THE LOSS SURFACE AND EXPRESSIVITY OF DEEP CONVOLUTIONAL NEURAL NETWORKS

Quynh Nguyen & Matthias Hein
Department of Mathematics and Computer Science
Saarland Informatics Campus, Saarland University, Germany

ABSTRACT

We analyze the expressiveness and loss surface of practical deep convolutional neural networks (CNNs) with shared weights. We show that such CNNs produce linearly independent features (and thus linearly separable) at every "wide" layer which has more neurons than the number of training samples. This condition holds e.g. for the VGG network. Furthermore, we provide for such wide CNNs necessary and sufficient conditions for global minima with zero training error. For the case where the wide layer is followed by a fully connected layer we show that almost every critical point of the empirical loss is a global minimum with zero training error. Our analysis suggests that both depth and width are equally important in deep learning. While depth brings more representational power and allows the network to learn high level features, width smoothes the optimization landscape of the loss function in the sense that a sufficiently wide CNN has a well-behaved loss surface with almost no bad local minima.

1 SETTING

Let N be the number of training samples, and $X = [x_1, \ldots, x_N]^T \in \mathbb{R}^{N \times d}, Y = [y_1, \ldots, y_N]^T \in \mathbb{R}^{N \times m}$ the input resp. output matrix for the training data, where d is the input dimension and m the number of classes. Let L be the number of layers of the network, where each layer is either a convolutional or fully connected layer. The layers are indexed from $k = 0, 1, \ldots, L$ which corresponds to input layer, 1st hidden layer, ..., and output layer. Let n_k be the width of layer k and $f_k : \mathbb{R}^d \to \mathbb{R}^{n_k}$ the function that computes for every input its feature vector at layer k. The convolutional layer consists of a set of patches of equal length where every patch is a subset of neurons from the same layer. Let P_k and l_k be the number and size of patches at layer k. For every input $x \in \mathbb{R}^d$, let $\{f_k^1(x), \ldots, f_k^{P_k}(x)\} \in \mathbb{R}^{l_k}$ be the set of patches at layer k. For consistency, let $f_k(x) = x$ for $k = 0$, which denotes the input. Each of the T_k convolutional filters of layer k will be applied to the same set of patches at layer $k-1$. We denote by $W_k = [w_k^1, \ldots, w_k^{T_k}] \in \mathbb{R}^{l_{k-1} \times T_k}$ the corresponding parameter matrix of the convolutional layer k. Each column of W_k corresponds to one filter. Furthermore, $b_k \in \mathbb{R}^{n_k}$ is the bias vector and $\sigma_k : \mathbb{R} \to \mathbb{R}$ the activation function for the k-th layer. All functions are applied componentwise, and $[a]$ denotes the set of integers $\{1, 2, \ldots, a\}$. In this paper, CNN architectures consist of standard convolutional layers and fully connected layers. As a common practice, we assume that the output layer is always fully connected.

Definition 1.1 *A layer k in a deep CNN architecture is called*

- *convolutional layer if its output $f_k(x) \in \mathbb{R}^{n_k}$ is defined for every $x \in \mathbb{R}^d$ as*

$$f_k(x)_h = \sigma_k\Big(\big\langle w_k^t, f_{k-1}^p(x) \big\rangle + (b_k)_h \Big) \tag{1}$$

  for every $p \in [P_{k-1}], t \in [T_k], h := (p-1)T_k + t$.

- *fully connected layer if its output $f_k(x) \in \mathbb{R}^{n_k}$ is defined for every $x \in \mathbb{R}^d$ as*

