next_layer_function = []
    return length(activation_regions)
```

sion boundary. At the end of this process, one gets the total number of linear pieces in the decision boundary.

3.1.4 Algorithm discussion

There are two useful modifications to the method. First, to count the number of regions in a ReLU network instead of systems of $(K-1)$ linear inequalities, one can use inequalities of the form $w \cdot x + b \geq 0$ and $w \cdot x + b \leq 0$.

Second, to compute the number of activation regions in a slice, one can define a parametrization of the input space. We consider as the slice of a cube $\mathcal{C}$ the 2-space through three points $x_1, x_2, x_3 \in \mathbb{R}^{n_0}$, meaning the slice has the form $V = \{x = v_0 + v_1 y_1 + v_2 y_2 \in \mathbb{R}^{n_0} \colon (y_1, y_2) \in \mathbb{R}^2 \cap \mathcal{C}\}$, where $v_0 = (x_1 + x_2 + x_3)/3 \in \mathbb{R}^{n_0}$, and $v_1, v_2 \in \mathbb{R}^{n_0}$ are an orthogonal basis of span$\{x_2 - x_1, x_3 - x_1\}$, and v_1, v_2 are orthonormal. We can evaluate the network function over such a slice by augmenting the network by a linear layer $\phi \colon \mathbb{R}^2 \to \mathbb{R}^{n_0}$ with weights v_1, v_2 and biases v_0. We used images from 3 different classes as the points that define the slice.

We usually performed the computation in a 2D slice, which is reasonably fast because the number of regions is not large if the input dimension is not high, as suggested by Theorem 3.9. Additionally, note that the check for a given unit is embarrassingly parallel, meaning the computation can be accelerated. To demonstrate that the computation can be carried out in a reasonable time, we also analyze the algorithm's space-time complexity.

Space-time complexity of the algorithm

To start, we estimate complexities for some number of activation regions R. Firstly, consider the

Algorithm 3.2 Auxiliary Function That Checks if a Pre-Activation Feature Creates a New Region

```
function NewRegionCheck(unit_features, feature, region)
    objective = zeros
    inequalites = region.inequalities
    unit_features.weights = unit_features.weights $\times$ region.weights
    unit_features.biases = unit_features.weights $\times$ region.biases
                                              + unit_features.biases
    for another_feature in unit_features $\setminus$ feature do
        inequalities.append(another_feature.weights - feature.weights $\times x$
                                  $\leq$ feature.bias - another_feature.bias)
    if LinearProgramming.Solve(objective, inequalities) then
        next_layer_function = region.next_layer_function
                                  + [feature.weights, feature.bias]
        return Region(inequalites, region.function, next_layer_function)
    return None
```

space complexity. Since we store all activation regions, the space requirement grows as R multiplied by an activation region size. We store a region as a constant size function computed on it and as a system of linear inequalities. The maximum number of inequalities is attained when each of N neurons adds a new system of inequalities to the region, while $K-1$ inequalities determine that one pre-activation feature attains a maximum. Therefore, the space complexity of the algorithm is $\mathcal{O}(RKN)$.

Now consider the time complexity. Since we traverse the network unit by unit, and for each pre-activation feature of a unit and each available activation region, we solve a linear programming problem, the time complexity is $\mathcal{O}$ of RKN times the time complexity of a linear programming method. We have used an interior point method that has a polynomial-time complexity of $\mathcal{O}(\frac{n^3}{\log n}L)$, see Anstreicher (1999), where n is the dimension of the variables, which is the dimension of the network input n_0, and L is the number of bits used to represent the method input. The input is the set of inequalities, and as we have just discussed, its size is $\mathcal{O}(KN)$. Combining everything and using $\mathcal{O}(n^3L)$ instead of $\mathcal{O}(\frac{n^3}{\log n}L)$ for simplicity, we get that the time complexity of the whole algorithm is $\mathcal{O}(RK^2N^2n_0^3)$.

To get complexities for the average case, assume $N \geq n_0$. Then, based on Theorem 3.9, R grows as $\mathcal{O}((K^3N)^{n_0})$. Therefore, the space complexity is $\mathcal{O}(KN(K^3N)^{n_0})$ and the time complexity is $\mathcal{O}(K^2N^2n_0^3(K^3N)^{n_0})$. Both space and time complexities grow exponentially with the input dimension but polynomially with the number of neurons and a maxout unit's rank.

3.J Parameter initialization

3.J.1 He initialization

We briefly recall the parameter initialization procedure for ReLU networks which is commonly referred to as "He initialization" (He et al., 2015). This follows the motivation of the work by Xavier and co-authors (Glorot and Bengio, 2010). To train deep networks, one would like to avoid vanishing or exploding gradients. The approach formulates a sufficient condition for the norms of the activations across layers to not blow up or vanish. For ReLU networks, this leads to sampling the weights from a distribution with standard deviation $\sqrt{2/n_l}$.

3.J.2 He-like initialization for maxout (Maxout-He)

We follow the derivation from Glorot and Bengio (2010) and He et al. (2015) but for the case of maxout units. We note that a He-like initialization for maxouts was considered by Sun et al. (2018) but only for $K = 2$. We focus on the forward pass and consider fully-connected layers. The idea is to investigate the variance of the responses in each layer. We use the following notations. For a given layer l with d units and n_l inputs, a (pre-activation) response is $\boldsymbol{y}_l = W_l \boldsymbol{x}_l + \boldsymbol{b}_l$, where $\boldsymbol{x}_l \in \mathbb{R}^{n_l}$ is an input vector to the layer, $W_l \in \mathbb{R}^{d \times n_l}$ is a matrix, $\boldsymbol{b}_l \in \mathbb{R}^d$ is a vector of biases. We have $\boldsymbol{x}_l = \phi(\boldsymbol{y}_{l-1})$, where ϕ is the activation function.

We assume the elements in W_l are independent and identically distributed (iid). We assume that the elements in $\boldsymbol{x}_l$ are also iid. We assume that $\boldsymbol{x}_l$ and W_l are independent of each other. Denote y_l, w_l, and x_l the random variables of each element in $\boldsymbol{y}_l$, W_l, and $\boldsymbol{x}_l$ respectively. In the following we assume that biases are zero. Then we have:

$$\mathrm{Var}[y_l] = n_l \mathrm{Var}[w_l \cdot x_l].$$

If we assume further that w_l has zero mean, then the variance of the product of independent variables gives us:

$$\mathrm{Var}[y_l] = n_l \mathrm{Var}[w_l] \mathbb{E}[x_l^2]. \tag{3.13}$$

We need to estimate $\mathbb{E}[x_l^2]$. For ReLU, $\mathbb{E}[x_l^2] = \frac{1}{2}\mathrm{Var}[y_{l-1}]$. For maxout we get a different result. Let K be the rank of a maxout unit. Then $x_l = \phi(y_{l-1}) = \max_{k \in [K]}\{y_{l-1,k}\}$. The $y_{l-1,1}, \ldots, y_{l-1,K}$ are independent and have the same distribution. We denote $f(t)$ and $F(t)$ the pdf and cdf of this distribution. The cdf for $x_l = \max_{k \in [K]}\{y_{l-1,k}\}$ is, dropping the subscript $l-1$

of $y_{l-1,k}$ for simplicity of notation,

$$\mathsf{Pr}\left(\max_{k\in[K]}\{y_k\}<t\right)=\mathsf{Pr}\left(y_1,\dots,y_K<t\right)=\prod_{k=1}^{K}\mathsf{Pr}\left(y_k<t\right)=(F(t))^K.$$

In turn, the expectation of the square is

$$\mathbb{E}\left[\max_{k\in[K]}\{y_k\}^2\right]=\int_{\mathbb{R}}t^2\frac{d}{dt}\left[(F(t))^K\right]dt=K\int_{\mathbb{R}}t^2\left(F(t)\right)^{K-1}f(t)dt.$$

Now we can apply this formula to discuss the cases of a uniform distribution on an interval and a normal distribution. If we assume that w_{l-1} has a symmetric distribution around zero, then y_{l-1} has zero mean and has a symmetric distribution around zero.

Uniform Distribution Assuming y_{l-1} has a uniform distribution on the interval $[-a,a]$, we get $\mathsf{Var}[y_{l-1}]=a^2/3$, and

$$\begin{aligned}K=2:\mathbb{E}[x_l^2]&=\frac{a^2}{3}=\mathsf{Var}[y_{l-1}],\\K=3:\mathbb{E}[x_l^2]&=\frac{2a^2}{5}=\frac{6}{5}\mathsf{Var}[y_{l-1}],\\K=4:\mathbb{E}[x_l^2]&=\frac{7a^2}{15}=\frac{7}{5}\mathsf{Var}[y_{l-1}],\\K=5:\mathbb{E}[x_l^2]&=\frac{11a^2}{21}=\frac{11}{7}\mathsf{Var}[y_{l-1}].\end{aligned}$$

More generally, $\mathbb{E}[x_l^2]=4a^2(\frac{K}{K+2}-\frac{K}{K+1}+\frac{K}{4K})$.

Normal Distribution Assuming y_{l-1} has a normal distribution $\mathcal{N}(0,\sigma^2)$, the closed form solution is available for up to $K=4$. We have:

$$\begin{aligned}K=2:\mathbb{E}[x_l^2]&=\mathsf{Var}[y_{l-1}],\\K=3:\mathbb{E}[x_l^2]&=\frac{\sqrt{3}+2\pi}{2\pi}\mathsf{Var}[y_{l-1}],\\K=4:\mathbb{E}[x_l^2]&=\frac{\sqrt{3}+\pi}{\pi}\mathsf{Var}[y_{l-1}],\\K=5:\mathbb{E}[x_l^2]&\approx 1.80002\mathsf{Var}[y_{l-1}].\end{aligned}$$

Inserting the expressions for $\mathbb{E}[x_l^2]$ into (3.13),

$$\mathsf{Var}[y_l]=n_l\mathsf{Var}[w_l]c\mathsf{Var}[y_{l-1}],$$

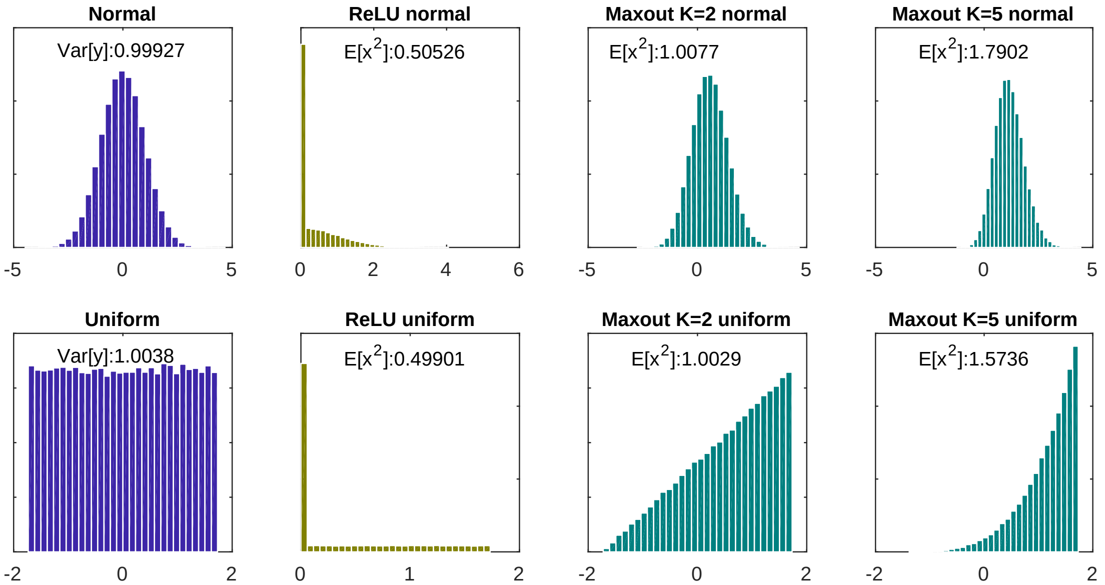


Figure 3.10: Shown are normal (top) and uniform (bottom) input distributions, as well as the corresponding response distributions for ReLU, maxout of rank $K = 2$, and maxout of rank $K = 5$. The expectation of the square response for maxouts of rank $K > 2$ depends not only on the variance but also on the particular shape of the input distribution.

where c depends on the distribution and on K. Putting the results together for all layers,

$$\text{Var}[y_L] = \text{Var}[y_1] \prod_{l=2}^{L} c n_l \text{Var}[w_l].$$

A sufficient condition for this product not to increase or decrease exponentially in L is that, for each layer, $c n_l \text{Var}[w_l] = 1$. This is achieved by setting the standard deviation (std) of w_l as $\sqrt{1/c n_l}$. For $K = 2$ this is $\sqrt{1/n_l}$ for both uniform and normal distribution. For a uniform distribution, we obtain the condition $\text{Var}[w_l] = \frac{1}{n_l(\frac{1}{4} - \frac{K}{(K+2)(K+1)})}$.

We notice that for ReLU, the particular shape of the distribution of the (pre-activation) response y_{l-1} does not impact the expected square of the activation x_l. Indeed, as soon as w_l is assumed to be symmetric around zero, one obtains $\mathbb{E}[x_l^2] = \frac{1}{2}\text{Var}[y_{l-1}]$. In contrast, for maxout units of rank $K > 2$, the particular shape of the distribution of y_{l-1} does affect the value of $\mathbb{E}[x_l^2]$. This is why we obtain different conditions on the standard deviation of the weight distributions depending on the assumed response distribution. The situation is illustrated in Figure 3.10. Among the possible distributions that one might assume for y_{l-1}, a normal distribution appears most natural. Therefore, we take the standard deviations obtained under this assumption as the ones defining the maxout-He initialization procedure. The values of the std of w_l for K up to 5 for normal distributions are shown in Table 3.1.

3.J.3 Sphere initialization

If we initialize the pre-activation features of a maxout unit independently, then we expect the number of regions of the unit will be significantly smaller than K, as discussed in Section 3.C. In view of Proposition 3.20, the number of regions of a maxout unit with weights $w_1, \dots, w_K \in \mathbb{R}^n$ and biases $b_1, \dots, b_K \in \mathbb{R}$ is equal to the number of upper vertices of the polytope $\operatorname{conv}\{(w_r, b_r) \colon r \in [K]\}$. Hence one way to have each rank-K maxout unit have K linear regions over its input at initialization is to initialize the pre-activation feature parameters as points in the upper half-sphere $\{(w, b) \in \mathbb{R}^{n+1} \colon \|(w, b)\| = 1, b > 0\}$. This can be done as follows. For each pre-activation feature $i = 1, \dots, K$:

1. Sample (w_i, b_i) from a Gaussian on $\mathbb{R}^{n+1}$.
2. Normalize $(w_i, b_i)/\|(w_i, b_i)\|$.
3. Replace b_i with $|b_i|$.

If desired, subtract a constant c from each of the biases $b_1, \dots, b_K$. For instance, one may choose c so that the mean output of the maxout unit is approximately 0 for inputs from a Gaussian distribution. We have used $c = 1/\sqrt{Kn_l}$ in our implementation, and Gaussian had zero mean and unit covariance.

3.J.4 Many regions initialization

We can initialize the parameters of a maxout layer so that the layer has the largest possible number of linear regions over its input space. A description of parameter choices maximizing the number of regions for a layer of maxout units has been given by Montúfar et al. (2022, Proposition 3.4). The number of regions of a layer of maxout units corresponds to the number of upper vertices of a Minkowski sum of polytopes. A construction maximizing the number of vertices of Minkowski sums was presented earlier by Weibel (2012). The procedure is as follows. Let the layer have input dimension n. For each unit $j = 1, \dots, m$:

1. Sample a vector $v_j \in \mathbb{R}^n$ from a distribution which has a density.
2. For each pre-activation feature $i = 1, \dots, K$ set the weights and bias as $w_{j,i} = v_j \cos(\pi i/K)$ and $b_{j,i} = \sin(\pi i/K)$.

This construction ensures that each unit has K linear regions separated by $K - 1$ parallel hyperplanes, and the hyperplanes of different units are in general position. Then the number of regions of the layer is the one indicated in the first item of Theorem 3.21.

If desired, one can add some noise to each of the above parameters (e.g. standard normal times a small constant) in order to have a parameter distribution which has a density. If desired, one can also normalize the initialization by subtracting an appropriate constant (e.g. to achieve a zero mean activation) and dividing by an appropriate standard deviation (e.g. to achieve that the activations have a unit mean norm). We were sampling v_j from a Gaussian distribution with mean zero and std chosen according to maxout-He.

3.J.5 Steinwart-like initialization for maxout

Steinwart (2019) investigated initialization in ReLU networks. He suggested that having the non-linear locus of different units evenly spaced over the input space at initialization could lead to faster convergence of training, which he also supported with experiments on the datasets from the UCI repository. We can formulate a version of this general idea for the case of maxout networks as follows.

1. Assume we have some generic initialization procedure for individual units, which gives us weights $w_1, \dots, w_K \in \mathbb{R}^n$ and biases $b_1, \dots, b_K \in \mathbb{R}$. The initialization procedure could be for instance "Sphere". Upon initialization, our unit is computing a function $x \mapsto \max\{\langle w_1, x\rangle + b_1, \dots, \langle w_K, x\rangle + b_K\}$ with non-linear locus that we denote L.
2. For each unit, sample a vector c uniformly from the cube $[-1, 1]^n$. Alternatively, sample c as a random convex combination of a random subset of the training data, so that $c = \sum_{i=1}^{m} p_i x_i$, where $(p_1, \dots, p_m)$ is a random probability vector and $x_1, \dots, x_m$ are m randomly selected training input examples.
3. Now set the weights as $w_1, \dots, w_K$ and the biases as $b_1 + \langle w_1, c\rangle, \dots, b_K + \langle w_K, c\rangle$. Now our unit is computing a function $x \mapsto \max\{\langle w_k, x\rangle + b_k + \langle w_k, c\rangle\} = \max\{\langle w_k, x + c\rangle + b_k\}$. Hence the nonlinear becomes $L - c$.

3.K Experiment details and additional experiments

In this section, we provide details on the implementation and additional experimental results. All the experiments were implemented in Python using PyTorch (Paszke et al., 2019), numpy (Harris et al., 2020), scipy (Jones et al., 2001) and mpi4py (Dalcin et al., 2011), with plots created using matplotlib (Hunter, 2007). In the experiments concerning the network training, we used the MNIST dataset (LeCun and Cortes, 2010). PyTorch, numpy, scipy and mpi4py are made available under the BSD license, matplotlib under the PSF license, and MNIST dataset under the Creative Commons Attribution-Share Alike 3.0 license. We conducted all experiments on a CPU cluster that uses Intel Xeon IceLake-SP processors (Platinum 8360Y) with 72 cores per node and 256 GB RAM. The most extensive experiments were usually running for 2-3 days on 32 nodes. The computer implementation of the key functions is available on GitHub at `https://github.com/hanna-tseran/maxout_complexity`.

For the MNIST experiments, we use the Adam optimizer with mini-batches of size 128 with the learning rate 0.001 and the standard Adam hyperparameters from PyTorch (betas are 0.9 and 0.999). Counting at initialization was performed in the window $[-50, 50]^2$, in the training experiments in the window $[-400, 400]^2$ defined on the slice, and images of the regions and the decision boundary were obtained in the window $[-50, 50]^2$ also defined on the slice. All results are averaged over 30 instances where applicable. The network architectures are specified in the individual experiments. The parameter initialization procedures are implemented following the descriptions in Section 3.J. For the experiments counting the number of activation regions and pieces in the decision boundary, we use homemade implementations of the algorithms described in Section 3.I. Further below, we present the details and additional results of the individual experiments.

Details on Figure 3.1 We consider a network with 2 input units, three layers of rank-3 maxout units of width 3, and a single linear output unit. We fix three parameter vectors $\theta_0, \theta_1, \theta_2$ drawn from a normal distribution over the parameter space and define a grid of parameter values $\theta(\xi_1, \xi_2) = \theta_0 + \xi_1\theta_1 + \xi_2\theta_2$ with (ξ_1, ξ_2) taking 102400 uniformly spaced values in $[-1, 1]^2$. For each of these parameter values, we estimate the number of linear regions that the represented function has over the square $[-1, 1]^2$ in the input space. To this end, we evaluate the gradient of the function over 102400 uniformly spaced input points and take the number of distinct values as an estimate for the number of linear regions. Then we plot the estimated number of linear regions as a function of (ξ_1, ξ_2). A subset of 25 out of the evaluated functions is shown in Figure 3.11.

Comparison to the upper bound Figures 3.12 and 3.13 complement Figure 3.2. Figure 3.12 compares the number of activation regions and linear pieces in the decision boundary to the formulas both with and without the constants, while Figure 3.13 demonstrates the results for different values of K.

Effects of the depth and the number of units on the number of linear regions Results adding more information to Figure 3.3 are in Figure 3.14. It shows that ReLU networks and maxout networks with $K = 2$ have a similar number of activation regions that does not depend on the network depth but rather on the total number of units. This figure also shows that maxout networks with ranks $K > 2$ tend to have fewer regions as the depth increases, but the number of units remains constant, and that the difference in the number of regions becomes more apparent for larger ranks.

Effects of different initializations on training Figure 3.16 is a more detailed version of Figure 3.6. It shows how convergence speed changes for different network depths and different maxout ranks given different initializations. The improvement from maxout-He, sphere, and many regions initializations compared to ReLU-He initialization becomes more noticeable with larger network depth and larger maxout rank. We have also checked how the Steinwart initialization affects the convergence speed, but found no significant difference in this particular experiment. We used the approach where c is taken as a convex combination of all training data points (weights p uniformly at random from the probability simplex). The results are shown in Figure 3.15.

Effects of different initializations on the number of activation regions and pieces in the decision boundary during training Figure 3.18 adds more information to Figure 3.5 and demonstrates how the number of activation regions and linear pieces in the decision boundary changes for different initializations during training on the MNIST dataset. We observe that the number of activation regions and pieces of the decision boundary increase for all tested initialization procedures as training progresses. Nonetheless, the number remains much lower than the theoretical maximum. Figure 3.17 illustrates how linear regions and the decision boundary evolve during training.

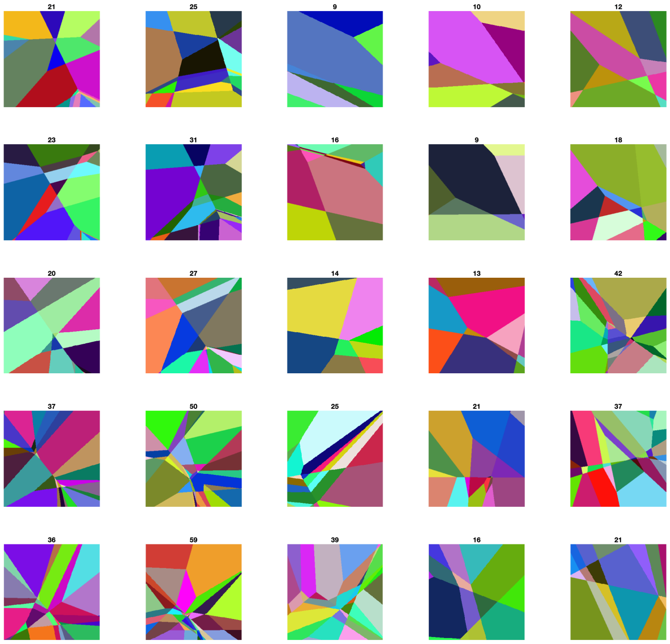
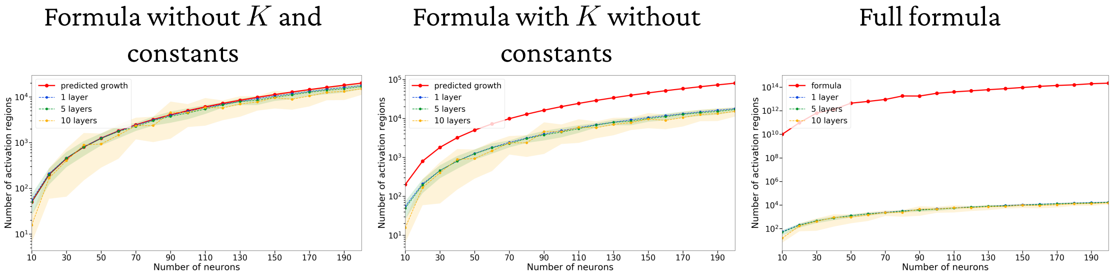
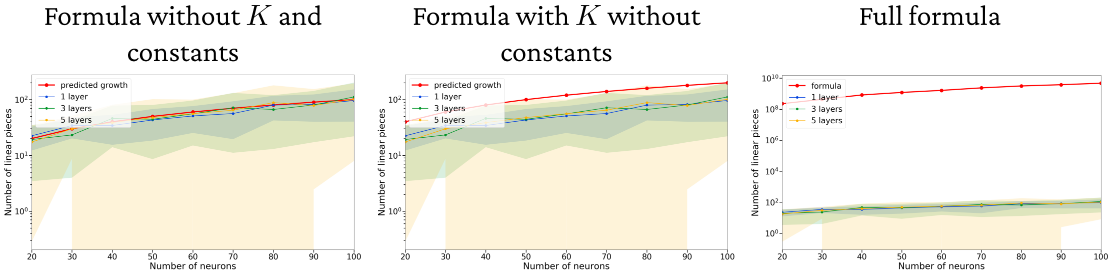


Figure 3.11: A few functions represented by a maxout network for different parameter values in a 2D slice of parameter space. For each function, we plot regions of the input space with different gradient values using different colors.

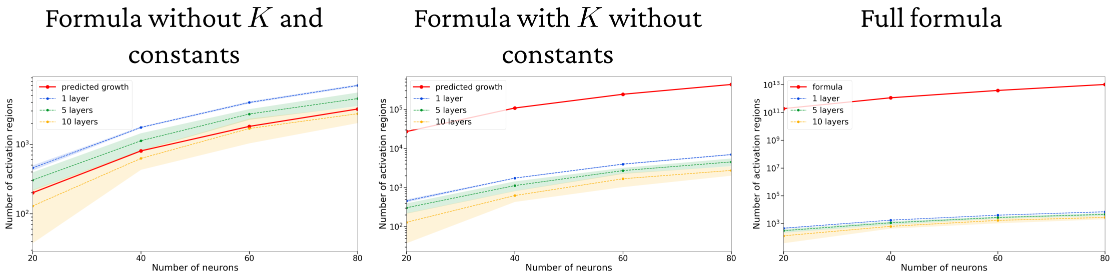


(a) Number of activation regions for a network with ReLU-He normal initialization.



(b) Number of linear pieces in the decision boundary for a network with maxout-He normal initialization.

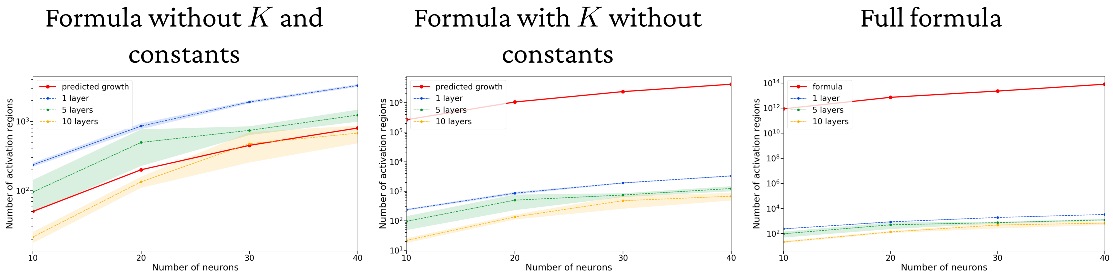
Figure 3.12: Comparison to the formulas with and without the constants for the number of activation regions and linear pieces in the decision boundary from Theorem 3.9 and Theorem 3.11 respectively. Networks had 100 units and maxout rank $K = 2$. The settings are similar to those in Figure 3.2.


Formula without K and constants

Formula with K without constants

Full formula

(a) $K = 3$.

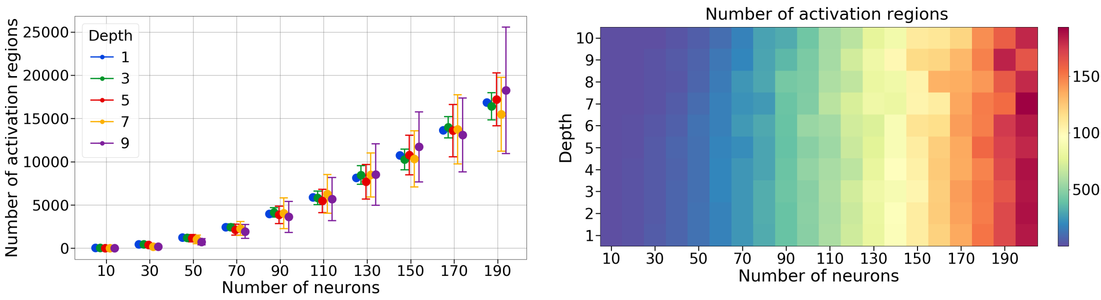

Formula without K and constants

Formula with K without constants

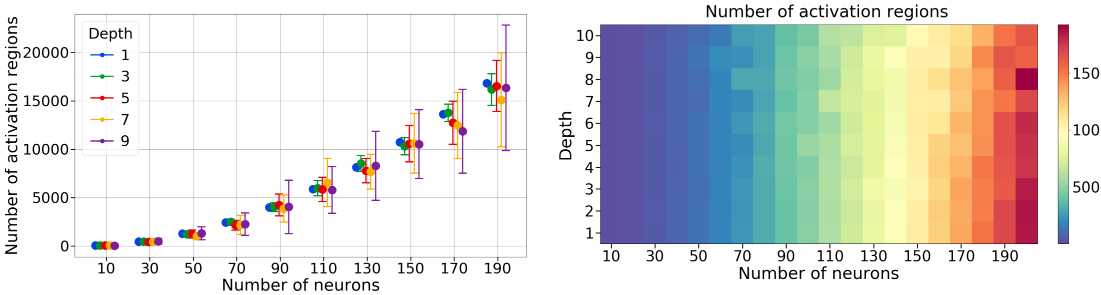
Full formula

(b) $K = 5$.

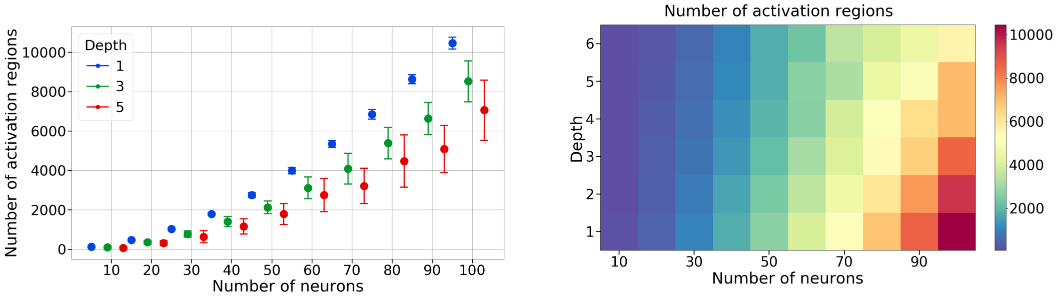
Figure 3.13: Comparison to the formula from Theorem 3.9 for maxout ranks $K = 3$ and $K = 5$. The networks were initialized with maxout-He normal initialization. We observe the increase in the number of activation regions as the maxout rank increases, and for networks with higher maxout rank deeper networks tend to have less regions than less deep networks with the same rank.



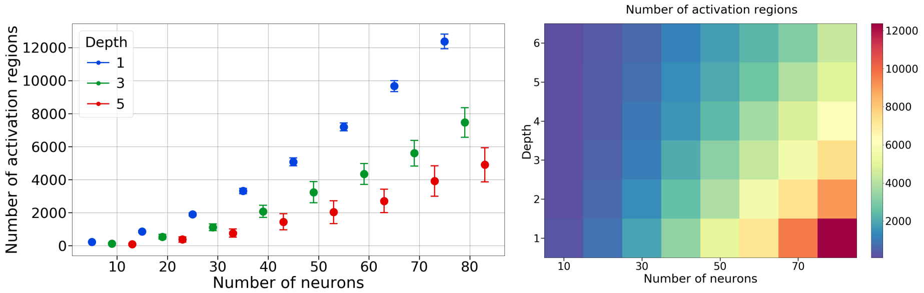
(a) ReLU network with ReLU-He normal initialization.



(b) Maxout network with maxout rank $K = 2$ and ReLU-He normal initialization.



(c) Maxout network with $K = 3$. Maxout-He normal initialization.



(d) Maxout network with $K = 5$. Maxout-He normal initialization.

Figure 3.14: Difference between the effects of depth and number of neurons on the number of activation regions. These plots are additional to Figure 3.3 and have similar settings. ReLU and maxout networks with $K = 2$ have a similar number of linear regions. For maxout rank $K > 2$ deeper networks tend to have less regions than less deep networks with the same rank. For $K = 3$ the gaps between different depths are smaller than for $K = 5$.

