Enhancing CNN Robustness By Using Data Augmentation and Oscillation Block

Hao Luo Haiyi Wang Manav Patel Christina Zeng

December 2024

Abstract

This project investigates how structural modifications can improve convolutional neural networks (CNNs) in image classification tasks. We applied preprocessing methods like rotation, zooming, and positional adjustments to simulate real-world image variations. We also added an oscillation block to capture "tilted" features in rotated data without introducing new training data. Additionally, we used data augmentation to increase the size and diversity of the training dataset. Our experiments showed that the oscillation block improved the model's performance on rotated data. However, for mixed or normal datasets, the performance was slightly lower than that of standard CNNs. Overall, these techniques resulted in a more robust model, especially for handling rotated data, but also highlighted performance trade-offs in different training conditions.

1 Introduction

Convolutional Neural Networks (CNNs) have become one of the most widely used deep learning architectures due to their efficiency and effectiveness in various machine learning tasks. According to Albawi et al. (2017), CNNs have gained popularity in fields such as image classification, computer vision, and natural language processing. CNNs consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers. The primary function of the convolutional layer is to extract features from an input image by applying smaller matrices (kernels or filters) that slide over the image to generate feature maps. The pooling layer helps reduce the dimensionality of feature maps while retaining important information, and the fully connected layer connects all neurons between layers, flattening the output from previous layers and passing it to the output layer for classification. This architecture allows CNNs to learn with fewer parameters and less data compared to traditional artificial neural networks (ANNs), making them computationally efficient. Additionally, CNNs are inspired by time-delay neural networks, where shared weights along the temporal dimension further reduce computation costs.

Liu et al. (2017) highlighted CNNs as the first successful deep learning architecture, achieving impressive results in a variety of applications. Image

classification, a crucial application of CNNs, involves assigning a label to an input image from a fixed set of categories. Applications of image classification span across various fields, including robotics, object identification, autonomous vehicles, and traffic signal processing. Feature extraction plays a vital role in image classification, as it helps represent images efficiently for classification tasks. CNNs applied to large-scale datasets learn image representations and reuse them for classification tasks, often outperforming other neural network architectures, as demonstrated by Wang Xi (2015) in their work on CIFAR-10 classification.

The goal of this project is to enhance the performance of CNNs for image classification tasks by incorporating an oscillation block to capture tilted features in rotated images. We also apply preprocessing techniques, such as rotation, zoom, and positional adjustments, to simulate real-world variations in images.

The structure of this project is as follows: Section 2 reviews related work, Section 3 describes the methodology, Section 4 presents the experimental results, Section 5 discusses the contributions, Section 6 provides conclusions, Section 7 lists the references, and Section 8 includes the appendices.

2 Related Work

Convolutional Neural Networks (CNNs) have been extensively studied for image classification tasks due to their ability to learn hierarchical feature representations effectively. The study by Kadam et al. (2020), titled "CNN Model for Image Classification on MNIST and Fashion-MNIST Dataset", explores five CNN architectures with varying convolutional layers, filter sizes, and fully connected layers, systematically evaluating hyperparameters such as activation functions, optimizers, and dropout rates. Their experiments reveal that CNN models consistently achieve over 99 percent accuracy on the MNIST dataset, while the Fashion-MNIST dataset poses greater complexity, with the best-performing architecture achieving slightly lower accuracy. This study underscores the need for careful architectural and hyperparameter selection when tackling more complex datasets. Building on their findings, our project examines the robustness of CNNs by applying various transformations to the Fashion-MNIST datasets, such as rotation, zoom, and positional adjustments, to simulate real-world variations in data.

Additionally, the work titled "Rotated Region Based CNN for Ship Detection" addresses the challenges of detecting rotated objects in remote sensing images, introducing Rotated Region-based CNN (RR-CNN). This model incorporates a Rotated Region of Interest (RRoI) pooling layer, a Rotated Bounding Box (RBB) regression model, and a multi-task approach for Non-Maximal Suppression (NMS). These innovations enable the RR-CNN to accurately extract features from rotated regions, outperforming traditional methods like Fast R-CNN on the HRSC2016 dataset. While the primary focus of RR-CNN is on object detection in high-resolution satellite imagery, its emphasis on rotation-

invariant feature extraction directly inspires our exploration of how transformations like rotation affect CNN performance on image classification tasks.

3 Methodology

3.1 Dataset Preparation

We utilized the Fashion-MNIST dataset as the benchmark for our experiments. Fashion-MNIST is composed of grayscale images of size 28×28, categorized into 10 classes. Three versions of the dataset were created:

- **Clean Dataset:** The clean dataset contains unmodified 28×28 grayscale images from the MNIST and Fashion-MNIST datasets.
- **Polluted Dataset:**The polluted dataset represents a transformed version of the original dataset. This dataset applies controlled rotations to the images using the CRRNWEP transformation function. The rotation range is set to $-30°$ to $-10°$ and $10°$ to $30°$. These rotations simulate real-world noise while ensuring that the images remain visually recognizable.