$$f_k(x) = \sigma_k\Big(W_k^T f_{k-1}(x) + b_k \Big). \tag{2}$$

Specifically, the value of each neuron indexed by h at a convolutional layer k is computed by first taking the inner product between a filter of layer k and a patch at layer $k-1$, adding the bias and then applying the activation function. The width of layer k is thus $n_k = T_k P_{k-1}$. For each convolutional layer, we denote by $\mathcal{M}_k : \mathbb{R}^{l_{k-1}\times T_k} \to \mathbb{R}^{n_{k-1}\times n_k}$ the linear map that returns for every parameter matrix $W_k \in \mathbb{R}^{l_{k-1}\times T_k}$ the corresponding full weight matrix $U_k = \mathcal{M}_k(W_k) \in \mathbb{R}^{n_{k-1}\times n_k}$. We define $U_k = \mathcal{M}_k(W_k) = W_k$ if layer k is fully connected. For example, suppose that layer k has two filters of length 3, that is, $W_k = [w_k^1, w_k^2] = \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}$, and $n_{k-1} = 5$ and patches given by a 1D-convolution with stride 1 and no padding then: $U_k^T = \mathcal{M}_k(W_k)^T = \begin{bmatrix} a & b & c & 0 & 0 \\ d & e & f & 0 & 0 \\ 0 & a & b & c & 0 \\ 0 & d & e & f & 0 \\ 0 & 0 & a & b & c \\ 0 & 0 & d & e & f \end{bmatrix}$. Suppose that there is no non-linearity at the output layer, the feature maps $f_k : \mathbb{R}^d \to \mathbb{R}^{n_k}$ can be written as

$$f_k(x) = \sigma_k\Big(U_k^T f_{k-1}(x) + b_k\Big) \;\; \forall 1 \le k \le L-1, \quad f_L(x) = U_L^T f_{L-1}(x) + b_L.$$

By stacking the feature vectors of layer k for all the training samples into a matrix, we define $F_k = [f_k(x_1), \ldots, f_k(x_N)]^T \in \mathbb{R}^{N\times n_k}$ and refer to F_k as the output matrix at layer k. We generally assume in the following that for every convolutional layer k there exists one parameter matrix $W_k \in \mathbb{R}^{l_{k-1}\times T_k}$ for which $U_k = \mathcal{M}_k(W_k) \in \mathbb{R}^{n_{k-1}\times n_k}$ has full rank. One can see that this is satisfied in practice if every neuron of a convolutional layer belongs to at least one patch and there are no identical patches. Out of space reasons, we present in the following the list of all assumptions needed to derive our main results in this paper even though not all of them are necessary in each theorem.

Assumption 1.2 1. *The patches of different training samples are non-identical, that is, $x_i^p \neq x_j^q$ for all $i \neq j$ and $p, q \in [P_0]$.*

2. *All the activation functions $(\sigma_1, \ldots, \sigma_{L-1})$ are one of the following functions:*
   - *Sigmoid: $\sigma(t) = \big(1 + e^{-t}\big)^{-1}$*
   - *Softplus: $\sigma_\alpha(t) = \frac{1}{\alpha}\ln(1 + e^{\alpha t})$ for $\alpha > 0$*
3. *There exists a hidden layer $1 \le k \le L-1$ such that the width of layer k is larger than number of training samples, that is, $n_k = T_k P_{k-1} \ge N$ and the network has pyramidal structure from layer $k+1$ till the output layer, that is, $n_{k+1} \ge \ldots \ge n_L$.*

The first condition of Assumption 1.2 is very weak and even if it does not hold then there exists an arbitrarily small perturbation of the data such that it holds for the perturbed training set. Moreover, note that ReLU can be approximated arbitrarily well using softplus: $\lim\limits_{\alpha\to\infty} \frac{1}{\alpha}\ln(1 + e^{\alpha t}) = \max(0, t)$.

2 Main results

Do CNNs Learn Linearly Independent Features? We first show that CNNs with a wide layer can easily learn linearly independent features even when all the weights are randomly generated under certain distribution. Note that linear independence implies linear separability.

