Deep Networks on Toroids: Removing Symmetries Reveals the Structure of Flat Regions in the Landscape Geometry

Fabrizio Pittorino [1] **Antonio Ferraro** [1] **Gabriele Perugini** [1 2] **Christoph Feinauer** [1] **Carlo Baldassi** [1] **Riccardo Zecchina** [1]

Abstract

We systematize the approach to the investigation of deep neural network landscapes by basing it on the geometry of the space of implemented functions rather than the space of parameters. Grouping classifiers into equivalence classes, we develop a standardized parameterization in which all symmetries are removed, resulting in a toroidal topology. On this space, we explore the error landscape rather than the loss. This lets us derive a meaningful notion of the flatness of minimizers and of the geodesic paths connecting them. Using different optimization algorithms that sample minimizers with different flatness we study the mode connectivity and relative distances. Testing a variety of state-of-the-art architectures and benchmark datasets, we confirm the correlation between flatness and generalization performance; we further show that in function space flatter minima are closer to each other and that the barriers along the geodesics connecting them are small. We also find that minimizers found by variants of gradient descent can be connected by zero-error paths composed of two straight lines in parameter space, i.e. polygonal chains with a single bend. We observe similar qualitative results in neural networks with binary weights and activations, providing one of the first results concerning the connectivity in this setting. Our results hinge on symmetry removal, and are in remarkable agreement with the rich phenomenology described by some recent analytical studies performed on simple shallow models.

[1]AI Lab, Institute for Data Science and Analytics, Bocconi University, 20136 Milano, Italy [2]Dept. of Applied Science and Technology, Politecnico di Torino, 10129 Torino, Italy. Correspondence to: Fabrizio Pittorino <fabrizio.pittorino@unibocconi.it>, Carlo Baldassi <carlo.baldassi@unibocconi.it>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

1. Introduction

The loss landscape of a typical deep neural network performing a supervised learning task is in general highly non-convex. Moreover, even small networks (by the current standards) have a huge number of configurations of small loss, corresponding to zero or near-zero training error. In this sense, most modern networks operate in a strongly over-parameterized regime. Understanding how simple variants of first-order algorithms are able to escape bad local minima and yet avoid overfitting is a fundamental problem, which has received a lot of attention from several perspectives (Belkin et al., 2019; Rocks & Mehta, 2020).

A natural and promising approach for addressing this issue is to investigate the geometrical properties of the loss landscape. Broadly speaking, there are two related but conceptually distinct main research directions in this area: one is about the dynamics of gradient-based learning algorithms (e.g. Feng & Tu (2021)); the other concerns a static description of the geometry, its overall structure and its relation to the generalization properties of the network on unseen data (e.g. Gotmare et al. (2018)). In this paper, we focus on the latter.

A first basic observation is that (near-)minimizers of the loss, corresponding to (near-)zero training error, can have dramatically different generalization properties (Keskar et al., 2016; Liu et al., 2020; Pittorino et al., 2021). A growing amount of evidence shows a consistent correlation between the flatness of the minima of the loss and the test accuracy, across a large number of models and with several alternative measures of flatness, see e.g. Dziugaite & Roy (2017); Jiang* et al. (2020); Pittorino et al. (2021); Yue et al. (2020). Moreover, several studies indicate that stochastic gradient descent (SGD) and its variants introduce a bias, compared to full-batch gradient descent, towards flatter minima (Keskar et al., 2016; Chaudhari & Soatto, 2018; Feng & Tu, 2021; Pittorino et al., 2021). This effect seems to be amplified by other operating procedures, e.g. the use of the cross-entropy loss function, drop-out, judicious initialization, ReLU transfer functions (Baldassi et al., 2018; 2020; 2019; Liu et al., 2020; Zhang et al., 2021). Therefore, in practical applications, bad minima are seldom reported or observed, even

though they exist in the landscape.

In this paper, we present a coherent empirical exploration of the structure of the minima of the landscape. Our work has two main features that, taken together, sets it apart from the majority of existing literature (see also sec. 2 on related work): 1) The main object of our study is the (train) error (also called "energy") landscape, rather than the landscape given by the objective loss with which networks are optimized; 2) Our underlying geometrical space and topology is that of networks, intended as functional relations, rather than the space of parameters.

The first point is less crucial to our results, although it affects the second one. Providing a detailed description of the dynamics requires studying the train loss (usually the cross-entropy) landscape. However, we argue that the train error landscape is a similar but more basic object of study for a static analysis, since it is directly related to the observable behavior of the network, especially the generalization error. This is assuming that the end goal of the training is to obtain a classifier whose output is the argmax over the last layer. The objective loss function on the other hand uses the entire output of the layer, which is not normally needed after the training is complete. The train error landscape is also more amenable to theoretical analysis, e.g. Baldassi et al. (2015); Dziugaite & Roy (2017).

The second point is inspired from similar considerations. We posit that, after training, two networks that implement the same input-output relation (not only on the training set, but on any input) must be identified, even if their parameterization differs, and as such we group them into equivalence classes. In order to define a topology and a metric over this space, we standardize the parameterization of the networks (not during the training, but only for the geometrical description), thereby removing the symmetries that affect the usual parameterizations. In most common networks there are two of them: a continuous one, a scale invariance that allows to renormalize the weights, and a discrete one, a permutation symmetry that reflects the fact that in a hidden layer the labels of units (or the labels of filters in convolutional layers) can be exchanged. The latter in particular means that two networks may appear to be very distant from each other in parameter space even if they are very similar or even identical in the function they implement. For example, this could happen if we measure the Euclidean distance between the parameters of two networks, where one is identical to the other up to a permutation of the hidden units within the layers. The scale symmetry can also have this effect, and moreover it may affect many measures of flatness making them imprecise at best and misleading at worst (Dinh et al., 2017). These problems obviously affect every other investigation, such as studying the paths that join two configurations.

Our approach is thus as follows: 1) we choose a normalization method that leaves the network behavior unchanged while fixing the norm of each of the hidden units, projecting them onto a hyper-toroidal manifold; 2) when we compare two networks, we normalize and align them first, and consider the geodesic paths between them in normalized space. With respect to the original goal of exploring function space, this approach is approximate, mainly for computational reasons (all our procedures are efficient and have polynomial running time with the number of parameters), and because in our characterization of the landscape we neglect the biases and batch-norm parameters. Yet, the approximation appears to be very good and the effect is critical. We have applied these techniques to several networks (continuous and discrete[1], multi-layer perceptrons and convolutional networks) and datasets (both real-world and synthetic). In each case, we used different training protocols aimed at sampling zero-error configurations (also called solutions) with different characteristics. In particular, we compared the kind of solutions found by SGD or its variants with momentum, with the (typically flatter and more accurate ones) found by the Replicated-SGD (RSGD) algorithm (Pittorino et al., 2021), and with some poorly-generalizing solutions found by adversarial initialization (Liu et al., 2020). With these, we could explore the local landscape of each solution and the paths connecting any two of them.

Remarkably, our results display some qualitative features that are shared by all networks and datasets, but which are visible and stable only in the space of networks as described above, and not in that of their parameters. Besides confirming that flatter minima generalize better than sharper ones, we found that they are also closer to each other in parameter space[2] (with respect to the Euclidean distance for networks with continuous weights and with respect to the Hamming distance for networks with binary weights), and that geodesic paths between them encounter lower barriers. Also, with few exceptions, the solutions we find can be connected by paths of (near-)zero error with a single bend. Overall, our results are compatible with the analysis of the geometry of the space of solutions in binary, shallow networks reported in Baldassi et al. (2021a;b), according to which efficient algorithms target large connected structures of solutions with the more robust (flatter) ones at the center, radiating out into progressively sharper ones.

2. Related Work

Early works relating flatness and generalization performance are Hochreiter & Schmidhuber (1997); Hinton &

[1] For the discrete networks that we consider the rescaling symmetry is substituted with another discrete (sign-reversal) symmetry.

[2] See Fig. 5 for a discussion concerning the distances among different types of solutions.

Van Camp (1993). In Keskar et al. (2016), the authors show that minimizers with different geometrical properties can be found by varying algorithmic choices like the batch-size. In Jiang* et al. (2020) a large scale experiment exploring the correlation between generalization and different complexity measures reported some flatness-based measures as the most robust predictors of good generalization performance. Several optimization algorithms explicitly designed for finding flatter minima have been presented in the literature, resulting in improved generalization performance in several settings (Chaudhari et al., 2019; Pittorino et al., 2021; Yue et al., 2020). Some analytical investigations of phenomena related to flatness, their relation to generalization and algorithmic implications can be found in Baldassi et al. (2015; 2016); Zhou et al. (2018); Dziugaite & Roy (2017).

Several recent works analyze the topic of mode connectivity empirically and analytically. In Draxler et al. (2018), the authors construct low-loss paths between minima of networks trained on image data using a variant of the nudged elastic band algorithm (Jónsson et al., 1998). In the closely connected work Garipov et al. (2018), the authors develop a method for finding low-loss paths as simple as a polygonal chain with a single bend and use this insight for creating a fast ensembling method. In Gotmare et al. (2018) the authors show that minima found with different training and initialization strategies can be connected by high accuracy paths. Notably for the present work, the authors note that some of these choices like batch size or the optimizers are expected to have an effect on the flatness of the found solutions. Mode connectivity in the context of adversarial attacks has been studied in Zhao et al. (2020), where the authors also propose to exploit mode connectivity to *repair* tampered models. In Kuditipudi et al. (2019) it is shown analytically that, given some suitable assumptions on the robustness of the network, a low-loss path can be constructed between the solutions of ReLU networks with multiple layers. A similar result is derived in Shevchenko & Mondelli (2020) for networks trained specifically with SGD.

The importance of symmetries for the question of flatness has been highlighted in Dinh et al. (2017), where symmetries are used to show that simple notions of flatness are problematic. In Brea et al. (2019), low-loss paths between minimizers are constructed that cross 'permutation points', which are points where the weights associated with two hidden neurons in the same layer coincide. In Tatro et al. (2020), a method for aligning neurons based on matching activation distributions is introduced and the authors show analytically and empirically that this method increases mode connectivity. In Singh & Jaggi (2020) a neuron alignment method based on optimal transport is introduced in the context of *model fusion*, where one tries to merge two or more trained models into a single model. Among other results, the authors show that matching is crucial when averaging the weights of models trained in the same or different settings (for example trained on different subsets of the labels). In Entezari et al. (2021), the authors test the hypothesis that barriers on the *linear* path between minima of ReLU networks found by SGD vanish if the permutation symmetry is taken into account. They present evidence in the form of extensive numerical tests, exploring different settings with respect to the width, depth and architectures used.

3. Numerical tools for the study of the energy landscape geometry

As stated in the introduction, our aim is to study the geometry of the error landscape (which we will also call the "energy" landscape) in the space of networks. In particular, after sampling solutions (zero-error configurations) with different characteristics we study their flatness profiles and two types of paths connecting them: *geodesic* and *optimized* paths. The latter are obtained by finding a midpoint between two networks, using it as the starting point of a new optimization process that reaches zero error, and then connecting the new solution to the two original endpoints via two geodesic paths (this is a modified procedure of what is called *polygonal chains with one bend* in Garipov et al. (2018), where the authors use euclidean geometry and do not account for the permutation symmetry). To make these definitions precise, we first need to describe the tools by which we remove the symmetries in the neural networks (code available at `https://gitlab.com/bocconi-artlab/matchingnn`).

We will use the following notation. We denote a whole neural network (NN) with Θ, and with L the number of its layers. We use the index $\ell = 1, \dots, L$ for the layers, and denote the number of units in the layer by H^ℓ. Each layer has associated parameters that we collectively call θ^ℓ (which include batch-norm parameters if used). The input weights for the k-th unit are w_k^ℓ and its bias b_k^ℓ. For all layers except the last, the activations are $x_k^\ell = \mathrm{ReLU}\left(\Delta_k^\ell\right)$, where Δ_k^ℓ are the pre-activations. In binary networks we change the ReLU with a sign. The last layer is linear and the output of the network is given by $\mathrm{argmax}_k(x_k^L)$. In binary classification tasks we use a single output unit, and a sign instead of an argmax.

3.1. Breaking the symmetries

3.1.1. Normalization

Networks with continuous parameters and ReLU activation functions are invariant to the rescaling of the weights of the units in each hidden layer by a positive real value (that may be different for each unit), since the ReLU function has the property $\mathrm{ReLU}(a\Delta) = a\,\mathrm{ReLU}(\Delta)$ for any $a > 0$.

To break this symmetry, we apply the following normalization procedure, starting from the first layer and proceeding upward, see alg. 1.