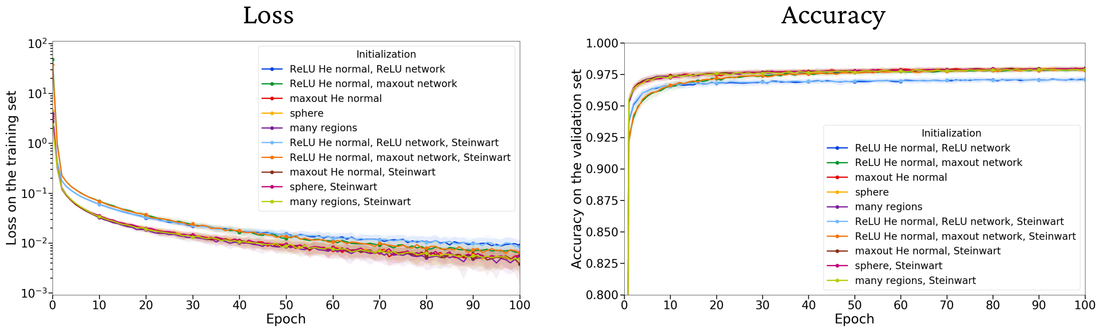


Figure 3.15: Effect of the Steinwart initialization approach on the convergence speed during training on the MNIST dataset for a network with 200 units and 5 layers. Maxout rank was $K = 5$. In this experiment, for various initialization procedures, the addition or omission of a random shift of the non-linear regions of different units led to similar training curves.

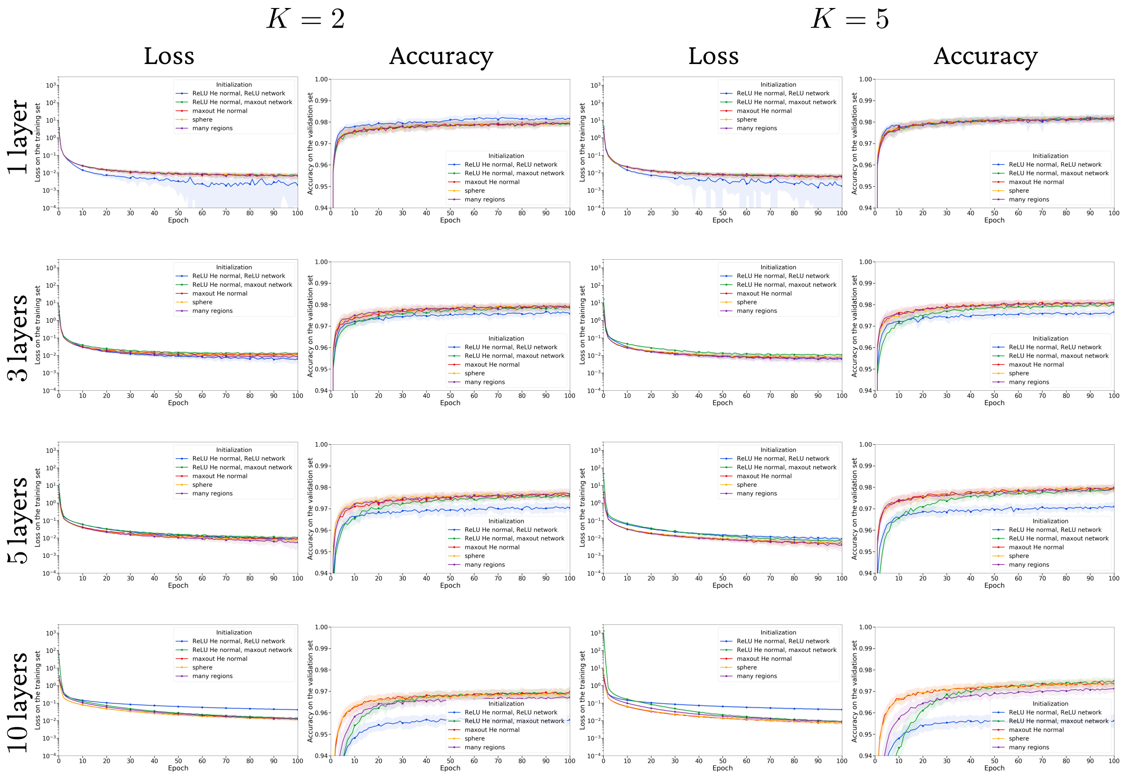
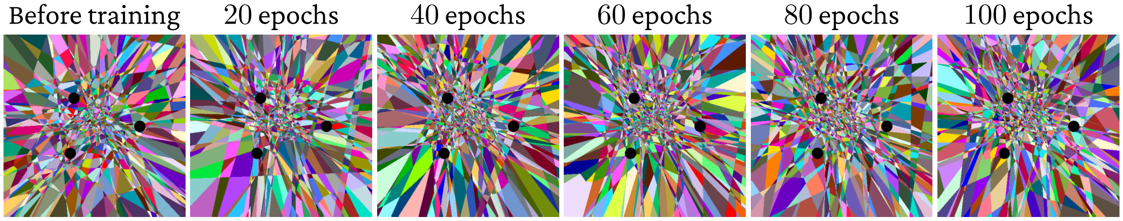
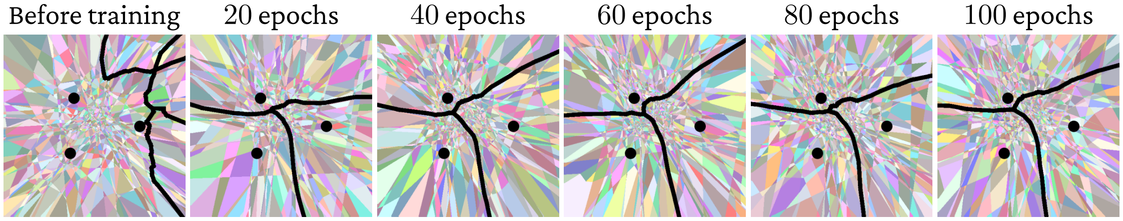


Figure 3.16: Effect of the initialization on the convergence speed during training on the MNIST dataset of networks with 200 units depending on the network depth and the maxout rank. Maxout-He, sphere, and many regions initializations behave similarly, and the improvement in the convergence speed becomes more noticeable for larger network depth and maxout rank.

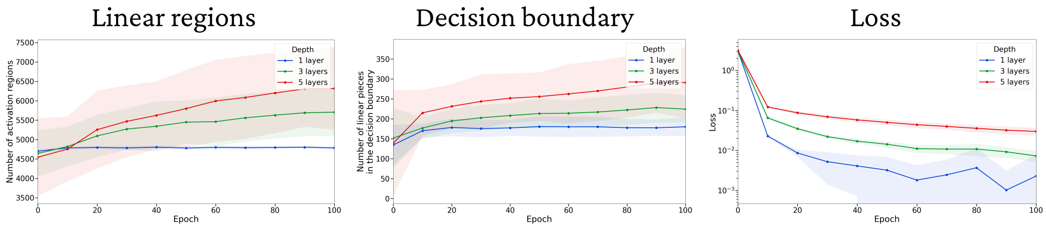


(a) Linear regions.



(b) Decision boundary.

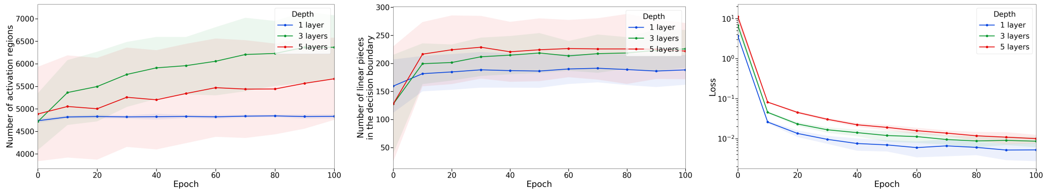
Figure 3.17: Evolution of the linear regions and the decision boundary during training on MNIST in a 2D slice determined by three random input points from the dataset. The network had 3 layers, a total of 100 maxout units of rank $K = 2$, and was initialized with the maxout-He initialization.


Linear regions

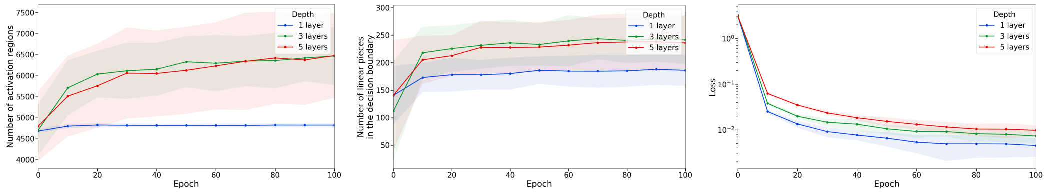
Decision boundary

Loss

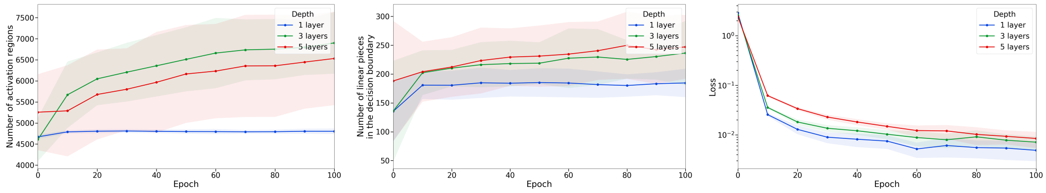
(a) ReLU network with the ReLU-He normal initialization.



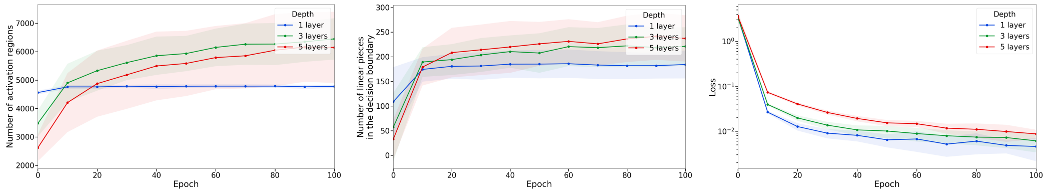
(b) Maxout network with the ReLU-He normal initialization.



(c) Maxout network with the maxout-He normal initialization.



(d) Maxout network with the sphere initialization.



(e) Maxout network with the many regions initialization.

Figure 3.18: Change in the number of linear regions and the decision boundary pieces during 100 training epochs given different initializations. Networks had 100 neurons and for maxout networks $K = 2$. Both the number of linear regions and linear pieces of the decision boundary increase during training for all initializations but remain much smaller than the theoretical maximum. The settings were the same as in Figure 3.5.

Chapter 4

Expected gradients of maxout networks and consequences to parameter initialization

4.1 Introduction

We study the gradients of maxout networks and derive a rigorous parameter initialization strategy as well as several implications for stability and expressivity. Maxout networks were proposed by Goodfellow et al. (2013) as an alternative to ReLU networks with the potential to improve issues with dying neurons and attain better model averaging when used with Dropout (Hinton et al., 2012). Dropout is used in transformer architectures (Vaswani et al., 2017), and maximum aggregation functions are used in Graph Neural Networks (Hamilton, 2020). Therefore, we believe that developing the theory and implementation aspects of maxout networks can serve as an interesting platform for architecture design. We compute bounds on the moments of the gradients of maxout networks depending on the parameter distribution and the network architecture. The analysis is based on the input-output Jacobian. We discover that, in contrast to ReLU networks, when initialized with a zero-mean Gaussian distribution, the distribution of the input-output Jacobian of a maxout network depends on the network input, which may lead to unstable gradients and training difficulties. Nonetheless, we can obtain a rigorous parameter initialization recommendation for wide networks. The analysis of gradients also allows us to refine previous bounds on the expected number of linear regions of maxout networks at initialization and derive new results on the length distortion and the NTK.

Maxout networks A rank-K maxout unit, introduced by Goodfellow et al. (2013), computes the maximum of K real-valued parametric affine functions. Concretely, a rank-K maxout unit with n inputs implements a function $\mathbb{R}^n \to \mathbb{R}; \boldsymbol{x} \mapsto \max_{k\in[K]}\{\langle W_k, \boldsymbol{x}\rangle + b_k\}$, where $W_k \in \mathbb{R}^n$ and

$b_k \in \mathbb{R}, k \in [K] := \{1, \dots, K\}$, are trainable weights and biases. The K arguments of the maximum are called the pre-activation features of the maxout unit. This may be regarded as a multi-argument generalization of a ReLU, which computes the maximum of a real-valued affine function and zero. Goodfellow et al. (2013) demonstrated that maxout networks could perform better than ReLU networks under similar circumstances. Additionally, maxout networks have been shown to be useful for combating catastrophic forgetting in neural networks (Goodfellow et al., 2015). On the other hand, Castaneda et al. (2019) evaluated the performance of maxout networks in a big data setting and observed that increasing the width of ReLU networks is more effective in improving performance than replacing ReLUs with maxout units and that ReLU networks converge faster than maxout networks. We observe that proper initialization strategies for maxout networks have not been studied in the same level of detail as for ReLU networks and that this might resolve some of the problems encountered in previous maxout network applications.

Parameter initialization The vanishing and exploding gradient problem has been known since the work of Hochreiter (1991). It makes choosing an appropriate learning rate harder and slows training (Sun, 2019). Common approaches to address this difficulty include the choice of specific architectures, e.g. LSTMs (Hochreiter, 1991) or ResNets (He et al., 2016), and normalization methods such as batch normalization (Ioffe and Szegedy, 2015) or explicit control of the gradient magnitude with gradient clipping (Pascanu et al., 2013). We will focus on approaches based on parameter initialization that control the activation length and parameter gradients (LeCun et al., 2012; Glorot and Bengio, 2010; He et al., 2015; Gurbuzbalaban and Hu, 2021; Zhang et al., 2019; Bachlechner et al., 2021). He et al. (2015) studied forward and backward passes to obtain initialization recommendations for ReLU. A more rigorous analysis of the gradients was performed by Hanin and Rolnick (2018); Hanin (2018), who also considered higher-order moments and derived recommendations on the network architecture. Sun et al. (2018) derived a corresponding strategy for rank $K = 2$ maxout networks. For higher maxout ranks, Tseran and Montúfar (2021) considered balancing the forward pass, assuming Gaussian or uniform distribution on the pre-activation features of each layer. However, this assumption is not fully justified. We will analyze maxout network gradients, including the higher order moments, and give a rigorous justification for the initialization suggested by Tseran and Montúfar (2021).

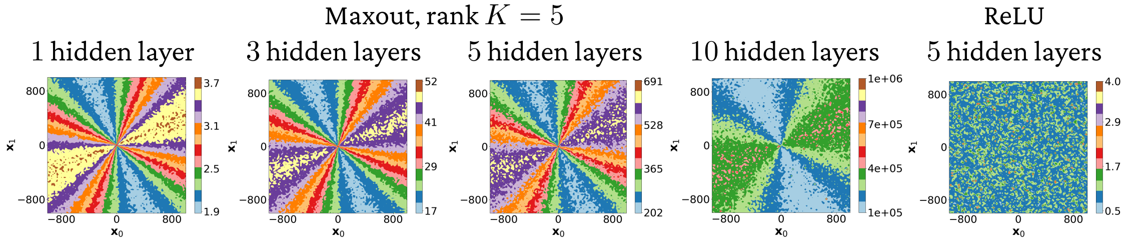
Expected number of linear regions Neural networks with piecewise linear activation functions subdivide their input space into linear regions, i.e., regions over which the computed function is (affine) linear. The number of linear regions serves as a complexity measure to differentiate network architectures (Pascanu et al., 2014; Montufar et al., 2014; Telgarsky, 2015, 2016). The first results on the expected number of linear regions were obtained by Hanin and Rolnick (2019a,b) for ReLU networks, showing that it can be much smaller than the maximum possible number. Tseran and Montúfar (2021) obtained corresponding results for maxout networks. An important factor controlling the bounds in these works is a constant depending on the gradient of the neuron activations with respect to the network input. By studying the input-output Jacobian of maxout networks, we obtain a refined bound for this constant and, consequently, the expected number of linear regions.

Expected curve distortion Another complexity measure is the distortion of the length of an input curve as it passes through a network. Poole et al. (2016) studied the propagation of Riemannian curvature through wide neural networks using a mean-field approach, and later, a related notion of "trajectory length" was considered by Raghu et al. (2017). It was demonstrated that these measures can grow exponentially with the network depth, which was linked to the ability of deep networks to "disentangle" complex representations. Based on these notions, Murray et al. (2022) studies how to avoid rapid convergence of pairwise input correlations, vanishing and exploding gradients. However, Hanin et al. (2021) proved that for a ReLU network with He initialization the length of the curve does not grow with the depth and even shrinks slightly. We establish similar results for maxout networks.

NTK It is known that the neural tangent kernel (NTK) of a finite network can be approximated by its expectation (Jacot et al., 2018). However, for ReLU networks Hanin and Nica (2020a) showed that if both the depth and width tend to infinity, the NTK does not converge to a constant in probability. By studying the expectation of the gradients, we show that similarly to ReLU, the NTK of maxout networks does not converge to a constant when both width and depth are sent to infinity.

Contributions Our contributions can be summarized as follows.

- For **expected gradients**, we derive stochastic order bounds for the directional derivative of the input-output map of a deep fully-connected maxout network (Theorem 4.1) as well as bounds for the moments (Corollary 4.2). Additionally, we derive equality in distribution for the directional derivatives (Theorem 4.3), based on which we also discuss the moments (Remark 4.4) in wide networks. We further derive the moments of the activation length of a fully-connected maxout network (Corollary 4.5).
- We rigorously derive **parameter initialization** guidelines for wide maxout networks preventing vanishing and exploding gradients and formulate architecture recommendations. We experimentally demonstrate that they make it possible to train standard-width deep fully-connected and convolutional maxout networks using simple procedures (such as SGD with momentum and Adam), yielding higher accuracy than other initializations or ReLU networks on image classification tasks.
- We derive **several implications** refining previous bounds on the expected number of linear regions (Corollary 4.6), and new results on length distortion (Corollary 4.7) and the NTK (Corollary 4.9).



Maxout, rank $K = 5$

1 hidden layer 3 hidden layers 5 hidden layers 10 hidden layers

ReLU

5 hidden layers

Figure 4.1: Expectation of the directional derivative of the input-output map $\mathbb{E}[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2]$ for width-2 fully-connected networks with inputs in $\mathbb{R}^2$. For maxout networks, this expectation depends on the input, while for ReLU networks, it does not. Input points $\boldsymbol{x}$ were generated as a grid of 100×100 points in $[-10^3, 10^3]^2$, and $\boldsymbol{u}$ was a fixed vector sampled from the unit sphere. The expectation was estimated based on $10{,}000$ initializations with weights and biases sampled from $N(0, 1)$.

4.2 Preliminaries

Architecture We consider feedforward fully-connected maxout neural networks with n_0 inputs, L hidden layers of widths $n_1, \ldots, n_{L-1}$, and a linear output layer, which implement functions of the form $\mathcal{N} = \psi \circ \phi_{L-1} \circ \cdots \circ \phi_1$. The l-th hidden layer is a function $\phi_l \colon \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ with components $i \in [n_l] := \{1, \ldots, n_l\}$ given by the maximum of $K \geq 2$ trainable affine functions $\phi_{l,i} \colon \mathbb{R}^{n_{l-1}} \to \mathbb{R}; \boldsymbol{x}^{(l-1)} \mapsto \max_{k \in [K]}\{W_{i,k}^{(l)} \boldsymbol{x}^{(l-1)} + \boldsymbol{b}_{i,k}^{(l)}\}$, where $W_{i,k}^{(l)} \in \mathbb{R}^{n_{l-1}}$, $\boldsymbol{b}_{i,k} \in \mathbb{R}$. Here $\boldsymbol{x}^{(l-1)} \in \mathbb{R}^{n_{l-1}}$ denotes the output of the $(l-1)$th layer and $\boldsymbol{x}^{(0)} := \boldsymbol{x}$. We will write $\boldsymbol{x}_{i,k}^{(l)} = W_{i,k}^{(l)} \boldsymbol{x}^{(l-1)} + \boldsymbol{b}_{i,k}^{(l)}$ to denote the kth pre-activation of the ith neuron in the lth layer. Finally $\psi \colon \mathbb{R}^{n_{L-1}} \to \mathbb{R}^{n_L}$ is a linear output layer. We will write $\boldsymbol{\Theta} = \{\boldsymbol{W}, \boldsymbol{b}\}$ for the parameters. Unless stated otherwise, we assume that for each layer, the weights and biases are initialized as i.i.d. samples from a Gaussian distribution with mean 0 and variance c/n_{l-1}, where c is a positive constant. For the linear output layer, the variance is set as $1/n_{L-1}$. We shall study appropriate choices of c. We will use $\|\cdot\|$ to denote the ℓ_2 vector norm. We recall that a real-valued random variable X is said to be smaller than Y in the stochastic order, denoted by $X \leq_{st} Y$, if $\Pr(X > x) \leq \Pr(Y > x)$ for all $x \in \mathbb{R}$. In Section 4.A, we list all the variables and symbols with their definitions, and in Section 4.B, we review basic notions about maxout networks and random variables that we will use in our results.

Input-output Jacobian and activation length We are concerned with the gradients of the outputs with respect to the inputs, $\nabla \mathcal{N}_i(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \mathcal{N}_i$, and with respect to the parameters, $\nabla \mathcal{N}_i(\boldsymbol{\Theta}) = \nabla_{\boldsymbol{\Theta}} \mathcal{N}_i$. In our notation, the argument indicates the variables with respect to which we are taking the derivatives. To study these gradients, we consider the input-output Jacobian $\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x}) = [\nabla \mathcal{N}_1(\boldsymbol{x}), \ldots, \nabla \mathcal{N}_{n_L}(\boldsymbol{x})]^T$. To see the connection to the gradient with respect to the network parameters, consider any loss function $\mathcal{L} : \mathbb{R}^{n_L} \to \mathbb{R}$. A short calculation shows that, for a fixed input $\boldsymbol{x} \in \mathbb{R}^{n_0}$, the derivative of the loss with respect to one of the weights $W_{i,k',j}^{(l)}$ of a maxout unit is

$\left\langle \nabla\mathcal{L}(\mathcal{N}(\boldsymbol{x})), \boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x}_i^{(l)})\right\rangle \boldsymbol{x}_j^{(l-1)}$ if $k' = \operatorname{argmax}_k\{\boldsymbol{x}_{i,k}^{(l)}\}$ and zero otherwise, i.e.

$$\frac{\partial\mathcal{L}(\boldsymbol{x})}{\partial W_{i,k',j}^{(l)}} = C(\boldsymbol{x}, W) \left\|\boldsymbol{J}_{\mathcal{N}}\left(\boldsymbol{x}^{(l)}\right)\boldsymbol{u}\right\| \boldsymbol{x}_j^{(l-1)}, \tag{4.1}$$

where $C(\boldsymbol{x}, W) := \|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x}_i^{(l)})\|^{-1}\langle\nabla\mathcal{L}(\mathcal{N}(\boldsymbol{x})), \boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x}_i^{(l)})\rangle$ and $\boldsymbol{u} = \boldsymbol{e}_i \in \mathbb{R}^{n_l}$. A similar decomposition of the derivative was used by Hanin (2018); Hanin and Rolnick (2018) for ReLU networks. By (4.1) the fluctuation of the gradient norm around its mean is captured by the joint distribution of the squared norm of the directional derivative $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$ and the normalized activation length $A^{(l)} = \|\boldsymbol{x}^{(l)}\|^2/n_l$. We also observe that $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$ is related to the singular values of the input-output Jacobian, which is of interest since a spectrum concentrated around one at initialization can speed up training (Saxe et al., 2014; Pennington et al., 2017, 2018): First, the sum of singular values is $\operatorname{tr}(\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})^T\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})) = \sum_{i=1}^{n_L}\langle\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})^T\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}_i, \boldsymbol{u}_i\rangle = \sum_{i=1}^{n_L}\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}_i\|^2$, where the vectors $\boldsymbol{u}_i$ form an orthonormal basis. Second, using the Stieltjes transform, one can show that singular values of the Jacobian depend on the even moments of the entries of $\boldsymbol{J}_{\mathcal{N}}$ (Hanin, 2018, Section 3.1).

4.3 Results

4.3.1 Bounds on the input-output Jacobian

Theorem 4.1 (Bounds on $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$). *Consider a maxout network with the settings of Section 4.2. Assume that the biases are independent of the weights but otherwise initialized using any approach. Let $\boldsymbol{u} \in \mathbb{R}^{n_0}$ be a fixed unit vector. Then, almost surely, with respect to the parameter initialization, for any input into the network $\boldsymbol{x} \in \mathbb{R}^{n_0}$, the following stochastic order bounds hold:*

$$\frac{1}{n_0}\chi_{n_L}^2\prod_{l=1}^{L-1}\frac{c}{n_l}\sum_{i=1}^{n_l}\xi_{l,i}(\chi_1^2, K) \leq_{st} \|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2 \leq_{st} \frac{1}{n_0}\chi_{n_L}^2\prod_{l=1}^{L-1}\frac{c}{n_l}\sum_{i=1}^{n_l}\Xi_{l,i}(\chi_1^2, K),$$

where $\xi_{l,i}(\chi_1^2, K)$ and $\Xi_{l,i}(\chi_1^2, K)$ are respectively the smallest and largest order statistic in a sample of size K of chi-squared random variables with 1 degree of freedom, independent of each other and of the vectors $\boldsymbol{u}$ and $\boldsymbol{x}$.

The proof is in Section 4.C. It is based on appropriate modifications to the ReLU discussion of Hanin and Nica (2020b); Hanin et al. (2021) and proceeds by writing the Jacobian norm as the product of the layer norms and bounding them with $\min_{k\in[K]}\{\langle W_{i,k}^{(l)}, \boldsymbol{u}^{(l-1)}\rangle^2\}$ and $\max_{k\in[K]}\{\langle W_{i,k}^{(l)}, \boldsymbol{u}^{(l-1)}\rangle^2\}$. Since the product of a Gaussian vector with a unit vector is always Gaussian, the lower and upper bounds are distributed as the smallest and largest order statistics in a sample of size K of chi-squared random variables with 1 degree of freedom. In contrast to ReLU networks, we found that for maxout networks, it is not clear how to obtain equality in distribution involving only independent random variables because of the dependency of the distribution of

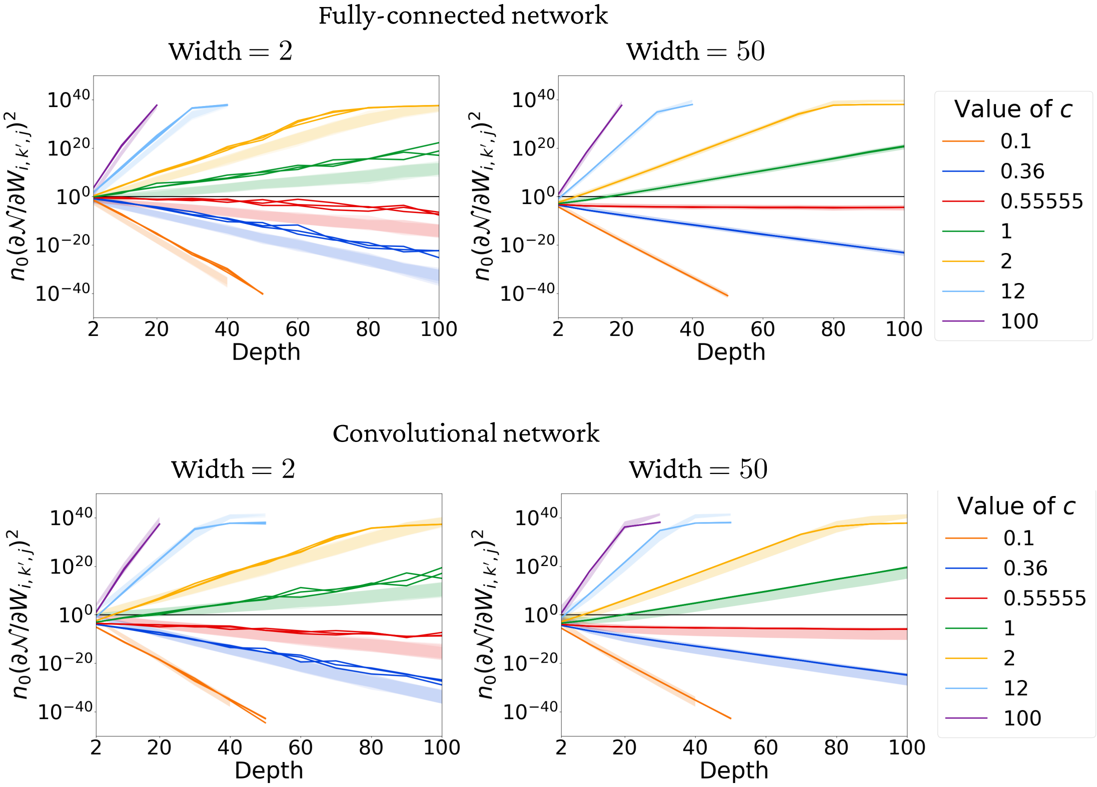


Figure 4.2: Expected value and interquartile range of the squared gradients $n_0(\partial\mathcal{N}/\partial W_{i,k',j})^2$ as a function of depth. Weights are sampled from $N(0, c/\text{fan-in})$ in fully-connected networks and $N(0, c/(k^2 \cdot \text{fan-in}))$, where k is the kernel size, in CNNs. Biases are zero, and the maxout rank K is 5. The gradient is stable in wide fully-connected and convolutional networks with $c = 0.55555$ (red line), the value suggested in Section 4.4. The dark and light blue lines represent the bounds from Corollary 4.2, and equal $1/\mathcal{L} = 0.36$ and $1/\mathcal{S} = 12$. The yellow line corresponds to the ReLU-He initialization. We compute the mean and quartiles from 100 network initializations and a fixed input. The same color lines that are close to each other correspond to 3 different unit-norm network inputs.

$\|\boldsymbol{J}_\mathcal{N}(\boldsymbol{x})\boldsymbol{u}\|^2$ on the network input $\boldsymbol{x}$ and the direction vector $\boldsymbol{u}$ (see Figure 4.1). We discuss this in more detail in Section 4.3.2.

Corollary 4.2 (Bounds on the moments of $\|\boldsymbol{J}_\mathcal{N}(\boldsymbol{x})\boldsymbol{u}\|^2$)**.** *Consider a maxout network with the settings of Section 4.2. Assume that the biases are independent of the weights but otherwise initialized using any approach. Let $\boldsymbol{u} \in \mathbb{R}^{n_0}$ be a fixed unit vector and $\boldsymbol{x} \in \mathbb{R}^{n_0}$ be any input into the network, Then*

(i) $$\frac{n_L}{n_0}(c\mathcal{S})^{L-1} \le \mathbb{E}[\|\boldsymbol{J}_\mathcal{N}(\boldsymbol{x})\boldsymbol{u}\|^2] \le \frac{n_L}{n_0}(c\mathcal{L})^{L-1},$$

(ii) $$\operatorname{Var}\left[\|\boldsymbol{J}_\mathcal{N}(\boldsymbol{x})\boldsymbol{u}\|^2\right] \le \left(\frac{n_L}{n_0}\right)^2 c^{2(L-1)}\left(K^{2(L-1)}\exp\left\{4\left(\sum_{l=1}^{L-1}\frac{1}{n_l K} + \frac{1}{n_L}\right)\right\} - \mathcal{S}^{2(L-1)}\right),$$

(iii) $\mathbb{E}\left[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^{2t}\right] \leq \left(\frac{n_L}{n_0}\right)^t (cK)^{t(L-1)} \exp\left\{t^2\left(\sum_{l=1}^{L-1}\frac{1}{n_l K}+\frac{1}{n_L}\right)\right\},\quad t \in \mathbb{N}$*,*

where the expectation is taken with respect to the distribution of the network weights. The constants $\mathcal{S}$ and $\mathcal{L}$ depend on K and denote the means of the smallest and the largest order statistic in a sample of K chi-squared random variables. For $K = 2, \ldots, 10$, $\mathcal{S} \in [0.02, 0.4]$ and $\mathcal{L} \in [1.6, 4]$. See Table 4.9 in Section 4.D for the exact values.

Notice that for $t \geq 2$, the tth moments of the input-output Jacobian depend on the architecture of the network, but the mean does not (Corollary 4.2), similarly to their behavior in ReLU networks Hanin (2018). We also observe that the upper bound on the tth moments can grow exponentially with the network depth depending on the maxout rank. However, the upper bound on the moments can be tightened provided corresponding bounds for the largest order statistics of the chi-squared distribution.

4.3.2 Distribution of the input-output Jacobian

Here we present the equality in distribution for the input-output Jacobian. It contains dependent variables for the individual layers and thus cannot be readily used to obtain bounds on the moments, but it is particularly helpful for studying the behavior of wide maxout networks.

