Essentially No Barriers in Neural Network Energy Landscape

Felix Draxler [1 2] **Kambis Veschgini** [2] **Manfred Salmhofer** [2] **Fred A. Hamprecht** [1]

Abstract

Training neural networks involves finding minima of a high-dimensional non-convex loss function. Relaxing from linear interpolations, we construct continuous paths between minima of recent neural network architectures on CIFAR10 and CIFAR100. Surprisingly, the paths are essentially flat in both the training and test landscapes. This implies that minima are perhaps best seen as points on a single connected manifold of low loss, rather than as the bottoms of distinct valleys.

1. Introduction

Neural networks have achieved remarkable success in practical applications such as object recognition (He et al., 2016; Huang et al., 2017), machine translation (Bahdanau et al., 2015; Vinyals & Le, 2015), speech recognition (Hinton et al., 2012; Graves et al., 2013; Xiong et al., 2017) etc. Theoretical insights on why neural networks can be trained successfully despite their high-dimensional and non-convex loss functions are few or based on strong assumptions such as the eigenvalues of the Hessian at critical points being random (Dauphin et al., 2014), linear activations (Choromanska et al., 2014; Kawaguchi, 2016) or wide hidden layers (Soudry & Carmon, 2016; Nguyen & Hein, 2017).

In the current literature, minima of the loss function are typically depicted as points at the bottom of a strictly convex valley of a certain width that reflects the generalisation of the network, with network parameters given by the location of the minimum (Keskar et al., 2016). This is also the picture obtained when the loss function of neural networks is visualised in low dimension (Li et al., 2017).

In this work, we conjecture that neural network loss minima are not isolated points in parameter space, but essentially form a connected manifold. More precisely, we argue that the part of the parameter space where the loss remains below a certain low threshold forms one single connected component.

We support the above claim by studying the energy landscape of several ResNets and DenseNets on CIFAR10 and CIFAR100: For random pairs of minima, we construct continuous paths through parameter space for which the loss remains very close to the value found directly at the minima. An example for such a path is shown in Figure 1.

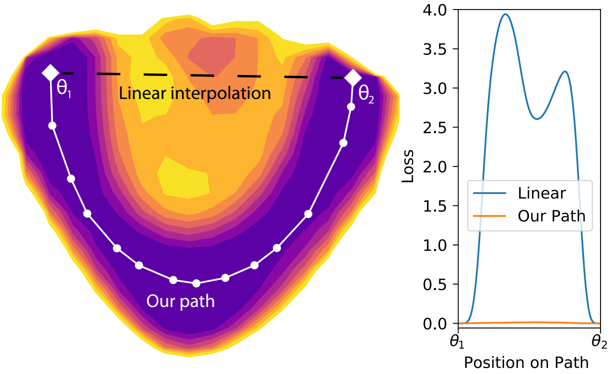

Figure 1. Left: A slice through the one million-dimensional training loss function of DenseNet-40-12 on CIFAR10 and the minimum energy path found by our method. The plane is spanned by the two minima and the mean of the nodes of the path. *Right:* Loss along the linear line segment between minima, and along our high-dimensional path. Surprisingly, the energy along this path is essentially flat.

Our main contribution is the finding of paths

1. that connect minima trained from different initialisations which are not related to each other via known loss-conserving operations like rescaling,
2. along which the training loss remains essentially at the same value as at the minima,
3. along which the test loss remains essentially constant while the test error rate slightly increases.

The existence of such paths suggests that modern neural networks have enough parameters such that they can achieve

[1] Heidelberg Collaboratory for Image Processing (HCI), IWR, Heidelberg University, D-69120 Heidelberg, Germany [2] Institut für Theoretische Physik, Heidelberg University, D-69120 Heidelberg, Germany. Correspondence to: Fred A. Hamprecht <fred.hamprecht@iwr.uni-heidelberg.de>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

good predictions while a big part of the network undergoes structural changes. In closing, we offer qualitative justification of this behaviour that may offer a handle for future theoretical investigation.

2. Related Work

In discussions about why neural networks generalise despite the extremely large number of parameters, one often finds the argument that wide minima generalise better (Keskar et al., 2016). This picture is confirmed when visualising the parameter space on a random plane around a minimum (Li et al., 2017). We draw a completely different image of the loss landscape: Minima are not located in finite-width valleys, but there are paths through the parameter space along which the loss remains very close to the value at the minima. A similar view had previously been conjectured by (Sagun et al., 2017). They find flat linear paths between minima that are close in parameter space by construction. We extend their work by constructing flat paths between arbitrary minima.

It has previously been shown that minima of networks with ReLU activations are degenerate (Dinh et al., 2017): One can scale all parameters in one layer by a constant α and in following layer by α^{-1} without changing the output of the network. Here, we provide evidence for a different kind of degeneracy: We construct paths between independent minima that are essentially flat.

(Freeman & Bruna, 2016) showed that local minima are connected without large barriers for a CNN on MNIST and an RNN on PTB next word prediction. On CIFAR10, they found a more frustrated landscape. We extend their work in two ways: First, we consider ResNets and DenseNets that outperform plain CNNs by a large margin. Second, we apply a state of the art method for connecting minima from molecular statistical mechanics: The Automated Nudged Elastic Band (AutoNEB) algorithm (Kolsbjerg et al., 2016) which in turn is based on the Nudged Elastic Band (NEB) algorithm (Jónsson et al., 1998). We additionally systematically replace paths that contain relatively high loss barriers. Combining the above we find paths with essentially no energy barrier.

NEB has so far been applied to a multi-layer perceptron with a single hidden layer (Ballard et al., 2016). High energy barriers between the minima of network were found when using three hidden neurons, and disappeared upon adding more neurons to the hidden layer. In follow-up work, (Ballard et al., 2017) trained a multi-layer perceptron with a single hidden layer on MNIST. They found that with l^2-regularisation, the landscape had no significant energy barriers. However, for their network they report an error rate of 14.8% which is higher than the 12% achieved even by a linear classifier (LeCun et al., 1998) and the 0.35% achieved with a standard CNN (Ciresan et al., 2011).

In this work, we apply AutoNEB to a nontrivial network for the first time, and make the surprising observation that different minima of state of the art networks on CIFAR10 and CIFAR100 are connected through essentially flat paths.

