Why Are Deep Networks Fragile: Deformation of Intertwined Data

Amirata Ghorbani [1] **James Zou** [2]

Abstract

In many classification tasks, the training samples are highly intertwined in the original input space. In image datasets, for example, a large fraction of training samples near each training point belong to different classes. The neural network seeks to learn a new representation of data—the last layer of the network—in which the training samples are linearly separable. Going from intertwined training data to such separable representation necessarily introduces severe deformation to the representation space, whereby points close to each other in the input space are mapped to be far apart in the new representation. In this work, we develop metrics to rigorously quantify how intertwined the input data is and how much space deformation the neural network produces during its training. Such deformation is a fundamental reason why these neural networks are fragile to adversarial perturbations. Our experiments quantify how fitting an intertwined dataset requires the model to deform the original space of the datasets in a way that small perturbations can result in big changes in the model's output.

1. Introduction

Suppose we have a machine learning classifier f and and let x be a correctly classified sample($f(x) = y_{\text{true}}$). An adversarial example x' is a point which is proximate to x in the original input space but is misclassified by f to be of another class. Recent works have demonstrated that adversarial perturbations is pervasive in deep learning (Szegedy et al., 2013; Biggio et al., 2013; Goodfellow et al., 2014; Tramr et al., 2017). This raises the troubling perspective that neural networks could be fundamentally fragile to certain small perturbations.

[1]Stanford Univesity, Stanford, California, USA [2]Stanford Univesity, Stanford, California, USA. Correspondence to: Amirata Ghorbani <amiratag@stanford.edu>, James Zou <jamesyzou@gmail.com>.

Presented at Reliable Machine Learning in the Wild Workshop, International Conference on Machine Learning, Sydney, Australia, 2017

In this work, we show that for a natural dataset, in order to achieve high classification accuracy, the neural network f maps the original input space of data to a new representation in a way that relative distances get highly deformed. Moreover how much deformation is created is related to how intertwined the original training samples are. We develop efficient methods to quantify intertwinedness and deformation. Our experiments suggest that large deformation is a fundamental reason why the classifier is vulnerable to adversarial perturbations. In Section 2, we discuss how training samples of different classes in a dataset are inseparably intertwined. In Section 3, we empirically quantify the space deformation due to convolutional neural networks and and discuss the connections between deformation and adversarial perturbations.

2. Data is intertwined in the input space

What is intertwinedness A learned machine learning classifier f consists of two stages: feature extraction and classification. We can view f as a composition, $f = c \circ g$ where g transforms samples from the original input space X into the feature space S where a simple classifier c (e.g. linear classifier, nearest neighbour, etc) is able to classify samples of different classes.

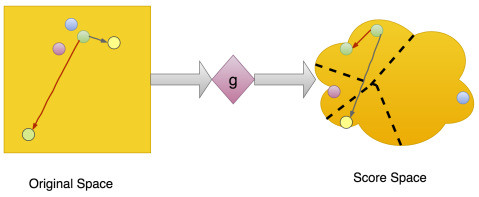


Figure 1. Intertwinedness of data results in space deformation.

Taking the recent state-of-the-art image classifiers as an example: X here is the pixel space, g is a deep neural network, S is the last hidden layer of the network, and c is a simple linear classifier (Krizhevsky et al., 2012; He et al., 2016; Simonyan & Zisserman, 2014). The fact that function c is a linear classifier means that g has mapped data samples in a manner that different classes are linearly separable in S. In the original space X, however, training data points from different classes could be closer to each other

than data points that share the same class by l_2 metric (Figure 1). We refer to this phenomenon as intertwinedness of the input data.

Quantifying intertwinedness There are several approaches to measure intertwinedness of a dataset in the original and feature space. In our first measure, $\text{ITW}_n(x)$, we take a test sample x and find the ratio of its n-nearest-neighbors in the training set that are of the same class as x. Dataset is more intertwined if the ratio is small. In Figure 2(a), the average $\text{ITW}_n(x)$ of 1000 test images in the CIFAR10 dataset (Krizhevsky, 2009) is depicted for n=$1, \ldots, 1000$ in the original and feature space of a 3 layer CNN[1].

We also use a complementary method to quantify intertwinedness. For a test sample x, $\text{ITW}_r(x)$ is the percentage of training samples that belong to the same class as x among all training samples which distance to x is smaller than r if there are any. In Figure 2(b) the average ITW_r for 2000 CIFAR10 test images in the original space X and feature space S of the three layer CNN are displayed for different values of r. The results of both measures demonstrate that samples of different classes are much less intertwined in the feature space compared to the original space.

How intertwinedness is connected to deformation of space As described in above, in a trained classifier, the feature extracting function g transforms a highly intertwined space of data points to a less intertwined space to enable linear separation. Therefore, g maps samples from different classes that are proximate in the original space to be far away in the feature space. This suggests that g must deform the input space in a way that small variations could result in big changes in the feature space. Similarly points far apart in the input space could be mapped close together in the feature space.