Theorem 4.3 (Equality in distribution for $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$)**.** *Consider a maxout network with the settings of Section 4.2. Let $\boldsymbol{u} \in \mathbb{R}^{n_0}$ be a fixed unit vector and $\boldsymbol{x} \in \mathbb{R}^{n_0}, \boldsymbol{x} \neq \boldsymbol{0}$ be any input into the network. Then, almost surely, with respect to the parameter initialization, $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$ equals in distribution*

$$\frac{1}{n_0}\chi^2_{n_L}\prod_{l=1}^{L-1}\frac{c}{n_l}\sum_{i=1}^{n_l}\left(v_i\sqrt{1-\cos^2\gamma_{\mathfrak{x}^{(l-1)},\mathfrak{u}^{(l-1)}}}+\Xi_{l,i}(N(0,1),K)\cos\gamma_{\mathfrak{x}^{(l-1)},\mathfrak{u}^{(l-1)}}\right)^2,$$

where v_i and $\Xi_{l,i}(N(0,1),K)$ are independent, $v_i \sim N(0,1)$, $\Xi_{l,i}(N(0,1),K)$ is the largest order statistic in a sample of K standard Gaussian random variables. Here $\gamma_{\mathfrak{x}^{(l)},\mathfrak{u}^{(l)}}$ denotes the angle between $\mathfrak{x}^{(l)} := (\boldsymbol{x}_1^{(l)}, \ldots, \boldsymbol{x}_{n_l}^{(l)}, 1)$ and $\mathfrak{u}^{(l)} := (\boldsymbol{u}_1^{(l)}, \ldots, \boldsymbol{u}_{n_l}^{(l)}, 0)$ in $\mathbb{R}^{n_l+1}$, where $\boldsymbol{u}^{(l)} = \overline{W}^{(l)}\boldsymbol{u}^{(l-1)}/\|\overline{W}^{(l)}\boldsymbol{u}^{(l-1)}\|$ when $\overline{W}^{(l)}\boldsymbol{u}^{(l-1)} \neq 0$ and 0 otherwise, and $\boldsymbol{u}^{(0)} = \boldsymbol{u}$. The matrices $\overline{W}^{(l)}$ consist of rows $\overline{W}_i^{(l)} = W_{i,k'}^{(l)} \in \mathbb{R}^{n_{l-1}}$, where $k' = \operatorname{argmax}_{k\in[K]}\{W_{i,k}^{(l)}\boldsymbol{x}^{(l-1)} + b_{i,k}^{(l)}\}$.

This statement is proved in Section 4.E. The main strategy is to construct an orthonormal basis $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n_l})$, where $\boldsymbol{b}_1 := \mathfrak{x}^{(l)}/\|\mathfrak{x}^{(l)}\|$, which allows us to express the layer gradient depending on the angle between $\mathfrak{x}^{(l)}$ and $\mathfrak{u}^{(l)}$.

Remark 4.4 (Wide networks)**.** By Theorem 4.3, in a maxout network the distribution of $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$ depends on the $\cos\gamma_{\mathfrak{x}^{(l-1)},\mathfrak{u}^{(l-1)}}$, which changes as the network gets wider or deeper. Since independent and isotropic random vectors in high-dimensional spaces tend to be almost orthogonal, we expect that the cosine will be close to zero for the earlier layers of wide networks, and individual units will behave similarly to squared standard Gaussians. In wide and deep networks, if the network parameters are sampled from $N(0, c/n_{l-1})$, $c = 1/\mathcal{M}$, and $K \geq 3$, we expect that $|\cos \gamma_{\mathbf{x}^{(l)},\mathbf{u}^{(l)}}| \approx 1$ for the later layers of deep networks and individual units will behave more as the squared largest order statistics. Here $\mathcal{M}$ is the second moment of the largest order statistic in a sample of size K of standard Gaussian random variables. Based on this, for deep and wide networks, we can expect that

$$\mathbb{E}[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2] \approx \frac{n_L}{n_0}(c\mathcal{M})^{L-1} = \frac{n_L}{n_0}. \tag{4.2}$$

This intuition is discussed in more detail in Section 4.E. According to (4.2), we expect that the expected gradient magnitude will be stable with depth when an appropriate initialization is used. See Figure 4.2 for a numerical evaluation of the effects of the width and depth on the gradients.

4.3.3 Activation length

To have a full picture of the derivatives in (4.1), we consider the activation length. The full version and proof of Corollary 4.5 are in Section 4.F. The proof is based on Theorem 4.3, replacing $\mathbf{u}$ with $\mathbf{x}/\|\mathbf{x}\|$.

Corollary 4.5 (Moments of the normalized activation length). *Consider a maxout network with the settings of Section 4.2. Let $\boldsymbol{x} \in \mathbb{R}^{n_0}$ be any input into the network. Then, for the moments of the normalized activation length $A^{(l')}$ of the l'th layer we have*

Mean: $$\mathbb{E}\left[A^{(l')}\right] = \|\mathbf{x}^{(0)}\|^2 \frac{1}{n_0}\left(c\mathcal{M}\right)^{l'} + \sum_{j=2}^{l'}\left(\frac{1}{n_{j-1}}\left(c\mathcal{M}\right)^{l'-j+1}\right),$$

Moments of order $t \geq 2$: $$G_1\left(\left(c\mathcal{M}\right)^{tl'}\right) \leq \mathbb{E}\left[\left(A^{(l')}\right)^t\right] \leq G_2\left(\left(cK\right)^{tl'} \exp\left\{\sum_{l=1}^{l'} \frac{t^2}{n_l K}\right\}\right).$$

The expectation is taken with respect to the distribution of the network weights and biases, and $\mathcal{M}$ is a constant depending on K that can be computed approximately, see Table 4.9 for the values for $K = 2, \dots, 10$. See Section 4.F for the variance bounds and details on functions G_1, G_2.

We could obtain an exact expression for the mean activation length for a finitely wide maxout network since its distribution only depends on the norm of the input, while this is not the case for the input-output Jacobian (Sections 4.3.1 and 4.3.2). We observe that the variance and the tth moments, $t \geq 2$, have an exponential dependence on the network architecture, including the maxout rank, whereas the mean does not, similarly to the input-output Jacobian (Corollary 4.2). Such behavior also occurs for ReLU networks (Hanin and Rolnick, 2018). See Figure 4.9 in Section 4.F for an evaluation of the result.

Table 4.1: Recommended values for the constant c for different maxout ranks K based on Section 4.4.

K	2	3	4	5	6	7	8	9	10
c	1	0.78391	0.64461	0.55555	0.49462	0.45039	0.41675	0.39023	0.36872

4.4 Implications to initialization and network architecture

We now aim to find initialization approaches and architectures that can avoid exploding and vanishing gradients. We take the annealed exploding and vanishing gradients definition from Hanin (2018) as a starting point for such investigation for maxout networks. Formally, we require

$$\mathbb{E}\left[\left(\frac{\partial \mathcal{L}(\boldsymbol{x})}{\partial W^{(l)}_{i,k',j}}\right)^2\right] = \Theta(1), \quad \mathrm{Var}\left[\left(\frac{\partial \mathcal{L}(\boldsymbol{x})}{\partial W^{(l)}_{i,k',j}}\right)^2\right] = \Theta(1),$$

$$\sup_{l\geq 1} \mathbb{E}\left[\left(\frac{\partial \mathcal{L}(\boldsymbol{x})}{\partial W^{(l)}_{i,k',j}}\right)^{2t}\right] < \infty, \qquad \forall t \geq 3,$$

where the expectation is with respect to the weights and biases. Based on (4.1) these conditions can be attained by ensuring that similar conditions hold for $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$ and $A^{(l)}$.

Initialization recommendations Based on Corollary 4.2, the mean of $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$ can be stabilized for some $c \in [1/\mathcal{L}, 1/\mathcal{S}]$. However, Theorem 4.3 shows that $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$ depends on the input into the network. Hence, we expect that there is no value of c stabilizing input-output Jacobian for every input simultaneously. Nevertheless, based on Remark 4.4, for wide and deep maxout networks, $\mathbb{E}[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2] \approx n_L/n_0$ if $c = 1/\mathcal{M}$, and the mean becomes stable. While Remark 4.4 does not include maxout rank $K = 2$, the same recommendation can be obtained for it using the approach from He et al. (2015), see Sun et al. (2018). Moreover, according to Corollary 4.5, the mean of the normalized activation length remains stable for different network depths if $c = 1/\mathcal{M}$. Hence, we recommend $c = 1/\mathcal{M}$ as an appropriate value for initialization. See Table 4.1 for the numerical value of c for $K = 2, \dots, 10$. We call this type of initialization, when the parameters are sampled from $N(0, c/\text{fan-in})$, $c = 1/\mathcal{M}$, “maxout initialization”. We note that this matches the previous recommendation from Tseran and Montúfar (2021), which we now derived rigorously.

Architecture recommendations In Corollaries 4.2 and 4.5 the upper bound on the moments $t \geq 2$ of $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$ and $A^{(l)} = \|\boldsymbol{x}^{(l)}\|^2/n_l$ can grow exponentially with the depth depending on the values of $(cK)^L$ and $\sum_{l=1}^{L-1} 1/(n_l K)$. Hence, we recommend choosing the widths such that $\sum_{l=1}^{L-1} 1/(n_l K) \leq 1$, which holds, e.g., if $n_l \geq L/K, \forall l = 1, \dots, L-1$, and choosing a moderate value of the maxout rank K. However, the upper bound can still tend to infinity for the high-order moments. From Remark 4.4, it follows that for $K \geq 3$ to have a stable initialization independent of the network input, a maxout network has to be deep and wide. Experimentally, we observe that for 100-neuron wide networks with $K = 3$, the absolute value of the cosine that determines the initialization stability converges to 1 at around 60 layers, and for $K = 4, 5$, at around 30 layers. See Figure 4.6 in Section 4.E. To sum up, we recommend working with deep and wide maxout networks with widths satisfying $\sum_{l=1}^{L-1} 1/(n_l K) \leq 1$, and choosing the maxout-rank not too small nor too large, e.g., $K = 5$.

4.5 Implications to expressivity and NTK

With Theorems 4.1 and 4.3 in place, we can now obtain maxout versions of the several types of results that previously have been derived only for ReLU networks.

4.5.1 Expected number of linear regions of maxout networks

For a piece-wise linear function $f\colon \mathbb{R}^{n_0} \to \mathbb{R}$, a linear region is defined as a maximal connected subset of $\mathbb{R}^{n_0}$ on which f has a constant gradient. Tseran and Montúfar (2021) and Hanin and Rolnick (2019b) established upper bounds on the expected number of linear regions of maxout and ReLU networks, respectively. One of the key factors controlling these bounds is C_{grad}, which is any upper bound on $(\sup_{\boldsymbol{x}\in\mathbb{R}^{n_0}} \mathbb{E}[\|\nabla\zeta_{z,k}(\boldsymbol{x})\|^t])^{1/t}$, for any $t \in \mathbb{N}$ and $z = 1, \ldots, N$. Here $\zeta_{z,k}$ is the kth pre-activation feature of the zth unit in the network, N is the total number of units, and the gradient is with respect to the network input. Using Corollary 4.2, we obtain a value for C_{grad} for maxout networks, which remained an open problem in the work of Tseran and Montúfar (2021). The proof of Corollary 4.6 and the resulting refined bound on the expected number of linear regions are in Section 4.G.

Corollary 4.6 (Value for C_{grad})**.** *Consider a maxout network with the settings of Section 4.2. Assume that the biases are independent of the weights but otherwise initialized using any approach. Consider the pre-activation feature $\zeta_{z,k}$ of a unit $z = 1, \ldots, N$. Then, for any $t \in \mathbb{N}$,*

$$\left(\sup_{\boldsymbol{x}\in\mathbb{R}^{n_0}} \mathbb{E}\left[\|\nabla\zeta_{z,k}(x)\|^t\right]\right)^{\frac{1}{t}} \leq n_0^{-\frac{1}{2}} \max\left\{1, (cK)^{\frac{L-1}{2}}\right\} \exp\left\{\frac{t}{2}\left(\sum_{l=1}^{L-1} \frac{1}{n_l K} + 1\right)\right\}.$$

The value of C_{grad} given in Corollary 4.6 grows as $O((cK)^{L-1} \exp\{t \sum_{l=1}^{L-1} 1/(n_l K)\})$. The first factor grows exponentially with the network depth if $cK > 1$. This is the case when the network is initialized as in Section 4.4. However, since K is usually a small constant and $c \leq 1$, $cK \geq 1$ is a small constant. The second factor grows exponentially with the depth if $\sum_{l=1}^{L-1} 1/(n_l K) > 1$. Hence, the exponential growth can be avoided if $n_l \geq (L-1)/K, \forall l = 1, \ldots, L-1$.

4.5.2 Expected curve length distortion

Let M be a smooth 1-dimensional curve in $\mathbb{R}^{n_0}$ of length $\text{len}(M)$ and $\mathcal{N}(M) \subseteq \mathbb{R}^{n_L}$ the image of M under the map $\boldsymbol{x} \mapsto \mathcal{N}(\boldsymbol{x})$. We are interested in the length distortion of M, defined as $\text{len}(\mathcal{N}(M))/\text{len}(M)$. Using the results from Section 4.3.1, observing that the input-output Jacobian of maxout networks is well defined almost everywhere, and following Hanin et al. (2021), we obtain the following corollary. The proof is in Section 4.H.

Corollary 4.7 (Expected curve length distortion)**.** *Consider a maxout network with the settings of Section 4.2. Assume that the biases are independent of the weights but otherwise initialized using any approach. Let M be a smooth 1-dimensional curve of unit length in $\mathbb{R}^{n_0}$. Then, the following upper bounds on the moments of* $\text{len}(\mathcal{N}(M))$ *hold:*

$$\begin{aligned}
&\mathbb{E}\left[\text{len}(\mathcal{N}(M))\right] \leq \left(\frac{n_L}{n_0}\right)^{\frac{1}{2}} (c\mathcal{L})^{\frac{L-1}{2}}, \\
&\text{Var}\left[\text{len}(\mathcal{N}(M))\right] \leq \frac{n_L}{n_0} (c\mathcal{L})^{L-1}, \\
&\mathbb{E}\left[\text{len}(\mathcal{N}(M))^t\right] \leq \left(\frac{n_L}{n_0}\right)^{\frac{t}{2}} (cK)^{\frac{t(L-1)}{2}} \exp\left\{\frac{t^2}{2}\left(\sum_{l=1}^{L-1} \frac{1}{n_l K} + \frac{1}{n_L}\right)\right\},
\end{aligned}$$

where $\mathcal{L}$ is a constant depending on K, see Table 4.9 in Section 4.D for values for $K = 2, \ldots, 10$.

Remark 4.8 (Expected curve length distortion in wide maxout networks)**.** If the network is initialized according to Section 4.4, using Remark 4.4 and repeating the steps of the proof of Corollary 4.7, we get $\mathbb{E}\left[\text{len}(\mathcal{N}(M))\right] \lesssim (n_L/n_0)^{1/2}$ and $\text{Var}\left[\text{len}(\mathcal{N}(M))\right] \approx n_L/n_0$.

Hence, similarly to ReLU networks, wide maxout networks, if initialized to keep the gradients stable, have low expected curve length distortion at initialization. However, we cannot conclude whether the curve length shrinks. For narrow networks, the upper bound does not exclude the possibility that the expected distortion grows exponentially with the network depth, depending on the initialization.

4.5.3 On-diagonal NTK

We denote the on-diagonal NTK with $K_{\mathcal{N}}(\boldsymbol{x}, \boldsymbol{x}) = \sum_i (\partial \mathcal{N}(\boldsymbol{x})/\partial \theta_i)^2$. In Section 4.I we show:

Corollary 4.9 (On-diagonal NTK)**.** *Consider a maxout network with the settings of Section 4.2. Assume that $n_L = 1$ and that the biases are initialized to zero and are not trained. Assume that $\mathcal{S} \leq c \leq \mathcal{L}$, where*

Table 4.2: Accuracy on the test set for networks trained using SGD with Nesterov momentum. Observe that maxout networks initialized with the maxout or max-pooling initialization perform significantly better than the ones initialized with other initializations and better or comparably to ReLU networks.

	MAXOUT				RELU
VALUE OF c	Small value 0.1	Max-pooling init Section 4.6 (Ours) 0.55 & 0.27	Maxout init Section 4.4 (Ours) 0.55555	Naive ReLU He 2	He init 2
			FULLY-CONNECTED		
MNIST	$11.35^{\pm 0.00}$	—	$\mathbf{97.8^{\pm 0.15}}$	$53.22^{\pm 24.08}$	$97.43^{\pm 0.06}$
Iris	$30.00^{\pm 0.00}$	—	$\mathbf{91.67^{\pm 3.73}}$	$82.5^{\pm 4.93}$	$\mathbf{91.67^{\pm 3.73}}$
			CONVOLUTIONAL		
MNIST	$11.35^{\pm 0.00}$	$99.58^{\pm 0.03}$	$\mathbf{99.59^{\pm 0.04}}$	$98.02^{\pm 0.21}$	$99.49^{\pm 0.04}$
CIFAR-10	$10.00^{\pm 0.00}$	$\mathbf{91.7^{\pm 0.17}}$	$91.21^{\pm 0.13}$	$44.84^{\pm 0.69}$	$90.12^{\pm 0.25}$
CIFAR-100	$1.00^{\pm 0.00}$	$65.33^{\pm 0.27}$	$\mathbf{65.39^{\pm 0.39}}$	$12.02^{\pm 0.8}$	$59.59^{\pm 0.82}$
Fashion MNIST	$10.00^{\pm 0.00}$	$\mathbf{93.55^{\pm 0.13}}$	$93.49^{\pm 0.13}$	$81.56^{\pm 0.15}$	$93.28^{\pm 0.11}$
SVHN	$19.59^{\pm 0.00}$	$97.3^{\pm 0.04}$	$\mathbf{97.78^{\pm 0.02}}$	$50.97^{\pm 1.71}$	$96.74^{\pm 0.03}$

the constants $\mathcal{S}, \mathcal{L}$ are as specified in Table 4.9. Then,

$$\|\boldsymbol{\mathfrak{x}}^{(0)}\|^2 \frac{(c\mathcal{S})^{L-2}}{n_0} P \leq \mathbb{E}[K_{\mathcal{N}}(\boldsymbol{x},\boldsymbol{x})] \leq \|\boldsymbol{\mathfrak{x}}^{(0)}\|^2 \frac{(c\mathcal{L})^{L-2}\mathcal{M}^{L-1}}{n_0} P,$$

$$\mathbb{E}[K_{\mathcal{N}}(\boldsymbol{x},\boldsymbol{x})^2] \leq 2PP_W(cK)^{2(L-2)} \frac{\|\boldsymbol{\mathfrak{x}}^{(0)}\|^4}{n_0^2} \exp\left\{\sum_{j=1}^{L-1} \frac{4}{n_j K} + 4\right\},$$

where $P = \sum_{l=0}^{L-1} n_l$, $P_W = \sum_{l=0}^{L} n_l n_{l-1}$, and $\mathcal{M}$ is as specified in Table 4.9.

By Corollary 4.9, $\mathbb{E}[K_{\mathcal{N}}(\boldsymbol{x},\boldsymbol{x})^2]/(\mathbb{E}[K_{\mathcal{N}}(\boldsymbol{x},\boldsymbol{x})])^2$ is in $O((P_W/P)C^L \exp\{\sum_{l=1}^{L} 1/(n_l K)\})$, where C depends on $\mathcal{L}, \mathcal{M}$ and K. Hence, if widths $n_1, \ldots, n_{L-1}$ and depth L tend to infinity, this upper bound does not converge to a constant, suggesting that the NTK might not converge to a constant in probability. This is in line with previous results for ReLU networks by Hanin and Nica (2020a).

4.6 Experiments

We check how the initialization proposed in Section 4.4 affects the network training. This initialization was first proposed heuristically by Tseran and Montúfar (2021), where it was tested for 10-layer fully-connected networks with an MNIST experiment. We consider both fully-connected

Table 4.3: Accuracy on the test set for the networks trained with Adam. Observe that maxout networks initialized with the maxout or max-pooling initialization perform better or comparably to ReLU networks, while maxout networks initialized with ReLU-He converge slower and perform worse.

		MAXOUT			RELU
		Max-pooling init Section 4.6 (Ours)	Maxout init Section 4.4 (Ours)	Naive ReLU He	He init
VALUE OF c		0.55 & 0.27	0.55555	2	2
		FULLY-CONNECTED			
MNIST	1/10 epochs	—	**$97.56^{\pm 0.18}$**	$97.40^{\pm 0.30}$	$96.72^{\pm 0.64}$
	2/10 epochs	—	**$98.10^{\pm 0.09}$**	$97.97^{\pm 0.12}$	$97.54^{\pm 0.16}$
	All epochs	—	$98.12^{\pm 0.10}$	**$98.13^{\pm 0.09}$**	$97.37^{\pm 0.08}$
		CONVOLUTIONAL			
MNIST	1/10 epochs	$99.06^{\pm 0.15}$	$98.59^{\pm 0.58}$	$98.54^{\pm 0.52}$	**$99.14^{\pm 0.32}$**
	2/10 epochs	$99.39^{\pm 0.13}$	$98.51^{\pm 0.25}$	$99.17^{\pm 0.13}$	**$99.41^{\pm 0.05}$**
	All epochs	**$99.53^{\pm 0.04}$**	$99.47^{\pm 0.07}$	$99.47^{\pm 0.04}$	$99.45^{\pm 0.06}$
Fashion MNIST	1/10 epochs	$92.04^{\pm 0.29}$	$92.35^{\pm 0.12}$	$87.95^{\pm 0.33}$	**$92.45^{\pm 0.41}$**
	2/10 epochs	$92.61^{\pm 0.22}$	**$92.85^{\pm 0.21}$**	$90.35^{\pm 0.38}$	$92.71^{\pm 0.25}$
	All epochs	**$93.57^{\pm 0.17}$**	$93.45^{\pm 0.10}$	$91.63^{\pm 0.36}$	$92.98^{\pm 0.13}$
CIFAR-10	1/10 epochs	**$88.25^{\pm 0.49}$**	$87.31^{\pm 0.51}$	$74.37^{\pm 0.37}$	$85.95^{\pm 0.30}$
	2/10 epochs	**$88.79^{\pm 0.72}$**	$87.96^{\pm 0.75}$	$81.94^{\pm 0.34}$	$87.12^{\pm 0.23}$
	All epochs	**$91.33^{\pm 0.31}$**	$91.06^{\pm 0.22}$	$85.23^{\pm 0.20}$	$87.70^{\pm 0.10}$
CIFAR-100	1/10 epochs	$50.30^{\pm 3.34}$	**$53.43^{\pm 1.08}$**	$19.22^{\pm 0.51}$	$50.39^{\pm 0.91}$
	2/10 epochs	$57.54^{\pm 1.64}$	**$57.65^{\pm 0.75}$**	$33.21^{\pm 0.51}$	$51.34^{\pm 0.51}$
	All epochs	**$65.33^{\pm 1.26}$**	$61.96^{\pm 0.58}$	$37.58^{\pm 0.23}$	$52.95^{\pm 0.30}$

and convolutional neural networks and run experiments for MNIST (LeCun and Cortes, 2010), Iris (Fisher, 1936), Fashion MNIST (Xiao et al., 2017), SVHN (Netzer et al., 2011), CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009). Fully connected networks have 21 layers and CNNs have a VGG-19-like architecture (Simonyan and Zisserman, 2015) with 20 or 16 layers depending on the input size, all with maxout rank 5. Weights are sampled from $N(0, c/\text{fan-in})$ in fully-connected networks and $N(0, c/(k^2 \cdot \text{fan-in}))$ in CNNs of kernel size k. The biases are initialized to zero. We report the mean and std of 4 runs.

We use plain deep networks without any kind of modifications or pre-training. We do not use normalization techniques, such as batch normalization (Ioffe and Szegedy, 2015), since this would obscure the effects of the initialization. Because of this, our results are not necessarily state-of-the-art. More details on the experiments are given in Section 4.J, and the implementation is made available at `https://github.com/hanna-tseran/maxout_expected_gradients`.

Max-pooling initialization To account for the maximum in max-pooling layers, a maxout layer appearing after a max-pooling layer is initialized as if its maxout rank was $K \times m^2$, where m^2 is the max-pooling window size. For example, we used $K = 5$ and $m^2 = 4$, resulting in $c = 0.26573$ for such maxout layers. All other layers are initialized according to Section 4.4. We observe that max-pooling initialization often leads to slightly higher accuracy.

Results for SGD with momentum Table 4.2 reports test accuracy for networks trained using SGD with Nesterov momentum. We compare ReLU and maxout networks with different initializations: maxout, max-pooling, small value $c = 0.1$, and He $c = 2$. We observe that maxout and max-pooling initializations allow training deep maxout networks and obtaining better accuracy than ReLU networks, whereas performance is significantly worse or training does not progress for maxout networks with other initializations.

Results for Adam Table 4.3 reports test accuracy for networks trained using Adam (Kingma and Ba, 2015). We compare ReLU and maxout networks with the following initializations: maxout, max-pooling, and He $c = 2$. We observe that, compared to He initialization, maxout and max-pooling initializations lead to faster convergence and better test accuracy. Compared to ReLU networks, maxout networks have better or comparable accuracy if maxout or max-pooling initialization is used.

4.7 Discussion

We study the gradients of maxout networks with respect to the parameters and the inputs by analyzing a directional derivative of the input-output map. We observe that the distribution of the input-output Jacobian of maxout networks depends on the network input (in contrast to ReLU networks), which can complicate the stable initialization of maxout networks. Based on bounds on the moments, we derive an initialization that provably avoids vanishing and exploding gradients in wide networks. Experimentally, we show that, compared to other initializations, the suggested approach leads to better performance for fully connected and convolutional deep networks of standard width trained with SGD or Adam and better or similar performance compared to ReLU networks. Additionally, we refine previous upper bounds on the expected number of linear regions. We also derive results for the expected curve length distortion, observing that it does not grow exponentially with the depth in wide networks. Furthermore, we obtain bounds on the maxout NTK, suggesting that it might not converge to a constant when both the width and depth are large. These contributions enhance the applicability of maxout networks and add to the theoretical exploration of activation functions beyond ReLU.

Limitations Even though our proposed initialization is optimal in the sense of the criteria specified at the beginning of Section 4.4, our results are applicable only when the weights are sampled from $N(0, c/\text{fan-in})$ for some c. Further, we derived theoretical results only for fully-connected networks. Our experiments indicate that they also hold for CNNs: Figure 4.2 demonstrates that gradients behave according to the theory for fully connected and convolutional networks, and Tables 4.2 and 4.3 show improvement in CNNs performance under the initialization suggested in Section 4.4. However, we have yet to conduct a theoretical analysis of CNNs.

Future work In future work, we would like to obtain more general results in settings involving multi-argument functions, such as aggregation functions in graph neural networks, and investigate the effects that initialization strategies stabilizing the initial gradients have at later stages of training.

Proofs and experiment details

Proofs and experiment details are organized as follows.

- 4.A Notation
- 4.B Basics
- 4.C Bounds for the input-output Jacobian norm $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$
- 4.D Moments of the input-output Jacobian norm $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$
- 4.E Equality in distribution for the input-output Jacobian norm and wide network results
- 4.F Activation length
- 4.G Expected number of linear regions
- 4.H Expected curve length distortion
- 4.I NTK
- 4.J Experiment details and additional experiments

4.A Notation

We use the following notation in the paper.

4.A.1 Variables

Network definition

$\mathcal{N}$	network
L	number of the network layers
l	index of a layer
n_0	input dimension

n_l	width of the lth layer
K	maxout rank
k	index of a pre-activation feature, $k = 1, \dots, K$
k'	argmax of the collection of pre-activation features, $k' = \text{argmax}_k \{\boldsymbol{x}_{i,k}^{(l)}\}$
ϕ_l	function implemented by the lth hidden layer, $\phi_l \colon \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$
ψ	linear output layer, $\psi \colon \mathbb{R}^{n_{L-1}} \to \mathbb{R}^{n_L}$
$\boldsymbol{W}$	collection of all network weights
$\boldsymbol{b}$	collection of all network biases
$\boldsymbol{\Theta}$	collection of all network parameters, $\boldsymbol{\Theta} = \{\boldsymbol{W}, \boldsymbol{b}\}$
θ_i	ith network parameter; here $i = 1, \dots, \lvert\boldsymbol{\Theta}\rvert$
$W_{i,k}^{(l)}$	network weights of the kth pre-activation function of the ith neuron in the lth layer, $W_{i,k}^{(l)} \in \mathbb{R}^{n_{l-1}}$
$\boldsymbol{b}_{i,k}$	network bias of the kth pre-activation function of the ith neuron in the lth layer, $\boldsymbol{b}_{i,k} \in \mathbb{R}$
$\boldsymbol{x}, \boldsymbol{x}^{(0)}$	network input, $\boldsymbol{x} \in \mathbb{R}^{n_0}$, $\boldsymbol{x} = \boldsymbol{x}^{(0)}$
$\boldsymbol{x}^{(l)}$	output of the lth layer, $\boldsymbol{x}^{(l)} \in \mathbb{R}^{n_l}$
$\boldsymbol{x}_{i,k}^{(l)}$	kth pre-activation of the ith neuron in the lth layer, $\boldsymbol{x}_{i,k}^{(l)} = W_{i,k}^{(l)} \boldsymbol{x}^{(l-1)} + \boldsymbol{b}_{i,k}^{(l)}$
N	total number of the network units
z	index of a neuron in the network, $z = 1, \dots, N$
$\zeta_{z,k}$	kth pre-activation feature of the zth unit in the network

Network initialization

$N(\mu, \sigma^2)$	Gaussian distribution with mean μ and variance σ^2
c	a positive constant; we assume that the weights and biases for each hidden layer are initialized as i.i.d. samples from $N(0, c/n_{l-1})$

Network optimization

$\mathcal{L}$	loss function, $\mathcal{L} : \mathbb{R}^{n_L} \to \mathbb{R}$
$\nabla \mathcal{N}_i(\boldsymbol{x})$	gradients of the network outputs with respect to the inputs, $\nabla \mathcal{N}_i(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \mathcal{N}_i$
$\nabla \mathcal{N}_i(\boldsymbol{\Theta}), \nabla_{\boldsymbol{\Theta}} \mathcal{N}_i$	gradients of the network outputs with respect to the parameters, $\nabla \mathcal{N}_i(\boldsymbol{\Theta}) = \nabla_{\boldsymbol{\Theta}} \mathcal{N}_i$
$\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})$	input-output Jacobian of the network, $\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x}) = [\nabla \mathcal{N}_1(\boldsymbol{x}), \dots, \nabla \mathcal{N}_{n_L}(\boldsymbol{x})]^T$

Variables appearing in the results

$\boldsymbol{u}$	a fixed unit vector, $\boldsymbol{u} \in \mathbb{R}^{n_0}$
$A^{(l)}$	normalized activation length of the lth layer, $A^{(l)} = \|\boldsymbol{x}^{(l)}\|^2/n_l$
t	order of a moment
$\xi_{l,i}(\chi_1^2, K)$	the smallest order statistic in a sample of size K of chi-squared random variables with 1 degree of freedom
$\Xi_{l,i}(\chi_1^2, K)$	the largest order statistic in a sample of size K of chi-squared random variables with 1 degree of freedom
$\Xi_{l,i}(N(0,1), K)$	the largest order statistic in a sample of size K of standard Gaussian random variables
$\mathcal{S}$	mean of the smallest order statistic in a sample of K chi-squared random variables; see Table 4.9 for the exact values
$\mathcal{L}$	mean of the largest order statistic in a sample of K chi-squared random variables; see Table 4.9 for the exact values
$\mathcal{M}$	the second moment of the largest order statistic in a sample of size K of standard Gaussian random variables; see Table 4.9 for the exact values
v_i	standard Gaussian random variable $v_i \sim N(0,1)$
$\overline{W}^{(l)}$	matrices consisting of rows $\overline{W}_i^{(l)} = W_{i,k'}^{(l)} \in \mathbb{R}^{n_{l-1}}$, where $k' = \operatorname{argmax}_{k \in [K]}\{W_{i,k}^{(l)}\boldsymbol{x}^{(l-1)} + b_{i,k}^{(l)}\}$
$\boldsymbol{u}^{(l)}$	$\boldsymbol{u}^{(l)} = \overline{W}^{(l)}\boldsymbol{u}^{(l-1)}/\|\overline{W}^{(l)}\boldsymbol{u}^{(l-1)}\|$ when $\overline{W}^{(l)}\boldsymbol{u}^{(l-1)} \neq 0$ and 0 otherwise; $\boldsymbol{u}^{(0)} = \boldsymbol{u}$
$\gamma_{\mathfrak{x}^{(l)}, \mathbf{u}^{(l)}}$	angle between $\mathfrak{x}^{(l)} := (\boldsymbol{x}_1^{(l)}, \ldots, \boldsymbol{x}_{n_l}^{(l)}, 1)$ and $\mathbf{u}^{(l)} := (\boldsymbol{u}_1^{(l)}, \ldots, \boldsymbol{u}_{n_l}^{(l)}, 0)$ in $\mathbb{R}^{n_l+1}$
C_{grad}	any upper bound on $(\sup_{\boldsymbol{x} \in \mathbb{R}^{n_0}} \mathbb{E}[\|\nabla \zeta_{z,k}(\boldsymbol{x})\|^t])^{1/t}$, for any $t \in \mathbb{N}$; here the gradient is with respect to the network input
M	smooth 1-dimensional curve of unit length in $\mathbb{R}^{n_0}$
$\mathcal{N}(M)$	image of the curve M under the map $\boldsymbol{x} \mapsto \mathcal{N}(\boldsymbol{x}), \mathcal{N}(M) \subseteq \mathbb{R}^{n_L}$
$K_{\mathcal{N}}(\boldsymbol{x}, \boldsymbol{x})$	on-diagonal NTK, $K_{\mathcal{N}}(\boldsymbol{x}, \boldsymbol{x}) = \sum_i (\partial \mathcal{N}(\boldsymbol{x})/\partial \theta_i)^2$

4.A.2 Symbols

$[n]$	$\{1, \ldots, n\}$
$\|\cdot\|$	ℓ_2 vector norm
$\leq_{st}$ ($\geq_{st}$)	smaller (larger) in the stochastic order; see Section 4.B.2 for the definition
$\overset{d}{=}$	equality in distribution; see Section 4.B.2 for the definition
$\operatorname{len}(\cdot)$	length of a curve

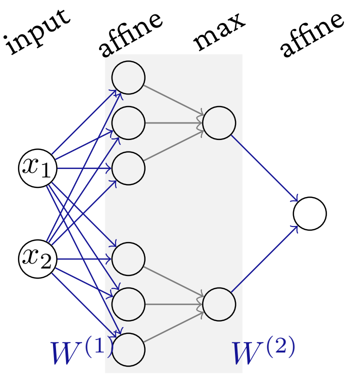


Figure 4.3: Illustration of a simple maxout network with two input units, one hidden layer consisting of two maxout units of rank 3, and an affine output layer with a single output unit.

