Analyzing the Effects of Data Corruptions on Machine Learning Models

George Shen

Received November 06, 2024
Accepted January 14, 2025
Electronic access February 15, 2025

The main objective of this paper is to discover how different distortions and other factors that are involved in image classification will affect the accuracy in interpreting and classifying images. The experimentation includes training and testing with a noisy dataset and a normal dataset. We also experimented with different amounts of hidden layers and different learning rates to determine how those factors affect the overall training and testing accuracies of the machine learning model. We used a linear neural network and a greyscale dataset consisting of images of clothing items. Our primary finding was that if you know the type of distortions that will be encountered at test time, training on the same type of distortion will result in the greatest accuracy. As may be expected, training and testing on uncorrupted images achieved the highest accuracy, of around 80.63%. However, if we know we will encounter noisy images at test time, we can do 27.78% better by training on noisy images rather than training on uncorrupted images. We also conducted some experiments varying hyperparameters and concluded that larger hidden layers resulted in slightly higher accuracies, but training at a fast learning rate achieves higher accuracies.

Introduction

The evolving field of machine learning has been prominent for the past few years with an increase in the demand for machine learning tools and analysis in a variety of fields. Machine learning has lots of different applications, including decision making in businesses, fraud detection, and pattern finding in biological sciences. Machine learning is important because it can allow organizations to improve their weaknesses and make thoughtful decisions.

One of the most important applications of machine learning is for image recognition. Convolutional neural networks have allowed computers to process and classify images with high accuracy. They have also allowed for effectively capturing the spatial dependencies in an image. Image recognition has various critical applications. In the medical field, it is used for identifying tumors in MRI scans and also for detecting anomalies in X-ray scans. Image recognition is also used for face ID in mobile phones, in which the device recognizes the face of the owner and locks the phone if the face of the human does not match the owners. Lastly, image recognition is useful in self-driving cars, in which they have to spot and recognize pedestrians and cars.

When we deploy machine learning models in the real world, we find that real data often differs from the data the model was trained on. Robustness research aims to ensure that models can perform well under a variety of settings. Adversarial robustness involves training models to withstand intentionally crafted perturbations designed to fool the model. On the other hand, natural robustness refers to models that can withstand "natural" perturbations such as camera effects, weather effects, or simple image manipulations. Techniques like adversarial training, where models are trained on both clean and adversarial examples, help improve resilience to such attacks. Training on corrupted images helps improve the accuracy of image classification, and it ensures that the models can handle noise and different kinds of image corruptions. This is important because of the need to train and then deploy models in the real world where the model may encounter data with a range of perturbations.

In this paper, we focus on natural robustness by studying two image perturbations: the addition of Gaussian noise, and horizontal flips. We find that we can obtain the best test results by training on a dataset with the same perturbations as the test set, for instance, both training and testing on a noisy dataset. Additionally, we study several hyperparameters to determine their effect on training. The main objectives of this study include:

- To evaluate the performance of the neural network model using normal and noisy image datasets.
- To perform test performance analysis using several noisy images by using random flip scenarios.
- To evaluate the impact of the hidden layer sizes and learning rates on the accuracy of the models.

Background/Related Works

Researchers have already studied how to protect their data from being corrupted. Dong et al. have researched this, in which