After submission of this work to the International Machine Learning Conference (ICML) 2018, (Garipov et al., 2018) independently reported that they also constructed paths between neural network minima. They study the loss landscape of several architectures on CIFAR10 and CIFAR100 and report the same surprising observation: minima are connected by paths with constantly low loss.

3. Method

In the following, we use the terms *energy* and *loss* interchangeably.

3.1. Minimum Energy Path

A neural network loss function depends on the architecture, the training set and the network parameters θ. Keeping the former two fixed, we simply write $L(\theta)$ and start with two parameter sets θ_1 and θ_2. In our case, they are minima of the loss function, i.e. they result from training the networks to convergence. The goal is to find the continuous path p^* from θ_1 to θ_2 through parameter space with the lowest maximum loss:

$$p(\theta_1, \theta_2)^* = \underset{p \text{ from } \theta_1 \text{ to } \theta_2}{\arg\min} \left\{ \max_{\theta \in p} L(\theta) \right\}.$$

For this optimisation to be tractable, the loss function must be sufficiently smooth, i.e. contain no jumps along the path. The output and loss of neural networks are continuous functions of the parameters (Montúfar et al., 2014); only the derivative is discontinuous for the case of ReLU activations. However, we cannot give any bounds on how steep the loss function may be. We address this problem by sampling all paths very densely.

Such a lowest path p^* is called the *minimum energy path* (MEP) (Jónsson et al., 1998). We refer to the parameter set with the maximum loss on a path as the "saddle point" of the path because it is a true saddle point of the loss function.

In low-dimensional spaces, it is easy to construct the exact minimum energy path between two minima, for example by using dynamic programming on a densely sampled grid.

This is not possible for present day's neural networks with parameter spaces that have millions of dimensions. We thus must resort to methods that construct an approximation of the MEP between two points using some local heuristics. In particular, we resort to the Automated Nudged Elastic

Band (AutoNEB) algorithm (Kolsbjerg et al., 2016). This method is based on the Nudged Elastic Band (NEB) algorithm (Jónsson et al., 1998).

NEB bends a straight line segment by applying gradient forces until there are no more gradients perpendicular to the path. Then, as for the MEP, the highest point of the resulting path is a critical point. While this critical point is not necessarily the saddle point we were looking for, it gives an upper bound for the energy at the saddle point.

In the following, we present the mechanical model behind and the details of NEB. We then proceed to AutoNEB.

Mechanical Model A chain of $N+2$ pivots (parameter sets) p_i for $i = 0, \ldots, N+1$ is connected via springs of stiffness k. The initial and the final pivots are fixed to the minima to connect, i.e. $p_0 = \theta_1$ and $p_{N+1} = \theta_2$. Using gradient descent, the path that minimises the following energy function is found:

$$E(p) = \sum_{i=1}^{N} L(p_i) + \sum_{i=0}^{N} \frac{1}{2} k \left\| p_{i+1} - p_i \right\|^2 \tag{1}$$

The problem with this energy formulation lies in the choice of the spring constant: If, on the one hand, k is too small, the distances between the pivots become larger in areas with high energy. However, identifying the highest point on the path and its energy is the very goal of the algorithm, so the sampling rate should be high in the high-energy regions. If, on the other hand, k is chosen too large, it becomes energetically advantageous to shorten and hence straighten the path as the spring energy grows quadratically with the total length of the path. This cuts into corners of the loss surface and the resulting path can miss the saddle point.

Nudged Elastic Band Inspired by the above model, (Jónsson et al., 1998) presented the *Nudged Elastic Band* (NEB). For brevity, we directly present the improved version by (Henkelman & Jónsson, 2000). The force resulting from Equation (1) consists of a force derived from the loss and a force originating from the springs:

$$F_i = -\nabla_{p_i} E(p) = F_i^L + F_i^S$$

For NEB, the physical forces are modified, or *nudged*, so that the loss force only acts perpendicularly to the path and the spring force only parallelly to the path (see also Figure 2):

$$F_i^{\text{NEB}} = F_i^L\big|_\perp + F_i^S\big|_\parallel.$$

The direction of the path is defined by the local tangent $\hat{\tau}_i$ to the path. The two forces now read:

$$\begin{aligned} F_i^L\big|_\perp &= -(\nabla L(p_i) - (\nabla L(p_i) \cdot \hat{\tau}_i)\hat{\tau}_i) \\ F_i^S\big|_\parallel &= (F_i^S \cdot \hat{\tau}_i)\hat{\tau}_i \end{aligned} \tag{2}$$

where the spring force opposes unequal distances along the path:

$$F_i^S = -k(\|p_i - p_{i-1}\| - \|p_{i+1} - p_i\|) \tag{3}$$

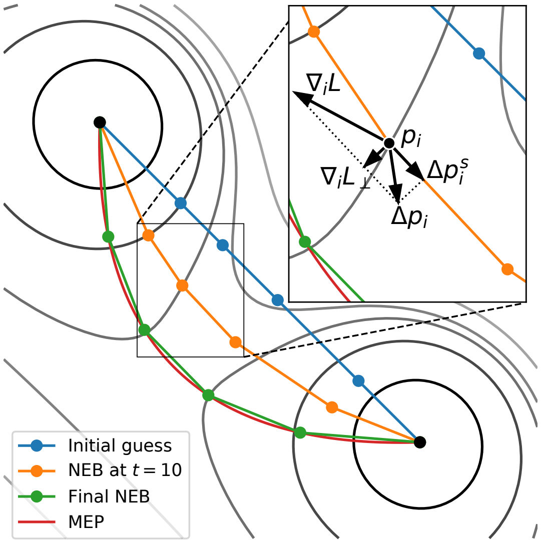

Figure 2. Two dimensional loss surface, with two minima connected by a minimum energy path (MEP) and a nudged elastic band (NEB) at iteration 0, 10 and converged. Construction of NEB update Δp_i for one pivot. The tangent points to the neighbouring pivot with higher energy. Re-distribution Δp_i^s acts parallelly and the loss force $\nabla_i L$ perpendicularly to the tangent.

In this formulation, high energy pivots no longer "slide down" from the saddle point. The spring force only re-distributes pivots on the path, but does not straighten it. Pivots can be spaced unequally by introducing target distances or unequal spring constants into Equation (3).

