

Stiffness: A New Perspective on Generalization in Neural Networks

Stanislav Fort^{1,2} Paweł Krzysztof Nowak¹ Srinu Narayanan¹

Abstract

We investigate neural network training and generalization using the concept of stiffness. We measure how stiff a network is by looking at how a small gradient step on one example affects the loss on another example. In particular, we study how stiffness varies with 1) class membership, 2) distance between data points (in the input space as well as in latent spaces), 3) training iteration, and 4) learning rate. We empirically study the evolution of stiffness on MNIST, FASHION MNIST, CIFAR-10 and CIFAR-100 using fully-connected and convolutional neural networks. Our results demonstrate that stiffness is a useful concept for diagnosing and characterizing generalization. We observe that small learning rates lead to initial learning of more specific features that do not translate well to improvements on inputs from all classes, whereas high learning rates initially benefit all classes at once. We measure stiffness as a function of distance between data points and observe that higher learning rates induce positive correlation between changes in loss further apart, pointing towards a regularization effect of learning rate. When training on CIFAR-100, the stiffness matrix exhibits a coarse-grained behavior suggestive of the model’s awareness of super-class membership.

1. Introduction

Neural networks are a class of highly expressive function approximators that proved to be successful in approximating solutions to complex tasks across many domains such as vision, natural language understanding, and game-play. They have long been recognized as universal function approximators (Hornik et al., 1989; Cybenko, 1989; Leshno et al., 1993). The specific details that lead to their expressive

power have recently been studied in Montúfar et al. (2014); Raghu et al. (2017); Poole et al. (2016). Empirically, neural networks have been extremely successful at generalizing to new data despite their over-parametrization for the task at hand, as well as their proven ability to fit arbitrary random data perfectly Zhang et al. (2016); Arpit et al. (2017).

The fact that gradient descent is able to find good solutions given the highly over-parametrized family of functions has been studied theoretically in Arora et al. (2018) and explored empirically in Li et al. (2018), where the effective low-dimensional nature of many common learning problems is shown. Fort & Scherlis (2018) extends the analysis in Li et al. (2018) to demonstrate the role of initialization on the effective dimensionality.

Du et al. (2018a) and Du et al. (2018b) use a Gram matrix to study convergence in neural network empirical loss. Pennington & Worah (2017) study the concentration properties of a similar covariance matrix formed from the output of the network. Both concepts are closely related to our definition of stiffness.

To explain the remarkable generalization properties of neural networks, it has been proposed (Rahaman et al., 2018) that the function family is biased towards low-frequency functions. The role of similarity between the neural network outputs to similar inputs has been studied in Schoenholz et al. (2016) for random initializations and explored empirically in Novak et al. (2018).

1.1. Our contribution

In this paper, we study generalization through the lens of *stiffness*. We measure how *stiff* a neural network is by analyzing how a small gradient step based on one input affects the loss on another input. Mathematically, if the gradient of the loss at point X_1 with respect to the network weights is $\nabla_W \mathcal{L}(X_1) = \vec{g}_1$, and the gradient at point X_2 is \vec{g}_2 , we define stiffness $\propto \vec{g}_1 \cdot \vec{g}_2$. We specifically focus on the sign of $\vec{g}_1 \cdot \vec{g}_2$ which captures the resistance of the functional approximation learned to deformation by gradient steps. We find the concept of stiffness useful in diagnosing and characterizing generalization. As a corollary, we use stiffness to characterize the regularization power of learning rate, and show that higher learning rates bias the functions learned towards higher stiffness.

¹Google AI, Zurich, Switzerland ²Google AI Resident (<https://g.co/airesidency>). Correspondence to: Stanislav Fort <stanislavfort@google.com>, Paweł Krzysztof Nowak <pawel-now@google.com>.

We show that stiffness is directly related to generalization when evaluated on the held-out validation set. Stiff functions are less flexible and therefore less prone to overfitting to the particular details of a dataset. We explore the concept of stiffness for fully-connected (FC) and convolutional neural networks (CNN) on 4 classification datasets (MNIST, FASHION MNIST, CIFAR-10, CIFAR-100) and on synthetic data comprising spherical harmonics. We focus on how stiffness between data points varies with their 1) class membership, 2) distance between each other (both in the space of inputs as well as in latent spaces), 3) training iteration, and 4) the choice of learning rate.

We observed the stiffness between validation set data points based on their class membership and noticed a clear evolution towards high stiffness within examples of the same class, as well as between different classes as the model trains. We diagnose and characterize the class-dependent stiffness matrix for fully-connected and convolutional neural networks on the datasets mentioned above in different stages of training. We observe the stiffness between inputs to regress to zero with the onset of overfitting, demonstrating the clear connection to generalization.

The choice of learning rate effects the stiffness properties of the learned function significantly. High learning rates induce functional approximations that are stiffer over larger distances (i.e. data points further apart respond similarly to gradient updates) and that the features learned generalize better to inputs from different classes (i.e. data points from different classes respond similarly to gradient updates). Lower learning rates, on the other hand, seem to learn more detailed, specific features that, even though leading to the same loss on the training set, do not generalize to other classes as well. This points towards high learning rates being not only advantageous due to the smaller number of steps needed to converge, but also due to the higher generalizability of the features they tend to learn, i.e. that high learning rates act as an effective regularizer.

