Extensions of the Methods for Understanding Neural Network Generalization

Thomas Walker

Contents

Affiliation Imperial College London
Correspondence thomas.walker21@imperial.ac.uk
Date July 2023

Part I
Introduction

In this report, I intend to extend some of the work investigated in 'A Survey of the Methods for Deriving Bounds on the Generalization Error of Deep Neural Networks'.

1 Components of Neural Network Generalization

There are three main components of neural networks that are used to understand their generalization capacities. These components are the training data, the training algorithm and the architecture. Some approaches deal exclusively with one of these components while other approaches aim to capture multiple components. For example, information-theoretic approaches focus on the training data and how information is transferred to the representation of the network. On the other hand, Bayesian machine learning is a framework for investigating the learning algorithm. Forming data-inspired priors was an extension of this framework to incorporate training data to get tighter bounds on the generalization error. To get a comprehensive picture of neural network generalization we ought to incorporate each of these three components.

Part II
Extensions

2 Stiffness

Recall that stiffness from [1] investigates the quantity

$$\bar{g} = \nabla_W \mathcal{L}(f_W(X), y),$$

for a functional approximation f, parameterized by a trainable parameter W. Where $\mathcal{L}(f_W(X), y))$ is the loss function.

Definition 2.1 *For two data points (X_1, y_1) and (X_2, y_2) define the sign stiffness to be*

$$S_{\text{sign}}\left((X_1, y_1), (X_2, y_2); f\right) = \mathbb{E}\left(\text{sign}\left(\bar{g}_1 \cdot \bar{g}_2\right)\right).$$

Definition 2.2 *For two data points (X_1, y_1) and (X_2, y_2) define the cosine stiffness to be*

$$S_{\cos}\left((X_1, y_1), (X_2, y_2); f\right) = \mathbb{E}\left(\cos\left(\bar{g}_1 \cdot \bar{g}_2\right)\right),$$

where

$$\cos\left(\bar{g}_1 \cdot \bar{g}_2\right) = \frac{\bar{g}_1 \cdot \bar{g}_2}{|\bar{g}_1||\bar{g}_2|}.$$

Stiffness captures information from the training data as the gradient of the loss function is evaluated at data points. It also captures information about the learning algorithm as the gradient of the loss function with respect to the parameters determines the update step of stochastic gradient descent. In the subsequent section, I look to investigate how stiffness can be related to the architecture of the neural network.

2.1 Distance

For a data point, we can investigate the stiffness in relation to the examples within a neighbourhood defined by some metric. For a clustering task, we can consider Euclidean distance to identify the boundaries of a cluster. Refer to Figure 1 for a visualization of this for a two-dimensional example.

2.2 Layers

Instead of representing the parameters as a singular vector, we can decompose them to reflect the architecture of the network. In the following we conduct similar analyses but with the gradient taken with respect to a subset of the parameters to identify how stiffness evolves through the layers of a network.

In the following, we consider the stiffness of data points in relation to a balanced sample of data points from the categories of the dataset. In Figure 2 we compare how the stiffness varies through the layers of a trained and untrained network. Then, in Figure 3 we look at how stiffness varies through the layers for networks of different architectures.

2.3 Activation Functions

In the original experiment, ReLU activation functions connected the layers of the networks. We can instead use tanh and LeakyReLU activation and observe how this affects the stiffness through the layers. My intention here is to understand whether the conclusions of [2] are present when neural networks are investigated from this perspective. Figure 4 shows the results for tanh activation, whereas, Figure 5 shows the results for networks with LeakyReLU activation.

2.4 Depth

By keeping the width of our networks constant we can investigate how stiffness varies through the layers as the depth of the network is changed. The result of these investigations can be seen in Figure 6.

2.5 Conclusions

We now compare our results with the conclusions made by [2]. It is clear that both ReLU and LeakyReLU activations extenuate the features of the dataset faster and with greater precision than tanh activations. Which supports the conclusions of [2]. ReLU was seen to cause the largest topological changes, and in our experimentation, we also see that ReLU identifies the features of the data set more clearly than the other activation functions. The effect of the bottleneck identified in [2] was similarly seen in our results. In [2] by changing the depth of a neural network of constant width it was shown that the burden of inducing topological changes was shifted toward the later layers. This is equally seen in my work, as the features start to become prominent in the latter layers of networks of increased depth.

References

[1] Stanislav Fort, Pawel Krzysztof Nowak, and Srini Narayanan. “Stiffness: A New Perspective on Generalization in Neural Networks”. In: *CoRR* (2019).

[2] Gregory Naitzat, Andrey Zhitnikov, and Lek-Heng Lim. “Topology of deep neural networks”. In: *CoRR* (2020).