3. Deformation in deep networks

Deformation Metrics Intuitively, deformation corresponds to stretching and compression of space in different directions. We develop two complementary simple metrics to quantify the deformation caused by a neural network.

In the first approach, we examine the perturbation of the space surrounding a test sample through the presentation mapping g by generating a set of synthetic data points located on radius r small sphere around an actual sample x. We map x and each of these synthetic points through the neural network and measure the l_2 distances from x in the feature space. The *sphere deformation* could be interpreted as a condition number that quantifies the maximum stretching of the sphere divided by the minimum stretching. The specifics are described in Algorithm 1 and Figure 3(a) depicts the histogram of k for 1000 runs of the algorithm for the 3 layer CNN where $r = 1$, $n = 1000$. It shows that many points have deformation condition number greater than 10 suggesting a substantial heterogeneity in how different directions of the original sphere is stretched by the network. (More results are displayed in Appendix B.)

Algorithm 1 Sphere Deformation

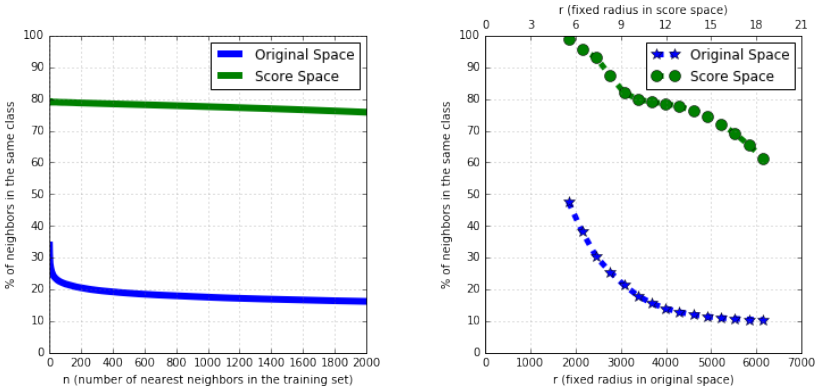
Input: sample test data $x \in R^D$, radius r
sample n random points uniformly with distance r from x: $p_1, \ldots, p_n$
map the test sample through g: $g(x)$
for $i = 1$ **to** n **do**
  map p_i through g: $g(p_i)$
  calculate the distance in feature space:
  $d_i = ||g(x) - g(p_i)||_2$
end for
calculate the maximum ratio of the distances:
$k = \frac{\max_{i=1,\ldots,n} d_i}{\min_{i=1,\ldots,n} d_i}$

The second approach randomly generate N line segments of various length in the input space and tracks how the rankings of the N lengths changes after mapping to the last layer of the neural network. We quantify the mapping changes using Spearman's rank order correlation. This approach is described in Algorithm 2. Figure 3(b) depicts the histogram of Spearman's rank order correlation values for running the algorithm 10000 times for the 3 layer CNN where $n = 100$. This experiments show that there is low correlation between the input lengths orders and the lengths orders in the new representation. This also suggests that there has been a substantial amount of space deformation.

How deformation is related to the complexity of the network Training deeper networks results in higher accuracy in general. For that reason, for the same original space X and same intertwindness, the network gets better at separating samples of each class from samples of other classes in the feature space S. This suggests that the deformation increases as the networks become deeper. Indeed, Figure 4 shows that the the sphere deformation increases and the rank-order correlation decreases as the number of layers in the network increases.

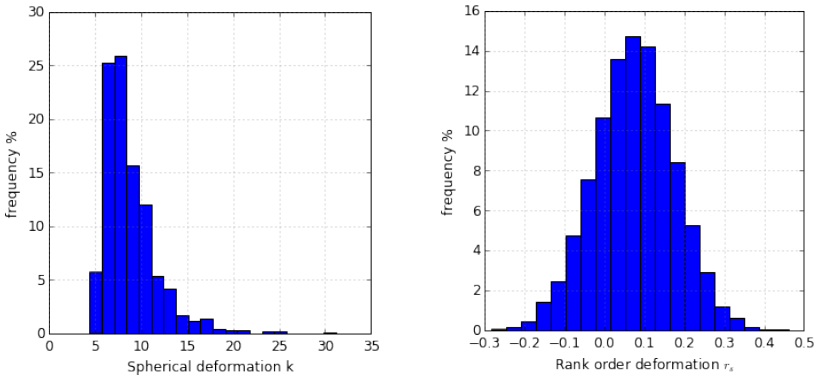
Deformation evolution during training After training, the network's performance at separating samples of different classes in the original space improves. Accordingly, we expect the deformation to increase with training. In Figure 4(c) shows the increase in space deformation of three different networks structures during training epochs using

[1]The details of all convolutional neural networks utilized in this work are described in Appendix A