Algorithm 1 Neural Network Normalization

Input: A NN with continuous weights, L layers, parameters $\{\theta^\ell\}_{\ell=1}^L$ and ReLU activations.
for $\ell = 1$ **to** $L-1$ **do**
  $\{|w_k^\ell|\}_{k=1}^{H^\ell} = \mathrm{ComputeUnitNorms}(\theta^\ell)$
  $\mathrm{Rescale}(\theta^\ell, \{|w_k^\ell|\}_{k=1}^{H^\ell})$
  $\mathrm{InverseRescale}(\theta^{\ell+1}, \{|w_k^\ell|\}_{k=1}^{H^\ell})$
end for
$|w|^L = \mathrm{ComputeLayerNorm}(\theta^L)$
$\mathrm{Normalize}(\theta^L, |w|^L)$

For each layer ℓ, starting from the first and up to $L-1$, we calculate the norm $|w_k^\ell|$ of each hidden unit k in the layer, and we multiply its incoming weights, bias and batch-norm parameters by $|w_k^\ell|^{-1}$, and its outgoing weights by $|w_k^\ell|$. As a result, all the units in the layer have norm 1, while the function realized by the network (and also its loss) remains unaffected. The last layer does not have this symmetry, but it can be globally rescaled by a positive factor since the output of the network is invariant with respect to this operation (as a consequence of using the argmax operation). For consistency with the other layers, we normalize the last layer to $\sqrt{H^L}$. The resulting space is a product of normalized hyper-spheres (one for each hidden unit apart from the ones in the last layer, which is globally normalized to a single hyper-sphere), inducing a generalized hyper-toroidal topology.

This normalization choice is rather natural, results in simple expressions for the computation of the geodesics (see below), and leads to sensible results; it is certainly not the only possible (or reasonable) one, and other possibilities might be worth exploring.

3.1.2. Alignment

Multi-layer networks (continuous or discrete) also have a discrete permutation symmetry, that allows to exchange the units in the hidden layers (neurons in fully-connected layers and filters in convolutional layers). When comparing two networks Θ_a and Θ_b, we break the symmetry by first normalizing both networks, and then by applying the following alignment procedure, again starting from the first layer and proceeding upward, see alg. 2.

For each layer $\ell = 1, \dots, L-1$, we use a matching algorithm to find the permutation π of the indices of the second network that maximizes the cosine similarity[3] $\sum_{k=1}^{H^\ell} (w^a)_k^\ell \cdot (w^b)_{\pi(k)}^\ell$. The permutation is applied to both the ingoing and the outgoing indices of the weights of layer ℓ of the second network (as well as other parameters associated to the units such as the biases and the batch-norm parameters). For the matching algorithm, we use `linear_sum_assignment` from SciPy (Virtanen et al., 2020), which uses a quadratic modified Jonker-Volgenant algorithm. Any algorithm solving the minimum weight matching in bipartite graphs problem would be appropriate.

Algorithm 2 Neural Network Alignment

Input: Two normalized NNs of the same type Θ_a, Θ_b with L layers and parameters $\{\theta_a^\ell\}_{\ell=1}^L$, $\{\theta_b^\ell\}_{\ell=1}^L$.
for $\ell = 1$ **to** $L-1$ **do**
  $\pi^\ell = \mathrm{Match}(\theta_a^\ell, \theta_b^\ell)$
  $\mathrm{PermutePrev}(\theta_b^\ell, \pi^\ell)$
  $\mathrm{PermuteNext}(\theta_b^{\ell+1}, \pi^\ell)$
end for

In the case of discrete networks, we use the sign activation function instead of the ReLU, thus instead of a continuous rescaling symmetry there is a discrete sign-reversal one (allowing to flip the signs of all ingoing and outgoing weights of any hidden unit). We break this symmetry during the alignment step: in the matching step, we use the absolute value of the cosine similarities in the optimization objective, then we apply the permutation, and finally for each unit we either flip its sign or not based on which choice maximizes the cosine similarity.

This procedure is not guaranteed to realize a global distance minimization between the two networks in a worst-case scenario (and it is not clear whether such goal would be computationally feasible). It also does not take into account the biases, if present. However, it guarantees that if Θ_a and Θ_b are the same network with shuffled hidden units labels, at the end they will be perfectly matched almost surely. Furthermore, it is simple to implement, computationally efficient, and rather general (it applies to fully-connected and convolutional layers, to continuous or discrete weights). Basing the matching on the weights (as opposed to using e.g. the activations) is also data-independent and consistent with our overall geometrical picture.

3.1.3. Geodesic paths

Given two (non-normalized) continuous networks Θ_a and Θ_b, we want to consider the *geodesic paths* between the networks in function space. Formally, this is defined as the shortest path between all possible permutations of the networks in the normalized space. We approximate this with the path between the normalized-and-aligned

[3] We can ignore the norms, thanks to our choice of the normalization and the fact that we do not match the last layer. For the same reasons, this is also equivalent to minimizing a squared distance.

networks, which can be computed easily thanks to our choice of the normalization. Consider first a hyper-sphere of norm n, and two vectors v_1, v_2 on it. The angle between them is $\phi = \arccos(v_1 \cdot v_2 / n^2)$, and their (geodesic) distance is $n\phi$. A generic point along the geodesic, located at distance $xn\phi$ from v_1 where $x \in [0, 1]$, can be expressed as $n\,\mathrm{Normalize}(v_1 + t(x)(v_2 - v_1))$ where $t(x) = (1 - \cos(\phi) + \sin(\phi)/\tan(\phi x))^{-1}$. This is easily extended to the metric on the full network: given two normalized-and-aligned networks, we can simply apply this formula independently (but with the same x) to each hidden unit of the first $L-1$ layers (with norm 1) and to the last layer (with norm $\sqrt{H^L}$). Similarly, the overall squared geodesic distance is just the sum of the squared geodesic distances within each spherical sub-manifold.

For discrete binary networks we use the Hamming distance to measure the discrepancy between two (aligned) networks. In this case there is no analog to the geodesic path, since there are multiple shortest paths connecting any two networks: each path corresponds to a choice of the order in which to change the weights that differ between the two networks. In our tests, we simply pick one such path at random. In this case, all curves are averaged over different random paths realizations, and although there is of course some variability (see error bars in the plots), it does not affect the results. Optimizing the curves using e.g. simulated annealing turned out to be too computationally expensive.

3.2. Minima with Different Flatness and Generalization

We sample different kinds of solutions by using different algorithms. The *standard* minima are found by using the Stochastic Gradient Descent (SGD) algorithm with Nesterov momentum. In order to find *flatter* minima, we use replicated-SGD (RSGD), see Pittorino et al. (2021), which was designed for this purpose. To find *sharper* minima, we use the the adversarial initialization described in Liu et al. (2020) followed by the SGD algorithm without momentum. We call this method ADV; it was developed to overfit the dataset and produce poorly generalizing solutions, but since there is a known correlation between flatness and generalization error, we expect (and indeed confirm a posteriori) that these solutions are sharp.

In all cases, we do not use ℓ_2 regularization or data augmentation; the only image pre-processing in our experiments is normalization of images to zero mean and unit variance.

In the case of discrete binary networks we used the same techniques, but based on top of the BinaryNet training scheme (see e.g. Simons & Lee (2019)), which is a variant of SGD tailored to binary weights. Our implementation differs from the original one (Hubara et al., 2016) in that the output layer is also binary (details in SI Sec. B).

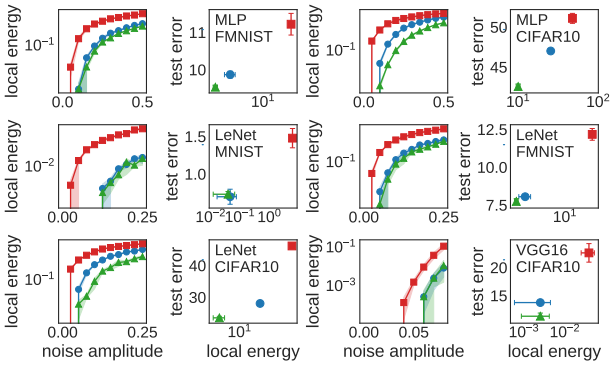

Figure 1. Flatness and generalization for several algorithms. Green triangles: RSGD; blue circles: SGD; red squares: ADV. Columns 1 and 3: local energies as a function of noise amplitude; columns 2 and 4: corresponding local energies (at maximal reported noise) versus test error. Shades and error-bars are standard deviations (over 100 sampled configurations for each noise amplitude).

4. Neural Networks with Continuous Weights

In this section we present results on the energy landscape geometry and connectivity obtained on NNs with continuous weights. We consider 3 different architectures of increasing complexity: a) a multi-layer perceptron (MLP) with 2 hidden layers of 512 units; b) a small LeNet-like architecture with 2 convolutional layers of 20 and 50 5×5 filters followed by a 2×2 MaxPool and one fully-connected layer of 500 units; c) the VGG16 architecture (Zhang et al., 2015) with batch normalization (the only network we train with batch-normalization: the corresponding parameters and running means/variances have to be permuted/rescaled coherently with the other layers' parameters). We train architectures a) and b) on MNIST, Fashion-MNIST and CIFAR-10, while architecture c) is trained on the CIFAR-10.

4.1. Flatness and generalization

We train these 3 networks with the 3 algorithms described in Sec. 3.2 (RSGD, SGD, ADV) for 300 epochs, sufficient to find a configuration at zero training error (or with error $< 1\%$ (Jiang* et al., 2020)). In Fig. 1 we show that the flatness of the solutions, measured by the *local energy*, follows the expected ranking between the algorithms, and the expected correlation with the generalization error, confirming that flatter minima generalize better. The local energy δE_{train} is defined as the average train error difference obtained by perturbing a given configuration w with a multiplicative Gaussian noise of varying amplitude σ, in formulas: $\delta E_{\text{train}}(w, \sigma) = \mathbb{E}_z\, E_{\text{train}}(w + \sigma z \odot w) - E_{\text{train}}(w)$ where $\odot$ is element-wise multiplication and $z \sim \mathcal{N}(0, 1)$. We notice here that the local energy is one among possible flatness definitions (with respect to weights perturbations) that correlates with generalization, and that other choices are possible, e.g. the robustness with respect to input perturbations.

4.2. Landscape geometry and connectivity

Analyzing the connectivity of minima with different flatness levels allows us to explore the fine structure of the energy landscape, shedding light on the geometry of possible connected basins and/or on the presence of isolated regions. To this aim we choose 5 distinct pairs of solutions (in order to calculate standard deviations, represented by shaded areas in the figures) for each of the 6 possible paths among the 3 types of solutions: the paths between solutions of the same flatness (RSGD-RSGD, SGD-SGD, ADV-ADV) and of different flatness (RSGD-SGD, RSGD-ADV, SGD-ADV). We calculate the train error of configurations along the paths connecting these different solutions and define the barrier along the path as the highest encountered train error (solutions are at training error $< 1\%$).

Let us notice that what we call *energy* along the paper, i.e. the train error, is directly related to applications and exhibits more symmetries (e.g., in the norm of the last layer) than the objective loss with which the networks have been optimized (in our work the cross-entropy). When calculating the loss along the one-dimensional paths and the bi-dimensional sections, defining representatives for the equivalence classes is less straightforward, and results are less uniform across models. This is possibly due to the fact that the last layer, that is not normalized when considering the loss rescaling symmetry, may result in having very different norms across different solutions.

Geodesic paths We show the error along the geodesic paths connecting pairs of minima for the different networks and datasets in Fig. 2. The top row in the figure demonstrates the effect of accounting for the symmetries, by showing three sets of curves - one set per panel, left to right: linear paths between raw configurations as output by the algorithms; linear paths in which the endpoint networks are aligned but not normalized[4]; the geodesic path between the normalized-and-aligned networks. We can see that: a) taking into account the permutation symmetry the barriers are lowered; b) following the geodesic removes the distortion in the paths, such that they appear flatter towards flatter solutions as they should c) considering these two symmetries the barriers heights (in particular their maximal value) follow a general overall ranking which is correlated to the flatness level of the corresponding solutions: the RSGD-RSGD one is consistently the lowest, followed by RSGD-SGD and SGD-SGD, then RSGD-ADV and SGD-ADV, and finally ADV-ADV is consistently the highest.

The top row of Fig. 2 is a representative example; analogous figures for all the other networks/datasets considered in the paper are reported in the Appendix, Sec. A.2. In all cases, accounting for the symmetries of the network is indeed critical to reveal these seemingly very general geometrical features. Let us notice that while the principal effect on the barrier's height is due to the permutation symmetry, moving along the geodesic path reflects in the particular shape of the path itself and on distances along the path (see also Fig. 5); as the train error is invariant w.r.t. the normalization procedure, large variations in barriers' height when considering this symmetry are not expected (but still possible due to the change in the path). In the case of solutions with different flatness level, a left-right asymmetry in the geodesic paths appears (even if not always with the same intensity), by coherently assigning a flatter profile towards the flatter configuration.

Notably, by sampling solutions with high flatness level using RSGD, we are able to target increasingly connected structures, even in cases like VGG in which they could appear as missing, see the black path in the lower-right panel of Fig. 2. We obtain, for example, much smaller barriers for linear mode connectivity than the ones found for the same type of network and dataset in (Entezari et al., 2021), although the barriers are still around 20% of train error.

