Effects of Image Resolution on Accuracy and Robustness in CNNs and Smoothed Classifiers

Daniel A. Nissan
Department of Information Systems Engineering
Technical University Of Vienna
Vienna, CT 1220
e11810773@student.tuwien.ac.at
Supervised by Univ.Ass.in Dipl.-Ing. Tamara Drucks

Abstract

Generalisation matters where training data ends and test data begins. It is easy for deep neural networks (DNNs) to memorize the training data set but separating the feature space too strictly leads to poor performance on novel data points, whereas a linear separation in most cases does not suffice for the training data to begin with. This bias-variance trade-off leads to the study of generalisation which is what allows us to walk the fine line between overfitting and underfitting. There are many miscellaneous strategies which can enhance the effectiveness of generalisation in machine learning models, such as data augmentation, selection of appropriate activation functions, hyperparameter tweaking, early stopping, and batch normalisation. However, DNNs' accuracy is more reliable than their robustness. A promising and simple technique to obtain certified robustness is randomized smoothing.
In this paper, I analyse accuracy and robustness of CNN classifiers before and after the application of randomized smoothing in the context of the MNIST dataset. Furthermore, I examine how this behaviour extrapolates to different input space dimensionalities and network architectures.
Results showed: Modern CNN architectures can correctly classify noised MNIST samples under heavy noise that is out of range for regular smoothing (σ=255) therefore completely flattening the trade-off across all input scales in accuracy and robustness one would usually expect between regular and smoothed classifiers. This is hypothesized to be a consequence of the low complexity of MNIST thus warranting further examination in more complex datasets.

1 Introduction

1.1 Theoretical Foundation of Generalisation

Gradient descent is the main driver of generalisation in neural networks. A practical and computationally efficient method of computing it is called stochastic gradient descent. The goal of gradient descent is to minimize the loss function $f(x) := \mathbb{R}^d \rightarrow \mathbb{R}$. Generally in order to calculate the gradient $\nabla f(x) = (\frac{\partial f}{\partial x_1} ... \frac{\partial f}{\partial x_i})$ f has to be differentiable. Having found the gradient the algorithm descends into the found direction by a distance labeled as step size.
GD can still be performed on and generalised to non-convex functions albeit without the guarantee that convex functions have no saddle points which asserts that all local minima are global minima. Lee et al. [2016] showed that GD in the case of non-convex functions still manages to converge at local minima in NP time without getting stuck on saddle points given an appropriate step size.

In the context of machine learning algorithms, the goal of GD and other optimizers is to minimize the loss function $\mathcal{L}$. $\mathcal{L}$ is an essential function that allows us to evaluate the prediction error of classifiers.
Let $f(x)$ be a true labeling function that maps from an input space $\mathcal{X}$ to a label space $\mathcal{Y}$ and $h(x)$ a hypothesis e.g. a neural network that aims to approximate the behaviour of f without knowing the underlying hidden distribution $\mathbf{D}$ from which the inputs are sampled.
The expected loss or generalisation error is given by the expected value of the mismatch between $h(x)$ and $f(x)$ over all possible inputs x drawn from some hidden distribution D:

$$\mathcal{L}_{\mathcal{D},f} := \mathbb{P}_{x\sim\mathcal{D}}[h(x) \neq f(x)]$$

The expected loss is a value that can be interpreted as how far off the model's prediction was on a before unseen value from the true value. Therefore, minimizing it means minimizing the error in the model's predictions.
Regular loss or training loss on the other hand acts as a measurement for how far off a prediction is in the training context. Notable examples would be mean squared error and mean absolute error.

Shalev-Shwartz and Ben-David [2014]

1.2 Robustness

Despite the high accuracy DNNs are able to achieve on classification problems, their robustness to adversarial attacks still leaves room for improvement. Carlini and Wagner [2017]
These attacks may be as simple as a rotated image or an image with added Gaussian noise. Perturbations like this are often unrecognizable to the human eye and bound by a norm.
Formally: $f(x) \neq f(x + \Delta x)$ with the restriction on size of the perturbation $||\Delta x||_p \leq \varepsilon$ given by a norm l_p e.g. l_2, l_∞. Chen et al. [2021]
These norms can be used to construct a notion of distance in the normed vector spaces given by their d dimension dependent Minkowski distance:

$$(\sum_{i=1}^{d} |x_i - y_i|^p)^{\frac{1}{p}}$$

The most commonly used norms are

- l_2: Euclidian norm: Outliers have more impact because each dimension is squared
- l_∞: Infinity norm: Selects the highest value of the vector