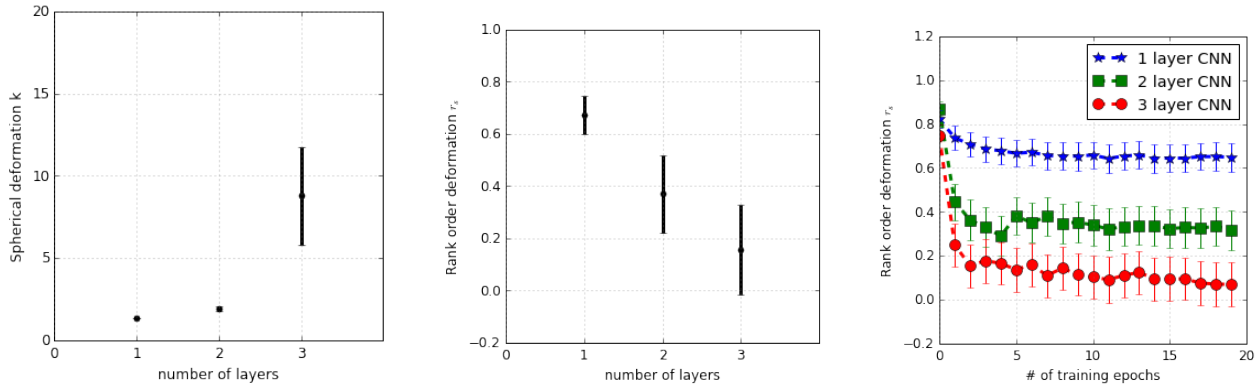
(a) (b)

Figure 2. Average percentage of (a) n nearest training samples (b) neighbors in training set within radius r, in CIFAR10 dataset that share the same class with a test data point of CIFAR10 dataset in the input space versus the feature space of the 3 layer CNN (for 1000 test data points).



(a) (b)

Figure 3. Using CIFAR10 dataset for the 3 layer CNN, (a) depicts histogram of sphere deformation using Algorithm 1 and (b) displays the histogram of rank order deformation using Algorithm 2



(a) (b) (c)

Figure 4. For the CIFAR10 dataset: (a) Average sphere deformation for 1000 test data points via three networks with different number of layers where $n = 1000$ and $r = 1$. (error bars stand for standard deviation) (b) Average rank order deformation for 1000 test data points via three networks with different number of layers where $n = 100$.(error bars stand for standard deviation) (c) Evolution of deformation using rank order deformation algorithm 1000 times with $n = 100$ after each training epoch.(error bars stand for standard deviation)

Algorithm 2 Rank Order Deformation

Input: n
Sample n pairs of test samples $(x_{1a}, x_{1b}), \ldots, (x_{na}, x_{nb})$
for $i = 1$ **to** n **do**
  $d_i = ||x_{ia} - x_{ib}||_2$
  map the pair through g and calculate the distances in the feature space:
  $D_i = ||g(x_{ia}) - g(x_{ib})||_2$
end for
Form the order set of distances in
sample n random points uniformly with distance r from x: $p_1, \ldots, p_n$
map the test sample through g: $g(x)$
for $i = 1$ **to** n **do**
  map p_i through g: $g(p_i)$
  calculate the distance in feature space:
  $d_i = ||g(x) - g(p_i)||_2$
end for
r_S = Spearman's rank order correlation between d_is and D_is

Algorithm 2. Additionally, for all structures the deformation is small for the randomly initialized network before training, suggesting that strong deformation is a result of confronting the intertwinedness of the data points rather than being an inherent feature of neural networks (Results for using Algorithm 1 are discussed in Appendix D.)

Deformation increases as training data becomes more intertwined. To quantify the relationship between data intertwinedness and network deformation, we create synthetic datasets where we can directly control intertwinedness. The synthetic data is basically a mixture of Gaussians where the different Gaussians are given different labels(more details in Algorithm 3 described in Appendix E). We control intertwindness by increasing the ℓ_2 distance between expected values. We train the same 3 layer CNN structure for each dataset. The results of 1000 runs of Algorithm 2 with $n = 100$ for each trained classifier are depicted in Figure 5 and confirms the relationship between intertwinedness and space deformation.

4. Network deformation and fragility

As described in Section 2, in the original space, the data is highly intertwined. Training and accurate classifier would cause the proximate data points to be distant and separable in feature space and the opposite happens for samples of each class that are not necessarily proximate. Consider a new test data point, as it is in the same intertwined space and as the classifier's mapping has high deformation, there would be directions moving toward which will cause big jumps in the feature space which result would be misclassification. Figure 6 shows the decrease in robustness as dataset becomes more intertwined(Details in Appendix G).

