On the Loss Landscape of a Class of Deep Neural Networks with No Bad Local Valleys

On the loss landscape of a class of deep neural networks with no bad local valleys

Quynh Nguyen
Saarland University, Germany

Mahesh Chandra Mukkamala
Saarland University, Germany

Matthias Hein
University of Tübingen, Germany

Abstract

We identify a class of over-parameterized deep neural networks with standard activation functions and cross-entropy loss which provably have no bad local valley, in the sense that from any point in parameter space there exists a continuous path on which the cross-entropy loss is non-increasing and gets arbitrarily close to zero. This implies that these networks have no sub-optimal strict local minima.

1 Introduction

It has been empirically observed in deep learning (Dauphin et al., 2014; Goodfellow et al., 2015) that the training problem of over-parameterized[1] deep CNNs (LeCun et al., 1990; Krizhevsky et al., 2012) does not seem to have a problem with bad local minima. In many cases, local search algorithms like stochastic gradient descent (SGD) frequently converge to a solution with zero training error even though the training objective is known to be non-convex and potentially has many local minima (Auer et al., 1996; Safran & Shamir, 2018), even for simple models like deep linear networks (Kawaguchi, 2016). This indicates that the problem of training practical over-parameterized neural networks is still far from the worst-case scenario where the problem is known to be NP-hard (Blum & Rivest., 1989; Sima, 2002; Livni et al., 2014; Shalev-Shwartz et al., 2017). A possible hypothesis is that the loss landscape of these networks is"well-behaved" so that it becomes amenable to local search algorithms like SGD and its variants. As not all neural networks have a well-behaved loss landscape, it is interesting to identify sufficient conditions on their architecture so that this is guaranteed. In this paper our motivation is to come up with such a class of networks in a practically relevant setting, that is we study multi-class problems with the usual empirical cross-entropy loss and deep (convolutional) networks and almost no assumptions on the training data, in particular no distributional assumptions. Thus our results directly apply to the networks which we use in the experiments.

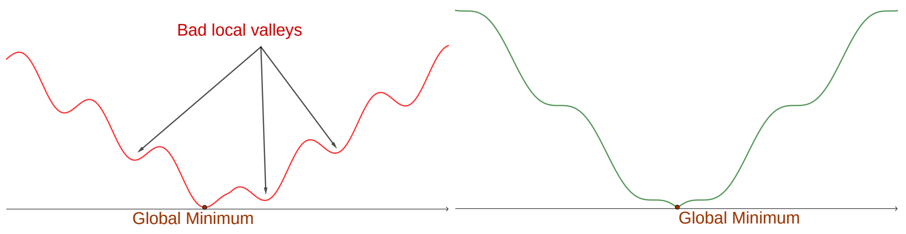

Figure 1: An example loss landscape with bad local valleys (left) and without bad local valley (right).

Our contributions. We identify a family of deep networks with skip connections to the output layer whose loss landscape has no bad local valleys (see Figure 1 for an illustration). Our setting is

[1] These are the networks which have more parameters than necessary to fit the training data

for the empirical loss and there are no distributional assumptions on the training data. Moreover, we study directly the standard cross-entropy loss for multi-class problems. There are little assumptions on the network structure which can be arbitrarily deep and can have convolutional layers (weight sharing) and skip-connections between hidden layers. From a practical perspective, one can generate an architecture which fulfills our conditions by taking an existing CNN architecture and then adding skip-connections from a random subset of N neurons in the network to the output layer (see Figure 2 for an illustration). For these networks we show that there always exists a continuous path from any point in parameter space on which the loss is non-increasing and gets arbitrarily close to zero.

Beside the theoretical analysis, we show in experiments that despite achieving zero training error, the aforementioned class of neural networks generalize well in practice when trained with SGD whereas an alternative training procedure guaranteed to achieve zero training error has significantly worse generalization performance and is overfitting. Thus we think that the presented class of neural networks offer an interesting test bed for future work to study the implicit bias/regularization of SGD.

2 Description of network architecture

We consider a family of deep neural networks which have d input units, H hidden units, m output units and satisfy the following conditions:

1. Every hidden unit of the first layer can take as input an arbitrary subset of units of the input layer.
2. Every hidden unit at higher layers can take as input an arbitrary subset of hidden units from an arbitrary subset of previous hidden layers.
3. Any group of hidden units lying on the same layer can have non-shared or shared weights, in which case the number of incoming units has to be equal.
4. There exist N hidden units which are connected to the output nodes with independent weights (N denotes the number of training samples).
5. The output of every hidden unit j in the network, denoted as $f_j : \mathbb{R}^d \to \mathbb{R}$, is given as

$$f_j(x) = \sigma_j\Big(b_j + \sum_{k:k\to j} f_k(x) u_{k\to j}\Big)$$

where $x \in \mathbb{R}^d$ is an input vector of the network, $\sigma_j : \mathbb{R} \to \mathbb{R}$ is the activation function of unit j, $b_j \in \mathbb{R}$ is the bias of unit j, and $u_{k\to j} \in \mathbb{R}$ the weight from unit k to unit j.

This definition covers a class of deep fully connected and convolutional neural networks with an additional condition on the number of connections to the output layer. In particular, while conventional architectures have just connections from the last hidden layer to the output, we require in our setting that there must exist at least N neurons, “regardless” of their hidden layer, that are connected to the output layer. Essentially, this means that if the last hidden layer of a traditional network has just $L < N$ neurons then one can add connections from $N - L$ neurons in the hidden layers below it to the output layer so that the network fulfills our conditions.

Similar skip-connections have been used in DenseNet (Huang et al., 2017) which are different from identity skip-connections as used in ResNets (He et al., 2016). In Figure 2 we illustrate a network with and without skip connections to the output layer which is analyzed in this paper. We note that several architectures like DenseNets Huang et al. (2017) already have skip-connections between hidden layers in their original architecture, whereas our special skip-connections go from hidden layers directly to the output layer. As our framework allow both kinds to exist in the same network (see Figure 2 for an example), we would like to separate them from each other by making the convention that in the following skip-connections, if not stated otherwise, always refer to ones which connect hidden neurons to output neurons.