The local tangent is chosen to point in the direction of one of the adjacent pivots ($\mathcal{N}$ normalises to length one):

$$\hat{\tau}_i = \mathcal{N} \begin{cases} p_{i+1} - p_i & \text{if } L(p_{i+1}) > L(p_{i-1}) \\ p_i - p_{i-1} & \text{else.} \end{cases}$$

This particular choice of $\hat{\tau}$ prevents kinks in the path and ensures a good approximation near the saddle point (Henkelman & Jónsson, 2000).

The above procedure requires the following hyperparameters: The spring stiffness k and number of pivots N.

(Sheppard et al., 2008) claim that a wide range of k leads to the same result on a given loss surface. However, if chosen too large, the optimisation can become unstable. If it is too small, an excessive number of iterations are needed before the pivots become equally distributed. We did not find a value for k that worked well across different loss surfaces and number of pivots N. Instead, we re-distribute the pivots in each iteration t and set the actual spring force to zero. The loss force is still restricted to act parallelly to the path. In the literature, this is sometimes referred to as the *string method* (Sheppard et al., 2008).

Algorithm 1 shows how the initial path is iteratively updated using the above forces. As a companion, Figure 2 visualises the forces in one update step for a two dimensional example. In this formulation, we use gradient descent to update the path. Any other gradient based optimiser can be used. It typically introduces additional hyperparameters, for example a learning rate γ. The number of iterations T should be chosen large enough for the optimisation to converge.

Algorithm 1 NEB

```
Input: initial path $p^{(0)}$ with $N + 2$ pivots,
    $p^{(0)}_0 = \theta_1$ and $p^{(0)}_{N+1} = \theta_2$.
for t = $1, \dots, T$ do
    Redistribute pivots on path $p^{(t-1)}$ and store as $p$.
    for i = $1, \dots, N$ do
        Compute projected loss force $F_i = F^L_i\big|_\perp$.
        Store pivot $p^{(t)}_i = p_i + \gamma F_i$.
    end for
end for
return final path $p^{(T)}$
```

The evaluation time of Algorithm 1 rises linearly with the number of iterations and the number of pivots on the path. Computing the NEB forces can trivially be parallelised over the pivots.

The number of pivots N trades off between computational effort on the one hand and subsampling artefacts on the other hand. In neural networks, it is not known what sampling density is needed for traversing the parameter space. We use an adaptive procedure that inserts more pivots where needed:

AutoNEB The Automated Nudged Elastic Band (AutoNEB, Algorithm 2) wraps the above NEB algorithm (Kolsbjerg et al., 2016). It runs NEB only for a small number of iterations T at a time, initially with a small number of pivots N. It is then checked if the current pivots accurately sample the path. If sampling is not dense enough, new pivots are added at locations where it is estimated that the path requires more accuracy, see Appendix A. This procedure is repeated several times.

3.2. Local minimum energy paths

AutoNEB is not guaranteed to find the true MEP. Instead, it can get stuck in local minimum energy paths (local MEPs) with spuriously high saddle point losses. The good news is that the graph of minima and local MEPs has an ultrametric property: Suppose some local MEPs from a minimum A to B and from B to C are known. We call them p_{AB} and p_{BC}. The respective saddle point energies give an upper bound for the true saddle point energies (marked with an asterisk):

Algorithm 2 AutoNEB

```
Input: Minima to connect $\theta_1, \theta_2$.
Initialise $N$ pivots equally spaced on line segment
$(\theta_1, \theta_2)$.
for $t' = 1, \dots, T'$ do
    Optimise path using NEB (see Algorithm 1).
    Evaluate loss along NEB.
    Insert pivots where residuum is large.
end for
return path after final iteration.
```

$$L^*_{AB} \leq L_{AB} = \max_{\theta \in p_{AB}} L(\theta)$$
$$L^*_{BC} \leq L_{BC} = \max_{\theta \in p_{BC}} L(\theta)$$

The concatenation of the two paths yields an upper bound for the true saddle point energy between A and C (ultrametric triangle inequality):

$$L^*_{AC} \leq \max\{L_{AB}, L_{BC}\}$$

Proof. Concatenating the paths p_{AB} and p_{BC} gives a new path p_{AC} connecting A to C. The saddle point is located at the maximum loss along a path and hence the saddle point energy of p_{AC} is $L_{AC} = \max\{L_{AB}, L_{BC}\}$. □

This has three consequences:

1. As soon as the minima and computed local MEPs form one connected graph, upper bounds for all saddle energies are available. We can hence very quickly get upper bounds for all pairs of minima by connecting one minimum to all others.

2. When AutoNEB finds a bad local MEP, this can be addressed by computing paths between other pairs of minima. As soon as a lower path is found by concatenating other paths, the bad local MEP can be removed. This means that the bad local paths can easily be corrected for.

3. When we evaluate the saddle point energies of a set of computed local MEPs, we can ignore paths with higher energy than the concatenation of paths with a lower maximal energy.
   These lowest local MEPs form a minimum spanning tree in the available graph (Gower & Ross, 1969). A

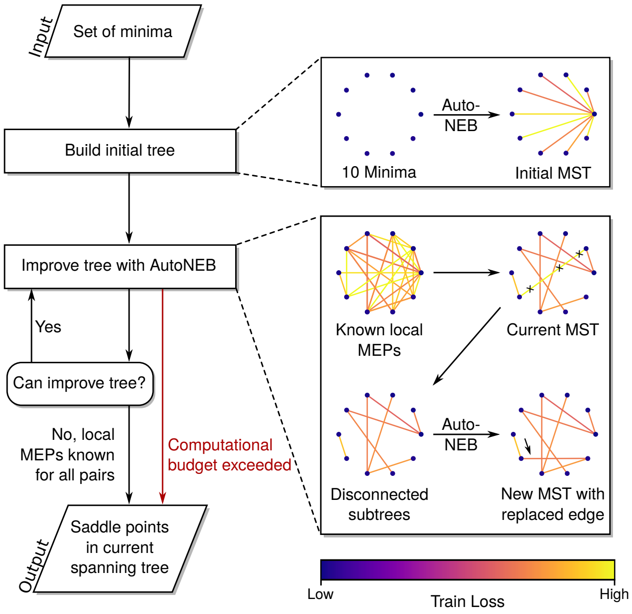

Figure 3. Overview over Algorithm 3 and examples for the first nine iterations and some later iteration. First, all minima are connected to one particular minimum. Then, AutoNEB computes new local MEPs to circumvent the worst local MEPs in the minimum spanning tree. This is repeated until local MEPs are known between all pairs of minima or the procedure is stopped early. Whenever the algorithm stops, an upper bound for each pair of minima is available via the minimum spanning tree.

