A. Theoretical Gauss Length Analysis

In this section, we provide an analysis of the MLI property via the Gauss Length. In particular, we prove Theorem 3 that states that if the logit interpolation of a network (from initialization to optimum) has small Gauss length then it must satisfy the MLI property. Using Theorem 3, we provide sufficient conditions for the MLI property to hold for two-layer linear models. And prove, under a class of these models satisfying some standard assumptions, that the MLI property holds almost surely.

Let's first recall the definition of the Gauss length.

Definition 2 (Gauss length)**.** *Given a curve* $\mathbf{z} : (0,1) \to \mathbb{R}^d$*. Let* $\hat{\mathbf{v}}(\alpha) = \frac{\partial \mathbf{z}}{\partial \alpha} / \| \frac{\partial \mathbf{z}}{\partial \alpha} \|_2$ *denote the normalized tangent vectors. The length under the Gauss map (Gauss length) is given by:*

$$\int_0^1 \sqrt{\langle \partial_\alpha \hat{\mathbf{v}}(\alpha), \partial_\alpha \hat{\mathbf{v}}(\alpha) \rangle} d\alpha,$$

where $\partial_\alpha \hat{\mathbf{v}}(\alpha)$ *denotes the pushforward of the Gauss map acting on the acceleration vector.*

Explicitly, we have,

$$\langle \partial_\alpha \hat{\mathbf{v}}(\alpha), \partial_\alpha \hat{\mathbf{v}}(\alpha) \rangle = \frac{(\mathbf{v} \cdot \mathbf{v})(\mathbf{a} \cdot \mathbf{a}) - (\mathbf{a} \cdot \mathbf{v})^2}{\mathbf{v} \cdot \mathbf{v}} = \kappa(\alpha)^2 (\mathbf{v} \cdot \mathbf{v}),$$

where $\mathbf{a} = \frac{\partial \mathbf{v}}{\partial \alpha}$ and κ denotes the curvature of $\mathbf{z}$. Theorem 3 is reproduced below for convenience.

Theorem 3 (Small Gauss length gives monotonicity)**.** *Let* $\mathcal{L}(\mathbf{z}) = \|\mathbf{z} - \mathbf{z}^*\|_2^2$ *for* $\mathbf{z}^* \in \mathbb{R}^d$*, and let* $\mathbf{z} : (0,1) \to \mathbb{R}^d$ *be a smooth curve in* $\mathbb{R}^d$ *with* $\mathbf{z}(1) = \mathbf{z}^*$ *and* $\mathcal{L}(\mathbf{z}(0)) > 0$*. If the Gauss length of* $\mathbf{z}$ *is less than* $\pi/2$*, then* $\mathcal{L} \circ \mathbf{z}(\alpha)$ *is monotonically decreasing in* α*.*

To prove this result, we will require the following Lemma.

Lemma 4. *Let* $\mathbf{x}^* \in \mathbb{R}^d$*. Consider a smooth curve* $\mathbf{z}(t) \in \mathbb{R}^d$ *for* $t \in [0,1)$ *with* $\|\mathbf{z}(0) - \mathbf{x}^*\| > 0$ *and* $\mathbf{z}(1) = \mathbf{x}^*$*. If there exists* $b \in [0,1)$ *with,*

$$\|\mathbf{z}(b) - \mathbf{x}^*\|_2 > \|\mathbf{z}(0) - \mathbf{x}^*\|_2,$$

then there exists $t_1 \in [0, b)$ *and* $t_2 \in (b, 1)$ *such that* $\langle \dot{\mathbf{z}}(t_1), \dot{\mathbf{z}}(t_2) \rangle \leq 0$*.*

Proof. We prove the contrapositive statement: If for all $t_1 \in [0, b)$ and $t_2 \in (b, 1)$, we have $\langle \dot{\mathbf{z}}(t_1), \dot{\mathbf{z}}(t_2) \rangle > 0$, then, for all $b \in [0,1)$, we have $\|\mathbf{z}(b) - \mathbf{x}^*\|_2 \leq \|\mathbf{z}(0) - \mathbf{x}^*\|_2$.

By the fundamental theorem of calculus, we have,

$$\begin{aligned} 0 < \int_b^1 \int_0^b \langle \dot{\mathbf{z}}(t_1), \dot{\mathbf{z}}(t_2) \rangle dt_1 dt_2 &= \langle \mathbf{z}(b) - \mathbf{z}(0), \mathbf{x}^* - \mathbf{z}(b) \rangle, \\ &= \langle \mathbf{x}^* - \mathbf{z}(0) + \mathbf{z}(b) - \mathbf{x}^*, \mathbf{x}^* - \mathbf{z}(b) \rangle \\ &= \langle \mathbf{x}^* - \mathbf{z}(0), \mathbf{x}^* - \mathbf{z}(b) \rangle - \|\mathbf{x}^* - \mathbf{z}(b)\|_2^2, \end{aligned}$$

Now, notice that as $\|\mathbf{x}^* - \mathbf{z}(b)\|_2^2 \geq 0$, we must have $\langle \mathbf{x}^* - \mathbf{z}(0), \mathbf{x}^* - \mathbf{z}(b) \rangle > 0$. Thus, by applying the Cauchy-Schwarz inequality,

$$\begin{aligned} \langle \mathbf{x}^* - \mathbf{z}(0), \mathbf{x}^* - \mathbf{z}(b) \rangle - \|\mathbf{x}^* - \mathbf{z}(b)\|_2^2 &\leq \|\mathbf{x}^* - \mathbf{z}(0)\|_2 \|\mathbf{x}^* - \mathbf{z}(b)\|_2 - \|\mathbf{x}^* - \mathbf{z}(b)\|_2^2, \\ &= \|\mathbf{x}^* - \mathbf{z}(b)\|_2 \left(\|\mathbf{x}^* - \mathbf{z}(0)\|_2 - \|\mathbf{x}^* - \mathbf{z}(b)\|_2 \right). \end{aligned}$$

It follows immediately that for any b we must have,

$$\|\mathbf{z}(b) - \mathbf{x}^*\|_2 \leq \|\mathbf{z}(0) - \mathbf{x}^*\|_2,$$

as required. □

With this result, we proceed with the proof of Theorem 3.

Proof. We prove this theorem by considering the contrapositive statement: if there exists $a < b \in (0, 1)$ such that $f(\mathbf{z}(a)) < f(\mathbf{z}(b))$ then the Gauss length is greater than $\pi/2$.

Given such a pair (a, b), we consider the restriction of $\mathbf{z}$ to $[a, 1)$. By Lemma 4, there exists t_1 and t_2 such that $\langle \dot{\mathbf{z}}(t_1), \dot{\mathbf{z}}(t_2) \rangle < 0$.

Therefore, the normalized tangents also satisfy $\langle \hat{\mathbf{v}}(t_1), \hat{\mathbf{v}}(t_2) \rangle < 0$. Thus, the angle between the two normalized tangents (considered in the plane containing these two points and $\mathbf{x}^*$) is at least $\pi/2$. Therefore, the Gauss length of the curve on $(a, 1)$ must be at least $\pi/2$ (with the minimum Gauss length path given by the shortest path on the projective plane connecting $\hat{\mathbf{v}}(t_1)$ and $\hat{\mathbf{v}}(t_2)$). □

Finally, we note here that the converse of Theorem 3 does not hold. For example, one may define a curve that spirals towards the minima. This curve is monotonically decreasing but has arbitrarily large Gauss length.

A.1. Two-layer linear models and the MLI property

In this section, we apply Theorem 3 to two-layer linear models. In particular, we prove sufficient conditions on any two-layer linear model to satisfy the MLI property and then prove that under certain assumptions the MLI property holds almost surely.

Our focus is on two-layer linear models, of the form $f(\mathbf{x}) = VW\mathbf{x}$, for $W \in \mathbb{R}^{k \times d}$ and $V \in \mathbb{R}^{m \times k}$. We consider optimizing these models with respect to the mean squared error.

$$\mathcal{L}(X, Y; V, W) = \frac{1}{2n} \|VWX - Y\|_2^2,$$

where $X \in \mathbb{R}^{d \times n}$ and $Y \in \mathbb{R}^{m \times n}$. Note that this model also captures the linear autoencoder, when we set $X = Y$ with $m = d$.

We consider learning in the student-teacher setting, where the labels Y are provided by a two-layer linear model with k hidden units. This allows the application of Theorem 3, as the interpolation trajectory can reach the minimum of the objective. However, outside of this realizable setting we can still apply Theorem 3 to the surrogate objective with Y replaced by the minimum achievable target $\hat{Y}$ — this objective aligns with the original at the global minimum.

Now consider a linear interpolation over initial parameters V_0, W_0 and final parameters V_T, W_T, denoted,

$$\mathbf{z}(\alpha) = (V_0 + \alpha(V_T - V_0))\,(W_0 + \alpha(W_T - W_0))\,X.$$

Going forwards, we write $D_1 = (V_T - V_0)$ and $D_2 = (W_T - W_0)$. We first observe that the tangent to this curve is a linear function of α:

$$\mathbf{z}'(\alpha) = (D_1 W_0 + V_0 D_2 + 2\alpha D_1 D_2)\,X. \tag{4}$$

The Gauss length of the interpolated trajectory is given by the length of the projection of the tangent vectors onto the projective space, in this case the sphere with antipodal points identified. Immediately, we note that this line projects onto the sphere as an arc, with end points given by the projection of $\mathbf{z}'(0)$ and $\mathbf{z}'(1)$. Following this, the Gauss length is less than $\pi/2$ exactly when the (vectorized) inner product of the two endpoint is positive (implying the angle between them is at most $\pi/2$). Furthermore, the Gauss length of the interpolation path is at most π for any initial-final parameter pair.

The two endpoints are given by:

$$\mathbf{z}'(0) = (D_1 W_0 + V_0 D_2)\,X \quad \text{and} \quad \mathbf{z}'(1) = (D_1 W_T + V_T D_2)\,X \tag{5}$$

Recall the Kronecker product identity $\mathrm{vec}(AX) = (I \otimes A)\mathrm{vec}(X)$, where $\mathrm{vec}(\cdot)$ indicates column-major vectorization. Then we have,

$$\begin{aligned}
\langle \mathbf{z}'(0), \mathbf{z}'(1) \rangle &= \mathrm{vec}((D_1 W_0 + V_0 D_2)\,X)^\top \mathrm{vec}((D_1 W_T + V_T D_2)\,X) \\
&= \mathrm{vec}(X)^\top \left(I \otimes (D_1 W_0 + V_0 D_2)^\top\right) (I \otimes (D_1 W_T + V_T D_2))\,\mathrm{vec}(X) \\
&= \mathrm{vec}(X)^\top \left(I \otimes \left((D_1 W_0 + V_0 D_2)^\top (D_1 W_T + V_T D_2)\right)\right) \mathrm{vec}(X)
\end{aligned}$$

Now, noting that $I \otimes A$ has the same eigenvalues as A (with increased multiplicity), we have $\langle \mathbf{z}'(0), \mathbf{z}'(1) \rangle > 0$ for all X if and only if all eigenvalues of $(D_1 W_0 + V_0 D_2)^\top (D_1 W_T + V_T D_2)$ are positive.