We denote by d the dimension of the input and index all neurons in the network from the input layer to the output layer as $1, 2, \ldots, d, d+1, \ldots, d+H, d+H+1, \ldots, d+H+m$ which correspond to d input units, H hidden units and m output units respectively. As we only allow directed arcs from lower layers to upper layers, it follows that $k < j$ for every $k \to j$. Let N be the number of training samples. Suppose that there are M hidden neurons which are directly connected

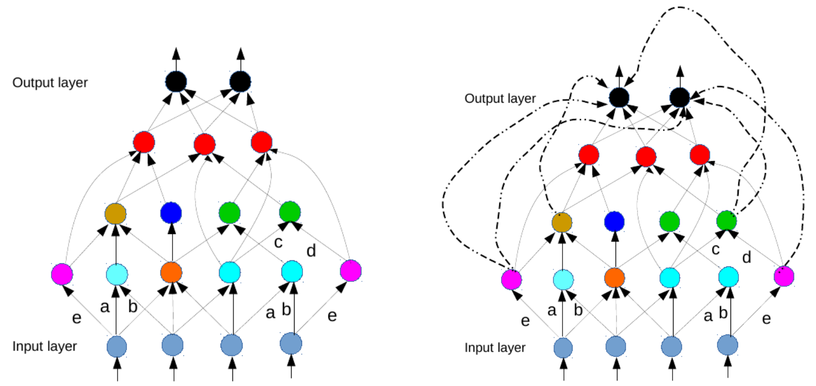

Figure 2: **Left**: An example neural network represented as directed acyclic graph. **Right**: The same network with skip connections added from a subset of hidden neurons to the output layer. All neurons with the same color can have shared or non-shared weights.

to the output with independent weights where it holds $N \leq M \leq H$. Let $\{p_1, \ldots, p_M\}$ with $p_j \in \{d+1, \ldots, d+H\}$ be the set of hidden units which are directly connected to the output units. Let $\mathrm{in}(j)$ be the set of incoming nodes to unit j and $u_j = [u_{k \to j}]_{k \in \mathrm{in}(j)}$ the weight vector of the j-th unit. Let $U = (u_{d+1}, \ldots, u_{d+H}, b_{d+1}, \ldots, b_{d+H})$ denote the set of all weights and biases of all hidden units in the network. Let $V \in \mathbb{R}^{M \times m}$ be the weight matrix which connects the M hidden neurons to the m output units of the network. An important quantity in the following is the matrix $\Psi \in \mathbb{R}^{N \times M}$ defined as

$$\Psi = \begin{bmatrix} f_{p_1}(x_1) & \ldots & f_{p_M}(x_1) \\ \vdots & & \vdots \\ f_{p_1}(x_N) & \ldots & f_{p_M}(x_N) \end{bmatrix} \tag{1}$$

As Ψ depends on U, we write Ψ_U or $\Psi(U)$ as a function of U. Let $G \in \mathbb{R}^{N \times m}$ be the output of the network for all training samples. In particular, G_{ij} is the value of the j-th output neuron for training sample x_i. It follows from our definition that

$$G_{ij} = \langle \Psi_{i:}, V_{:j} \rangle = \sum_{k=1}^{M} f_{p_k}(x_i) V_{kj}, \quad \forall i \in [N], j \in [m]$$

Let $(x_i, y_i)_{i=1}^N$ be the training set where y_i denotes the target class for sample x_i. In this paper we want to analyze the commonly used cross-entropy loss given as

$$\Phi(U, V) = \frac{1}{N} \sum_{i=1}^{N} -\log\left(\frac{e^{G_{iy_i}}}{\sum_{k=1}^{m} e^{G_{ik}}}\right) \tag{2}$$

The cross-entropy loss is bounded from below by zero but this value is not attained. In fact the global minimum of the cross-entropy loss need not exist e.g. if a classifier achieves zero training error then by upscaling the function to infinity one can drive the loss arbitrarily close to zero. Due to this property, we do not study the global minima of the cross-entropy loss but the question if and how one can achieve zero training error. Moreover, we note that sufficiently small cross-entropy loss implies zero training error as shown in the following lemma.

Lemma 2.1 *If $\Phi(U, V) < \frac{\log(2)}{N}$, then the training error is zero.*

Proof: We note that if $\Phi(U, V) < \frac{\log(2)}{N}$, then it holds due to the positivity of the loss,

$$\max_{i=1,\ldots,N} -\log\left(\frac{e^{G_{iy_i}}}{\sum_{k=1}^{m} e^{G_{ik}}}\right) \leq \sum_{i=1}^{N} -\log\left(\frac{e^{G_{iy_i}}}{\sum_{k=1}^{m} e^{G_{ik}}}\right) < \log(2).$$

This implies that for all $i = 1, \ldots, N$,

$$\log\Big(1 + \sum_{k \neq y_i} e^{G_{ik} - G_{iy_i}}\Big) < \log(2) \quad \Longrightarrow \quad \sum_{k \neq y_i} e^{G_{ik} - G_{iy_i}} < 1.$$

In particular: $\max_{k\neq y_i} e^{G_{ik}-G_{iy_i}} < 1$ and thus $\max_{k\neq y_i} G_{ik} - G_{iy_i} < 0$ for all $i = 1,\dots,N$ which implies the result. $\square$

3 MAIN RESULT

The following conditions are required for the main result to hold.

Assumption 3.1 *1. All activation functions $\{\sigma_{d+1},\dots,\sigma_{d+H}\}$ are real analytic and strictly increasing*

2. *Among M neurons $\{p_1,\dots,p_M\}$ which are connected to the output units, there exist N neurons ($N \leq M$) whose activation functions satisfy one of the following conditions:*
   - *σ_{p_j} is bounded and $\lim_{t\to-\infty}\sigma_{p_j}(t) = 0,\ \forall\, 1 \leq j \leq M$*
   - *σ_{p_j} are the softplus activation function (see Equation (3)), and from every neuron p_j there exists a backward path to the first hidden layer such that on this path there is no neuron with skip-connection to the output layer.*
3. *Let n_1 be the number of units in the first hidden layer and denote by S_i for $i \in [d+1, d+n_1]$ their support, then for all $r \neq s \in N$ and $i \neq j \in [d+1, d+n_1]$, it holds*
$$x_r|_{S_i} \neq x_s|_{S_j},$$
where we assume that $|S_i| = |S_j|$ for all $i, j \in [d+1, d+n_1]$.

The first condition of Assumption 3.1 is satisfied for softplus, sigmoid, tanh, etc, whereas the second condition is fulfilled for sigmoid and softplus. For softplus activation function (smooth approximation of ReLU),

$$\sigma_\gamma(t) = \frac{1}{\gamma}\log(1+e^{\gamma t}), \qquad \text{for some } \gamma > 0, \tag{3}$$

we require an additional assumption on the network architecture. The third condition is needed for CNN architectures and could be violated if they have very small receptive fields. However, if the condition is violated for the given training set then after an arbitrarily small random perturbation of all training inputs it will be satisfied with probability 1. Note that the M neurons which are directly connected to the output units can lie on different hidden layers in the network. Also there is no condition on the width of every individual hidden layer as long as the total number of hidden neurons in the network is larger than N so that our condition $M \geq N$ is feasible.

Overall, we would like to stress that Assumption 3.1 covers a quite large class of interesting network architectures but nevertheless allows us to show quite strong results on their empirical loss landscape.

The following key lemma shows that for almost all U, the matrix $\Psi(U)$ has full rank.