> Minimum Spanning Tree (MST) can be found efficiently, e.g. using Kruskal's algorithm.

We resort to a heuristic (Figure 3, Algorithm 3) to systematically sample edge costs from a latent graph to find or approximate its MST. Since running AutoNEB is computationally expensive (comparable to training the corresponding network once), we stop the iteration when the lightened spanning tree found so far contains only similar saddle point energies.

4. Experiments

We connect minima of different CNNs, ResNets (He et al., 2016) and DenseNets (Huang et al., 2017) on the image classification tasks CIFAR10 and CIFAR100 (Krizhevsky & Hinton, 2009) using AutoNEB. Per architecture, we consider ten minima.[1]

The minima are constructed from multiple random initialisations and are truly distinct: On the test data, the set of misclassified images differs between the minima. More precisely, on the ResNet and DenseNet architectures, we observe a maximum 70% overlap of the samples that are misclassified at two minima, proving their distinctiveness.

Algorithm 3 Energy Landscape Exploration

```
**Input:** set of minima $\theta_i$.
Connect $\theta_1$ to all $\theta_i, i \neq 1$, yielding a spanning tree.
**repeat**
  Remove edge $p_o$ with highest loss from spanning tree.
  From each resulting tree, try to select one minimum,
    so that no local MEP is known for the pair.
  **if** search failed **then**
    Re-insert $p_o$ and ignore it when searching for the
      highest edge in the future.
  **else**
    Compute new path $p_n$ using AutoNEB.
    **if** $L_{p_n} < L_{p_o}$ **then**
      Add $p_n$ to the tree, making tree "lighter".
    **else**
      Re-insert $p_o$ to the tree (no better path was found).
    **end if**
  **end if**
**until** one local MEP is known for each pair of minima
  or computational budget is exceeded.
**return** saddle points in minimum spanning tree.
```

We report the average cross-entropy loss and misclassification rates over the full training and test data for the minima found. For the final evaluation, we reduce the saddle points to the minimum spanning tree with the saddle training loss as weight.

4.1. AutoNEB schedule

The set-up is identical for all network architectures, except for the batch sizes which we note in each case.

The minimum pairs to connect are ordered by Algorithm 3. For each minimum pair, AutoNEB (see Algorithm 2) is run for a total of 14 cycles of NEB. The loss is evaluated for each pivot on a random batch.

After each cycle, new pivots are inserted at positions where the loss exceeds the energy estimated by linear interpolation between pivots by at least 20% compared to the total energy difference along the path. Comparing to the total loss difference prioritises big errors which is beneficial as each additional pivot implies one more loss evaluations per iteration. The energy is evaluated on nine points between each pair of neighbouring pivots.

As optimiser, we use SGD with momentum 0.9 and l^2-regularisation with $\lambda = 0.0001$.

The NEB cycles are configured with a learning rate decay:

1. Four cycles of 1000 steps each with learning rate 0.1.

[1] Source code is available at https://github.com/fdraxler/PyTorch-AutoNEB.

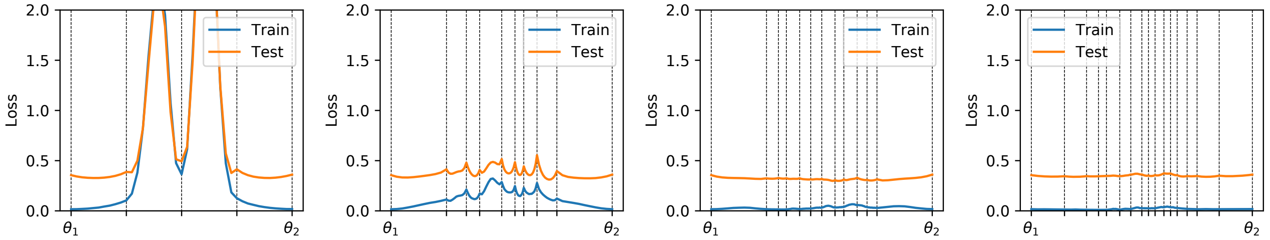

Figure 4. Typical snapshots of the loss along a path connecting two minima, as pivots are inserted into a nudged elastic band: (1) After the first cycle, typically one or two corners are cut. New pivots are inserted at high loss values, here between the second and the third pivot. (2) After four cycles of high learning rate, the highest loss on the path is reduced by a factor of five. Between pivots we find low energy regions that we attribute to the high learning rate of 0.1. (3) The first round with low learning rate of 0.01 reduces the energy by another factor of two. (4) After 14 cycles, no major energy bumps exist between the pivots, the procedure is converged. Results shown are for a ResNet-20 on CIFAR10.

2. Two cycles with 2000 steps and learning rate 0.1. The number of steps was increased as it did not prove necessary inserting new pivots after 1000 steps.

3. Four cycles of 1000 steps with learning rate 0.01. The loss drops significantly in this phase.

4. No big improvement was seen in the last four cycles of 1000 steps each with a learning rate of 0.001.

Figure 4 shows typical snapshots of the loss-along-path between the above cycles.

4.2. Architectures

We consider a wide range of architectures, from shallow CNNs to recent deep networks with skip connections.

Basic CNN We analyse CNNs without skip connections with a variety of depths and widths on both CIFAR10 and CIFAR100. We name them "CNN-$W{\times}D$" where W corresponds to the width of each layer (number of channels) and D to the number of convolutional layers. Each convolution is 5×5, a max pooling layer of 2 is attached to each convolution, and a single hidden fully connected layer of width 256 and batch normalisation (Ioffe & Szegedy, 2015) are used. We consider the one-layer CNN-12×1, CNN-24×1, CNN-36×1, CNN-48×1 and CNN-96×1, and the multi-layer CNN-48×2 and CNN-48×3.