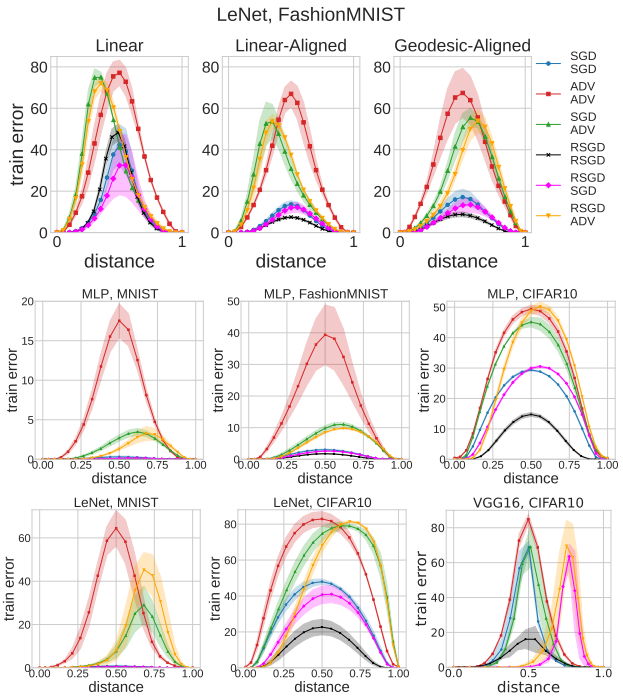

Figure 2. Error landscape along one-dimensional paths connecting minima of different flatness (minima appear on left/right following the up/down order of the legend). All distances have been rescaled in $[0, 1]$. The top row highlights the general effect of the symmetries in a representative example (LeNet on Fashion-MNIST). All other panels report the geodesic-aligned paths representing the final result of our analysis. Shades are standard deviations over 5 distinct pairs of solutions.

[4]We still maximize the layers' cosine similarity when performing the alignment.

Optimized paths Our procedure to find the optimized paths is the following: we first align the two endpoint networks, then find their midpoint, optimize it for 300 epochs with SGD in order to reach zero training error, and explore the two aligned geodesics connecting the optimized midpoint and the endpoints. The barrier in this case is defined as the highest training error encountered along the union of these two geodesics. We define the optimized path in this way (Garipov et al., 2018) because we seek a low-error path that is close to a straight path and also simple to find and define.

We report a representative example in Fig. 3 (for the other networks/datasets see Appendix Sec. A.2). Again, we contrast three situations, one in which we use straight segments between non-normalized unaligned configurations, one where we remove the permutation symmetry, and one where we use the geodesic paths. The barriers are much lower than for the unoptimized paths even in the unaligned linear case, but removing the permutation symmetry has still an important effect in lowering the barriers, and in most cases (like the one shown here) it removes them entirely. The presence of non-zero barriers in the SGD-ADV and RSGD-ADV cases in the left panel of Fig. 3, which would seem counter-intuitive given the absence of barriers in the ADV-ADV case, is removed after taking symmetries into account, thus showing an example of the need of this operation in order to obtain uniform and interpretable results. Intriguingly, in the (very high-dimensional) VGG case, the optimized barriers are smaller than 0.1% in all cases (see lower-right panel of Fig. 10), even though the barriers in the linear connectivity remain significantly different from zero even after removing symmetries (see lower-right panel of Fig. 2).

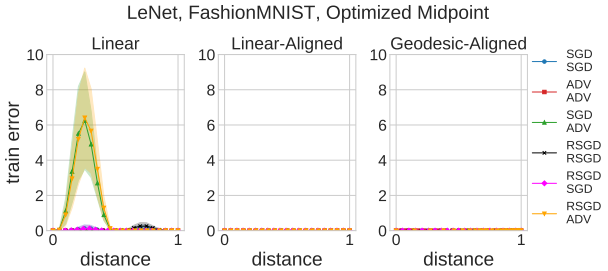

Figure 3. Removing the permutation symmetry eliminates the barriers that may appear in the single-bend optimized paths. Left to right: linear path; linear-aligned path; geodesic-aligned path. Shades are standard deviations over 5 distinct pairs of solutions.

Bi-dimensional visualizations Following Garipov et al. (2018), we study the error landscape on bi-dimensional sections of the parameters space using the Gram-Schmidt procedure. This procedure consists in considering the 3 solution vectors that specify the bi-dimensional plane, defining an orthonormal basis of this plane containing these 3 vectors, defining a Cartesian grid in this basis and evaluating the error of the networks corresponding to each of the points in the grid (see Garipov et al. (2018) for details).

In Fig. 4 we show the results on the planes defined by an RSGD solution (left-most point), an unaligned ADV solution (right-most point) and the corresponding aligned ADV solution (top point). We show the non-normalized (left panels) and normalized (right panels) cases. In the normalized case, the plane as a whole does not lie in normalized space (only the three chosen points do) and thus the plot suffers from some (presumably mild) distortion. Nevertheless, we can still see that accounting for the permutation symmetry alone reduces the distance between different minima and can lower the barriers between them, but also that only after normalization the expected relative sizes of the basins are revealed.

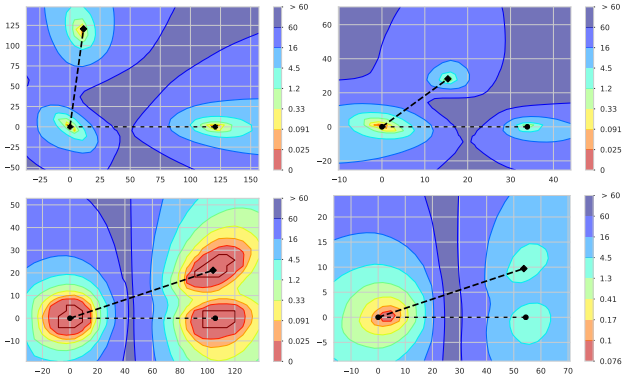

Figure 4. Bi-dimensional landscape sections obtained using the Gram-Schmidt procedure (see main text). Top row: LeNet, Fashion-MNIST. Bottom row: VGG16, CIFAR10. Left column: without normalization, lines are linear paths. Right column: with normalization, lines are distorted geodesics. In each panel the left point is RSGD, the right point unaligned ADV, the middle/top point is aligned ADV. Aligning the NNs can lower the barriers (not for VGG in this RSGD-ADV case), normalization reveals the geometry around them.

Distances. We studied the distances between pairs of solutions, categorized according to their flatness. Some representative results are reported in Fig. 5, the full results are in Appendix Sec. A.4. When NNs are normalized and aligned, we consistently find that flatter solutions are closer to each other than sharper ones. In optimized paths, the optimized midpoints end up closer to the flatter solutions. Although our sampling is very limited, these results (together with all the previous ones) are compatible with the octopus-shaped geometrical structures described in (Baldassi et al., 2021b).

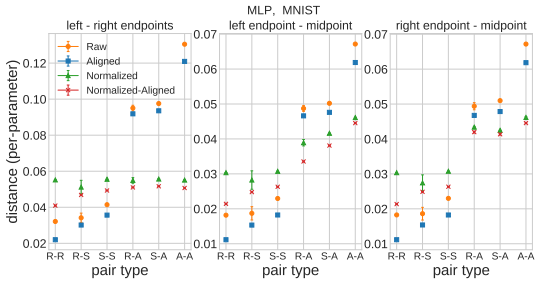

Figure 5. Distances between configurations, MLP on MNIST. (Left panel) Euclidean distance between configurations independently sampled by different algorithms (on the x-axis: R≡RSGD; S≡SGD; A≡ADV). (Middle panel) Euclidean distance between the optimized midpoint initialized as the mean of the two configurations in the x-axis with the one indicated on the left; (left panel) same as the middle panel but the distance is between the optimized midpoint and the configuration indicated on the right in the x-axis.

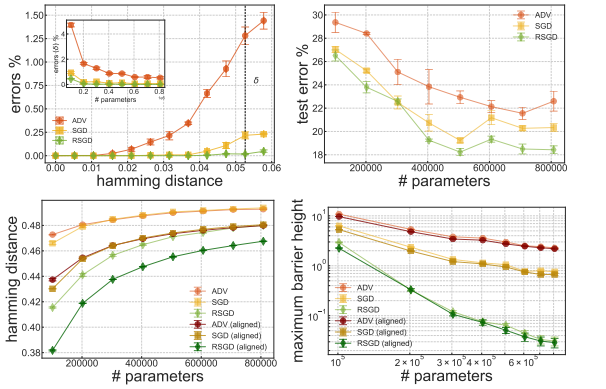

Figure 6. Fully connected binary CM trained on HMM data (see text). Top Left: Local energies for different classes of solutions. Inset: local energy at a fixed distance from a solution (δ in the main plot) as a function of the number of parameters. Top right: Test errors decrease with overparameterization and correlate with local energies. Bottom left: Average hamming distances between different solutions, before and after removal of symmetries. All distances grow with overeparameterization. Bottom right: Maximum barrier height (train error percentage) along a random shortest path connecting two solutions. Barriers go to zero with overparameterization.

5. Neural Networks with Binary Weights

In this section we present some results on the error landscape connectivity of binary NNs, in which each weight (and activation) is either 1 or -1, a topic that is almost absent in literature.

We considered two main scenarios: shallow networks on synthetic datasets, for which a comparison with some theoretical results is possible, and deep networks, both MLP and convolutional NNs, on real data.

5.1. Over-parameterized Shallow Networks on Synthetic Datasets

In order to bridge the gap with the theory, we performed numerical experiments on the error landscape connectivity in shallow binary architectures trained on data generated by the so called Hidden Manifold Model (HMM), also known as Random Features Model (Goldt et al., 2020; Gerace et al., 2020; Baldassi et al., 2021a).

The HMM is a mismatched student-teacher setting. The teacher generating the labels is a simple one-layer network, whose inputs are D-dimensional random i.i.d. patterns. The students does not see these original patterns, but a non-linear random projection onto an N-dimensional space. By varying the relative size N/D of the projection, the degree of overparameterization can be controlled. This arrangement aims to provide an analytically tractable model that retains some relevant features of the most common real-world vision datasets (Goldt et al., 2020).

We trained both a binary perceptron and a fully connected binary committee machine (CM), i.e. a network with a single hidden layer where the weights of the output layer are fixed to 1. Using data from a HMM, we analyzed the error landscape around solutions of different flatness, at varying levels of overparameterization (all implementation details are reported in Appendix Sec. B).

In this regime, the analysis of Baldassi et al. (2021a) suggests a scenario where algorithmically accessible solutions are arranged in a connected zero-error landscape, with flatter solutions surrounded by sharper ones. Overall, the numerical findings we report here are consistent with this scenario.

In Fig. 6 we report the results for the binary CM (similar results were obtained for the binary perceptron, see Appendix Sec. B). As the number of parameters is increased, the flatness of all the solutions increases (while the ranking RSGD-SGD-ADV is preserved), and the generalization error is correlated with the flatness, as expected. As more parameters are added and solutions become flatter, the average maximum error along random linear paths connecting two solutions decreases. We observe a robust correlation between barrier heights and flatness: the flatter the solution, the lower the barrier. However, the barriers are not significantly affected by aligning the networks.

5.2. Deep Architectures on Real-World Datasets

We consider two deep binary NNs: a MLP with 3-hidden-layers of 1001 units each, trained on the Fashion-MNIST dataset, and a 5-layers convolutional NN trained on CIFAR10 (implementation details in Appendix Sec. B).

For both architectures we analyzed the error landscape along

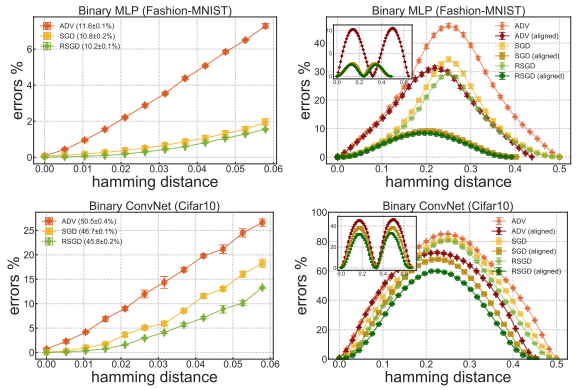

Figure 7. Three hidden layer binary MLP trained on the Fashion-MNIST dataset (top row) and binary convolutional NNs trained on CIFAR-10 (bottom row). Left: Local energies for different types of solutions (test errors are reported in the legend). Right: Train error percentage along random linear paths connecting solutions, for raw solutions (light curves) and aligned solutions (darker curves). Insets: errors along the optimized paths. The change in distance due to symmetries removal may be appreciated in the x-axis.