© The National High School Journal of Science 2025

they focused on convolutional neural networks (CNNs) trained with adversarial training techniques and assessed them against ℓ_p-norm-based adversarial perturbations, including untargeted, targeted, and black-box attacks[1]. While their research provides comprehensive evaluations, it does not address the performance of models under noise corruptions, which are also important in the real-world. Another prominent study was conducted by Lu and Weng, in which they examined the problems with image classification[2]. Their research emphasized techniques used for improving classification accuracy and highlighted the importance of preprocessing steps, such as feature extraction and multisource data integration. However, their studies lack evaluation on adversarial robustness, which limits their applicability. Additionally, Li et al. conducted research on Hyperspectral image (HSI) classification[3]. Their research goal is to accurately classify challenging data that would be difficult to classify for traditional machine learning methods. They emphasized deep learning's benefit over traditional machine learning methods in addressing HSI's nonlinear and high-dimensional characteristics. However, their survey focused on classification accuracy and strategies to handle limited training samples, without addressing robustness to corruptions. Madry et al. researched about potential danger from adversarial attacks[4]. Their work demonstrated that deep neural networks are very vulnerable to adversarial attacks, and so they developed a method to train the networks to have greater resistance against these types of attacks. However, their focus was limited to robustness against adversarial perturbations and not the model's performance under natural corruptions. Also, Hendrycks and Dietterich studied how well image classifiers can handle corruption and disturbances[5]. The research paper also explores ways to improve robustness. They introduced two datasets, ImageNet-C and ImageNet-P, one which focuses on corruption robustness and the other on how to handle common disturbances. From the results, they found out that certain defenses against attacks can improve performance on common disturbances. While their work focused on enhancements for corruption robustness, they did not explore adversarial robustness or a combined evaluation of these scenarios. Additionally, Croce et al. proposed AutoAttack, a standardized ensemble of white- and black-box attacks to evaluate adversarial robustness more accurately[6]. They also established a comprehensive leaderboard for benchmarking adversarial robustness. While their focus is primarily on adversarial robustness, our study extends adversarial training to include resilience against natural corruptions. In another study, Augustin et al. introduced RATIO, a training procedure integrating adversarial training for both in-distribution and out-of-distribution samples[7]. Their method achieved the best adversarial robustness on CIFAR-10 while limiting the accuracy trade-off associated with adversarial training. In our study, we also focus on adversarial training, but with natural corruptions. Mitra et al. have investigated the trade-off of uncertainty measurements, natural corruption robustness, as well as the performance of CNNs[8]. They revealed that post-hoc pruning can improve both uncertainty calibration and natural corruption robustness without compromising clean accuracy. Our study aims to focus more on the natural corruption and how well the model performs. Mintun et al. analyzed the relationship between data augmentations and robustness to test-time corruptions, showing that augmentation strategies can significantly impact corruption robustness[9]. They discovered that test percentage is higher when training on similar augmentations and that data augmentations may not be generalized beyond a benchmark. Rather than focusing on the relationship between augmentations and corruptions for robustness, our work focuses on image classification performance optimization. Lastly, Schneider et al. argued that ImageNet-C benchmarks underestimates model robustness by not accounting for unsupervised online adaptation to corruption statistics[10]. Their results demonstrate significant robustness improvements through adapted batch normalization statistics. While their approach enhanced robustness to natural corruption, it did not address adversarial attacks.

Methods/Implementation Basics

For this project, we used a feedforward neural network to classify images from FashionMNIST[11]. FashionMNIST is a dataset of 28x28 pixel grayscale images that depict 10 different clothing items. We used this dataset because we did not have access to GPUs, so we had to use small images and a small neural network. One of the most standard datasets for small scale image processing is MNIST, which is 28x28 handwritten digits[12]. FashionMNIST has similar features to MNIST since it has images of the same size and it also has 10 classes, but it can be a more interesting prediction task because it contains images of a variety of clothing items rather than handwritten digits. We found FashionMNIST to be the best tradeoff between size and interest. It contains 60,000 train samples and 10,000 test samples. See figure 1 (left) for an example data point.

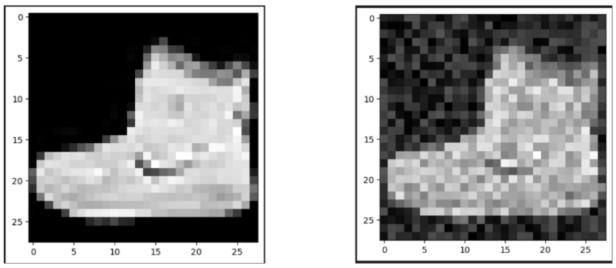

Figure 1 (left): A FashionMNIST image of a shoe without any noise added. Figure 1 (right): The same FashionMNIST image with noise added.

Similarly, to our choice of dataset, we chose a simple linear neural network for computational reasons. This network is much faster to train than more intricate architectures. Although the

© The National High School Journal of Science 2025

small scale of our network and dataset is a limitation, we still find interesting insights from our setup. We implemented our neural network using Python and Pytorch[7]. The base network we used has 5 layers. We flatten each input image and then feed it through a series of linear and ReLU layers. See figure 2 for an architectural illustration of our model.