ResNet We train ResNets on both CIFAR10 and CIFAR100 (ResNet-20, -32, -44 and -56) following the training procedure in (He et al., 2016). For ResNet-20 and ResNet-32, the best local MEPs were found using a batch size of 512 training samples. For ResNet-44 and ResNet-56, this number was decreased to 256.

DenseNet We train a DenseNet-40-12 and a DenseNet-100-12-BC on both CIFAR10 and CIFAR100 following the training procedure in (Huang et al., 2017). The AutoNEB batch size was set to 256.

4.3. Saddle point losses

The saddle point losses for both training and test sets found by AutoNEB are shown in Figure 5, and listed in detail in Table B.1 in Appendix B. They are small for the shallow networks and almost negligible for the deep residual networks.

Compare the saddle point loss to the loss at the minima on the training and on the test set. For the shallow CNNs on the one hand, the saddle loss is found quite close to the test loss. On the other hand, the saddle loss of the ResNets and DenseNets lies very close to the training loss.

Further, we measure how late during training the learning curve crosses the saddle loss, as visualised in Figure 6. The learning curve falls below the saddle point energy only after the first learning rate decay and an additional significant drop of the loss for all architectures. For the wider CNNs on CIFAR10 and the majority of ResNets and DenseNets, the losses meet even after the second decay, i.e. in the final phase of learning.

We observe the following trend: The *deeper* and *wider* an architecture, the *lower* are the saddles between minima until they essentially *vanish for current-day deep architectures*. The more complex dataset CIFAR100 raises the barriers.

At the same time, the test accuracy is preserved: The classification error only increases slightly by maximally 0.5% (2.2%) for all deep architectures on CIFAR10 (CIFAR100) compared to the minima.

We conclude that the saddle points have surprisingly low loss with respect to the metrics above. In other words, there are essentially no loss barriers in current-day deep architectures.

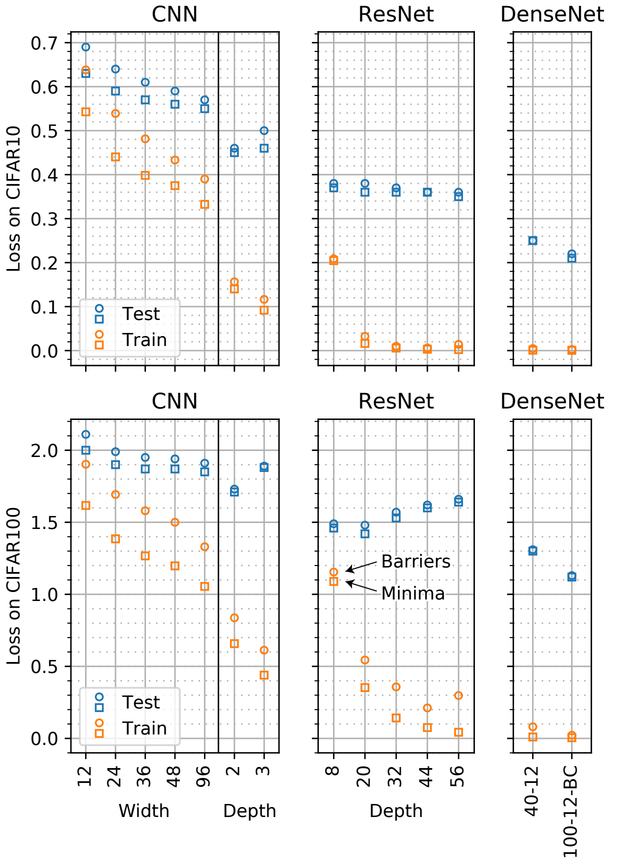

Figure 5. Core result: Minimum (■) and saddle point (●) losses on training and test set. The training (orange) saddle losses are almost identical to the minima on CIFAR10 on the deep architectures (middle and right). On CIFAR100, a small gap remains that vanishes in comparison to the loss of a randomly initialised network. On a simple architecture (left), increasing width and especially depth closes the gap between the barriers and the minima. On the test set (blue), the points of the minima and the saddle points are very close. Error bars are omitted since they would be smaller than the markers.

4.4. Properties of obtained local MEPs

The local MEPs between the minima not only have very low loss, they also follow simple trajectories. Figure 7 shows some coordinates of two local MEPs in a parallel coordinate plot. We find that each coordinate has a smooth path. The largest deviations occur near the saddle point of the path. The paths are between 50% to 2.5 times longer than the direct connection between the minima.

5. Discussion

We have pointed out an intriguing property of the loss surface of current-day deep networks, by upper-bounding the saddle points between the parameter sets that result from stochastic gradient descent, a.k.a. “minima”. These empirical upper bounds are astonishingly close to the loss at the minima themselves. The experiments on the CNNs suggest that the disappearance of barriers emerges as the networks get wider and especially deeper. At this point, we cannot give a formal characterization of the regime in which this finding holds. A formal proof is also complicated by the fact that the loss surface is a function not only of the parameters and the architecture, but also of the training set; and the distribution of real-world structured data such as images or sentences does not lend itself to a compact mathematical representation. That said, we want to make two related arguments that may help explain why we observe no substantial barrier between minima.

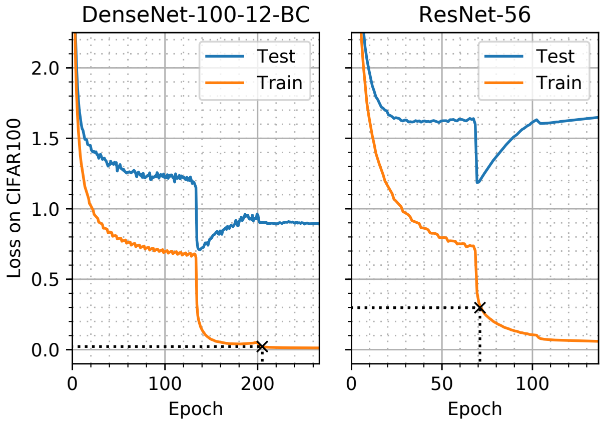

Figure 6. Learning curves illustrating how late in training the loss drops below the saddles. The training loss passes the mean saddle point energy at about 205 (72) epochs or 78% (52%) of training. Among all deep architectures considered, the average saddle point crosses the training loss last for DenseNet-100-12-BC and earliest for ResNet-56, both on CIFAR100. The crossing for the DenseNet had to be identified in a log plot.