Proving that the MLI property holds with probability 1. Under the *tabula rasa* assumptions from Saxe et al. (2019) we can prove that the MLI property holds almost surely. The assumptions that underly this setting are as follows.

1. The inputs are whitened ($\frac{1}{n}XX^\top = I$).
2. Initialization is balanced ($V_0 = W_0^\top$).
3. The learning rate of gradient descent is sufficiently small (relative to the largest singular value of the input-output correlation matrix ($\frac{1}{n}YX^\top = USR^\top$).

Under these assumptions, Saxe et al. (2019) prove that,

$$W(t) = Q\sqrt{A(t)}U^\top \text{ and } V(t) = U\sqrt{A(t)}Q^{-1},$$

for some invertible matrix $Q \in \mathbb{R}^{k\times k}$.

Under these dynamics, we have,

$$D_1 = U(\sqrt{A(t)} - \sqrt{A(0)})Q^{-1} \text{ and } D_2 = Q\left(\sqrt{A(t)} - \sqrt{A(0)}\right)U^\top.$$

Thus,

$$\begin{aligned}(D_1W_0 + V_0D_2)^\top &(D_1W_T + V_TD_2) \\ &= 4U\left(\sqrt{A(t)} - \sqrt{A(0)}\right)\sqrt{A(0)}U^\top U\left(\sqrt{A(t)} - \sqrt{A(0)}\right)\sqrt{A(t)}U^\top \\ &= 4U\sqrt{A(0)A(t)}\left(\sqrt{A(t)} - \sqrt{A(0)}\right)^2 U^\top\end{aligned}$$

This matrix is positive definite, and thus has positive eigenvalues. Therefore, the found solution will satisfy the MLI property.

B. Experiment Details

In this section, we provide full details of our experimental setup. For all experiments, we discretize α in the interval $[0, 1]$ using 50 uniform steps to examine the MLI property. When training networks with SGD, we used a momentum coefficient of 0.9 and when training networks with the Adam optimizer, we used $\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 1e-08$. Unless specified otherwise, we used a batch size of 128.

B.1. Image reconstruction experiments

In the image reconstruction experiments, we used deep autoencoders with the ReLU activation function. Our architecture consisted of $784 \to 512 \to H \to 512 \to 784$ units in each respective layer with $H \in \{1, 2, 5, 10, 25, 50, 100\}$. We trained the networks using either SGD with momentum or Adam. Each model was trained for 200 epochs using fixed learning rates in the set $\{0.3, 0.1, 0.03, 0.01, 0.003, 0.001, 0.0003, 0.0001\}$ and batch sizes of 512.

B.2. Image classification experiments

In the image classification experiments, we explored a large number of different architectures. We summarize all setting we explored below.

Multilayer Perceptron. We train fully connected networks with varying widths and depths. For all experiments (except Figure 26), widths were chosen from the set {16, 128, 1024, 2048, 4096} and depth was chosen from {2, 4, 8}. We trained each model using one of SGD, RMSProp, Adam, or KFAC, with fixed learning rates from the set {3.0, 1.0, 0.3, 0.1, 0.03, 0.01, 0.003, 0.001, 0.0003, 0.0001}. We experimented with 3 activation functions: tanh, sigmoid, and ReLU. We trained the networks for 200 epochs both with and without batch normalization.

Convolution Neural Network. We trained Simple CNN, VGG16, VGG19, and ResNet-{18, 20, 32, 44, 50, 56} with and without batch normalization, on CIFAR-10 and CIFAR-100. The Simple CNN had two convolutional layers with a 5×5 kernel followed by a single fully connected layer. We trained the networks with both SGD and the Adam optimizer. For all models, we used an initial learning rate in the set $\{0.3, 0.1, 0.03, 0.01, 0.003, 0.001, 0.0003, 0.0001\}$. For most models, we fixed the learning rate throughout training but for the ResNets we used a waterfall learning rate decay (at 60, 90, and 120 epochs).

For the ResNet experiments without batch normalization, when using the Fixup (Zhang et al., 2019b) or block identity initialization (Goyal et al., 2017) we replaced the batch normalization layers with scale and bias parameters taking the role of the standard batch norm affine transformation. The block identity initialization essentially consists of setting the final scale/bias parameters in each residual block to zero, so that the block computes only the skip connection (with possible down-sampling).

B.3. Language modelling experiments

We trained LSTM and transformer-based architectures on the WikiText-2 dataset with the number layers chosen from the set {2, 3, 4}. We trained the networks for 40 epochs with the initial learning rates in the set {40.0, 30.0, 20.0, 10.0, 1.0, 0.1, 0.01, 0.001}. The learning rates were decayed by a factor of 4 when there was no decrease in the validation loss. We used both SGD and Adam optimizers. We also trained a RoBERTa transfomer-based model (Liu et al., 2019) on the language modelling task on an Esperanto dataset with the Huggingface framework (Wolf et al., 2020), as described in their tutorial[1] and building on a notebook they published[2]. We trained the model from two distinct random initializations for 1 epoch (taking approximately 2 hours on a free Google Colab GPU).

B.4. Experiment specifics

MNIST & Fashion-MNIST batch norm comparison. We describe the experimental set-up used to produce Figure 3 and Figure 25. We trained fully-connected networks whose architecture consisted of $784 \to 1024 \to 1024 \to 10$ units in each layer. We explored ReLU, sigmoid, and tanh activation functions and trained the networks with and without batch norm layers, that when used were inserted after each linear layer (except the last layer). The networks were trained for 200 epochs using fixed learning rates in the set $\{3.0, 1.0, 0.3, 0.1, 0.03, 0.01, 0.003, 0.001\}$ and with either the Adam optimizer or SGD with momentum.

Problem difficulty experiments. For the experiments evaluating problem difficulty (parameter complexity and label corruption), described in Appendix C.11, we trained fully-connected networks on the FashionMNIST dataset. In all cases, the networks used ReLU activations and were trained with batch sizes of at most 512 (depending on dataset size), and for 200 epochs. Learning rates were fixed throughout training. When varying the dataset size, we trained models on random subsets of FashionMNIST with sizes in the set $\{10, 30, 100, 300, 1000, 3000, 10000, 30000, 60000\}$. We evaluated networks trained with learning rates in the set $\{0.03, 0.1, 0.3, 1.0\}$. For the experiments with varying levels of label corruption, we trained fully-connected networks with 2 hidden-layers each of width 1024 and without batch normalization.

C. Extended empirical evaluation

In our exploration of the MLI property, we performed many additional experiments. Generally, we found that turning common knobs of neural network training did not have a significant impact on networks satisfying the MLI property. For example, varying activation functions, loss functions, batch size, regularization, and different forms of initialization had no significant effect on the MLI property. In this section, we present a few of the more interesting additional experiments that we performed.

C.1. Relationship between the MLI property and generalization

We are interested in whether the success/failure of the MLI property impacts the generalization ability of a neural network. To better understand this, we examined the test accuracy of the models we trained and studied the correlation with the

[1] https://huggingface.co/blog/how-to-train

[2] https://colab.research.google.com/github/huggingface/blog/blob/master/notebooks/01_how_to_train.ipynb

	LR:	0.0003	0.001	0.003	0.01	0.03	0.1	0.3
SGD	BN	0.00 (3)	0.25 (4)	0.38 (13)	0.12 (17)	0.06 (17)	0.24 (17)	0.35 (17)
	No BN	-	0.00 (7)	0.00 (14)	0.24 (17)	0.00 (17)	0.25 (16)	0.00 (15)
Adam	BN	0.38 (16)	0.18 (17)	0.35 (17)	0.29 (17)	0.67 (9)	1.00 (6)	1.00 (1)
	No BN	0.00 (13)	0.00 (17)	0.00 (17)	0.62 (8)	0.00 (5)	0.00 (11)	0.00 (6)

Table 3. Proportion of trained CIFAR-10 and CIFAR-100 classifiers (achieving better than 1.0/2.0 training loss respectively) that had non-monotonic interpolations from initialization to final solution. The total number of runs in each bin is displayed in parentheses next to the proportion. A dashed line indicates that no networks achieved the threshold loss.

MLI property on MNIST and CIFAR-10 datasets. Figure 10 shows the relationship between the test accuracy and $\min \Delta$. Note that we considered fully connected networks for MNIST dataset and VGG architectures for CIFAR-10. For the MNIST experiments, configurations that violated the MLI property had an average test accuracy of $96.94(\pm 0.015)$ and those that satisfied the MLI property had the averaged test accuracy of $97.14(\pm 0.018)$. Similarly, for CIFAR-10 experiments, configurations that violated the MLI property had an average test accuracy of $75.83(\pm 0.084)$ and those that satisfied the MLI property had an average test accuracy of $76.99(\pm 0.082)$. Overall, we did not identify a clear pattern between the MLI property and the generalization property of the neural network. At the very least, we ascertain that models violating the MLI property can achieve competitive test accuracy.

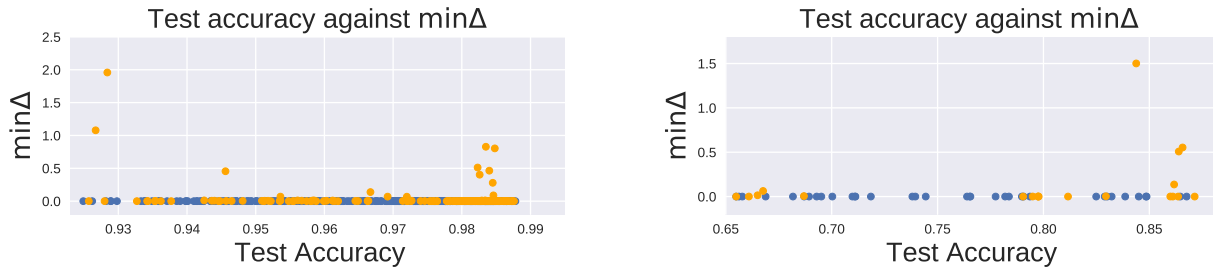

Figure 10. Relationship between test accuracy and non-monotonicity in MNIST (left) and CIFAR10 (right) datasets. **Blue** points represent networks where the MLI property holds and **orange** points are networks where the MLI property fails.