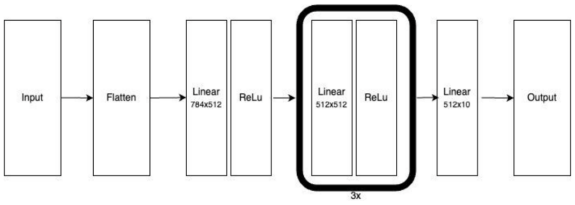

Figure 2

We split the data into training and test sets and put each set into a data loader. We then implemented a training loop which involves passing data through the model, evaluating the loss, and updating the weights of the model.

Our main experiment was to study the performance of a network on classifying normal and noisy images. We created an AddNoise class to transform our data. This class adds noise of a specified strength to each image. See figure 1 (right) for an example image with noise added. We created four datasets/data loaders: normal train, noisy train, normal test, and noisy test. In standard training, we would train on the "normal train" dataset and test on the "normal test" dataset. However, we would expect a performance drop, which we did verify in practice, when training on "normal train" and evaluating on "noisy test". Therefore, we run additional experiments where we train on a "noisy train" in the spirit of Madry et al. We selected a noise level of 0.5 since it seemed to corrupt the image sufficiently without destroying all information.

We also chose it based on visual observations and because it was in the middle of the spectrum. We ran 50 epochs for each of the different combinations and noted down the resulting training and test accuracy for each one. We used a batch size of 64, a default learning rate of 10^{-3}, and the Stochastic Gradient Descent (SGD) optimizer. We also tried to change the noise level to see how that would affect the resulting accuracy. All of the hyperparameters are additionally listed in Table 1.

Additionally, we implemented a flip image class, in which there is a probability to flip the image over the x axis and the y axis. We experimented with different probabilities to see how that would affect the final accuracy after training for 50 epochs.

We also changed the learning rate and also varying the amount of hidden layers. We experimented with different learning rates and hidden layers to see how that would affect the final accuracy after training for 50 epochs. See figure 3 for a visual representation of the methodology used in our implementation.

Name	Parameters
Number of Epochs	50
Batch Size	64
Metrics	Accuracy and Loss
Optimizer	SGD
Activation Functions	ReLu

Table 1

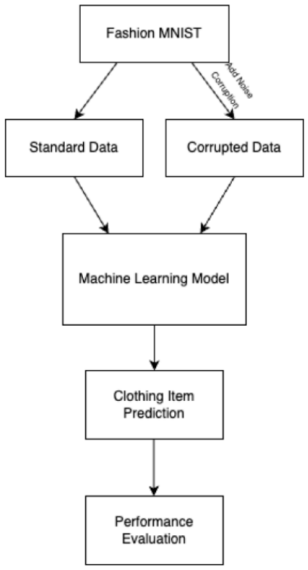

Figure 3

Results

In Figure 4, we show a matrix of the testing accuracy when training on normal/noisy data and testing on normal/noisy data. Darker colored squares represent higher testing accuracy. For example, the upper left quadrant achieves the highest testing accuracy of 80.63% when training and testing on normal data. Each number is an average over 4 training runs with the standard deviations included as well. We see that training on normal, testing on normal is similar to training on noisy and testing on noisy. Additionally, we see that there is a drastic difference when it comes to training on normal and testing on noisy versus training on normal and testing on normal.

In Figure 5, the bar graph shows the accuracies of image classification after training and/or testing with a chance of flipping the images. The red bars show the accuracy for training with the normal unflipped dataset and test on a 50% chance of flipping each image. The green bars show the accuracy for both training and testing on a 50% chance of flipping each image.

The purple bars show the accuracy for training with the nor-

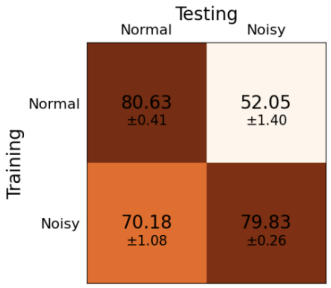

Figure 4: Testing accuracy with standard deviations across several training and testing settings.

mal unflipped dataset and test on a 70% chance of flipping each image. The results show that training and testing with 50% chance of flipping each image performs better than the others.