This paper is structured as follows: we introduce the concept of stiffness and the relevant theory in Section 2. We describe our experimental setup in Section 3, and discuss their results in Section 4. We conclude with Section 5.

2. Theoretical background

2.1. Stiffness – definitions

Let a functional approximation (e.g. a neural network) f be parametrized by tunable parameters W . Let us assume a classification task and let a data point X have the ground truth label y . A loss $\mathcal{L}(f_W(X), y)$ gives us the amount of mismatch between the functions output at input X and the ground truth y . The gradient of the loss with respect to the

parameters

$$\vec{g} = \nabla_W \mathcal{L}(f_W(X), y) \quad (1)$$

is the direction in which, if we were to change the parameters W , the loss would change the most rapidly (at least for infinitesimal step sizes). Gradient descent uses this step to update the weights and gradually tune the functional approximation to better correspond to the desired outputs on the training dataset inputs.

Let there be two data points with their ground truth labels (X_1, y_1) and (X_2, y_2) . We construct a gradient with respect to example 1 as $\vec{g}_1 = \nabla_W \mathcal{L}(f_W(X_1), y_1)$ and ask, how do the losses on data points 1 and 2 change as a result of a small change of W in the direction \vec{g}_1 , i.e. what is

$$\Delta \mathcal{L}_1 = \mathcal{L}(f_{W+\varepsilon \vec{g}_1}(X_1), y_1) - \mathcal{L}(f_W(X_1), y_1), \quad (2)$$

which is equivalent to

$$\Delta \mathcal{L}_1 = \varepsilon \nabla_{\varepsilon} \mathcal{L}(f_{W+\varepsilon \vec{g}_1}(X_1), y_1) = \varepsilon \vec{g}_1 \cdot \vec{g}_1 \quad (3)$$

The change in loss on input 2 due to the gradient step from input 1 becomes equivalently

$$\Delta \mathcal{L}_2 = \varepsilon \nabla_{\varepsilon} \mathcal{L}(f_{W+\varepsilon \vec{g}_1}(X_2), y_2) = \varepsilon \vec{g}_1 \cdot \vec{g}_2. \quad (4)$$

We are interested in the correlation in loss changes $\Delta \mathcal{L}_1$ and $\Delta \mathcal{L}_2$. We know that $\Delta \mathcal{L}_1 < 0$ since we constructed the gradient update accordingly. We define positive stiffness to mean $\Delta \mathcal{L}_2 < 0$ as well, i.e. that losses at both inputs went down. There would be no stiffness if $\Delta \mathcal{L}_2 = 0$ and the two inputs would be anti-stiff, i.e. negative stiffness, if $\Delta \mathcal{L}_2 > 0$. The equations above show that this can equivalently be thought of as the overlap between the two gradients $\vec{g}_1 \cdot \vec{g}_2$ being positive for positive stiffness, and negative for negative stiffness. We illustrate this in Figure 1.

The above indicate that what we initially thought of as a change in loss due to the application of a small gradient update from one input to another is in fact equivalent to analyzing *gradient alignment* between different datapoints.

We define stiffness to be the expected sign of $\vec{g}_1 \cdot \vec{g}_2$ (or equivalently the expected sign of $\Delta \mathcal{L}_1 \Delta \mathcal{L}_2$) as

$$S((X_1, y_1), (X_2, y_2); f) = \mathbb{E} [\text{sign}(\vec{g}_1 \cdot \vec{g}_2)] , \quad (5)$$

where stiffness depends on the dataset from which X_1 and X_2 are drawn (e.g. examples of the same class, examples a certain distance apart etc.) as well as on the particular architecture and weights specifying the neural net/function approximator f . The sign of $\vec{g}_1 \cdot \vec{g}_2$ is positive, when \vec{g}_1 points in the same half-space as \vec{g}_2 . That means that positive stiffness corresponds to the weight updated optimal for input 1 having at least a partial alignment with the optimal weight update for input 2. We illustrate this in Figure 1.

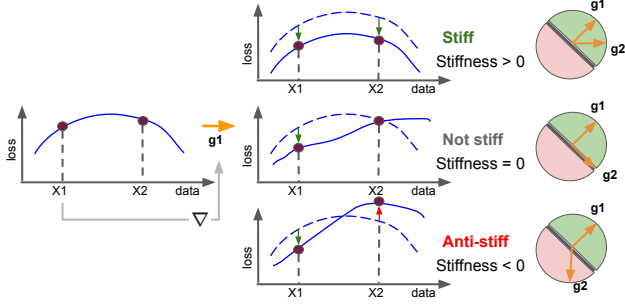


Figure 1. A diagram illustrating the concept of stiffness. It can be viewed as the change in loss in an input induced by application of a gradient update based on another input. This is equivalent to the gradient alignment between gradients taken at the two inputs.