  The dataset features images with controlled rotations to introduce real-world variations and challenges. This design combines clean and rotated images to diversify the training process. This approach aims to enhance the model's ability to generalize under varying conditions.

3.2 Rotated Training

Rotated training is a way to make the training dataset more diverse. It rotates images by a specific angle to create new samples. As shown in Figure 1, our Rotated Fashion-MNIST dataset includes images rotated between -30° and +30°. This helps the model recognize objects from different angles.

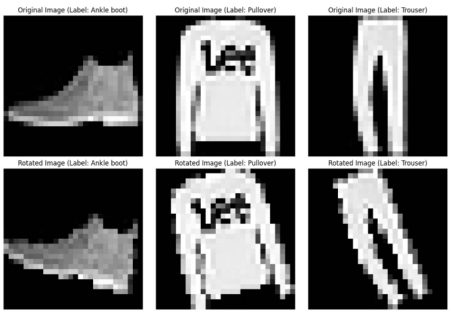

Figure 1: Rotated fashion-MNIST samples

However, rotated training has some problems. One problem is how to handle the extra space created after rotation. This space is often filled with padding,

but padding may add useless information. Another option is cropping, but cropping may cut off important parts of the image. Both options can reduce the quality of the training data.

Another problem is that rotating images too much can make them look very different from the original. This can confuse the model instead of helping it. Also, if too many rotated images are added, the model might focus too much on them and not on normal images.

To deal with these problems, we will use Oscillation Techniques. This method can improve the model without cutting off image details or adding too much extra information.

3.3 Oscillation Structure

The oscillation structure enhances image features by moving the pixels clockwise or counterclockwise. First, we divide the image into smaller blocks based on a parameter called "block size." In each round of oscillation, the pixels within each block move one position clockwise or counterclockwise. After each movement, the order of pixels within the block changes, creating a "tilted" texture in the image.

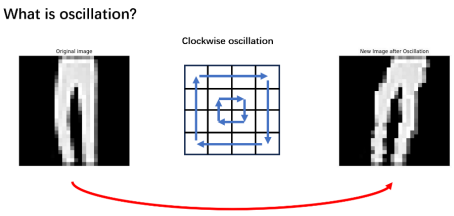

Figure 2: Clockwise oscillation step

The oscillation process involves two movement patterns shown in figure 2. The first pattern moves the pixels clockwise, and the second pattern moves them counterclockwise. These two patterns alternate. The generated images are then sent to three separate convolutional blocks for processing. Finally, the outputs from these three convolutional blocks are combined using a max-pooling layer; after the oscillation process and max-pooling, we can see in figure 3 that the resulting feature maps are passed into a CNN or ResNet structure.For CNN, the feature maps are typically sent through additional convolutional layers, followed by fully connected layers, and then to the output layer for classification. For ResNet, the feature maps are passed through residual blocks, which contain skip connections to improve the flow of gradients during backpropagation, making the model deeper and more efficient. This method introduces controlled transformations to the image. The oscillation operation does not require additional training data, allowing for more diverse input data. While the oscillation structure adds richer image features and more varied training data, it also increases the complexity and training time of the model.

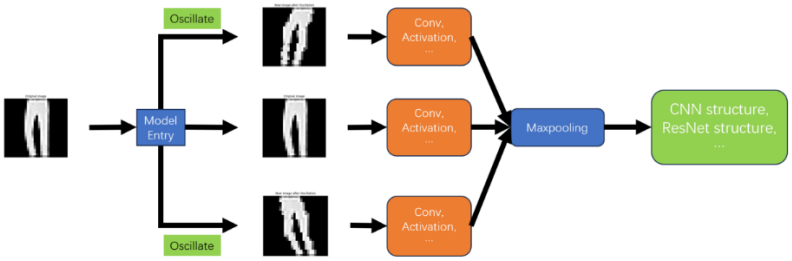

Figure 3: This figure illustrates the clockwise and counterclockwise pixel movement in blocks, creating a tilted texture. The resulting images are processed by three separate convolutional blocks. After max-pooling, the feature maps are passed into either a CNN or ResNet structure for further processing.

Our experiments found that using the oscillation structure to train the Fashion-MNIST dataset increased the training time by about 40%. Furthermore, when applied to more complex images, the effect of oscillation may weaken, leading to less significant improvements in model performance.

The oscillation structure has several advantages and disadvantages. On the positive side, it enhances the image with a tilted texture, which improves model robustness without the need for additional datasets. This is beneficial because it allows for more diverse input variations while keeping the data augmentation process efficient. On the downside, the implementation of oscillation increases the model's complexity and training time. For instance, training on the Fashion-MNIST dataset with oscillation resulted in a 40% increase in training duration. Moreover, the effectiveness of oscillation diminishes when applied to more complex images, where the transformation may not yield significant improvements.