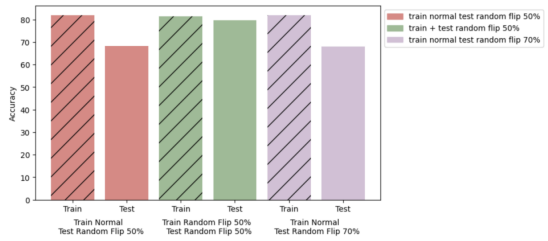

Figure 5: Bar graph showing the accuracies of image classification after training and/or testing with a chance of flipping the images

In figure 6: the graph shows the training and testing accuracy performances of training and testing on different hidden layer sizes. The red bars show the accuracies for training and testing on hidden layers of size 675. The green bars show the accuracies for training and testing on hidden layers of size 300. The results show that the performances when training and testing with hidden layers of size 675 and hidden layers of size 300 are very similar, but with the training and testing with a bigger size performing slightly better.

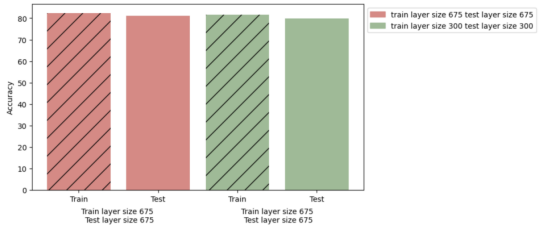

Figure 6: Bar graph showing the training and testing accuracy performances of training and testing on different sizes of hidden layers

In figure 7, the graph shows the training and testing accuracies for different learning rates. The red bars show the accuracies for training and testing with a learning rate of 0.1. The green bars show the accuracies for training and testing with a learning rate of 10^{-5}. The results show that the accuracies for training and testing at a rate of 10^{-5} are significantly lower than training and testing at a rate of 0.1.

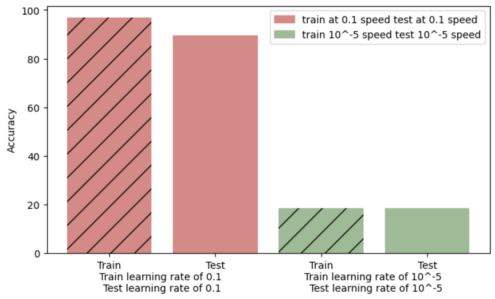

Figure 7: Bar graph showing the training and testing accuracies for different learning rates

Analysis

When we compare adding noise to a dataset versus just the original dataset, as shown in Figure 4, we can see that training with normal data and testing with noisy data gives the lowest accuracy for classifying images. This might be because the network was trained to classify normal images and not noisy images. Since noisy images are distorted, the network would basically be trying to identify photos that it has never seen before. Without the familiarity with the actual with-noise image, the network has a lower accuracy when classifying the images.

The highest accuracy is revealed when we trained on normal data and tested on normal data. From these results, we can hypothesize that the model is better at classifying normal images and can train faster with normal images. However, if we know that the model will encounter noisy images at test time, it is best to also train on noisy images, since training on normal data and testing on noisy data did worse than training on noisy data and testing on noisy data. However if you train on noisy and test on normal, noisy is harder in a way, as you are adding extra information to the image. As a result, noisy data is basically saying that we are adding complexity onto an image. As an analogy, if we only do the hardest questions for practice and are tested on easier questions, then we will do better than if we practiced on easy questions and tested on hard questions. Therefore, these reasons could explain why training on normal and testing on noisy resulted in a lower accuracy than training on noisy and testing on normal, though both resulted in lower accuracies than training and testing on the same dataset.

Figure 7 shows the different learning rate during training. Training and testing with noise, horizontal flip, and with different sizes of hidden layers made less impact than the different learning rate. These different experiments all had accuracies in the 65-85% relative percent range. However, when I trained with a faster learning rate, the training and testing accuracy increased to 96.9% and 89.6%. From this conclusion, we can hypothesize

© The National High School Journal of Science 2025

that a faster learning rate will lead to the network making more calculations and analyzing images faster, therefore, it will more correctly classify the images in the FashionMNIST dataset. Alternatively, the increase in learning rate will make the model converge faster. From this, we can also hypothesize that if we train the model for more epochs but at a slower learning rate, the accuracy will eventually match the one of the faster learning rate.

The results of the training process using different sizes of hidden layers show that training and testing with smaller hidden layers results in lower accuracies. That might be because hidden layers help analyze the images more, and so if there are bigger hidden layers, the network will be able to better process information in the images, therefore resulting in a higher accuracy.