In the empirical part of this paper, we study the average stiffness between inputs X_1 and X_2 as a function of their different properties. We define the relevant details in the following subsections.

2.2. Stiffness based on class membership

A natural question to ask is whether a gradient taken with respect to an input X_1 in class c_1 will also decrease the loss for example X_2 with true class c_2 . In particular, we define the *class stiffness matrix*

$$C(c_a, c_b) = \mathbb{E}_{X_1 \in c_a, X_2 \in c_b} [S((X_1, y_1), (X_2, y_2))] \quad (6)$$

The on-diagonal elements of this matrix correspond to the suitability of the current gradient update to the members of a class itself. In particular, they correspond to *within class* generalizability. The off-diagonal elements, on the other hand, express the amount of improvement transferred from one class to another. They therefore directly diagnose the amount of generality the currently improved features have. We work with the stiffness properties of the validation set, and therefore investigate generalization directly.

A consistent summary of generalization between classes is the off-diagonal sum of the class stiffness matrix

$$S_{\text{between classes}} = \frac{1}{N_c(N_c - 1)} \sum_{c_1} \sum_{c_2 \neq c_1} C(c_1, c_2) \quad (7)$$

In our experiments, we track this value as a function of learning rate once we reached a fixed loss. The quantity is related to how generally applicable the learned features are, i.e. how well they transfer from one class to another. For example, for CNNs learning good edge detectors in initial layers typically benefits all downstream tasks, regardless of the particular class in question.

2.3. Stiffness as a function of distance

We investigate how stiff two inputs are based on how far away from each other they are. We can think of neural networks as a form of kernel learning and here we are investigating the particular form of the learned kernel. This links our results to the work on spectral bias (towards slowly-varying, low frequency functions) in Rahaman et al. (2018). We are able to directly measure the characteristic size of the stiff regions in neural networks trained on real tasks, i.e. what the characteristic size on which data points very together is for our trained networks.

Let us have two inputs X_1 and X_2 , that are pre-processed to zero mean and unit length. Those are then fed into a multi-layer neural network, where each layer will produce a representation R of the input and pass it on to the next layer. Schematically, the network forms a set of representations as

$$(X = R^{(0)}) \rightarrow R^{(1)} \rightarrow R^{(2)} \dots \rightarrow R^{(L)} \rightarrow \mathcal{L} \quad (8)$$

We study how stiffness between two inputs X_1 and X_2 depends on their mutual distance. We investigate L_1 and L_2 distances, as well as the dot product distance between representations. We look at both the input (pixel) space distances and distances between representations formed by the network itself.

The distance metric that we use is the dot product distance

$$\text{dot}(R_1, R_2) = \frac{R_1 \cdot R_2}{|R_1| |R_2|}, \quad (9)$$

which has the advantage of being bounded between -1 and 1 and therefore makes it easier to compare distances between different layers.

We identify a sharp decline in the amount of stiffness between inputs further than a threshold distance from each other in all representations including the input space. We track this threshold distance as a function of training and learning rate to estimate the characteristic size of the stiff regions of a neural net.

3. Methods

3.1. Experimental setup

We ran a large number of experiments with fully-connected (FC) and convolutional neural networks (CNN) on 4 classification datasets: MNIST (LeCun & Cortes, 2010), FASHION MNIST Xiao et al. (2017), CIFAR-10, and CIFAR-100 Krizhevsky (2009). Using those experiments, we investigated the behavior of stiffness as a function of 1) training iteration, 2) the choice of learning rate, 3) class membership, and 4) distance between images (in the input space as well as representation spaces within the networks themselves).

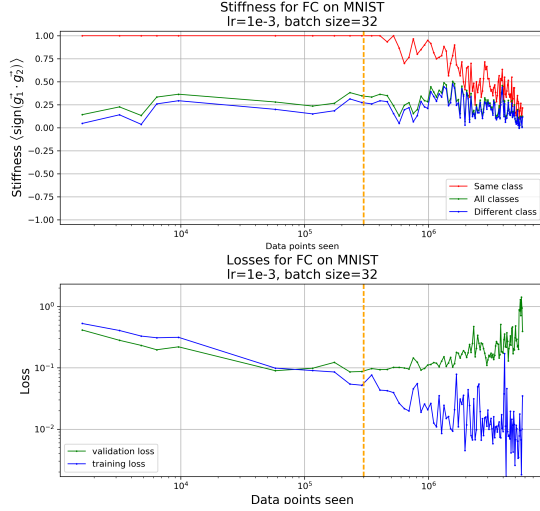


Figure 2. The evolution of training and validation losses, and stiffness. The graphs show the evolution of training and validation loss (lower panel) and class-dependent stiffness properties (upper panel) as a function of the number of images seen during training. The onset of over-fitting (the split between training and validation loss curves) is marked with a yellow line. The within-class stiffness, i.e. the transfer of improvement from one input image to another *within the same class* declines during overfitting and regresses to 0. The between-classes stiffness, starts plateauing and regresses back to 0. This demonstrates the direct connection between stiffness properties measured on the validation set and generalization, and shows that stiffness is a relevant property to study.