3.4 Residual Networks

Deep Residual Networks (ResNets) were introduced to address the degradation problem that occurs when networks become very deep. In traditional deep networks, increasing the depth often leads to higher training errors, making it difficult to train the model. ResNet tackles this issue by introducing the concept of "residual learning" through the use of residual blocks and skip connections, which allow the input to be directly added to the output. This structure alleviates problems such as vanishing and exploding gradients, enabling the training of very deep networks.

Mathematically, a residual unit in ResNet can be expressed as:

$$y_l = h(x_l) + F(x_l, W_l),$$
$$x_{l+1} = f(y_l),$$

where x_l and x_{l+1} represent the input and output of the l-th unit, respectively, F is a residual function, and $h(x_l) = x_l$ is the identity mapping. The function f typically refers to a ReLU activation function [**?**]. This design allows the network to retain useful information from earlier layers, facilitating learning in deeper architectures. As a result, ResNet has proven to be highly effective for tasks such as image classification and object detection, where capturing complex image features is crucial.

By incorporating residual blocks, ResNet improves training stability and performance, even as the network depth increases. This design makes it a robust choice for modern convolutional networks.

3.5 Cross-Validation

Cross-validation is a widely used technique for assessing the performance of machine learning models, especially when working with small datasets or when there is a risk of overfitting. It involves dividing the dataset into several subsets (typically k-folds). For each iteration, one subset is used as the validation set, while the remaining subsets are used for training. The model is trained and evaluated multiple times, each time using a different validation subset. The final performance is then averaged over all iterations. The main advantage of cross-validation is that it provides a more reliable estimate of model performance by reducing dependency on a single data split, thereby minimizing the risk of overfitting.

4 Experimental Results

In this section, we evaluate different models on the Fashion-MNIST and MNIST datasets using three training strategies: clean data, rotated data, and mixed data. We also incorporate the oscillation block technique in our model to assess its impact on performance. Additionally, we apply cross-validation and utilize the ResNet architecture to improve feature extraction and evaluate model robustness across various data splits.

4.1 Result 1

The first experiment evaluated the performance of the "Pure CNN" model on the MNIST dataset. As shown in Table 1, when the model was trained on clean data, it achieved a test accuracy of 99.22%. However, when tested on rotated images, the accuracy dropped to 93.39%. This result indicates that the standard CNN model performs well on clean data but struggles with rotated data, showing that rotations in the input images impact the model's performance.

MNIST	Train on clean data
Test accuracy on clean data	99.22
Test accuracy on rotated data	93.39

Table 1: Performance of "Pure CNN" model on MNIST

4.2 Result 2

Next, we tested the "Pure CNN" model on the Fashion-MNIST dataset. As shown in Table 2, the model achieved an accuracy of 92.23% on clean data. However, when trained on rotated data, the accuracy dropped significantly to 46.0%. This highlights the challenge of training a model that can handle rotated images without the right augmentation techniques.

When we combined clean and rotated data in mixed training, the model's performance on clean data remained nearly the same at 92.56%. However, the accuracy on rotated data increased to 91.19%. This suggests that adding rotated data to the training process helps the model become more robust to rotated images without sacrificing much accuracy on clean data.

Fashion MNIST	Clean data	Rotated data	Mixed data
Acc on clean data	92.23	85.71	92.56
Acc on rotated data	**46.0**	91.24	91.19

Table 2: Performance of "Pure CNN" model on Fashion MNIST

4.3 Result 3

To address the limitations of the "Pure CNN" model, we incorporated the oscillation block technique into our model. As shown in Table 3, the "CNN with Oscillation Block" model achieved an accuracy of 90.75% on clean data. While this is slightly lower than the 92.23% accuracy of the "Pure CNN" model, the performance on rotated data improved significantly, achieving an accuracy of 51.06%, compared to the 46.0% accuracy of the "Pure CNN" model. This shows that the oscillation block technique provides a clear improvement in handling rotated images.

Additionally, the model's accuracy on mixed data increased to 89.94%, slightly higher than the 91.19% accuracy of the "Pure CNN" model on mixed data. The oscillation block adds a "tilted" texture to the images, improving the model's robustness to rotated data without the need for additional rotated datasets.

Overall, the oscillation block technique improved the accuracy of the model on rotated data by approximately 5%. While this improvement came at the cost of a slight decrease in accuracy on clean and mixed data, the overall performance of the model on rotated data was significantly enhanced.