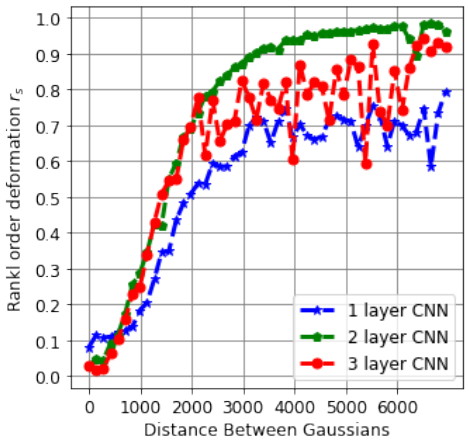


Figure 5. Rank order deformation versus the intertwindness of the dataset. (Error bars stand for standard deviation)

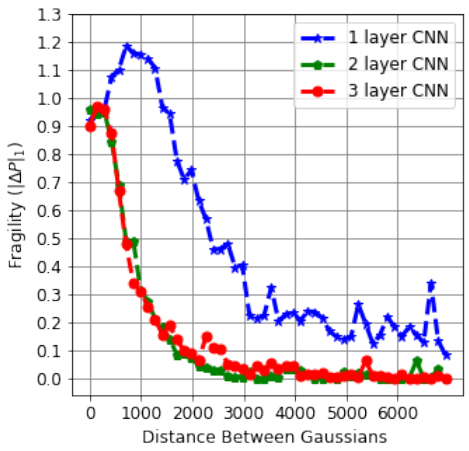


Figure 6. Using fast gradient sign method(FGSM) (Goodfellow et al., 2014) with parameter $\epsilon = 2$, we perturb 1000 randomly chosen unseen data points and measure the average ℓ_1 norm change in the output probability vector over class labels of the 3 layer CNN

Discussion We developed intuitive metrics to quantify intertwinedness and deformation of representation learning. Our experiments demonstrate how deformation increases during network training and also increases as the network becomes deeper. Our initial results suggest how deformation could cause adversarial examples, though additional experiments are also needed to flush out this relation.

References

Biggio, Battista, Corona, Igino, Maiorca, Davide, Nelson, Blaine, , Nedim Srndic, Laskov, Pavel, Giacint, Giorgio, and Roli, Fabio. Evasion attacks against machine learning at test time. *ECML-KDD*, pp. 387402, 2013.

Goodfellow, Ian J., Shlens, Jonathon, and Szegedy, Christian. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014. URL http://arxiv.org/abs/1412.6572.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Krizhevsky, Alex. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, Alex, Sutskever, Ilyaand, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012.

Langley, P. Crafting papers on machine learning. In Langley, Pat (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL http://arxiv.org/abs/1409.1556.

Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian J., and Fergus, Rob. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013. URL http://arxiv.org/abs/1312.6199.

Tramr, Florian, Papernot, Nicolas, Goodfellow, Ian, Boneh, Dan, and McDaniel, Patrick. The space of transferable adversarial examples. *arXiv:1704.03453*, 2017.

Appendix A

In what follows we talk about the convolutional neural networks utilized in this work. In all of the following networks batch normalization (Ioffe & Szegedy, 2015) is used after each convolutional layer, the activation function in all layers is rectified linear unit (ReLU), and ℓ_2 weight decay regularization is used. We user ADAM optimizer (Kingma & Ba, 2014) with default parameters except for the learning rates which are 0.0001,0.0005, 0.001 for the three following structures respectively.

1 layer CNN

The network structure is:

- one 5×5 stride 2 convolutional layer with 64 channels
- 8×8 stride 8 maxpooling layer
- feedforward network with one hidden layer with 1024 hidden units

The test accuracy of the network is 71.1 %.

2 layer CNN

The network structure is:

- one 5×5 stride 2 convolutional layer with 64 channels
- 2×2 stride 2 maxpooling layer
- one 5×5 stride 2 convolutional layer with 128 channels
- 2×2 stride 2 maxpooling layer
- feedforward network with one hidden layer with 1024 hidden units

The test accuracy of the network is 72.7 %.

3 layer CNN

The network structure is:

- one 5×5 stride 2 convolutional layer with 64 channels
- 2×2 stride 2 maxpooling layer
- one 5×5 stride 2 convolutional layer with 128 channels
- 2×2 stride 2 maxpooling layer
- one 5×5 stride 2 convolutional layer with 256 channels
- 2×2 stride 2 maxpooling layer
- feedforward network with one hidden layer with 1024 hidden units

The test accuracy of the network is 79.3 %.

Appendix B

In this appendix, for the three CNNs we had, we have displayed the result of sphere deformation algorithm for spheres with different radius sizes. As the figures suggest, although the network deforms distance in different directions in a different manner, the ratio of deformation between different directions is not affected with regards to the size of perturbations.

3 layer CNN

For the 3 layer CNN, Figure 7 displays the histogram of 1000 sphere deformation k values where r parameter is equal to 0.01,0.1,1,10,and 100 respectively. As it is observed, the distribution of k values is not highly dependent on r.

