Investigating the Impact of Conditioning on Neural Network Adversarial Robustness

Ayesha Bajwa, Zareen Choudhury, Sanjana Srivastava
{abajwa, zareenc, sanjanas}@mit.edu

MIT 18.065/18.0651
May 16, 2018

***Abstract*—Is neural network susceptibility to adversarial examples related to ill-conditioning of weight matrices? We explore this relationship for a simple feedforward and a convolutional neural network under two adversarial attack methods, FGSM and FGMT. Adversarial attacks are carefully engineered transformations of input images that fool otherwise high-performing neural networks. A loss of performance occurs despite the fact that the transformed input images do not look particularly different from the originals. We inspect the condition numbers of the trained weight matrices of our models, and investigate whether condition numbers have an impact on adversarial susceptibility for different perturbations. We find that the models experience a significant drop in accuracy under the adversarial attacks for the same perturbation. Both models perform similarly under FGMT, but the convolutional network performs significantly worse than the simple model under FGSM, implying that it is overfit. We observe that the condition numbers of the weight matrices of the convolutional layers increase with training iteration, suggesting a correlation between ill-conditioned weight matrices and susceptibility to adversarial attacks.**

I. Introduction

The success of deep learning models is well-characterized in the literature but remains poorly understood from a mathematical standpoint. In particular, while deep models can show remarkable success in image classification on many standard datasets [4], [5], [6], [11], they are often highly susceptible to carefully crafted adversarial attacks [12]: it is usually possible to modify an image dataset such that there is no visible difference to humans, but a deep neural network classifier is easily, or at least partially, fooled [7], [8], [14].

In this paper, we explore the link between adversarial susceptibility of neural network models and the condition numbers of their trained weight matrices. Since the current science around deep learning and adversarial attacks remains largely unprincipled, there are competing philosophies on adversarial examples [13]. Some philosophies say that networks get fooled when they are overly complex, while others suggest that insufficiently complex networks are more easily fooled. We hope to explore these concepts and philosophies through our study of weight matrix conditioning.

II. Related Work

Our work in this paper is primarily inspired by the recent work of Singh et al. [10] in exploring ill-conditioned weight matrices with respect to adversarial examples. Like Singh et al., we use a variety of trained MNIST models and study weight matrix conditioning under multiple distinct attack methods with varying perturbations. We are also interested in studying the evolution of the weight matrices and their condition numbers as training progresses, since there are potential implications for methods like early stopping if the conditioning of the weight matrices varies greatly throughout training.

Researchers have studied ill-conditioning in neural network weight matrices for a number of years, traditionally in the context of training convergence. Previous research has drawn links between well-conditioned weight matrices and convergence to reasonable solutions or in reasonable time frames.

For example, the *Hessian matrix* – the matrix of second order derivatives of the error with respect to weights and biases – having a low condition number (close to 1.0) is associated with the negative gradient of the error pointing straight at the error surface minimum [9]. Therefore, standard optimization algorithms like *backpropagation* and *steepest* *descent* with *momentum*, which are based on taking steps in the direction of the negative gradient, have proved practical to use [9]. Additionally, due to the usual ill-conditioning of the error function derivative, neural network error landscapes often contain many local optima and saddle points, so finding global optima becomes difficult and convergence is either slow or not realized [15].

Instead of focusing on the derivatives with respect to the error function used in training, we study the weight matrices of neural networks directly (as do Singh et al [10]). The rationale for expecting well-conditioned weight matrices to provide higher robustness in the face of perturbation-based adversarial attacks is detailed in Section III-B.

III. Theory

A. Neural Networks

Neural networks account for a large fraction of state-of-the-art machine learning models. While they can be highly complex, their foundations involve fairly simple mathematics.