There exists a fundamental connection between a neural network's robustness against l_2 and l_∞ perturbations and a property called bounded Lipschitzness. Zhang et al. [2022]
Let $(X, d_x), (Y, d_y)$ be two metric spaces where d_n denotes their respective metric and f a function that maps from Y to X. f is called Lipschitz continuous if there exists a constant $K \in \mathbb{R}$ such that

$$dy(f(x_1), f(x_2)) \leq K d_x(x_1, x_2) \quad | \quad \forall x_1, x_2 \in X$$

On an intuitive level, this makes sense because bounded Lipschitzness implies a certain smoothness of the classifier which again makes it less sensitive to random noise. We differentiate between local

and global bounds for the Lipschitz constant because on a global scale, we can guarantee a smaller radius of smoothness than on the scale of individual inputs. Huang et al. [2021]

Randomized smoothing is a similar technique that makes use of this smooth property to achieve certifiable robustness:

1.3 Randomized Smoothing

Similar to how perturbations can be applied to training data in a process called data augmentation which has been shown to increase robustness they can also be applied to a trained classifier through a process called randomized smoothing in order to obtain robustness guarantees. Panda and Roy [2021] This method requires no knowledge of the underlying classifier.
Let $f : \mathbb{R}^d \to [0, 1]$ be an arbitrary classifier that maps some input to the probability of belonging to a class and D a distribution [1] on $\mathbb{R}^d$. The classifier f can be converted to an associated smoothed classifier s such that $s(x)$ returns the most likely prediction of f when the input x is perturbed by some random noise sampled from D. Formally:

$$s(x) = \underset{c \in Y}{\operatorname{argmax}} \Pr_{\epsilon \sim D}[f(x + \epsilon) = c]$$

Inherent in the definition lies the guarantee of a certain robustness. Let P_a be the probability of the most likely prediction and P_b the runnerup:

$$a = \underset{a \in Y}{\operatorname{argmax}} \quad s(x)_a, \qquad P_b = s(x)_a$$

$$b = \underset{b \in Y, b \neq a}{\operatorname{argmax}} \quad s(x)_b, \qquad P_b = s(x)_b$$

Now consider a perturbation sphere of maximal l_2 radius ε that still produces the right result:

$$P_a(x) - P_a(x + \varepsilon) \leq ||\varepsilon||_2$$

Then if s is Lipschitz for a constant c the classifier is robust to perturbations of radius ε

$$||\varepsilon||_2 = \frac{1}{2} c(P_a - P_b)$$

Cohen et al. [2019]
This holds for randomized smoothing because it is known that the convolution with the Gaussian distribution independent of f

$$s(x) = (f * \mathcal{N}(0, \mathcal{I})(x))$$

is Lipschitz since by computing the gradient for the Gaussian density γ_σ:

$$\nabla s = f * \nabla \gamma \sigma$$

We obtain a σ dependent guarantee of

$$||\varepsilon||_2 = \frac{1}{2} \sigma(P_a - P_b)$$

at a given input x with a probability of 1-α due to the non-deterministic nature of this technique[2] and a Lipschitz constant of 1. Salman et al. [2019]

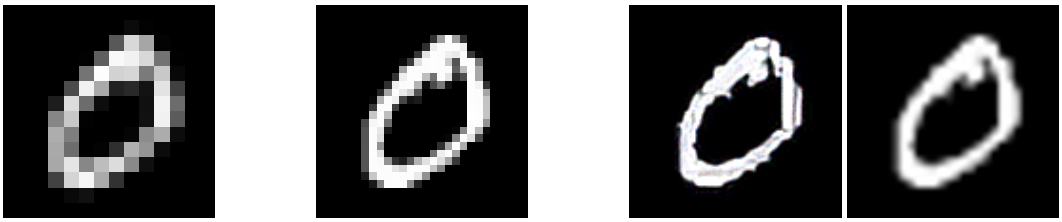
[1] In this experiment ϵ is always be sampled from the isotropic Gaussian distribution D so henceforth any unspecified perturbation will be sampled from $\mathcal{N}(x, \sigma^2 \mathcal{I})$ where $\mathcal{I}$ is the covariance matrix which in this case takes on the scalar value of 1.

[2] See 2.3

2 Experiment

2.1 Setup

The main goal is to compare a regular CNN to its corresponding smoothed classifier. The key metrics to be evaluated are accuracy and robustness of the models on various resolution scales of MNIST. The default resolution of MNIST is 28x28 so in order to achieve 56x56 I upscaled the data using the enhanced super-resolution adversarial generative network ESRGAN Wang et al. [2018] and 14x14 was resized using the Pythons' PIL libraries bilinear algorithm. A bilinearly interpolated data set acts as the control group for the upscaled dataset created by using the PIL libraries resize function.