Our model	Clean data	Rotated data	Mixed data
Acc on clean data	9 0.75	82.21	91.32
Acc on rotated data	**51.06**	89.11	89.94

Table 3: Performance of "CNN with Oscillation Block" model on Fashion MNIST

4.4 Result 4

In this experiment part, we use ResNet and cross-validation to enhance both model performance and evaluation reliability. ResNet, with its residual connections, enables the network to effectively extract features even in very deep architectures, overcoming training challenges associated with deeper networks. This improves the model's accuracy and robustness. At the same time, cross-validation ensures that the model's performance is consistent across different data splits, providing a more reliable and generalized evaluation. By combining these techniques, we ensure that the model is not only efficient in extracting features but also consistently performs well across various data subsets, reducing the risk of overfitting and ensuring robust performance.

Training Type	Test Type	Test Accuracy
Mixed	Normal	92.42%
Mixed	Polluted	90.96%
Normal	Normal	91.96%
Normal	Polluted	**48.38%**
Polluted	Normal	83.94%
Polluted	Polluted	90.98%

Table 4: Test Results of CNN with ResNet and Cross validation

Training Type	Test Type	Test Accuracy
Mixed	Normal	88.50%
Mixed	Polluted	85.85%
Normal	Normal	87.49%
Normal	Polluted	**51.41%**
Polluted	Normal	67.42%
Polluted	Polluted	86.65%

Table 5: Test Results of RDEnCNN ResNet and Cross validation

As shown in Table 4 and Table 5, we can see that when using only the original (clean) data for training, the CNN with ResNet model achieved an accuracy of 48.38% on the Polluted (rotated) test set, while the RDEnCNN ResNet model attained an accuracy of 51.41%. This difference indicates that the RDEnCNN ResNet model is better at handling polluted (rotated) data when trained on only the original dataset.

In other scenarios, such as training on Mixed data and testing on Normal and Polluted test sets, the CNN with ResNet model generally outperforms the RDEnCNN ResNet model. For example, on the Normal test set, the CNN with ResNet model achieved an accuracy of 92.42% with Mixed training data, compared to 88.50% for the RDEnCNN ResNet model. Similarly, on the Polluted test set, their accuracies were 90.96% and 85.85%, respectively.

Our RDEnCNN ResNet model improves accuracy on the polluted (rotated) test set when only the original training data is used, with an accuracy increase from 48.38% to 51.41%. However, its performance on other data conditions (such as Mixed data training and Normal test sets) is generally lower compared to the CNN with ResNet model. Thus, while our model shows advantages in handling rotated data, it may perform worse in other scenarios.

5 Conclusion

In this work, we explored the impact of various techniques on CNN performance, focusing on the oscillation block. The oscillation block, added to the model's structure, effectively captured the "tilted" features in rotated images without requiring additional training data. This resulted in improved performance on the polluted (rotated) test set. We also applied preprocessing techniques to enhance input data quality and used data augmentation to increase the size and variability of the training dataset. These methods, combined with the oscillation block, significantly improved the model's ability to handle rotated data. However, the model's performance on other types of data, such as mixed training or normal test sets, showed a slight decline compared to the baseline CNN with the ResNet model. Overall, using the oscillation block and preprocessing and augmentation techniques led to better performance in specific scenarios, particularly with rotated data.

6 Contributions

- **Hao Luo**: Responsible for the implementation of the project and the development of the code.
- **Haiyi Wang**: Responsible for writing the final project report.
- **Manav Patel**: Improved the code and added results for training with ResNet and cross-validation after the final presentation.
- **Christina Zeng**: Responsible for creating the final presentation slides.

Hao Luo	501305093	hao.luo@torontomu.ca
Haiyi Wang	501293742	haiyi1.wang@torontomu.ca
Manav Patel	500967756	manav1.patel@torontomu.ca
Christina Zeng	501335842	christina.zeng@torontomu.ca

Table 6: Group Members Information

7 References

Kadam, Shivam S., Amol C. Adamuthe, and Ashwini B. Patil. "CNN model for image classification on MNIST and fashion-MNIST dataset." Journal of scientific research 64.2 (2020): 374-384.

Wang, Chen, and Yang Xi. "Convolutional neural network for image classification." Johns Hopkins University Baltimore, MD 21218 (1997): 425.

Albawi, S., Mohammed, T. A., Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In 2017 International Conference on Engineering and Technology (ICET) (pp. 1-6). IEEE.

Liu, Weibo, et al. "A survey of deep neural network architectures and their applications." Neurocomputing 234 (2017): 11-26.

Z. Liu, J. Hu, L. Weng and Y. Yang, "Rotated region based CNN for ship detection," 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 2017, pp. 900-904

LeCun, Y., Cortes, C., Burges, C. J. (2010). MNIST handwritten digit database. ATT Labs. Available: http://yann.lecun.com/exdb/mnist/

Xiao, H., Rasul, K., Vollgraf, R. (2017). Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv preprint arXiv:1708.07747.

He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.