C.2. Impact of large learning rate

In Table 3, we show the proportion of ResNets trained on CIFAR-10 and CIFAR-100 that violated the MLI property. The experimental set up matches that used to produce Tables 2 and Table 4. There is a general trend towards higher learning rates encouraging non-monotonicity though the correlation is weaker than for the MNIST/Fashion-MNIST classifiers.

We provide additional evaluations of large learning rates in Figure 26, where we evaluate the effect of changing network depth and width over varying learning rates. Full details are given in Appendix C.10.

C.3. Impact of optimization algorithm

In Figure 11, we show the training loss over the line connecting the initial and final parameters. We found that adaptive optimizers such as RMSProp and Adam consistently find final solutions that violate the MLI property. To better understand this feature, we compare the distance travelled for all optimization methods. In Figure 12 (left), we show the distance travelled in weight space when trained with SGD and Adam for MNIST & Fashion-MNIST classification tasks. When trained with Adam, the optimizer moved further away from the initialization — confirming the results of Amari et al. (2020). On the other hand, models trained with SGD often traveled less. Moreover, non-monotonic configurations occurred more frequently for networks that travelled far from initialization, suggesting that the non-monotonicity of adaptive optimizers may be due to them encouraging parameters to travel far from initialization.

We also investigated the relationship between non-monotonicity and Gauss length over varying optimizers. In Figure 12 (right), we show the Gauss length for networks trained using SGD and the Adam optimizer. When trained with Adam, on average the interpolation paths have a larger Gauss length and lead to more failures of the MLI property.

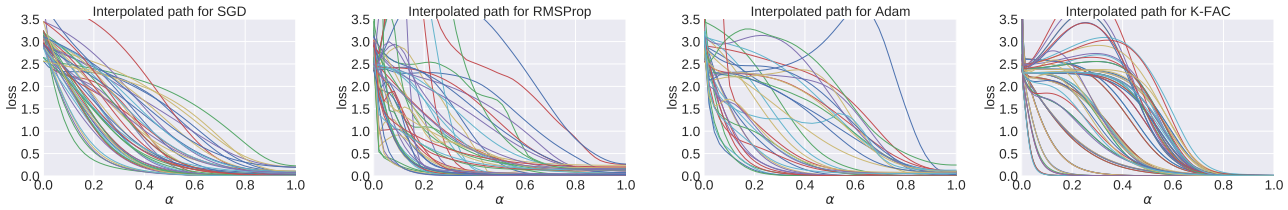

Figure 11. Training loss over the linear interpolation connecting initial and final parameters. Each curve represents a network trained on MNIST & Fashion-MNIST with different optimization algorithms. The MLI property generally holds for networks trained with SGD, but often fails for networks trained with RMSProp, Adam, and K-FAC.

		BN	BN-I	NBN-I	NBN-F
SGD	% Non-monotonic	0.45 (20)	0.00 (16)	0.00 (18)	0.28 (18)
	min Δ	0.055	0	0	0.082
Adam	% Non-monotonic	0.62 (16)	0.00 (15)	0.00 (15)	0.00 (19)
	min Δ	0.487	0	0	0

Table 4. Evaluation of effect of batch normalization, initialization, and choice of optimizer for residual networks trained on CIFAR-100 (achieving at least 2.0 training loss). Full explanation of table is given in main text, Section 4.1.3.

Table 4 contains our evaluation of the MLI property for ResNets trained with different architectures, optimizers, and initialization schemes (as in Table 2 for CIFAR-10 in the main paper). The general trends observed align with those observed on CIFAR-10 in the main paper.

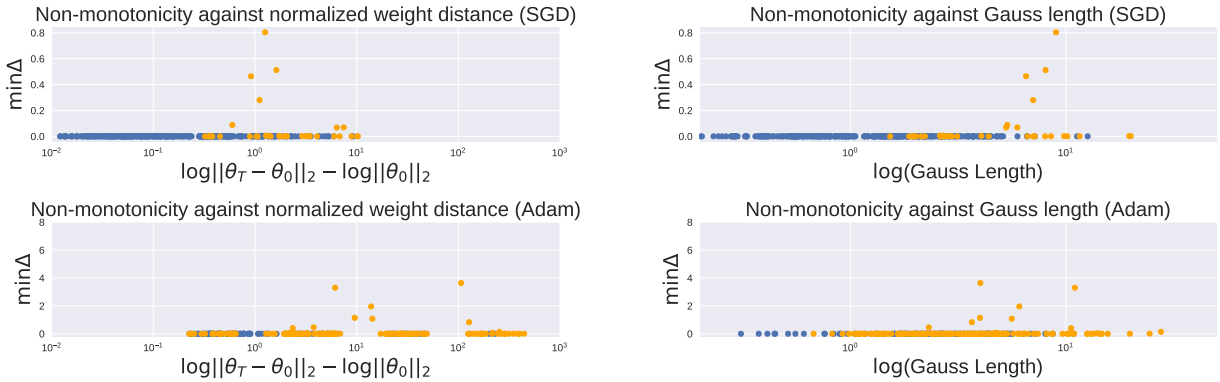

Figure 12. For each MNIST & Fashion-MNIST classifier, we compute the minimum Δ such that the interpolated loss is Δ-monotonic. We plot models trained with SGD and Adam in the top and bottom rows respectively. On the left, we compare the distance moved in the weight space. On the right, we compare the Gauss length of the interpolated network outputs. **Blue** points represent networks where the MLI property holds and **orange** points are networks where the MLI property fails.

C.4. Optimizer Ablations

To get a better understanding of the influence of different optimization algorithms and the effects of moving in weight and function space, we conduct detailed experiments where we switch the optimizer during training. We use an architecture with 2 hidden layers of 1024 units on MNIST. The architecture has tanh activations and no batch normalization. We report the mean and standard error across five random seeds.

In the (SGD $\rightarrow$ Adam) experiments, we train the first $t = \{2, 10, 50\}$ epochs with SGD, using learning rates in the set $\{0.001, 0.003, 0.01, 0.03, 0.1\}$ and finish the training with Adam (with LR 0.001) for $200 - t$ epochs. While using just SGD leads to a monotonic interpolation, switching to Adam made all runs not monotonic. These results are reported in Table 5. Similarly, in Table 5, we report results for (Adam $\rightarrow$ SGD). We switch to SGD with a learning rate of 0.03.

SGD LR \ switch_epoch	2	10	50	None
0.001	7.0822 ± 0.0898	7.3366 ± 0.0034	6.9005 ± 0.0861	0.5341 ± 0.0069
0.003	7.2208 ± 0.1517	7.0211 ± 0.0816	6.8331 ± 0.016	0.5773 ± 0.0069
0.01	7.1144 ± 0.0773	7.2666 ± 0.0943	7.2307 ± 0.1776	0.6939 ± 0.0096
0.03	7.3067 ± 0.026	7.3285 ± 0.0703	6.9953 ± 0.1248	1.1351 ± 0.0522
0.1	7.3783 ± 0.1223	7.4592 ± 0.0714	6.8579 ± 0.1251	2.9144 ± 0.0824
Distance	2	10	50	None
0.001	345.209 ± 2.917	340.998 ± 1.385	303.907 ± 0.809	8.096 ± 0.012
0.003	346.826 ± 0.962	339.54 ± 0.39	303.721 ± 0.314	9.369 ± 0.015
0.01	349.413 ± 0.856	339.056 ± 0.502	302.552 ± 0.368	10.893 ± 0.021
0.03	345.592 ± 0.828	339.428 ± 0.608	304.109 ± 1.031	14.177 ± 0.041
0.1	349.016 ± 1.009	350.103 ± 0.475	326.38 ± 0.628	108.508 ± 2.497

Table 5. Average Gauss length (top) and distance traveled (bottom) for given SGD learning rate and switching to Adam with learning rate 1e-3 during training.

Adam LR \ switch_epoch	2	10	50	None
0.001	1.447 ± 0.036	2.349 ± 0.024	4.472 ± 0.061	7.135 ± 0.048
0.003	3.766 ± 0.052	7.325 ± 0.071	9.365 ± 0.115	9.933 ± 0.097
0.01	6.156 ± 0.266	7.391 ± 0.423	9.873 ± 0.427	10.313 ± 0.231
Distance	2	10	50	None
0.001	22.257 ± 0.113	59.986 ± 0.089	174.958 ± 0.061	350.174 ± 0.704
0.003	58.773 ± 0.565	196.794 ± 0.773	420.652 ± 1.321	659.236 ± 2.785
0.01	185.892 ± 2.26	384.983 ± 13.863	726.063 ± 8.888	1220.432 ± 6.893

Table 6. Average Gauss length (top) and distance traveled (bottom) for given Adam learning rate and switching to SGD with learning rate 0.03 during training.

As in Amari et al. (2020), we again find that Adam leads to much greater distance moved in weight space. Perhaps more surprisingly, the distance moved in weight space and the average Gauss length are largely consistent across different choices of the learning rate for SGD and the epoch at which we switch to Adam. In Table 6, we see that switching from Adam to SGD reduces both the average Gauss length and the distance travelled.

To investigate why Adam is responsible for breaking monotonicity further, we follow the "grafting" experiment described in (Agarwal et al., 2020), where two optimizers are combined by using the step magnitude from the first and step direction from the second. Results where we use the SGD step magnitude (which varies with LR) and Adam direction are shown in 7. All the runs are monotonic, so the direction chosen by Adam is not the primary influence on the optimization trajectory. In contrast, when we use the SGD step direction and the Adam magnitude we observe all runs to be non-monotonic and find the average distance traveled to be 381.65, suggesting that the magnitude of the updates is responsible for breaking monotonicity.

C.5. Additional weight distance experiments

In this section, we investigate the relationship between normalized weight distance and non-monotonicity on image reconstruction (MNIST) and image classification (CIFAR-10 & CIFAR-100) tasks.

In Figure 13, we show the correlation between the distance travelled in parameter space and the smallest Δ such that the

lr \ Optimizer	SGD	Adam
0.001	9.198 ± 0.016	8.096 ± 0.012
0.003	10.932 ± 0.015	9.369 ± 0.015
0.01	12.978 ± 0.013	10.893 ± 0.021
0.03	16.619 ± 0.037	14.177 ± 0.041

Table 7. Average distance traveled where we use the SGD step magnitude and step direction given by SGD or Adam respectively

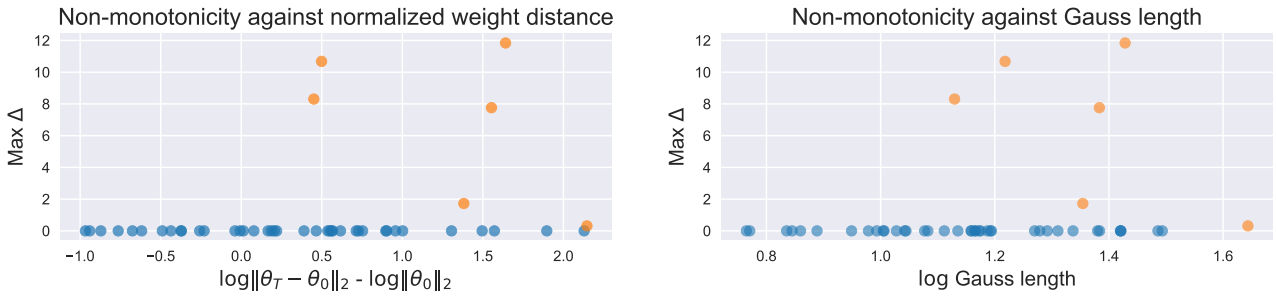

Figure 13. Weight distance (left) and Gauss length (right) against maximum non-monotonic bump height for image reconstruction task. For clarity, **Blue** points represent networks where the MLI property holds and **orange** points are networks where the MLI property fails.

