Applying and analyzing the performance of wavelet transforms in conjunction with a convolutional neural network for image classification

Xiang Ma, Andrew Curtis, Justin Whitaker

December 17, 2022

1 Abstract

Image classification is a popular machine-learning task used for a wide range of purposes. Utilizing neural networks is now a common and effective approach for this task. Many small improvements to methods and neural network architectures are being tested within the field. One such type of improvement is using transforms to augment images and remap the features to a different domain. The neural network can then learn from this new representation which can help to improve performance. Some transforms are already being used in state-of-the-art pre-trained image classification architectures like ResNet50. One transform type that has not yet been thoroughly tested or utilized is wavelet transforms. These are similar to commonly-used Fourier transforms but have the ability to capture more localized wave oscillation information. In this research, we conducted empirical testing of the use of wavelet transforms with a convolutional neural network to see if image classification can be improved over a simple CNN baseline. The results indicate that wavelet transforms speed up initial learning but achieve similar or slightly worse final accuracy. The wavelet transform appears to have the potential for faster training, creating smaller networks, and preventing over-fitting.

2 Introduction

Image classification is a popular machine-learning task that has a wide variety of applications. In previous years, state-of-the-art approaches were to use a variety of machine learning algorithms in order to classify the images. These were then combined in ensemble methods in order to produce the best results. However, in recent years, neural networks have seen great success in beating these previous methods. Specifically, the development of convolutional neural networks (CNNs), which learn new feature mappings of the images, are able to improve the speed of learning and prediction accuracy. Research on various methods applied to CNNs has further improved performance, such as adjusting network architecture and applying transforms to the data. For our research, we chose to focus on testing the performance of a transform not yet commonly utilized, the wavelet transform. The potential of wavelet transforms is to improve training times, create smaller networks, improve performance and avoid over-fitting.

2.1 Convolutional Neural Networks

A convolutional neural network (CNN) is a type of neural network that uses a mathematical process called convolution as a layer of the network. A convolutional step scans over sections of pixels called a kernel and remaps the features into a new representation called a feature map. This feature map is then fed as the input to the next layer. This process can help the neural network learn new local representations of the data. After several convolutional layers, the input is passed through a set of fully-connected neural network layers. A typical CNN architecture is displayed below,

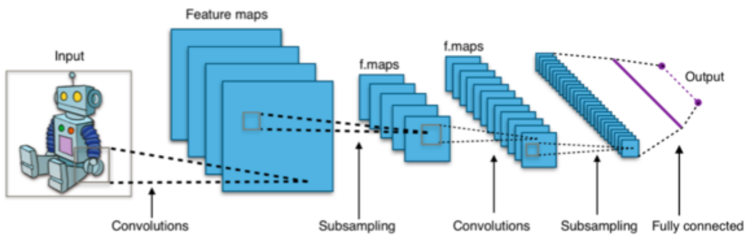

Figure 1: Typical CNN Architecture

2.2 Wavelet Transform

Transformations are commonly used to augment data. These transforms can serve a variety of purposes. For image processing, the transforms convert the image from one domain to another. This allows for the learning of a new set of features from the image. Some transforms are already being used in state-of-the-art approaches, an example is the Fourier transform, Radon transform, and Discrete Cosine transform all being utilized in the pre-trained ResNet50 network. The Fourier transform decomposes a function into a simple combination of sine waves. However, a disadvantage of Fourier transforms is that they only capture global frequency information. Wavelets can similarly capture wave-like oscillations but have scale and location components. This allows for capturing time-sensitive oscillation information.

There are a number of common wavelet types that can be used for transformation. The simplest is the Haar transform. This wavelet appears almost as a square wave. Other, more complex wavelets are also available for testing. Some example wavelet types are shown below.