For experiments with fully-connected neural networks, we used a 6 layer ReLU network of the form $X \rightarrow 400 \rightarrow 200^5 \rightarrow y$. For experiments with convolutional neural networks, we used a 5 layer network with filter size 3 and the numbers of channels being 32, 64, 128 and 256 after the respective convolutional layer, each followed by 2×2 max pooling. The final layer was fully-connected. No batch normalization was used.

We pre-processed the network inputs to have zero mean and unit variance. We used Adam with different (constant) learning rates as our optimizer and a default batch size of 32.

3.2. Training and stiffness evaluation

We evaluated stiffness properties between data points in the *validation set* to study generalization. We used the training set to train our model. The procedure was as follows:

1. Train for a number of steps on the *training* set and update the network weights accordingly.
2. Fix the network weights.

3. Go through tuples of images in the validation set.
4. For each tuple calculate the loss gradients g_1 and g_2 , and check $g_1 \cdot g_2$.
5. Log distances between the images as well as other relevant features.

In our experiments, we used a fixed subset (typically of ≈ 150 images for experiments with 10 classes, and ≈ 1000 for experiments with 100 classes) of the validation set to evaluate the stiffness properties on. We convinced ourselves that such a subset is sufficiently large to provide measurements with small enough statistical uncertainties.

3.3. Learning rate dependence

We investigated how stiffness properties depend on the learning rate used in training. To be able to compare training runs with different learning rates fairly, we looked at them at the time they reached the same training loss. Our results are presented in Figure 8.

4. Results and discussion

4.1. Stiffness properties based on class membership

We explored the stiffness properties of validation set data points based on their class membership as a function of training iteration. Our results are summarized in Figures 3, 5, and 6 for MNIST, FASHION MNIST and CIFAR-10 with true labels respectively, and in Figure 4 for MNIST with randomly permuted training set labels.

Stiffness between two data points characterizes the amount of correlation between changes in loss on the two due to the application of a gradient update based on one of them. This, as we show in Section 2, can be thought of as the amount of alignment between gradients at the two input points.

We focused on stiffness between inputs in the validation set as we wanted to explore generalization. If a gradient step taken with respect to a validation set input would improve loss on another validation set input, the gradient step potentially represents a genuinely generalizable feature that is relevant to the underlying generator of the data.

Figures 3, 5, and 6 show the stiffness matrix at 4 stages of training: at initialization (before any gradient step was taken), early in the optimization, and at two late-time stages.

Initially, an improvement based on an input from a particular class benefits only members of the same class. Intuitively, this could be due to some crude features shared within a class (such as the typical overall intensity, or the average color) being learned. There is no consistent stiffness between different classes at initialization. As training progresses, within-class stiffness stays high. In addition,

stiffness between classes increases as well, given the model were is powerful enough for the dataset. Features, that are beneficial to almost all inputs are being learned at this stage. The pattern is visible in Figures 3, 5, and 6, where the off-diagonal elements of the stiffness matrix become consistently positive with training. With the onset of overfitting, as shown in Figure 2, the model becomes increasingly less stiff until even stiffness for inputs within the same class is lost. This is due to the model overfitting to the specific details of the training set. The features learned are too specific to the training data, and therefore do not generalize to the validation set, which leads to the loss of stiffness.

We ran experiments with randomly permuted training set labels to explore the evolution of stiffness there. In Figure 4, the stiffness of a fully-connected network trained on MNIST with permuted labels is shown. As there are no general features to be learned, the model converges to a stage with no positive between classes stiffness. The reason for the off-diagonal, i.e. between different classes, stiffness converges to -1 is the fact that the optimal response is to give all classes equal probability. Any gradient update based on a particular input will necessarily lead to a preference to one of the classes (the one that was randomly assigned to this data point), which in turn increases loss on other inputs on average.

In our experiments with CIFAR-100 we notice a block-like structure in the stiffness matrix shown in Figure 7. The coarse-grained pattern is suggestive of the networks knowledge of the super-classes (groups of 5 classes), on which the network, however, was not trained. This is due to the similarity between images within the super-class and points strengthens the connection of stiffness and generalization.

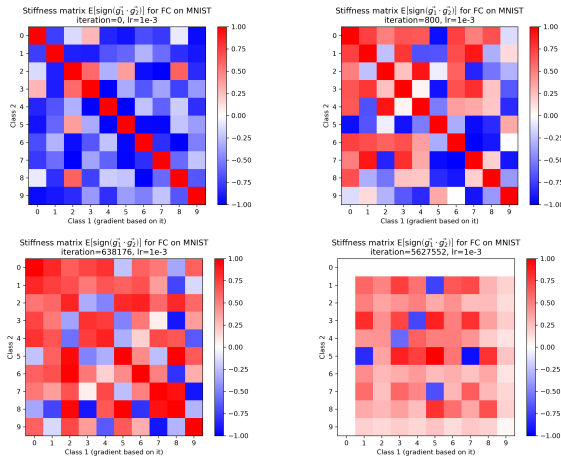


Figure 3. Class-membership dependence of stiffness for a fully-connected network on MNIST.