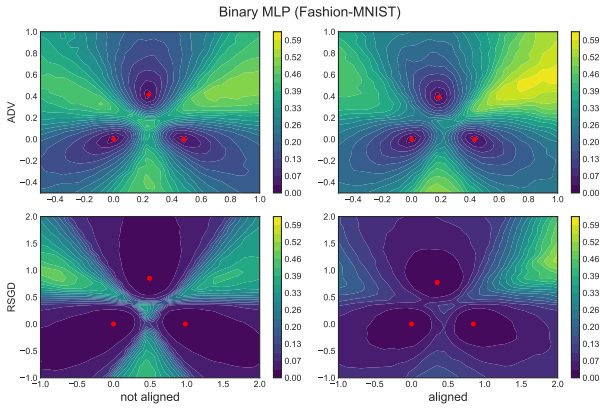

Figure 8. Train errors in the plane spanned by three solutions for a binary MLP trained on Fashion-MNIST, before (left column) and after (right column) symmetries removal. We compare ADV (top row) and RSGD (bottom row) solutions.

random shortest paths connecting pairs of solutions with different flatness. Results are reported in Fig. 7. As expected, the average barrier height correlates with the local energies of the solutions (and in turn with test errors). The barriers are lowered when symmetries are removed, and are lowered further when the paths are optimized (although they remain considerably large, especially for the convolutional NNs).

The effect of removing symmetries on the error landscape connectivity can be appreciated in Fig. 8 (see also Appendix Fig. 23 for analogous results on the convolutional network), where we projected the error landscape onto the plane spanned by the different solutions. We used the internal continuous weights used by BinaryNet (of which the actual weights are just the sign) and proceeded as in the continuous case (Sec. 4.2), but at each point we binarized the configurations in order to obtain the errors (details in Appendix Sec. B). The resulting projections are a heavily distorted representation, but the effect of symmetry removal on the barriers is rather striking, especially in the case of the wider minima, revealing the presence of a connected structure.

6. Conclusions and discussion

We investigated numerically several features of the error landscape in a wide variety of neural networks, building on a systematic geometric approach that dictates the removal of all the symmetries in the represented functions. The methods we developed are approximate but simple, rather general and efficient, and proved to be critically important to our findings. By sampling different kinds of minima, investigating the landscape around them and the paths connecting them, we found a number of fairly robust features shared by all models. In particular, besides confirming the known connection between wide minima and generalization, our results support the conjecture of Baldassi et al. (2021a): that, for sufficiently overparameterized networks, wide regions of solutions are organized in a complex, octopus-shaped structure with the denser regions clustered at the center, from which progressively sharper minima branch out. Intriguingly, a similar phenomenon has been recently observed also in the—rather different—context of systems of coupled oscillators (Zhang & Strogatz, 2021).

Our work lies at the intersection of two lines of research that have seen significant interest lately: one on mode connectivity and the structure of neural network landscapes and the other on flat minima and their connection to generalization performance. In this context, our work contributes to a deeper understanding of how deep neural networks operate. This may have algorithmic implications, for example previous investigations led to enabling better ensembling schemes (see, e.g. Garipov et al. (2018)), hyperparameter choices (see, e.g. Foret et al. (2021)) and methods for improving generalization (Pittorino et al., 2021), and has the potential to also help architecture design. Work is in progress to investigate the algorithmic implications of our results. We believe that further systematic explorations of the topics treated in this paper can produce results of great theoretical interest and significant algorithmic implications.

Acknowledgements

FP acknowledges the European Research Council for Grant No. 834861 SO-ReCoDi.

References

Baldassi, C., Ingrosso, A., Lucibello, C., Saglietti, L., and Zecchina, R. Subdominant dense clusters allow for simple learning and high computational performance in neural networks with discrete synapses. *Physical review letters*, 115(12):128101, 2015. doi: 10.1103/PhysRevLett.115.128101. URL https://doi.org/10.1103/PhysRevLett.115.128101.

Baldassi, C., Borgs, C., Chayes, J. T., Ingrosso, A., Lucibello, C., Saglietti, L., and Zecchina, R. Unreasonable effectiveness of learning neural networks: From accessible states and robust ensembles to basic algorithmic schemes. *Proceedings of the National Academy of Sciences*, 113(48):E7655–E7662, 2016. doi: 10.1073/pnas.1608103113. URL https://doi.org/10.1073/pnas.1608103113.

Baldassi, C., Gerace, F., Kappen, H. J., Lucibello, C., Saglietti, L., Tartaglione, E., and Zecchina, R. Role of synaptic stochasticity in training low-precision neural networks. *Physical review letters*, 120(26):268103, Jun 2018. doi: 10.1103/PhysRevLett.120.268103. URL https://link.aps.org/doi/10.1103/PhysRevLett.120.268103.

Baldassi, C., Malatesta, E. M., and Zecchina, R. Properties of the geometry of solutions and capacity of multilayer neural networks with rectified linear unit activations. *Phys. Rev. Lett.*, 123:170602, Oct 2019. doi: 10.1103/PhysRevLett.123.170602. URL https://link.aps.org/doi/10.1103/PhysRevLett.123.170602.

Baldassi, C., Pittorino, F., and Zecchina, R. Shaping the learning landscape in neural networks around wide flat minima. *Proceedings of the National Academy of Sciences*, 117(1):161–170, 2020. ISSN 0027-8424. URL https://www.pnas.org/content/117/1/161.

Baldassi, C., Lauditi, C., Malatesta, E. M., Pacelli, R., Perugini, G., and Zecchina, R. Learning through atypical"phase transitions"in overparameterized neural networks. *arXiv preprint arXiv:2110.00683*, 2021a. URL https://arxiv.org/abs/2110.00683.

Baldassi, C., Lauditi, C., Malatesta, E. M., Perugini, G., and Zecchina, R. Unveiling the structure of wide flat minima in neural networks. *Physical Review Letters*, 127(27):278301, Dec 2021b. doi: 10.1103/PhysRevLett.127.278301. URL https://link.aps.org/doi/10.1103/PhysRevLett.127.278301.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019. ISSN 0027-8424. doi: 10.1073/pnas.1903070116. URL https://www.pnas.org/content/116/32/15849.

Brea, J., Simsek, B., Illing, B., and Gerstner, W. Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape. *arXiv preprint arXiv:1907.02911*, 2019. URL https://arxiv.org/abs/1907.02911.

Chaudhari, P. and Soatto, S. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–10. IEEE, 2018. doi: 10.1109/ITA.2018.8503224. URL https://doi.org/10.1109/ITA.2018.8503224.

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., and Zecchina, R. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019. doi: 10.1088/1742-5468/ab39d9. URL https://doi.org/10.1088/1742-5468/ab39d9.

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pp. 1019–1028. PMLR, 2017.

Draxler, F., Veschgini, K., Salmhofer, M., and Hamprecht, F. Essentially no barriers in neural network energy landscape. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1309–1318. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/draxler18a.html.

Dziugaite, G. K. and Roy, D. M. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017. URL https://arxiv.org/abs/1703.11008.

Entezari, R., Sedghi, H., Saukh, O., and Neyshabur, B. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296*, 2021. URL https://arxiv.org/abs/2110.06296.

Feng, Y. and Tu, Y. The inverse variance–flatness relation in stochastic gradient descent is critical for finding flat minima. *Proceedings of the National Academy of Sciences*, 118(9), 2021. ISSN 0027-8424. doi: 10.1073/pnas.2015617118. URL https://www.pnas.org/content/118/9/e2015617118.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=6Tm1mposlrM.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. Loss surfaces, mode connectivity, and fast ensembling of dnns. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/be3087e74e9100d4bc4c6268cdbe8456-Paper.pdf.

Gerace, F., Loureiro, B., Krzakala, F., Mézard, M., and Zdeborová, L. Generalisation error in learning with random features and the hidden manifold model. In III, H. D. and Singh, A. (eds.), *International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3452–3462. PMLR, PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/gerace20a.html.

Goldt, S., Mézard, M., Krzakala, F., and Zdeborová, L. Modeling the influence of data structure on learning in neural networks: The hidden manifold model. *Physical Review X*, 10(4):041044, 2020. doi: 10.1103/PhysRevX.10.041044. URL https://link.aps.org/doi/10.1103/PhysRevX.10.041044.

Gotmare, A., Keskar, N. S., Xiong, C., and Socher, R. Using mode connectivity for loss landscape analysis. *arXiv preprint arXiv:1806.06977*, 2018. URL https://arxiv.org/abs/1806.06977.

Hinton, G. E. and Van Camp, D. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pp. 5–13, 1993. URL https://dl.acm.org/doi/pdf/10.1145/168304.168306.

Hochreiter, S. and Schmidhuber, J. Flat minima. *Neural computation*, 9(1):1–42, 1997. doi: 10.1162/neco.1997.9.1.1. URL https://doi.org/10.1162/neco.1997.9.1.1.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/d8330f857a17c53d217014ee776bfd50-Paper.pdf.

Jiang*, Y., Neyshabur*, B., Mobahi, H., Krishnan, D., and Bengio, S. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgIPJBFvH.

Jónsson, H., Mills, G., and Jacobsen, K. W. Nudged elastic band method for finding minimum energy paths of transitions. 1998.

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*, 2016.

Kuditipudi, R., Wang, X., Lee, H., Zhang, Y., Li, Z., Hu, W., Arora, S., and Ge, R. Explaining landscape connectivity of low-cost solutions for multilayer nets. *arXiv preprint arXiv:1906.06247*, 2019.

Liu, S., Papailiopoulos, D., and Achlioptas, D. Bad global minima exist and sgd can reach them. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 8543–8552. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/618491e20a9b686b79e158c293ab4f91-Paper.pdf.

Mei, S. and Montanari, A. The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 2019.

Pittorino, F., Lucibello, C., Feinauer, C., Perugini, G., Baldassi, C., Demyanenko, E., and Zecchina, R. Entropic gradient descent algorithms and wide flat minima. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=xjXg0bnoDmS.

Rocks, J. W. and Mehta, P. Memorizing without overfitting: Bias, variance, and interpolation in over-parameterized models. *arXiv preprint arXiv:2010.13933*, 2020. URL https://arxiv.org/abs/2010.13933.

Shevchenko, A. and Mondelli, M. Landscape connectivity and dropout stability of SGD solutions for over-parameterized neural networks. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8773–8784. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/shevchenko20a.html.

Simons, T. and Lee, D.-J. A review of binarized neural networks. *Electronics*, 8(6):661, 2019.

Singh, S. P. and Jaggi, M. Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33, 2020.

Tatro, N., Chen, P.-Y., Das, P., Melnyk, I., Sattigeri, P., and Lai, R. Optimizing mode connectivity via neuron alignment. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 15300–15311. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/aecad42329922dfc97eee948606e1f8e-Paper.pdf.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Yue, X., Nouiehed, M., and Kontar, R. A. Salr: Sharpness-aware learning rates for improved generalization. *arXiv preprint arXiv:2011.05348*, 2020.

Zhang, X., Zou, J., He, K., and Sun, J. Accelerating very deep convolutional networks for classification and detection. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):1943–1955, 2015. doi: 10.1109/TPAMI.2015.2502579.

Zhang, Y. and Strogatz, S. H. Basins with tentacles. *Phys. Rev. Lett.*, 127:194101, Nov 2021. doi: 10.1103/PhysRevLett.127.194101. URL https://link.aps.org/doi/10.1103/PhysRevLett.127.194101.

Zhang, Z., Zhou, H., and Xu, Z.-Q. J. A variance principle explains why dropout finds flatter minima. *arXiv preprint arXiv:2111.01022*, 2021. URL https://arxiv.org/abs/2111.01022.

Zhao, P., Chen, P.-Y., Das, P., Ramamurthy, K. N., and Lin, X. Bridging mode connectivity in loss landscapes and adversarial robustness. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgwzCEKwH.

Zhou, W., Veitch, V., Austern, M., Adams, R. P., and Orbanz, P. Non-vacuous generalization bounds at the imagenet scale: a pac-bayesian compression approach. *arXiv preprint arXiv:1804.05862*, 2018.

A. Neural Networks with Continuous Weights

We report complete and additional results on the Neural Networks (NNs) with continuous weights studied in the main paper.

A.1. Numerical details and parameters

The training parameters for the 3 algorithms for all the networks/datasets tested in the main paper are: (SGD) SGD with Nesterov momentum and initial learning rate 0.02 with cosine annealing (0.002 for MLP on CIFAR-10); (RSGD) Replicated-SGD with Nesterov momentum and initial learning rate 0.05 with cosine annealing, $y = 5$ replicas, with an exponential schedule on the interaction parameter $\gamma_t = \gamma_0(1 + \gamma_1)^t$ with γ_0 and γ_1 automatically chosen (see Pittorino et al. (2021) for details on this algorithm); (ADV) for the configurations obtained with the adversarial initialization, we use vanilla SGD and initial learning rate 0.02 with cosine annealing (see Liu et al. (2020) for details on the generation of this initialization: we replicate one time the dataset with random labels, i.e. $R = 1$, and we zero-out a 10% random fraction of the pixels in each image). We train all networks with batch-size 128 and for 300 epochs in order to reach training errors $< 1\%$. Neither data augmentation nor ℓ_2 regularization are used in our experiments.