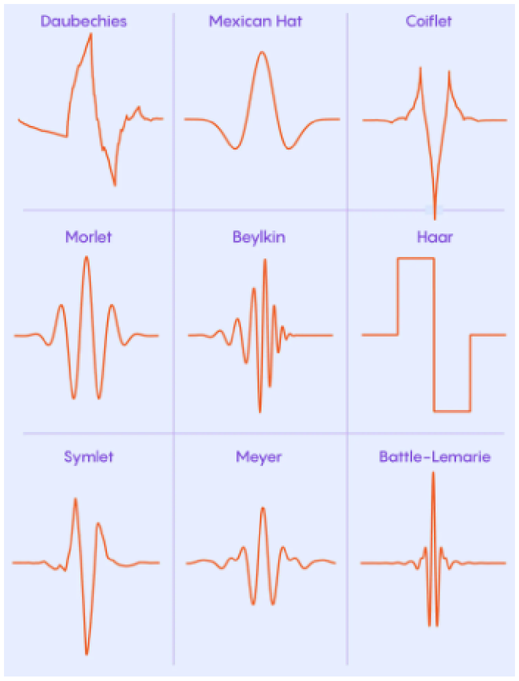

Figure 2: Example Wavelets

2.3 2D Discrete Wavelet Transform

The wavelet transforms we used here to process the image data are two-dimensional discrete wavelet transforms. The diagram is pretty clear, multiple low-pass(LP)/high-pass(HP) filters and downsamplers are used. It is shown in Fig. 3. Mathematically, the filtering operation is equivalent to convolution in the time/spatial domain. The formulation of 2D convolution is given by

$$y[i,j] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h[m,n] \cdot x[i-m, j-n] \tag{1}$$

$h[m,n]$ is the filter. And the wavelet filters mentioned above can be used to construct the LPF and HPF.

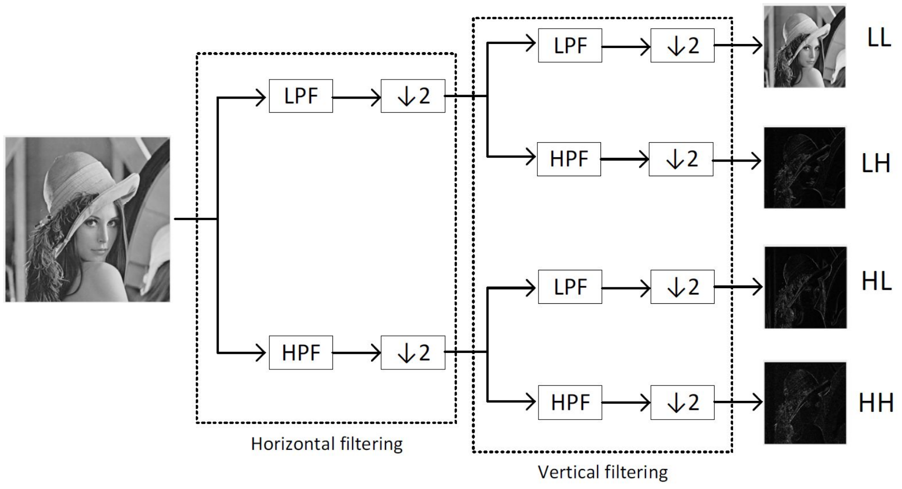

Figure 3: 2D Wavelet Transform Diagram

After decomposition, the images are reconstructed to get the original size in the following way

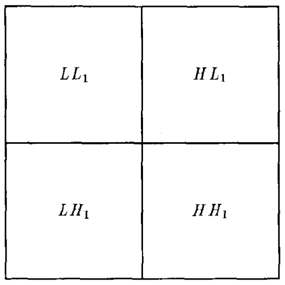

Figure 4: 2D Wavelet Transform Reconstruction

that is, the low-resolution part is located in the top left, while the high-resolution parts are located in the horizontal, vertical, and diagonal parts.

3 Data

Several datasets were chosen to be tested. These included some industry-standard datasets for image classification as well as some more complex image sets. The simpler datasets are useful for conducting empirical testing as well as comparing to other baseline machine learning algorithms.