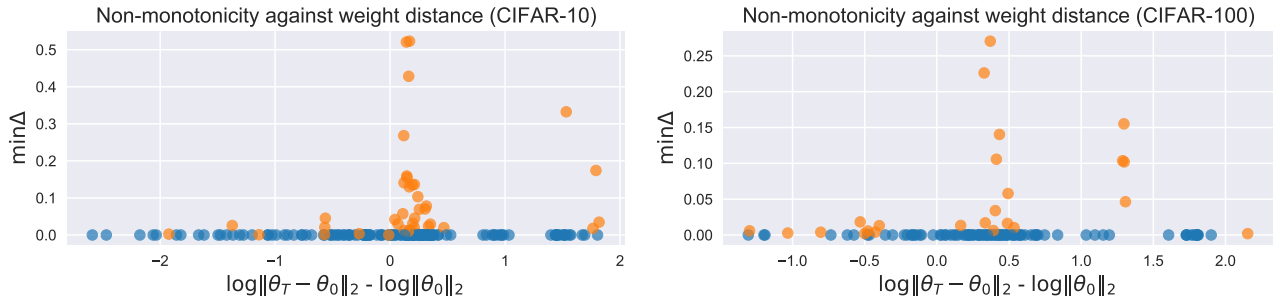

Figure 14. Distance moved in weight space against the minimum Δ such that ResNets trained on CIFAR-10 (left) and CIFAR-100 (right) have Δ-monotonic interpolations from initialization to final parameters. We observe a general trend that larger distance in weight space corresponds to more significant non-monotonicities.

loss interpolation is Δ-monotonic (Definition 1) for image reconstruction task. As expected, a small distance moved (or Gauss length, respectively) leads to monotonic interpolation. And beyond the strict limits of our analysis, we observed that larger weight distances are correlated with non-monotonicity.

In Figure 14, we display the distance moved in weight space against the minimum Δ such that ResNets trained on CIFAR-10 and CIFAR-100 have Δ-monotonic interpolations from initial to final parameters. In general, larger bumps occur at larger distances moved, as in our other experiments.

On the left side of Figure 15, we also show the distance moved in weight space against the minimum Δ on language modelling tasks with LSTM and Transformer architectures. In accordance to our findings on reconstruction and classification tasks, the trained runs that travelled further in parameter space were more likely to violate the MLI property.

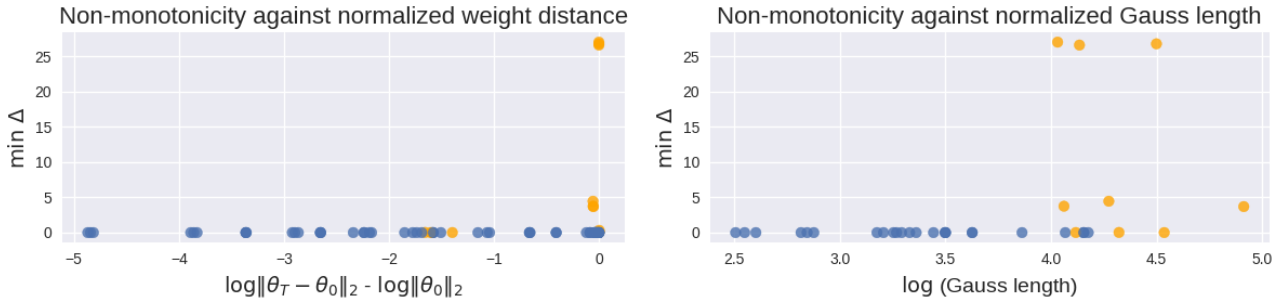

Figure 15. Weight distance (left) and Gauss length (right) against maximum non-monotonic bump height for language modelling task. For clarity, **Blue** points represent networks where the MLI property holds and **orange** points are networks where the MLI property fails.

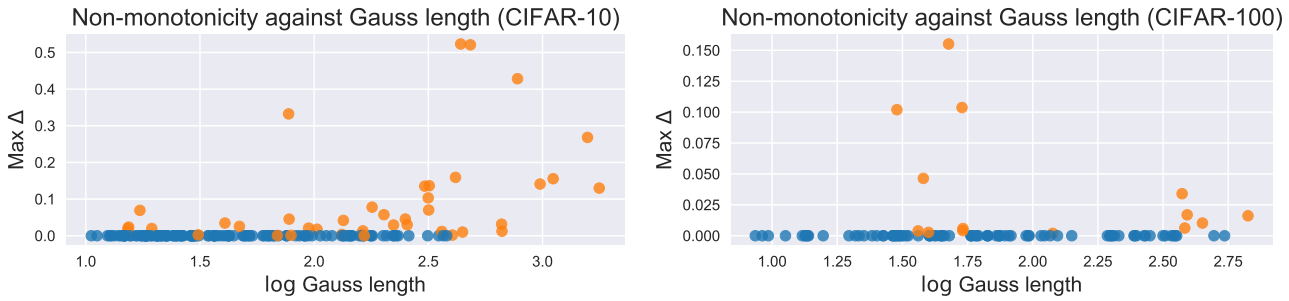

Figure 17. Gauss length of function-space interpolation path against the minimum Δ such that ResNets trained on CIFAR-10 (left) and CIFAR-100 (right) have Δ-monotonic interpolations. At small Gauss lengths, the networks generally satisfy the MLI property while larger bumps in the interpolation path are achieved for interpolations with larger Gauss lengths.

C.6. Additional Gauss length experiments

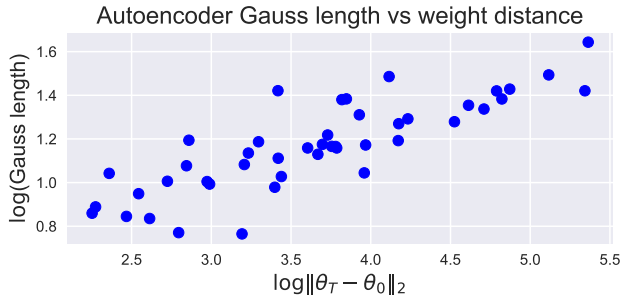

Figure 16. Power law relationship between Gauss length and weight distance for autoencoders trained on MNIST ($R^2 = 0.705$).

In this section, we investigate the relationship between Gauss length and non-monotonicity on image reconstruction and image classification (CIFAR-10 & CIFAR-100) tasks.

In Figure 13, we show the correlation between the Gauss length and the smallest Δ such that the loss interpolation is Δ-monotonic for the image reconstruction task. Similar to the weight distances, a small Gauss length leads to monotonic interpolation. We also observe that larger Gauss length are correlated with the failure of MLI property.

In Figure 14, we display the distance moved in weight space against the minimum Δ such that ResNets trained on CIFAR-10 and CIFAR-100 have Δ-monotonic interpolations from initial to final parameters. In general, larger bumps occur at larger distances moved, as in our other experiments.

C.7. Additional Gauss length vs weight distance

In this section, we investigate the relationship between Guass length and weight distance travelled for image reconstruction and image classification (CIFAR-10 & CIFAR-100) tasks.

In Figure 16, we plot the Gauss length of the interpolation path against the distance moved in weight space for autoencoders trained on MNIST. In this case, as in Figure 7, we observe a clear power-law relationship between the two.

In Figure 18, we plot the Gauss length of the interpolation path against the distance moved in weight space for ResNets trained on CIFAR-10 and CIFAR-100, over varying initialization schemes, optimizers, and the use of batch normalization (as in Tables 2 and 4). In this case, there is not a clear power law relationship but nonetheless a clear positive correlation remains between the Gauss length and the distance moved in weight space.

C.8. Impact of batch normalization

In Figure 19, we compare the distance travelled between models trained with batch normalization and without batch normalization for classifiers trained on the MNIST & Fashion-MNIST datasets. We plot the minimum Δ such that the interpolated loss is Δ-monotonic against the distance moved in weight space. We observe that models trained with batch normalization had a higher variance of distance travelled in weight space compared to models trained without batch normalization. Hence, when batch normalization is used, there are more configurations that travelled further in weight space. Consistent with our prior analysis, configurations that travelled far in parameter space tend to more break the MLI property. This hints that such behaviour of batch normalization can cause more frequent violation of the MLI property.

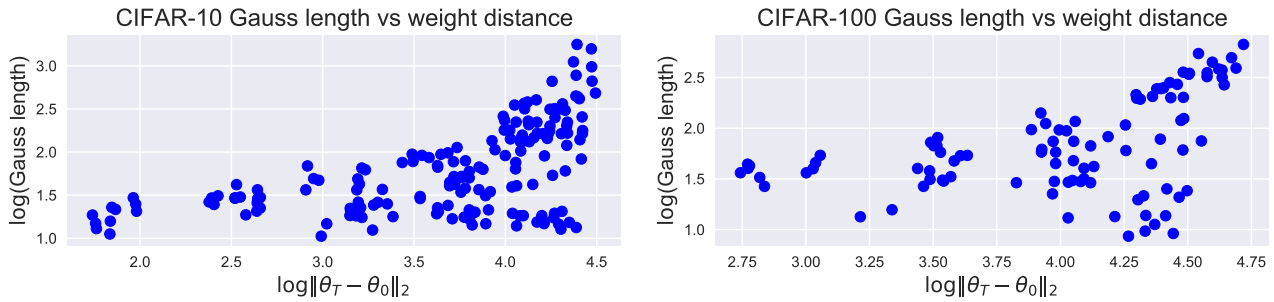

Figure 18. Gauss length of interpolation path against distance moved in weight space for ResNets trained on CIFAR-10 and CIFAR-100. While there is a positive correlation, the goodness of fit is lower for these networks than the MNIST classifiers and autoencoders ($R^2 = 0.399$ for CIFAR-10 and $R^2 = 0.402$ for CIFAR-100).