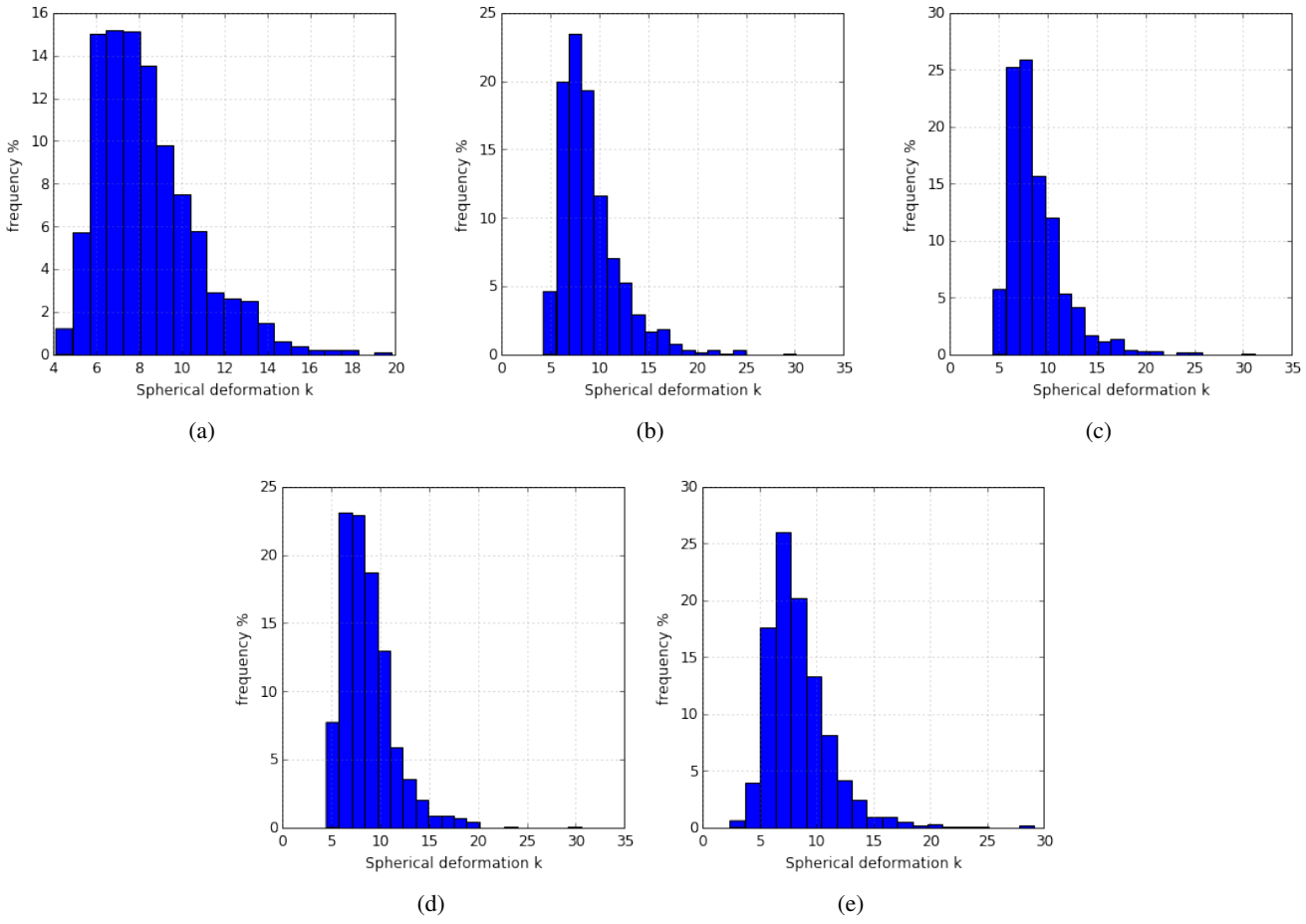


Figure 7. Histogram of sphere deformation k values for the 3 layer CNN where (a)r=0.01, (b)r=0.1 (c)r=1 (d)r=10 (e)r=100

2 layer CNN

For the 2 layer CNN, Figure 8 displays the histogram of 1000 sphere deformation k values where r parameter is equal to 0.01,0.1,1,10,and 100 respectively. As it is observed, the distribution of k values is not highly dependent on r.