In a *feedforward* architecture, every layer has corresponding *nodes*, or artificial neurons, with floating point weights leading to and from them. The calculation between two layers can be represented as a linear combination of the earlier layer's outputs with the weights of the connecting edges. For example, the vector output of layer $\boldsymbol{x_i}$ are multiplied by the weight matrix $\boldsymbol{W_i}$, perhaps with some bias term $\boldsymbol{b_i}$ added, to produce the $\boldsymbol{z_i}$ vector input to the next layer's nodes:

$$\boldsymbol{z_i} = \boldsymbol{W_i^T x_i} + \boldsymbol{b_i}$$

At the later layer's node, an activation function f is applied to $\boldsymbol{z_i}$, producing the *activation* $\boldsymbol{a_i} = \boldsymbol{x_{i+1}}$, the output of that layer.

$$\boldsymbol{a_i} = f(\boldsymbol{z_i})$$

common activation functions include the rectified linear unit (ReLU), sigmoid, and softmax.

Feedforward architectures are often *fully-connected*, meaning that the *directed acyclic graph* created by the neural network includes every possible edge between pairs of nodes in two adjacent layers. These feedforward structures are also sometimes referred to as *multi layer perceptrons*, since they are extensions of the simple linear classifier and learning algorithm known as *perceptron*.

Convolutional neural networks are a popular alternative to the feedforward architecture. Their ability to preserve 2D input structure makes them especially common for *computer vision* and *image classification* tasks. Their main innovation is to, using the mathematical operation of *convolution*, combine adjacent features (such as pixel values) into aggregated features through the use of *filters* (also called *kernels* or *feature detectors*). These filters are usually 2-dimensional, in contrast to the feedforward architecture's constraint of a 1-dimensional input (such as a *flattened* image). Additionally, convolutional networks may operate with multiple *channels*, each with their own set of filters. In convolutional architectures, it is common to use *pooling* layers for downsampling.

For neural networks in general, once an architecture is specified, the problem is reduced from a model-fitting problem to a parameter-fitting problem. The *weights* and *biases* of a neural network are the parameters that are learning by the training algorithm – they are usually initialized randomly. We aim to investigate the conditioning of the weight matrices during and after training.

B. Matrix Conditioning

The condition number $\kappa(\boldsymbol{A})$ of matrix $\boldsymbol{A}$ is defined as the ratio of its largest singular value σ_1 to its smallest singular value σ_n:

$$\kappa(\boldsymbol{A}) = \frac{\sigma_1}{\sigma_n}$$

Here, the singular values are obtained from the singular value decomposition (SVD) of the matrix $\boldsymbol{A}$, which is written as $\boldsymbol{A} = \boldsymbol{U\Sigma V}^T$. The condition number of a singular matrix is infinite, the condition number of an almost-singular matrix is very large, and the condition number of an orthogonal matrix is 1. We refer to matrices with small condition numbers as being *well-conditioned*, while matrices with large condition numbers are referred to as being *ill-conditioned* because they are close to singular.

We calculate the condition number $\kappa(\boldsymbol{A})$ of a square matrix $\boldsymbol{A}$ using:

$$\kappa(\boldsymbol{A}) = ||\boldsymbol{A}||||\boldsymbol{A}^{-1}||$$

More generally, the condition number $\kappa(\boldsymbol{B})$ for a non-square matrix $\boldsymbol{B}$ is:

$$\kappa(\boldsymbol{B}) = ||\boldsymbol{B}||||\boldsymbol{B}^{+}||$$

The general condition number simply comes from using the pseudoinverse as opposed to the inverse. We use the standard definition of matrix norm in calculating condition number.

$$||\boldsymbol{B}|| = max_{\boldsymbol{x}=0}\frac{||\boldsymbol{B}\boldsymbol{x}||}{||\boldsymbol{x}||}$$

Note the matrix norm can vary. We investigate the 1-norm, the Euclidean norm (2-norm), the ∞-norm, and the Frobenius norm.