Lemma 3.2 *Under Assumption 3.1, the set of U such that $\Psi(U)$ has not full rank N has Lebesgue measure zero.*

While we conjecture that the result of Lemma 3.2 holds for softplus activation function without the additional condition as mentioned in Assumption 3.1, the proof of this is considerably harder for such a general class of neural networks since one has to control the output of neurons with skip connection from different layers which depend on each other. However, please note that the condition is also not too restrictive as it just might require more connections from lower layers to upper layers but it does not require that the network is wide.

We are now ready to state our main result. In the following, we define the α-sublevel set of Φ as $L_\alpha = \{(U,V) \mid \Phi(U,V) < \alpha\}$. We define a local valley to be a connected component of a certain sublevel set L_α. A bad local valley is a local valley on which the loss cannot be made "arbitrarily small". Intuitively, a typical example of a bad local valley is a small neighborhood around a sub-optimal strict local minimum.

Theorem 3.3 *The following holds under Assumption 3.1:*

1. *There exist uncountably many solutions with zero training error.*
2. *The loss landscape of Φ does not have any bad local valley.*
3. *There exists no suboptimal strict local minimum.*
4. *There exists no local maximum.*

Proof:

1. By Lemma 3.2 the set of U such that $\Psi(U)$ has not full rank N has Lebesgue measure zero. Given U such that Ψ has full rank, the linear system $\Psi(U)V = Y$ has for every possible target output matrix $Y \in \mathbb{R}^{N\times m}$ at least one solution V. As this possible for almost all U, there exist uncountably many solutions achieving zero training error.
2. Let C be a non-empty, connected component of some α-sublevel set L_α for $\alpha > 0$. Suppose by contradiction that the loss on C cannot be made arbitrarily small, that is there exists an $\epsilon > 0$ such that $\Phi(U,V) \geq \epsilon$ for all $(U,V) \in C$, where $\epsilon < \alpha$. By definition, L_α can be written as the pre-image of an open set under a continuous function, that is $L_\alpha = \Phi^{-1}(\{a \mid a < \alpha\})$, and thus L_α must be an open set (see Proposition A.2). Since C is a non-empty connected component of L_α, C must be an open set as well, and thus C has non-zero Lebesgue measure. By Lemma 3.2 the set of U where $\Psi(U)$ has not full rank has measure zero and thus C must contain a point (U,V) such that $\Psi(U)$ has full rank. Let Y be the usual zero-one one-hot encoding of the target network output. As $\Psi(U)$ has full rank, there always exist V^* such that $\Psi(U)V^* = Yt^*$, where $t^* = \log\big(\frac{m-1}{e^{\frac{\epsilon}{2}}-1}\big)$ Note that the loss of (U,V^*) is
$$\Phi(U,V^*) = -\log\Big(\frac{e^{t^*}}{e^{t^*}+(m-1)}\Big) = \log(1+(m-1)e^{-t^*}) = \frac{\epsilon}{2}.$$
As the cross-entropy loss $\Phi(U,V)$ is convex in V and $\Phi(U,V) < \alpha$ we have for the line segment $V(\lambda) = \lambda V + (1-\lambda)V^*$ for $\lambda \in [0,1]$,
$$\Phi(U,V(\lambda)) \leq \lambda\Phi(U,V) + (1-\lambda)\Phi(U,V^*) < \lambda\alpha + (1-\lambda)\frac{\epsilon}{2} < \alpha.$$
Thus the whole line segment is contained in L_α and as C is a connected component it has to be contained in C. However, this contradicts the assumption that for all $(U,V) \in C$ it holds $\Phi(U,V) \geq \epsilon$. Thus on every connected component C of L_α the training loss can be made arbitrarily close to zero and thus the loss landscape has no bad valleys.
3. Let (U_0,V_0) be a strict suboptimal local minimum, then there exists $r > 0$ such that $\Phi(U,V) > \Phi(U_0,V_0) > 0$ for all $(U,V) \in B((U_0,V_0),r) \setminus \{(U_0,V_0)\}$ where $B(\cdot,r)$ denotes a closed ball of radius r. Let $\alpha = \min_{(U,V)\in\partial B\big((U_0,V_0),r\big)} \Phi(U,V)$ which exists as Φ is continuous and the boundary $\partial B\big((U_0,V_0),r\big)$ of $B\big((U_0,V_0),r\big)$ is compact. Note that $\alpha > \Phi(U_0,V_0)$ as (U_0,V_0) is a strict local minimum. Consider the sub-level set $D = L_{\frac{\alpha+\Phi(U_0,V_0)}{2}}$. As $\Phi(U_0,V_0) < \frac{\alpha+\Phi(U_0,V_0)}{2}$ it holds $(U_0,V_0) \in D$. Let E be the connected component of D which contains (U_0,V_0), that is, $(U_0,V_0) \in E \subseteq D$. It holds $E \subset B\big((U_0,V_0),r\big)$ as $\Phi(U,V) < \frac{\alpha+\Phi(U_0,V_0)}{2} < \alpha$ for all $(U,V) \in E$. Moreover, $\Phi(U,V) \geq \Phi(U_0,V_0) > 0$ for all $(U,V) \in E$ and thus Φ can not be made arbitrarily small on a connected component of a sublevel set of Φ and thus E would be a bad local valley which contradicts 3.3.2.
4. Suppose by contradiction that (U,V) is a local maximum. Then the Hessian of Φ is negative semi-definite. However, as submatrices of negative semi-definite matrices are again negative semi-definite, then also the Hessian of Φ w.r.t V must be negative semi-definite However, Φ is always convex in V and thus its Hessian restricted to V is positive semi-definite. The only matrix which is both p.s.d. and n.s.d. is the zero matrix. It follows that $\nabla^2_V\Phi(U,V) = 0$. One can easily show that
$$\nabla^2_{V_{:j}}\Phi = \sum_{i=1}^{N} \frac{e^{G_{ij}}}{\sum_{k=1}^m e^{G_{ik}}}\Big(1 - \frac{e^{G_{ij}}}{\sum_{k=1}^m e^{G_{ik}}}\Big)\Psi_{i:}\Psi_{i:}^T$$

From Assumption 3.1 it holds that there exists $j \in [N]$ s.t. σ_{p_j} is strictly positive, and thus some entries of $\Psi_{i:}$ must be strictly positive. Moreover, one has $\frac{e^{G_{ij}}}{\sum_{k=1}^{m} e^{G_{ik}}} \in (0,1)$. It follows that some entries of $\nabla^2_{V_{:j}} \Phi$ must be strictly positive. Thus $\nabla^2_{V_{:j}} \Phi$ cannot be identically zero, leading to a contradiction. Therefore Φ has no local maximum.

□

Theorem 3.3 shows that there are infinitely many solutions which achieve zero training error, and the loss landscape is nice in the sense that from any point in the parameter space there exists a continuous path that drives the loss arbitrarily close to zero (and thus a solution with zero training error) on which the loss is non-increasing.