3.1 Showcased Datasets

There are two main datasets for which results are showcased. The first is the Kuzushiji-MNIST (KMNIST) dataset which is a drop-in replacement for the original MNIST dataset (28x28, grayscale, with 10 output classes). Some sample images are shown below This dataset is a "nice" dataset in that it is grayscale, and in

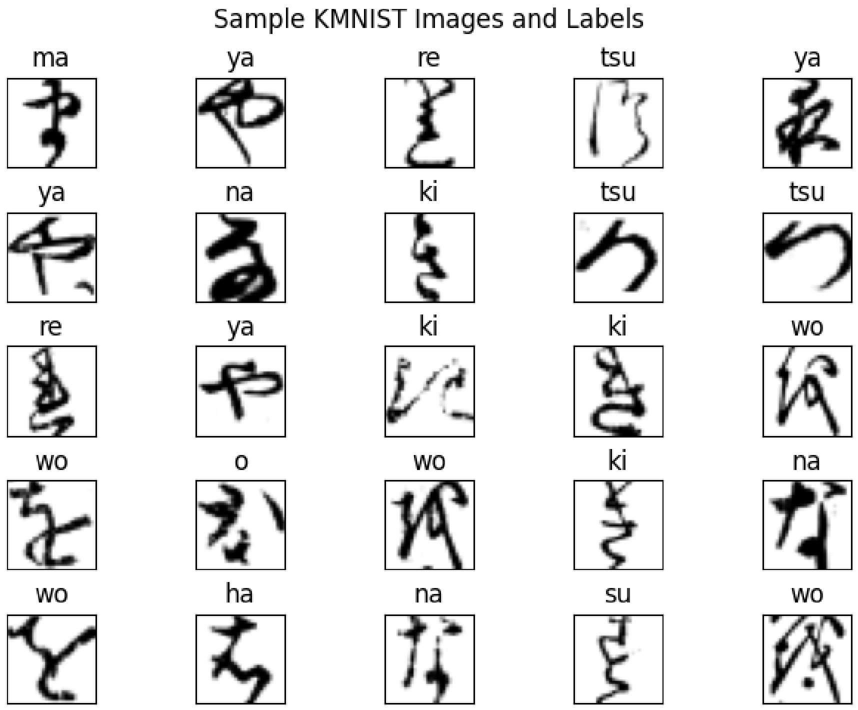

Figure 5: Sample images from the KMNIST dataset with their corresponding labels.

the same format as the well-known MNIST dataset, but is a little more nuanced than the original MNIST dataset.

The second main dataset we have used is a butterfly classification dataset from Kaggle. These images are color and 224x224 with 100 output classes. Some samples are shown in Fig. 6.

This dataset presents a more challenging problem by having many more classes to identify, larger color images, and the images of the butterflies are from many different angles and perspectives, although they are all centered pretty well on the butterfly itself. It also is more difficult as there are fewer images (13,000 training samples) on top of more classes.

4 Methods

Currently, the diagram for the image classification task is

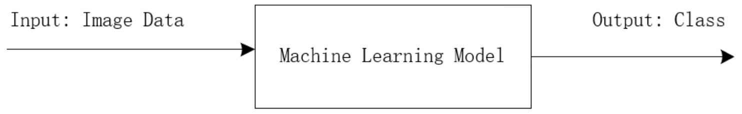

Figure 7: Diagram for Image Classification

Our proposed scheme is to apply wavelet transforms before feeding the image data to the machine learning models. That is