5.1. Resilience

State of the art neural networks have dozens or hundreds of neurons / channels per layer, and skip connections between non-adjacent layers. Assume that by training, a parameter set with low loss has been identified. Now if we perturb a single parameter, say by adding a small constant, but leave the others free to adapt to this change to still minimise the loss, it may be argued that by adjusting somewhat, the myriad other parameters can “make up” for the change imposed on only one of them. After this relaxation, the procedure and argument can be repeated, though possibly with the perturbation of a different parameter.

This type of resilience is exploited and encouraged by procedures such as Dropout (Srivastava et al., 2014) or ensembling (Hansen & Salamon, 1990). It is also the reason why neural networks can be greatly condensed before a substantial increase in loss occurs (Liu et al., 2017).

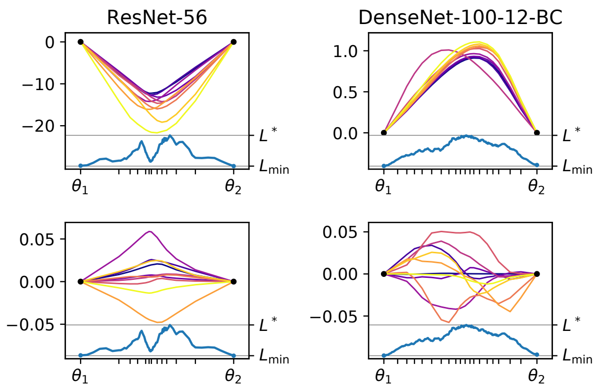

Figure 7. Network parameters follow a smooth trajectory along the MEPs. For each coordinate, the distance to the linear interpolation is shown (negative means closer to origin). The largest deviations from the linear path occur near the saddle point. The upper plots show those ten coordinates that deviate most from the linear interpolation. The ten coordinates in the lower row are chosen randomly. Also shown is the training loss along the two paths. The dataset is CIFAR100.

5.2. Redundancy

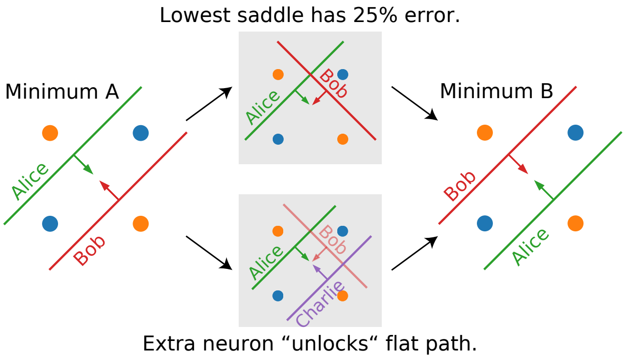
Lowest saddle has 25% error.
Minimum A
Minimum B
Alice
Bob
Charlie
Extra neuron “unlocks“ flat path.

Figure 8. Network capacity for XOR dataset: The continuous transition from one minimum *(left)* to another minimum *(right)* is not possible without misclassifying at least one instance *(upper middle)*. *(Lower middle)* Adding one helper neuron makes the transition possible while always predicting the right class for all data points, i.e. by turning off the outgoing weight of Bob.

Consider the textbook example of a two-layer perceptron that can fit the XOR problem. The two neurons traditionally used in the first hidden layer – let's call them Alice and Bob – are shown in Figure 8 on the left. We can obtain an equivalent network by exchanging Alice and Bob (and permuting the weights of the neuron in the second hidden layer, not shown). This network, also corresponding to a minimum of the loss surface, is shown in Figure 8 on the right. Now, any path between these two minima will entail parameter sets such as the one in the upper centre of Figure 8 that incur high loss.

If, on the other hand, we introduce an auxiliary neuron, Charlie, we can play a small choreography: Enter Charlie. Charlie stands in for Bob. Bob transitions to Alice's role via Figure 8, lower centre. Alice takes over from Charlie. Exit Charlie. If the neuron in the second hidden layer adjusts its weights so as to disregard the output from the neuron-in-transition, the entire network incurs no higher loss than at the two original minima.

We have constructed a perfect minimum energy path through increasing the *width*. Similarly, it is possible to construct a zero-loss path by adding a second two-neuron layer to the network, that is by increasing the *depth* of the network.

6. Conclusion

We find that the loss surface of deep neural networks contains paths with constantly low loss. The paths connect the minima so that they form one single connected component in the loss landscape. The barriers are especially low with increasing depth and width.

We put forth two closely related explanations in the above. Both hold only if the network has some extra capacity, or degrees of freedom, to spare. Empirically, this seems to be the case for modern-day architectures applied to standard problems.

This has the profound implication that low Hessian eigenvalues exist apart from the eigenvectors with analytically zero eigenvalues due to scaling.

We introduce AutoNEB for the characterisation of current-day architectures for the first time. The method opens the door to further empirical research on the energy landscape of neural networks. When the hyperparameters of AutoNEB are further refined, we expect to find even lower paths up to the level where the true saddle points are recovered. It is then interesting to see if certain minima have a higher barrier between them than others. This makes it possible to recursively form clusters of minima, i.e. using single-linkage clustering. In the traditional energy landscape literature, this kind of clustering is summarised in disconnectivity graphs (Wales et al., 1998) which can help visualise very high-dimensional surfaces.

On the practical side, we envisage using the resulting paths as a large ensemble of neural networks (Garipov et al., 2018), especially given that we observe marginally lower test loss along the path.

More importantly, we hope these observations will stimulate new theoretical work to better understand the nature of the loss surface, and why local optimisation on such surfaces results in networks that generalize so well.

Acknowledgements

FAH gratefully acknowledges support by DFG under grant no. HA 4364/9-1.

References

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

Ballard, A. J., Stevenson, J. D., Das, R., and Wales, D. J. Energy landscapes for a machine learning application to series data. *J. Chem. Phys.*, 144(12):124119, Mar 2016. ISSN 1089-7690. doi: 10.1063/1.4944672. URL http://dx.doi.org/10.1063/1.4944672.