While the networks are over-parameterized, we show in the next Section 4 that the modification of standard networks so that they fulfill our conditions leads nevertheless to good generalization performance, often even better than the original network. We would like to note that the proof of Theorem 3.3 also suggests a different algorithm to achieve zero training error: one initializes all weights, except the weights to the output layer, randomly (e.g. Gaussian weights), denoted as U, and then just solves the linear system $\Psi(U)V = Y$ to obtain the weights V to the output layer. Basically, this algorithm uses the network as a random feature generator and fits the last layer directly to achieve zero training error. The algorithm is successful with probability 1 due to Lemma 3.2. Note that from a solution with zero training error one can drive the cross-entropy loss to zero by upscaling to infinity but this does not change the classifier. We will see, that this simple algorithm shows bad generalization performance and overfitting, whereas training the full network with SGD leads to good generalization performance. This might seem counterintuitive as our networks have more parameters than the original networks but is inline with recent observations in Zhang et al. (2017) that state-of-the art networks, also heavily over-parameterized, can fit even random labels but still generalize well on the original problem. Due to this qualitative difference of SGD and the simple algorithm which both are able to find solutions with zero training error, we think that our class of networks is an ideal test bed to study the implicit regularization/bias of SGD, see e.g. Soudry et al. (2018).

4 Experiments

The main purpose of this section is to investigate the generalization ability of practical neural networks with skip-connections added to the output layer to fulfill Assumption 3.1.

Datasets. We consider MNIST and CIFAR10 datasets. MNIST contains 5.5×10^4 training samples and 10^4 test samples, and CIFAR10 has 5×10^4 training samples and 10^4 test samples. We do not use any data pre-processing in all of our experiments. For every experiment on CIFAR10 described below, we consider both settings with and without data-augmentation. For data-augmentation, we follow the procedure as described in Zagoruyko & Komodakis (2016) by considering random crops of size 32×32 after 4 pixel padding on each side of the training images and random horizontal flips with probability 0.5.

Network architectures. For MNIST, we use a plain CNN architecture with 13 layers, denoted as CNN13 (see Table 3 in the appendix for more details about this architecture). For CIFAR10 we use VGG11, VGG13, VGG16 (Simonyan & Zisserman, 2015) and DenseNet121 (Huang et al., 2017). As the VGG models were originally proposed for ImageNet and have very large fully connected layers, we adapted these layers for CIFAR10 by reducing their width from 4096 to 128. For each given network, we create the corresponding skip-networks by adding skip-connections to the output so that our condition $M \geq N$ from the main theorem is satisfied. In particular, we aggregate all neurons of all the hidden layers in a pool and randomly choose from there a subset of N neurons to be connected to the output layer (see e.g. Figure 2 for an illustration). As existing network architectures have a large number of feature maps per layer, the total number of neurons is often very large compared to number of training samples, thus it is easy to choose from there a subset of N neurons to connect to the output. In the following, we test both sigmoid and softplus activation function ($\gamma = 20$) for each network architecture and their skip-variants. We use the standard cross-entropy loss and train all models with SGD+Nesterov momentum for 300 epochs. The initial learning rate is set to 0.1 for Densenet121 and 0.01 for the other architectures. Following Huang et al. (2017), we also divide the learning rate by 10 after 50% and 75% of the total number of training epochs. Note that we do not use any explicit regularization like weight decay or dropout.

In all our experiments, the conditions of Assumption 3.1 are satisfied for the sigmoid activation function, and thus the main results of Theorem 3.3 hold. However, for $\mathrm{softplus}$, we do not check if the additional condition in Assumption 3.1 that there exists a backward path from all N skip-neurons to the first hidden layer only visiting neurons which are not skip-neurons as this is quite costly. Our main goal in the experiments is to investigate the influence of the additional skip-connections to the output layer on the generalization performance. We report the test accuracy for the original models and the ones with skip-connections to the output layer. For the latter one we have two different algorithms: standard SGD for training the full network as described above (SGD) and the randomized procedure (rand). The latter one uses a slight variant of the simple algorithm described at the end of the last section: randomly initialize the weights of the network U up to the output layer by drawing each of them from a truncated Gaussian distribution with zero mean and variance $\frac{2}{d}$ where d is the number of weight parameters and the truncation is done after ± 2 standard deviations (standard keras initialization), then use SGD to optimize the weights V for a linear classifier with fixed features $\Psi(U)$ which is a convex optimization problem. Note that the rand algorithm cannot be used with data augmentation in a straightforward way and thus we skip it for this part.

Our experimental results are summarized in Table 1 for MNIST and Table 2 for CIFAR10. For skip-models, we report mean and standard deviation over 8 random choices of the subset of N neurons connected to the output.

	Sigmoid activation function	**Softplus activation function**
CNN13	11.35	99.20
CNN13-skip (SGD)	98.40 ± 0.07	99.14 ± 0.04

Table 1: Test accuracy ($\%$) of CNN13 on MNIST dataset. CNN13 denotes the original architecture from Table 3 while CNN13-skip denotes the corresponding skip-model. There are in total $179,840$ hidden neurons from the original CNN13 (see Table 3), out of which we choose a random subset of $N = 55,000$ neurons to connect to the output layer to obtain CNN13-skip.

Discussion of results. First of all, we note that adding skip connections to the output improves the test accuracy in almost all networks (with the exception of Densenet121) when the full network is trained with SGD. In particular, for the sigmoid activation function the skip connections allow for all models except Densenet121 to get reasonable performance whereas training the original model fails. This effect can be directly related to our result of Theorem 3.3 that the loss landscape of skip-networks has no bad local valley and thus it is not difficult to reach a solution with zero training error. The exception is Densenet121 which gets already good performance for the sigmoid activation function for the original model. We think that the reason is that the original Densenet121 architecture has already quite a lot of skip-connections between the hidden layers which thus improves the loss surface already so that the additional connections added to the output units are not necessary anymore.

The second interesting observation is that we do not see any sign of overfitting for the SGD version even though we have increased for all models the number of parameters by adding skip connections to the output layer and we know from Theorem 3.3 that for all the skip-models one can easily achieve zero training error. This is in line with the recent observation of Zhang et al. (2017) that modern heavily over-parameterized networks can fit everything (random labels, random input) but nevertheless generalize well on the original training data when trained with SGD. This is currently an active research area to show that SGD has some implicit bias (Neyshabur et al., 2017; Brutzkus et al., 2018; Soudry et al., 2018) which leads to a kind of regularization effect similar to the linear least squares problem where SGD converges to the minimum norm solution. Our results confirm that there is an implicit bias of SGD as we see a strong contrast to the (rand) results obtained by using the network as a random feature generator and just fitting the last layer which also leads to solutions with zero training error with probability 1 as shown in Lemma 3.2 and the proof of Theorem 3.3. For this (rand) version we see that the test accuracy gets worse as one is moving from simpler networks (VGG11) to more complex ones (VGG16 and Densenet121) which is a sign of overfitting. Thus we think that our class of networks is also an interesting test bed to understand the implicit regularization