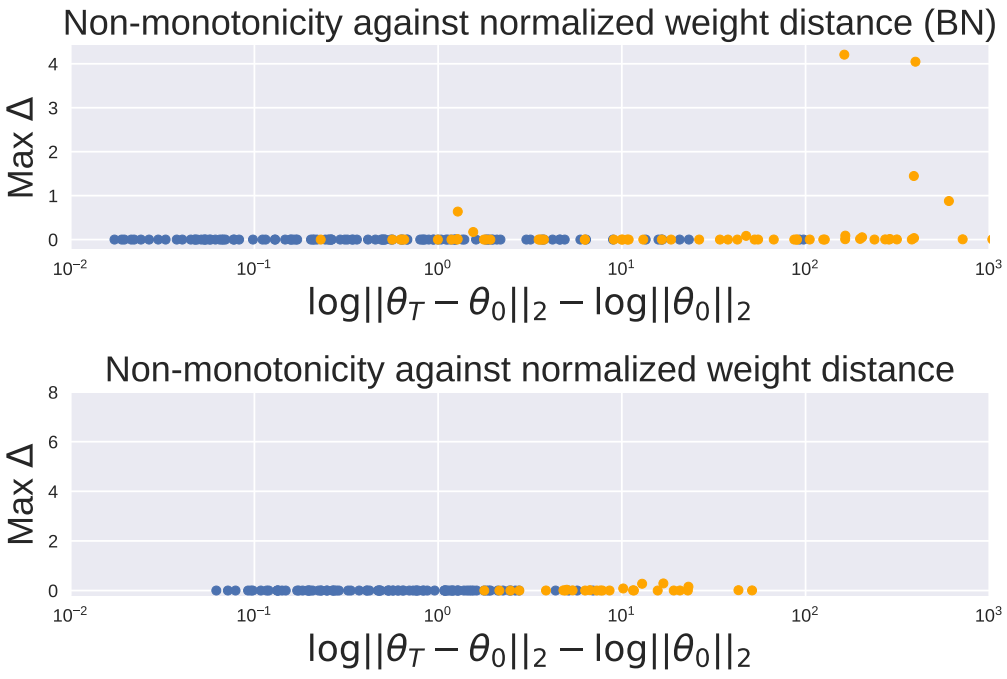

Figure 19. Monotonicity against distance moved in weight space for MNIST & Fashion-MNIST classifiers. **Blue** points represent networks where the MLI property holds and **orange** points are networks where the MLI property fails.

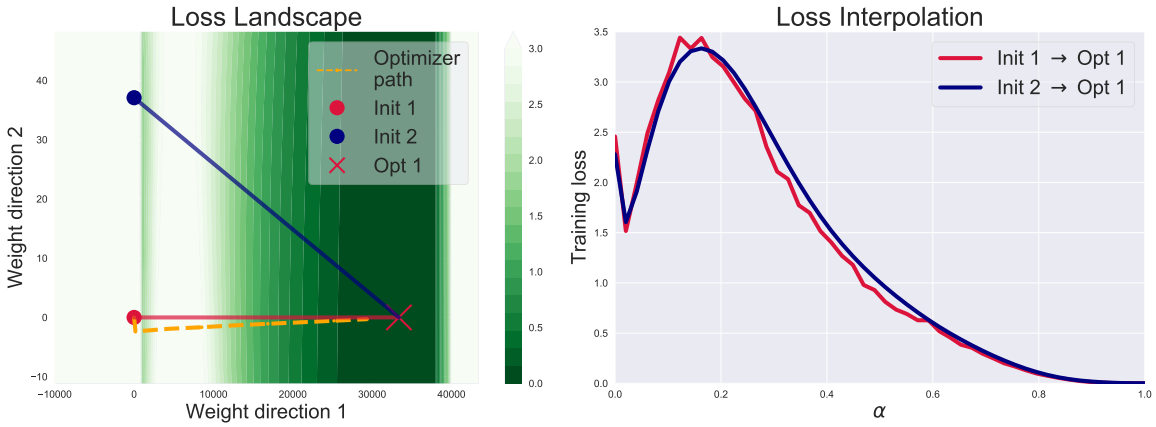

Figure 20. Loss landscape projection of a FashionMNIST classifier that does not satisfy the MLI property. We observe a barrier in the loss followed by a region of extremely flat curvature.

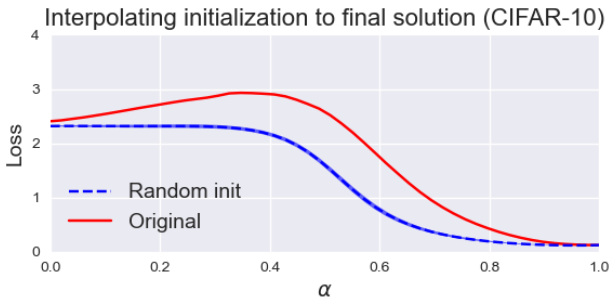

Figure 21. Interpolating from random intializations to SGD solution, with original initialization-solution pair being non-monotonic. The initialization scheme used differs from the one used to train the network, and surprisingly leads to monotonic interpolations.

C.9. Additional loss landscape experiments

Loss landscape for network failing MLI. In Section 4.2.3, we showed loss landscape visualizations for networks that satisfied the MLI property. In Figure 20, we show the 2D projection of the loss landscape for a fully-connected FashionMNIST classifier that does not satisfy the MLI property. In this 2D slice, we observe a wide barrier in the loss landscape followed by a region of extremely flat curvature.

MLI over permutation symmetries. In addition to random initializations, we explored interpolations over the permutation symmetry group of initialization-solution pairs for fully-connected networks on MNIST. In Figure 22, we utilized the fact that adjacent linear layers can be permuted without modifying the output function to randomly permute the initialization and final solution. This leads to different paths through weight space but with the end-points of the interpolation fixed at the original values. We observed that these permutations preserve the MLI property.

Interpolations with different initialization distributions. In Figure 6, we showed that the monotonic (or non-monotonic) interpolations persist across different random initializations for a given final solution. However, it is possible that the monotonicity of the interpolations can change if we modify the initialization scheme. We took the network from the bottom right plot of Figure 6 and chose our initializations according to the scheme described in Goyal et al. (2017) — where the final batch norm layer in each residual block is initialized to be zero so that the network function is close to the identity function. The result is shown in Figure 21, in this case, the random initializations are linearly connected to the solution while random samples from the original initialization distribution are not.

Additional landscape visualizations. In Figure 23, we show additional 2D projections of the loss landscapes for ResNet20v1 networks on CIFAR-10. This confirms results seen elsewhere: linear interpolation between unrelated initial points and optima yield monotonic decreases in training loss and monotonic increases in accuracy.

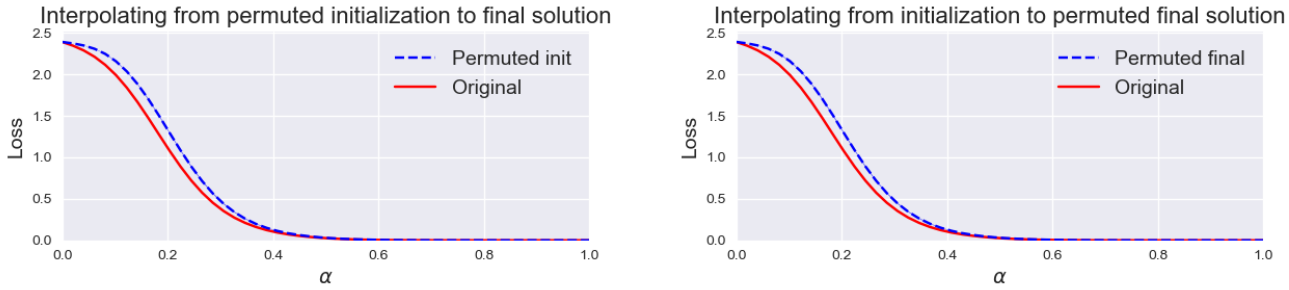

Figure 22. Interpolation loss between initial points and final solution on training set. (Left) random permutations of the initialization are shown. (Right) Random permutations of the solution are shown. Mean loss shown with (± 1) standard deviation as filled region.

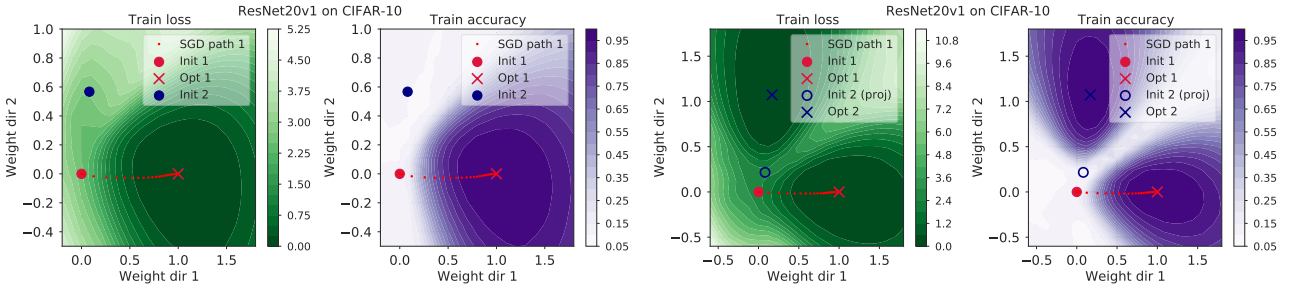

Figure 23. Two-dimensional sections of the weight space for ResNet20v1 trained on CIFAR-10. Left: The plane is defined by 2 initializations (circles) and an optimum (cross) reached from one of them. Right: The plane is defined by an initialization (circle) and two optima (crosses). For both training loss (**green**) and training accuracy (**purple**), interpolations between both minima and optima yield monotonic decreases/increases, respectively.

In Figure 24, we show the same loss landscape projections as displayed in Figure 9. Additionally, here we include the loss over the interpolated paths.

C.10. Additional MNIST results

In Figure 25, we show the full set of network interpolations used to produce Table 1. Overall, we observed a significant effect from introducing batch normalization across all other settings considered, but particularly when the learning rate is large.

Varying depth and hidden size. We explored the effect of varying depth and hidden size on the MLI property. Overall, we did not observe any substantial correlation between these factors and the MLI property (especially, when taking into account implicit effects on the critical learning rate).

In Figure 26, we display heatmaps of $\min \Delta$ as a function of the learning rate, hidden size and depth of fully-connected neural networks trained on MNIST and Fashion-MNIST. Overall, we do not observe any significant effect from changing either the hidden size or the network depth — the learning rate accounts for the dominant changes in the monotonicity of the interpolation. We trained each network with ReLU activations for 200 epochs with batch sizes of 512, using both Adam and SGD and with/without batch normalization. Only those models that achieved a training loss of 0.1 are displayed (cyan patches indicate that no model met these criteria for the corresponding configuration).