Ballard, A. J., Das, R., Martiniani, S., Mehta, D., Sagun, L., Stevenson, J. D., and Wales, D. J. Energy landscapes for machine learning. *Physical Chemistry Chemical Physics (Incorporating Faraday Transactions)*, 19:12585–12603, 2017. doi: 10.1039/C7CP01108C.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. The loss surface of multilayer networks. *CoRR*, abs/1412.0233, 2014. URL http://arxiv.org/abs/1412.0233.

Ciresan, D. C., Meier, U., Masci, J., Maria Gambardella, L., and Schmidhuber, J. Flexible, high performance convolutional neural networks for image classification. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, pp. 1237. Barcelona, Spain, 2011.

Dauphin, Y., Pascanu, R., Gülçehre, Ç., Cho, K., Ganguli, S., and Bengio, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *CoRR*, abs/1406.2572, June 2014. URL http://arxiv.org/abs/1406.2572.

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1019–1028, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/dinh17b.html.

Freeman, C. D. and Bruna, J. Topology and Geometry of Half-Rectified Network Optimization. *ArXiv e-prints*, November 2016. URL http://arxiv.org/abs/1611.01540.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs. *ArXiv e-prints*, February 2018. URL http://arxiv.org/abs/1802.10026.

Gower, J. C. and Ross, G. J. S. Minimum spanning trees and single linkage cluster analysis. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 18(1): 54–64, 1969. ISSN 00359254, 14679876. URL http://www.jstor.org/stable/2346439.

Graves, A., Mohamed, A.-r., and Hinton, G. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pp. 6645–6649. IEEE, 2013.

Hansen, L. K. and Salamon, P. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Henkelman, G. and Jónsson, H. Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points. *The Journal of chemical physics*, 113(22):9978–9985, 2000.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

Huang, G., Liu, Z., Weinberger, K. Q., and van der Maaten, L. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, pp. 3, 2017.

Ioffe, S. and Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ArXiv e-prints*, February 2015.

Jónsson, H., Mills, G., and Jacobsen, K. W. Nudged elastic band method for finding minimum energy paths of transitions. In *Classical and quantum dynamics in condensed phase simulations*, pp. 385–404. World Scientific, 1998.

Kawaguchi, K. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pp. 586–594, 2016.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, September 2016. URL http://arxiv.org/abs/1609.04836.

Kolsbjerg, E. L., Groves, M. N., and Hammer, B. An automated nudged elastic band method. *The Journal of chemical physics*, 145(9):094107, 2016.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. 2009.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, H., Xu, Z., Taylor, G., and Goldstein, T. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2736–2744, 2017.

Montúfar, G., Pascanu, R., Cho, K., and Bengio, Y. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pp. 2924–2932, 2014.

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

Sagun, L., Evci, U., Ugur Guney, V., Dauphin, Y., and Bottou, L. Empirical Analysis of the Hessian of Over-Parametrized Neural Networks. *ArXiv e-prints*, June 2017. URL http://arxiv.org/abs/1706.04454.

Sheppard, D., Terrell, R., and Henkelman, G. Optimization methods for finding minimum energy paths. *The Journal of Chemical Physics*, 128(13):134106, Apr 2008. ISSN 1089-7690. doi: 10.1063/1.2841941. URL http://dx.doi.org/10.1063/1.2841941.

Soudry, D. and Carmon, Y. No bad local minima: Data independent training error guarantees for multilayer neural networks. *ArXiv e-prints*, May 2016. URL http://arxiv.org/abs/1605.08361.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Vinyals, O. and Le, Q. V. A neural conversational model. *CoRR*, abs/1506.05869, 2015. URL http://arxiv.org/abs/1506.05869.

Wales, D. J., Miller, M. A., and Walsh, T. R. Archetypal energy landscapes. *Nature*, 394(6695):758, 1998.

Xiong, W., Droppo, J., Huang, X., Seide, F., Seltzer, M. L., Stolcke, A., Yu, D., and Zweig, G. Toward human parity in conversational speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(12):2410–2423, Dec 2017. ISSN 2329-9290. doi: 10.1109/TASLP.2017.2756440.

Appendices

A. AutoNEB insertion

New pivots are inserted at locations where the loss values resulting from evaluations rise higher than a certain threshold above the the estimate given the adjacent pivots, see Figure A.1. More formally, the loss curve between pivots i and $i+1$ can be approximated by interpolating the pivot values:

$$L_{\text{guess}}^{i,i+1}(\alpha) = L(p_i)(1-\alpha) + L(p_{i+1})\alpha$$

where $\alpha \in [0,1]$ interpolates between the pivots. The true loss value a the same position is:

$$L^{i,i+1}(\alpha) = L\big(p_i(1-\alpha) + p_{i+1}\alpha\big).$$

Denote the difference as:

$$\Delta L^{i,i+1}(\alpha) = L^{i,i+1}(\alpha) - L_{\text{guess}}^{i,i+1}(\alpha)$$

If the true loss rises too high above the estimated loss, a new pivot should be inserted. It is beneficial to first insert pivots at the highest residuum: Each new pivot requires more expensive gradient evaluations. The deviation between true loss and estimate is evaluated at discrete positions $\alpha \in \mathcal{A} \subset (0,1)$, $|\mathcal{A}| < \infty$. The differences $\Delta L^{i,i+1}(\alpha)$ are normalised to the range of values of $L(p_i)$. When this normalised deviation rises above a certain threshold ϑ, a new pivot is inserted, i.e. insert a pivot between i and $i+1$ when:

$$\vartheta > \frac{\Delta L^{i,i+1}(\alpha)}{\max_i L(p_i) - \min_i L(p_i)} =: \Delta l^{i,i+1}(\alpha) \qquad (4)$$

Only one pivot is inserted per line segment per AutoNEB iteration. If several α for one line segment fulfil the above condition, only the position with the highest residuum is chosen. Additionally, the total number of pivots to insert per iteration is limited so that the highest deviations are prioritised.

B. Quantitative minimum and saddle point losses

Table B.1 shows the full list of results. Here, we compare the numbers in detail to characteristic metrics of a neural network, with error margins below 10% for all energy and error rate values:

The *loss of an untrained network* amounts to $-\log(0.1) = 2.3$ on CIFAR10 and $-\log(0.01) = 4.6$ on CIFAR100.