As Singh et al. explain [10], we can use the linear system defined by a neural network layer, before activation is applied, to indicate the effect of a perturbation δ. For the system:

$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + b$$

we have the perturbation:

$$\boldsymbol{z} + \delta\boldsymbol{z} = \boldsymbol{W}(\boldsymbol{x} + \delta\boldsymbol{x}) + b$$

which can also be described using condition number, after combining b and z into z_1:

$$\frac{||\delta\boldsymbol{z_1}||}{||\boldsymbol{z_1}||} \leq \kappa(\boldsymbol{W})\frac{||\delta\boldsymbol{x}||}{||\boldsymbol{x}||}$$

Since we are interested in adversarial examples involving the introduction of strategic perturbations to the input data x, it is reasonable to hypothesize that a well-conditioned weight matrix W could produce a model more robust to perturbation and therefore to these types of adversarial attacks.

C. Adversarial Attack Methods

We use two attack methods based on perturbation for generating adversarial examples: Fast Gradient Sign Method (FGSM) and Fast Gradient Method with Target (FGMT).

FGSM. The FSGM attack [3] bases its perturbation η in the cost function $J(\boldsymbol{\theta}, \boldsymbol{x}, y))$ used to train the neural network:

$$\boldsymbol{\eta} = \epsilon sign(\boldsymbol{\nabla_x} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$$

The gradient ∇_x is calculated in the usual way, through backpropagation. The adversarially generated examples are then characterized by:

$$\boldsymbol{x_{adv}} = \boldsymbol{x} + \boldsymbol{\eta}$$

We explore adversarial examples generated with the FGSM attack for ϵ ranging from 0.001 to 0.2.

FGMT. The FGMT attack is similar to the FSGM attack, but requires a specifying a particular target incorrect class for each adversarial class example. The implementation we use [2] for FGMT targets the lowest-probability class as the desired incorrect class in the adversarially generated dataset. Similarly to FGSM, we explore adversarial examples generated with the FGMT attack for ϵ ranging from 0.001 to 0.2.

IV. Methods

We explore weight matrix conditioning using the standard 10-class MNIST dataset of handwritten digits. Each example is a 28x28 pixel image that has the label associated with the digit it represents. Therefore, MNIST is a 10-class classification problem. We develop a pipeline in which we first train neural network models implemented in TensorFlow [1] to classify MNIST digits, then generate adversarial examples based on FGSM and FGMT attacks, and finally test each model's performance against the adversarial examples.

The neural network models we explore are adapted from the TensorFlow MNIST tutorials[1,2]. Their architectures are described below:

Simple. The simple model consists of a single layer with a softmax activation function. It uses cross entropy as its loss function. This model is expected to achieve approximately 92% test accuracy on non-adversarial MNIST examples.

Convolutional. The convolutional model is a convolutional neural network (CNN) consisting of the following layers in order: a first convolutional layer with a 3×3 kernel and ReLU activation, a second convolutional layer with a 3×3 kernel and ReLU activation, a densely connected layer

[1]https://www.tensorflow.org/versions/r1.1/get_started/mnist/beginners
[2]https://www.tensorflow.org/versions/r1.1/get_started/mnist/pros

with ReLU activation and dropout regularization, and a final layer with softmax activation. It also uses cross entropy as its loss function. This model is expected to achieve approximately 99% test accuracy on non-adversarial MNIST examples.

We modify the model training code to extract the condition number $\kappa(\boldsymbol{W})$ of the weight matrices $\boldsymbol{W}$ during each epoch of training. We calculate the condition number using the L_1, L_2, L_∞, and Frobenius norm. For the simple model, there is only one weight matrix corresponding to the single softmax layer. For the convolutional model, there are four weight matrices corresponding to each layer, which we refer to as `conv0`, `conv1`, `dense`, and `logits`. We extract four sets of condition numbers from the simple model's training process, and three sets from the CNN's training process; these counts are determined by the existing model implementations.

We adapt an existing implementation of FGSM and FGMT attacks [2] to generate adversarial examples of MNIST images for each model. Adversarial examples are produced for ϵ within the range [0.005, 0.3] with a step size of 0.05. We test each model against each adversarial example and record its test accuracy.

V. Results

A. Adversarial Examples

Figures 1 and 2 depict adversarial examples of a single digit generated with different ϵ values for FGSM and FGMT respectively. These figures show that the human readability of the digits declines sharply past $\epsilon = 0.01$.