	Sigmoid activation function		Softplus activation function	
Model	C-10	C-10$^+$	C-10	C-10$^+$
VGG11	10	10	78.92	88.62
VGG11-skip (rand)	62.81 ± 0.39	-	64.49 ± 0.38	-
VGG11-skip (SGD)	**72.51** ± 0.35	**85.55** ± 0.09	**80.57** ± 0.40	**89.32** ± 0.16
VGG13	10	10	80.84	90.58
VGG13-skip (rand)	61.50 ± 0.34	-	61.42 ± 0.40	-
VGG13-skip (SGD)	**70.24** ± 0.39	**86.48** ± 0.32	**81.94** ± 0.40	**91.06** ± 0.12
VGG16	10	10	81.33	90.68
VGG16-skip (rand)	61.57 ± 0.41	-	61.46 ± 0.34	-
VGG16-skip (SGD)	**70.61** ± 0.36	**86.42** ± 0.31	**81.91** ± 0.24	**91.00** ± 0.22
Densenet121	**86.41**	**90.93**	**89.31**	**94.20**
Densenet121-skip (rand)	52.07 ± 0.48	-	55.39 ± 0.48	-
Densenet121-skip (SGD)	81.47 ± 1.03	90.32 ± 0.50	86.76 ± 0.49	93.23 ± 0.42

Table 2: Test accuracy (%) of several CNN architectures with/without skip-connections on CIFAR10 ($^+$ denotes data augmentation). For each model A, A-skip denotes the corresponding skip-model in which a subset of N hidden neurons "randomly selected" from the hidden layers are connected to the output units. For Densenet121, these neurons are randomly chosen from the first dense block. The names in open brackets (rand/SGD) specify how the networks are trained: **rand** (U is randomized and fixed while V is learned with SGD), **SGD** (both U and V are optimized with SGD).

effect of SGD. It seems that SGD selects from the infinite pool of solutions with zero training error one which generalizes well, whereas the randomized feature generator selects one with much worse generalization performance.

5 Related work

In the literature, many interesting theoretical results have been developed on the loss landscape of neural networks Haeffele & Vidal (2017); Choromanska et al. (2015); Kawaguchi (2016); Safran & Shamir (2016); Hardt & Ma (2017); Yun et al. (2017); Venturi et al. (2018); Zhang et al. (2018) and the behavior of SGD for the minimization of training objective has been also analyzed for various settings (Andoni et al., 2014; Sedghi & Anandkumar, 2015; Janzamin et al., 2016; Gautier et al., 2016; Brutzkus & Globerson, 2017; Soltanolkotabi, 2017; Soudry & Hoffer, 2017; Zhong et al., 2017; Tian, 2017; Du et al., 2018; Wang et al., 2018) to name a few. Most of current results on the loss landscape of neural networks are however limited to shallow networks (one hidden layer), deep linear networks and/or make assumptions on the distribution of the data. An interesting recent exception is Liang et al. (2018) where they show for binary classification one neuron with a skip-connection to the output layer and exponential activation function is enough to eliminate all bad local minima under mild conditions on the loss function. More closely related in terms of the setting are (Nguyen & Hein, 2017; 2018) where they study the loss surface of fully connected and convolutional networks if one of the layers has more neurons than the number of training samples for the standard multi-class problem. However, the presented results are stronger as we show that our networks do not have any suboptimal strict local minima and there is less over-parameterization if the number of classes is small.

6 Conclusion

We have identified a class of deep neural networks whose loss landscape has no bad local valleys. While our networks are over-parameterized and can easily achieve zero training error, they generalize well in practice when trained with SGD. Interestingly, a simple different algorithm using the network as random feature generator also achieves zero training error but has significantly worse generalization performance. Thus we think that our class of models is an interesting test bed for studying the implicit regularization effect of SGD.

REFERENCES

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

T. M. Apostol. *Mathematical analysis*. Addison Wesley, 1974.

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

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

A. Brutzkus and A. Globerson. Globally optimal gradient descent for a convnet with gaussian inputs. *ICML*, 2017.

A. Brutzkus, A. Globerson, E. Malach, and S. Shalev-Shwartz. Sgd learns over-parameterized networks that provably generalize on linearly separable data. *ICLR*, 2018.

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

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. *NIPS*, 2014.

S. Du, J. Lee, Y. Tian, A. Singh, and B. Póczos. Gradient descent learns one-hidden-layer cnn: Don't be afraid of spurious local minima. *ICML*, 2018.

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

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

B. D. Haeffele and R. Vidal. Global optimality in neural network training. *CVPR*, 2017.

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

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

G. Huang, Z. Liu, L. Maaten, and K. Weinberger. Densely connected convolutional networks. *CVPR*, 2017.

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

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *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. *NIPS*, 1990.

S. Liang, R. Sun, J. D. Lee, and R. Srikant. Adding one neuron can eliminate all bad local minima. *arXiv:1805.08671*, 2018.

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

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

B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. Exploring generalization in deep learning. *NIPS*, 2017.

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

Q. Nguyen and M. Hein. Optimization landscape and expressivity of deep cnns. *ICML*, 2018.

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

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

I. Safran and O. Shamir. Spurious local minima are common in two-layer relu neural networks. *ICML*, 2018.

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

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

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. *ICLR*, 2015.

M. Soltanolkotabi. Learning relus via gradient descent. *NIPS*, 2017.

D. Soudry and E. Hoffer. Exponentially vanishing sub-optimal local minima in multilayer neural networks. *ICLR Workshop 2018*, 2017.

D. Soudry, E. Hoffer, M. S. Nacson, and N. Srebro. The implicit bias of gradient descent on separable data. *ICLR*, 2018.

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

L. Venturi, A. S. Bandeira, and J. Bruna. Spurious valleys in two-layer neural network optimization landscapes. *arXiv:1802.06384*, 2018.

G. Wang, G. B. Giannakis, and J. Chen. Learning relu networks on linearly separable data: Algorithm, optimality, and generalization. *arXiv:1808.04685*, 2018.

C. Yun, S. Sra, and A. Jadbabaie. Global optimality conditions for deep neural networks. *ICLR*, 2017.

S. Zagoruyko and N. Komodakis. Wide residual networks. *BMCV*, 2016.

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

H. Zhang, J. Shao, and R. Salakhutdinov. Deep neural networks with multi-branch architectures are less non-convex. *arXiv:1806.01845*, 2018.

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

A Mathematical Tools

In the proof of Lemma 3.2 we make use of the following property of analytic functions.

Lemma A.1 *(Nguyen, 2015; Mityagin, 2015) If $f : \mathbb{R}^n \to \mathbb{R}$ is a real analytic function which is not identically zero then the set $\{x \in \mathbb{R}^n \mid f(x) = 0\}$ has Lebesgue measure zero.*

We recall the following standard result from topology (see *e.g.* Apostol (1974), Theorem 4.23, p. 82), which is used in the proof of Theorem 3.3.