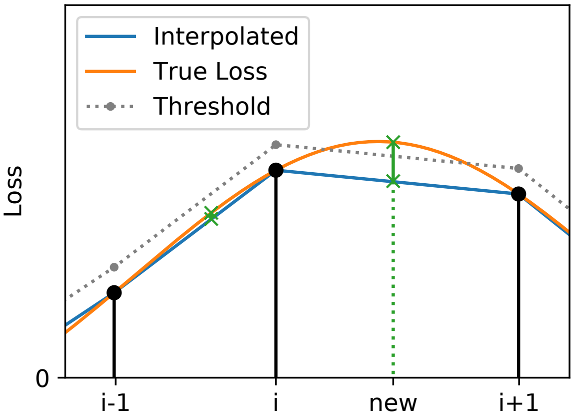

Figure A.1. New items are inserted in each cycle of AutoNEB when the true energy at an interpolated position between two points rises too high compared to the interpolated energy. Between i and $i+1$, a new pivot is inserted. Between $i-1$ and i, the difference is small enough that no additional pivot is needed.

The saddle point energies are between $1/3$ to $1/20$ of the initial loss for the shallow networks (all CNNs and ResNet-8) and *about two orders of magnitude smaller* for the deep residual networks.

The *test loss* at the saddle points of the smallest one-layer CNNs comes close to the test loss of the minima. The wider and especially deeper the CNN, the closer the saddle loss approaches the minima. The saddle point energies of the deep residual networks (not ResNet-8) on CIFAR10 are about one order of magnitude smaller than the average minimum loss on the test set. On CIFAR100, the saddle point energies of the ResNets are smaller than a third of the value on the test set. For the DenseNets, they are *at least one order of magnitude smaller*.

The *training loss* at the saddle points is at most eight times as large as the training loss of the minima. This ratio, reported as “factor” in Table B.1, is hard to interpret directly as it can approach zero when the network fits the data perfectly. Instead, we report that all saddle losses are closer to the training than the test loss for all but the smallest three basic CNNs. For all deeper architectures, the saddle loss is *much closer to the training than to the test loss*, see Figure 5.

The *classification performance* does not decrease significantly along the paths. On the ResNets, the error rises by maximally 0.7% on CIFAR10 and 2.9% on CIFAR100. For the DenseNets, the error rises by up to 0.4% on CIFAR10 and 1.5% on CIFAR100. These *differences are small* compared to the error rate at the minima.

Table B.1. Quantitative results. “Min.” denotes the average value at the minima. For the saddle point values (“Sadd.”), the maximum value of each metric along the local MEPs is computed and the results are averaged. The “epoch” is measured at the point where the loss falls below the saddle point loss for the first time. It is noted in **bold** if it belongs to the third part of training with learning rate $\gamma = 10^{-3}$. Basic CNNs and ResNets are trained for 136 epochs, DenseNets for 266 epochs. The “factor” is the ratio between average saddle point and minima loss. The standard deviations of all values at the minima and saddle are smaller than 10% when averaged over the instances or over the mini-batches.

		Train energy				Test energy		Test error rate [%]		
		Min.	Sadd.	Factor	Epoch	Min.	Sadd.	Min.	Sadd.	Δ
Dataset	Architecture									
C10+	CNN-12	0.5428	0.6381	1.2	78	0.63	0.69	21.4	23.6	2.2
	CNN-24	0.4403	0.5390	1.2	84	0.59	0.64	19.8	21.8	2.0
	CNN-36	0.3982	0.4814	1.2	91	0.57	0.61	19.0	20.8	1.8
	CNN-48	0.3750	0.4331	1.2	**103**	0.56	0.59	18.6	19.9	1.2
	CNN-96	0.3324	0.3900	1.2	**103**	0.55	0.57	17.9	19.3	1.4
	CNN-48x2	0.1402	0.1564	1.1	**136**	0.45	0.46	13.4	14.2	0.8
	CNN-48x3	0.0918	0.1164	1.3	**110**	0.46	0.50	13.4	15.2	1.7
	ResNet-8	0.2045	0.2086	1.0	**136**	0.37	0.38	12.0	12.5	0.4
	ResNet-20	0.0162	0.0324	2.0	**104**	0.36	0.38	8.5	8.9	0.5
	ResNet-32	0.0057	0.0097	1.7	**107**	0.36	0.37	7.5	7.8	0.3
	ResNet-44	0.0031	0.0060	1.9	**122**	0.36	0.36	7.1	7.4	0.3
	ResNet-56	0.0022	0.0141	6.5	85	0.35	0.36	6.9	7.4	0.5
	DenseNet-40-12	0.0008	0.0046	5.9	**205**	0.25	0.25	5.6	6.0	0.3
	DenseNet-100-12-BC	0.0005	0.0026	4.8	**205**	0.21	0.22	4.9	5.1	0.2
C100+	CNN-12	1.6167	1.9029	1.2	69	2.00	2.11	51.0	54.2	3.2
	CNN-24	1.3854	1.6930	1.2	70	1.90	1.99	48.2	51.5	3.3
	CNN-36	1.2670	1.5801	1.2	71	1.87	1.95	47.2	50.2	3.0
	CNN-48	1.1977	1.5002	1.3	73	1.87	1.94	46.6	49.5	2.9
	CNN-96	1.0549	1.3304	1.3	82	1.85	1.91	45.6	48.4	2.8
	CNN-48x2	0.6579	0.8372	1.3	91	1.71	1.73	41.0	42.6	1.6
	CNN-48x3	0.4393	0.6124	1.4	91	1.88	1.89	43.7	45.5	1.8
	ResNet-8	1.0894	1.1547	1.1	**103**	1.46	1.49	39.6	40.6	1.0
	ResNet-20	0.3528	0.5442	1.5	79	1.42	1.48	33.3	34.7	1.4
	ResNet-32	0.1422	0.3576	2.5	77	1.53	1.57	31.5	33.7	2.2
	ResNet-44	0.0753	0.2117	2.8	85	1.60	1.62	30.8	32.5	1.7
	ResNet-56	0.0428	0.2978	7.0	71	1.64	1.66	30.3	32.4	2.0
	DenseNet-40-12	0.0101	0.0808	8.0	166	1.30	1.31	26.3	27.7	1.4
	DenseNet-100-12-BC	0.0050	0.0223	4.4	**205**	1.12	1.13	23.7	24.6	0.8