A.2. One-Dimensional Paths

We add here further results on the comparison among Linear, Linear-Aligned and Geodetic-Aligned one-dimensional paths on the networks/dataset with continuous weights studied in the main paper. We report in Fig. 9 the paths without optimizing the midpoint, while in Fig. 10 the results obtained by optimizing it (the single-bend optimized paths).

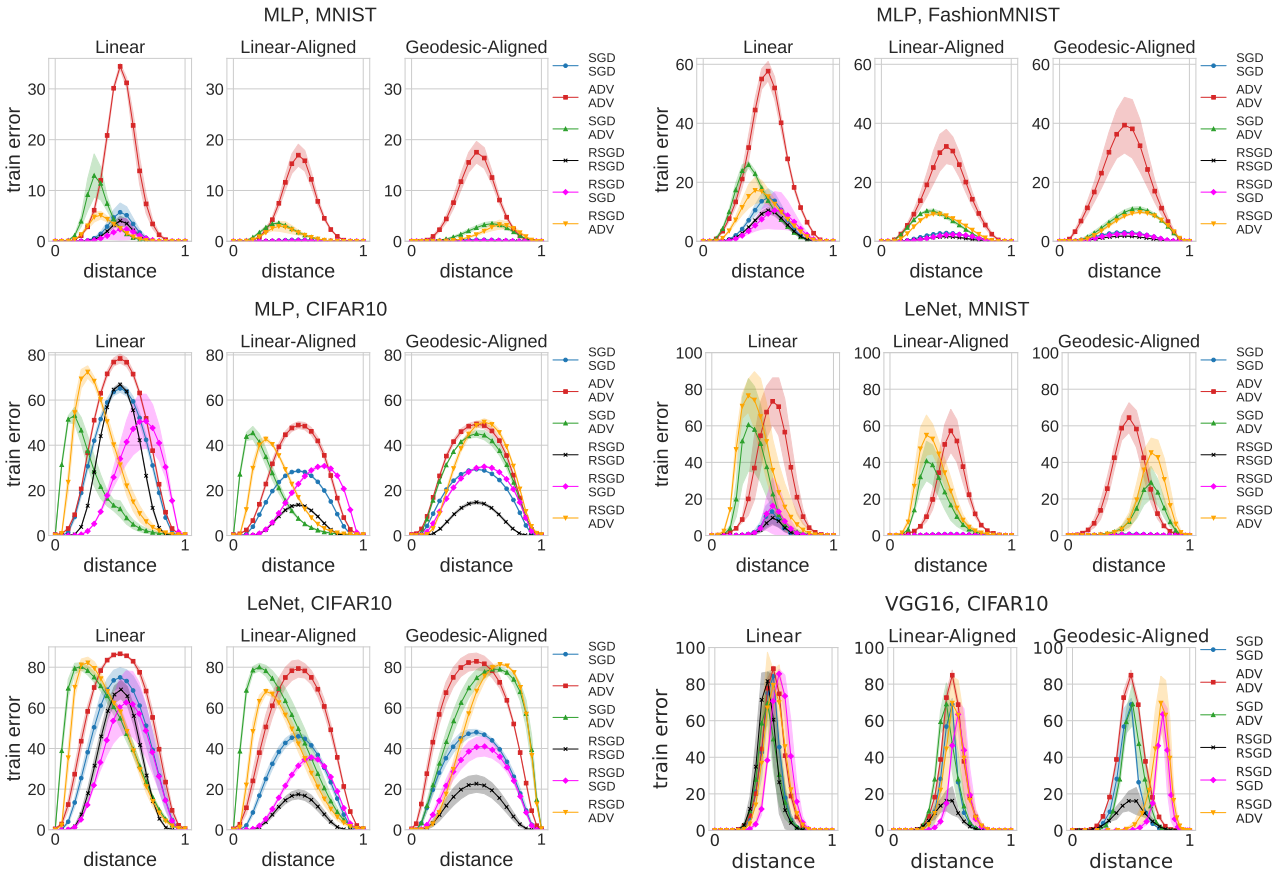

Figure 9. Error landscape along one-dimensional paths connecting minima of different flatness (minima appear on left/right following the up/down order of the legend). For each network/dataset the comparison highlights the general effect of the symmetries. The geodesic-aligned paths represents the final result of our analysis.

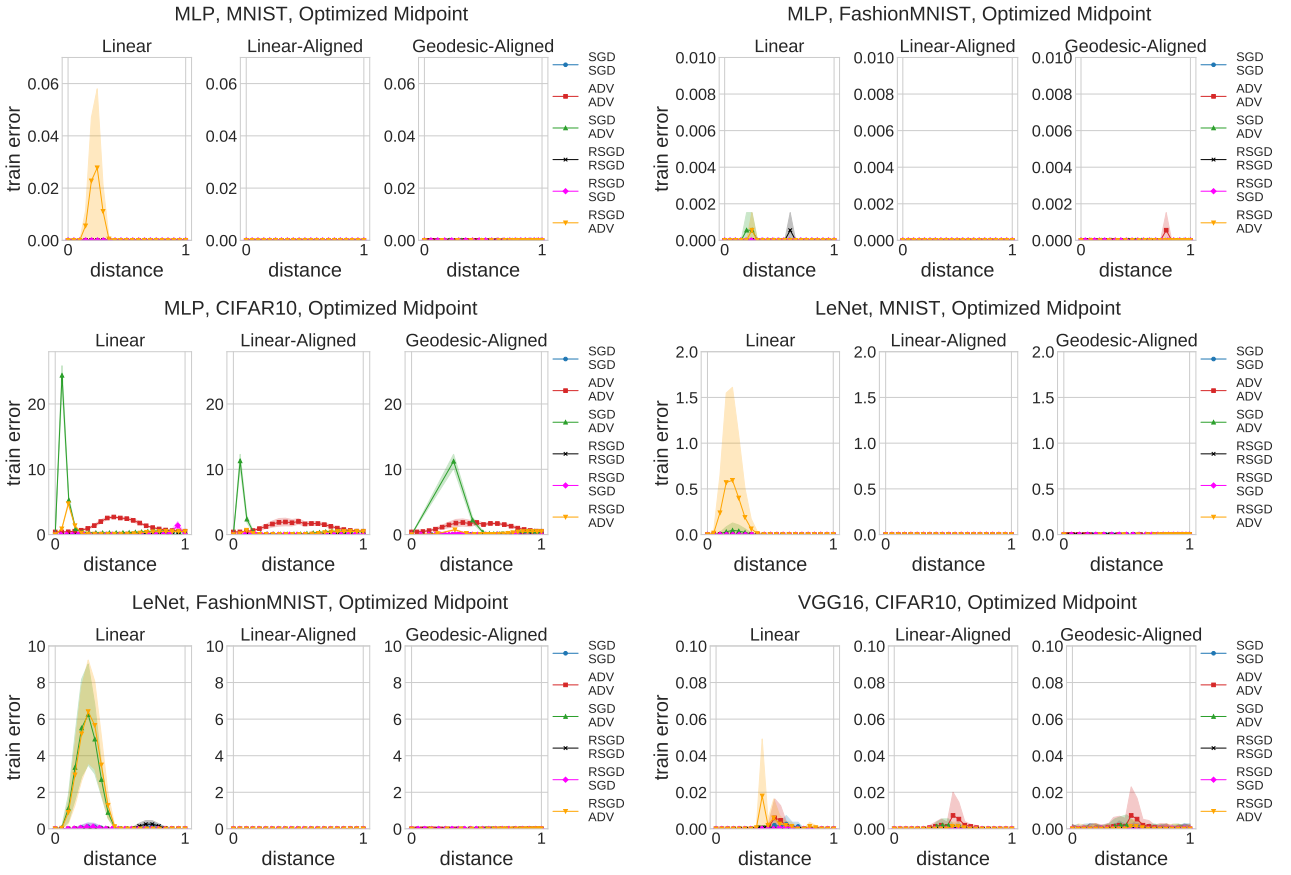

Figure 10. Removing the permutation symmetry eliminates the barriers that may appear in the single-bend optimized paths. For each network/dataset, left to right: linear path; linear-aligned path; geodesic-aligned path. For these optimized paths with one bend, the midpoint is optimized with batch-size 128 for 300 epochs with Nesterov momentum and initial learning rate 0.02 with cosine annealing in order to reach training error equal or almost equal to zero.

A.3. Bi-Dimensional Visualization

In this section we report bi-dimensional visualizations for some of the networks/datasets explored in the main paper. In Fig. 11 we compare train and test errors for LeNet on Fashion-MNIST (RSGD-ADV); In Figs. 12 and 13 we report visualizations for all pairs of LeNet on Fashion-MNIST (without and with normalization respectively); In Figs. 14 and 15 we report visualizations for all pairs of MLP on CIFAR-10 (without and with normalization respectively); in Fig. 16 we show the effects of permutation symmetry removal on VGG16 on CIFAR-10 for solutions of varying sharpness.

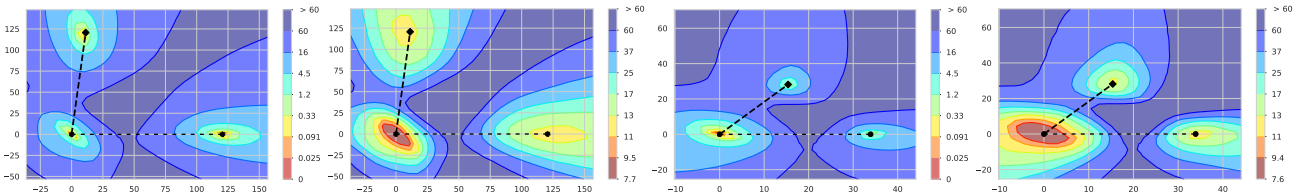

Figure 11. Bi-dimensional sections, LeNet on Fashion-MNIST. Train errors (panels 1 and 3) and test errors (panels 2 and 4). Comparison of RSGD (left points), unaligned ADV (right points), aligned ADV (top points). Panels 1 and 2: without normalization. Panels 3 and 4: with normalization. Dashed lines represent linear (panels 1 and 2) and geodesic paths (panels 3 and 4).

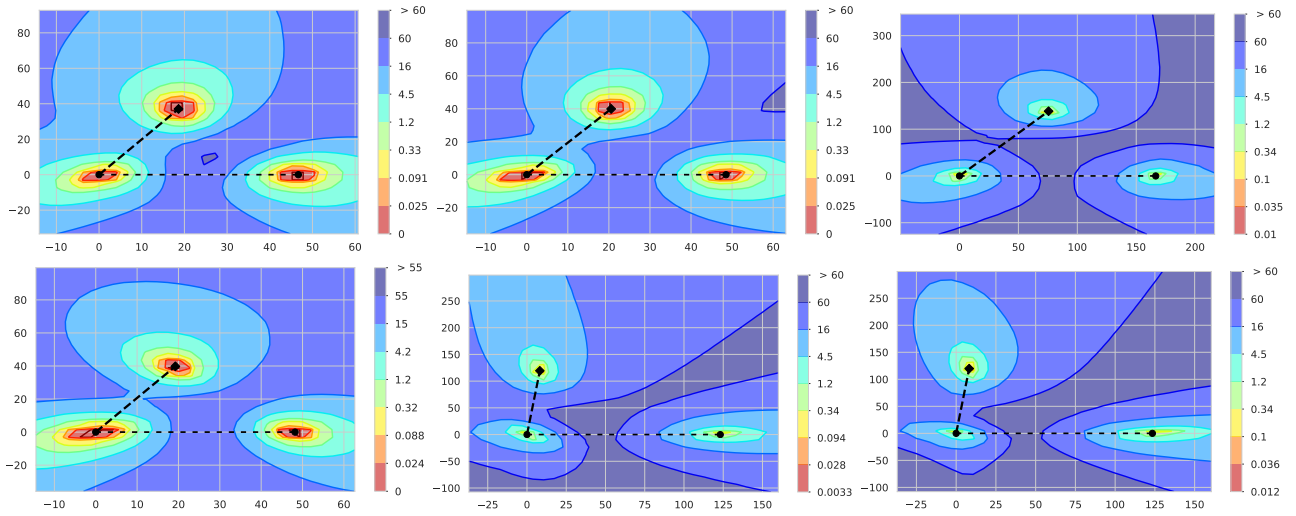

Figure 12. Bi-dimensional sections of the train error, LeNet on Fashion-MNIST, without normalization. Top points are the aligned version of the right point w.r.t. the left point. Top row: (left) left-right points: RSGD-RSGD; (middle) left-right points: SGD-SGD; (right) left-right points: ADV-ADV. Bottom row: (left) left-right points: RSGD-SGD; (middle) left-right points: RSGD-ADV; (right) left-right points: SGD-ADV. Aligning the NNs lowers the barriers and in some cases reveals that solutions lie in closer and connected basins. Dashed lines represent linear paths.