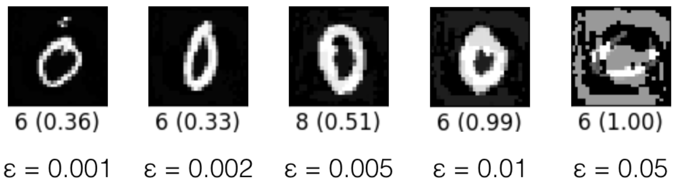

Fig. 1. **Adversarial Examples generated by Fast Gradient Sign Method (FGSM) for varying** ϵ. The visible difference in perturbation is clear for FGSM, declining sharply past $\epsilon = 0.01$

Figure 3 shows a few examples of adversarially generated MNIST images using FGSM with $\epsilon = 0.01$, along with the number that each digit most commonly misclassified as with the given error. The

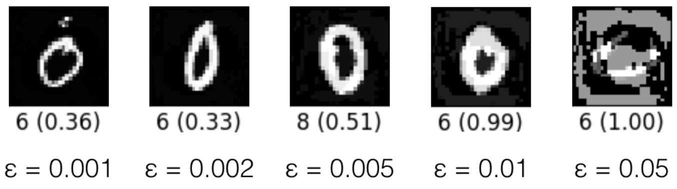

Fig. 2. **Adversarial Examples generated by Fast Gradient Method with Target (FGMT) for varying** ϵ. As for FGSM, the visible difference in perturbation is clear for FGMT, declining sharply past $\epsilon = 0.01$

examples produced using FGMT are very similar to those shown in Figure 3, so we do not show them separately.

As can be seen in Figure 3, the digits are still easily identifiable to human observers, but consistently fool both the simple and convolutional models, despite the relatively small perturbation.

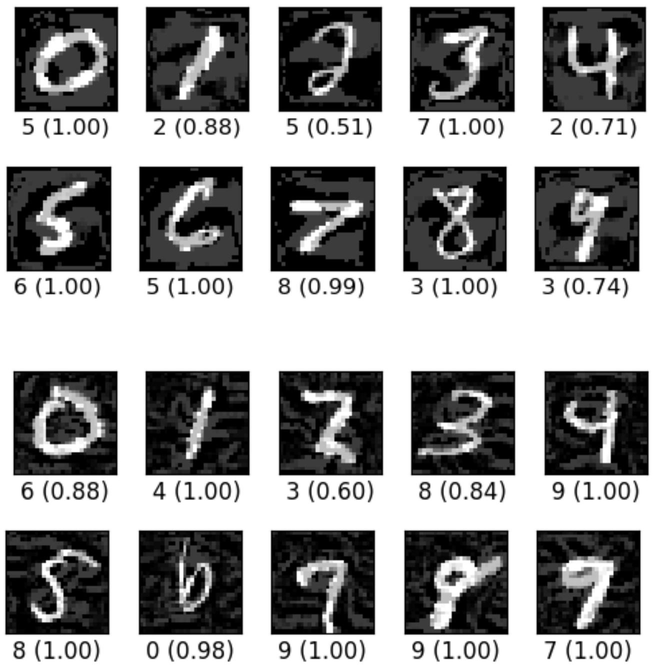

Fig. 3. **Adversarial Examples generated by Fast Gradient Sign Method (FGSM) for ϵ = 0.01**. Adversarial FGSM MNIST examples from each class are shown for the simple model (*top set*) and the convolutional model (*bottom set*).