Theorem 2.1 *Let a deep CNN satisfy Assumption 1.2 for some hidden layer $1 \le k \le L-1$. Then the set of parameters of the first k layers $(W_l, b_l)_{l=1}^k$ for which the set of feature vectors $\{f_k(x_1), \ldots, f_k(x_N)\}$ of layer k are **not** linearly independent has Lebesgue measure **zero**.*

A crucial step of the proof of Theorem 2.1 shows the existence of network parameters such that F_k has full rank, and then uses properties of analytic activation functions to derive that the set of parameters where F_k has not full rank has Lebesgue measure zero. The above result can be used to show that a wide CNN can fit exactly any training set if $n_k \ge N$. This condition is fulfilled for the VGG or Inception-v3/4 networks. Theorem 2.1 explains previous empirical observations, e.g. Czarnecki et al. (2017) have shown empirically that linear separability is often obtained in the first few hidden layers of neural networks. Furthermore, Theorem 2.1 is in line with recent empirical observations for CNNs that one has little loss in performance if the weights of the initial layers are chosen randomly without training (Jarrett et al., 2009; Saxe et al., 2011; Yosinski et al., 2014).

Loss Surface of CNNs: We study least squares loss, but our results can be extended to all loss functions where the global minimum is attained for $F_L = Y$. Let $\mathcal{P}$ denote the space of all parameters of the network. The final training objective $\Phi : \mathcal{P} \to \mathbb{R}$ is

$$\Phi\Big((W_l, b_l)_{l=1}^L\Big) = \frac{1}{2} \left\| F_L - Y \right\|_F^2 , \tag{3}$$

where F_L is the output matrix of the network defined in previous section.

In the following, we examine conditions for the global optimality of critical points of Φ inside a subset $S_k \subseteq \mathcal{P}$, defined for every $1 \leq k \leq L-1$ as

$$S_k := \Big\{ (W_l, b_l)_{l=1}^L \;\Big|\; \textit{rank}(F_k) = N \text{ and } U_l \text{ has full rank for every } l \in [k+2, L] \Big\} .$$

Essentially, S_k is the set of parameters where the feature vectors at layer k are linearly independent and all the weight matrices from layer $k+2$ till the output layer have full rank. It is thus important to note that S_k can cover already almost the whole parameter space under Assumption 1.2.

Lemma 2.2 *Let a deep CNN architecture satisfy Assumption 1.2 for some hidden layer* $1 \leq k \leq L-1$. *Then the set* $\mathcal{P} \setminus S_k$ *has Lebesgue measure* ***zero****.*

Our next result is motivated by the fact that empirically when training over-parameterized neural networks with shared weights and sparsity structure like CNNs, there seem to be no problems with sub-optimal local minima. In many cases, even when training labels are completely random, local search algorithms like stochastic gradient descent can converge to a solution with almost zero training error (Zhang et al., 2017). To achieve a better understanding on this phenomenon, we first characterize in the following Theorem 2.3 the set of points in parameter space with zero training loss, and then analyze in Theorem 2.4 the loss surface for a special case of the network. We emphasize that our results hold for standard deep CNNs with convolutional layers with shared weights and fully connected layers. In the following, Φ can also be seen as a function of $(U_l, b_l)_{l=1}^L$ since U_k is a function of the true optimization variables W_k by definition. One has the relation $\frac{\partial \Phi}{\partial (W_k)_{rs}} = \sum_{i,j} \frac{\partial \Phi}{\partial (U_k)_{ij}} \frac{\partial (U_k)_{ij}}{\partial (W_k)_{rs}}$.

Theorem 2.3 (Necessary and Sufficient Condition for Zero Training Error) *Let a deep CNN architecture satisfies Assumption 1.2 for some hidden layer* $1 \leq k \leq L-1$. *Let the training objective* $\Phi : \mathcal{P} \to \mathbb{R}$ *be defined as in* (3)*. Given* ***any*** *point* $(W_l, b_l)_{l=1}^L \in S_k$. *Then it holds that* $\Phi\Big((W_l, b_l)_{l=1}^L\Big) = 0$ *if and only if* $\frac{\partial \Phi\big((W_l, b_l)_{l=1}^L\big)}{\partial U_{k+1}} = 0$.