Part III
Appendix

3 Figures

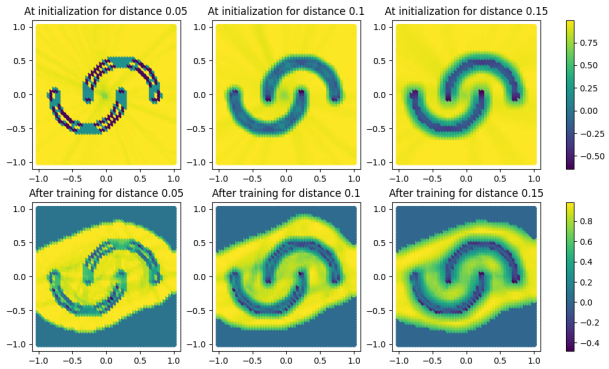

Figure 1: The stiffness of points within a neighbourhood can be used to identify the learned boundary of a classification problem.

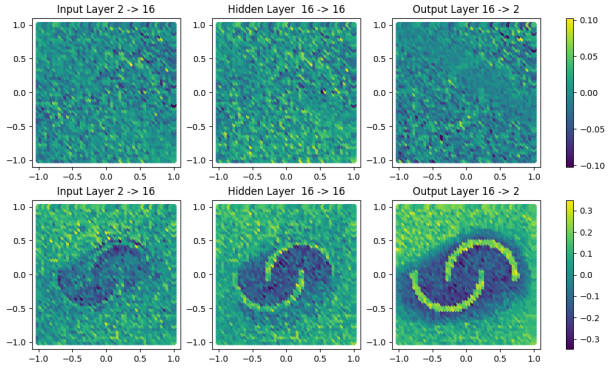

Figure 2: Average stiffness between an example and a random sample from the training data containing samples from both categories.

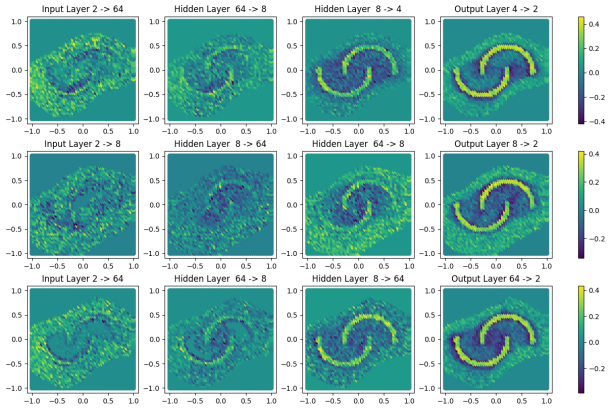

Figure 3: This shows how the stiffness of examples changes through the layers of networks with different numbers of hidden units.

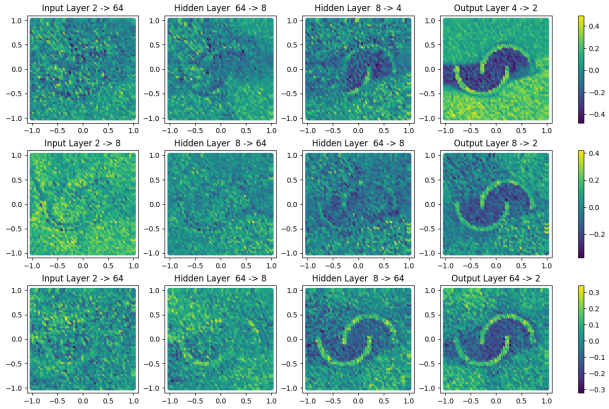

Figure 4: Stiffness through network layers connected by tanh activations

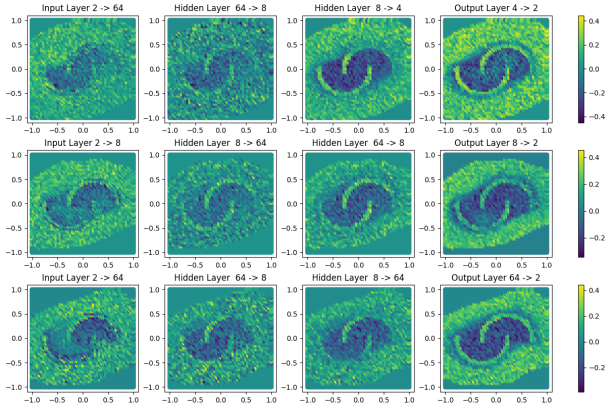

Figure 5: Stiffness through network layers connected by LeakyReLU activations

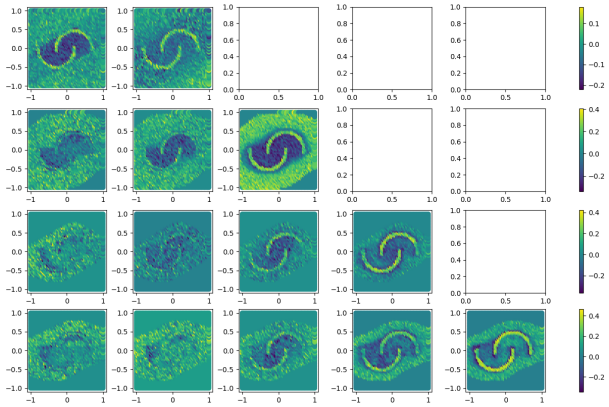

Figure 6: Stiffness between layers of a ReLU network with constant width of 16 hidden units.