C.11. Problem Difficulty

We revisited the conclusion of Goodfellow et al. (2014) that the MLI property holds due to the relative ease of optimization. We explored this question on three fronts. First, we used a fixed network size and varied the number of data points in the dataset. Second, we used a fixed dataset size and varied the number of hidden units in a network of fixed depth (Figure 26). And third, we varied random corruption of labels in the training dataset.

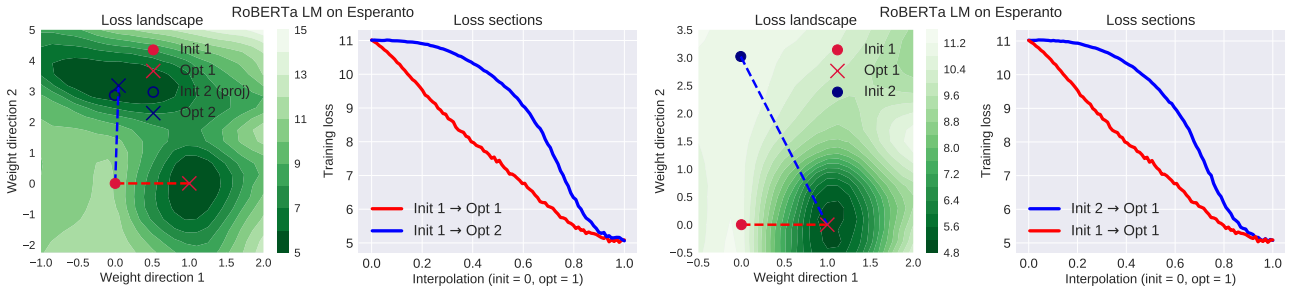

Figure 24. Loss landscapes with loss over linear interpolations between initial and final parameters for RoBERTa trained as language model on Esperanto. Linear interpolation between all pairs leads to monotonic reduction in the loss. Left: The loss over the plane defined by the initial parameters, optimum found by SGD, and an unrelated optimum. Right: The loss over the plane defined by the initial parameters, optimum found by SGD, and an unrelated initialization.

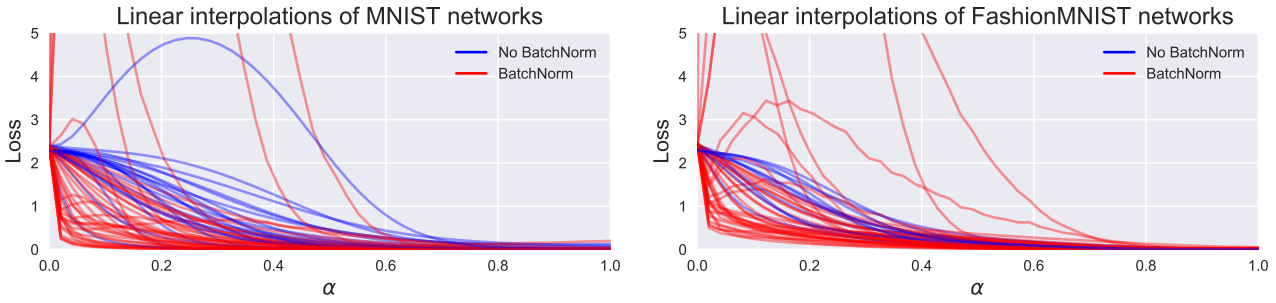

Figure 25. Linear interpolations for MNIST (top) and Fashion-MNIST (bottom). Different curves represent trained networks with varying activation function, learning rate, choice of optimizer, and batch normalization. All networks achieve at most 0.1 final training loss.

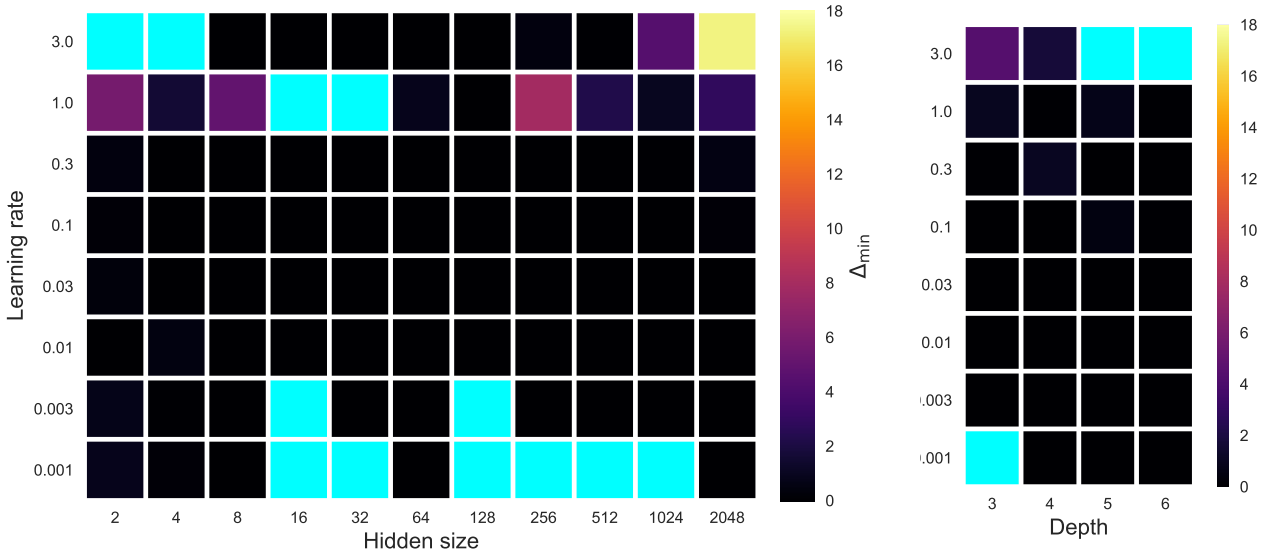

Figure 26. Heatmaps of the average $\min \Delta$ as a function of the learning rate, hidden size and network depth for fully-connected networks trained on MNIST and FashionMNIST. On the left, depth 3 networks with varying hidden sizes are compared. On the right, networks with hidden size 1024 are compared over varying depth. **Cyan** patches indicate that no model with the given configuration achieved a minimum training loss of 0.1.

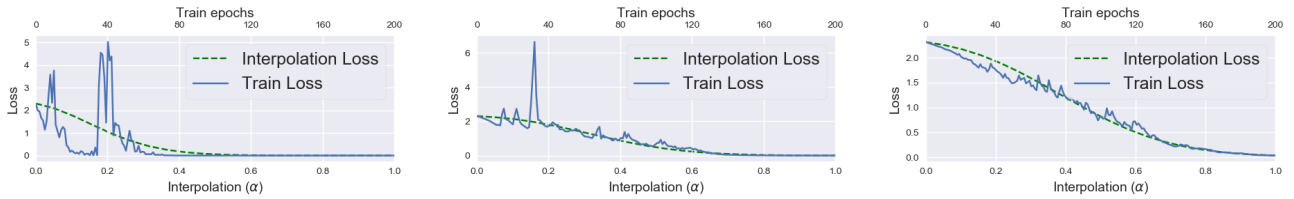

Figure 27. Linear interpolations (green) for neural networks trained on varying dataset sizes (30, 300, 3000 from left-to-right), with loss during training overlaid (blue). Even when the training dynamics are unstable and highly non-linear, the interpolation produces a smooth monotonic curve.

Dataset size. We trained fully-connected networks on the Fashion-MNIST dataset using SGD with a learning rate of 0.1. The networks had a single hidden layer with 1000 hidden units, and we varied the dataset size from 10 up to the full size 60000. Figure 27 shows the linear interpolation trained on varying dataset sizes. We observed that even when the training dynamics are unstable and highly non-linear, the interpolation is still monotonically decreasing.

MLI vs. label corruption. When the dataset is sufficiently simple, the learning problem is easy and SGD consistently finds solutions with the MLI property. To explore this hypothesis, we trained neural networks with label corruption. We trained a neural network with two hidden layers each with 1024 units (more details can be found at Appendix B.4). The labels were corrupted by uniformly sampling labels for some proportion of the data points. We varied the label corruption from 0% to 100% in 2.5% intervals. We varied the proportion of label corruption from 0% up to 100%. At all levels of label corruption, the MLI property persisted. One possible explanation for this result follows from the fact that logit gradients cluster together by logit index — even for inputs belonging to different true classes (Fort & Ganguli, 2019). This provides an explanation for gradient descent exploring a low dimensional subspace relative to the parameter space. Therefore, corrupting the label will not disrupt this clustering at initialization and, as empirically verified, is unlikely to prevent the MLI property from holding.

C.12. Learning Dynamics

Lewkowycz et al. (2020) observed a region of critical large learning rates wherein gradient descent breaks out of high-curvature regions at initialization and explores regions of high-loss before settling in a low-loss region with lower curvature. We might expect that such trajectories lead to initialization-solution pairs that do not satisfy the MLI property. On one hand, in Figure 27, we observed several runs where SGD is seen to overcome large barriers but the MLI property holds. However, in Figure 20 we observe a projection of the loss landscape which aligns with the qualitative description of the catapult phase: a barrier in the loss, with SGD settling in a region of much lower curvature. Overall, we consider our findings inconclusive on this front.

C.13. MLI on held-out data

In this work, we are primarily concerned with better understanding of the interaction between the MLI property and the training loss. Therefore, all of the results that we have reported are based on statistics computed over the training set. However, the same observations also hold generally when evaluating using held-out data (up to overfitting effects). This was confirmed by Goodfellow et al. (2014), and in this section, we provide a short qualitative study verifying this for the settings that we have studied.

Image reconstruction. In Figure 28 (left two plots), we compare the loss interpolations on the training set and test set for two trained autoencoders. In the first plot, the network satisfies the MLI property but in the second it does not. In both cases, the test loss interpolation closely follows the training loss.

MNIST Classifiers. The third and fourth plots in Figure 28 show the train and test loss interpolations for fully-connected MNIST classifiers. In this case, the test loss increases towards the end of the interpolation path while the training loss stays small. This happens because the network becomes over-confident in its predictions and pays a larger cost for misclassification on the test-set (even though the accuracy remains the same). This observed behaviour is one reason why we favour exploration of the training loss throughout our work. Despite this, we do still observe the test loss following the general shape of the training loss for most of the interpolation path.