Proposition A.2 *Let $f : \mathbb{R}^m \to \mathbb{R}^n$ be a continuous function. If $U \subseteq \mathbb{R}^n$ is an open set then $f^{-1}(U)$ is also open.*

B Proof of Lemma 3.2

Proof: We assume w.l.o.g. that $\{p_1, \dots, p_N\}$ is a subset of the neurons with skip connections to the output layer (see Assumption 3.1). In the following, we will show that there exists a weight configuration U such that the submatrix $\Psi_{1:N,1:N}$ has full rank. Using then that the determinant is an analytic function together with Lemma A.1, we will conclude that the set of weight configurations U such that Ψ has *not* full rank has Lebesgue measure zero.

We remind that all the hidden units in the network are indexed from the first hidden layer till the higher layers as $d+1, \dots, d+H$. For every hidden neuron $j \in [d+1, d+H]$, u_j denotes the associated weight vector

$$u_j = [u_{k\to j}]_{k \in \mathrm{in}(j)} \in \mathbb{R}^{|\mathrm{in}(j)|}, \quad \text{where } \mathrm{in}(j) = \text{the set of incoming units to unit } j.$$

Let n_1 be the number of units of the first hidden layer.

1. We first pick for all hidden units of the first layer the weights $\{u_{d+1}, \dots, u_{d+n_1}\}$ such that

$$\sum_{k\to j} f_k(x_i) u_{k\to j} \neq \sum_{k\to j} f_k(x_{i'}) u_{k\to j} \quad \forall i \neq i', j \in [d+1, d+n_1]$$

   Note that if a unit j belongs to the first hidden layer, then every incoming unit $k \to j$ must come from the input layer, that is, $k \in [1, d]$. Thus the above condition can be rewritten as

$$\sum_{k\to j} (x_i)_k u_{k\to j} \neq \sum_{k\to j} (x_{i'})_k u_{k\to j} \quad \forall i \neq i', j \in [d+1, d+n_1] \tag{4}$$

   Intuitively, this condition guarantees that the values of every individual unit from the first hidden layer are different across all training samples. Note that the above sums can be rewritten as an inner product of some input patch and the corresponding weight vector. Thus using condition iii) from the Assumptions 3.1, the set of weights of the first hidden layer which do not satisfy the condition (4) has Lebesgue measure zero.

2. We choose $\{u_{d+n_1+1}, \dots, u_{d+H}\}$ s.t. every weight vector u_j has exactly one 1 in and 0 elsewhere.

3. Let $\alpha := (\alpha_{d+1}, \dots, \alpha_{d+H})$ be a tuple of positive scalars and let $\beta \in \mathbb{R}$ such that $\sigma_{p_j}(\beta) \neq 0$ for every $j \in [N]$. For every neuron $p_j (j \in [N])$ which has a skip-connection to the output layer, let us pick the bias

$$b_{p_j} = \beta - \alpha_{p_j} \sum_{k\to p_j} f_k(x_j) u_{k\to p_j}.$$

   The biases for the other hidden neurons are set to zero, that is, $b_j = 0$ for every $j \in \{d+1, \dots, d+H\} \setminus \{p_1, \dots, p_N\}$.

4. In the following, we consider a family of configurations of network parameters of the form $(\alpha_j u_j, b_j)_{j=d+1}^{d+H}$ where $(u_j, b_j)_{j=d+1}^{d+H}$ are chosen as described above. By our construciton so far, the output of each hidden neuron is

$$\begin{aligned} f_{p_j}(x_i) &= \sigma_{p_j}\Big(\beta + \alpha_{p_j} \sum_{k\to p_j} \big(f_k(x_i) - f_k(x_j)\big) u_{k\to p_j}\Big) \quad \forall j \in [N], \\ f_j(x_i) &= \sigma_j\Big(\alpha_j \sum_{k\to j} f_k(x_i) u_{k\to j}\Big) \quad \forall j \in \{d+1, \dots, d+H\} \setminus \{p_1, \dots, p_N\}. \end{aligned} \tag{5}$$

5. Now, one can show for every $\alpha > 0$ and every $j \in [N]$ that

$$\sum_{k\to p_j} f_k(x_i) u_{k\to p_j} \neq \sum_{k\to p_j} f_k(x_{i'}) u_{k\to p_j} \quad \forall i \neq i'.$$

   Moreover, if one sorts all elements of the set $\Big\{\sum_{k\to p_j} f_k(x_1) u_{k\to p_j}, \dots, \sum_{k\to p_j} f_k(x_N) u_{k\to p_j}\Big\}$ in increasing order then this order is invariant w.r.t. every positive tuple α.

Proof: The statement is true for every $j \in [N]$ where $p_j \in [d+1, d+n_1]$ by our construction in (4). Note that for those $j \in [N]$ where $p_j \notin [d+1, d+n_1]$, it holds by our contruction in 2) that u_{p_j} has exactly one 1 in its entries and zero elsewhere. For this purpose, let $c(j)$ be the index of an incoming unit to every hidden unit $j \in \{d+n_1+1, \ldots, d+H\}$ such that $u_{c(j)\to j} = 1$ and $u_{k\to j} = 0$ for every $k \neq c(j)$. With this notation, it holds for every $j \in [N]$ where $p_j \notin [d+1, d+n_1]$ that

$$\sum_{k\to p_j} f_k(x_i) u_{k\to p_j} = f_{c(p_j)}(x_i) \tag{6}$$

By considering the value of the sum $\sum_{k\to p_j} f_k(x) u_{k\to p_j}$ at different training samples, it holds for every $i \neq i'$ that

$$\begin{aligned} & \sum_{k\to p_j} f_k(x_i) u_{k\to p_j} < \sum_{k\to p_j} f_k(x_{i'}) u_{k\to p_j} \\ \iff\ & f_{c(p_j)}(x_i) < f_{c(p_j)}(x_{i'}) \\ \iff\ & \sigma^{-1}_{c(p_j)}\Big(f_{c(p_j)}(x_i)\Big) < \sigma^{-1}_{c(p_j)}\Big(f_{c(p_j)}(x_{i'})\Big) \\ \iff\ & \alpha_{c(p_j)} \sum_{k\to c(p_j)} f_k(x_i) u_{k\to c(p_j)} < \alpha_{c(p_j)} \sum_{k\to c(p_j)} f_k(x_{i'}) u_{k\to c(p_j)} \\ \iff\ & \sum_{k\to c(p_j)} f_k(x_i) u_{k\to c(p_j)} < \sum_{k\to c(p_j)} f_k(x_{i'}) u_{k\to c(p_j)} \end{aligned}$$

where the first step follows from (6), the second step follows from the fact that all activation functions are strictly increasing by Assumption 3.1 and thus there exists an inverse function, the third step follows from (5), and the last step follows from the fact that $\alpha_{c(p_j)} > 0$.