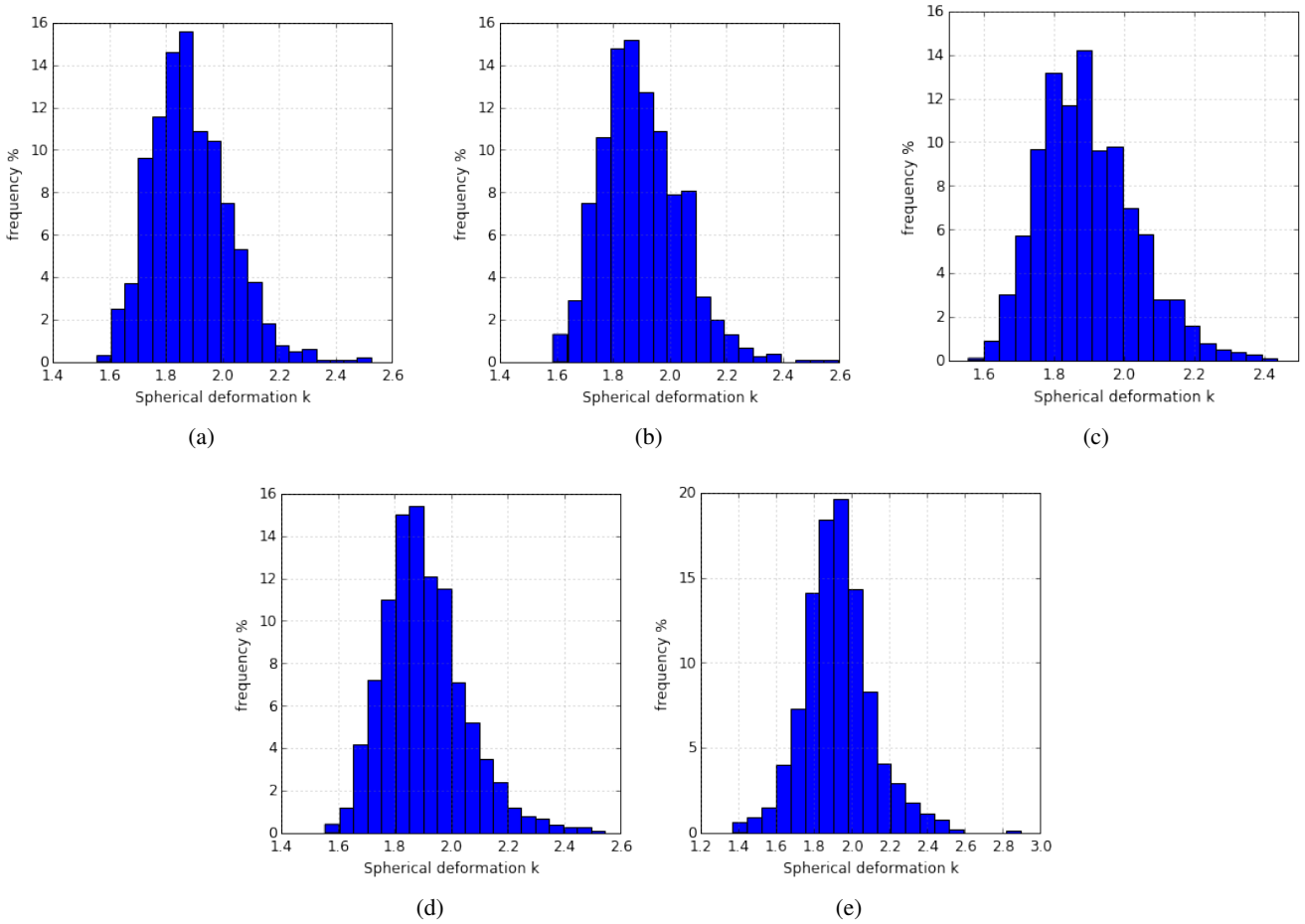


Figure 8. Histogram of sphere deformation k values for the 2 layer CNN where (a)r=0.01, (b)r=0.1 (c)r=1 (d)r=10 (e)r=100

1 layer CNN

For the 1 layer CNN, Figure 9 displays the histogram of 1000 sphere deformation k values where r parameter is equal to 0.01,0.1,1,10,and 100 respectively. As it is observed, the distribution of k values is not highly dependent on r.