(a) Downscaled 14x14 196px

(b) Normal 28x28 784px

(c) Upscaled ESR 56x56 3316px

(d) Control Bilinear 56x56 3316px

Figure 1: Resolution Samples

2.2 Hypothesis and Convolutional Neural Networks

The main motivation for using different resolutions is to observe and find potential advantages for practical application. Due to the fact that only models of each resolution category will be compared to each other only their relative performance matters.
The issue of scaling a network's size to the input resolution is an unsolved problem to this day. For that reason and due to hardware limitations I will be linearly scaling the network layer sizes but not the number of layers. Adding more layers usually aids in extracting new features which might not be as helpful in this experiment. Thambawita et al. [2021] All networks are scaled linearly from the base network by the same factor the inputs were scaled by from the following base architecture:

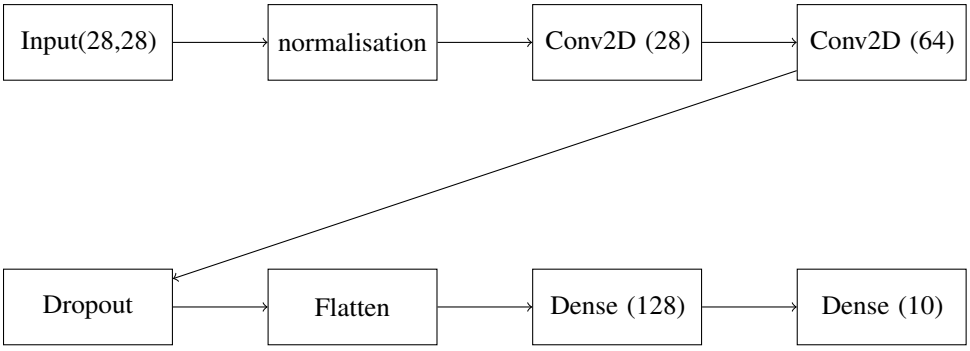


Figure 2: CNN Architecture

To account for any network scaling artifacts both the upscaled and the control CNN had a counterpart that had the same architecture as the default net with only the input layer being scaled to 56x56.

To fit the models the Adam optimizer was used with the default learning rate and the chosen loss function was sparse categorical cross-entropy. All six models were trained for 10 epochs.

2.3 Computing Randomized Smoothing

The practical implementation of a smoothed classifier makes use of a Gaussian majority vote. First n samples of noise $\varepsilon_1 ... \varepsilon_n$ are drawn and convolved with the input image x. Then $f(x + \varepsilon_i)$ is evaluated by the base classifier for all n noised inputs. If class A is the most often predicted class of all n samples by some threshold α the classifier answers A otherwise, it will abstain from making a prediction. In the pseudocode, the lower bound of the prediction is calculated via the p-value of a two-sided Bernoulli hypothesis test.

---

samples$\leftarrow$ SampleUnderNoise(f, x, σ, n)
votes$\leftarrow$ predictNoisedSamples$(f,$ samples$)$
$C_A, C_B \leftarrow$ top two counts in votes
if BinomP-Value$(C_A, C_A + C_B, 0.5) \leq \alpha$ **then**
    return C_A
else return ABSTAIN
end if

Cohen et al. [2019]
It is important to note that while in the original paper the smoothed classifier abstained from certain predictions in my implementation this scenario never arose and hence was not included in the implementation.

2.4 Robustness Test Using Fast Gradient Sign and FoolBox

Robustness tests were performed using the fast gradient sign method (FGSM).
Let θ be the parameters of a model. Then for a given input x and true label y_t the cost is given by $J(\theta, x, y_t)$. The FGSM evaluates the gradient ∇J and uses this information about direction to perturb the input with noise going the opposite direction of the gradient such that the model misclassifies the input.

$$x_{adv} = x + \epsilon \cdot \text{sign}(\nabla x J(\theta, x, y_t))$$

The result is an image that is indistinguishable to the human eye and is misclassified by the CNN.
Goodfellow et al. [2014]
The implementation by FoolBox was used in this experiment for the FGSM attack.
Rauber et al. [2017]
All other code was written in the Tensorflow framework.

3 Results

The tests were run on a sample size of $n = 1000$ for both accuracy and adversarial robustness and $\sigma = 0.15$ for the isotropic Gaussian noise. Randomized smoothing was performed with 10 samples per value.

Accuracy	CNN Accuracy	CNN Loss	RS Accuracy	RS Loss
Downscaled	0.982	0.0892	0.982	0.0860
Normal	0.98	0.97	0.98	0.1043
UpscaledAI	0.975	0.2463	0.975	0.1964
Control	0.974	0.1986	0.974	0.1973
UpscaledAI Small	0.975	0.2163	0.975	0.2045
Control Small	0.972	0.1860	0.972	0.1847

For the FGSM attack an ϵ value of 10 was chosen due to the high innate robustness of the models.