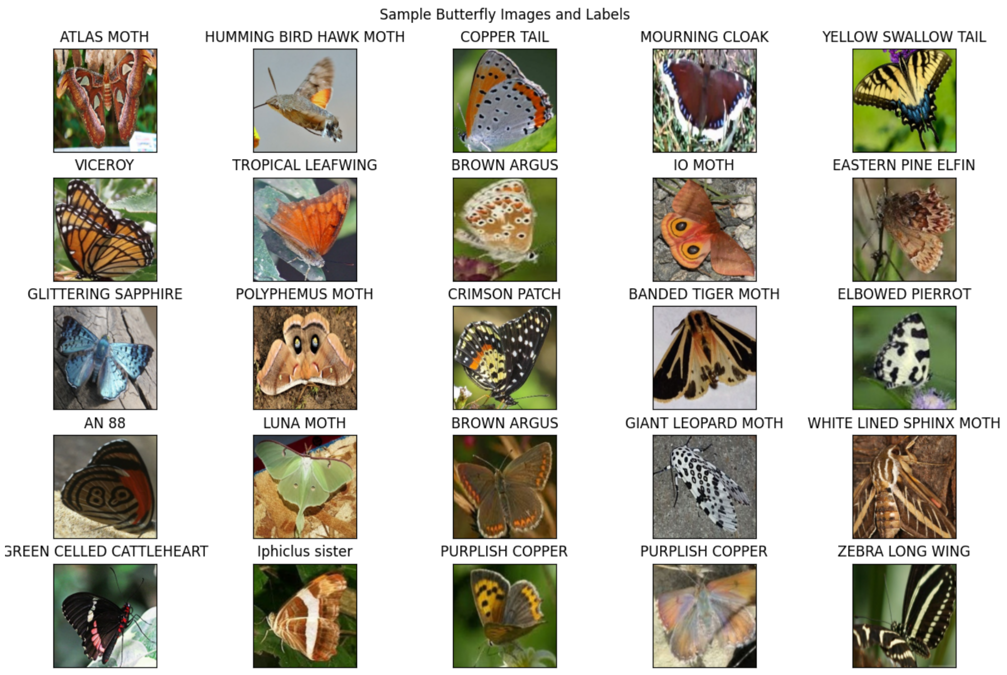

Figure 6: Sample images from the butterfly dataset and their corresponding labels

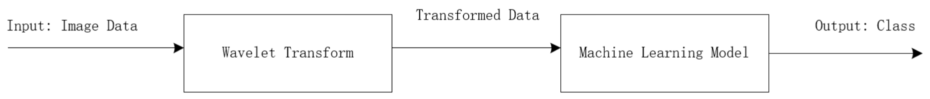

Figure 8: Diagram for Our Proposed Image Classification Scheme

So, the wavelet-transformed data may have more features than the unprocessed image data. The transformed data is shown below

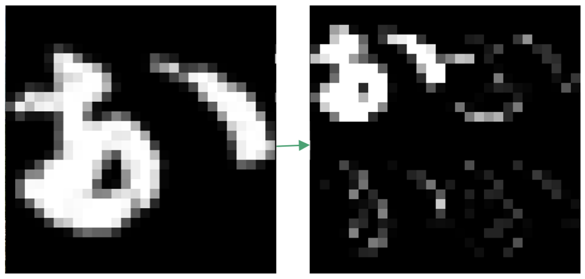

Figure 9: Sample KMNIST Image after Wavelet Transform

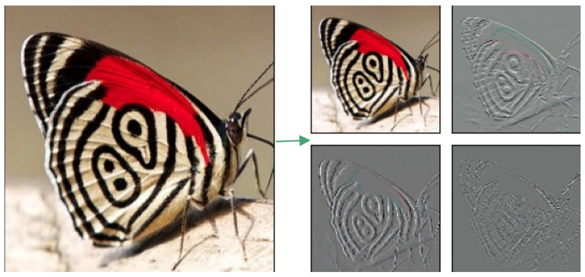

Figure 10: Sample Butterfly Image after Wavelet Transform

Whether the image is gray or color, the low-resolution part of the processed image still keeps the detail of the unprocessed image. While the high-resolution parts get the texture of the object. And they can be regarded as extra features than unprocessed images. This might be the reason the ML model learns fast.

4.1 ML Model on KMNIST

The machine learning model we used on KMNIST is a convolutional neural network (CNN). It includes 5 convolutional layers followed by max-pooling players. And then 4 linear fully connected layers with a softmax layer at the end.

To prove the effectiveness of the wavelet transform, we first use the simplest Haar wavelet filter Haar (db1). And then other wavelet filters are also used such as db2, db3, db4, sym2, sym3, sym4, and bior/rbio series.

Also, the multi-level wavelet decomposition can extract different resolutions of the image to different areas. Here, we use a second-level decomposition to see the potential improvements.

Above all scenarios, the image size after wavelet transform does not change, so the ML model input size does not change. Since we use the same model on the unprocessed image data and wavelet-transformed data, the number of model parameters does not change. So the result is comparable.