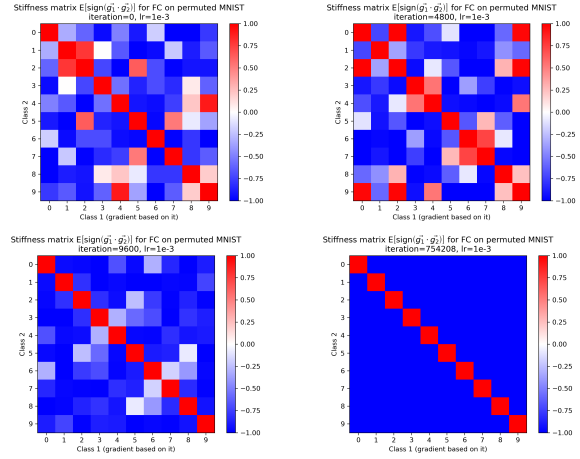


Figure 4. Class-membership dependence of stiffness for a fully-connected network on MNIST when trained with *randomly permuted labels*.

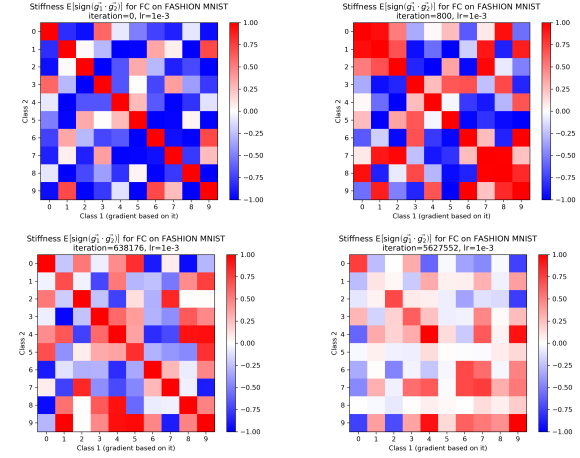


Figure 5. Class-membership dependence of stiffness for a fully-connected network on FASHION MNIST.

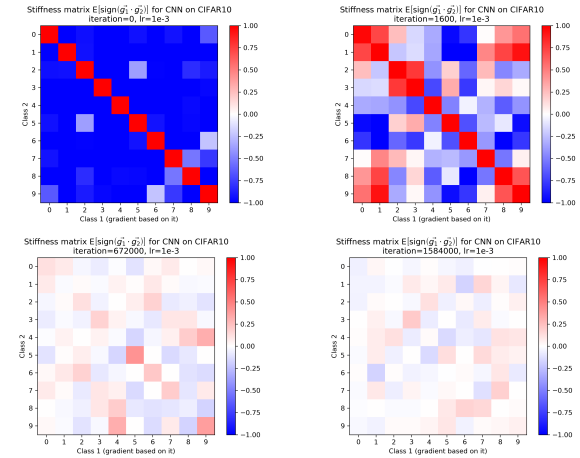


Figure 6. Class-membership dependence of stiffness for a convolutional neural network on CIFAR-10.

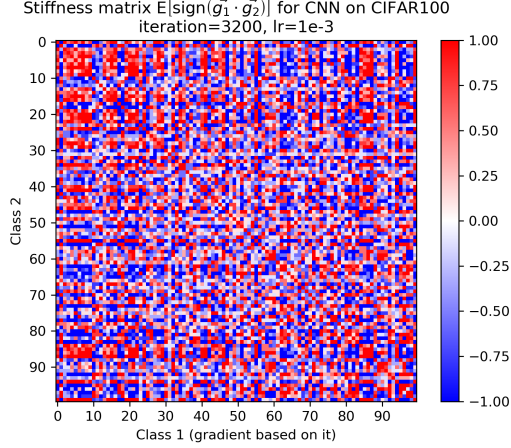


Figure 7. Class-membership dependence of stiffness for a convolutional neural network on CIFAR-100 in early stages of training. A prominent block-like structure is visible in the matrix, hinting at the network being aware of the super-class structure of CIFAR-100.

4.2. The effect of learning rate on stiffness

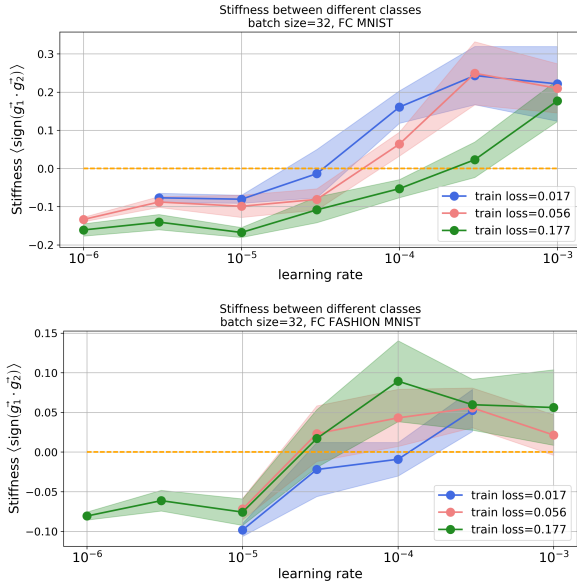


Figure 8. Stiffness between different classes reached when training with different learning rates on MNIST and FASHION MNIST. The two figures present the mean of the off-diagonal elements of the class dependent stiffness matrix for three different training losses. Higher learning rates lead to higher stiffness between inputs from different classes, which suggests that they learn features that are more generalizable between different classes. This suggests that higher learning rates act effectively as regularizers towards stiffer functions.