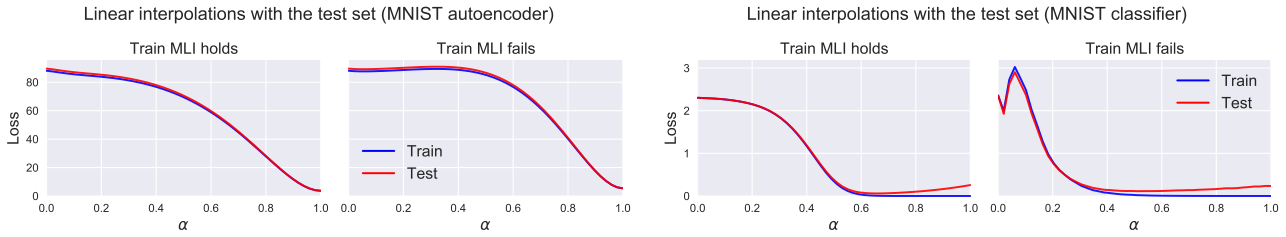

Figure 28. Comparing loss interpolations on the train and test set. In the first and second plots, fully-connected autoencoders trained on MNIST are evaluated that do/don't satisfy the MLI property (respectively). The third and fourth plots display fully-connected MNIST classifiers.

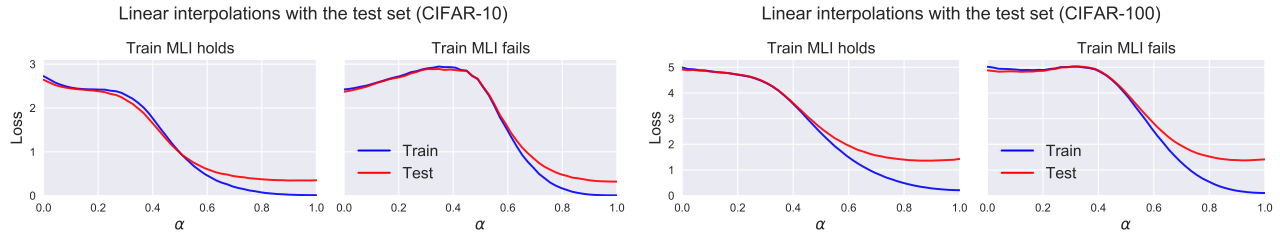

Figure 29. Comparing loss interpolations on the train and test set. In the first and second plots, ResNets trained on CIFAR-10 are evaluated that do/don't satisfy the MLI property (respectively). The third and fourth plots display interpolation plots for ResNets trained on CIFAR-100.

CIFAR-10 & CIFAR-100 Classifiers. In Figure 29, we compare the loss interpolations on the training set and test set for ResNets trained on CIFAR-10 and CIFAR-100. The first two plots show CIFAR-10 classifiers with the third and fourth plot showing CIFAR-100 classifiers. The first and third plots show networks that satisfy the MLI property on the training loss, while the second and fourth show networks that fail to satisfy the MLI property on the training loss. As with the MNIST classifiers, we observe that the test loss has a tendency to increase towards the end of the interpolation path (while following the overall trend of the training loss).

D. Additional Theoretical Analysis

In this section, we present additional theoretical analysis of the MLI property.

D.1. Wide neural networks

In this section, we prove that sufficiently wide fully-connected networks satisfy the MLI property. To do so, we lean on prior analysis from Lee et al. (2019). We assume that the fully-connected network has the following layer sizes $d \to m \to \ldots m \to k$, with $m \to \infty$. We also assume our loss function is mean-squared error,

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^{n} \| f_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathbf{y}_i \|^2.$$

Assumptions. We borrow the setting established by Lee et al. (2019) that consists of four assumptions.

1. The widths of the hidden layers are identical (as stated above).
2. The neural tangent kernel, $\frac{1}{n} J(\boldsymbol{\theta})^\top J(\boldsymbol{\theta})$, is full-rank with finite singular values. I.e.,

$$0 < \lambda_{\min} \left(\frac{1}{n} J(\boldsymbol{\theta})^\top J(\boldsymbol{\theta}) \right) \leq \lambda_{\max} \left(\frac{1}{n} J(\boldsymbol{\theta})^\top J(\boldsymbol{\theta}) \right) < \infty.$$

   Further, we define $\eta_{\text{critical}} := 2/(\lambda_{\min} + \lambda_{\max})$

3. The training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ is contained in a compact set and contains no duplicate inputs.
4. The activation function, ϕ, in the network satisfies the following,

$$|\phi(0)| < \infty, \;\; \|\phi'\|_\infty < \infty, \;\; \sup_{\mathbf{x} \neq \tilde{\mathbf{x}}} \frac{|\phi'(\mathbf{x}) - \phi'(\tilde{\mathbf{x}})|}{\mathbf{x} - \tilde{\mathbf{x}}} < \infty$$

Background. We utilize two results from Lee et al. (2019). The first of which bounds the Jacobian matrix in Frobenius norm about initialization.

Lemma 5. *(Locally Lipschitz Jacobian [Lemma 1 (Lee et al., 2019)]) Assume conditions 1-4 above. There is a $K > 0$ such that for every $C > 0$, with high probability over random initialization,*

$$\frac{1}{\sqrt{m}} \|J(\boldsymbol{\theta})\|_F \leq K,$$
$$\frac{1}{\sqrt{m}} \|J(\boldsymbol{\theta}) - J(\boldsymbol{\theta}')\|_F \leq K \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2,$$

for all $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ such that $\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\| \leq C m^{-1/2}$ and $\|\boldsymbol{\theta}' - \boldsymbol{\theta}_0\| \leq C m^{-1/2}$.

In words, Lemma 5 guarantees that the Frobenius norm of the Jacobian is close to initialization as width grows and that it does not vary too quickly. The second of these two constraints also guarantees that the norm of the network Hessian is bounded (by considering $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ arbitrarily close).

The second result that we borrow provides a high-probability guarantee that infinitely wide neural networks find solutions near to their initialization (the lazy training regime (Chizat et al., 2018)).

Lemma 6. *(Lazy training [Theorem G.1 (Lee et al., 2019)]) Assume conditions 1-4 above. For all $\delta > 0$ and $\eta_0 < \eta_{critical}$, there exists $M \in \mathbb{N}$, $R_0 > 0$, and $K > 1$ such that for every $m > M$, with probability at least $1 - \delta$ over random initialization, gradient descent with learning rate $\eta = \eta_0 / m$ applied for T steps satisfies,*

$$\|\boldsymbol{\theta}_T - \boldsymbol{\theta}_0\|_2 \leq \frac{3 K R_0}{\lambda_{\min}} m^{-1/2}.$$

MLI for infinite width networks. From the above, we can prove that in the limit of infinite width, gradient descent with a suitably small learning rate finds a solution that is linearly connected to the initialization.

Intuitively, this result holds as in a region near a minimum the objective is locally convex. As the width of the network grows, the minimum found by gradient descent becomes arbitrarily close to initialization and thus the linear interpolation is acting over a convex function.

For completeness, we first provide a simple proof that linear interpolations satisfy the MLI property in convex loss landscapes. The result itself follows from standard techniques presented in, for example, Boyd et al. (2004).

Lemma 7 (Linearity and convexity gives MLI)**.** *Let $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ be a convex, differentiable loss function. Further, let $\boldsymbol{\theta}^* \in \arg\min \mathcal{L}$. Then, for all $\boldsymbol{\theta}_0 \in \mathbb{R}^d$, $g(\alpha) := \mathcal{L}(\boldsymbol{\theta}_0 + \alpha(\boldsymbol{\theta}^* - \boldsymbol{\theta}_0))$ is monotonically decreasing for $\alpha \in [0, 1)$.*

Proof. We have that $g(\alpha)$ is also a convex, differentiable function. Therefore, using the first-order convexity condition on g,

$$g'(\alpha) \leq \frac{g(1) - g(\alpha)}{1 - \alpha} \leq 0$$

□

We now proceed with the main result of this section.

Theorem 8 (Wide networks satisfy the MLI Property)**.** *Assume conditions 1-4 above. For all $\delta > 0$ and $\eta_0 < \eta_{critical}$, there exists $M \in \mathbb{N}$ such that for every $m > M$, with probability at least $1 - \delta$ over random initialization, gradient descent with learning rate $\eta = \eta_0 / m$ satisfies,*

$$\mathcal{L}(\boldsymbol{\theta}_{\alpha_2}) - \mathcal{L}(\boldsymbol{\theta}_{\alpha_1}) \leq 0,$$

for all $\alpha_2 > \alpha_1 \in [0, 1)$.

Proof. For brevity, we write $\Delta\boldsymbol{\theta} = \boldsymbol{\theta}_T - \boldsymbol{\theta}_0$, with $\boldsymbol{\theta}_\alpha = \boldsymbol{\theta}_0 + \alpha\Delta\boldsymbol{\theta}$. Our approach is to linearize the loss in function-space and show that all remaining terms are quadratic in $\Delta\boldsymbol{\theta}$ and so are dominated by the linear terms for a sufficiently wide network.

We begin by considering the Taylor series of $\mathcal{L}(\boldsymbol{\theta}_\alpha)$ about $\boldsymbol{\theta}_0$, using the Lagrange form of the remainder,

$$\mathcal{L}(\boldsymbol{\theta}_\alpha) = \mathcal{L}(\boldsymbol{\theta}_0) + \alpha\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_0)^\top\Delta\boldsymbol{\theta} + \frac{1}{2}\alpha^2\Delta\boldsymbol{\theta}^\top\nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\mathbf{x}_i;\bar{\boldsymbol{\theta}}_\alpha)\Delta\boldsymbol{\theta}, \tag{6}$$

$$= \mathcal{L}(\boldsymbol{\theta}_0) + \frac{\alpha}{2n}\sum_{i=1}^{n}(f(\mathbf{x}_i;\boldsymbol{\theta}_0) - \mathbf{y}_i)^\top J(\mathbf{x}_i;\boldsymbol{\theta}_0)\Delta\boldsymbol{\theta} + \frac{1}{2}\alpha^2\Delta\boldsymbol{\theta}^\top\nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\mathbf{x}_i;\bar{\boldsymbol{\theta}}_\alpha)\Delta\boldsymbol{\theta}, \tag{7}$$

for some $\bar{\boldsymbol{\theta}}_\alpha$ on the line $[\boldsymbol{\theta}_0, \boldsymbol{\theta}_\alpha]$. Now, noting that the Hessian of f with respect to $\boldsymbol{\theta}$ is a third-order tensor, we can utilize the integral form of the Taylor expansion to write,

$$(J(\mathbf{x}_i;\boldsymbol{\theta}_0)\Delta\boldsymbol{\theta})_j = f(\mathbf{x}_i;\boldsymbol{\theta}_T)_j - f(\mathbf{x}_i;\boldsymbol{\theta}_0)_j - \frac{1}{2}\Delta\boldsymbol{\theta}^\top\left(\int_0^1 \frac{\partial^2 f_j}{\partial\boldsymbol{\theta}^2}(\mathbf{x}_i;\boldsymbol{\theta}_{\alpha'})d\alpha'\right)\Delta\boldsymbol{\theta}, \tag{8}$$

where the j subscript notation indicates vector indexing. Collecting terms, we have