B. Model Performance

Table I and Table II show the results of the simple and convolution model performances on FGSM and FGMT attacks, respectively. The

TABLE I
MNIST ACCURACIES AND CONDITIONING WITH FGSM ATTACK.

	simple	conv			
tr. acc.	0.918	0.989			
cond. no.		conv0	conv1	dense	logits
$L_1\ \kappa(W)$	159.33	518.48	2.08e4	1.93e9	55.29
$L_2\ \kappa(W)$	36.48	140.31	3.22e3	1.83e9	19.32
$L_\infty\ \kappa(W)$	115.99	262.98	5.02e4	8.76e10	83.35
$L_F\ \kappa(W)$	62.33	189.11	8.53e3	1.89e9	36.84
FGSM test acc.					
ϵ = 0.001	0.903	0.99			
ϵ = 0.002	0.847	0.984			
ϵ = 0.005	0.456	0.932			
ϵ = 0.01	0.088	0.583			
ϵ = .05	0.034	0.006			
ϵ = 0.1	0.031	0.005			
ϵ = 0.15	0.031	0.004			
ϵ = 0.2	0.031	0.004			

TABLE II
MNIST ACCURACIES AND CONDITIONING WITH FGMT ATTACK.

	simple	conv			
tr. acc.	0.917	0.993			
cond. no.		conv0	conv1	dense	logits
$L_1 \kappa(W)$	180.12	460.18	1.83e4	7.42e8	59.82
$L_2 \kappa(W)$	35.55	111.74	2.94e3	6.94e8	20.60
$L_\infty \kappa(W)$	125.40	257.57	6.52e4	3.56e10	86.20
$L_F \kappa(W)$	63.01	168.89	8.14e3	8.57e8	36.46
FGMT test acc.					
ϵ = 0.001	0.920	0.991			
ϵ = 0.002	0.921	0.991			
ϵ = 0.005	0.893	0.983			
ϵ = 0.01	0.637	0.913			
ϵ = .05	0.000	0.000			
ϵ = 0.1	0.000	0.000			
ϵ = 0.15	0.000	0.000			
ϵ = 0.2	0.000	0.000			

tables include the original model test accuracies on standard (non-adversarial) examples, the condition numbers of their weight matrices calculated under different norms, and the test accuracies on adversarial examples generated with different ϵ. The non-adversarial test accuracies of the simple and convolutional models are around 91-92% and 98-99% as expected (Sec. III-B).

As predicted, simple and convolutional models both perform significantly worse on adversarial inputs than on regular inputs. In particular, all reported adversarial test accuracies between 0.006 and 0.0 are consistently worse than the baseline random guessing of digit labels in the 10-class problem, which would result in an accuracy of 0.10.

While small FGSM perturbations (ϵ of 0.002 and 0.005) immediately show a reduction in accuracy from 0.99, small FGMT perturbations barely reduce the accuracy from 0.99; only at $\epsilon = 0.01$ does the accuracy drop to 0.91. However, for larger FGMT perturbations (ϵ of 0.05, 0.1, 0.15, and 0.2), the accuracy is zero, implying that the attack "always" works and consistently fools the classifier for the degree of precision reported. We compare this to large FGSM perturbations, which do not always fool the classifier (though it becomes less than 1% accurate).

The accuracies decline with increasing ϵ, then plateau after a certain point (Table I and II). This also follows our expectations, as increasing perturbations should obscure the defining features of the digit more, making it more difficult for the model to identify it. After a certain point, there is so much noise in the images that adding even more perturbation does not affect model performance.

The different norms (L_1, L_2 L_∞, L_F) used to calculate condition number follow the same trend over training steps for a given model. We do not observe any emergent trends when comparing the calculated norms (Table I and II).