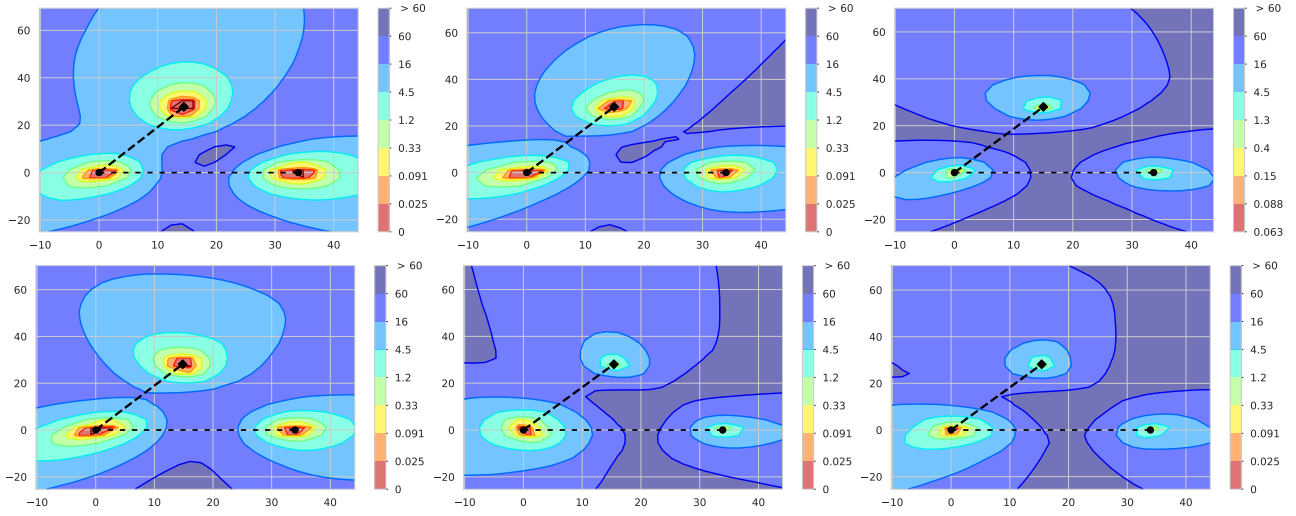

Figure 13. Bi-dimensional sections of the train error, LeNet on Fashion-MNIST, with normalization. Top points are always the aligned version of the right point w.r.t. the left point. Top row: (left) left-right points: RSGD-RSGD; (middle) left-right points: SGD-SGD; (right) left-right points: ADV-ADV. Bottom row: (left) left-right points: RSGD-SGD; (middle) left-right points: RSGD-ADV; (right) left-right points: SGD-ADV. Normalization reveals the geometry around the solutions. Dashed lines represent distorted geodesic paths.

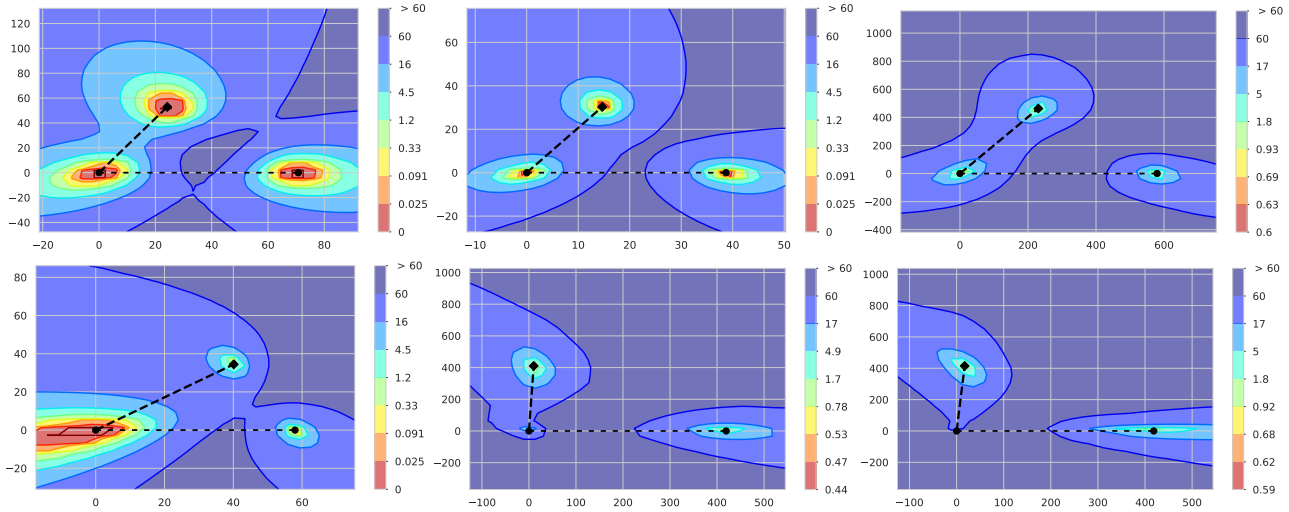

Figure 14. Bi-dimensional sections of the train error, MLP on CIFAR-10, without normalization. Top points are the aligned version of the right point w.r.t. the left point. Top row: (left) left-right points: RSGD-RSGD; (middle) left-right points: SGD-SGD; (right) left-right points: ADV-ADV. Bottom row: (left) left-right points: RSGD-SGD; (middle) left-right points: RSGD-ADV; (right) left-right points: SGD-ADV. Aligning the NNs lowers the barriers and in some cases reveals that solutions lie in closer and connected basins. Dashed lines represent linear paths.

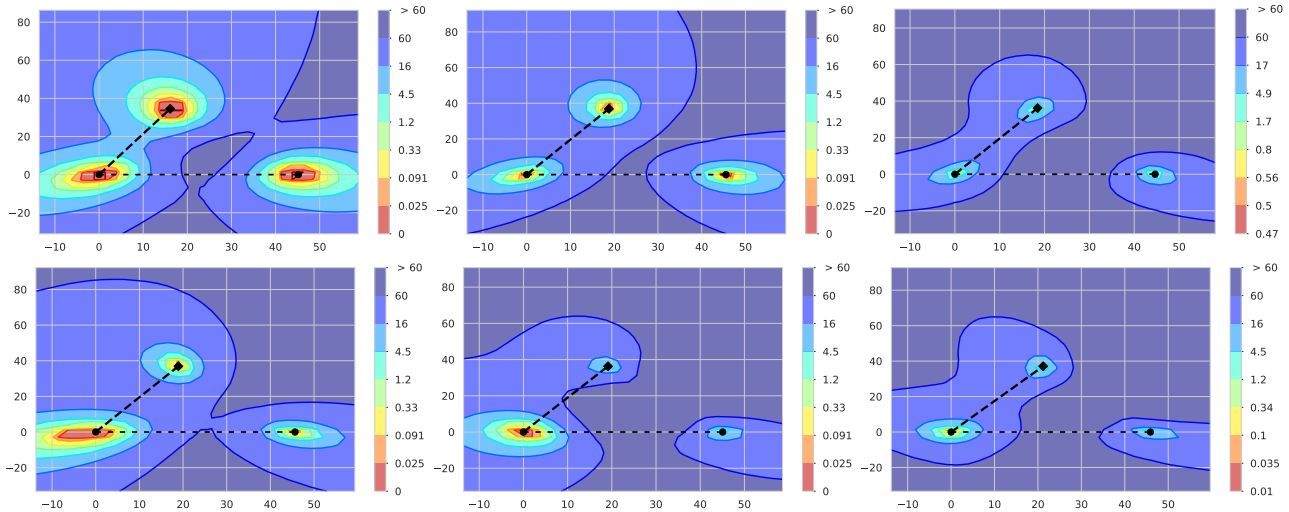

Figure 15. Bi-dimensional sections of the train error, MLP on CIFAR-10, with normalization. Top points are always the aligned version of the right point w.r.t. the left point. Top row: (left) left-right points: RSGD-RSGD; (middle) left-right points: SGD-SGD; (right) left-right points: ADV-ADV. Bottom row: (left) left-right points: RSGD-SGD; (middle) left-right points: RSGD-ADV; (right) left-right points: SGD-ADV. Normalization reveals the geometry around the solutions. Dashed lines represent distorted geodesic paths.

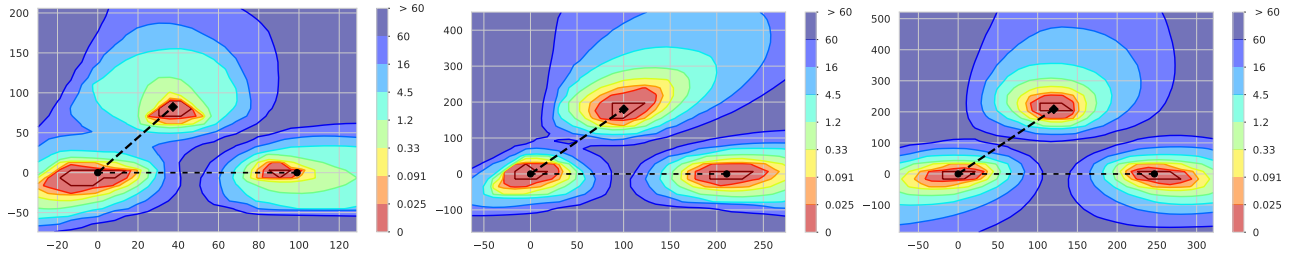

Figure 16. Bi-dimensional sections of the train error, effect of the permutation symmetry on VGG16 trained on CIFAR-10. Top points are the aligned version of the right point w.r.t. the left point. (Left) left-right points: RSGD-RSGD; (Middle) left-right points: SGD-SGD; (right) left-right points: ADV-ADV. Aligning the NNs lowers the barriers and in some cases reveals that solutions lie in closer and connected basins (for RSGD-RSGD and to a lesser extent for SGD-SGD), while it is not sufficient in other cases (ADV-ADV). Dashed lines represent linear paths.

A.4. Distances

We report in Fig. 17 the distances between configurations and optimized midpoints for the networks/datasets studied in the main paper.

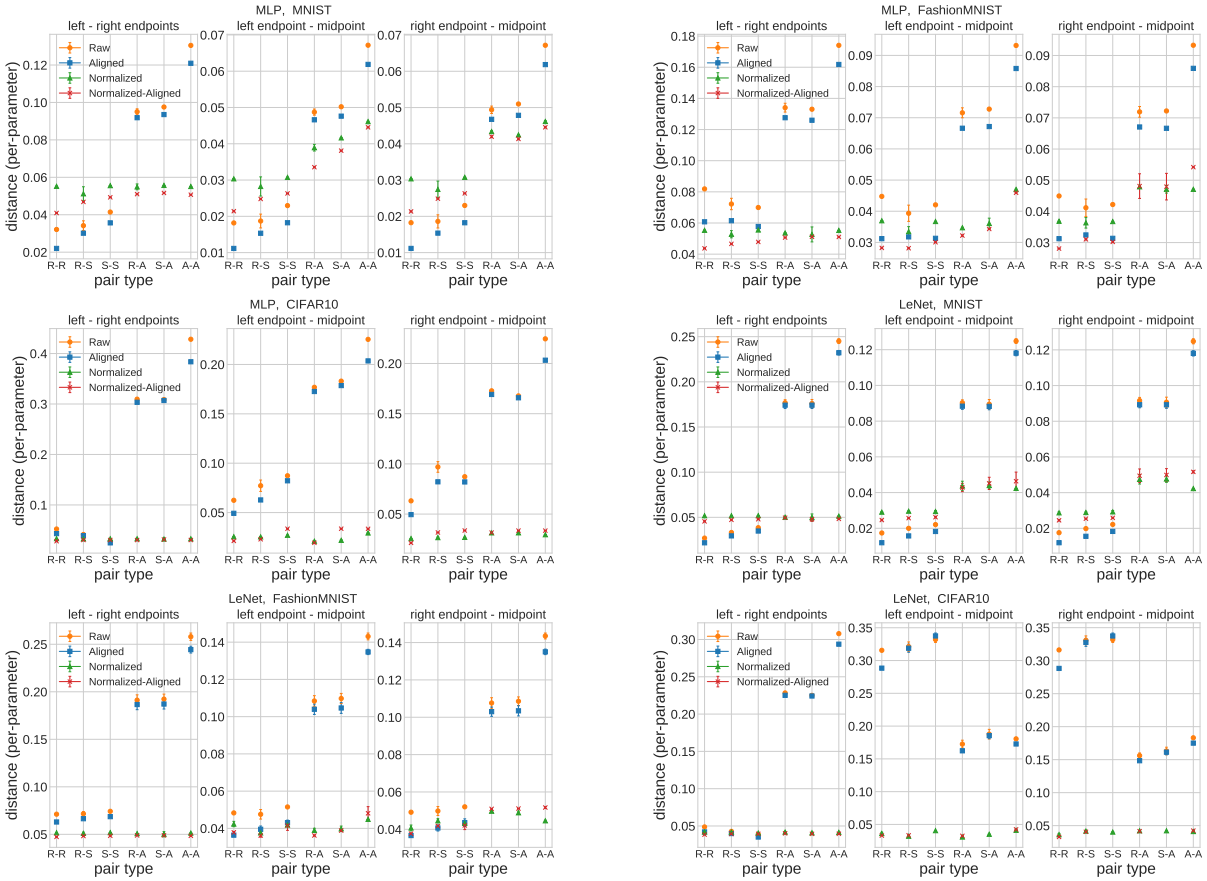

Figure 17. Distances between optimized configurations. For all networks/datasets: (Left panel) distance between configurations independently sampled by different algorithms (on the x-axis: R≡RSGD; S≡SGD; A≡ADV). (Middle panel) distance between the optimized midpoint initialized as the mean of the two configurations in the x-axis with the one indicated on the left; (left panel) same as the middle panel but the distance is between the optimized midpoint and the configuration indicated on the right.