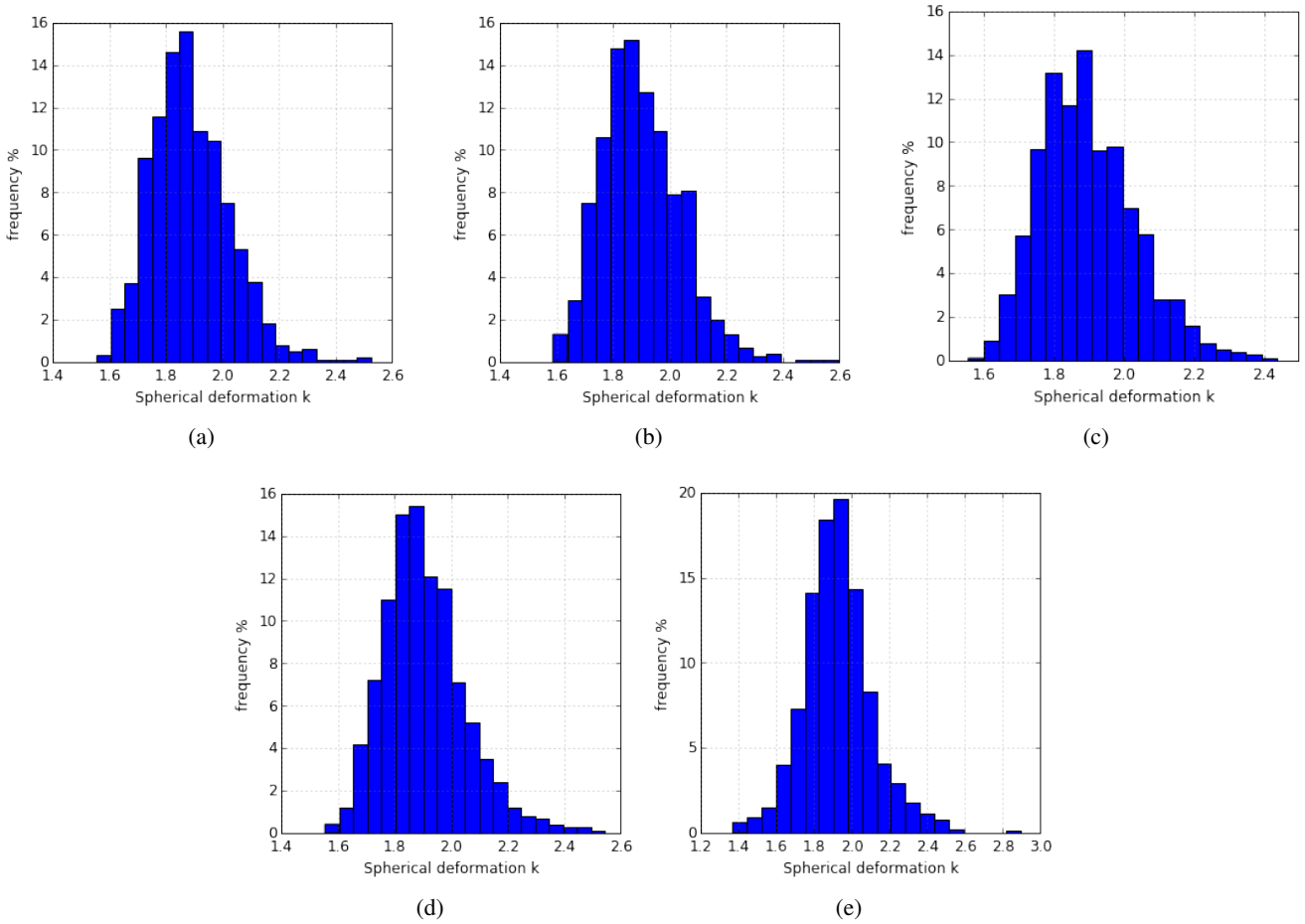
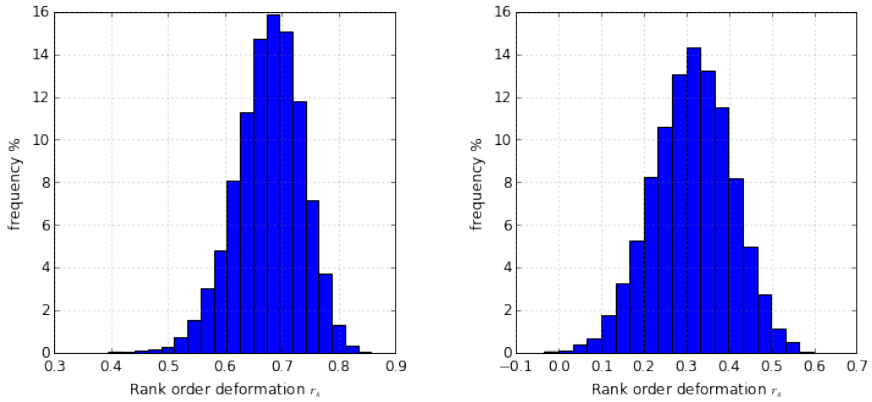


Figure 9. Histogram of sphere deformation k values for the 1 layer CNN where (a)r=0.01, (b)r=0.1 (c)r=1 (d)r=10 (e)r=100

Appendix C

The histogram of r_s values for 10000 runs of Algorithm 2 where $n = 100$ for 1 layer and 2 layer CNNs are shown in Figure 10(a) and Figure 10(b) respectively.

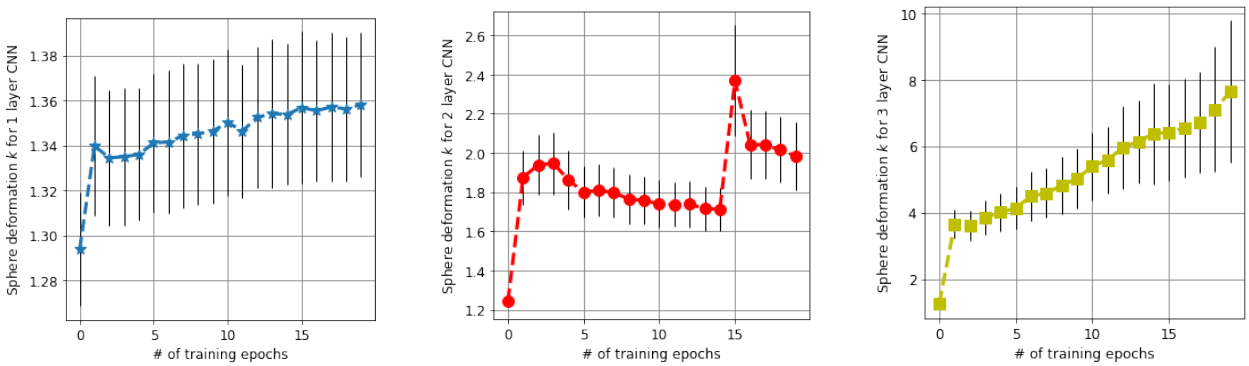


(a) (b)

Figure 10. Histogram of 10000 rank order deformation r_s values for n=100 and (a) 1 layer CNN and (b) 2 layer CNN

Appendix D

The evolution of sphere deformation metric for 1000 runs of Algorithm 1 for the three networks we have is depicted in Figure 11. The same as rank order deformation results, it could be seen that the deformation increases as networks' are trained and before training all networks have nearly the same sphere deformation.