Now, if $c(p_j)$ is already a hidden unit of the first hidden layer then we are done. Otherwise, one can repeatedly apply the same chain of inequalities to $c(p_j), c(c(p_j)), \ldots$ until one eventually reaches a neuron from the first hidden layer.

In summary, we have shown that the order of all elements from the set $\Big\{\sum_{k\to p_j} f_k(x_1) u_{k\to p_j}, \ldots, \sum_{k\to p_j} f_k(x_N) u_{k\to p_j}\Big\}$ is fully determined by the order of elements from the set $\Big\{\sum_{k\to q_j} (x_1)_k u_{k\to q_j}, \ldots, \sum_{k\to q_j} (x_N)_k u_{k\to q_j}\Big\}$ where $q_j = c(c(\ldots c(p_j)\ldots)) \in [d+1, d+n_1]$ is some neuron in the first hidden layer. Moreover, this order is independent of the chosen α. Note that this order can be different for different neurons q_j in the first hidden layer, and thus can be different for different p_j's.

6. Let π be a permutation such that it holds for every $j = 1, 2, \ldots, N$ that

$$\pi(j) = \operatorname*{arg\,max}_{i\in\{1,\ldots,N\}\setminus\{\pi(1),\ldots,\pi(j-1)\}} \sum_{k\to p_j} f_k(x_i) u_{k\to p_j} \tag{7}$$

It follows from our previous argument that π is invariant w.r.t. every $\alpha > 0$. By definition, it holds that

$$\sum_{k\to p_j} f_k(x_{\pi_j}) u_{k\to p_j} > \sum_{k\to p_j} f_k(x_{\pi_i}) u_{k\to p_j} \quad \forall\, i, j \in [N], i > j$$

Since π is independent of every positive tuple α, it can be fixed in the beginning by (7) with some fixed choice of $\alpha > 0$. Thus one can assume w.l.o.g. that π is the identity permutation as otherwise one can relabel the training data according to π and the rank of Ψ is invariant under relabeling. Thus it holds for every $\alpha > 0$ that

$$\delta_{ij} := \sum_{k\to p_j} f_k(x_i) u_{k\to p_j} - \sum_{k\to p_j} f_k(x_j) u_{k\to p_j} < 0 \quad \forall\, i, j \in [N], i > j \tag{8}$$

Now, we are ready to show that there exists a tuple $\alpha = (\alpha_{d+1}, \ldots, \alpha_{d+N}) > 0$ for which Ψ has full rank. We consider two cases:

- In the first case, the activation functions $\sigma_{p_j} : \mathbb{R} \to \mathbb{R}$ for every $j \in [N]$ are strictly increasing, bounded and $\lim_{t\to-\infty} \sigma_{p_j}(t) = 0$. In the following, let $l(j)$ denote the layer index of the hidden unit j. For every hidden unit $j \in \{d+1, \ldots, d+H\}$ we set

$$\alpha_j = \max\left\{1, \max_{k\in[N] \mid l(p_k)=l(j)} \max_{i>k} \frac{\sigma_{p_k}^{-1}(\epsilon) - \beta}{f_{c(p_k)}(x_i) - f_{c(p_k)}(x_k)}\right\} \tag{9}$$

where $\epsilon > 0$ is an arbitrarily small constant which will be specified later. There are a few remarks we want to make for Eq. (9) before proceeding with our proof. First, the second term in (9) can be empty if there is no skip-connection unit p_k which lies on the same layer as unit j, in which case α_j is simply set to 1. Second, α_j's are well-defined by constructing the values $f_{c(p_k)}(x_r)$, $r = 1, \ldots, N$ by a forward pass through the network (note that the network is a directed, acyclic graph; in particular, in the formula of α_j, one has $l(c(p_k)) < l(p_k) = l(j)$ and thus the computation of α_j is feasible given the values of hidden units lying below the layer of unit j, namely $f_{c(p_k)}$). Third, since we are considering the parameter configuration of the form $(\alpha_j u_j, b_j)_{j=d+1}^{d+H}$ and we want to show that there exists $\{\alpha_j\}_{j=d+1}^{d+H}$ for which Ψ has full rank, we need to guarantee that by choosing the values of $\{\alpha_j\}_{j=d+1}^{d+H}$ all the potential parameter sharing conditions of multiple hidden units from the same layer can still be satisfied. It turns out that this is indeed guaranteed by our construction. In particular, if j and j' are two hidden units from the same layer, *i.e.* $l(j) = l(j')$, then it follows from (9) that $\alpha_j = \alpha_{j'}$. Moreover, the weight vectors u_j and $u_{j'}$ can be always chosen to be identical while still satisfying the properties mentioned in the first two construction steps of the proof. Thus it holds that $\alpha_j u_j = \alpha_{j'} u'$ for every two hidden units j, j' lying on the same layer, meaning that all the potential weight sharing conditions between multiple hidden units of the same layer in the network are satisfied.

  The main idea of choosing the values of α_{p_j} (extracted from (9)) is to obtain

$$\Psi_{ij} = f_{p_j}(x_i) \le \epsilon \quad \forall\, i, j \in [N], i > j. \tag{10}$$

  To see this, one first observes from (9) that $\alpha > 0$ and thus it follows from (8) that

$$\delta_{ij} = f_{c(p_j)}(x_i) - f_{c(p_j)}(x_j) < 0 \quad \forall\, i, j \in [N], i > j. \tag{11}$$

  Since (9) holds for all the hidden units j in the network, one can replace j in (9) with every skip-connection unit p_j in order to obtain

$$\alpha_{p_j} > \max_{i>j} \frac{\sigma_{p_j}^{-1}(\epsilon) - \beta}{f_{c(p_j)}(x_i) - f_{c(p_j)}(x_j)} \quad \forall\, j \in [N]$$

  which combined with (11) leads to

$$\alpha_{p_j}\big(f_{c(p_j)}(x_i) - f_{c(p_j)}(x_j)\big) \le \sigma_{p_j}^{-1}(\epsilon) - \beta \quad \forall\, i, j \in [N], i > j.$$

  From (5) one has for every $i, j \in [N], i > j$ that

$$\begin{aligned} f_{p_j}(x_i) &= \sigma_{p_j}\Big(\beta + \alpha_{p_j} \sum_{k\to p_j} \big(f_k(x_i) - f_k(x_j)\big) u_{k\to p_j}\Big) \\ &= \sigma_{p_j}\Big(\beta + \alpha_{p_j}\big(f_{c(p_j)}(x_i) - f_{c(p_j)}(x_j)\big)\Big) \\ &\le \sigma_{p_j}\Big(\beta + \sigma_{p_j}^{-1}(\epsilon) - \beta\Big) \\ &= \epsilon \end{aligned}$$

  which finishes the proof of (10).