4.2 ML Model on Butterfly

Concatenating the resulting filters from the wavelet transform into one image of the same size as the original image is nice for comparisons as the exact same network architecture can be used on the original image and the transformed image. This, however, isn't the most natural way to include this transform in a convolutional network architecture. It is more natural to treat the output of the wavelet transform more like the output of a convolutional layer, as a set of channels or filters with a new size. Treating the transform in this way means that a typical three-channel image would be treated as a set of twelve filter channels after the transform (4 transform filters per channel).

In order to maintain some level of comparable expressibility, the corresponding conventional CNN could have a starting convolutional layer designed to mimic the number of filters and resulting filter sizes of the wavelet transform. In other words, the wavelet transform output for the $3 \times 224 \times 224$ butterfly images is $12 \times 112 \times 112$, and so a convolutional layer (with max pooling of 2) can be used to get an output of the same size. In reality, as we did not use padding with the convolution, the output of the first layer was actually $12 \times 110 \times 110$. This could easily be fixed with some padding in the first convolution but didn't seem to hinder the conventional CNN's performance compared to the wavelet CNN, so this was left for future work to investigate. The full architectures of the two models can be seen in Fig. 11.

No preprocessing (other than the wavelet transform) was performed on the data. Two experiments were run. In both the optimizer used was the Adam optimizer, but with a learning rate of 0.001 and no regularization in the first and a learning rate of 0.0001 and l2 regularization with $\lambda = 0.0001$ in the second. These parameters were the same for both networks in both experiments. Both networks were trained for 50 epochs in both experiments as well.

The primary CNN results on the Butterfly dataset are also compared to a baseline k-nearest-neighbors (KNN) approach. Models with 5, 10, and 20 neighbors are used for comparison.

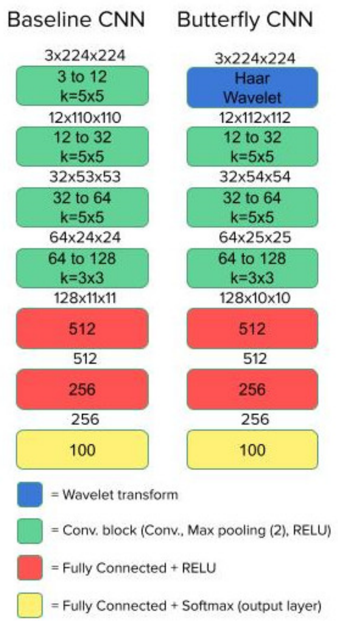

Figure 11: The architectures of the two CNNs used for the Butterfly dataset.

5 Results

5.1 KMNIST

The KMNIST dataset was used for most of our empirical testing of baseline machine learning algorithm methods. In table 2. below, the test accuracy of several tuned machine learning algorithms are displayed as well as the performance of our base CNN architecture. Even the base CNN architecture was enough to significantly outperform all of the tuned machine learning algorithms. This is evidence of the effectiveness of convolutional neural networks on the image classification task. Going forward on other datasets, we mostly selected the CNN architecture without transforms to be the baseline model for comparison in testing.

Method	Accuracy
AdaBoost Classifier	52.2%
K-Nearest Neighbors	91.4%
Support Vector Machines "RBF"	92.9%
Random Forest	86.2%
Base CNN	96.0%

Table 1: Test accuracy of machine learning algorithms on KMNIST dataset.

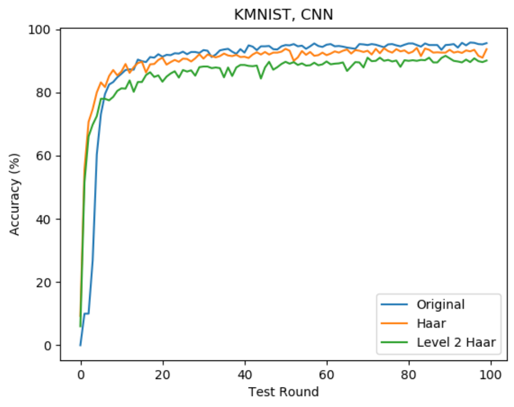

Figure 12: First and Second Level Haar Wavelet Transform

Haar wavelet transform does help to improve the learning speed at the initial learning period. But after some point, the test accuracy is not as good as the CNN directly applied to the unprocessed image. The second-level wavelet decomposition gets an even worse result. One of the reasons might be the image size of KMNIST is 28×28, which is not sufficiently large.