4.B Basics

4.B.1 Basics on maxout networks

4.B.1.1 Definition

As mentioned in the introduction, a rank-K maxout unit computes the maximum of K real-valued affine functions. Concretely, a rank-K maxout unit with n inputs implements a function

$$\mathbb{R}^n \to \mathbb{R}; \quad \boldsymbol{x} \mapsto \max_{k\in[K]}\{\langle W_k, \boldsymbol{x}\rangle + b_k\},$$

where $W_k \in \mathbb{R}^n$ and $b_k \in \mathbb{R}$, $k \in [K] := \{1, \dots, K\}$, are trainable weights and biases. The K arguments of the maximum are called the pre-activation features of the maxout unit. A rank-K maxout unit can be regarded as a composition of an affine map with K outputs and a maximum gate. A layer corresponds to the parallel computation of several such units. For instance, a layer with n inputs and m maxout units computes functions of the form

$$\mathbb{R}^n \to \mathbb{R}^m; \quad \boldsymbol{x} \mapsto \begin{bmatrix} \max_{k\in[K]}\{\langle W^{(1)}_{1,k}, \boldsymbol{x}\rangle + \boldsymbol{b}^{(1)}_{1,k}\} \\ \vdots \\ \max_{k\in[K]}\{\langle W^{(1)}_{m,k}, \boldsymbol{x}\rangle + \boldsymbol{b}^{(1)}_{m,k}\} \end{bmatrix},$$

where now $W^{(1)}_{i,k}$ and $\boldsymbol{b}^{(1)}_{i,k}$ are the weights and biases of the kth pre-activation feature of the ith maxout unit in the first layer. The situation is illustrated in Figure 4.3 for the case of a network with two inputs, one layer with two maxout units of rank three, and one output layer with a single output unit.

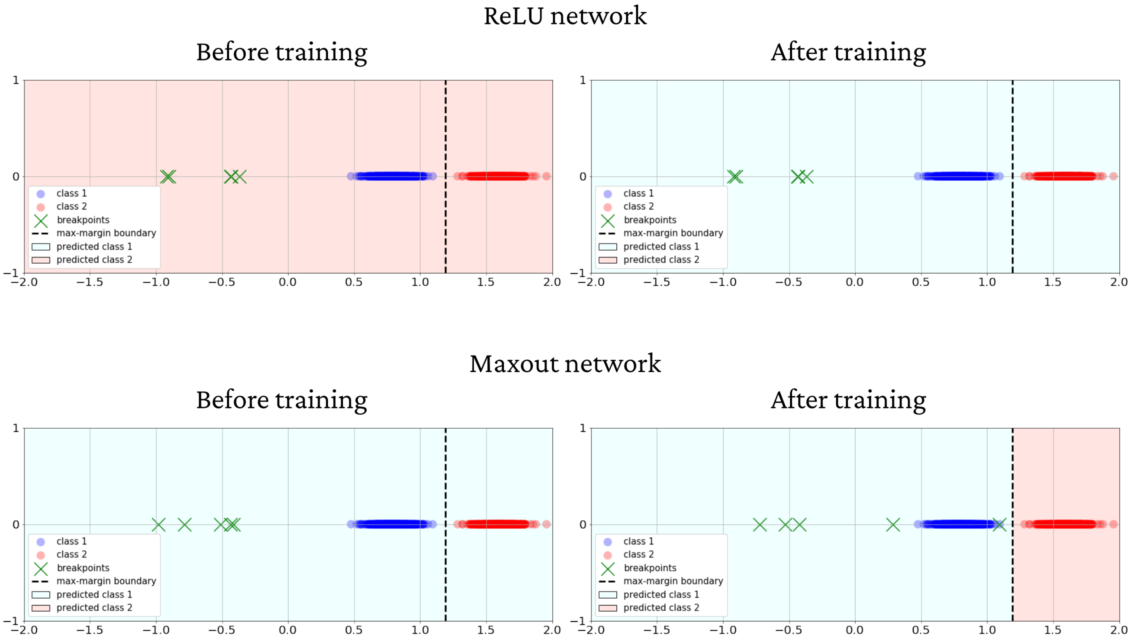


Figure 4.4: Example of a situation where the training is unsuccessful for a ReLU network because all neurons in the first layer are dead, while a maxout network trains successfully on the same dataset. We plot the breakpoints in the first layer. Notice that they do not move during the training of a ReLU network but change their positions in a maxout network.

4.B.1.2 Dying neurons problem

The dying neurons problem in ReLU networks refers to ReLU neurons being inactive on a dataset and never getting updated during optimization. It can lead to a situation when the training cannot commence if all neurons in one layer are dead. This problem never occurs in maxout networks since maxout units are always active. We design a simple experiment to illustrate this issue.

We consider a binary classification task on a dataset sampled from a Gaussian mixture of two univariate Gaussians $N(0.8, 0.1)$ and $N(1.6, 0.1)$. We sample 600 training, 200 validation, and 200 test points. We construct maxout and ReLU networks with 5 layers and 5 units per layer. Maxout units rank equals 2. We set weights and biases in the first layer so that the breakpoints are left of the data. For ReLU, we also ensure that the weights are negative to guarantee that the neurons in the first layer are inactive. Hence, all the units in the first layer of the ReLU network are dead. Then we train the network for 20 epochs using SGD with a learning rate of 0.5 and batch size of 32. For the ReLU networks, since all units in the first layer are dead, the training is unsuccessful, and the accuracy on the test set is 50%. In contrast, for the maxout network, the test set accuracy is 100%. Figure 4.4 illustrates this example.

4.B.2 Basic notions of probability

We ought to remind several probability theory notions that we use to state our results. Firstly, recall that if $v_1, \dots, v_k$ are independent, univariate standard normal random variables, then the sum of their squares, $\sum_{i=1}^{k} v_i^2$, is distributed according to the chi-squared distribution with k degrees of freedom. We will denote such a random variable with χ_k^2.

Secondly, the largest order statistic is a random variable defined as the maximum of a random sample, and the smallest order statistic is the minimum of a sample. And finally, a real-valued random variable X is said to be smaller than Y in the stochastic order, denoted by $X \leq_{st} Y$, if $\Pr(X > x) \leq \Pr(Y > x)$ for all $x \in \mathbb{R}$. We will also denote with $\overset{d}{=}$ equality in distribution (meaning the cdfs are the same). With this, we start with the results for the squared norm of the input-output Jacobian $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$.

4.B.3 Details on the equation (4.1)

In (4.1) we are investigating magnitude of $\frac{\partial \mathcal{L}(\boldsymbol{x})}{\partial W_{i,k',j}^{(l)}}$. The reason we focus on the Jacobian norm rather than on C is as follows. We have

$$\begin{aligned}\frac{\partial \mathcal{L}(\boldsymbol{x})}{\partial W_{i,k',j}^{(l)}} =& \langle \nabla_{\mathcal{N}} \mathcal{L}(\mathcal{N}(\boldsymbol{x})), \boldsymbol{J}_{\mathcal{N}}(W_{i,k',j}^{(l)}) \rangle \\ =& \langle \nabla_{\mathcal{N}} \mathcal{L}(\mathcal{N}(\boldsymbol{x})), \boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x}_i^{(l)}) \rangle \boldsymbol{x}_j^{(l-1)} \\ =& \langle \nabla_{\mathcal{N}} \mathcal{L}(\mathcal{N}(\boldsymbol{x})), \boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x}^{(l)}) \boldsymbol{u} \rangle \boldsymbol{x}_j^{(l-1)}, \quad \boldsymbol{u} = \boldsymbol{e}_i \\ =& C(\boldsymbol{x}, W) \|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x}^{(l)}) \boldsymbol{u}\| \boldsymbol{x}_j^{(l-1)}\end{aligned}$$

Note that $C(\boldsymbol{x}, W) = \langle \nabla_{\mathcal{N}} \mathcal{L}(\mathcal{N}(\boldsymbol{x})), \boldsymbol{v} \rangle$ with $\boldsymbol{v} = \boldsymbol{J}_{\mathcal{N}}\left(\boldsymbol{x}^{(l)}\right) \boldsymbol{u} / \|\boldsymbol{J}_{\mathcal{N}}\left(\boldsymbol{x}^{(l)}\right) \boldsymbol{u}\|$, $\|\boldsymbol{v}\| = 1$. Hence $C(\boldsymbol{x}, W) \leq \|\nabla_{\mathcal{N}} \mathcal{L}(\mathcal{N}(\boldsymbol{x}))\| \|\boldsymbol{v}\| = \|\nabla_{\mathcal{N}} \mathcal{L}(\mathcal{N}(\boldsymbol{x}))\|$. The latter term does not directly depend on the specific parametrization nor the specific architecture of the network but only on the loss function and the prediction. In view of the description of $\frac{\partial \mathcal{L}(\boldsymbol{x})}{\partial W_{i,k',j}^{(l)}}$, the variance depends on the square of $\boldsymbol{x}_j^{(l-1)}$. Similarly, the variance of the gradient $\nabla_{W^{(l)}} \mathcal{L}(\boldsymbol{x}) = (\frac{\partial \mathcal{L}(\boldsymbol{x})}{\partial W_{i,k',j}^{(l)}})_j$ depends on $\boldsymbol{x}^{(l-1)} = (\boldsymbol{x}_j^{(l-1)})_j$ and thus depends on $\|\boldsymbol{x}^{(l-1)}\|^2$. This is how activation length appears in (4.1).

4.C Bounds for the input-output Jacobian norm $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$

4.C.1 Preliminaries

We start by presenting several well-known results we will need for further discussion.

Product of a Gaussian matrix and a unit vector

Lemma 4.10. *Suppose W is an $n \times n'$ matrix with i.i.d. Gaussian entries and $\boldsymbol{u}$ is a random unit vector in $\mathbb{R}^{n'}$ that is independent of W but otherwise has any distribution. Then*

1. *$W\boldsymbol{u}$ is independent of $\boldsymbol{u}$ and is equal in distribution to $W\boldsymbol{v}$ where $\boldsymbol{v}$ is any fixed unit vector in $\mathbb{R}^{n'}$.*
2. *If the entries of W are sampled i.i.d. from $N(\mu, \sigma^2)$, then for all $i = 1, \dots, n$, $W_i\boldsymbol{u} \sim N(\mu, \sigma^2)$ and independent of $\boldsymbol{u}$.*
3. *If the entries of W are sampled i.i.d. from $N(0, \sigma^2)$, then the squared ℓ_2 norm $\|W\boldsymbol{u}\|^2 \overset{d}{=} \sigma^2\chi_n^2$, where χ_n^2 is a chi-squared random variable with n degrees of freedom that is independent of $\boldsymbol{u}$.*

Proof. Statement 1 was proved in, e.g., Hanin et al. (2021, Lemma C.3) by considering directly the joint distribution of $W\boldsymbol{u}$ and $\boldsymbol{u}$.

Statement 2 follows from Statement 1 if we pick $\boldsymbol{v} = \boldsymbol{e}_1$.

To prove Statement 3, recall that by definition of the ℓ_2 norm, $\|W\boldsymbol{u}\|^2 = \sum_{i=1}^n (W_i\boldsymbol{u})^2$. By Statement 2, for all $i = 1, \dots, n$, $W_i\boldsymbol{u}$ are Gaussian random variables independent of $\boldsymbol{u}$ with mean zero and variance σ^2. Since any Gaussian random variable sampled from $N(\mu, \sigma^2)$ can be written as $\mu + \sigma v$, where $v \sim N(0, 1)$, we can write $\sum_{i=1}^n (W_i\boldsymbol{u})^2 = \sigma^2 \sum_{i=1}^n v_i^2$. By definition of the chi-squared distribution, $\sum_{i=1}^n v_i^2$ is a chi-squared random variable with n degrees of freedom denoted with χ_n^2, which leads to the desired result. □

Stochastic order We recall the definition of a stochastic order. A real-valued random variable X is said to be smaller than Y in the stochastic order, denoted by $X \leq_{st} Y$, if $\Pr(X > x) \leq \Pr(Y > x)$ for all $x \in \mathbb{R}$.

Remark 4.11 (Stochastic ordering for functions)**.** Consider two functions $f : \boldsymbol{X} \to \mathbb{R}$ and $g : \boldsymbol{X} \to \mathbb{R}$ that satisfy $f(x) \leq g(x)$ for all $x \in \boldsymbol{X}$. Then, for a random variable X, $f(X) \leq_{st} g(X)$. To see this, observe that for any $y \in \mathbb{R}$, $\Pr(f(X) > y) = \Pr(X \in \{x\colon f(x) > y\})$ and $\Pr(g(X) > y) = \Pr(X \in \{x\colon g(x) > y\})$. Since $f(x) \leq g(x)$ for all $x \in \boldsymbol{X}$, $\{x\colon f(x) > y\} \subseteq \{x\colon g(x) > y\}$. Hence, $\Pr(f(X) > y) \leq \Pr(g(X) > y)$, and $f(X) \leq_{st} g(X)$.

Remark 4.12 (Stochastic order and equality in distribution)**.** Consider real-valued random variables X, Y and $\hat{Y}$. If $X \leq_{st} Y$ and $Y \overset{d}{=} \hat{Y}$, then $X \leq_{st} \hat{Y}$. Since Y and $\hat{Y}$ have the same cdfs by definition of equality in distribution, for any $y \in \mathbb{R}$, $\Pr(X > y) \leq \Pr(Y > y) = \Pr(\hat{Y} > y)$.

4.C.2 Expression for $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$

Before proceeding to the proof of the main statement, given in Theorem 4.1, we present Proposition 4.13. Firstly, in Proposition 4.13 below, we prove an equality that holds almost surely for an input-output Jacobian under our assumptions. In this particular statement the reasoning closely follows Hanin et al. (2021, Proposition C.2). The modifications are due to the fact that a maxout network

Jacobian is a product of matrices consisting of the rows of weights that are selected based on which pre-activation feature attains maximum, while in a ReLU network, the rows in these matrices are either the neuron weights or zeros.

Proposition 4.13 (Equality for $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$)**.** *Let $\mathcal{N}$ be a fully-connected feed-forward neural network with maxout units of rank K and a linear last layer. Let the network have L layers of widths $n_1, \dots, n_L$ and n_0 inputs. Assume that the weights are continuous random variables (that have a density) and that the biases are independent of the weights but otherwise initialized using any approach. Let $\boldsymbol{u} \in \mathbb{R}^{n_0}$ be a fixed unit vector. Then, for any input into the network, $\boldsymbol{x} \in \mathbb{R}^{n_0}$, almost surely with respect to the parameter initialization the Jacobian with respect to the input satisfies*

$$\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2 = \|W^{(L)}\boldsymbol{u}^{(L-1)}\|^2 \prod_{l=1}^{L-1} \sum_{i=1}^{n_l} \langle \overline{W}_i^{(l)}, \boldsymbol{u}^{(l-1)} \rangle^2, \tag{4.3}$$

where vectors $\boldsymbol{u}^{(l)}$, $l = 1, \dots, L-1$ are defined recursively as $\boldsymbol{u}^{(l)} = \overline{W}^{(l)}\boldsymbol{u}^{(l-1)} / \|\overline{W}^{(l)}\boldsymbol{u}^{(l-1)}\|$ when $\overline{W}^{(l)}\boldsymbol{u}^{(l-1)} \neq 0$ and 0 otherwise, and $\boldsymbol{u}^{(0)} = \boldsymbol{u}$. The matrices $\overline{W}^{(l)}$ consist of rows $\overline{W}_i^{(l)} = W_{i,k'}^{(l)} \in \mathbb{R}^{n_{l-1}}$, $i = 1, \dots, n_l$, where $k' = \operatorname{argmax}_{k \in [K]}\{W_{i,k}^{(l)}\boldsymbol{x}^{(l-1)} + b_{i,k}^{(l)}\}$, $\boldsymbol{x}^{(l)}$ is the output of the lth layer, and $\boldsymbol{x}^{(0)} = \boldsymbol{x}$.

Proof. The Jacobian $\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})$ of a network $\mathcal{N}(\boldsymbol{x}) \colon \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ can be written as a product of matrices $\overline{W}^{(l)}$, $l = 1, \dots, L$, depending on the activation region of the input $\boldsymbol{x}$. The matrix $\overline{W}^{(l)}$ consists of rows $\overline{W}_i^{(l)} = W_{i,k'}^{(l)} \in \mathbb{R}^{n_{l-1}}$, where $k' = \operatorname{argmax}_{k \in [K]}\{W_{i,k}^{(l)}\boldsymbol{x}^{(l-1)} + b_{i,k}^{(l)}\}$ for $i = 1, \dots, n_l$, and $\boldsymbol{x}^{(l-1)}$ is the lth layer's input. For the last layer, which is linear, we have $\overline{W}^{(L)} = W^{(L)}$. Thus,

$$\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2 = \|W^{(L)}\overline{W}^{(L-1)} \cdots \overline{W}^{(1)}\boldsymbol{u}\|^2. \tag{4.4}$$

Further we denote $\boldsymbol{u}$ with $\boldsymbol{u}^{(0)}$ and assume $\|\overline{W}^{(1)}\boldsymbol{u}^{(0)}\| \neq 0$. To see that this holds almost surely, note that for a fixed unit vector $\boldsymbol{u}^{(0)}$, the probability of $\overline{W}^{(1)}$ being such that $\|\overline{W}^{(1)}\boldsymbol{u}^{(0)}\| = 0$ is 0. This is indeed the case since to satisfy $\|\overline{W}^{(1)}\boldsymbol{u}^{(0)}\| = 0$, the weights must be a solution to a system of n_1 linear equations and this system is regular when $\boldsymbol{u} \neq 0$, so the solution set has positive co-dimension and hence zero measure. Multiplying and dividing (4.4) by $\|\overline{W}^{(1)}\boldsymbol{u}^{(0)}\|^2$,

$$\begin{aligned}
\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2 &= \left\| W^{(L)}\overline{W}^{(L-1)} \cdots \overline{W}^{(2)} \frac{\overline{W}^{(1)}\boldsymbol{u}^{(0)}}{\|\overline{W}^{(1)}\boldsymbol{u}^{(0)}\|} \right\|^2 \|\overline{W}^{(1)}\boldsymbol{u}^{(0)}\|^2 \\
&= \left\| W^{(L)}\overline{W}^{(L-1)} \cdots \overline{W}^{(2)}\boldsymbol{u}^{(1)} \right\|^2 \|\overline{W}^{(1)}\boldsymbol{u}^{(0)}\|^2,
\end{aligned}$$

where $\boldsymbol{u}^{(1)} = \overline{W}^{(1)}\boldsymbol{u}^{(0)} / \|\overline{W}^{(1)}\boldsymbol{u}^{(0)}\|$. Repeating this procedure layer-by-layer, we get

$$\|W^{(L)}\boldsymbol{u}^{(L-1)}\|^2 \|\overline{W}^{(L-1)}\boldsymbol{u}^{(L-2)}\|^2 \cdots \|\overline{W}^{(1)}\boldsymbol{u}^{(0)}\|^2. \tag{4.5}$$

By definition of the ℓ_2 norm, for any layer l, $\|\overline{W}^{(l)}\boldsymbol{u}^{(l-1)}\|^2 = \sum_{i=1}^{n_l}\langle\overline{W}_i^{(l)}, \boldsymbol{u}^{(l-1)}\rangle^2$. Substituting this into (4.5) we get the desired statement. □

4.C.3 Stochastic ordering for $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$

Now we prove the result for the stochastic ordering of the input-output Jacobian in a finite-width maxout network.

Theorem 4.1 (Bounds on $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$). *Consider a maxout network with the settings of Section 4.2. Assume that the biases are independent of the weights but otherwise initialized using any approach. Let $\boldsymbol{u} \in \mathbb{R}^{n_0}$ be a fixed unit vector. Then, almost surely, with respect to the parameter initialization, for any input into the network $\boldsymbol{x} \in \mathbb{R}^{n_0}$, the following stochastic order bounds hold:*

$$\frac{1}{n_0}\chi^2_{n_L}\prod_{l=1}^{L-1}\frac{c}{n_l}\sum_{i=1}^{n_l}\xi_{l,i}(\chi^2_1, K) \leq_{st} \|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2 \leq_{st} \frac{1}{n_0}\chi^2_{n_L}\prod_{l=1}^{L-1}\frac{c}{n_l}\sum_{i=1}^{n_l}\Xi_{l,i}(\chi^2_1, K),$$

where $\xi_{l,i}(\chi^2_1, K)$ and $\Xi_{l,i}(\chi^2_1, K)$ are respectively the smallest and largest order statistic in a sample of size K of chi-squared random variables with 1 degree of freedom, independent of each other and of the vectors $\boldsymbol{u}$ and $\boldsymbol{x}$.

Proof. From Proposition 4.13, we have the following equality

$$\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2 = \|W^{(L)}\boldsymbol{u}^{(L-1)}\|^2\prod_{l=1}^{L-1}\sum_{i=1}^{n_l}\langle\overline{W}_i^{(l)}, \boldsymbol{u}^{(l-1)}\rangle^2, \tag{4.6}$$

where vectors $\boldsymbol{u}^{(l)}$, $l = 0, \dots, L-1$ are defined recursively as $\boldsymbol{u}^{(l)} = \overline{W}^{(l)}\boldsymbol{u}^{(l-1)}/\|\overline{W}^{(l)}\boldsymbol{u}^{(l-1)}\|$ and $\boldsymbol{u}^{(0)} = \boldsymbol{u}$. Matrices $\overline{W}^{(l)}$ consist of rows $\overline{W}_i^{(l)} = W_{i,k'}^{(l)} \in \mathbb{R}^{n_{l-1}}$, $i = 1, \dots, n_l$, where $k' = \text{argmax}_{k\in[K]}\{W_{i,k}^{(l)}\boldsymbol{x}^{(l-1)} + b_{i,k}^{(l)}\}$, and $\boldsymbol{x}^{(l-1)}$ is the lth layer's input, $\boldsymbol{x}^{(0)} = \boldsymbol{x}$.

We assumed that weights in the last layer are sampled from a Gaussian distribution with mean zero and variance $1/n_{L-1}$. Then, by Lemma 4.10 item 3, $\|W^{(L)}\boldsymbol{u}^{(L-1)}\|^2 \overset{d}{=} (1/n_{L-1})\chi^2_{n_L}$ and is independent of $\boldsymbol{u}^{(L-1)}$. In equation (4.6), using this observation and then multiplying and dividing the summands by c/n_{l-1} and rearranging we obtain

$$\begin{aligned}\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2 &\overset{d}{=} \frac{1}{n_{L-1}}\chi^2_{n_L}\prod_{l=1}^{L-1}\frac{c}{n_{l-1}}\sum_{i=1}^{n_l}\left(\frac{n_{l-1}}{c}\langle\overline{W}_i^{(l)}, \boldsymbol{u}^{(l-1)}\rangle^2\right)\\ &= \frac{1}{n_0}\chi^2_{n_L}\prod_{l=1}^{L-1}\frac{c}{n_l}\sum_{i=1}^{n_l}\left(\frac{n_{l-1}}{c}\langle\overline{W}_i^{(l)}, \boldsymbol{u}^{(l-1)}\rangle^2\right).\end{aligned}$$

Now we focus on $\sqrt{n_{l-1}/c}\,\langle\overline{W}_i^{(l)}, \boldsymbol{u}^{(l-1)}\rangle$. Since we have assumed that the weights are sampled from a Gaussian distribution with zero mean and variance c/n_{l-1}, any weight $W_{i,k,j}^{(l)}$, $j =$

$1, \dots, n_{l-1}$, can be written as $\sqrt{c/n_{l-1}}v_{i,k,j}^{(l)}$, where $v_{i,k,j}^{(l)}$ is a standard Gaussian random variable. We also write $W_{i,k}^{(l)} = \sqrt{c/n_{l-1}}V_{i,k}^{(l)}$, where $V_{i,k}^{(l)}$ is an n_{l-1}-dimensional standard Gaussian random vector. Observe that for any $k' \in [K]$, $\langle W_{i,k'}^{(l)}, \boldsymbol{u}^{(l-1)} \rangle^2 \leq \max_{k \in [K]}\{\langle W_{i,k}^{(l)}, \boldsymbol{u}^{(l-1)} \rangle^2\}$ and $\langle W_{i,k'}^{(l)}, \boldsymbol{u}^{(l-1)} \rangle^2 \geq \min_{k \in [K]}\{\langle W_{i,k}^{(l)}, \boldsymbol{u}^{(l-1)} \rangle^2\}$. Therefore,

$$\frac{c}{n_{l-1}} \min_{k \in [K]} \left\{ \left\langle V_{i,k}^{(l)}, \boldsymbol{u}^{(l-1)} \right\rangle^2 \right\} \leq \langle \overline{W}_i^{(l)}, \boldsymbol{u}^{(l-1)} \rangle^2 \leq \frac{c}{n_{l-1}} \max_{k \in [K]} \left\{ \left\langle V_{i,k}^{(l)}, \boldsymbol{u}^{(l-1)} \right\rangle^2 \right\}.$$

Notice that vectors $\boldsymbol{u}^{(l-1)}$ are unit vectors by their definition. By Lemma 4.10, the inner product of a standard Gaussian vector and a unit vector is a standard Gaussian random variable independent of the given unit vector.

By definition, a squared standard Gaussian random variable is distributed as χ_1^2, a chi-squared random variable with 1 degree of freedom. Hence, $\max_{k \in [K]}\{\langle V_{i,k}^{(l)}, \boldsymbol{u}^{(l-1)} \rangle^2\}$ is distributed as the largest order statistic in a sample of size K of chi-squared random variables with 1 degree of freedom. We will denote such a random variable with $\Xi_{l,i}(\chi_1^2, K)$. Likewise, $\min_{k \in [K]}\{\langle V_{i,k}^{(l)}, \boldsymbol{u}^{(l-1)} \rangle^2\}$ is distributed as the smallest order statistic in a sample of size K of chi-squared random variables with 1 degree of freedom, denoted with $\xi_{l,i}(\chi_1^2, K)$.

Combining results for each layer, we obtain the following bounds

$$\begin{aligned}
\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2 &\leq \frac{1}{n_0}\chi_{n_L}^2 \prod_{l=1}^{L-1} \frac{c}{n_l} \sum_{i=1}^{n_l} \max_{k \in [K]} \left\{ \left\langle V_{i,k}^{(l)}, \boldsymbol{u}^{(l-1)} \right\rangle^2 \right\} \overset{d}{=} \frac{1}{n_0}\chi_{n_L}^2 \prod_{l=1}^{L-1} \frac{c}{n_l} \sum_{i=1}^{n_l} \Xi_{l,i}(\chi_1^2, K), \\
\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2 &\geq \frac{1}{n_0}\chi_{n_L}^2 \prod_{l=1}^{L-1} \frac{c}{n_l} \sum_{i=1}^{n_l} \min_{k \in [K]} \left\{ \left\langle V_{i,k}^{(l)}, \boldsymbol{u}^{(l-1)} \right\rangle^2 \right\} \overset{d}{=} \frac{1}{n_0}\chi_{n_L}^2 \prod_{l=1}^{L-1} \frac{c}{n_l} \sum_{i=1}^{n_l} \xi_{l,i}(\chi_1^2, K).
\end{aligned}$$

Then, by Remarks 4.11 and 4.12, the following stochastic ordering holds

$$\frac{1}{n_0}\chi_{n_L}^2 \prod_{l=1}^{L-1} \frac{c}{n_l} \sum_{i=1}^{n_l} \xi_{l,i}(\chi_1^2, K) \leq_{st} \|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2 \leq_{st} \frac{1}{n_0}\chi_{n_L}^2 \prod_{l=1}^{L-1} \frac{c}{n_l} \sum_{i=1}^{n_l} \Xi_{l,i}(\chi_1^2, K),$$

which concludes the proof. ☐

4.D Moments of the input-output Jacobian norm $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$

In the proof on the bounds of the moments, we use an approach similar to Hanin et al. (2021) for upper bounding the moments of the chi-squared distribution.

Corollary 4.2 (Bounds on the moments of $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$)**.** *Consider a maxout network with the settings of Section 4.2. Assume that the biases are independent of the weights but otherwise initialized using any approach. Let* $\boldsymbol{u} \in \mathbb{R}^{n_0}$ *be a fixed unit vector and* $\boldsymbol{x} \in \mathbb{R}^{n_0}$ *be any input into the network, Then* *(i)* $\frac{n_L}{n_0}(c\mathcal{S})^{L-1} \leq \mathbb{E}[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2] \leq \frac{n_L}{n_0}(c\mathcal{L})^{L-1}$,

(ii) $\text{Var}\left[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2\right] \leq \left(\frac{n_L}{n_0}\right)^2 c^{2(L-1)}\left(K^{2(L-1)}\exp\left\{4\left(\sum_{l=1}^{L-1}\frac{1}{n_l K}+\frac{1}{n_L}\right)\right\} - \mathcal{S}^{2(L-1)}\right)$,

(iii) $\mathbb{E}\left[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^{2t}\right] \leq \left(\frac{n_L}{n_0}\right)^t (cK)^{t(L-1)}\exp\left\{t^2\left(\sum_{l=1}^{L-1}\frac{1}{n_l K}+\frac{1}{n_L}\right)\right\}, \quad t \in \mathbb{N}$,

where the expectation is taken with respect to the distribution of the network weights. The constants $\mathcal{S}$ and $\mathcal{L}$ depend on K and denote the means of the smallest and the largest order statistic in a sample of K chi-squared random variables. For $K = 2, \ldots, 10$, $\mathcal{S} \in [0.02, 0.4]$ and $\mathcal{L} \in [1.6, 4]$. See Table 4.9 in Section 4.D for the exact values.

Proof. We first prove results for the mean, then for the moments of order $t > 1$, and finish with the proof for the variance.