  Coming back to the main proof of the lemma, since σ_{p_j} $(j \in [N])$ are bounded there exists a finite positive constant C such that it holds that

$$|\Psi_{ij}| \le C \quad \forall\, i, j \in [N] \tag{12}$$

  By the Leibniz-formula one has

$$\det(\Psi_{1:N,1:N}) = \prod_{j=1}^{N} \sigma_{p_j}(\beta) + \sum_{\pi\in S_N\setminus\{\gamma\}} \operatorname{sign}(\pi) \prod_{j=1}^{N} \Psi_{\pi(j)j} \tag{13}$$

where S_N is the set of all $N!$ permutations of the set $\{1,\ldots,N\}$ and γ is the identity permutation. Now, one observes that for every permutation $\pi \neq \gamma$, there always exists at least one component j where $\pi(j) > j$ in which case it follows from (10) and (12) that

$$\Big|\sum_{\pi \in S_N \setminus \{\gamma\}} \operatorname{sign}(\pi) \prod_{j=1}^{N} \Psi_{\pi(j)j}\Big| \;\leq\; N!\, C^{N-1}\,\epsilon$$

By choosing $\epsilon = \frac{\left|\prod_{j=1}^{N} \sigma_{p_j}(\beta)\right|}{2N!C^{N-1}}$, we get that

$$\det(\Psi_{1:N,1:N}) \geq \prod_{j=1}^{N} \sigma_{p_j}(\beta) - \frac{1}{2}\prod_{j=1}^{N} \sigma_{p_j}(\beta) = \frac{1}{2}\prod_{j=1}^{N} \sigma_{p_j}(\beta) \neq 0$$

and thus Ψ has full rank.

- In the second case we consider the softplus activation function under the condition that there exist N neurons with skip-connection to the output layer which have a path backward through the network which does not contain any skip-connection neurons.

  We choose the α for the non-skip connection neurons as before noting again that by the particular choice of u it holds,

$$\sum_{k \to p_j} f_k(x_i) u_{k \to p_j} = f_{c(p_j)}(x_i)$$

  Then we set all $\alpha_{p_1}, \ldots, \alpha_{p_N}$ to α we get

$$\begin{aligned} f_{p_j}(x_i) &= \sigma_{p_j}\Big(\beta + \alpha \sum_{k \to p_j} \big(f_k(x_i) - f_k(x_j)\big) u_{k \to p_j}\Big) \quad \forall j \in [N], \\ &= \sigma_{p_j}\Big(\beta + \alpha\big(f_{c(p_j)}(x_i) - f_{c(p_j)}(x_j)\big)\Big) \\ f_j(x_i) &= \sigma_j\big(f_{c(j)}(x_i)\big) \quad \forall j \in \{d+1,\ldots,d+H\} \setminus \{p_1,\ldots,p_N\}\,. \end{aligned} \tag{14}$$

  Note that by assumption the path $c^{(k)}(p_j)$ does not contain any skip connection unit and will eventually end up at some neuron $q_j \in [d+1, d+n_1]$ of the first hidden layer after some L_j steps. Thus we can write

$$f_{p_j}(x_i) = \sigma_{p_j}\Big(\beta + \alpha\big(g(x_i) - g(x_j)\big)\Big),$$

  where

$$g(x_i) = \sigma_{c(p_j)}(\sigma_{c(c(p_j))}(\ldots(\sum_{k \to q_j} (x_i)_k u_{k \to q_j})\ldots)) \quad \forall\, i \in [N].$$

  Moreover, it holds that $g(x_i) < g(x_j)$ for every $i > j$. Note that softplus fulfills for $t < 0$, $\sigma_\gamma(t) \leq \frac{1}{\gamma} e^{\gamma t}$, whereas for $t > 0$ one has $\sigma_\gamma(t) \leq \frac{1}{\gamma} + t$. The latter property implies $\sigma^{(K)}(t) \leq \frac{K}{\gamma} + t$. Finally, this together implies that there exist positive constants c_1, c_2, c_3, c_4 such that it holds for all $j \in [N]$ that

$$|\prod_{j=1}^{N} \Psi_{\pi(j)j}| \;\leq\; c_1 e^{-\alpha c_2} (c_3 + \alpha)^{N-1}.$$

  This can be made arbitrarily small by increasing α. We thus get

$$\lim_{\alpha \to \infty} \det(\Psi_{1:N,1:N}) = \prod_{j=1}^{N} \sigma_{p_j}(\beta) \neq 0$$

So far, we have shown that there always exist U such that Ψ has full rank. Every entry of Ψ is real analytic as σ is analytic by Assumption 3.1 (note that also the softplus activation function σ_γ is analytic). The set of low rank matrices Ψ can be characterized by a system of equations such that all the $\binom{M}{N}$ determinants of all $N \times N$ sub-matrices of Ψ are zero. As the determinant is a polynomial in the entries of the matrix and thus an analytic function of the entries and composition of analytic functions are again analytic, we conclude that each determinant is an analytic function of U. As shown above, there exists at least one U such that one of these determinant functions is not identically zero and thus by Lemma A.1, the set of U where this determinant is zero has measure zero. But as all submatrices need to have low rank in order that Ψ has low rank, it follows that the set of U where Ψ has low rank has just measure zero. $\square$

C Experiment details

Layer	**Output size**	**#neurons**
Input: 28×28	$28 \times 28 \times 1$	
$3 \times 3\,\text{conv} - 64,\ \ \text{stride}\,1$	$28 \times 28 \times 64$	50176
$3 \times 3\,\text{conv} - 64,\ \ \text{stride}\,1$	$28 \times 28 \times 64$	50176
$3 \times 3\,\text{conv} - 64,\ \ \text{stride}\,2$	$14 \times 14 \times 64$	12544
$3 \times 3\,\text{conv} - 128,\ \text{stride}\,1$	$14 \times 14 \times 128$	25088
$3 \times 3\,\text{conv} - 128,\ \text{stride}\,1$	$14 \times 14 \times 128$	25088
$3 \times 3\,\text{conv} - 128,\ \text{stride}\,2$	$7 \times 7 \times 128$	6272
$3 \times 3\,\text{conv} - 256,\ \text{stride}\,1$	$7 \times 7 \times 256$	12544
$1 \times 1\,\text{conv} - 256,\ \text{stride}\,1$	$7 \times 7 \times 256$	12544
$3 \times 3\,\text{conv} - 256,\ \text{stride}\,2$	$4 \times 4 \times 256$	4096
$3 \times 3\,\text{conv} - 256,\ \text{stride}\,1$	$4 \times 4 \times 256$	4096
$3 \times 3\,\text{conv} - 256,\ \text{stride}\,2$	$2 \times 2 \times 256$	1024
$3 \times 3\,\text{conv} - 256,\ \text{stride}\,1$	$2 \times 2 \times 256$	1024
$3 \times 3\,\text{conv} - 256,\ \text{stride}\,2$	$1 \times 1 \times 256$	256
Fully connected, 10 output units		

Table 3: The architecture of CNN13 for MNIST dataset. There are in total $179,840$ hidden neurons.