B. Neural Networks with Binary weights

Here we provide implementation details for the experiments presented in Section 5, as well as additional numerical tests. In all our experiments we used the BinaryNet training scheme (see e.g. Simons & Lee (2019)), which is a variant of SGD tailored to binary weights. Notice that our implementation differs from the original one (Hubara et al., 2016) because we use binary weights in the whole network, including the output layer. In all the cases, for each solution type (ADV, SGD, RSGD) we were able to obtain solutions with zero or close to zero ($< 1\%$) train error.

Local Energies The local energy provides a robust measure of the flatness of a given solution. It has been shown to be highly correlated with generalization errors (Jiang* et al., 2020; Pittorino et al., 2021). It is defined as the average error as a function of the distance from a reference solution. In the case of binary weights neural networks, we perturb a solution by changing sign to a random fraction ε of the weights, and measure the train error for random choices of the perturbed weights. $\delta E(w, \varepsilon) = \mathbb{E}_{\varepsilon}\left[E(w, \varepsilon)\right] - E(w, \varepsilon = 0)$, where $E(w, \varepsilon = 0)$ is the train error of the unperturbed solution. By varying ε we are able to obtain the profile of the errors as a function of the hamming distance from the reference solution. In all the experiments, for each value of ε we average the errors over 10 independent realizations of the choice of the perturbed weights.

Random Linear Paths and Optimized Paths In order to explore the barriers between different solutions we measured the train error along the shortest paths connecting them. Given a source solution and a destination solution, we simply count the number of weights with different sign (that corresponds to the extensive hamming distance), progressively change the sign of those weights in the source solution in order to approach the destination solution, and measure the errors along the path. In all the experiments we consider 3 solutions for each ADV, SGD, RSGD (see Sec. 3.2). The reported paths are averaged over all the 6 possible paths among solutions of a given type (back and forth), and over 5 realizations of the random path (the weight flipping order).

Optimized paths are random linear paths with one bend. We take a weight vector located at equal distance between two solutions, and use it to initialize SGD. In all the experiments the middle point has been optimized using the same parameters of SGD-type solutions. We then report the random linear paths between the source and the optimized middle point, and the optimized middle point and the destination. For each of the 6 possible couples of solutions we averaged over 3 independent choices of the middle point and 5 realization of the random paths.

Shallow binary networks and the Hidden Manifold Model We trained both a binary perceptron and a binary fully-connected committee machine (CM) on synthetic data generated by the so-called Hidden Manifold Model (HHM) (Goldt et al., 2020; Gerace et al., 2020; Baldassi et al., 2021a), also known as the Random Features Model (Mei & Montanari, 2019).

In the HMM the data are first generated in a D-dimensional manifold, where a perceptron teacher $\{w_i^T\}_{i=1}^D$ assigns to a set of P random i.i.d. patterns $\{\boldsymbol{\xi}^\mu\}_{\mu=1}^P$ a label, according to $y^\mu = \text{sign}\left(\boldsymbol{w}^T \cdot \boldsymbol{\xi}^\mu\right)$. Then the student sees a non-linear random projection of the data into an N-dimensional manifold: $x_i^\mu = \text{sign}\left(\sum_{k=1}^D F_{ik}\xi_k^\mu\right)$, where $F \in \mathbb{R}^{N\times D}$ is a fixed random matrix with i.i.d. elements, and classify them according to the teacher label. In our experiments we fixed the size of the perceptron teacher (i.e. the dimension of the data points in their hidden manifold) to $D = 501$ and the pattern number to $P = 1503$, so that we are working at $\alpha_T \equiv P/D = 3$. By increasing the size N of the projection, we are able to study the system in the over-parameterized regime ($\alpha_D \equiv D/N \to 0$) (see also (Baldassi et al., 2021a)). In the case of the CM we fixed the hidden layer size to $H = 101$ and vary the input size N.

In Fig. 18 we report the results for the binary perceptron, analogous to the ones presented in the main text in Fig. 6. In the perceptron case there is no redundancy in the function expressed in the student model, so that solutions do not need to be aligned. As for the CM, in the limit $D/N \to 0$ we can see that solutions get flatter and the maximum error along random paths connecting them approaches zero. Even in this case, we observe a strong correlation among flatness (as measured with the local energy), generalization errors and maximum barrier heights.

We report training parameters for both models. Each model is trained with batchsize 100 for 200 epochs with binary cross entropy loss and SGD without momentum. For both the binary perceptron and the CM we used the following parameters: (SGD) lr$=$ 1.0, (RSGD) lr=1.0, with 5 replicas coupled with an elastic constant $\gamma(t) = 0.002 * (1.002)^t$, where t is the epoch. (ADV) networks have been first trained on a modified train set with randomized labels with lr$=$ 10.0 for 500 epochs. The resulting configuration is used as an initial condition and has been optimized with SGD for 200 epochs and lr$=$ 5.0.

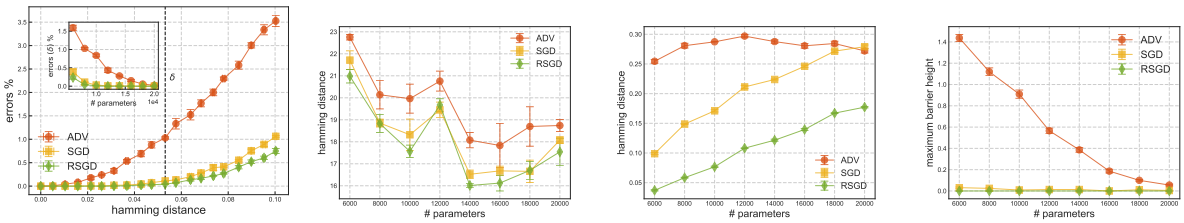

Figure 18. Binary Perceptron trained on data generated by the Hidden Manifold Model (see also Fig. 6) (Left) Local energy of different solutions. Inset: local energy at a fixed distance (δ) as a function of the number of parameters. (Center Left) Generalization errors of different solutions as a function of the number of parameters. (Center Right) Average hamming distance of different solutions. (Right) Average maximum train error (percentage) along random linear path connecting different solutions.

Binary Multi-layer Perceptrons We consider two binary Multi-layer perceptron (MLP) architectures: a) a two hidden layer MLP trained on 10k MNIST images (binary classification of the parity of digits), and b) a three hidden layer MLP trained on FashionMNIST. For architecture a) we analyzed the effect of removing symmetries as the network size is increased, by increasing the width of the hidden layers, while MLP b) has three hidden layer of fixed size $H = 1001$. In case a), for all hidden layer widths, we optimized the binary cross-entropy loss for 1000 epochs using SGD with no momentum, with fixed batchsize 100 and lr=10.0. For the ADV solutions we used as a starting point for SGD the results of 1000 epochs of SGD optimization on a train set where the label have been randomized. For RSGD solutions we used 5 replicas coupled with an epochs-dependent elastic constant $\gamma(t) = 0.002 * (1.002)^t$. In case b) we used Adam optimization with lr=0.001 for both SGD and RSGD solutions. For RSGD we used the same number of replica and elastic constant as in case a). The ADV solutions have been obtained by first training with SGD with no momentum and lr=5.0 for 500 on the train set with random labels, and then for other 1000 epochs with the same lr and optimization algorithm.

Results for case b) are reported in the main text (see Fig. 7, 8).

In case a) we performed analogous experiments as the one presented for shallow binary NNs on the HMM (see Fig. 18, 6). The results are reported in Fig. 19. As in the simpler synthetic dataset scenario, there is a strong correlation between flatness, generalization errors and barrier heights. However, while in this case removing the permutation and sign-reversal symmetries between solutions considerably lowers the respective barriers and average distances, the barriers do not seem to approach zero error in the limit of a large number of parameters.

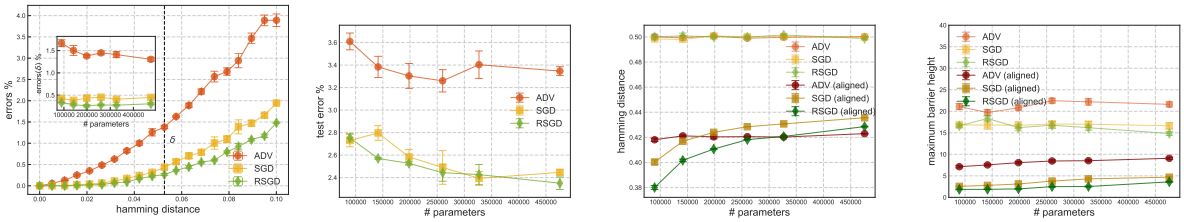

Figure 19. Binary MLP with 2 hidden layers with H units each, trained on 10k examples of the MNIST dataset (binary classification). (Left) Local energy for the tree type of solutions considered at $H = 201$. Inset: local energy at a fixed distance δ as a function of the number of parameters. (Center Left) Test errors for the three type of solutions as a function of the number of parameters. (Center Right) Average hamming distance among solutions, before (light colors) and after (dark colors) solutions have been aligned. (Right) Maximum barrier heights (in percentage of train errors) along random linear paths connecting solutions, before and after solutions have been aligned.

Binary Convolutional Network We trained a 5 layers convolutional networks with binary weights. The first two layers are convolutional layers with 20 and 50 5×5 filters with no padding, each followed by a 2×2 maxpool layer and sign activation function. The two convolutional layers are followed by 3 fully connected layers with 2001 units each. We used the following optimization parameters: (SGD) we trained the model for 400 epochs with Adam optimization with batchsize=128 and lr=0.005 that is multiplied by 0.5 every 50 epochs. (RSGD) We used 3 replicas coupled with an elastic constant $\gamma(t) = 0.001 * (1.001)^t$, where t is the epoch. All the other parameters are the same as SGD, except that we optimized for 300 epochs. (ADV) We first initialize the solutions by optimizing a modified train set where the labels have been randomized for 200 epochs with SGD with lr=10.0, and then train them for 200 epochs using Adam with lr=0.01 that

is halved at epoch $10, 20, 50$ and then every 50 epochs.

Bi-Dimensional Error Landscapes We describe the procedure to produce the bi-dimensional error landscape plot reported in the main text (Fig. 8). Given three solutions, we pick the continuous weights associated with the binary ones[5] and use them to construct an orthonormal basis using the Gram-Schmidt procedure (as for the case of continuous NNs, Sec. 4.2). At each point in the plane, we binarize the weights by taking their sign, and report the train error. With this bi-dimensional projection of the error landscape one can graphically appreciate the effect of removing symmetries. In Fig. 20, 21, 22 we report the error landscape for the three types of solutions considered (ADV, SGD, RSGD) in the case of the 2-hidden layer MLP trained on MNIST, as a function of the number of parameters. Without taking into account symmetries, the solutions appear to be isolated, while their flatness increase. However, once solutions are aligned by removing symmetries, a different landscape appears, where they are connected with low errors paths. In Fig. 23 we show the effect of removing symmetries for ADV and RSGD solutions of the binary convolutional network (analogous to Fig. 8).