Mean Using mutual independence of the variables in the bounds in Theorem 4.1, and that if two non-negative univariate random variables X and Y are such that $X \leq_{st} Y$ then $\mathbb{E}[X^n] \leq \mathbb{E}[Y^n]$ for all $n \geq 1$ (Müller and Stoyan, 2002, Theorem 1.2.12),

$$\frac{1}{n_0}\mathbb{E}\left[\chi^2_{n_L}\right]\prod_{l=1}^{L-1}\frac{c}{n_l}\sum_{i=1}^{n_l}\mathbb{E}\left[\xi_{l,i}\right] \leq \mathbb{E}[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2] \leq \frac{1}{n_0}\mathbb{E}\left[\chi^2_{n_L}\right]\prod_{l=1}^{L-1}\frac{c}{n_l}\sum_{i=1}^{n_l}\mathbb{E}\left[\Xi_{l,i}\right].$$

where we used $\xi_{l,i}$ and $\Xi_{l,i}$ as shorthands for $\xi_{l,i}(\chi^2_1, K)$ and $\Xi_{l,i}(\chi^2_1, K)$. Using the formulas for the largest and the smallest order statistic pdfs from Remark 4.14, the largest order statistic mean equals

$$\mathbb{E}\left[\Xi_{l,i}\right] = \frac{K}{\sqrt{2\pi}}\int_0^\infty \left(\text{erf}\left(\sqrt{\frac{x}{2}}\right)\right)^{K-1} x^{1/2}e^{-x/2}dx = \mathcal{L},$$

and the smallest order statistic mean equals

$$\mathbb{E}\left[\xi_{l,i}\right] = \frac{K}{\sqrt{2\pi}}\int_0^\infty \left(1-\text{erf}\left(\sqrt{\frac{x}{2}}\right)\right)^{K-1} x^{1/2}e^{-x/2}dx = \mathcal{S}.$$

Here we denoted the right hand-sides with $\mathcal{L}$ and $\mathcal{S}$, which are constants depending on K, and can be computed exactly for $K = 2$ and $K = 3$, and approximately for higher K-s, see Table 4.9. It is known that $\mathbb{E}\left[\chi^2_{n_L}\right] = n_L$. Combining, we get

$$\frac{n_L}{n_0}(c\mathcal{S})^{L-1} \leq \mathbb{E}[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2] \leq \frac{n_L}{n_0}(c\mathcal{L})^{L-1}.$$

Moments of order $t > 1$ As above, using mutual independence of the variables in the bounds in Theorem 4.1, and that if two non-negative univariate random variables X and Y are such that $X \leq_{st} Y$ then $\mathbb{E}[X^n] \leq \mathbb{E}[Y^n]$ for all $n \geq 1$ (Müller and Stoyan, 2002, Theorem 1.2.12),

$$\begin{aligned}\mathbb{E}[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^{2t}] &\leq \mathbb{E}\left[\left(\frac{1}{n_0}\chi^2_{n_L}\prod_{l=1}^{L-1}\frac{c}{n_l}\sum_{i=1}^{n_l}\Xi_{l,i}\right)^t\right] \\ &= \left(\frac{n_L}{n_0}\right)^t\left(\frac{1}{n_L}\right)^t\mathbb{E}\left[\left(\chi^2_{n_L}\right)^t\right]\prod_{l=1}^{L-1}\left(\frac{c}{n_l}\right)^t\mathbb{E}\left[\left(\sum_{i=1}^{n_l}\Xi_{l,i}\right)^t\right].\end{aligned} \tag{4.7}$$

Upper-bounding the maximum of chi-squared variables with a sum,

$$\left(\frac{c}{n_l}\right)^t\mathbb{E}\left[\left(\sum_{i=1}^{n_l}\Xi_{l,i}\right)^t\right] \leq \left(\frac{c}{n_l}\right)^t\mathbb{E}\left[\left(\sum_{i=1}^{n_l}\sum_{k=1}^{K}(\chi^2_1)_{l,i,k}\right)^t\right] = \left(\frac{c}{n_l}\right)^t\mathbb{E}\left[\left(\chi^2_{n_lK}\right)^t\right],$$

where we used that a sum of n_lK chi-squared variables with one degree of freedom is a chi-squared variable with n_lK degrees of freedom. Using the formula for noncentral moments of the chi-squared distribution and the inequality $1 + x \leq e^x$,

$$\begin{aligned}&\left(\frac{c}{n_l}\right)^t\mathbb{E}\left[\left(\chi^2_{n_lK}\right)^t\right] = \left(\frac{c}{n_l}\right)^t(n_lK)\,(n_lK+2)\cdots(n_lK+2t-2) \\ &= c^tK^t\cdot 1\cdot\left(1+\frac{2}{n_lK}\right)\cdots\left(1+\frac{2t-2}{n_lK}\right) \leq c^tK^t\exp\left\{\sum_{i=0}^{t-1}\frac{2i}{n_lK}\right\} \leq c^tK^t\exp\left\{\frac{t^2}{n_lK}\right\},\end{aligned}$$

where we used the formula for calculating the sum of consecutive numbers $\sum_{i=1}^{t-1} i = t(t-1)/2$. Similarly,

$$\left(\frac{1}{n_L}\right)^t\mathbb{E}\left[\left(\chi^2_{n_L}\right)^t\right] \leq \exp\left\{\frac{t^2}{n_L}\right\}.$$

Combining, we upper bound (4.7) with

$$\left(\frac{n_L}{n_0}\right)^t(cK)^{t(L-1)}\exp\left\{t^2\left(\sum_{l=1}^{L-1}\frac{1}{n_lK}+\frac{1}{n_L}\right)\right\}.$$

Variance Combining the upper bound on the second moment and the lower bound on the mean, we get the following upper bound on the variance

$$\operatorname{Var}\left[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2\right] \leq \left(\frac{n_L}{n_0}\right)^2c^{2(L-1)}\left(K^{2(L-1)}\exp\left\{4\left(\sum_{l=1}^{L-1}\frac{1}{n_lK}+\frac{1}{n_L}\right)\right\} - \mathcal{S}^{2(L-1)}\right).$$

which concludes the proof. □

Remark 4.14 (Computing the constants)**.** Here we provide the derivations necessary to compute the constants equal to the moments of the largest and the smallest order statistics appearing in the results. Firstly, the cdf of the largest order statistic of independent univariate random variables $y_1, \dots, y_K$ with cdf $F(x)$ and pdf $f(x)$ is

$$\Pr\left(\max_{k\in[K]}\{y_k\} < x\right) = \Pr\left(\bigcap_{k=1}^{K}(y_k \ < x)\right) = \prod_{k=1}^{K}\Pr\left(y_k < x\right) = (F(x))^K.$$

Hence, the pdf is $K(F(x))^{K-1}f(x)$. For the smallest order statistic, the cdf is

$$\Pr\left(\min_{k\in[K]}\{y_k\} < x\right) = 1 - \prod_{k=1}^{K}\Pr\left(y_k \geq x\right) = 1 - (1 - F(x))^K.$$

Thus, the pdf is $K\left(1 - F(x)\right)^{K-1} f(x)$.

Now we obtain pdfs for the distributions that are used in the results.

Chi-squared distribution The cdf of a chi-squared random variable χ_k^2 with $k = 1$ degree of freedom is $F(x) = (\Gamma(k/2))^{-1}\gamma(k/2, x/2) = \text{erf}(\sqrt{x/2})$, and the pdf is $f(x) = (2^{k/2}\Gamma(k/2))^{-1}x^{k/2-1}e^{-x/2} = (2\pi)^{-1/2}x^{-1/2}e^{-x/2}$. Here we used that $\Gamma(1/2) = \sqrt{\pi}$ and $\gamma(1/2, x/2) = \sqrt{\pi}\,\text{erf}(\sqrt{x/2})$. Therefore, the pdf of the largest order statistic in a sample of K chi-squared random variables with 1 degree of freedom $\Xi_{l,i}(\chi_1^2, K)$ is

$$K\left(\text{erf}\left(\sqrt{\frac{x}{2}}\right)\right)^{K-1}\frac{1}{\sqrt{2\pi}}x^{-\frac{1}{2}}e^{-\frac{x}{2}}.$$

The pdf of the smallest order statistic in a sample of K chi-squared random variables with 1 degree of freedom $\xi_{l,i}(\chi_1^2, K)$ is

$$K\left(1 - \text{erf}\left(\sqrt{\frac{x}{2}}\right)\right)^{K-1}\frac{1}{\sqrt{2\pi}}x^{-\frac{1}{2}}e^{-\frac{x}{2}}.$$

Standard Gaussian distribution Recall that the cdf of a standard Gaussian random variable is $F(x) = 1/2(1 + \text{erf}(x/\sqrt{2}))$, and the pdf is $f(x) = 1/\sqrt{2\pi}\exp\{-x^2/2\}$. Then, for the pdf of the largest order statistic in a sample of K standard Gaussian random variables $\Xi_{l,i}(N(0,1), K)$ we get

$$\frac{K}{2^{K-1}\sqrt{2\pi}}\left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right)^{K-1}e^{-\frac{x^2}{2}}.$$

Table 4.9: Constants $\mathcal{L}$ and $\mathcal{S}$ denote the means of the largest and the smallest order statistics in a sample of size K of chi-squared random variables with 1 degree of freedom. Constant $\mathcal{M}$ denotes the second moment of the largest order statistic in a sample of size K of standard Gaussian random variables. See Remark 4.14 for the explanation of how these constants are computed.

MAXOUT RANK	$\mathcal{L}$	$\mathcal{S}$	$\mathcal{M}$
2	1.63662	0.36338	1
3	2.10266	0.1928	1.27566
4	2.47021	0.1207	1.55133
5	2.77375	0.08308	1.80002
6	3.03236	0.06083	2.02174
7	3.25771	0.04655	2.2203
8	3.45743	0.0368	2.39954
9	3.63681	0.02984	2.56262
10	3.79962	0.0247	2.7121

Constants Now we obtain formulas for the constants. For the mean of the smallest order statistic in a sample of K chi-squared random variables with 1 degree of freedom $\xi_{l,i}(\chi_1^2, K)$, we get

$$\mathcal{S} = \frac{K}{\sqrt{2\pi}} \int_0^\infty x^{\frac{1}{2}} \left(1 - \text{erf}\left(\sqrt{\frac{x}{2}}\right)\right)^{K-1} e^{-\frac{x}{2}} dx.$$

The mean of the largest order statistic in a sample of K chi-squared random variables with 1 degree of freedom $\Xi_{l,i}(\chi_1^2, K)$ is

$$\mathcal{L} = \frac{K}{\sqrt{2\pi}} \int_0^\infty x^{\frac{1}{2}} \left(\text{erf}\left(\sqrt{\frac{x}{2}}\right)\right)^{K-1} e^{-\frac{x}{2}} dx.$$

The second moment of the largest order statistic in a sample of K standard Gaussian random variables $\Xi_{l,i}(N(0,1), K)$ equals

$$\mathcal{M} = \frac{K}{2^{K-1}\sqrt{2\pi}} \int_{-\infty}^\infty x^2 \left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right)^{K-1} e^{-\frac{x^2}{2}} dx.$$

These constants can be evaluated using numerical computation software. The values estimated for $K = 2, \dots, 10$ using Mathematica (Wolfram Research, Inc, 2022) are in Table 4.9.

4.E Equality in distribution for the input-output Jacobian norm and wide network results

Here we prove results from Section 4.3.2. We will use the following theorem from Anderson (2003). We reference it here without proof, but remark that it is based on the well-known result that uncorrelated jointly Gaussian random variables are independent.

Theorem 4.15 (Anderson 2003, Theorem 3.3.1)**.** *Suppose $\boldsymbol{X}_1, \dots, \boldsymbol{X}_N$ are independent, where $\boldsymbol{X}_\alpha$ is distributed according to $N(\boldsymbol{\mu}_\alpha, \boldsymbol{\Sigma})$. Let $\boldsymbol{C} = (c_{\alpha\beta})$ be an $N \times N$ orthogonal matrix. Then $\boldsymbol{Y}_\alpha = \sum_{\beta=1}^{N} c_{\alpha\beta}\boldsymbol{X}_\beta$ is distributed according to $N(\boldsymbol{\nu}_\alpha, \boldsymbol{\Sigma})$, where $\boldsymbol{\nu}_\alpha = \sum_{\beta=1}^{N} c_{\alpha\beta}\boldsymbol{\mu}_\beta$, $\alpha = 1, \dots, N$, and $\boldsymbol{Y}_1, \dots, \boldsymbol{Y}_N$ are independent.*

Remark 4.16. We will use Theorem 4.15 in the following way. Notice that it is possible to consider a vector $\boldsymbol{v}$ with entries sampled i.i.d. from $N(0, \sigma^2)$ in Theorem 4.15 and treat entries of $\boldsymbol{v}$ as a set of 1-dimensional vectors $\boldsymbol{X}_1, \dots, \boldsymbol{X}_N$. Then we can obtain that products of the columns of the orthogonal matrix $\boldsymbol{C}$ and the vector $\boldsymbol{v}$, $\boldsymbol{Y}_\beta = \sum_{\alpha=1}^{N} c_{\alpha\beta}\boldsymbol{v}_\alpha$, are distributed according to $N(0, \sigma^2)$ and are mutually independent.

Theorem 4.3 (Equality in distribution for $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$)**.** *Consider a maxout network with the settings of Section 4.2. Let $\boldsymbol{u} \in \mathbb{R}^{n_0}$ be a fixed unit vector and $\boldsymbol{x} \in \mathbb{R}^{n_0}$, $\boldsymbol{x} \neq \boldsymbol{0}$ be any input into the network. Then, almost surely, with respect to the parameter initialization, $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$ equals in distribution*

$$\frac{1}{n_0}\chi^2_{n_L} \prod_{l=1}^{L-1} \frac{c}{n_l} \sum_{i=1}^{n_l} \left(v_i \sqrt{1 - \cos^2 \gamma_{\mathfrak{x}^{(l-1)}, \mathfrak{u}^{(l-1)}}} + \Xi_{l,i}(N(0,1), K) \cos \gamma_{\mathfrak{x}^{(l-1)}, \mathfrak{u}^{(l-1)}} \right)^2,$$

where v_i and $\Xi_{l,i}(N(0,1), K)$ are independent, $v_i \sim N(0,1)$, $\Xi_{l,i}(N(0,1), K)$ is the largest order statistic in a sample of K standard Gaussian random variables. Here $\gamma_{\mathfrak{x}^{(l)}, \mathfrak{u}^{(l)}}$ denotes the angle between $\mathfrak{x}^{(l)} := (\boldsymbol{x}_1^{(l)}, \dots, \boldsymbol{x}_{n_l}^{(l)}, 1)$ and $\mathfrak{u}^{(l)} := (\boldsymbol{u}_1^{(l)}, \dots, \boldsymbol{u}_{n_l}^{(l)}, 0)$ in $\mathbb{R}^{n_l+1}$, where $\boldsymbol{u}^{(l)} = \overline{W}^{(l)}\boldsymbol{u}^{(l-1)} / \|\overline{W}^{(l)}\boldsymbol{u}^{(l-1)}\|$ when $\overline{W}^{(l)}\boldsymbol{u}^{(l-1)} \neq 0$ and 0 otherwise, and $\boldsymbol{u}^{(0)} = \boldsymbol{u}$. The matrices $\overline{W}^{(l)}$ consist of rows $\overline{W}_i^{(l)} = W_{i,k'}^{(l)} \in \mathbb{R}^{n_{l-1}}$, where $k' = \operatorname{argmax}_{k \in [K]}\{W_{i,k}^{(l)}\boldsymbol{x}^{(l-1)} + b_{i,k}^{(l)}\}$.

Proof. By Proposition 4.13, almost surely with respect to the parameter initialization,

$$\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2 = \|W^{(L)}\boldsymbol{u}^{(L-1)}\|^2 \prod_{l=1}^{L-1} \sum_{i=1}^{n_l} \langle \overline{W}_i^{(l)}, \boldsymbol{u}^{(l-1)} \rangle^2, \tag{4.8}$$

where vectors $\boldsymbol{u}^{(l)}$, $l = 0, \dots, L-1$ are defined recursively as $\boldsymbol{u}^{(l)} = \overline{W}^{(l)}\boldsymbol{u}^{(l-1)} / \|\overline{W}^{(l)}\boldsymbol{u}^{(l-1)}\|$ and $\boldsymbol{u}^{(0)} = \boldsymbol{u}$. Matrices $\overline{W}^{(l)}$ consist of rows $\overline{W}_i^{(l)} = W_{i,k'}^{(l)} \in \mathbb{R}^{n_{l-1}}$, $i = 1, \dots, n_l$, where $k' = \operatorname{argmax}_{k \in [K]}\{W_{i,k}^{(l)}\boldsymbol{x}^{(l-1)} + b_{i,k}^{(l)}\}$, and $\boldsymbol{x}^{(l-1)}$ is the lth layer's input, $\boldsymbol{x}^{(0)} = \boldsymbol{x}$.

We assumed that weights in the last layer are sampled from a Gaussian distribution with mean zero and variance $1/n_{L-1}$. Then, by Lemma 4.10, $\|W^{(L)}\boldsymbol{u}^{(L-1)}\|^2 \overset{d}{=} (1/n_{L-1})\chi^2_{n_L}$ and is independent of $\boldsymbol{u}^{(L-1)}$. We use this observation in the equation (4.6), multiply and divide the summands in the expression by c/n_{l-1} and rearrange to obtain that

$$
\begin{aligned}
\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2 &\overset{d}{=} \frac{1}{n_{L-1}}\chi^2_{n_L}\prod_{l=1}^{L-1}\frac{c}{n_{l-1}}\sum_{i=1}^{n_l}\left(\frac{n_{l-1}}{c}\langle\overline{W}_i^{(l)},\boldsymbol{u}^{(l-1)}\rangle^2\right)\\
&= \frac{1}{n_0}\chi^2_{n_L}\prod_{l=1}^{L-1}\frac{c}{n_l}\sum_{i=1}^{n_l}\left(\frac{n_{l-1}}{c}\langle\overline{W}_i^{(l)},\boldsymbol{u}^{(l-1)}\rangle^2\right).
\end{aligned}
\tag{4.9}
$$

We define $\mathfrak{x}^{(l-1)} := (\boldsymbol{x}_1^{(l-1)},\ldots,\boldsymbol{x}_{n_{l-1}}^{(l-1)},1)\in\mathbb{R}^{n_{l-1}+1}$ and $\mathfrak{u}^{(l-1)} := (\boldsymbol{u}_1^{(l-1)},\ldots,\boldsymbol{u}_{n_{l-1}}^{(l-1)},0)\in\mathbb{R}^{n_{l-1}+1}$, $\|\mathfrak{u}\| = 1$. We append the vectors of biases to the weight matrices and denote obtained matrices with $\mathfrak{W}^{(l)}\in\mathbb{R}^{n_l\times(n_{l-1}+1)}$. Then (4.9) equals

$$
\frac{1}{n_0}\chi^2_{n_L}\prod_{l=1}^{L-1}\frac{c}{n_l}\sum_{i=1}^{n_l}\left(\frac{n_{l-1}}{c}\langle\overline{\mathfrak{W}}_i^{(l)},\mathfrak{u}^{(l-1)}\rangle^2\right).
$$

Now we focus on $\sqrt{n_{l-1}/c}\,\langle\overline{\mathfrak{W}}_i^{(l)},\mathfrak{u}^{(l-1)}\rangle$. Since we have assumed that the weights and biases are sampled from the Gaussian distribution with zero mean and variance c/n_{l-1}, any weight $W_{i,k,j}^{(l)}$, $j = 1,\ldots,n_{l-1}$ (or bias), can be written as $\sqrt{c/n_{l-1}}v_{i,k,j}^{(l)}$, where $v_{i,k,j}^{(l)}$ is standard Gaussian. Therefore,

$$
\frac{n_{l-1}}{c}\langle\overline{\mathfrak{W}}_i^{(l)},\mathfrak{u}^{(l-1)}\rangle^2 = \langle\overline{\mathfrak{V}}_i^{(l)},\mathfrak{u}^{(l-1)}\rangle^2,
\tag{4.10}
$$

where $\overline{\mathfrak{V}}_i^{(l)} = \mathfrak{V}_{i,k'}^{(l)}\in\mathbb{R}^{n_{l-1}+1}$, $k' = \operatorname{argmax}_{k\in[K]}\{\langle\mathfrak{V}_{i,k}^{(l)},\mathfrak{x}^{(l-1)}\rangle\}$, $\mathfrak{V}_{i,k}^{(l)}$ are $(n_{l-1}+1)$-dimensional standard Gaussian random vectors.

We construct an orthonormal basis $B = (\boldsymbol{b}_1,\ldots,\boldsymbol{b}_{n_{l-1}+1})$ of $\mathbb{R}^{n_{l-1}+1}$, where we set $\boldsymbol{b}_1 = \mathfrak{x}^{(l-1)}/\|\mathfrak{x}^{(l-1)}\|$ and choose the other vectors to be unit vectors orthogonal to $\boldsymbol{b}_1$. The change of basis matrix from the standard basis I to the basis B is given by B^T; see, e.g., Anton and Rorres (2013, Theorem 6.6.4). Then, any row $\mathfrak{V}_{i,k}^{(l)}$ an be expressed as

$$
\mathfrak{V}_{i,k}^{(l)} = c_{k,1}\boldsymbol{b}_1+\cdots+c_{k,n_{l-1}+1}\boldsymbol{b}_{n_{l-1}+1},
$$

where $c_{k,j} = \langle\mathfrak{V}_{i,k}^{(l)},\boldsymbol{b}_j\rangle$, $j = 1,\ldots,n_{l-1}+1$.

The coordinate vector of $\mathfrak{x}^{(l-1)}$ relative to B is $(\|\mathfrak{x}^{(l-1)}\|,0,\ldots,0)$. Vector $\mathfrak{u}^{(l-1)}$ in B has the coordinate vector $(\langle\mathfrak{u}^{(l-1)},\boldsymbol{b}_1\rangle,\ldots,\langle\mathfrak{u}^{(l-1)},\boldsymbol{b}_{n_{l-1}+1}\rangle)$. This coordinate vector has norm 1 since the change of basis between two orthonormal bases does not change the ℓ_2 norm; see, e.g., Anton and Rorres (2013, Theorem 6.3.2).

For the maximum, using the representation of the vectors in the basis B, we get

$$
\langle\overline{\mathfrak{V}}_i^{(l)},\mathfrak{x}^{(l-1)}\rangle = \max_{k\in[K]}\left\{\langle\mathfrak{V}_{i,k}^{(l)},\mathfrak{x}^{(l-1)}\rangle\right\} = \max_{k\in[K]}\left\{c_{k,1}\|\mathfrak{x}^{(l-1)}\|\right\} = \|\mathfrak{x}^{(l-1)}\|\max_{k\in[K]}\left\{c_{k,1}\right\}.
\tag{4.11}
$$

Therefore, in the basis B, $\overline{\mathfrak{V}}_i^{(l)}$ has components $(\max_{k\in[K]}\{c_{k,1}\}, c_{k',2}, \dots, c_{k',n_{l-1}+1})$. By Theorem 4.15, for all $k = 1, \dots, K$, $j = 1, \dots, n_{l-1}+1$, the coefficients $c_{k,j}$ are mutually independent standard Gaussian random variables that are also independent of vectors $\boldsymbol{b}_j, j = 1, \dots, n_{l-1}$, by Lemma 4.10 and of $\boldsymbol{u}^{(l-1)}$.

$$\begin{aligned}\langle\overline{\mathfrak{V}}_i^{(l)}, \mathbf{u}^{(l-1)}\rangle &= \max_{k\in[K]}\{c_{k,1}\}\langle\mathbf{u}^{(l-1)}, \boldsymbol{b}_1\rangle + \sum_{j=2}^{n_{l-1}+1} c_{k',j}\langle\mathbf{u}^{(l-1)}, \boldsymbol{b}_j\rangle \\ &\overset{d}{=} \Xi_{l,i}(N(0,1),K)\langle\mathbf{u}^{(l-1)}, \boldsymbol{b}_1\rangle + \sum_{j=2}^{n_{l-1}+1} v_j\langle\mathbf{u}^{(l-1)}, \boldsymbol{b}_j\rangle,\end{aligned} \tag{4.12}$$

where $\Xi_{l,i}(N(0,1),K)$ is the largest order statistic in a sample of K standard Gaussian random variables, and $v_j \sim N(0,1)$. Since we have simply written equality in distribution for $\max_{k\in[K]}\{c_{k,1}\}$ and $c_{k',j}$, the variables $\Xi_{l,i}(N(0,1),K)$ and $v_j, j = 2, \dots, n_{l-1}$ are also mutually independent, and independent of vectors $\boldsymbol{b}_j, j = 1, \dots, n_{l-1}$, and of $\boldsymbol{u}^{(l-1)}$. In the following we will use $\Xi_{l,i}$ as a shorthand for $\Xi_{l,i}(N(0,1),K)$.

A linear combination $\sum_{i=1}^n a_i, v_i$ of Gaussian random variables $v_1, \dots, v_n$, $v_j \sim N(\mu_j, \sigma_j^2)$, $j = 1, \dots, n$ with coefficients $a_1, \dots, a_n$ is distributed according to $N(\sum_{i=1}^n a_i\mu_i, \sum_{i=1}^n a_i^2\sigma_i^2)$. Hence, $\sum_{j=2}^{n_{l-1}+1} v_j\langle\mathbf{u}^{(l-1)}, \boldsymbol{b}_j\rangle \sim N(0, \sum_{j=2}^{n_{l-1}+1}\langle\mathbf{u}^{(l-1)}, \boldsymbol{b}_j\rangle^2)$. Since $\sum_{j=2}^{n_{l-1}+1}\langle\mathbf{u}^{(l-1)}, \boldsymbol{b}_j\rangle^2 = 1 - \langle\mathbf{u}^{(l-1)}, \boldsymbol{b}_1\rangle^2 = 1 - \cos^2\gamma_{\mathfrak{x}^{(l-1)},\mathbf{u}^{(l-1)}}$, we get

$$\begin{aligned}&\sum_{j=2}^{n_{l-1}+1} v_j\langle\mathbf{u}^{(l-1)}, \boldsymbol{b}_j\rangle + \Xi_{l,i}\langle\mathbf{u}^{(l-1)}, \boldsymbol{b}_1\rangle \\ &\qquad\overset{d}{=} v_i\sqrt{1 - \cos^2\gamma_{\mathfrak{x}^{(l-1)},\mathbf{u}^{(l-1)}}} + \Xi_{l,i}\cos\gamma_{\mathfrak{x}^{(l-1)},\mathbf{u}^{(l-1)}},\end{aligned} \tag{4.13}$$

where $v_i \sim N(0,1)$. Notice that $v_i\sqrt{1 - \cos^2\gamma_{\mathfrak{x}^{(l-1)},\mathbf{u}^{(l-1)}}}$ and $\Xi_{l,i}\cos\gamma_{\boldsymbol{u}^{(l-1)},\boldsymbol{x}^{(l-1)}}$ are stochastically independent because v_i and $\Xi_{l,i}$ are independent and multiplying random variables by constants does not affect stochastic independence. ☐

Remark 4.17. The result in Theorem 4.3 also holds when the biases are initialized to zero. The proof is simplified in this case. There is no need to define additional vectors $\mathfrak{x}^{(l-1)}$ and $\mathbf{u}^{(l-1)}$, and when constructing the basis, the first vector is defined as $\boldsymbol{b}_1 := \boldsymbol{x}^{(l-1)}/\|\boldsymbol{x}^{(l-1)}\|$. The rest of the proof remains the same.

Remark 4.18 (Effects of the width and depth on a maxout network)**.** According to Theorem 4.3, the behavior of $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$ in a maxout network depends on the $\cos\gamma_{\mathfrak{x}^{(l-1)},\mathbf{u}^{(l-1)}}$, which changes as the network gets wider or deeper. Figure 4.8 demonstrates how the width and depth affect $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$.

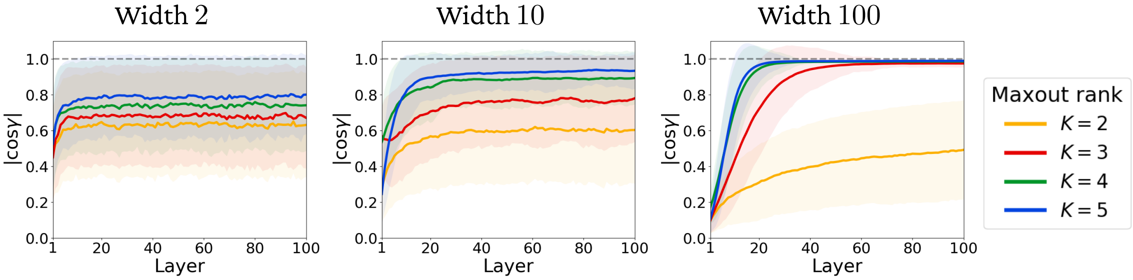


Figure 4.5: The plots show that $|\cos\gamma_{\mathfrak{x}^{(l)},\mathbf{u}^{(l)}}|$ grows with the network depth and eventually converges to 1 for wide networks and maxout rank $K > 2$. The results were averaged over 1000 parameter initializations, and both weights and biases were sampled from $N(0, c/\text{fan-in})$, $c = 1/\mathbb{E}[(\Xi(N(0,1),K))^2]$, as discussed in Section 4.4. Vectors $\boldsymbol{x}$ and $\boldsymbol{u}$ were sampled from $N(0, I)$.

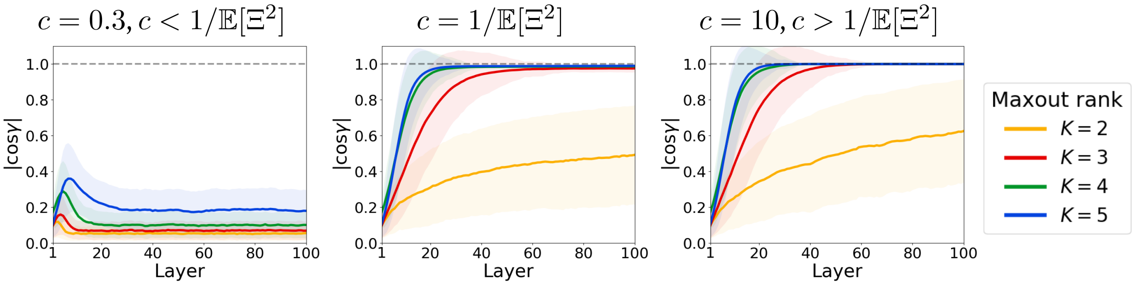


Figure 4.6: The plots show that $|\cos\gamma_{\mathfrak{x}^{(l)},\mathbf{u}^{(l)}}|$ does not converge to 1 for $c < 1/\mathbb{E}[\Xi^2]$ and converges for $c \geq 1/\mathbb{E}[\Xi^2]$. The network had 100 neurons at each layer, and both weights and biases were sampled from $N(0, c/\text{fan-in})$. The results were averaged over 1000 parameter initializations. Vectors $\boldsymbol{x}$ and $\boldsymbol{u}$ were sampled from $N(0, I)$.

Wide shallow networks Since independent and isotropic random vectors in high-dimensional spaces tend to be almost orthogonal (Vershynin, 2018, Remark 2.3.5), $\cos\gamma_{\boldsymbol{x}^{(0)},\boldsymbol{u}^{(0)}}$ will be close to 0 with high probability for wide networks if the entries of the vectors $\boldsymbol{x}$ and $\boldsymbol{u}$ are i.i.d. standard Gaussian (or i.i.d. from an isotropic distribution). Hence, we expect that the cosine will be around zero for the earlier layers of wide networks and individual units will behave more as the squared standard Gaussians.

Wide deep networks Consider wide and deep networks, where the layers $l = 0, \ldots, L-1$ are approximately of the same width $n_{l_1} \approx n_{l_2}, l_1, l_2 = 0, \ldots, L-1$. Assume that $c = 1/\mathcal{M} = 1/\mathbb{E}[(\Xi(N(0,1),K))^2]$. We will demonstrate that under these conditions. $|\cos\gamma_{\mathfrak{x}^{(l)},\mathbf{u}^{(l)}}| \approx 1$ for the later layers for $2 < K < 100$. Thus, individual units behave as the squared largest order statis-

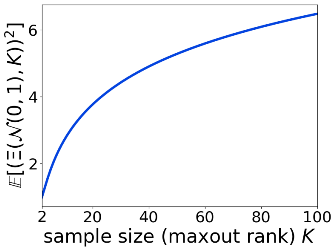


Figure 4.7: Second moment of $\Xi(N(0,1),K)$ for different sample sizes K. It increases with K for any $K, 2 \leq K \leq 100$, and $\mathbb{E}[(\Xi(N(0,1),K))^2] > 1$ for $K > 2$.

tics. To see this, we need to estimate $\cos \gamma_{\mathfrak{x}^{(l)}, \mathbf{u}^{(l)}}$ from Theorem 4.3, which is defined as

$$\cos \gamma_{\mathfrak{x}^{(l)}, \mathbf{u}^{(l)}} = \rho^{(l)}_{\mathfrak{x}\mathbf{u}} = \frac{\langle \mathfrak{x}^{(l)}, \mathbf{u}^{(l)} \rangle}{\|\mathfrak{x}^{(l)}\| \|\mathbf{u}^{(l)}\|} = \frac{\langle \mathfrak{x}^{(l)}, \overline{\mathbf{u}}^{(l)} \rangle}{\|\mathfrak{x}^{(l)}\| \|\overline{\mathbf{u}}^{(l)}\|} = \frac{\frac{n_{l-1}}{n_l} \langle \mathfrak{x}^{(l)}, \overline{\mathbf{u}}^{(l)} \rangle}{\left(\sqrt{\frac{n_{l-1}}{n_l}} \|\mathfrak{x}^{(l)}\|\right) \left(\sqrt{\frac{n_{l-1}}{n_l}} \|\overline{\mathbf{u}}^{(l)}\|\right)},$$

where we denoted $\cos \gamma_{\mathfrak{x}^{(l)}, \mathbf{u}^{(l)}}$ with $\rho^{(l)}_{\mathfrak{x}\mathbf{u}}$, and with $\overline{\mathbf{u}}^{(l)}$, $\mathbf{u}^{(l)}$ before the normalization.