We investigated the effect of learning rate on stiffness of the functions learned. In particular, we focused on the amount

of between-classes stiffness that characterizes the generality of the features learned and the transfer of knowledge from one class to another. We used the mean of the off-diagonal terms of the class stiffness matrix as described in Section 2.

In order to be able to compare the learned functions fairly for different learning rates, we decided to train until a particular training loss was reached. We then investigated the stiffness properties of the learned function. Our results are presented in Figure 8. We observe that the higher the learning rate, the more stiff the learned function is between different classes, i.e. that higher learning rates bias the models found towards features that benefit several classes at once. We observed this behavior for both MNIST and FASHION MNIST and at all three stopping training losses we investigated.

Our hypothesis is that high learning rates force the model to learn very generalizable, robust features that are not easily broken by the stochasticity provided by the large step size. Those features tend to be useful for multiple classes at once. We speculate that this points towards the regularization role of high learning rate that goes beyond the benefit of having a smaller number of steps until convergence. The concept of stiffness therefore sheds some light on the regularization value of high learning rates.

4.3. Stiffness as a function of distance and the role of learning rate

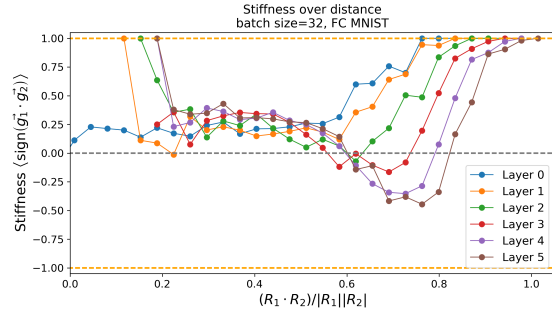


Figure 9. An example of the dependence of stiffness on distance. The plot shows how average stiffness between inputs changes with their distance (in input space as well as in induced layer representations) for a trained fully-connected network on MNIST.

We investigated stiffness between two inputs as a function of their distance in order to measure how large the patches of the learned function that move together under gradient updates are. This relates to the question of spectral bias of neural networks, however, the connection is not straightforward, as we will discuss later.

We studied distances in the input (pixel) space as well as distances between representations induced by each layer of our neural networks. We primarily focused on the dot-

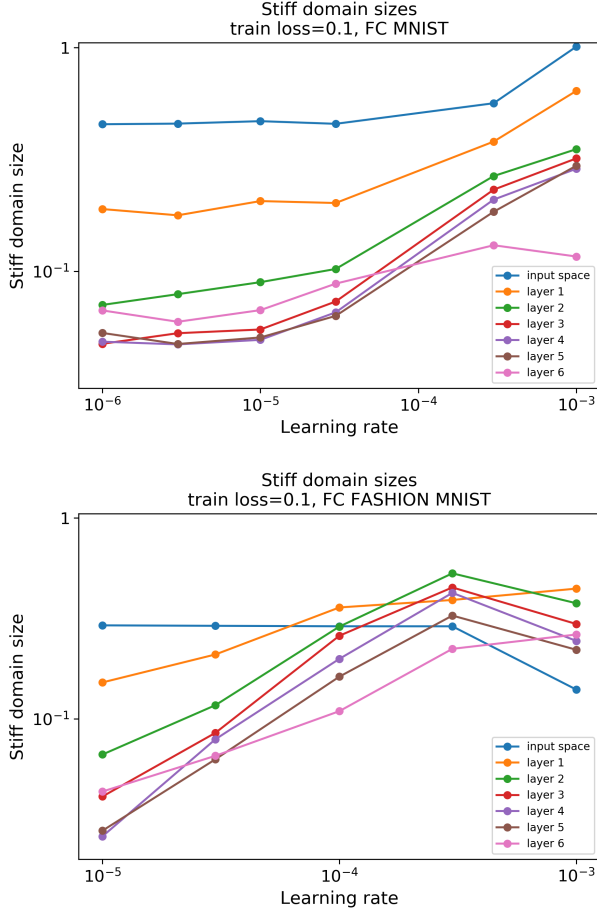


Figure 10. The size of stiff domains for a fully-connected neural network trained on MNIST and FASHION MNIST as a function of learning rate. The plots show the characteristic distances (both in the input pixel space as well as in the representation spaces induced by the network itself) under which data points tend to respond similarly to gradient updates. The stiff domain sizes grow with increasing learning rate.

product distance, which we defined to be the cosine of the angle between two input/representation vectors. This distance is bounded between -1 and 1 and is therefore easier to compare between layers.