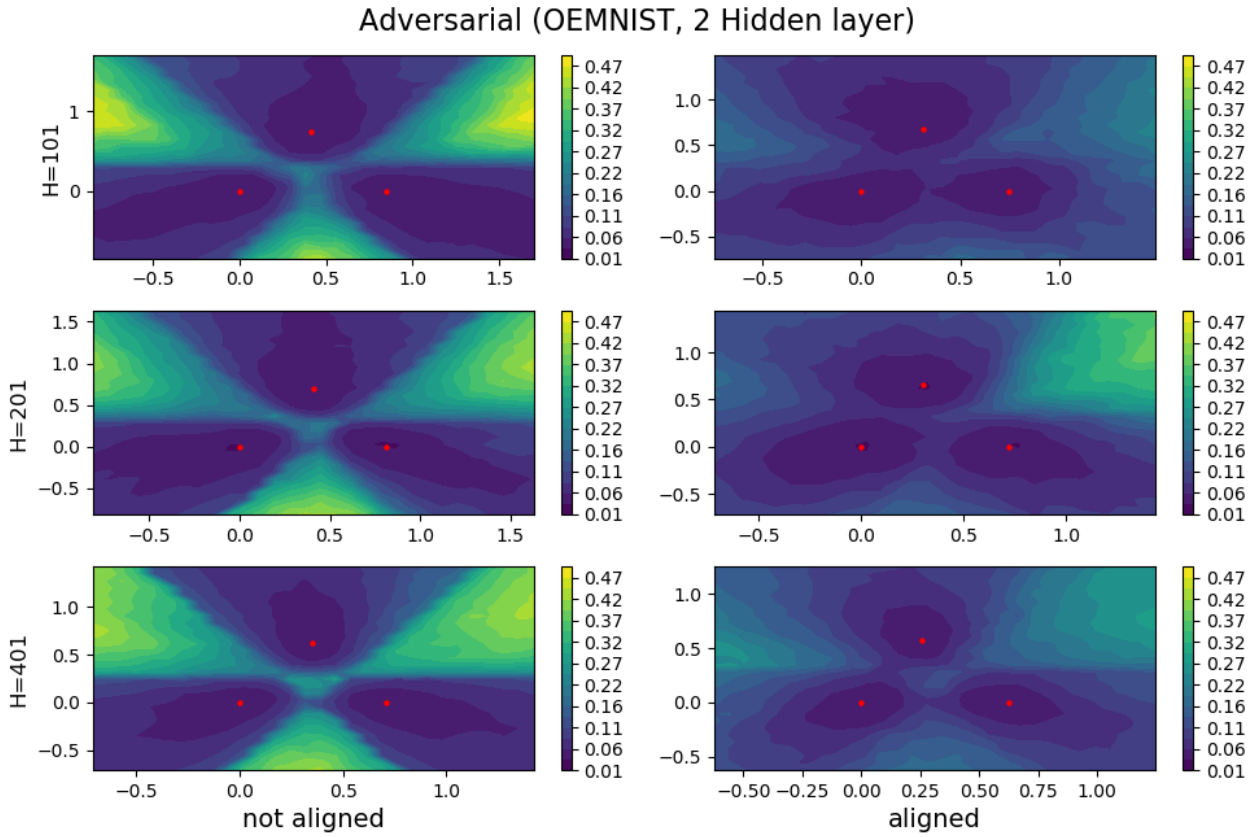

Figure 20. Bi-dimensional error landscape for ADV solutions of a 2-hidden layer MLP trained on MNIST. From top to bottom the width of hidden layers is increased. Left column: raw solutions. Right column: top and right solutions have been aligned to the left solutions.

[5] see (Hubara et al., 2016) for more insights on the relation between continuous and binary weights in the BinaryNet optimization scheme

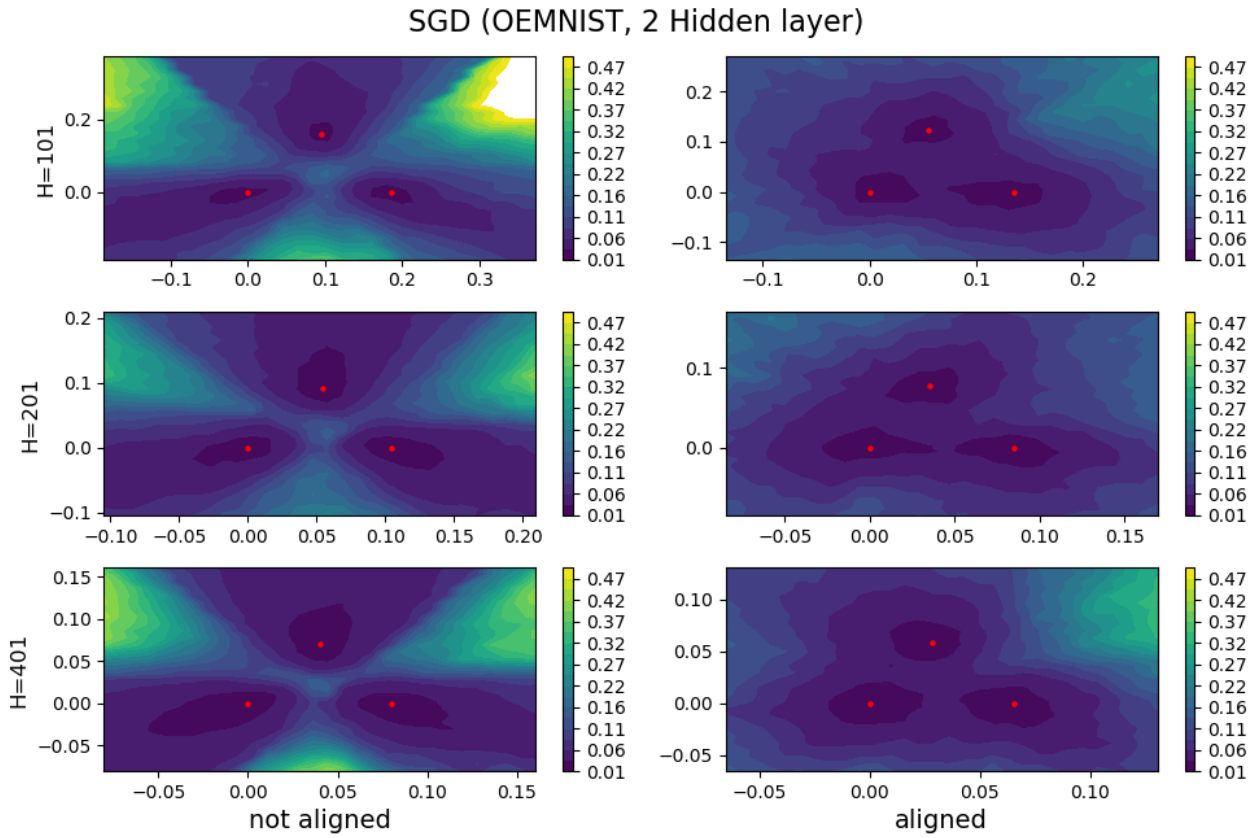

Figure 21. Bi-dimensional error landscape for SGD solutions of a 2-hidden layer MLP trained on MNIST. From top to bottom the width of hidden layers is increased. Left column: raw solutions. Right column: top and right solutions have been aligned to the left solutions.

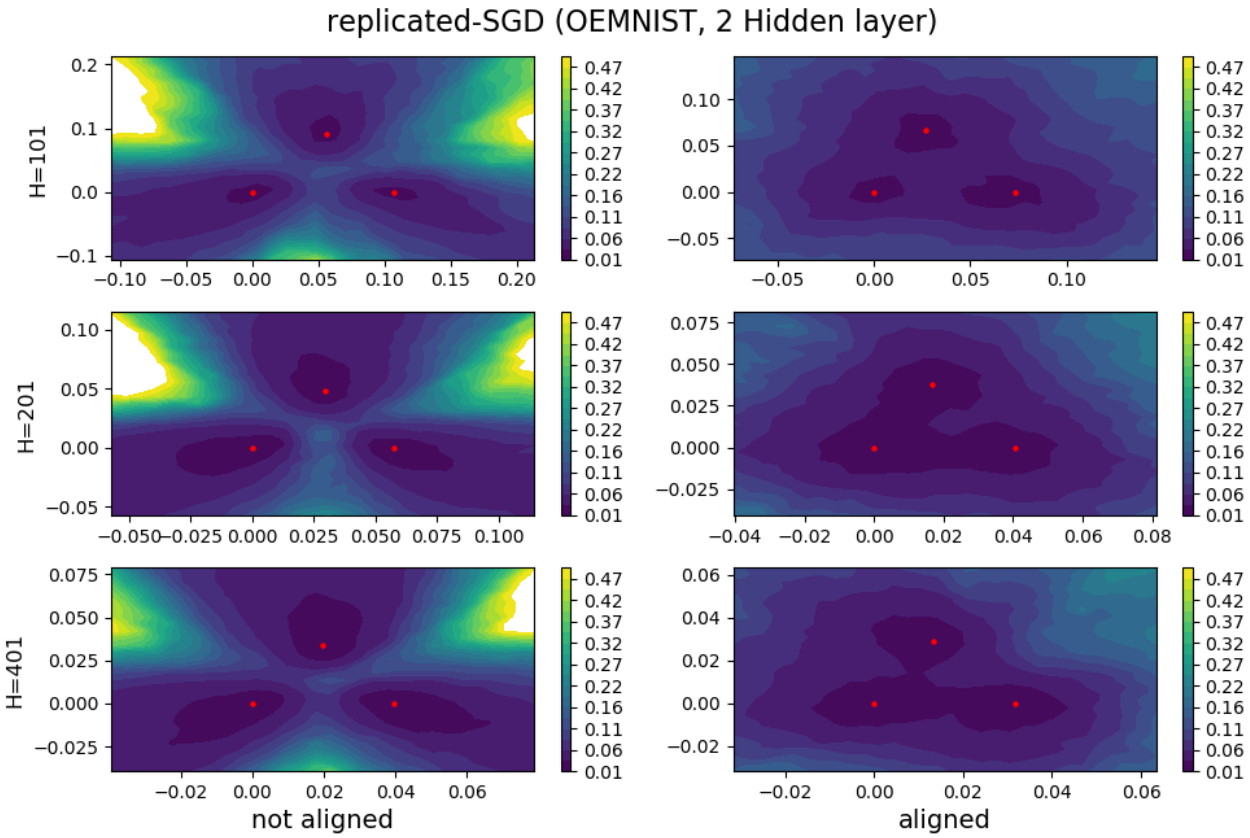

Figure 22. Bi-dimensional error landscape for RSGD solutions of a 2-hidden layer MLP trained on MNIST. From top to bottom the width of hidden layers is increased. Left column: raw solutions. Right column: top and right solutions have been aligned to the left solutions.

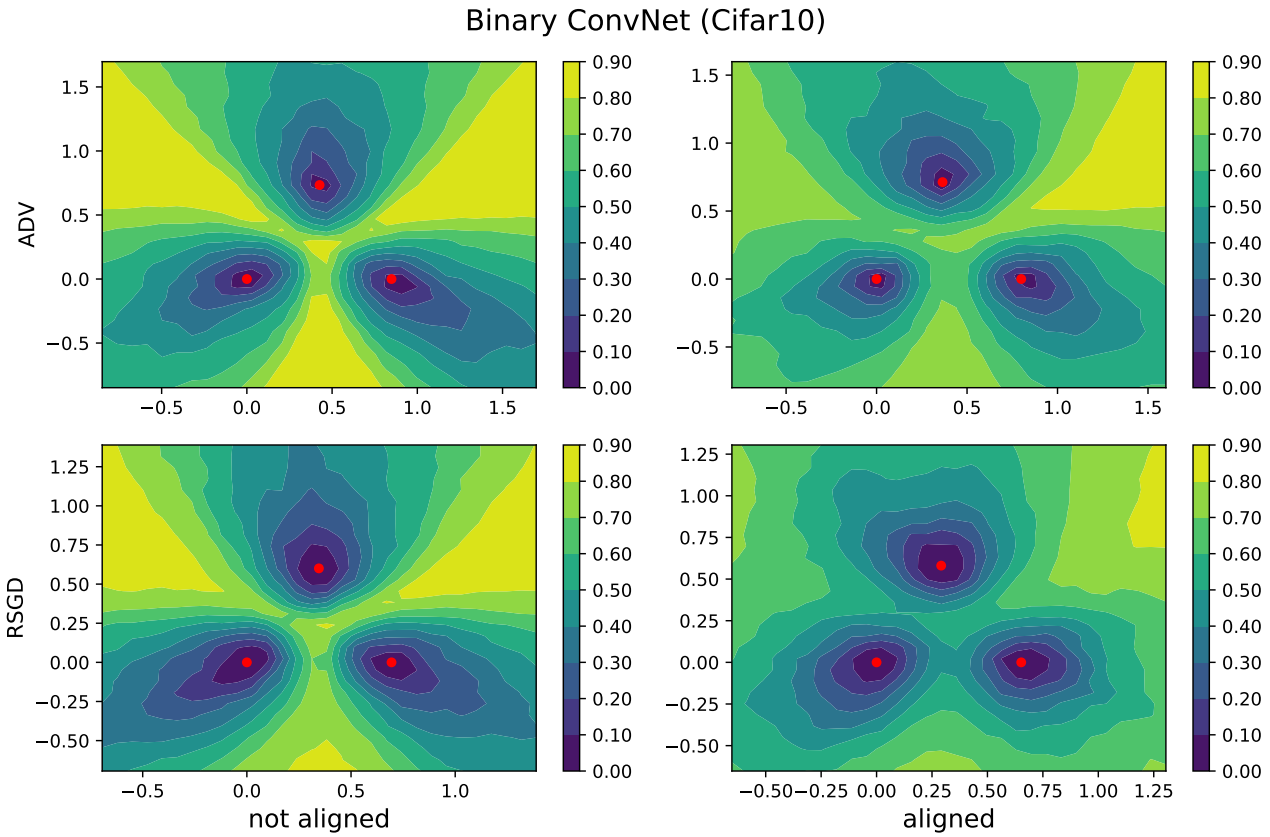

Figure 23. Train errors in the plane spanned by three solutions (for both ADV and RSGD solutions) for a binary convolutional NN trained on CIFAR-10 (see also Fig. 8). Going from left to right panels one can appreciate the effect of removing symmetries in the error landscape.