Firstly, for $\mathfrak{x}^{(l)}$ we get

$$\begin{aligned} \frac{n_{l-1}}{n_l} \|\mathfrak{x}^{(l)}\|^2 &= \frac{n_{l-1}}{n_l} \left(\sum_{i=1}^{n_l} \left(\max_{k \in [K]} \left\{ \mathfrak{W}^{(l)}_{i,k} \mathfrak{x}^{(l-1)} \right\} \right)^2 + 1 \right) \\ &= c \|\mathfrak{x}^{(l-1)}\|^2 \left(\frac{1}{n_l} \sum_{i=1}^{n_l} \left(\max_{k \in [K]} \left\{ \mathfrak{V}^{(l)}_{i,k} \frac{\mathfrak{x}^{(l-1)}}{\|\mathfrak{x}^{(l-1)}\|} \right\} \right)^2 + \frac{n_{l-1}}{c \|\mathfrak{x}^{(l-1)}\|^2 n_l} \right) \\ &\overset{d}{=} c \|\mathfrak{x}^{(l-1)}\|^2 \left(\frac{1}{n_l} \sum_{i=1}^{n_l} \Xi^2_{l,i} + \frac{n_{l-1}}{c \|\mathfrak{x}^{(l-1)}\|^2 n_l} \right), \end{aligned} \tag{4.14}$$

where in the second line we used that $\mathfrak{W}^{(l)}_{i,k} = \sqrt{c/n_{l-1}} \mathfrak{V}^{(l)}_{i,k}$, $\mathfrak{V}^{(l)}_{i,k,j} \sim N(0,1), j = 1, \ldots, n_{l-1}$. In the third line, $\Xi_{l,i} \overset{d}{=} \Xi(N(0,1),K)$ is the largest order statistic in a sample of K standard Gaussians, since by Lemma 4.10, $\mathfrak{V}^{(l)}_{i,k} \mathfrak{x}^{(l-1)} / \|\mathfrak{x}^{(l-1)}\|$ are mutually independent standard Gaussian random variables. When the network width is large, $1/n_l \sum_{i=1}^{n_l} \Xi^2_{l,i}$ approximates the second moment of the largest order statistic, and $n_{l-1}/n_l \approx 1$ when the layer widths are approximately the same. Then

$$\frac{n_{l-1}}{n_l} \|\mathfrak{x}^{(l)}\|^2 \approx c \|\mathfrak{x}^{(l-1)}\|^2 \left(\mathbb{E}\left[\Xi^2\right] + \frac{1}{c \|\mathfrak{x}^{(l-1)}\|^2} \right).$$

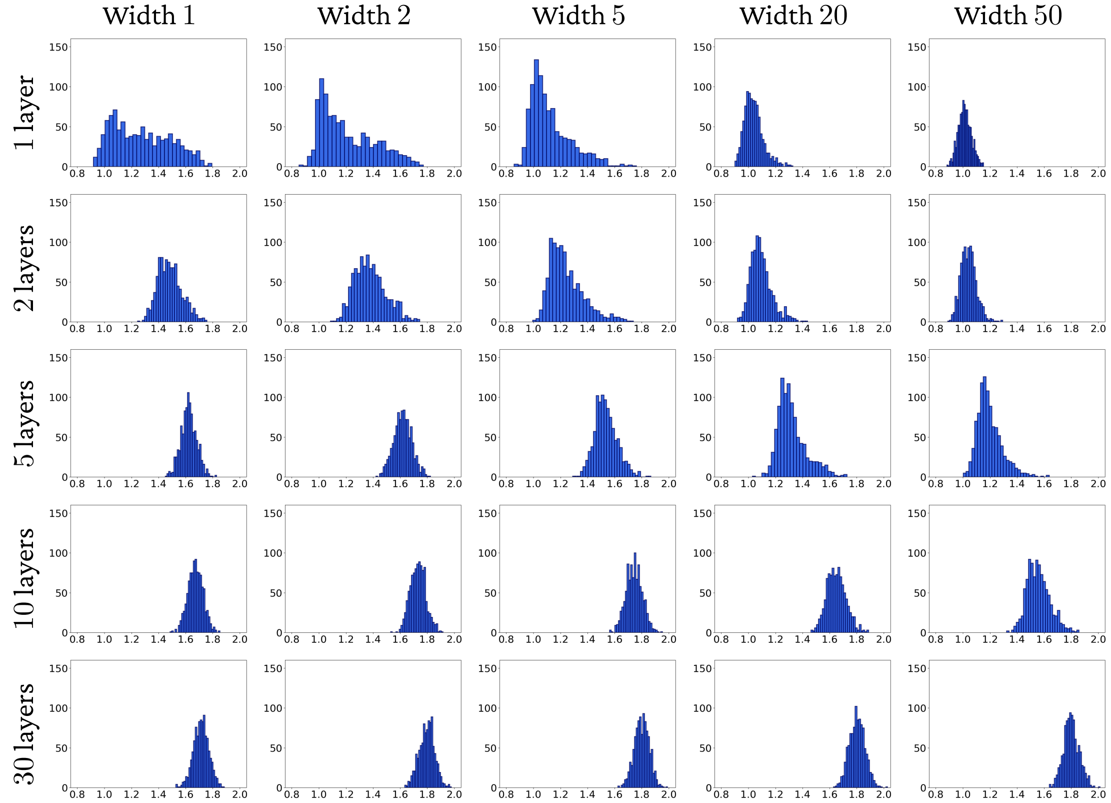


Figure 4.8: Shown is the expectation value of the square norm of the directional derivative of the input-output map of a maxout network for a fixed random direction with respect to the weights, plotted as a function of the input. Weights and biases are sampled from $N(0, 1/\text{fan-in})$, and biases are zero. Inputs are standard Gaussian vectors. Vector $\boldsymbol{u}$ is a one-hot vector with 1 at a random position, and it is the same for one setup. We sampled 1000 inputs and 1000 initializations for each input. The left end corresponds to the second moment of the Gaussian distribution, and the right end to the second moment of the largest order statistic. Observe that for wide and deep networks, the mean is closer to the second moment of the largest order statistic.

Now we will show that $1/\|\mathfrak{x}^{(l-1)}\|^2 \approx 0$. Firstly, by the same reasoning as above,

$$\|\mathfrak{x}^{(l-1)}\|^2 = \sum_{i=1}^{n_{l-1}} \left(\max_{k \in [K]} \left\{ \mathfrak{W}_{i,k}^{(l)} \mathfrak{x}^{(l-2)} \right\} \right)^2 + 1$$

$$\overset{d}{=} \|\mathfrak{x}^{(0)}\|^2 \frac{c^{l-1} n_{l-1}}{n_0} \prod_{j=1}^{l-1} \frac{1}{n_j} \sum_{i=1}^{n_j} \Xi_{l,i}^2 + \cdots + c^2 \frac{n_{l-1}}{n_{l-3}} \prod_{j=l-2}^{l-1} \frac{1}{n_j} \sum_{i=1}^{n_j} \Xi_{l,i}^2 + \frac{c n_{l-1}}{n_{l-2}} \frac{1}{n_{l-1}} \sum_{i=1}^{n_{l-1}} \Xi_{l,i}^2 + 1.$$

Since we assumed that the layer widths are large and approximately the same,

$$\|\mathfrak{x}^{(l-1)}\|^2 \approx \|\mathfrak{x}^{(0)}\|^2 \left(c\mathbb{E}[\Xi^2]\right)^{l-1} + \cdots + c\mathbb{E}[\Xi^2] + 1 = \|\mathfrak{x}^{(0)}\|^2 \left(c\mathbb{E}[\Xi^2]\right)^{l-1} + \sum_{j=0}^{l-2} \left(c\mathbb{E}[\Xi^2]\right)^j.$$

Using the assumption that $c = 1/\mathbb{E}[\Xi^2]$, we obtain that $\|\boldsymbol{\mathfrak{x}}^{(l-1)}\|^2 \approx \|\boldsymbol{\mathfrak{x}}^{(0)}\|^2 + (l-1)$ and goes to infinity with the network depth. Hence, $1/\|\boldsymbol{\mathfrak{x}}^{(l-1)}\|^2 \approx 0$ and

$$\frac{n_{l-1}}{n_l}\|\boldsymbol{\mathfrak{x}}^{(l)}\|^2 \approx c\|\boldsymbol{\mathfrak{x}}^{(l-1)}\|^2 \mathbb{E}\left[\Xi^2\right].$$

Now consider $\overline{\boldsymbol{\mathfrak{u}}}^{(l)}$. Using the reasoning from Theorem 4.3, see equations (4.12) and (4.13), $\overline{\boldsymbol{\mathfrak{u}}}_i^{(l)} \overset{d}{=} c/n_{l-1}(\Xi_{l,i}\rho_{\mathfrak{xu}}^{(l-1)} + v_i\sqrt{1-(\rho_{\mathfrak{xu}}^{(l-1)})^2}), i = 1, \ldots, n_l, v_i \sim N(0,1)$. Then in a wide network

$$\|\overline{\boldsymbol{\mathfrak{u}}}^{(l)}\|^2 \approx \frac{cn_l}{n_{l-1}}\mathbb{E}\left[\left(\Xi\rho_{\mathfrak{xu}}^{(l-1)} + v\sqrt{1-\left(\rho_{\mathfrak{xu}}^{(l-1)}\right)^2}\right)^2\right]. \tag{4.15}$$

Note that the random variable Ξ in equations (4.14) and (4.15) is the same based on the derivations in Theorem 4.3, to see this, compare equations (4.11) and (4.12).

Similarly, for the dot product $\langle\boldsymbol{\mathfrak{x}}^{(l)}, \overline{\boldsymbol{\mathfrak{u}}}^{(l)}\rangle$ in a wide network we obtain that

$$\langle\boldsymbol{\mathfrak{x}}^{(l)}, \overline{\boldsymbol{\mathfrak{u}}}^{(l)}\rangle \approx \|\boldsymbol{\mathfrak{x}}^{(l-1)}\|\frac{cn_l}{n_{l-1}}\mathbb{E}\left[\Xi\left(\Xi\rho_{\mathfrak{xu}}^{(l-1)} + v\sqrt{1-\left(\rho_{\mathfrak{xu}}^{(l-1)}\right)^2}\right)\right].$$

Hence, we have the following recursive map for $\rho_{\mathfrak{xu}}^{(l)}$

$$\begin{aligned}\rho_{\mathfrak{xu}}^{(l)} &= \frac{\mathbb{E}\left[\Xi\left(\Xi\rho_{\mathfrak{xu}}^{(l-1)} + v\sqrt{1-\left(\rho_{\mathfrak{xu}}^{(l-1)}\right)^2}\right)\right]}{\sqrt{\mathbb{E}\left[\Xi^2\right]}\sqrt{\mathbb{E}\left[\left(\Xi\rho_{\mathfrak{xu}}^{(l-1)} + v\sqrt{1-\left(\rho_{\mathfrak{xu}}^{(l-1)}\right)^2}\right)^2\right]}} \\ &= \frac{1}{\sqrt{\mathbb{E}\left[\Xi^2\right]}}\frac{\rho_{\mathfrak{xu}}^{(l-1)}\mathbb{E}[\Xi^2]}{\sqrt{\left(\rho_{\mathfrak{xu}}^{(l-1)}\right)^2(\mathbb{E}[\Xi^2]-1)+1}},\end{aligned}$$

where we used independence of v and Ξ, see Theorem 4.3, and that $\mathbb{E}[v] = 0$ and $\mathbb{E}[v^2] = 1$. This map has fixed points $\rho^* = \pm 1$, which can be confirmed by direct calculation. To check if these fixed points are stable, we need to consider the values of the derivative $\partial\rho_{\mathfrak{xu}}^{(l)}/\partial\rho_{\mathfrak{xu}}^{(l-1)}$ at them. We obtain

$$\frac{\partial\rho_{\mathfrak{xu}}^{(l)}}{\partial\rho_{\mathfrak{xu}}^{(l-1)}} = \left(\mathbb{E}[\Xi^2]\right)^{\frac{1}{2}}\left(\left(\rho_{\mathfrak{xu}}^{(l-1)}\right)^2(\mathbb{E}[\Xi^2]-1)+1\right)^{-\frac{3}{2}}.$$

When $\rho_{\mathfrak{xu}}^{(l-1)} = \pm 1$ this partial derivative equals $1/\mathbb{E}[\Xi^2] < 1$ for $K > 2$, since $\mathbb{E}[\Xi^2] > 1$, see Table 4.9 for $K = 2, \ldots, 10$ and Figure 4.7 for $K = 2, \ldots, 100$. Hence, the fixed points are stable (Strogatz, 2018, Chapter 10.1). Note that for $K = 2$, $1/\mathbb{E}[\Xi^2] = 1$, and this analysis is inconclusive. Therefore, if the network parameters are sampled from $N(0, c/n_{l-1})$, $c = 1/\mathcal{M} = 1/\mathbb{E}[\Xi(N(0,1),K)^2]$, we expect that $|\cos\gamma_{\mathfrak{x}^{(l)},\mathbf{u}^{(l)}}| \approx 1$ for the later layers of deep networks and individual units will behave more as the squared largest order statistics. Figure 4.5 demonstrates convergence of $|\cos\gamma_{\mathfrak{x}^{(l)},\mathbf{u}^{(l)}}|$ to 1 with the depth for wide networks, and Figure 4.6 shows that there is no convergence for $c < 1/\mathbb{E}[\Xi^2]$ and that the cosine still converges for $c > 1/\mathbb{E}[\Xi^2]$.

Remark 4.19 (Expectation of $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$ in a wide and deep network)**.** According to Remark 4.18, for deep and wide networks, we can expect that $|\cos\gamma_{\mathfrak{x}^{(l-1)},\mathbf{u}^{(l-1)}}| \approx 1$ if $c = 1/\mathcal{M}$, which allows obtaining an approximate equality for the expectation of $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2$. Hence, using Theorem 4.3,

$$\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2 \approx \frac{1}{n_0}\chi^2_{n_L}\prod_{l=1}^{L-1}\frac{c}{n_l}\sum_{i=1}^{n_l}\left(\Xi_{l,i}(N(0,1),K)\right)^2. \tag{4.16}$$

Then, using mutual independence of the variables in equation (4.16),

$$\mathbb{E}[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2] \approx \frac{1}{n_0}\mathbb{E}\left[\chi^2_{n_L}\right]\prod_{l=1}^{L-1}\frac{c}{n_l}\sum_{i=1}^{n_l}\mathbb{E}\left[\left(\Xi_{l,i}(N(0,1),K)\right)^2\right].$$

Since $\mathcal{M} = \mathbb{E}[(\Xi_{l,i}(N(0,1),K))^2]$, see Table 4.9, and $c = 1/\mathcal{M}$, we get

$$\mathbb{E}[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2] \approx \frac{n_L}{n_0}(c\mathcal{M})^{L-1} = \frac{n_L}{n_0}.$$

Remark 4.20 (Lower bound on the moments in a wide and deep network)**.** Using (4.16) and taking into account the mutual independence of the variables,

$$\begin{aligned}\mathbb{E}[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^{2t}] &\approx \mathbb{E}\left[\left(\frac{1}{n_0}\chi^2_{n_L}\prod_{l=1}^{L-1}\frac{c}{n_l}\sum_{i=1}^{n_l}\left(\Xi_{l,i}(N(0,1),K)\right)^2\right)^t\right]\\ &= \left(\frac{n_L}{n_0}\right)^t\left(\frac{1}{n_L}\right)^t\mathbb{E}\left[\left(\chi^2_{n_L}\right)^t\right]\prod_{l=1}^{L-1}\left(\frac{c}{n_l}\right)^t\mathbb{E}\left[\left(\sum_{i=1}^{n_l}\left(\Xi_{l,i}(N(0,1),K)\right)^2\right)^t\right]\\ &\geq \left(\frac{n_L}{n_0}\right)^t\left(\frac{1}{n_L}\right)^t\mathbb{E}\left[\left(\chi^2_{n_L}\right)^t\right]\prod_{l=1}^{L-1}\left(\frac{c}{n_l}\right)^t\left(\sum_{i=1}^{n_l}\mathbb{E}\left[\left(\Xi_{l,i}(N(0,1),K)\right)^2\right]\right)^t,\end{aligned} \tag{4.17}$$

where in the last inequality, we used linearity of expectation and Jensen's inequality since taking the tth power for $t \geq 1$ is a convex function for non-negative arguments. Using the formula for noncentral moments of the chi-squared distribution and the inequality $\ln x \geq 1 - 1/x, \forall x > 0$, meaning that $x = \exp\{\ln x\} \geq \exp\{1 - 1/x\}$, we get

$$\begin{aligned}\left(\frac{1}{n_L}\right)^t \mathbb{E}\left[\left(\chi^2_{n_L}\right)^t\right] &= \left(\frac{1}{n_l}\right)^t (n_L)(n_L+2)\cdots(n_L+2t-2) = \prod_{i=0}^{t-1}\left(1+\frac{2i}{n_L}\right) \\ &\geq \exp\left\{\sum_{i=1}^{t-1}\left(\frac{2i}{n_L+2i}\right)\right\} \geq \exp\left\{\frac{t-1}{2n_L}\right\},\end{aligned} \tag{4.18}$$

where in the last inequality, we used that $2i/(n_L+2i) \geq 2/(n_L+2) \geq 1/(2n_L)$ for all $i, n_L \geq 1$.

Using that $\mathbb{E}[(\Xi_{l,i}(N(0,1),K))^2] = \mathcal{M}$, see Table 4.9, and combing this with (4.17) and (4.18),

$$\mathbb{E}[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^{2t}] \gtrapprox \left(\frac{n_L}{n_0}\right)^t \exp\left\{\frac{t-2}{2n_L}\right\}(c\mathcal{M})^{t(L-1)}. \tag{4.19}$$

The bound in (4.19) can be tightened if a tighter lower bound on the moments of the sum of the squared largest order statistics in a sample of K standard Gaussians is known. To derive a lower bound on the moments $t \geq 2$ for the general case in Corollary 4.2, it is necessary to obtain a non-trivial lower bound on the moments of the sum of the smallest order statistics in a sample of K chi-squared random variables with 1 degree of freedom.

4.F Activation length

Here we prove the results from Subsection 4.3.3. Figure 4.9 demonstrates a close match between the estimated normalized activation length and the behavior predicted in Corollary 4.21 and 4.5.

Corollary 4.21 (Distribution of the normalized activation length)**.** *Consider a maxout network with the settings of Section 4.2. Then, almost surely with respect to the parameter initialization, for any input into the network $\boldsymbol{x} \in \mathbb{R}^{n_0}$ and $l' = 1, \ldots, L-1$, the normalized activation length $A^{(l')}$ is equal in distribution to*

$$\|\mathfrak{x}^{(0)}\|^2 \frac{1}{n_0}\prod_{l=1}^{l'}\left(\frac{c}{n_l}\sum_{i=1}^{n_l}\Xi_{l,i}(N(0,1),K)^2\right) + \sum_{j=2}^{l'}\left[\frac{1}{n_{j-1}}\prod_{l=j}^{l'}\left(\frac{c}{n_l}\sum_{i=1}^{n_l}\Xi_{l,i}(N(0,1),K)^2\right)\right],$$

where $\mathfrak{x}^{(0)} := (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n_0}, 1) \in \mathbb{R}^{n_0+1}$, $\Xi_{l,i}(N(0,1),K)$ is the largest order statistic in a sample of K standard Gaussian random variables, and $\Xi_{l,i}(N(0,1),K)$ are stochastically independent. Notice that variables $\Xi_{l,i}(N(0,1),K)$ with the same indices are the same random variables.

Proof. Define $\mathfrak{x}^{(l)} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n_l}, 1) \in \mathbb{R}^{n_l+1}$. Append the bias columns to the weight matrices and denote obtained matrices with $\mathfrak{W}^{(l)} \in \mathbb{R}^{n_l \times (n_{l-1}+1)}$. Denote $\mathfrak{W}^{(l)}_{i,k'} \in \mathbb{R}^{n_{l-1}+1}$, $k' = \operatorname{argmax}_{k\in[K]}\{\langle \mathfrak{W}^{(l)}_{i,k}, \mathfrak{x}^{(l-1)}\rangle\}$, with $\overline{\mathfrak{W}}^{(l)}_i$. Under this notation, $\|\mathfrak{x}^{(l)}\|^2 = (\|\boldsymbol{x}^{(l)}\|^2 + 1)$, $\|\boldsymbol{x}^{(l)}\| =$

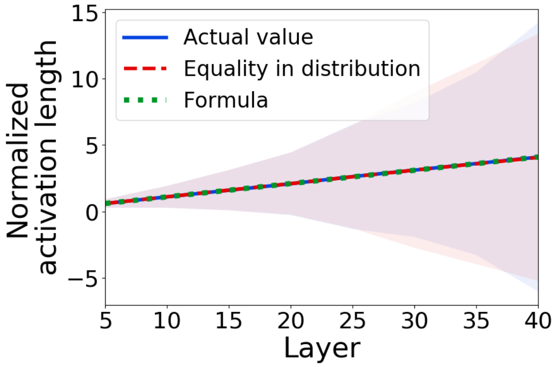


Figure 4.9: Comparison of the normalized activation length with the equality in distribution result from Corollary 4.21 and the formula for the mean from Corollary 4.5. Plotted are means and stds estimated with respect to the distribution of parameters / random variables $\Xi_{l,i}(N(0,1),K)$ averaged over $100,000$ initializations, and a numerically evaluated formula for the mean from Corollary 4.5. All layers had 10 neurons. The lines for the mean and areas for the std overlap. Note that there is no std for the formula in the plot.

$\|\overline{\mathfrak{W}}^{(l)}\mathfrak{x}^{(l-1)}\|$. Then $\|\boldsymbol{x}^{(l')}\|^2$ equals

$$\begin{aligned}\|\boldsymbol{x}^{(l')}\|^2 = \|\overline{\mathfrak{W}}^{(l')}\mathfrak{x}^{(l'-1)}\|^2 &= \left\|\overline{\mathfrak{W}}^{(l')}\frac{\mathfrak{x}^{(l'-1)}}{\|\mathfrak{x}^{(l'-1)}\|}\right\|^2 \|\mathfrak{x}^{(l'-1)}\|^2 \\ &= \left\|\overline{\mathfrak{W}}^{(l')}\frac{\mathfrak{x}^{(l'-1)}}{\|\mathfrak{x}^{(l'-1)}\|}\right\|^2 \left(\left\|\overline{\mathfrak{W}}^{(l'-1)}\frac{\mathfrak{x}^{(l'-2)}}{\|\mathfrak{x}^{(l'-2)}\|}\right\|^2 \|\mathfrak{x}^{(l'-2)}\|^2 + 1\right) \\ &= \cdots = \|\mathfrak{x}^{(0)}\|^2 \prod_{l=1}^{l'} \left\|\overline{\mathfrak{W}}^{(l)}\frac{\mathfrak{x}^{(l-1)}}{\|\mathfrak{x}^{(l-1)}\|}\right\|^2 + \sum_{j=2}^{l'}\left[\prod_{l=j}^{l'}\left\|\overline{\mathfrak{W}}^{(l)}\frac{\mathfrak{x}^{(l-1)}}{\|\mathfrak{x}^{(l-1)}\|}\right\|^2\right],\end{aligned}$$

where we multiplied and divided $\|\overline{\mathfrak{W}}^{(l)}\mathfrak{x}^{(l-1)}\|^2$ by $\|\mathfrak{x}^{(l-1)}\|^2$ at each step. Using the approach from Theorem 4.3, more specifically equations (4.10), (4.12) and (4.13), with $\mathfrak{u}^{(l)} = \mathfrak{x}^{(l)}/\|\mathfrak{x}^{(l)}\|$, implying that $\cos\gamma_{\mathfrak{x}^{(l)},\mathfrak{u}^{(l)}} = 1$,

$$\begin{aligned}A^{(l')} = \frac{1}{n_l}\|\boldsymbol{x}^{(l')}\|^2 &\overset{d}{=} \|\mathfrak{x}^{(0)}\|^2\frac{1}{n_{l'}}\prod_{l=1}^{l'}\left(\frac{c}{n_{l-1}}\sum_{i=1}^{n_l}\Xi_{l,i}^2\right) + \frac{1}{n_{l'}}\sum_{j=2}^{l'}\left[\prod_{l=j}^{l'}\left(\frac{c}{n_{l-1}}\sum_{i=1}^{n_l}\Xi_{l,i}^2\right)\right] \\ &= \|\mathfrak{x}^{(0)}\|^2\frac{1}{n_0}\prod_{l=1}^{l'}\left(\frac{c}{n_l}\sum_{i=1}^{n_l}\Xi_{l,i}^2\right) + \sum_{j=2}^{l'}\left[\frac{1}{n_{j-1}}\prod_{l=j}^{l'}\left(\frac{c}{n_l}\sum_{i=1}^{n_l}\Xi_{l,i}^2\right)\right],\end{aligned}$$

where $\Xi_{l,i} = \Xi_{l,i}(N(0,1),K)$ is the largest order statistic in a sample of K standard Gaussian random variables, and stochastic independence of variables $\Xi_{l,i}$ follows from Theorem 4.3. □

Corollary 4.5 (Moments of the activation length)**.** *Consider a maxout network with the settings of Section 4.2. Let $\boldsymbol{x} \in \mathbb{R}^{n_0}$ be any input into the network. Then, for the moments of the normalized activation length, the following results hold.*

Mean:

$$\mathbb{E}\left[A^{(l')}\right] = \|\mathfrak{x}^{(0)}\|^2 \frac{1}{n_0} (c\mathcal{M})^{l'} + \sum_{j=2}^{l'} \left(\frac{1}{n_{j-1}} (c\mathcal{M})^{l'-j+1} \right).$$

Variance:

$$\operatorname{Var}[A^{(l')}] \leq 2 \frac{\|\mathfrak{x}^{(0)}\|^4}{n_0^2} c^{2l'} K^{2l'} \exp\left\{ \sum_{l=1}^{l'} \frac{4}{n_l K} \right\}.$$

Moments of the order $t \geq 2$:

$$\begin{aligned}
\mathbb{E}\left[\left(A^{(l')}\right)^t\right] &\leq 2^{t-1} \frac{\|\mathfrak{x}^{(0)}\|^{2t}}{n_0^t} c^{tl'} K^{tl'} \exp\left\{ \sum_{l=1}^{l'} \frac{t^2}{n_l K} \right\} \\
&\quad + (2(l'-1))^{t-1} \sum_{j=2}^{l'} \left(\frac{(cK)^{t(l'-j+1)}}{n_{j-1}^t} \exp\left\{ \sum_{l=j}^{l'} \frac{t^2}{n_l K} \right\} \right), \\
\mathbb{E}\left[\left(A^{(l')}\right)^t\right] &\geq \frac{\|\mathfrak{x}^{(0)}\|^{2t}}{n_0^t} (c\mathcal{M})^{tl'} + \sum_{j=2}^{l'} \frac{(c\mathcal{M})^{t(l'-j+1)}}{n_{j-1}^t}.
\end{aligned}$$

where expectation is taken with respect to the distribution of the network weights and biases, and $\mathcal{M}$ is a constant depending on K that can be computed approximately, see Table 4.9 for the values for $K = 2, \dots, 10$.

Proof. **Mean** Taking expectation in Corollary 4.21 and using independence of $\Xi_{l,i}(N(0,1), K)$,

$$\mathbb{E}\left[A^{(l')}\right] = \|\mathfrak{x}^{(0)}\|^2 \frac{1}{n_0} (c\mathcal{M})^{l'} + \sum_{j=2}^{l'} \left(\frac{1}{n_{j-1}} (c\mathcal{M})^{l'-j+1} \right), \tag{4.20}$$

where $\mathcal{M}$ is the second moment of $\Xi_{l,i}(N(0,1), K)$, see Table 4.9 for its values for $K = 2, \dots, 10$.

Moments of the order $t \geq 2$ Using Corollary 4.21, we get

$$\begin{aligned}
&\mathbb{E}\left[\left(A^{(l')}\right)^t\right] \\
&\quad = \mathbb{E}\left[\left(\|\mathfrak{x}^{(0)}\|^2 \frac{1}{n_0} \prod_{l=1}^{l'} \left(\frac{c}{n_l} \sum_{i=1}^{n_l} \Xi_{l,i}^2 \right) + \sum_{j=2}^{l'} \left[\frac{1}{n_{j-1}} \prod_{l=j}^{l'} \left(\frac{c}{n_l} \sum_{i=1}^{n_l} \Xi_{l,i}^2 \right) \right] \right)^t \right].
\end{aligned} \tag{4.21}$$

Upper bound First, we derive an upper bound on (4.21). Notice that all arguments in (4.21) are positive except for a zero measure set of $\Xi_{l,i} \in \mathbb{R}^{\sum_{l=1}^{l'} n_l}$. According to the power mean inequality, for any $x_1, \ldots, x_n \in \mathbb{R}, x_1, \ldots, x_n > 0$ and any $t \in \mathbb{R}, t > 1$, $(x_1 + \cdots + x_n)^t \leq n^{t-1}(x_1^t + \cdots + x_n^t)$. Using the power mean inequality first on the whole expression and then on the second summand,

$$\begin{aligned}\mathbb{E}\left[\left(A^{(l')}\right)^t\right] \leq 2^{t-1}\Bigg(\mathbb{E}\left[\left(\|\mathfrak{x}^{(0)}\|^2 \frac{1}{n_0} \prod_{l=1}^{l'} \left(\frac{c}{n_l} \sum_{i=1}^{n_l} \Xi_{l,i}^2\right)\right)^t\right] \\ + (l'-1)^{t-1} \sum_{j=2}^{l'} \mathbb{E}\left[\left(\frac{1}{n_{j-1}} \prod_{l=j}^{l'} \left(\frac{c}{n_l} \sum_{i=1}^{n_l} \Xi_{l,i}^2\right)\right)^t\right]\Bigg).\end{aligned} \tag{4.22}$$

Using independence of $\Xi_{l,i}$, (4.22) equals

$$\begin{aligned}2^{t-1} \frac{\|\mathfrak{x}^{(0)}\|^{2t}}{n_0^t} \prod_{l=1}^{l'} \left(\frac{c^t}{n_l^t} \mathbb{E}\left[\left(\sum_{i=1}^{n_l} \Xi_{l,i}^2\right)^t\right]\right) \\ + (2(l'-1))^{t-1} \sum_{j=2}^{l'} \left(\frac{1}{n_{j-1}^t} \prod_{l=j}^{l'} \left(\frac{c^t}{n_l^t} \mathbb{E}\left[\left(\sum_{i=1}^{n_l} \Xi_{l,i}^2\right)^t\right]\right)\right).\end{aligned}$$

Upper-bounding the largest order statistic with the sum of squared standard Gaussian random variables, we get that $\sum_{i=1}^{n_l} \Xi_{l,i}^2 \leq \chi^2_{n_l K}$. Hence,

$$\begin{aligned}\mathbb{E}\left[\left(A^{(l')}\right)^t\right] \leq 2^{t-1} \frac{\|\mathfrak{x}^{(0)}\|^{2t}}{n_0^t} \prod_{l=1}^{l'} \left(\frac{c^t}{n_l^t} \mathbb{E}\left[\left(\chi^2_{n_l K}\right)^t\right]\right) \\ + (2(l'-1))^{t-1} \sum_{j=2}^{l'} \left(\frac{1}{n_{j-1}^t} \prod_{l=j}^{l'} \left(\frac{c^t}{n_l^t} \mathbb{E}\left[\left(\chi^2_{n_l K}\right)^t\right]\right)\right).\end{aligned} \tag{4.23}$$

Using the formula for noncentral moments of the chi-squared distribution and $1+x \leq e^x, \forall x \in \mathbb{R}$,

$$\begin{aligned}&\frac{c^t}{n_l^t} \mathbb{E}\left[\left(\chi^2_{n_l K}\right)^t\right] = \frac{c^t}{n_l^t} (n_l K)(n_l K + 2) \cdots (n_l K + 2t - 2) \\ &= c^t K^t \cdot 1 \cdot \left(1 + \frac{2}{n_l K}\right) \cdots \left(1 + \frac{2t-2}{n_l K}\right) \leq c^t K^t \exp\left\{\sum_{i=0}^{t-1} \frac{2i}{n_l K}\right\} \leq c^t K^t \exp\left\{\frac{t^2}{n_l K}\right\},\end{aligned}$$

where we used the formula for calculating the sum of consecutive numbers $\sum_{i=1}^{t-1} i = t(t-1)/2$.

Using this result in (4.23), we get the final upper bound

$$\begin{aligned}\mathbb{E}\left[\left(A^{(l')}\right)^t\right] \le 2^{t-1}\frac{\|\mathfrak{x}^{(0)}\|^{2t}}{n_0^t} c^{tl'} K^{tl'} \exp\left\{\sum_{l=1}^{l'} \frac{t^2}{n_l K}\right\} \\ + (2(l'-1))^{t-1}\sum_{j=2}^{l'}\left(\frac{(cK)^{t(l'-j+1)}}{n_{j-1}^t}\exp\left\{\sum_{l=j}^{l'}\frac{t^2}{n_l K}\right\}\right).\end{aligned}$$

Lower bound Using that arguments in (4.21) are non-negative and $t \ge 1$, we can lower bound the power of the sum with the sum of the powers and get,

$$\begin{aligned}&\mathbb{E}\left[\left(A^{(l')}\right)^t\right] \\ &\ge \frac{\|\mathfrak{x}^{(0)}\|^{2t}}{n_0^t}\prod_{l=1}^{l'}\left(\frac{c^t}{n_l^t}\mathbb{E}\left[\left(\sum_{i=1}^{n_l}\Xi_{l,i}^2\right)^t\right]\right) + \sum_{j=2}^{l'}\left(\frac{1}{n_{j-1}^t}\prod_{l=j}^{l'}\left(\frac{c^t}{n_l^t}\mathbb{E}\left[\left(\sum_{i=1}^{n_l}\Xi_{l,i}^2\right)^t\right]\right)\right) \\ &\ge \frac{\|\mathfrak{x}^{(0)}\|^{2t}}{n_0^t}\prod_{l=1}^{l'}\left(\frac{c^t}{n_l^t}\left(\sum_{i=1}^{n_l}\mathbb{E}\left[\Xi_{l,i}^2\right]\right)^t\right) + \sum_{j=2}^{l'}\left(\frac{1}{n_{j-1}^t}\prod_{l=j}^{l'}\left(\frac{c^t}{n_l^t}\left(\sum_{i=1}^{n_l}\mathbb{E}\left[\Xi_{l,i}^2\right]\right)^t\right)\right),\end{aligned}$$

where we used the linearity of expectation in both expressions and Jensen's inequality in the last line. Using that $\mathbb{E}[(\Xi_{l,i}(N(0,1),K))^2] = \mathcal{M}$, see Table 4.9, we get

$$\mathbb{E}\left[\left(A^{(l')}\right)^t\right] \ge \frac{\|\mathfrak{x}^{(0)}\|^{2t}}{n_0^t}(c\mathcal{M})^{tl'} + \sum_{j=2}^{l'}\frac{(c\mathcal{M})^{t(l'-j+1)}}{n_{j-1}^t}. \tag{4.24}$$

Variance We can use an upper bound on the second moment as an upper bound on the variance.