To be able to compare training at different learning rates, we trained until a particular training loss was reached and then analyzed the stiffness properties of the learned function. An example of the distance dependence of stiffness is presented in Figure 9. Note that the dot product distance of 1 corresponds to points being at the same place. The general pattern visible in Figure 9 is that there exists a critical distance within which input data points tend to move together under gradient updates, i.e. have positive stiffness. This holds true for all layers in the network, with the tendency of

deeper layers to have smaller stiff domain sizes.

We extracted the first zero-stiffness crossing, such as in 9, and obtained the variation of stiff domain sizes with learning rate. We observed that the characteristic size of stiff regions in the learned function increases with higher learning rates. The stiff region size corresponds to the distance between inputs (in the input pixel space as well as in the representation spaces of the neural network itself) under which a gradient update tends to improve all of them. It characterizes the spatial frequency of the learned function’s response to gradient updates. Our results are presented in Figure 10. Our observations seem connected to recent work on regularization using interpolation between data points in Verma et al. (2018).

4.4. Stiff domain size as the characteristic length scale?

A natural question arises as to whether the characteristic distance between two input points at which stiffness reaches zero defines the typical scale of spatial variation of the learned function. Unfortunately, that is *not* necessarily the case, though it can be for some families of functions. The stiff domain sizes visible in Figure 9 represent the typical length scale over which neural networks *react* similarly to gradient inputs, rather than the typical length scale of variation of the function value itself.

To illustrate the difference, imagine a function that varies rapidly over input data, however, whose losses over the same data move in the same direction on application of a gradient step based on any of the data points. This function would have a small characteristic length scale of value variation, yet large stiff domain size. We believe that these two length scales are likely to be connected, however, we have not explored this direction in this paper. We believe that for fully-connected ReLU networks the connection is likely.

5. Conclusion

We explored the concept of neural network stiffness and used it to diagnose and characterize generalization. We studied stiffness for models trained on real datasets, and measured its variation with training iteration, class membership, distance between data points, and the choice of learning rate. We focused on the stiffness of data points in the *validation set* in order to probe generalization and overfitting.

On real data, we explored models trained on MNIST, FASHION MNIST, CIFAR-10 and CIFAR-100 through the lens of stiffness. In essence, stiffness measures the alignment of gradients taken at different input data points, which we show is equivalent to asking whether a weight update based on one input will benefit the loss on another. We demonstrate the connection between stiffness and generalization and show

that with onset of overfitting to the training data, stiffness on the validation set decreases and eventually reaches 0, where even gradient updates taken with respect images of a particular class stop benefiting other members of the same class.

Having established the usefulness of stiffness as a diagnostic tool for generalization, we explored its dependence on class membership. We find that in general gradient updates with respect to a member of a class help to improve loss on data points in the same class, i.e. that members of the same class have high stiffness with respect to each other. This holds at initialization as well as throughout most of the training. The pattern breaks when the model starts overfitting to the training set, after which within-class stiffness eventually reaches 0. We observe this behavior with fully-connected and convolutional neural networks on MNIST, FASHION MNIST, CIFAR-10, and CIFAR-100. Stiffness between inputs from different classes relates to the generality of the features being learned and within-task transfer of improvement from class to class. We find that untrained models do not exhibit consistent stiffness between different classes, and that with training its amount increases. For a model with high-enough capacity for the task at hand, we observe positive stiffness between the majority of classes during training. With the onset of overfitting, the stiffness between different classes regresses to 0, as does within-class stiffness.

We experimented with training on data with randomly permuted labels, where no meaningful general patterns can be learned. There, the stiffness between examples disappears as the model trains. This is expected as for positive stiffness to appear, features that are beneficial for many inputs must develop, which is impossible with randomly permuted labels. This highlights the connection between stiffness and the generality of features learned. Since we measure stiffness on the validation set, explicitly probe generalization.

We observed that for a model trained on CIFAR-100, a block-like structure appears in the class-dependent stiffness matrix. We believe this is related to the same response of the network to gradient updates by images in the super-class in the dataset. This is another pointer towards the usefulness of stiffness in diagnosing generalization. Since our model had no access to the super-class (coarse grained) labels, the structure in the stiffness matrix likely came from general features being learned.

We investigated the effect of learning rate on stiffness and identified a tendency of high learning rates to induce more stiffness into the learned function. We find that for models trained with different learning rates and stopped at an identical training loss, the amount of stiffness between different classes is higher for higher learning rates. This points towards the role of high learning rate in learning more general features that are beneficial for inputs from many classes.

Lower learning rates, on the other hand, seem to learn more detail, class-specific features that do not transfer well to other classes.