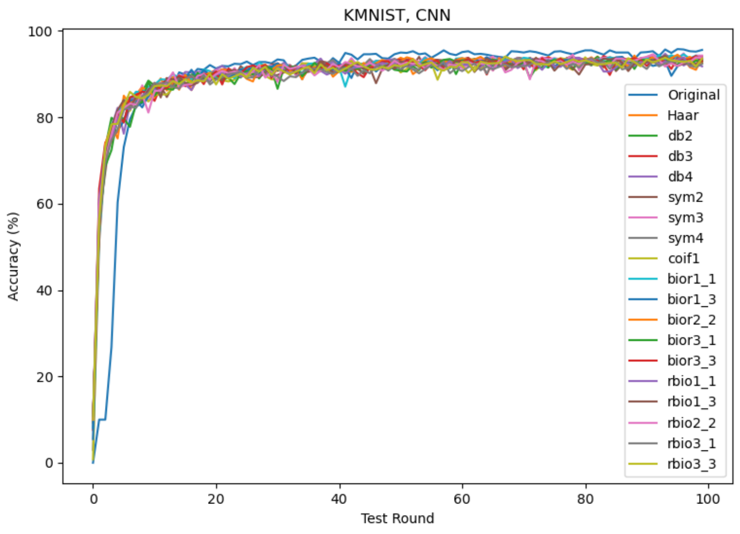

Figure 13: Different Wavelet Transform

For different wavelet filters applied, they all achieve similar results as the Haar wavelet transform. The wavelet transforms improve the initial learning speed significantly more than the unprocessed image.

The reason why the model learns first at the initial period but not achieves higher accuracy at the end might be the new representation of the image after the wavelet transform. Wavelet transform extracts different resolutions of the image and put the low-resolution image in the top left corner and the high-resolution image in other areas. It is like creating/extracting new features for the ML model to learn. And the ML model does not have to run multiple times to learn the hidden features. The reason why the wavelet-transformed image achieves lower final test accuracy than the unprocessed data might be that the resolution of KMNIST is too low. KMNIST is 28×28, after wavelet transform, the approximation part in the top left corner is 14×14, which loses the detail of the image. So we also use the high-resolution Butterfly dataset.

5.2 Butterflies

To start off with a baseline to gauge general performance, we used a KNN approach with 5, 10, and 20 neighbors. The resulting accuracies can be seen in Table 5.2 where all three variations performed with around 17% accuracy. While this leaves plenty of room for improvement on the table, it is worth noting that random guessing would result in 1% accuracy, so this is a significant improvement over that.

N-neighbors	Accuracy
5	17.8%
10	17.2%
20	16.8%

Table 2: The accuracy of a KNN with different numbersof neighbors on the Butterfly dataset.

Continuing on with the Butterfly dataset, initial CNN results were similar to those with the KMNIST data. At first, the only experiment that was run was with a learning rate of 0.001 with no regularization. The results of this experiment are shown in Fig. 14. In this experiment, the wavelet transform network was able to speed up initial training but was unable to provide long-term accuracy improvements.

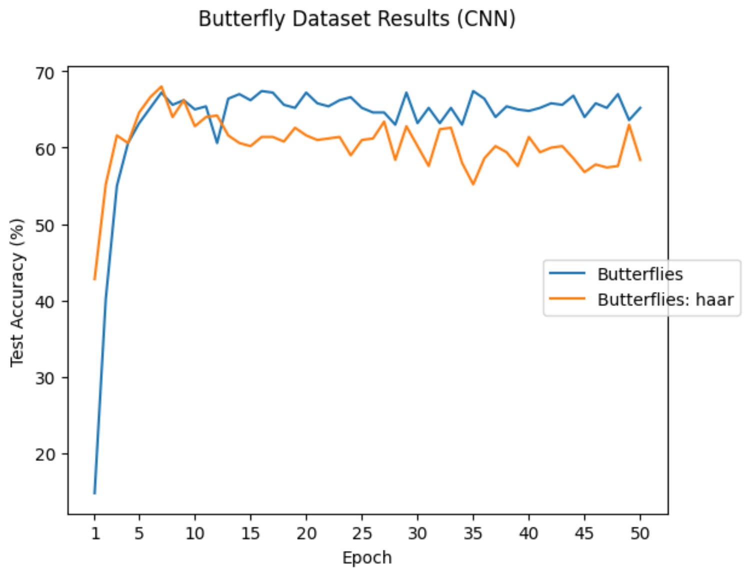

Figure 14: The initial results with the Butterfly dataset, with the Adam optimizer with a learning rate of 0.001 and no regularization