□

Remark 4.22 (Zero bias)**.** Similar results can be obtained for the zero bias case and would result in the same bounds without the second summand. For the proof one would work directly with the vectors $\boldsymbol{x}^{(l)}$, without defining the vectors $\mathfrak{x}^{(l)}$, and to obtain the equality in distribution one would use Remark 4.17.

4.G Expected number of linear regions

Here we prove the result from Section 4.5.1.

Corollary 4.6 (Value for C_{grad})**.** *Consider a maxout network with the settings of Section 4.2. Assume that the biases are independent of the weights but otherwise initialized using any approach. Consider the* *pre-activation feature* $\zeta_{z,k}$ *of a unit* $z = 1, \dots, N$. *Then, for any* $t \in \mathbb{N}$,

$$\left(\sup_{\boldsymbol{x}\in\mathbb{R}^{n_0}} \mathbb{E}\left[\|\nabla\zeta_{z,k}(x)\|^t\right]\right)^{\frac{1}{t}} \leq n_0^{-\frac{1}{2}} \max\left\{1, (cK)^{\frac{L-1}{2}}\right\} \exp\left\{\frac{t}{2}\left(\sum_{l=1}^{L-1}\frac{1}{n_l K}+1\right)\right\}.$$

Proof. Distribution of $\nabla\zeta_{z,k}$ is the same as the distribution of the gradient with respect to the network input $\nabla\widetilde{\mathcal{N}}_1(\boldsymbol{x})$ in a maxout network that has a single linear output unit and $\tilde{L} = l(z)$ layers, where $l(z)$ is the depth of a unit z. Therefore, we will consider $(\sup_{\boldsymbol{x}\in\mathbb{R}^{n_0}} \mathbb{E}[\|\nabla\widetilde{\mathcal{N}}_1(\boldsymbol{x})\|^{2t}])^{1/2t}$. Notice that since $n_{\tilde{L}} = 1$, $\nabla\widetilde{\mathcal{N}}_1(\boldsymbol{x}) = \boldsymbol{J}_{\widetilde{\mathcal{N}}}(\boldsymbol{x})^T = \boldsymbol{J}_{\widetilde{\mathcal{N}}}(\boldsymbol{x})^T\boldsymbol{u}$ for a 1-dimensional vector $\boldsymbol{u} = (1)$. Hence,

$$\|\nabla\widetilde{\mathcal{N}}_1(\boldsymbol{x})\| = \sup_{\|\boldsymbol{u}\|=1, \boldsymbol{u}\in\mathbb{R}^{n_{\tilde{L}}}} \|\boldsymbol{J}_{\widetilde{\mathcal{N}}}(\boldsymbol{x})^T\boldsymbol{u}\| = \|\boldsymbol{J}_{\widetilde{\mathcal{N}}}(\boldsymbol{x})^T\|, \tag{4.25}$$

where the matrix norm is the spectral norm. Using that a matrix and its transpose have the same spectral norm, (4.25) equals

$$\|\boldsymbol{J}_{\widetilde{\mathcal{N}}}(\boldsymbol{x})\| = \sup_{\|\boldsymbol{u}\|=1, \boldsymbol{u}\in\mathbb{R}^{n_0}} \|\boldsymbol{J}_{\widetilde{\mathcal{N}}}(\boldsymbol{x})\boldsymbol{u}\|.$$

Therefore, we need to upper bound

$$\begin{aligned}
&\left(\sup_{\boldsymbol{x}\in\mathbb{R}^{n_0}} \mathbb{E}\left[\left(\sup_{\|\boldsymbol{u}\|=1, \boldsymbol{u}\in\mathbb{R}^{n_0}} \|\boldsymbol{J}_{\widetilde{\mathcal{N}}}(\boldsymbol{x})\boldsymbol{u}\|\right)^t\right]\right)^{\frac{1}{t}} \\
&\qquad \leq \left(\sup_{\boldsymbol{x}\in\mathbb{R}^{n_0}} \mathbb{E}\left[\left(\sup_{\|\boldsymbol{u}\|=1, \boldsymbol{u}\in\mathbb{R}^{n_0}} \|\boldsymbol{J}_{\widetilde{\mathcal{N}}}(\boldsymbol{x})\boldsymbol{u}\|\right)^{2t}\right]\right)^{\frac{1}{2t}},
\end{aligned}$$

where we used Jensen's inequality.

Now we can use an upper bound on $\mathbb{E}[\|\boldsymbol{J}_{\widetilde{\mathcal{N}}}(\boldsymbol{x})\boldsymbol{u}\|^{2t}]$ from Corollary 4.2, which holds for any $\boldsymbol{x}, \boldsymbol{u} \in \mathbb{R}^{n_0}$, $\|\boldsymbol{u}\| = 1$, and thus holds for the suprema. Recalling that $n_{\tilde{L}} = 1$, we get

$$\mathbb{E}\left[\|\boldsymbol{J}_{\widetilde{\mathcal{N}}}(\boldsymbol{x})\boldsymbol{u}\|^{2t}\right] \leq \left(\frac{1}{n_0}\right)^t (cK)^{t(\tilde{L}-1)} \exp\left\{t^2\left(\sum_{l=1}^{\tilde{L}-1}\frac{1}{n_l K}+1\right)\right\}.$$

Hence,

$$\left(\mathbb{E}\left[\|\boldsymbol{J}_{\widetilde{\mathcal{N}}}(\boldsymbol{x})\boldsymbol{u}\|^{2t}\right]\right)^{\frac{1}{2t}} \leq n_0^{-\frac{1}{2}} (cK)^{\frac{\tilde{L}-1}{2}} \exp\left\{\frac{t}{2}\left(\sum_{l=1}^{\tilde{L}-1}\frac{1}{n_l K}+1\right)\right\}.$$

Taking the maximum over $\tilde{L} \in \{1, \dots, L\}$, the final upper bound is

$$n_0^{-\frac{1}{2}} \max\left\{1, (cK)^{\frac{L-1}{2}}\right\} \exp\left\{\frac{t}{2}\left(\sum_{l=1}^{L-1} \frac{1}{n_l K} + 1\right)\right\}.$$

$\square$

Now we provide an updated upper bound on the number of r-partial activation regions from Tseran and Montúfar (2021, Theorem 9). In this bound, the case $r = 0$ corresponds to the number of linear regions. For a detailed discussion of the activation regions of maxout networks and their differences from linear regions, see Tseran and Montúfar (2021). Since the proof of Tseran and Montúfar (2021, Theorem 9) only uses C_{grad} for $t \leq n_0$, we obtain the following statement.

Theorem 4.23 (Upper bound on the expected number of partial activation regions)**.** *Consider a maxout network with the settings of Section 4.2 with N maxout units. Assume that the biases are independent of the weights and initialized so that:*

1. *Every collection of biases has a conditional density with respect to Lebesgue measure given the values of all other weights and biases.*
2. *There exists $C_{\text{bias}} > 0$ so that for any pre-activation features $\zeta_1, \dots, \zeta_t$ from any neurons, the conditional density of their biases $\rho_{b_1,\dots,b_t}$ given all the other weights and biases satisfies*

$$\sup_{b_1,\dots,b_t \in \mathbb{R}} \rho_{b_1,\dots,b_t}(b_1, \dots, b_t) \leq C_{\text{bias}}^t.$$

Fix $r \in \{0, \dots, n_0\}$. Let $C_{\text{grad}} = n_0^{-1/2} \max\{1, (cK)^{(L-1)/2}\} \exp\{n_0/2(\sum_{l=1}^{L-1} 1/(n_l K) + 1)\}$ and $T = 2^5 C_{\text{grad}} C_{\textit{bias}}$. Then, there exists $\delta_0 \leq 1/(2C_{\text{grad}} C_{\textit{bias}})$ such that for all cubes $C \subseteq \mathbb{R}^{n_0}$ with side length $\delta > \delta_0$ we have

$$\frac{\mathbb{E}[\#\ r\text{-partial activation regions of } \mathcal{N} \text{ in } C]}{\text{vol}(C)} \leq \begin{cases} \binom{rK}{2r}\binom{N}{r} K^{N-r}, & N \leq n_0 \\ \frac{(TKN)^{n_0}\binom{n_0 K}{2n_0}}{(2K)^r n_0!}, & N \geq n_0 \end{cases}.$$

Here the expectation is taken with respect to the distribution of weights and biases in $\mathcal{N}$. Of particular interest is the case $r = 0$, which corresponds to the number of linear regions.

4.H Expected curve length distortion

In this section, we prove the result from Section 4.5.2.

Let M be a smooth 1-dimensional curve in $\mathbb{R}^{n_0}$. Fix a smooth unit speed parameterization of $M = \gamma([0, 1])$ with $\gamma : \mathbb{R} \to \mathbb{R}^{n_0}, \gamma(\tau) = (\gamma_1(\tau), \dots, \gamma_{n_0}(\tau))$. Then, parametrization of the curve

$\mathcal{N}(M)$ is given by a mapping $\Gamma := \mathcal{N} \circ \gamma, \Gamma : \mathbb{R} \to \mathbb{R}^{n_L}$. Thus, the length of $\mathcal{N}(M)$ is

$$\operatorname{len}(\mathcal{N}(M)) = \int_0^1 \|\Gamma'(\tau)\| d\tau.$$

Notice that the input-output Jacobian of maxout networks is well defined almost everywhere because for any neuron, using that the biases are independent from weights, and the weights are initialized from a continuous distribution, $P(k' = \operatorname{argmax}_{k\in[K]}\{W_{i,k}^{(l)}\boldsymbol{x}^{(l-1)} + b_{i,k}^{(l)}\}, k'' = \operatorname{argmax}_{k\in[K]}\{W_{i,k}^{(l)}\boldsymbol{x}^{(l-1)} + b_{i,k}^{(l)}\}) = 0, i = 1, \dots, n_{L-1}$. Hence, $\Gamma'(\tau) = \boldsymbol{J}_{\mathcal{N}}(\gamma(\tau))\gamma'(\tau)$, where we used the chain rule, and we can employ the following lemma from Hanin et al. (2021). We state it here without proof which uses Tonelli's theorem, power mean inequality and chain rule.

Lemma 4.24 (Connection between the length of the curve and $\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|$, Hanin et al. 2021, Lemma C.1)**.** *For any integer $t \geq 0$,*

$$\mathbb{E}\left[\operatorname{len}(\mathcal{N}(M))^t\right] \leq \int_0^1 \mathbb{E}\left[\|\boldsymbol{J}_{\mathcal{N}}(\gamma(\tau))\gamma'(\tau)\|^t\right] d\tau = \mathbb{E}\left[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^t\right],$$

where $\boldsymbol{u} \in \mathbb{R}^{n_0}$ is a unit vector.

Now we are ready to proof Corollary 4.7.

Corollary 4.7 (Expected curve length distortion)**.** *Consider a maxout network with the settings of Section 4.2. Assume that the biases are independent of the weights but otherwise initialized using any approach. Let M be a smooth 1-dimensional curve of unit length in $\mathbb{R}^{n_0}$. Then, the following upper bounds on the moments of* $\operatorname{len}(\mathcal{N}(M))$ *hold:*

$$\begin{aligned}
&\mathbb{E}\left[\operatorname{len}(\mathcal{N}(M))\right] \leq \left(\frac{n_L}{n_0}\right)^{\frac{1}{2}} (c\mathcal{L})^{\frac{L-1}{2}}, \\
&\operatorname{Var}\left[\operatorname{len}(\mathcal{N}(M))\right] \leq \frac{n_L}{n_0}(c\mathcal{L})^{L-1}, \\
&\mathbb{E}\left[\operatorname{len}(\mathcal{N}(M))^t\right] \leq \left(\frac{n_L}{n_0}\right)^{\frac{t}{2}} (cK)^{\frac{t(L-1)}{2}} \exp\left\{\frac{t^2}{2}\left(\sum_{l=1}^{L-1}\frac{1}{n_l K} + \frac{1}{n_L}\right)\right\},
\end{aligned}$$

where $\mathcal{L}$ is a constant depending on K, see Table 4.9 in Section 4.D for values for $K = 2, \dots, 10$.

Proof. By Lemma 4.24,

$$\mathbb{E}\left[\operatorname{len}(\mathcal{N}(M))^t\right] \leq \mathbb{E}\left[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^t\right] \leq \left(\mathbb{E}\left[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^{2t}\right]\right)^{\frac{1}{2}},$$

where we used Jensen's inequality to obtain the last upper bound. Hence, using Corollary 4.2, we get the following upper bounds on the moments on the length of the curve.

Mean

$$\mathbb{E}\left[\mathrm{len}(\mathcal{N}(M))\right] \leq \left(\mathbb{E}\left[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^2\right]\right)^{\frac{1}{2}} \leq \left(\frac{n_L}{n_0}\right)^{\frac{1}{2}} (c\mathcal{L})^{\frac{L-1}{2}}.$$

Variance

$$\mathrm{Var}\left[\mathrm{len}(\mathcal{N}(M))\right] \leq \mathbb{E}\left[\mathrm{len}(\mathcal{N}(M))^2\right] \leq \frac{n_L}{n_0}(c\mathcal{L})^{L-1}.$$

Moments of the order $t \geq 3$

$$\mathbb{E}\left[\mathrm{len}(\mathcal{N}(M))^t\right] \leq \left(\mathbb{E}\left[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x})\boldsymbol{u}\|^{2t}\right]\right)^{\frac{1}{2}} \leq \left(\frac{n_L}{n_0}\right)^{\frac{t}{2}} (cK)^{\frac{t(L-1)}{2}} \exp\left\{\frac{t^2}{2}\left(\sum_{l=1}^{L-1}\frac{1}{n_l K} + \frac{1}{n_L}\right)\right\}.$$

□

4.I NTK

Here we prove the results from Section 4.5.3.

Corollary 4.9 (On-diagonal NTK). *Consider a maxout network with the settings of Section 4.2. Assume that $n_L = 1$ and that the biases are initialized to zero and are not trained. Assume that $\mathcal{S} \leq c \leq \mathcal{L}$, where the constants $\mathcal{S}, \mathcal{L}$ are as specified in Table 4.9. Then,*

$$\begin{aligned}
\|\mathfrak{x}^{(0)}\|^2 \frac{(c\mathcal{S})^{L-2}}{n_0} P \leq \mathbb{E}[K_{\mathcal{N}}(\boldsymbol{x},\boldsymbol{x})] &\leq \|\mathfrak{x}^{(0)}\|^2 \frac{(c\mathcal{L})^{L-2}\mathcal{M}^{L-1}}{n_0} P, \\
\mathbb{E}[K_{\mathcal{N}}(\boldsymbol{x},\boldsymbol{x})^2] &\leq 2PP_W(cK)^{2(L-2)}\frac{\|\mathfrak{x}^{(0)}\|^4}{n_0^2}\exp\left\{\sum_{j=1}^{L-1}\frac{4}{n_j K} + 4\right\},
\end{aligned}$$

where $P = \sum_{l=0}^{L-1} n_l$, $P_W = \sum_{l=0}^{L} n_l n_{l-1}$, and $\mathcal{M}$ is as specified in Table 4.9.

Proof. Under the assumption that biases are not trained, on-diagonal NTK of a maxout network is

$$K_{\mathcal{N}}(\boldsymbol{x},\boldsymbol{x}) = \sum_{l=1}^{L}\sum_{i=1}^{n_l}\sum_{k=1}^{K}\sum_{j=1}^{n_{l-1}}\left(\frac{\partial\mathcal{N}}{\partial W_{i,k,j}^{(l)}}(\boldsymbol{x})\right)^2.$$

Since in maxout network for all k-s except $k = k' = \mathrm{argmax}_{k\in[K]}\{W_{i,k}^{(l)}\boldsymbol{x}^{(l-1)} + \boldsymbol{b}_{i,k}^{(l)}\}$ the derivatives with respect to the weights and biases are zero, on-diagonal NTK equals

$$K_{\mathcal{N}}(\boldsymbol{x},\boldsymbol{x}) = \sum_{l=1}^{L}\sum_{i=1}^{n_l}\sum_{j=1}^{n_{l-1}}\left(\frac{\partial\mathcal{N}}{\partial W_{i,k',j}^{(l)}}(\boldsymbol{x})\right)^2.$$

Notice that since we assumed a continuous distribution over the network weights and the biases are zero, the partial derivatives are defined everywhere except for the set of measure zero.

Part I. Kernel mean $\mathbb{E}[\boldsymbol{K}_{\mathcal{N}}(\boldsymbol{x},\boldsymbol{x})]$ Firstly, using the chain rule, a partial derivative with respect to the network weight is

$$\frac{\partial \mathcal{N}}{\partial W^{(l)}_{i,k',j}}(\boldsymbol{x}) = \frac{\partial \mathcal{N}}{\partial \boldsymbol{x}^{(l)}_i}(\boldsymbol{x})\boldsymbol{x}^{(l-1)}_j = \boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x}^{(l)})\boldsymbol{e}_i\boldsymbol{x}^{(l-1)}_j.$$

Recall that we assumed $n_L = 1$. Therefore, we need to consider $(\partial\mathcal{N}(\boldsymbol{x})\partial W^{(l)}_{i,k',j})^2 = \|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x}^{(l)})\boldsymbol{u}\|^2(\boldsymbol{x}^{(l-1)}_j)^2$, where $\boldsymbol{u} = \boldsymbol{e}_i$. Combining Theorem 4.1 and Corollary 4.21 in combination with Remark 4.22 for the zero-bias case and using the independence of the random variables in the expressions,

$$\begin{aligned}\mathbb{E}\left[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x}^{(l)})\boldsymbol{u}\|^2\left(\boldsymbol{x}^{(l-1)}_j\right)^2\right] &\leq_{so} \mathbb{E}\left[\frac{1}{n_l}\chi^2_{n_L}\prod_{j=l}^{L-1}\frac{c}{n_j}\sum_{i=1}^{n_j}\Xi_{j,i}(\chi^2_1,K)\right] \\ &\quad\cdot\mathbb{E}\left[\|\mathfrak{x}^{(0)}\|^2\frac{1}{n_0}\prod_{j=1}^{l-1}\left(\frac{c}{n_j}\sum_{i=1}^{n_j}\Xi_{j,i}(N(0,1),K)^2\right)\right],\end{aligned} \tag{4.26}$$

where we treat the $(l-1)$th layer as if it has one unit when we use the normalized activation length result. Then, using Corollaries 4.2 and 4.5,

$$\mathbb{E}\left[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x}^{(l)})\boldsymbol{u}\|^2\left(\boldsymbol{x}^{(l-1)}_j\right)^2\right] \leq \|\mathfrak{x}^{(0)}\|^2\frac{c^{L-2}}{n_0 n_l}\mathcal{L}^{L-l-1}\mathcal{M}^{l-1}.$$

Taking the sum, we get

$$\begin{aligned}\mathbb{E}[K_{\mathcal{N}}(\boldsymbol{x},\boldsymbol{x})] = \mathbb{E}[K_W] &= \mathbb{E}\left[\sum_{l=1}^{L}\sum_{i=1}^{n_l}\sum_{j=1}^{n_{l-1}}\left(\frac{\partial\mathcal{N}}{\partial W^{(l)}_{i,k',j}}(\boldsymbol{x})\right)^2\right] \\ &\leq \sum_{l=1}^{L}\sum_{i=1}^{n_l}\sum_{j=1}^{n_{l-1}}\|\mathfrak{x}^{(0)}\|^2\frac{c^{L-2}}{n_0 n_l}\mathcal{L}^{L-l-1}\mathcal{M}^{l-1} \leq \|\mathfrak{x}^{(0)}\|^2\frac{(c\mathcal{L})^{L-2}\mathcal{M}^{L-1}}{n_0}P,\end{aligned}$$

where $P = \sum_{l=0}^{L-1} n_l$ denotes the number of neurons in the network up to the last layer, but including the input neurons. Here we used that for $K \geq 2$, both $\mathcal{L},\mathcal{M} \geq 1$, see Table 4.9. Similarly,

$$\mathbb{E}[K_W] \geq \sum_{l=1}^{L}\sum_{i=1}^{n_l}\sum_{j=1}^{n_{l-1}}\|\mathfrak{x}^{(0)}\|^2\frac{c^{L-2}}{n_0 n_l}\mathcal{S}^{L-l-1}\mathcal{M}^{l-1} \geq \|\mathfrak{x}^{(0)}\|^2\frac{(c\mathcal{S})^{L-2}}{n_0}P.$$

Here we used that for $K \geq 2$, $\mathcal{S} \leq 1$ and $\mathcal{M} \geq 1$, see Table 4.9.

Part II. Second moment $\mathbb{E}[\boldsymbol{K_{\mathcal{N}}(x,x)^2}]$ Using equation (4.26) with Corollaries 4.2 and 4.5,

$$
\begin{aligned}
\mathbb{E}\left[\left(\frac{\partial \mathcal{N}}{\partial W^{(l)}_{i,k',j}}(\boldsymbol{x})\right)^4\right] &= \mathbb{E}\left[\|\boldsymbol{J}_{\mathcal{N}}(\boldsymbol{x}^{(l)})\boldsymbol{u}\|^4\left(\boldsymbol{x}^{(l-1)}_j\right)^4\right] \\
&\leq_{so} \mathbb{E}\left[\left(\frac{1}{n_l}\chi^2_{n_L}\prod_{j=l}^{L-1}\frac{c}{n_j}\sum_{i=1}^{n_j}\Xi_{j,i}(\chi^2_1, K)\right)^2\right] \qquad (4.27) \\
&\quad \cdot \mathbb{E}\left[\left(\|\mathfrak{x}^{(0)}\|^2\frac{1}{n_0}\prod_{j=1}^{l-1}\left(\frac{c}{n_j}\sum_{i=1}^{n_j}\Xi_{j,i}(N(0,1),K)^2\right)\right)^2\right] \\
&\leq 2(cK)^{2(L-2)}\frac{\|\mathfrak{x}^{(0)}\|^4}{n_l^2 n_0^2}\exp\left\{4\left(\sum_{j=1}^{L-1}\frac{1}{n_jK}-\frac{1}{n_lK}+1\right)\right\}. \qquad (4.28)
\end{aligned}
$$

Notice that all summands are non-negative. Then, using AM-QM inequality,

$$
\begin{aligned}
\mathbb{E}[K_{\mathcal{N}}(\boldsymbol{x},\boldsymbol{x})^2] &= \mathbb{E}\left[\left(\sum_{l=1}^{L}\sum_{i=1}^{n_l}\sum_{j=1}^{n_{l-1}}\left(\frac{\partial \mathcal{N}}{\partial W^{(l)}_{i,k',j}}(\boldsymbol{x})\right)^2\right)^2\right] \\
&\leq P_W\sum_{l=1}^{L}\sum_{i=1}^{n_l}\sum_{j=1}^{n_{l-1}}\mathbb{E}\left[\left(\frac{\partial \mathcal{N}}{\partial W^{(l)}_{i,k',j}}(\boldsymbol{x})\right)^4\right] \\
&\leq P_W\sum_{l=1}^{L}\sum_{i=1}^{n_l}\sum_{j=1}^{n_{l-1}} 2(cK)^{2(L-2)}\frac{\|\mathfrak{x}^{(0)}\|^4}{n_l^2 n_0^2}\exp\left\{4\left(\sum_{j=1}^{L-1}\frac{1}{n_jK}-\frac{1}{n_lK}+1\right)\right\} \\
&\leq 2PP_W(cK)^{2(L-2)}\frac{\|\mathfrak{x}^{(0)}\|^4}{n_0^2}\exp\left\{\sum_{j=1}^{L-1}\frac{4}{n_jK}+4\right\},
\end{aligned}
$$

where $P_W = \sum_{l=0}^{L} n_l n_{l-1}$ denotes the number of all weights in the network. ☐

4.J Experiment details and additional experiments

4.J.1 Experiments with SGD and Adam from Section 4.6

In this subsection, we provide more details on the experiments presented in Section 4.6. The implementation of the key routines is available at `https://github.com/hanna-tseran/maxout_expected_gradients`. Experiments were implemented in Python using TensorFlow (Martín Abadi et al., 2015), numpy (Harris et al., 2020) and mpi4py (Dalcin et al., 2011). The plots were created using matplotlib (Hunter, 2007). We conducted all training experiments from Section 4.6 on a GPU cluster with nodes having 4 Nvidia A100 GPUs with 40 GB of memory. The most extensive experiments were running for one day on one GPU. Experiment in Figure 4.2 was run on a CPU cluster that uses Intel Xeon IceLakeSP processors (Platinum 8360Y) with 72 cores per node and 256 GB RAM. All other experiments were executed on the laptop ThinkPad T470 with Intel Core i5-7200U CPU with 16 GB RAM.

Training experiments Now we discuss the training experiments. We use MNIST (LeCun and Cortes, 2010), Iris (Fisher, 1936), Fashion MNIST (Xiao et al., 2017), SVHN (Netzer et al., 2011), CIFAR-10 and CIFAR-100 datasets (Krizhevsky et al., 2009). All maxout networks have the maxout rank $K = 5$. Weights are sampled from $N(0, c/\text{fan-in})$ in fully-connected networks and $N(0, c/(k^2 \cdot \text{fan-in}))$, where k is the kernel size, in CNNs. The biases are initialized to zero. ReLU networks are initialized using He approach (He et al., 2015), meaning that $c = 2$. All results are averaged over 4 runs. We do not use any weight normalization techniques, such as batch normalization (Ioffe and Szegedy, 2015). We performed the dataset split into training, validation and test dataset and report the accuracy on the test set, while the validation set was used only for picking the hyper-parameters and was not used in training. The mini-batch size in all experiments is 32. The number of training epochs was picked by observing the training set loss and choosing the number of epochs for which the loss has converged. The exception is the SVHN dataset, for which we observe the double descent phenomenon and stop training after 150 epochs.

Network architecture Fully connected networks have 21 layers. Specifically, their architecture is

$$[5\times\text{fc}64,\ 5\times\text{fc}32,\ 5\times\text{fc}16,\ 5\times\text{fc}8,\ \text{out}],$$

where "$5\times$fc64" means that there are 5 fully-connected layers with 64 neurons, and "out" stands for the output layer that has the number of neurons equal to the number of classes in a dataset. CNNs have a VGG-19-like architecture (Simonyan and Zisserman, 2015) with 20 or 16 layers, depending the input size. The 20-layer architecture is

$$\begin{aligned}[2\times\text{conv}64,\ \text{mp},\ 2\times\text{conv}128,\ \text{mp},\ 4\times\text{conv}256,\ \text{mp},\ 4\times\text{conv}512,\ \text{mp},\ 4\times\text{conv}512,\ \text{mp},\\ 2\times\text{fc}4096,\ \text{fc}1000,\ \text{out}],\end{aligned}$$

where "conv64" stands for a convolutional layer with 64 neurons and "mp" for a max-pooling layer. The kernel size in all convolutional layers is 3×3. Max-pooling uses 2×2 pooling windows with stride 2. Such architecture is used for datasets with the images that have the side length greater or equal to 32: CIFAR-10, CIFAR-100 and SVHN. The 16-layer architecture is used for images with the smaller image size: MNIST and Fashion MNIST. This architecture does not have the last convolutional block of the 20-layer version. Concretely, it has the following layers:

$$[2\times\text{conv}64,\ \text{mp},\ 2\times\text{conv}128,\ \text{mp},\ 4\times\text{conv}256,\ \text{mp},\ 4\times\text{conv}512,\ \text{mp}, 2\times\text{fc}4096,\ \text{fc}1000,\ \text{out}],$$

Max-pooling initialization To account for the maximum in max-pooling layers, a maxout layer appearing after a max-pooling layer is initialized as if its maxout rank was $K \times m^2$, where m^2 is the max-pooling window size. The reason for this is that the outputs of a computational block consisting of a max-pooling window and a maxout layer are taking maxima over $K \times m^2$ linear functions, $\max\{W_1 \max\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{m^2}\} + \boldsymbol{b}_1, \dots, W_K \max\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{m^2}\} + \boldsymbol{b}_K\} = \max\{f_1, \dots, f_{Km^2}\}$, where the f_i are Km^2 affine functions. Therefore, we initialize the layers that follow max-pooling layers using the criterion for maxout rank $m^2 \times K$ instead of K. In our experiments, $K = 5, m = 2$, and $m^2 \times K = 20$. Hence, for such layers, we use the constant $c = 1/M = 0.26573$, where M is computed for $K = 20$ using the formula from Remark 4.14 in Section 4.D. All other layers that do not follow max-pooling layers are initialized as suggested in Section 4.4. We observe that max-pooling initialization often leads to slightly higher accuracy.

Data augmentation There is no data augmentation for fully connected networks. For convolutional networks, for MNIST, Fashion MNIST and SVHN datasets we perform random translation, rotation and zoom of the input images. For CIFAR-10 and CIFAR-100, we additionally apply a random horizontal flip.

Learning rate decay In all experiments, we use the learning rate decay and choose the optimal initial learning rate for all network and initialization types based on their accuracy on the validation dataset using grid search. The learning rate was halved every nth epoch. For SVHN, $n = 10$, and for all other datasets, $n = 100$.

SGD with momentum We use SGD with Nesterov momentum, with the momentum value of 0.9. Specific dataset settings are the following.

- MNIST (fully-connected networks). Networks are trained for 600 epochs. The learning rate is halved every 100 epochs. Learning rates: maxout networks with maxout initialization: 0.002, maxout networks with $c = 0.1$: 0.002, maxout networks with $c = 2$: 2×10^{-7}, ReLU networks: 0.002.

- Iris. Networks are trained for 500 epochs. The learning rate is halved every 100 epochs. Learning rates: maxout networks with maxout initialization: 0.01, maxout networks with $c = 0.1$: 0.01, maxout networks with $c = 2$: 4×10^{-8}, ReLU networks: 0.005.

- MNIST (convolutional networks). Networks are trained for 800 epochs. The learning rate is halved every 100 epochs. Learning rates: maxout networks with maxout initialization: 0.009, maxout networks with max-pooling initialization: 0.009, maxout networks with $c = 0.1$: 0.009, maxout networks with $c = 2$: 8×10^{-6}, ReLU networks: 0.01.

- Fashion MNIST. Networks are trained for 800 epochs. The learning rate is halved every 100 epochs. Learning rates: maxout networks with maxout initialization: 0.004, maxout networks with max-pooling initialization: 0.006, maxout networks with $c = 0.1$: 0.4, maxout networks with $c = 2$: 5×10^{-6}, ReLU networks: 0.01.
- CIFAR-10. Networks are trained for 1000 epochs. The learning rate is halved every 100 epochs. Learning rates: maxout networks with maxout initialization: 0.004, maxout networks with max-pooling initialization: 0.005, maxout networks with $c = 0.1$: 0.5, maxout networks with $c = 2$: 8×10^{-8}, ReLU networks: 0.009.
- CIFAR-100. Networks are trained for 1000 epochs. The learning rate is halved every 100 epochs. Learning rates: maxout networks with maxout initialization: 0.002, maxout networks with max-pooling initialization: 0.002, maxout networks with $c = 0.1$: 0.002, maxout networks with $c = 2$: 8×10^{-5}, ReLU networks: 0.006.
- SVHN. Networks are trained for 150 epochs. The learning rate is halved every 10 epochs. Learning rates: maxout networks with maxout initialization: 0.005, maxout networks with max-pooling initialization: 0.005, maxout networks with $c = 0.1$: 0.005, maxout networks with $c = 2$: 7×10^{-5}, ReLU networks: 0.005.

Adam We use Adam optimizer Kingma and Ba (2015) with default TensorFlow parameters $\beta_1 = 0.9, \beta_2 = 0.999$. Specific dataset settings are the following.

- MNIST (fully-connected networks). Networks are trained for 600 epochs. The learning rate is halved every 100 epochs. Learning rates: maxout networks with maxout initialization: 0.0008, maxout networks with $c = 2$: 0.0007, ReLU networks: 0.0008.
- MNIST (convolutional networks). Networks are trained for 800 epochs. The learning rate is halved every 100 epochs. Learning rates: maxout networks with maxout initialization: 0.0001, maxout networks with max-pooling initialization: 0.00006, maxout networks with $c = 2$: 0.00004, ReLU networks: 0.00009.
- Fashion MNIST. Networks are trained for 1000 epochs. The learning rate is halved every 100 epochs. Learning rates: maxout networks with maxout initialization: 0.00007, maxout networks with max-pooling initialization: 0.00008, maxout networks with $c = 2$: 0.00005, ReLU networks: 0.0002.
- CIFAR-10. Networks are trained for 1000 epochs. The learning rate is halved every 100 epochs. Learning rates: maxout networks with maxout initialization: 0.00009, maxout networks with

Table 4.10: Ablation study of the value of c. Reported is accuracy on the test set for maxout networks with maxout rank $K = 5$ trained using SGD with Nesterov momentum. Observe that the optimal value of c is close to $c = 0.55555$ which is suggested in Section 4.4.