We also investigated the characteristic size of stiff regions in our trained networks. By studying stiffness between two validation set inputs and measuring their distance in the input space as well as in the representation spaces induced by the neural network, we were able to show that size of stiff regions – regions of the data space that move together when a gradient update is applied – increases with increasing learning rate. We therefore find that higher learning rates tend to learn functions whose response to gradient updates varies over larger characteristic length scales. This is in line with our previous observation that the average stiffness is higher for higher learning rates. Both of these observations point towards the regularization effect of learning rate beyond the benefit of a smaller number of steps until convergence.

In future work, we are investigating four lines of inquiry which are suggested by this work.

1. In this paper, all the experiments were conducted with a fixed architecture. One obvious extension to the concept of stiffness would be to ascertain the role stiffness might play in architecture search. For instance, we expect locality (as in CNN) to reflect in higher stiffness properties. It is quite possible that stiffness could be a guiding parameter for meta-learning and explorations in the space of architectures.
2. One idea we are pursuing is the use of stiffness to measure the efficacy of a particular ordering of data in the training set. It has been suggested that different permutations of standard NLP datasets behave differently in terms of performance (Schluter & Varab, 2018). We think this could be reflected in the stiffness of the data, which is something we are exploring.
3. As we noted in the results section, the super-class structure was related to the stiffness value for the CIFAR-100 data. To what extent is such hierarchical or relational structure visible from the stiffness value change over time?
4. We would like to investigate the connection between the characteristic size of variation of the learned function value and how it relates to the typical size of stiff domains we observe in our experiments.

In summary, we defined the concept of stiffness, showed its utility in providing a perspective to better understand generalization characteristics in a neural network and observed its variation with learning rate.

References

- Arora, S., Cohen, N., and Hazan, E. On the optimization of deep networks: Implicit acceleration by over-parameterization. *CoRR*, abs/1802.06509, 2018. URL <http://arxiv.org/abs/1802.06509>.
- Arpit, D., Jastrzebski, S. K., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A. C., Bengio, Y., and Lacoste-Julien, S. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 233–242, 2017. URL <http://proceedings.mlr.press/v70/arpit17a.html>.
- Cybenko, G. Approximation by superpositions of a sigmoidal function. *MCSS*, 2:303–314, 1989.
- Du, S. S., Lee, J. D., Li, H., Wang, L., and Zhai, X. Gradient Descent Finds Global Minima of Deep Neural Networks. *arXiv:1811.03804 [cs, math, stat]*, November 2018a. URL <http://arxiv.org/abs/1811.03804>. arXiv: 1811.03804.
- Du, S. S., Zhai, X., Póczos, B., and Singh, A. Gradient Descent Provably Optimizes Over-parameterized Neural Networks. *arXiv:1810.02054 [cs, math, stat]*, October 2018b. URL <http://arxiv.org/abs/1810.02054>. arXiv: 1810.02054.
- Fort, S. and Scherlis, A. The goldilocks zone: Towards better understanding of neural network loss landscapes. *CoRR*, abs/1807.02581, 2018.
- Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, July 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8. URL [http://dx.doi.org/10.1016/0893-6080\(89\)90020-8](http://dx.doi.org/10.1016/0893-6080(89)90020-8).
- Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.
- LeCun, Y. and Cortes, C. MNIST handwritten digit database. 2010. URL <http://yann.lecun.com/exdb/mnist/>.
- Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6:861–867, 1993.
- Li, C., Farkhoor, H., Liu, R., and Yosinski, J. Measuring the intrinsic dimension of objective landscapes. *CoRR*, abs/1804.08838, 2018. URL <http://arxiv.org/abs/1804.08838>.
- Montúfar, G., Pascanu, R., Cho, K., and Bengio, Y. On the number of linear regions of deep neural networks. In *NIPS*, 2014.
- Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. Sensitivity and generalization in neural networks: an empirical study. *CoRR*, abs/1802.08760, 2018.
- Pennington, J. and Worah, P. Nonlinear random matrix theory for deep learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 2637–2646. Curran Associates, Inc., 2017.
- Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., and Ganguli, S. Exponential expressivity in deep neural networks through transient chaos. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3360–3368, 2016.
- Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Sohl-Dickstein, J. On the expressive power of deep neural networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2847–2854, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL <http://proceedings.mlr.press/v70/raghu17a.html>.
- Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F. A., Bengio, Y., and Courville, A. On the Spectral Bias of Neural Networks. *arXiv e-prints*, art. arXiv:1806.08734, June 2018.
- Schluter, N. and Varab, D. When data permutations are pathological: the case of neural natural language inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4935–4939. Association for Computational Linguistics, 2018. URL <http://aclweb.org/anthology/D18-1534>.
- Schoenholz, S. S., Gilmer, J., Ganguli, S., and Sohl-Dickstein, J. Deep information propagation. *CoRR*, abs/1611.01232, 2016. URL <http://arxiv.org/abs/1611.01232>.
- Verma, V., Lamb, A., Beckham, C., Courville, A., Mitliagkis, I., and Bengio, Y. Manifold mixup: Encouraging meaningful on-manifold interpolation as a regularizer. *arXiv e-prints*, 1806.05236, June 2018. URL <https://arxiv.org/abs/1806.05236>.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016.