Conclusion

In this paper, we experimented with noise, random horizontal flips, different learning rates, and different sizes of hidden layers, and how that affects the accuracy of an image classification model.

The results for training and testing on noisy or normal data shows that training and testing on normal data achieves the highest testing accuracy of 80.63%. Training on normal and testing on normal is similar to training on noisy and testing on noisy. There is also a noticeable difference of approximately 28.58% when it comes to training on normal and testing on noisy versus training on normal and testing on normal.

The results for random flips show that training and testing with 50% chance of flipping each image performs better than the others, due to its testing accuracy of 79.7% when compared to the others which are 68.2% (for train on normal and testing on 50% flip) and 68.0% (for train on normal and testing on 70% flip).

The results for different sizes of hidden layers show that the performances when training and testing with hidden layers of size 675 and hidden layers of size 300 are very similar, but with the training and testing with a bigger size performing slightly better by around 1.2%. The results for different learning rates show that the accuracies for training and testing at a rate of 10^{-5} are significantly lower than training and testing at a rate of 0.1 by around 78.47% for training and 71.1 for testing.

Based on all of the results, we can conclude that a higher learning rate and larger hidden layers will result in more accurate image classification. We can also conclude that we should choose how to train our model based on what kind of images the model will encounter at test time. If we will only see normal images, we should train on normal data, but if we will see noisy images, then we should train on noisy data instead.

The main limitations in this study are due to computational constraints. For instance, we were unable to access GPUS, which restricted us to only access smaller datasets and models.

Because of this, we used a simple linear neural network and the FashionMNIST dataset. This was a decision mainly made on the fact that our findings will share a parallel with those found in other settings. Also, due to the lack of computing power, we were unable to access convolutional neural networks (CNNs) or transformers, which are used more frequently in advanced image classification tasks.

In future works, we can utilize GPUs to train and test on more complex models, like CNNs or vision transformers. With these more advanced computational resources, we can also examine the performance of these models on larger datasets like ImageNet, CIFAR-10, and CIFAR-100. This would result in a better understanding of how different kinds of distortions and image perturbations affect more advanced models in the real world. Also, future study can incorporate a more diverse range of distortions, like blur, occlusion, and lighting variations. This would better reflect the distortions that happen in the real world. Lastly, in future works, we could utilize more advanced training techniques, like adversarial training or data augmentation, which will lead to better model robustness.

References

1 D. Yinpeng, Q. Fu, X. Yang, T. Pang, H. Su and Z. Xiao, *Benchmarking adversarial robustness on image classification*, 2020.

2 L. Dengsheng and Q. Weng, *A survey of image classification methods and techniques for improving classification performance*, 2007.

3 S. Li, W. Song, L. Fang, Y. Chen, P. Ghamisi and J. A. Benediktssonet, *Deep learning for hyperspectral image classification: An overview*, 2019.

4 A. Madry, A. Makelov, L. Schmidt, D. Tsipras and A. Vladu, *Towards deep learning models resistant to adversarial attacks*, arXiv preprint arXiv: 1706.06083, 2017.

5 D. Hendrycks and T. Dietterich, *Benchmarking neural network robustness to common corruptions and perturbations*, arXiv preprint arXiv:1903.12261, 2019.

6 F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal and M. Hein, *Robustbench: a standardized adversarial robustness benchmark*, arXiv preprint arXiv:2010.09670, 2020.

7 M. Augustin, A. Meinke and M. Hein, *Adversarial robustness on in-and out-distribution improves explainability*, 2020.

8 P. Mitra, G. Schwalbe and K. Klein, *Investigating Calibration and Corruption Robustness of Post-hoc Pruned Perception CNNs: An Image Classification Benchmark Study*, 2024.

9 E. Mintun, A. Kirillov and S. Xie, *On interaction between augmentations and corruptions in natural corruption robustness*, 2021.

10 S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel and M. Bethge, *Improving robustness against common corruptions by covariate shift adaptation*, 2020.

11 H. Xiao, K. Rashul and R. Vollgraf, *Fashion-MNIST: A Novel Image Dataset for Benchmarking*, arXiv preprint arXiv: 1708.07747, 2017.

12 A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, *Pytorch: An Imperative Style, High-Performance Deep Learning Library*, 2019.

© The National High School Journal of Science 2025