C. Evolution of Condition Number

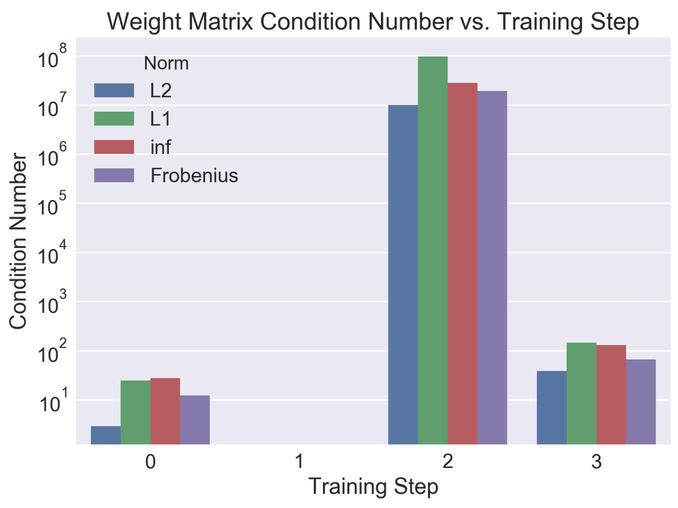

Fig. 4. **Evolution of condition number $\kappa(\boldsymbol{W})$ over optimizer training iterations for the simple model**. Condition number is calculated as given in the formulas in III.B, meaning that values of 0 indicate a norm of 0 in the pseudoinverse. Under the NumPy calculation, these return NaNs.

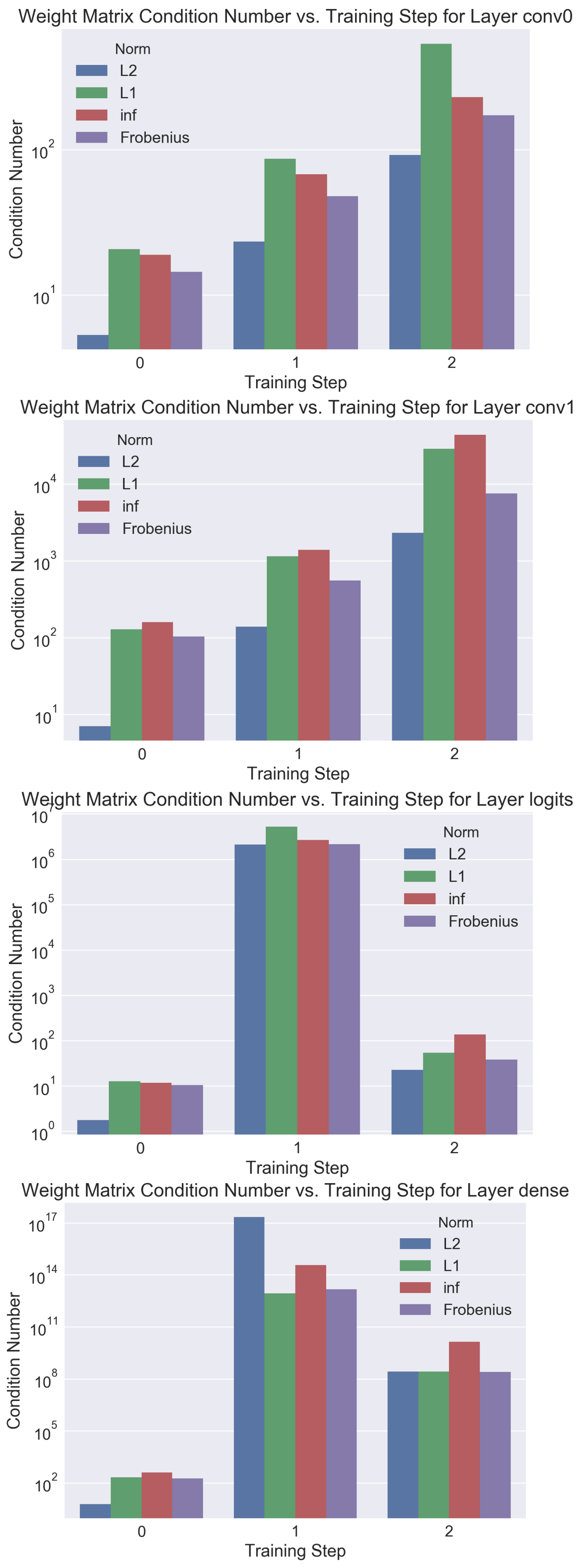

Fig. 5. **Evolution of condition number $\kappa(W)$ over optimizer training iterations for the convolutional MNIST model.**

Convolutional layers are initially four-dimensional; to calculate condition numbers for convolutional layers, a weight matrix of dimensions (K_x, K_y, C_{in}, N_f) is reshaped to dimensions $(K_x K_y C_{in}, N_f)$ [10].