VALUE OF c	FULLY-CONNECTED		CONVOLUTIONAL			
	MNIST	Iris	MNIST	CIFAR-10	CIFAR-100	Fashion MNIST
0.01	$11.35^{\pm0.00}$	$30^{\pm0.00}$	$11.35^{\pm0.00}$	$10^{\pm0.00}$	$1^{\pm0.00}$	$10^{\pm0.00}$
0.05	$11.35^{\pm0.00}$	$30^{\pm0.00}$	$11.35^{\pm0.00}$	$10^{\pm0.00}$	$1^{\pm0.00}$	$10^{\pm0.00}$
0.07	$11.35^{\pm0.00}$	$31.67^{\pm2.89}$	$11.35^{\pm0.00}$	$10^{\pm0.00}$	$1^{\pm0.00}$	$10^{\pm0.00}$
0.1	$11.35^{\pm0.00}$	$30^{\pm0.00}$	$11.35^{\pm0.00}$	$10^{\pm0.00}$	$1^{\pm0.00}$	$10^{\pm0.00}$
0.2	$11.35^{\pm0.00}$	$30^{\pm0.00}$	$99.56^{\pm0.03}$	$10^{\pm0.00}$	$1^{\pm0.00}$	$93.21^{\pm0.11}$
0.3	$97.63^{\pm0.16}$	$60.83^{\pm30.86}$	$99.55^{\pm0.02}$	$90.97^{\pm0.11}$	$64.71^{\pm0.25}$	$93.41^{\pm0.11}$
0.4	$97.89^{\pm0.12}$	$85^{\pm10.67}$	$\mathbf{99.6^{\pm0.03}}$	$91.15^{\pm0.07}$	$64.9^{\pm0.33}$	$93.21^{\pm0.11}$
0.5	$97.82^{\pm0.09}$	$\mathbf{92.5^{\pm1.44}}$	$99.56^{\pm0.05}$	$91.33^{\pm0.13}$	$65.48^{\pm0.43}$	$93.5^{\pm0.15}$
0.55555	$\mathbf{97.92^{\pm0.18}}$	$90.83^{\pm3.63}$	$99.57^{\pm0.07}$	$91.4^{\pm0.22}$	$65.38^{\pm0.32}$	$93.51^{\pm0.08}$
0.6	$97.77^{\pm0.17}$	$90.83^{\pm1.44}$	$99.59^{\pm00.02}$	$\mathbf{91.69^{\pm0.25}}$	$65.58^{\pm0.24}$	$93.54^{\pm0.13}$
0.7	$97.91^{\pm0.11}$	$90^{\pm0.00}$	$54.69^{\pm44.89}$	$50.83^{\pm40.83}$	$\mathbf{66.26^{\pm0.42}}$	$\mathbf{93.62^{\pm0.23}}$
0.8	$75.82^{\pm38.12}$	$30^{\pm0.00}$	$9.8^{\pm0.00}$	$10^{\pm0.00}$	$1^{\pm0.00}$	$72.66^{\pm36.17}$
0.9	$75.94^{\pm38.18}$	$30^{\pm0.00}$	$9.8^{\pm0.00}$	$10^{\pm0.00}$	$1^{\pm0.00}$	$10^{\pm0.00}$
1	$97.89^{\pm0.10}$	$30^{\pm0.00}$	$9.8^{\pm0.00}$	$10^{\pm0.00}$	$1^{\pm0.00}$	$10^{\pm0.00}$
1.5	$9.8^{\pm0.00}$	$30^{\pm0.00}$	$9.8^{\pm0.00}$	$10^{\pm0.00}$	$1^{\pm0.00}$	$10^{\pm0.00}$
2	$9.8^{\pm0.00}$	$30^{\pm0.00}$	$9.8^{\pm0.00}$	$10^{\pm0.00}$	$1^{\pm0.00}$	$10^{\pm0.00}$
10	$9.8^{\pm0.00}$	$30^{\pm0.00}$	$9.8^{\pm0.00}$	$10^{\pm0.00}$	$1^{\pm0.00}$	$10^{\pm0.00}$

max-pooling initialization: 0.00009, maxout networks with $c = 2$: 0.00005, ReLU networks: 0.0001.

- CIFAR-100. Networks are trained for 1000 epochs. The learning rate is halved every 100 epochs. Learning rates: maxout networks with maxout initialization: 0.00008, maxout networks with max-pooling initialization: 0.00009, maxout networks with $c = 2$: 0.00005, ReLU networks: 0.00009.

4.J.2 Ablation analysis

Table 4.10 shows the results of the additional experiments that use SGD with Nesterov momentum for more values of c and $K = 5$. From this, we see that the recommended value of c from Section 4.4 closely matches the empirical optimum value of c. Note that here we have fixed the learning rate across choices of c. More specifically, the following learning rates were used for the experiments with different datasets. MNIST with fully-connected networks: 0.002; Iris: 0.01; MNIST with convolutional networks: 0.009; CIFAR-10: 0.004; CIFAR-100: 0.002; Fashion MNIST: 0.004. These are the learning rates reported for the SGD with Nesterov momentum experiments in Section 4.J.1.

Table 4.11: Accuracy on the test set for maxout networks with maxout rank $K = 5$ that use batch normalization trained using SGD with Nesterov momentum. Observe that the optimal value of c is close to $c = 0.55555$ which is suggested in Section 4.4.

VALUE OF c	CONVOLUTIONAL			
	MNIST	CIFAR-10	CIFAR-100	Fashion MNIST
10^{-6}	$11.35^{\pm 0.00}$	$10^{\pm 0.00}$	$1^{\pm 0.00}$	$10^{\pm 0.00}$
10^{-5}	$11.35^{\pm 0.00}$	$10^{\pm 0.00}$	$1^{\pm 0.00}$	$10^{\pm 0.00}$
10^{-4}	$99.33^{\pm 0.09}$	$10^{\pm 0.00}$	$1^{\pm 0.00}$	$90.89^{\pm 0.27}$
0.001	$99.35^{\pm 0.05}$	$74.85^{\pm 3.29}$	$38.95^{\pm 4.46}$	$89.78^{\pm 1.2}$
0.01	$99.32^{\pm 0.05}$	$75.72^{\pm 4.94}$	$37.03^{\pm 4.19}$	$90.51^{\pm 0.19}$
0.1	$99.36^{\pm 0.04}$	$77.16^{\pm 1.84}$	$41.64^{\pm 1.53}$	$90.66^{\pm 0.3}$
0.55555	$\mathbf{99.41}^{\pm \mathbf{0.07}}$	$77.68^{\pm 1.07}$	$42.^{\pm 1.51}$	$90.89^{\pm 0.23}$
1	$99.39^{\pm 0.04}$	$\mathbf{79.26}^{\pm \mathbf{0.76}}$	$\mathbf{43.93}^{\pm \mathbf{1.04}}$	$\mathbf{90.93}^{\pm \mathbf{0.36}}$
10	$99.35^{\pm 0.02}$	$75.82^{\pm 1.05}$	$43.17^{\pm 0.28}$	$90.14^{\pm 0.18}$
100	$98.83^{\pm 0.07}$	$66.23^{\pm 1.69}$	$35.67^{\pm 0.88}$	$86.99^{\pm 0.39}$
1000	$97.69^{\pm 0.31}$	$50.97^{\pm 2.28}$	$21.95^{\pm 0.59}$	$80.93^{\pm 0.92}$
10^4	$95.11^{\pm 1.40}$	$43.81^{\pm 1.80}$	$19.87^{\pm 1.29}$	$76.02^{\pm 1.44}$
10^5	$93.09^{\pm 1.88}$	$39.27^{\pm 2.73}$	$14.28^{\pm 2.17}$	$73.71^{\pm 1.55}$
10^6	$87.63^{\pm 1.86}$	$40.27^{\pm 0.92}$	$14.91^{\pm 0.9}$	$71.71^{\pm 3.3}$

4.J.3 Batch normalization

Table 4.11 reports test accuracy for maxout networks with batch normalization trained using SGD with Nesterov momentum for various values of c. The implementation of the experiments is similar to that described in Section 4.J.1, except for the following differences: The networks use batch normalization after each layer with activations; The width of the last fully connected layer is 100, and all other layers of the convolutional networks are 8 times narrower; The learning rate is fixed at 0.01 for all experiments. We use the default batch normalization parameters from TensorFlow. Specifically, the momentum equals 0.99 and $\epsilon = 0.001$. We observe that our initialization strategy is still beneficial when training with batch normalization.

4.J.4 Comparison of maxout and ReLU networks in terms of the number of parameters

We should point out that what is a fair comparison is not as straightforward as matching the parameter count. In particular, wider networks have the advantage of having a higher dimensional representation. A fully connected network will not necessarily perform as well as a convolutional network with the same number of parameters, and a deep and narrow network will not necessarily perform as well as a wider and shallower network with the same number of parameters.

Nevertheless, to add more details to the results, we perform experiments using ReLU networks that have as many parameters as maxout networks. See Tables 4.12 and 4.13 for results. We modify network architectures described in Section 4.J.1 for these experiments in the following way.

In the first experiment, we use fully connected ReLU networks 5 times wider than maxout networks. For convolutional networks, however, the resulting CNNs with ReLU activations would be extremely wide, so we only made it 4 and 3 times wider depending on the depth of the network. In our setup, a 5 times wider CNN network would need to be trained for longer than 24 hours, which is difficult in our experiment environment. Maxout networks only required a much shorter time of around 10 hours, which indicates possible benefits in some cases.

In the second experiment, we consider ReLU networks that are 5 times deeper than maxout networks. More specifically, fully-connected ReLU networks have the following architecture:

$$[25{\times}\text{fc}64,\ 25{\times}\text{fc}32,\ 25{\times}\text{fc}16,\ 25{\times}\text{fc}8,\ \text{out}],$$

convolutional networks used for MNIST and Fashion MNIST datasets have the following layers:

$$[10{\times}\text{conv}64,\ \text{mp},\ 10{\times}\text{conv}128,\ \text{mp},\ 20{\times}\text{conv}256,\ \text{mp},\ 20{\times}\text{conv}512,\ \text{mp}, 10{\times}\text{fc}4096,\ 5{\times}\text{fc}1000,\ \text{out}],$$

and architecture of the convolutional networks used for CIFAR-10 and CIFAR-100 datasets is

$$[10{\times}\text{conv}64,\ \text{mp},\ 10{\times}\text{conv}128,\ \text{mp},\ 20{\times}\text{conv}256,\ \text{mp},\ 20{\times}\text{conv}512,\ \text{mp}, 20{\times}\text{conv}512,\ \text{mp}, 10{\times}\text{fc}4096,\ 5{\times}\text{fc}1000,\ \text{out}].$$

As expected, wider networks do better. On the other hand, deeper ReLU networks of the same width do much worse than maxout networks.

We performed a grid search based on the performance of the model on the validation dataset to determine the optimal learning rate for each ReLU network. Specifically, the following learning rates were used. In the experiment with ReLU networks that are wider than maxout networks, MNIST (fully-connected networks): 0.003, Iris: 0.009, MNIST (convolutional networks): 0.01, Fashion MNIST: 0.008, CIFAR-10: 0.007, CIFAR-100: 0.008. In the experiment with ReLU networks that are deeper than maxout networks, MNIST (fully-connected networks): 0.00002, Iris: 0.0006, MNIST (convolutional networks): 0.00008, Fashion MNIST: 0.0008, CIFAR-10: 0.00008, CIFAR-100: 0.00008.

4.J.5 Comparison of maxout and ReLU networks with dropout

One of the motivations for introducing maxout units in Goodfellow et al. (2013) was to obtain better model averaging by techniques such as dropout (Srivastava et al., 2014). The original paper (Goodfellow et al., 2013) conducted experiments comparing maxout and tanh. In Table 4.14, we show the results of an experiment demonstrating that in terms of allowing for a better approximation

Table 4.12: Accuracy on the test set for networks trained using SGD with Nesterov momentum. Fully-connected ReLU networks are 5 times wider than fully-connected maxout networks. Convolutional ReLU networks are 4 times wider than convolutional maxout networks for the MNIST and Fashion MNIST datasets and 3 times wider for the CIFAR-10 and CIFAR-100 datasets. All networks have the same number of layers.

	MAXOUT Maxout init	RELU He init
VALUE OF c	0.55555	2
	FULLY-CONNECTED	
MNIST	$97.8^{\pm 0.15}$	$\mathbf{98.11^{\pm 0.02}}$
Iris	$91.67^{\pm 3.73}$	$\mathbf{92.5^{\pm 2.76}}$
	CONVOLUTIONAL	
MNIST	$\mathbf{99.59^{\pm 0.04}}$	$99.55^{\pm 0.01}$
Fashion MNIST	$93.49^{\pm 0.13}$	$\mathbf{93.71^{\pm 0.19}}$
CIFAR-10	$91.21^{\pm 0.13}$	$\mathbf{91.24^{\pm 0.21}}$
CIFAR-100	$65.39^{\pm 0.39}$	$\mathbf{66^{\pm 0.45}}$

of model averaging based on dropout, maxout networks compare favorably against ReLU. This indicates that maxout units can indeed be more suitable for training with dropout when properly initialized. We point out that several contemporary architectures often rely on dropout, such as transformers (Vaswani et al., 2017).

4.J.6 Gradient values during training

The goal of the suggested initialization is to ensure that the training can start, while the gradients might vary during training. Nevertheless, it is natural to also consider the gradient values during training for a fuller picture. Hence, we demonstrate the gradients during training in Figures 4.10 and 4.11.

Table 4.13: Accuracy on the test set for networks trained using SGD with Nesterov momentum. ReLU networks are 5 times deeper than maxout networks but have the same width.

	MAXOUT	RELU
	Maxout init	He init
VALUE OF c	0.55555	2
	FULLY-CONNECTED	
MNIST	$\mathbf{97.8}^{\pm\mathbf{0.15}}$	$63.47^{\pm 33.32}$
Iris	$\mathbf{91.67}^{\pm\mathbf{3.73}}$	$75.83^{\pm 11.15}$
	CONVOLUTIONAL	
MNIST	$\mathbf{99.59}^{\pm\mathbf{0.04}}$	$99.4^{\pm 0.05}$
Fashion MNIST	$\mathbf{93.49}^{\pm\mathbf{0.13}}$	$93.25^{\pm 0.11}$
CIFAR-10	$\mathbf{91.21}^{\pm\mathbf{0.13}}$	$73.25^{\pm 3.19}$
CIFAR-100	$\mathbf{65.39}^{\pm\mathbf{0.39}}$	$17.97^{\pm 4.57}$

Table 4.14: Accuracy on the MNIST dataset of fully-connected networks trained with dropout with a rate of 0.5 and of average predictions of several networks in which half of the weights were masked. All results were averaged over 4 runs. Maxout rank $K = 5$. Networks had 3 layers with $128, 64$, and 32 neurons. Maxout networks were initialized using the initialization suggested in Section 4.4, and ReLU networks using He initialization with Gaussian distribution (He et al., 2015). ReLU networks with dropout give results closer to a single model, whereas maxout networks with dropout give results closer to the average of a larger number of models. This indicates that maxout units are more effective for obtaining better model averaging using dropout.

	DROPOUT	1 MODEL	AVERAGE OF 2 MODELS	AVERAGE OF 3 MODELS	AVERAGE OF 4 MODELS
ReLU	$97.04^{\pm 0.14}$	$97.09^{\pm 0.17}$	$97.73^{\pm 0.08}$	$97.87^{\pm 0.04}$	$97.94^{\pm 0.08}$
Maxout	$98.37^{\pm 0.09}$	$97.66^{\pm 0.04}$	$98.03^{\pm 0.05}$	$98.15^{\pm 0.08}$	$98.19^{\pm 0.06}$

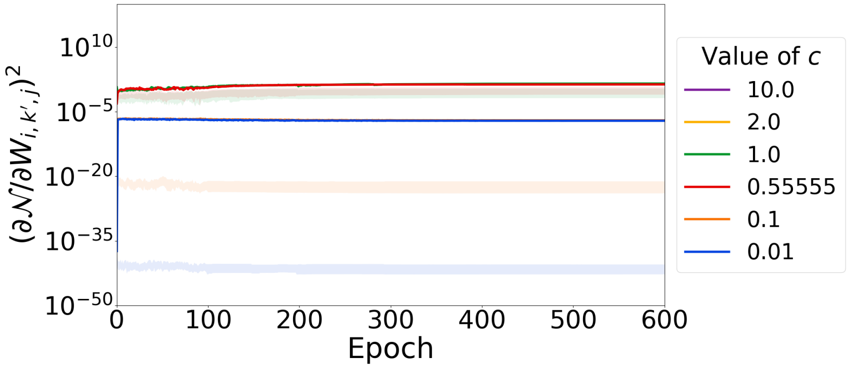


Figure 4.10: Expected value and interquartile range of the squared gradients $(\partial\mathcal{N}/\partial W_{i,k',j})^2$ of a fully-connected maxout network during training on the MNIST dataset. Weights are sampled from $N(0, c/\text{fan-in})$, and biases are initialized to zero. The maxout rank K is 5. We compute the mean and quartiles for 4 training runs using one random input that is fixed at the start of the training. The gradient values increase at the beginning of the training and then remain stable during training for all plotted initializations. Note that red and green lines corresponding to the values of $c = 0.55555$ and $c = 1$, respectively, overlap. Similarly, blue and orange lines corresponding to $c = 0.01$ and $c = 0.1$ overlap. Results for $c = 2$ and $c = 10$ are not shown in the plot since their gradients explode and go to NaN after the training starts.

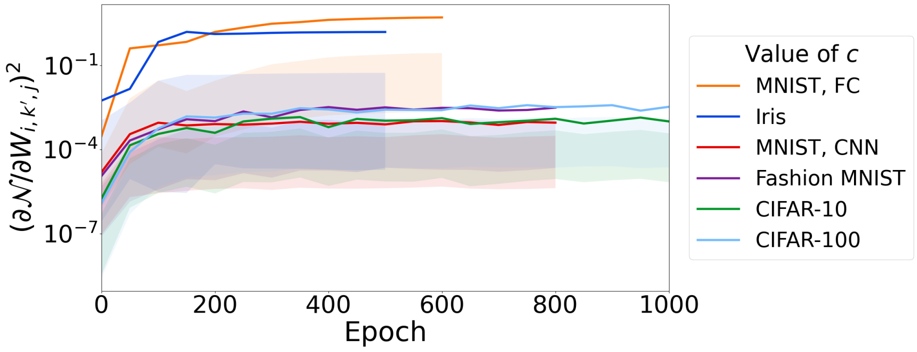


Figure 4.11: Expected value and interquartile range of the squared gradients $(\partial\mathcal{N}/\partial W_{i,k',j})^2$ of maxout networks during training. Weights are sampled from $N(0, c/\text{fan-in})$, where $c = 0.55555$, and biases are initialized to zero. The maxout rank K is 5. We use SGD with momentum and compute the mean and quartiles for 4 training runs using one random input that is fixed at the start of the training. All other experiment parameters are as described in Section 4.J.1. The gradient values increase at the beginning of the training and then remain stable during training for all datasets.

Bibliography

Adiprasito, K. (2017). Lefschetz and lower bound theorems for Minkowski sums. *arXiv preprint arXiv:1711.07218*.

Alfarra, M., Bibi, A., Hammoud, H., Gaafar, M., and Ghanem, B. (2020). On the decision boundaries of deep neural networks: A tropical geometry perspective. *arXiv preprint arXiv:2002.08838*.

Anderson, T. W. (2003). *An introduction to multivariate statistical analysis*. Wiley series in probability and statistics. Wiley-Interscience, 3rd ed edition.

Anstreicher, K. M. (1999). Linear programming in $O([n^3/\ln n]L)$ operations. *SIAM Journal on Optimization*, 9(4):803–812.

Anton, H. and Rorres, C. (2013). *Elementary linear algebra: applications version*. John Wiley & Sons.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R. R., and Wang, R. (2019). On exact computation with an infinitely wide neural net. *Advances in neural information processing systems*, 32.

Bachlechner, T., Majumder, B. P., Mao, H., Cottrell, G., and McAuley, J. (2021). Rezero is all you need: Fast convergence at large depth. In *Uncertainty in Artificial Intelligence*, pages 1352–1361. PMLR.

Baldi, P. and Vershynin, R. (2019). The capacity of feedforward neural networks. *Neural networks*, 116:288–311.

Balestriero, R., Cosentino, R., Aazhang, B., and Baraniuk, R. (2019). The geometry of deep networks: Power diagram subdivision. In *Advances in Neural Information Processing Systems*, pages 15832–15841.

Bianchini, M. and Scarselli, F. (2014). On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE transactions on neural networks and learning systems*, 25(8):1553–1565.

Bodnar, C., Frasca, F., Wang, Y. G., Otter, N., Montúfar, G., Liò, P., and Bronstein, M. (2021). Weisfeiler and Lehman go topological: Message passing simplicial networks. *arXiv preprint arXiv:2103.03212*.

Bowman, B. and Montufar, G. F. (2022). Spectral bias outside the training set for deep networks in the kernel regime. *Advances in Neural Information Processing Systems*, 35:30362–30377.

Buck, R. C. (1943). Partition of space. *The American Mathematical Monthly*, 50(9):541–544.

Bárány, I. and Vu, V. (2007). Central limit theorems for Gaussian polytopes. *The Annals of Probability*, 35(4):1593 – 1621.

Caron, R. and Traynor, T. (2005). The zero set of a polynomial. *WSMR Report*, pages 05–02.

Castaneda, G., Morris, P., and Khoshgoftaar, T. M. (2019). Evaluation of maxout activations in deep learning across several big data domains. *Journal of Big Data*, 6(1):72.

Chandru, V. and Hooker, J. (2011). *Optimization methods for logical inference*, volume 34. John Wiley & Sons.

Charisopoulos, V. and Maragos, P. (2018). A tropical approach to neural networks with piecewise linear activations. *arXiv preprint arXiv:1805.08749*.

Chizat, L., Oyallon, E., and Bach, F. (2019). On lazy training in differentiable programming. *Advances in neural information processing systems*, 32.

Croce, F., Andriushenko, M., and Hein, M. (2019). Provable robustness of ReLU networks via maximization of linear regions. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2057–2066. PMLR.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

Dalcin, L. D., Paz, R. R., Kler, P. A., and Cosimo, A. (2011). Parallel distributed computing using python. *Advances in Water Resources*, 34(9):1124–1139.

Delalleau, O. and Bengio, Y. (2011). Shallow vs. deep sum-product networks. *Advances in neural information processing systems*, 24.

Dukler, Y., Gu, Q., and Montúfar, G. (2020). Optimization theory for relu neural networks trained with normalization layers. In *International conference on machine learning*, pages 2751–2760. PMLR.

Ebrahimi, J., Gelda, D., and Zhang, W. (2020). How can self-attention networks recognize dyck-n languages? *arXiv preprint arXiv:2010.04303*.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.

Funahashi, K.-I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. In *International Conference on Machine Learning*, pages 1319–1327. PMLR.

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. (2015). An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv:1312.6211 [cs, stat]*.

Google Scholar (2023). Results for maxout networks search on Google Scholar. `https://scholar.google.com/scholar?as_ylo=2022&q=maxout+networks&hl=en&as_sdt=0,5`. [Accessed 11-Jun-2023].

Gover, E. and Krikorian, N. (2010). Determinants and the volumes of parallelotopes and zonotopes. *Linear Algebra and its Applications*, 433(1):28–40.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Gühring, I., Raslan, M., and Kutyniok, G. (2020). Expressivity of deep neural networks. *arXiv preprint arXiv:2007.04759*.

Gurbuzbalaban, M. and Hu, Y. (2021). Fractional moment-preserving initialization schemes for training deep neural networks. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pages 2233–2241. PMLR.

Hahn, M. (2020). Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171.

Hamilton, W. L. (2020). Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159.

Hanin, B. (2018). Which neural net architectures give rise to exploding and vanishing gradients? *Advances in Neural Information Processing Systems*, 31.

Hanin, B., Jeong, R., and Rolnick, D. (2021). Deep ReLU networks preserve expected length. *arXiv:2102.10492*.

Hanin, B. and Nica, M. (2020a). Finite depth and width corrections to the neural tangent kernel. In *International Conference on Learning Representations*.

Hanin, B. and Nica, M. (2020b). Products of many large random matrices and gradients in deep neural networks. *Communications in Mathematical Physics*, 376(1):287–322.

Hanin, B. and Rolnick, D. (2018). How to start training: The effect of initialization and architecture. *Advances in Neural Information Processing Systems*, 31.

Hanin, B. and Rolnick, D. (2019a). Complexity of linear regions in deep networks. In *International Conference on Machine Learning*, pages 2596–2604. PMLR.

Hanin, B. and Rolnick, D. (2019b). Deep ReLU networks have surprisingly few activation patterns. *Advances in Neural Information Processing Systems*, 32.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, Santiago, Chile. IEEE.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778. IEEE.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hinz, P. (2021). Using activation histograms to bound the number of affine regions in ReLU feed-forward neural networks. *arXiv preprint arXiv:2103.17174*.

Hinz, P. and Van de Geer, S. (2019). A framework for the construction of upper bounds on the number of affine linear regions of ReLU feed-forward neural networks. *IEEE Transactions on Information Theory*, 65(11):7304–7324.

Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1).

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Hug, D., Munsonius, G. O., and Reitzner, M. (2004). Asymptotic mean values of Gaussian polytopes. *Beiträge zur Algebra und Geometrie*, 45(2):531–548.

Hug, D. and Reitzner, M. (2005). Gaussian polytopes: Variances and limit theorems. *Advances in Applied Probability*, 37(2):297–320.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456. PMLR.

Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. *Advances in Neural Information Processing Systems*, 31.

Jameson, G. (2013). Inequalities for gamma function ratios. *The American Mathematical Monthly*, 120(10):936–940.

Jin, H. and Montúfar, G. (2023). Implicit bias of gradient descent for mean squared error regression with two-layer wide neural networks. *Journal of Machine Learning Research*, 24(137):1–97.

Jones, E., Oliphant, T., Peterson, P., et al. (2001). SciPy: Open source scientific tools for Python.

Joswig, M. (2022). *Essentials of Tropical Combinatorics*. Graduate Studies in Mathematics. AMS.

Karhadkar, K., Murray, M., Tseran, H., and Montúfar, G. (2023). Mildly overparameterized relu networks have a favorable loss landscape. *arXiv preprint arXiv:2305.19510*.

Kileel, J., Trager, M., and Bruna, J. (2019). On the expressive power of deep polynomial neural networks. *Advances in neural information processing systems*, 32.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Krizhevsky, A., Hinton, G., and others (2009). Learning multiple layers of features from tiny images. *Citeseer*.

LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). Efficient BackProp. In *Neural Networks: Tricks of the Trade: Second Edition*, pages 9–48. Springer Berlin Heidelberg.

Lee, G.-H., Alvarez-Melis, D., and Jaakkola, T. S. (2019a). Towards robust, locally linear deep networks. *arXiv preprint arXiv:1907.03207*.

Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. (2019b). Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32.

Maclagan, D. and Sturmfels, B. (2015). *Introduction to tropical geometry*, volume 161. Providence, RI: American Mathematical Society (AMS).

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng (2015). TensorFlow: Large-scale machine learning on heterogeneous systems.

McMullen, P. (1970). The maximum numbers of faces of a convex polytope. *Mathematika*, 17(2):179–184.

Montúfar, G. (2017). Notes on the number of linear regions of deep neural networks.

Montúfar, G., Ren, Y., and Zhang, L. (2022). Sharp bounds for the number of regions of maxout networks and vertices of minkowski sums. *SIAM J. Appl. Algebra Geom.*, 6(1):618–649.

Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. (2014). On the number of linear regions of deep neural networks. *Advances in Neural Information Processing Systems*, 27.

Murray, M., Abrol, V., and Tanner, J. (2022). Activation function design for deep networks: linearity and effective initialisation. *Applied and Computational Harmonic Analysis*, 59:117–154.

Müller, A. and Stoyan, D. (2002). *Comparison methods for stochastic models and risks*, volume 389. Wiley.

Nadarajah, S. (2008). Explicit expressions for moments of $\chi 2$ order statistics. *Bull. Inst. Math. Acad. Sin.(NS)*, 3:433–444.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning.

Nguyen, Q., Mondelli, M., and Montúfar, G. (2020). Tight bounds on the smallest eigenvalue of the neural tangent kernel for deep ReLU networks. *arXiv preprint arXiv:2012.11654*.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318. PMLR.

Pascanu, R., Montúfar, G., and Bengio, Y. (2014). On the number of response regions of deep feed forward networks with piece-wise linear activations. In *International Conference on Learning Representations 2014 (ICLR 2014), Banff, Alberta, Canada*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pennington, J., Schoenholz, S., and Ganguli, S. (2017). Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Pennington, J., Schoenholz, S., and Ganguli, S. (2018). The emergence of spectral universality in deep networks. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, pages 1924–1932. PMLR.

Phuong, M. and Lampert, C. H. (2019). Functional vs. parametric equivalence of ReLU networks. In *International Conference on Learning Representations*.

Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., and Ganguli, S. (2016). Exponential expressivity in deep neural networks through transient chaos. *Advances in Neural Information Processing Systems*, 29.

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Sohl-Dickstein, J. (2017). On the expressive power of deep neural networks. In *International Conference on Machine Learning*, pages 2847–2854. PMLR.

Safran, I. and Shamir, O. (2018). Spurious local minima are common in two-layer ReLU neural networks. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4433–4441. PMLR.

Safran, I., Vardi, G., and Lee, J. D. (2022). On the effective number of linear regions in shallow univariate relu networks: Convergence guarantees and implicit bias. *Advances in Neural Information Processing Systems*, 35:32667–32679.

Saxe, A. M., McClelland, J. L., and Ganguli, S. (2014). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations*.

Serra, T., Kumar, A., and Ramalingam, S. (2020). Lossless compression of deep neural networks. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 417–430. Springer.

Serra, T. and Ramalingam, S. (2020). Empirical bounds on linear regions of deep rectifier networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5628–5635.

Serra, T., Tjandraatmadja, C., and Ramalingam, S. (2018). Bounding and counting linear regions of deep neural networks. In *International Conference on Machine Learning*, pages 4558–4566. PMLR.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

Soudry, D. and Carmon, Y. (2016). No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv preprint arXiv:1605.08361*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Steiner, J. (1826). Einige Gesetze über die Theilung der Ebene und des Raumes. *Journal für die reine und angewandte Mathematik*, 1:349–364.

Steinwart, I. (2019). A Sober Look at Neural Network Initializations. *arXiv:1903.11482 [cs, stat]*.

Strogatz, S. H. (2018). *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press.

Sun, R. (2019). Optimization for deep learning: theory and algorithms. *arXiv:1912.08957 [cs, math, stat]*.

Sun, W., Su, F., and Wang, L. (2018). Improving deep neural networks with multi-layer maxout networks and a novel initialization method. *Neurocomputing*, 278:34–40.

Telgarsky, M. (2015). Representation benefits of deep feedforward networks. *arXiv preprint arXiv:1509.08101*.

Telgarsky, M. (2016). Benefits of depth in neural networks. In *Conference on learning theory*, pages 1517–1539. PMLR.

Telgarsky, M. (2021). Deep learning theory lecture notes. `https://mjt.cs.illinois.edu/dlt/`. Version: 2021-10-27 v0.0-e7150f2d (alpha).

Tseran, H. and Montúfar, G. (2023). Expected gradients of maxout networks and consequences to parameter initialization. *International Conference on Machine Learning*.

Tseran, H. and Montúfar, G. F. (2021). On the expected complexity of maxout networks. *Advances in Neural Information Processing Systems*, 34.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Vershynin, R. (2018). *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press.

Weibel, C. (2012). Maximal f-vectors of Minkowski sums of large numbers of polytopes. *Discrete Computational Geometry*, 47:519–537.

Williams, F., Trager, M., Panozzo, D., Silva, C., Zorin, D., and Bruna, J. (2019). Gradient dynamics of shallow univariate ReLU networks. *Advances in Neural Information Processing Systems*, 32.

Wolfram Research, Inc (2022). Mathematica, Version 13.1.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.

Xie, Y., Chen, G., and Li, Q. (2020). A general computational framework to measure the expressiveness of complex networks using a tighter upper bound of linear regions. *arXiv preprint arXiv:2012.04428*.

Xiong, H., Huang, L., Yu, M., Liu, L., Zhu, F., and Shao, L. (2020). On the number of linear regions of convolutional neural networks. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10514–10523. PMLR.

Zaslavsky, T. (1975). *Facing up to Arrangements: Face-Count Formulas for Partitions of Space by Hyperplanes*. American Mathematical Society: Memoirs of the American Mathematical Society. American Mathematical Society.

Zhang, H., Dauphin, Y. N., and Ma, T. (2019). Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*.

Zhang, L., Naitzat, G., and Lim, L.-H. (2018). Tropical geometry of deep neural networks. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5824–5832, Stockholmsmässan, Stockholm Sweden. PMLR.

Zhang, X. and Wu, D. (2020). Empirical studies on the properties of linear regions in deep neural networks. In *International Conference on Learning Representations*.

Bibliographische Daten

Expected Complexity and Gradients of Deep Maxout Neural Networks and Implications to Parameter Initialization
(Über den Erwartungswert der Funktionenkomplexität und Gradienten in Tiefen Max-Out Neuronalen Netzen)
Tseran, Hanna
Universität Leipzig, Dissertation, 2023
168 Seiten, 33 Abbildungen, 119 Referenzen

Selbstständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialien oder erbrachten Dienstleistungen als solche gekennzeichnet.

Leipzig, den November 9, 2023

. . . .
(Hanna Tseran)

Daten zum Autor

Name:	Hanna Tseran
Geburtsdatum:	15.09.1992 in Minsk (Belarus)
09/2010 - 06/2015	Diplom in Informatik Belarussische Staatliche Universität
09/2016 - 09/2018	Master in Informationswissenschaft und Technologie Universität Tokio
seit 01/2020	Doktorandin der Informatik