(a) (b) (c)

Figure 11. Average sphere deformation for 1000 test data points for (a) 1 layer CNN (b) 2 layer CNN (c) 3 layer CNN. Error bars stand for standard deviation.

Appendix E

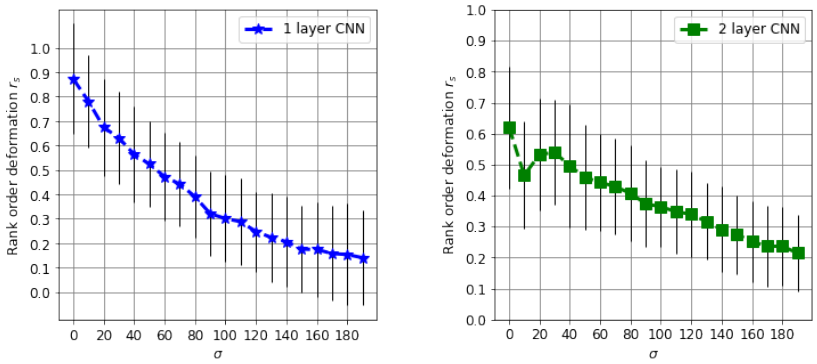
The following algorithm was used to generate synthetic datasets.

Algorithm 3 Synthetic Dataset Generation

Input: C, n, σ, Dateset Dimensions d, label set = $\{1, \ldots, L\}$:
for $i = 1$ **to** C **do**
  Randomly choose class label l from $\{1, \ldots, L\}$
  Sample $\mu \sim U([0, 256]^d)$
  Sample n data points from $\mathcal{N}(\mu, \sigma I)$
  Clip data points inside $[0, 256]^d$
  To all n data points assign class label l
end for

Appendix F

Using the same synthetic datasets described in Section 3, the evolution of ran order deformation metric for 1000 runs of Algorithm 2 for the one and two layer CNNs is depicted as we the datasets' intertwindness increases in Figure 12. As in the 3 layer CNN, increasing intertwindness increases the space deformation.



(a) (b)

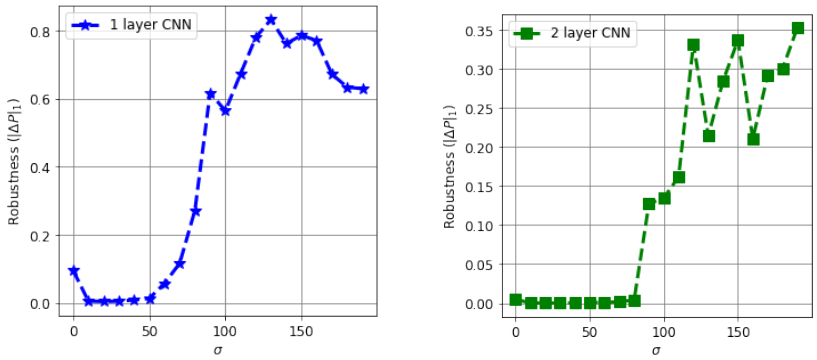
Figure 12. Rank order deformation for (a) 1 layer CNN (b) 2 layer CNN as dataset intertwindness is increased (error bars stand for standard deviation).

Appendix G

Using the synthetics datasets discussed in Appendix E, in order to measure the robustness of a trained network with loss function $\mathcal{L}(x, y)$, we track the change in ℓ_1 norm of last layer's output probability vector: For an unseen data point x with class label y, we perturb it using the Fast Gradient Sign Method(FGSM) discussed in (Goodfellow et al., 2014) which is as follows to get $x^\star$:

$$x^\star = x + \epsilon sign(\nabla_x \mathcal{L}(x, y))$$

Given that network's output probability vector for x and $x^\star$ to be $\mathbf{P}$ and $\mathbf{P}^\star$, we call the network is robust if $|\mathbf{P} - \mathbf{P}^\star|_1$ is small. Figure 13 displays the results for robustness versus intertwindness in synthetic datasets for one layer CNN and two layer CNN. The results confirm the assumption that increasing intertwindness results in robustness decrease.



(a) (b)

Figure 13. Average robustness versus the dataset intertwindness using 200 unseen data points for networks trained on synthetic datasets in (a) one layer CNN (b) two layer CNN