Adv. Accuracy	CNN Accuracy	RS Accuracy
Downscaled	0.975	0.975
Normal	0.973	0.973
UpscaledAI	0.916	0.916
Control	0.962	0.962
UpscaledAI Small	0.961	0.961
Control Small	0.957	0.957

From the data, it is evident that there is no relative performance difference between smoothed and non-smoothed classifiers in both accuracy and adversarial robustness. Upon further examination with σ values of up to 255 no difference was found. It turns out the CNNs performance is too high and completely offsets the effect of any random noise. I hypothesize that this is due to the simplistic nature of the MNIST dataset as it has already been established prior that randomized smoothing sacrifices performance for accuracy. Cohen et al. [2019] Since the evaluation was carried out on before unseen data overfitting can also be excluded as potential cause.

Another observation is that the performance gap between the downscaled and the normal network is small, indicating that downscaling simple data might be a viable strategy to reduce workload. Note that while interesting, assessing performance is outside the scope of this paper. Future research could examine the effect of different resolutions in more complex datasets as well as examining scenarios with various levels of noise-augmented training data.

4 Related Work

Randomized smoothing is a recent development in the field of machine learning that was first proposed for stochastic optimisation in 2012. Duchi et al. [2012] It has since been refined and led to significant advances in the area of security and robustness both practical and theoretical. Cohen et al. [2019] There is still ongoing research and room for improvement in the implementation. Mohapatra et al. [2020]

Appendix

Source code available at:
https://github.com/Everfade/Effects-of-Image-Resolution-on-Accuracy-and-Robustness-in-CNNs-and-Smoothed-Classifiers

References

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. *IEEE Symposium on Security and Privacy*, May 2017. doi: 10.1109/sp.2017.49.

Sizhe Chen, Qinghua Tao, Zhixing Ye, and Xiaolin Huang. Going far boosts attack transferability, but do not do it. *CoRR*, abs/2102.10343, 2021. URL https://arxiv.org/abs/2102.10343.

Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. *CoRR*, abs/1902.02918, 2019. URL http://arxiv.org/abs/1902.02918.

John C. Duchi, Peter L. Bartlett, and Martin J. Wainwright. Randomized smoothing for stochastic optimization. *SIAM Journal on Optimization*, 22(2):674–701, 2012. doi: 10.1137/110831659. URL https://doi.org/10.1137/110831659.

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

Yujia Huang, Huan Zhang, Yuanyuan Shi, J. Zico Kolter, and Anima Anandkumar. Training certifiably robust neural networks with efficient local lipschitz bounds. *CoRR*, abs/2111.01395, 2021. URL https://arxiv.org/abs/2111.01395.

Jason D. Lee, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. Gradient descent converges to minimizers, 2016. URL https://arxiv.org/abs/1602.04915.

Jeet Mohapatra, Ching-Yun Ko, Tsui-Wei Weng, Sijia Liu, Pin-Yu Chen, and Luca Daniel. Rethinking randomized smoothing for adversarial robustness. *CoRR*, abs/2003.01249, 2020. URL https://arxiv.org/abs/2003.01249.

Priyadarshini Panda and Kaushik Roy. Implicit adversarial data augmentation and robustness with noise-based learning. *Neural Netw.*, 141(C):120–132, sep 2021. ISSN 0893-6080. doi: 10.1016/j.neunet.2021.04.008. URL https://doi.org/10.1016/j.neunet.2021.04.008.

Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models, 2017. URL https://arxiv.org/abs/1707.04131.

Hadi Salman, Greg Yang, Jerry Li, Pengchuan Zhang, Huan Zhang, Ilya Razenshteyn, and Sebastien Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers, 2019. URL https://arxiv.org/abs/1906.04584.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms.* Cambridge University Press, 2014. ISBN 978-1-10-705713-5.

Vajira Thambawita, Inga Strümke, Steven A. Hicks, Pål Halvorsen, Sravanthi Parasa, and Michael A. Riegler. Impact of image resolution on deep learning performance in endoscopy image classification: An experimental study using a large dataset of endoscopic images. *Diagnostics*, 11(12), 2021. ISSN 2075-4418. doi: 10.3390/diagnostics11122183. URL https://www.mdpi.com/2075-4418/11/12/2183.

Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Chen Change Loy, Yu Qiao, and Xiaoou Tang. ESRGAN: enhanced super-resolution generative adversarial networks. *CoRR*, abs/1809.00219, 2018. URL http://arxiv.org/abs/1809.00219.

Bohang Zhang, Du Jiang, Di He, and Liwei Wang. Rethinking lipschitz neural networks and certified robustness: A boolean function perspective, 2022. URL https://arxiv.org/abs/2210.01787.