Fully-connected layers. We see that all fully-connected layers, whether in the simple model or the CNN, follow a pattern: they begin with a relatively low condition number, experience a large spike (in the simple model, this is to the point of having 0 for the pseudoinverse norm after the first training iteration), and then settle at a condition number that is higher than the initial, but much lower than condition numbers during the optimization process.

There are many factors involved, but this fits our intuition. A randomly initialized weight matrix is likely to have a low condition number given that it is likely to be far from singular. The weight matrix after the first iteration is generally very high (sometimes singular), an observation which is consistent with the fact that the weight matrix is heavily influenced by individual samples after one training step and has not yet generalized well. The final model has a lower condition number because it has generalized and become more robust; it is not as well-conditioned as the random initialization because it has learned certain patterns in the data and is likely to be closer to singular than a randomly initialized matrix.

Convolutional layers. Convolutional layers seem to follow a different pattern. They grow more and more ill-conditioned as training progresses, ultimately reaching moderate condition numbers. Again, we present this as an observation; the reasoning is based on several interacting factors. There is some intuition behind this effect: convolutional layer weight matrices deal with patches of a raw input image and have many input and output channels, whereas fully-connected layer weight matrices deal with a single input vector that has been processed by convolutional layers and produce a single output vector. Consequently, convolutional layer weight matrices deal with low-level features (specifically pixel data) that need to be gathered into high-level features and are only tractable with a high volume of parameters. As the model trains and convolutional layers strengthen their ability to draw high-level representations from low-level features, we may expect to see more active parameters and a higher condition number.

The fully-connected layers receive high-level representations from the convolutional layers. Their weight matrices thus have fewer significant inputs to transform and are further from singularity.

VI. Conclusion

In this paper, we train a simple and convolutional model on the standard MNIST dataset, generate adversarial examples based on FGMT and FGSM attack methods, and evaluate how the conditioning of the models relates to their performance under adversarial attacks.

We find that in the non-adversarial case, the convolutional model performs better than the simple network. When deployed against adversarial inputs, both models show significant drops in test accuracy under the same level of ϵ-perturbation. However, the accuracy of the simple model under FGSM attacks is almost a full order of magnitude larger than the accuracy of the convolutional model under FGSM attacks. This observation suggests that the convolutional model *overfits* more than the simple model; it is more accurate on test data chosen from the same distribution that the training data is chosen from, but is also more susceptible to the FGSM adversarial attack. Both models are equally susceptible to the FGMT attack due to its targeting ability and drop to $< 1\%$ accuracy, so this result is less salient.

In the context of condition numbers, we observe that the weight matrix of the simple model tends to have similar or lower condition numbers compared to the fully connected layer weight matrices of the CNN. The convolutional layer weight matrices generally have lower condition numbers than the fully connected layer weight matrices for either model, but the condition numbers increase steadily over time. This finding implies that the CNN is generally more complex, supporting the hypothesis that complexity and ill-conditioning heighten susceptibility to adversarial attacks.

References

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Gong, Z. Adversarial algorithms in tensorflow, Jan. 2018.

[3] Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples, 2014.

[4] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition, 2015.

[5] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1* (USA, 2012), NIPS'12, Curran Associates Inc., pp. 1097–1105.

[6] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (Nov 1998), 2278–2324.

[7] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks, 2017.

[8] Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. Practical black-box attacks against machine learning, 2016.

[9] Sarle, W. S. Ill-conditioning in neural networks.

[10] Singh, M., Sinha, A., and Krishnamurthy, B. Neural networks in adversarial setting and ill-conditioned weight space. *CoRR abs/1801.00905* (2018).

[11] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)* (2015).

[12] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. *CoRR abs/1312.6199* (2013).

[13] Tramer, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. Ensemble adversarial training: Attacks and defenses.

[14] Tramer, F., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. The space of transferable adversarial examples, 2017.

[15] van der Smagt, P., and Hirzinger, G. Solving the ill-conditioning in neural network learning. In *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 191–203.