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}_\alpha) - \mathcal{L}(\boldsymbol{\theta}_0) =& \frac{1}{2n}\sum_{i=1}^{n}\Big[\alpha(f(\mathbf{x}_i;\boldsymbol{\theta}_0) - \mathbf{y}_i)^\top(f(\boldsymbol{\theta}_T) - f(\boldsymbol{\theta}_0)) + \frac{1}{2}\alpha^2\Delta\boldsymbol{\theta}^\top\nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\mathbf{x}_i;\bar{\boldsymbol{\theta}}_\alpha)\Delta\boldsymbol{\theta},\\ &- \frac{1}{2}\alpha\sum_{j=1}^{k}(f(\mathbf{x}_i;\boldsymbol{\theta}_0) - \mathbf{y}_i)_k\Delta\boldsymbol{\theta}^\top\left(\int_0^1 \frac{\partial^2 f_k}{\partial\boldsymbol{\theta}^2}(\mathbf{x}_i;\boldsymbol{\theta}_{\alpha'})d\alpha'\right)\Delta\boldsymbol{\theta}\Big].\end{aligned}$$

Now, noting that $\mathcal{L}(\boldsymbol{\theta}_{\alpha_2}) - \mathcal{L}(\boldsymbol{\theta}_{\alpha_1}) = (\mathcal{L}(\boldsymbol{\theta}_{\alpha_2}) - \mathcal{L}(\boldsymbol{\theta}_0)) - (\mathcal{L}(\boldsymbol{\theta}_{\alpha_1}) - \mathcal{L}(\boldsymbol{\theta}_0))$, we have

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}_{\alpha_2}) - \mathcal{L}(\boldsymbol{\theta}_{\alpha_1}) =& \frac{1}{2n}\sum_{i=1}^{n}\Big[(\alpha_2 - \alpha_1)(f(\mathbf{x}_i;\boldsymbol{\theta}_0) - \mathbf{y}_i)^\top(f(\boldsymbol{\theta}_T) - f(\boldsymbol{\theta}_0))\\ &+ \frac{1}{2}\Delta\boldsymbol{\theta}^\top\left(\alpha_2^2\nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\mathbf{x}_i;\bar{\boldsymbol{\theta}}_{\alpha_2}) - \alpha_1^2\nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\mathbf{x}_i;\bar{\boldsymbol{\theta}}_{\alpha_2})\right)\Delta\boldsymbol{\theta},\\ &- \frac{1}{2}(\alpha_2 - \alpha_1)\sum_{j=1}^{k}(f(\mathbf{x}_i;\boldsymbol{\theta}_0) - \mathbf{y}_i)_k\Delta\boldsymbol{\theta}^\top\left(\int_0^1 \frac{\partial^2 f_k}{\partial\boldsymbol{\theta}^2}(\mathbf{x}_i;\boldsymbol{\theta}_{\alpha'})d\alpha'\right)\Delta\boldsymbol{\theta}\Big].\end{aligned}$$

The first term in the sum is negative as $\mathcal{L}$ is convex in f (and $\alpha_2 > \alpha_1$). It remains to show that the other terms behave asymptotically like $\|\Delta\boldsymbol{\theta}\|^2$. First, notice that we can decompose the Hessian of the loss as follows,

$$\nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\mathbf{x}_i;\boldsymbol{\theta}) = J(\mathbf{x}_i;\boldsymbol{\theta})^\top J(\mathbf{x}_i;\boldsymbol{\theta}) + \sum_{j=1}^{k}(f(\mathbf{x}_i;\boldsymbol{\theta}) - \mathbf{y}_i)_j\frac{\partial^2 f_j}{\partial\boldsymbol{\theta}^2}(\mathbf{x}_i;\boldsymbol{\theta}) \tag{9}$$

Furthermore, by Lemma 6, there exists an $M' \in \mathbb{N}$ such that for all $m > M'$ we have $\|\Delta\boldsymbol{\theta}\| \leq O(m^{-1/2})$ with probability at least $1 - \delta$. Under this event, we can apply Lemma 5 to guarantee that the average Jacobian and Hessian norms are bounded about initialization:

$$\frac{1}{n}\sum_{i=1}^{n}\|J(\mathbf{x}_i;\boldsymbol{\theta})\|_F^2 < \infty \quad \text{and} \quad \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k}\left\|\frac{\partial^2 f_j}{\partial\boldsymbol{\theta}^2}(\mathbf{x}_i;\boldsymbol{\theta})\right\|_F^2 < \infty.$$

Therefore, there exists an $M \geq M'$, such that for all $m > M$ the negative first-order term dominates the second order terms. Under the $1-\delta$ probability event, this guarantees that the loss is monotonically decreasing along the linear interpolation. □

D.2. A Noisy Quadratic Model

The noisy quadratic model (NQM) (Schaul et al., 2013; Wu et al., 2018; Zhang et al., 2019a) serves as a useful guide for understanding the effects of stochasticity in asymptotic neural network training. Indeed, Zhang et al. (2019a) demonstrate

that the NQM makes predictions that are aligned with experimental results on deep neural networks. Using this model, we can provide an explanation for one possible cause of non-monotonicity: an inflection point of the interpolation curve with positive second derivative close to $\alpha = 1$. Intuitively, we can imagine a bowl-shaped loss surface where the final parameters lies on the opposite side of the optima relative to the initialization. This non-monotonicity is likely to occur when training with smaller batch sizes and/or using larger (fixed) learning rates.

Let our loss function be as follows:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{\theta}^\top \mathbf{K}\boldsymbol{\theta}, \tag{10}$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ and $\mathbf{K} \in \mathbb{R}^{d\times d}$. The optimization algorithm receives stochastic gradients $\mathbf{K}\boldsymbol{\theta} + \mathbf{c}$, where $\mathbf{c} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$. Consider the iterates $\{\boldsymbol{\theta}_i\}_{i=0}^\top$ produced by gradient descent. With a sufficiently small learning rate, the expected value of the iterate converges i.e. $\lim_{t\to\infty} \mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_t)] = 0$.

Also consider interpolating between arbitrary $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$. The loss along the interpolation direction is $\mathcal{L}(\boldsymbol{\theta}_1 + \alpha(\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1))$. We compute the derivative with respect to α:

$$\frac{\partial \mathcal{L}}{\partial \alpha}(\boldsymbol{\theta}_1 + \alpha(\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1)) = \frac{\partial}{\partial \alpha}\left[\frac{1}{2}(\boldsymbol{\theta}_1 + \alpha(\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1))^\top \mathbf{K}(\boldsymbol{\theta}_1 + \alpha(\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1))\right] \tag{11}$$

$$= (\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1)^\top \mathbf{K}(\boldsymbol{\theta}_1 + \alpha(\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1)) \tag{12}$$

Hence, the loss is monotonically decreasing if, for all $\alpha \in [0, 1]$,

$$(\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1)^\top \mathbf{K}(\boldsymbol{\theta}_1 + \alpha(\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1)) < 0 \tag{13}$$

In the one dimension case, this equation is saying that interpolation is non-monotonic when $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are on the opposite side of the minima. More generally, note that because $\frac{\partial \mathcal{L}}{\partial \alpha}$ is linear in α, the interpolation is monotonically decreasing if and only if both of these conditions at the endpoints are satisfied:

$$(\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1)^\top \mathbf{K}\boldsymbol{\theta}_1 < 0 \tag{14}$$

$$(\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1)^\top \mathbf{K}\boldsymbol{\theta}_2 < 0 \tag{15}$$

These two conditions correspond to a negative derivative with respect to α at $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$. Since we choose a learning rate so that the loss decreases in expectation (and hence the derivative is anti-aligned with $\mathbf{c}_2 - \mathbf{c}_1$ at initialization), it suffices to check just the second condition.

We simulate learning in this model to measure the effect of stochasticity under varying learning rates on the MLI property. As in Zhang et al. (2019a), we use $\boldsymbol{\theta}_1 := \boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\mathbf{K} = diag\{1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{d}\}$. As $t \to \infty$, the point $\boldsymbol{\theta}_2 := \boldsymbol{\theta}_T \sim \mathcal{N}(\mathbf{0}, \eta\mathbf{K})$, where η is the final learning rate and the random variable comes from the noise in the gradient. Through empirical simulations, we verify that this is approximately a symmetric distribution about 0, so the probability we have monotonic interpolation is roughly $\frac{1}{2}$. This is empirically verified in Figure 30. A smaller learning rate means that the distribution of $(\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1)^\top \mathbf{K}\boldsymbol{\theta}_2$ has less variance. Because we discretize α when we check for MLI, we have $P((\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1)^\top \mathbf{K}\mathbf{c}_2) < \epsilon)$ increases as the learning rate decreases for some small ϵ.

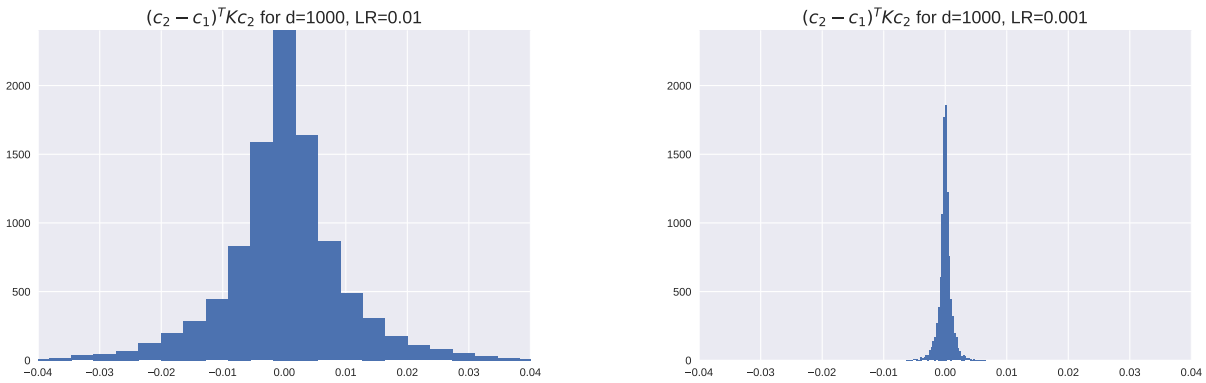

Figure 30. For smaller learning rates, the standard deviation of the distribution goes down. Hence the probability that $P((\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1)^\top \mathbf{K}\mathbf{c}_2) < \epsilon$ for some small ϵ goes up (indicating non-monotonicity from a inflection point near the optima that is hard to detect). We use an equal number of bins in both plots.