Note that Lemma 2.2 shows that the set of points which are not covered by Theorem 2.3 has just measure zero. The necessary and sufficient condition of Theorem 2.3 is rather intuitive as it requires the gradient of the training objective to vanish w.r.t. the full weight matrix of layer $k+1$ regardless of the architecture of this layer. It turns out that if layer $k+1$ is fully connected, then this condition is always satisfied at a critical point. Thus every critical point in S_k is a global minimum with zero training error. This is shown next, where we consider a classification task with m classes. Let $Z \in \mathbb{R}^{m \times m}$ be a full rank class encoding matrix, e.g. the identity matrix, such that $Y_{i:} = Z_{j:}$ whenever the training sample x_i belongs to class j for every $i \in [N], j \in [m]$.

Theorem 2.4 (Loss Surface of CNNs) *Let* (X, Y, Z) *be a training dataset where* Z *has full rank. Let a deep CNN architecture satisfy Assumption 1.2 for some hidden layer* $1 \leq k \leq L-1$*, and layer* $k+1$ *is fully connected. Then the following hold*

- *Every critical point* $(W_l, b_l)_{l=1}^L \in S_k$ *is a global minimum with* $\Phi\Big((W_l, b_l)_{l=1}^L\Big) = 0$
- *There exist infinitely many global minima* $(W_l, b_l)_{l=1}^L \in S_k$ *with* $\Phi\Big((W_l, b_l)_{l=1}^L\Big) = 0$

An interesting special case of Theorem 2.4 is when the network is fully connected, in which case all the results of Theorem 2.4 hold without any modifications. This can be seen as a formal proof for the implicit assumption used in the recent work (Nguyen & Hein, 2017) that there exists a global minimum with zero training error for the class of fully connected, deep and wide networks.

REFERENCES

G. Alain and Y. Bengio. Understanding intermediate layers using linear classifier probes. In *ICLR Workshop*, 2016.

S. An, F. Boussaid, and M. Bennamoun. How can deep rectifier networks achieve linear separability and preserve distances? In *ICML*, 2015.

A. Andoni, R. Panigrahy, G. Valiant, and L. Zhang. Learning polynomials with neural networks. In *ICML*, 2014.

P. Auer, M. Herbster, and M. K. Warmuth. Exponentially many local minima for single neurons. In *NIPS*, 1996.

P. Baldi and K. Hornik. Neural networks and principle component analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58, 1988.

A. Blum and R. L Rivest. Training a 3-node neural network is np-complete. In *NIPS*, 1989.

A. Brutzkus and A. Globerson. Globally optimal gradient descent for a convnet with gaussian inputs, 2017. arXiv:1702.07966.

F. Chollet. Xception: Deep learning with depthwise separable convolutions, 2016. arXiv:1610.02357.

A. Choromanska, M. Hena, M. Mathieu, G. B. Arous, and Y. LeCun. The loss surfaces of multilayer networks. In *AISTATS*, 2015.

N. Cohen and A. Shashua. Convolutional rectifier networks as generalized tensor decompositions. In *ICML*, 2016.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

W. M. Czarnecki, G. Swirszcz, M. Jaderberg, S. Osindero, O. Vinyals, and K. Kavukcuoglu. Understanding synthetic gradients and decoupled neural interfaces. In *ICML*, 2017.

Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *NIPS*, 2014.

O. Delalleau and Y. Bengio. Shallow vs. deep sum-product networks. In *NIPS*, 2011.

S. S. Du, J. D. Lee, and Y. Tian. When is a convolutional filter easy to learn?, 2017. arXiv:1709.06129.

R. Eldan and O. Shamir. The power of depth for feedforward neural networks. In *COLT*, 2016.

C. D. Freeman and J. Bruna. Topology and geometry of half-rectified network optimization. In *ICLR*, 2017.

A. Gautier, Q. Nguyen, and M. Hein. Globally optimal training of generalized polynomial neural networks with nonlinear spectral methods. In *NIPS*, 2016.

S. Goel and A. Klivans. Learning depth-three neural networks in polynomial time, 2017. arXiv:1709.06010.

I. J. Goodfellow, O. Vinyals, and A. M. Saxe. Qualitatively characterizing neural network optimization problems. In *ICLR*, 2015.

B. D. Haeffele and R. Vidal. Global optimality in tensor factorization, deep learning, and beyond, 2015. arXiv:1506.07540v1.

M. Hardt and T. Ma. Identity matters in deep learning. In *ICLR*, 2017.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and < 0.5mb model size, 2016. arXiv:1602.07360.

M. Janzamin, H. Sedghi, and A. Anandkumar. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. *arXiv:1506.08473*, 2016.

K. Jarrett, K. Kavukcuoglu, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *CVPR*, 2009.

K. Kawaguchi. Deep learning without poor local minima. In *NIPS*, 2016.

S. G. Krantz and H. R. Parks. *A Primer of Real Analytic Functions*. Birkhäuser, Boston, second edition, 2002.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Handwritten digit recognition with a back-propagation network. In *NIPS*, 1990.

Y. Li and Y. Yuan. Convergence analysis of two-layer neural networks with relu activation, 2017. arXiv:1705.09886.

S. Liang and R. Srikant. Why deep neural networks for function approximation? In *ICLR*, 2017.

R. Livni, S. Shalev-Shwartz, and O. Shamir. On the computational efficiency of training neural networks. In *NIPS*, 2014.

A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, 2015.

H. Mhaskar and T. Poggio. Deep vs. shallow networks : An approximation theory perspective, 2016. arXiv:1608.03287.

B. Mityagin. The zero set of a real analytic function, 2015. arXiv:1512.07276.

G. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *NIPS*, 2014.

Q. Nguyen and M. Hein. The loss surface of deep and wide neural networks. In *ICML*, 2017.

V. D. Nguyen. Complex powers of analytic functions and meromorphic renormalization in qft, 2015. arXiv:1503.00995.

R. Pascanu, G. Montufar, and Y. Bengio. On the number of response regions of deep feedforward networks with piecewise linear activations. In *ICLR*, 2014.

A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation, 2016. arXiv:1606.02147.

T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao. Why and when can deep – but not shallow – networks avoid the curse of dimensionality: a review, 2016. arXiv:1611.00740.

M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein. On the expressive power of deep neural networks. In *ICML*, 2017.

I. Safran and O. Shamir. On the quality of the initial basin in overspecified networks. In *ICML*, 2016.

I. Safran and O. Shamir. Depth-width tradeoffs in approximating natural functions with neural networks. In *ICML*, 2017.

A. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Y. Ng. On random weights and unsupervised feature learning. In *ICML*, 2011.

H. Sedghi and A. Anandkumar. Provable methods for training neural networks with sparse connectivity. In *ICLR Workshop*, 2015.

S. Shalev-Shwartz, O. Shamir, and S. Shammah. Failures of gradient-based deep learning. In *ICML*, 2017.

O. Shamir. Distribution-specific hardness of learning neural networks, 2017. arXiv:1609.01037.

J. Sima. Training a single sigmoidal neuron is hard. *Neural Computation*, 14:2709–2728, 2002.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

M. Soltanolkotabi. Learning relus via gradient descent, 2017. arXiv:1705.04591.

D. Soudry and E. Hoffer. Exponentially vanishing sub-optimal local minima in multilayer neural networks, 2017. arXiv:1702.05777.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015a.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision, 2015b. arXiv:1512.00567.

C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning, 2016. arXiv:1602.07261.

M. Telgarsky. Representation benefits of deep feedforward networks, 2015. arXiv:1509.08101v2.

M. Telgarsky. Benefits of depth in neural networks. In *COLT*, 2016.

Y. Tian. An analytical formula of population gradient for two-layered relu network and its applications in convergence and critical point analysis. In *ICML*, 2017.

D. Yarotsky. Error bounds for approximations with deep relu networks, 2016. arXiv:1610.01145.

J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.

J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. In *ICML*, 2015.

C. Yun, S. Sra, and A. Jadbabaie. Global optimality conditions for deep neural networks, 2017. arXiv:1707.02444.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and Oriol Vinyals. Understanding deep learning requires re-thinking generalization. In *ICLR*, 2017.

K. Zhong, Z. Song, P. Jain, P. Bartlett, and I. Dhillon. Recovery guarantees for one-hidden-layer neural networks. In *ICML*, 2017.