As we moved to explore more types of wavelet transforms, additional hyperparameters were explored to ensure that learning would occur in all cases, no matter which wavelet was used. This resulted in the selection of a learning rate of 0.0001 and the addition of l2 regularization with $\lambda = 0.0001$. These results are shown in Fig. 15, and seem to favor the wavelet transforms in this case. In this case, the wavelets were able to exhibit both faster learning and better long-term performance. While this is exciting, it prompts more questions than answers. In the other datasets that were tested, this was not the case, so why was better overall performance the result here? Does the additional difficulty of the underlying task hinder the non-wavelet-based CNN? Or would a more thorough hyperparameter search yield similar benefits with the other datasets? Does this dataset just happen to lend itself well to wavelet filters? These are all directions of exploration that future work could take. One possible explanation is that providing the network with pre-filtered data encourages the learning to pre-maturely learn to rely too heavily on the "low-hanging fruit" from those initial filters. Then, with a lower learning rate, and some regularization, it has more time to develop more problem-specific filters, more in line with what a typical CNN might learn.

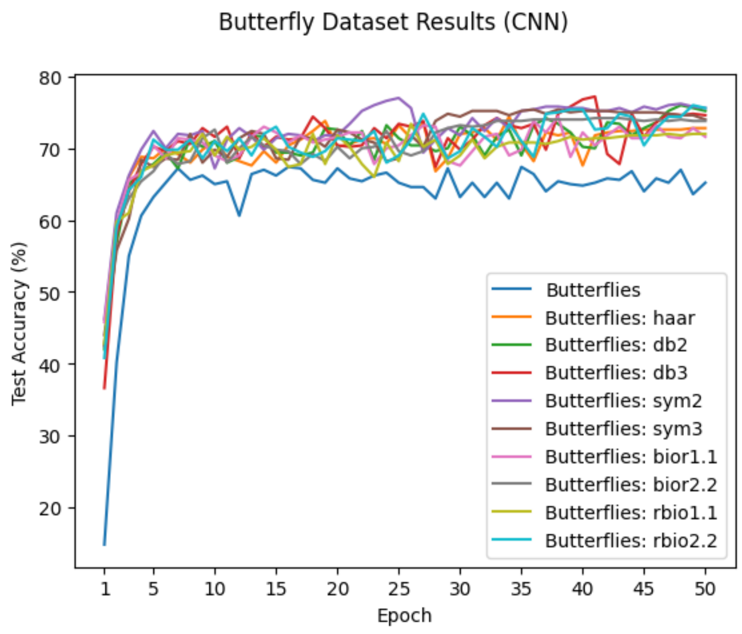

Figure 15: The results with the Butterfly dataset with several wavelet transforms, using the Adam optimizer with a learning rate of 0.0001 and l2 regularization with $\lambda = 0.0001$.

Additionally, as can be seen in Fig. 15, while there is some variation in the performance of the various wavelets, the overall trends and performance levels are quite similar. While the resulting overall performance uplift is beneficial in this case, as demonstrated by the other datasets, it may not always be a benefit of using wavelet transforms. The significant improvements in initial learning speed, however, are quite consistent across the datasets we tested; these speedups could be quite useful in cases where restrictions (such as dataset or model size, resource limitations, etc.) limit the amount of time available to train.

It is worth noting that there are many potential avenues of exploration to attempt to capitalize on the improved initial learning while maintaining long-term performance in cases where the wavelet transform hinders overall performance. One possibility is using the wavelet transform filters side by side with normal neural network filters on the original image. There are also related transforms, called scattering transforms, that have been used as neural network layers (or augmentations to layers) in the past (see https://www.kymat.io/index.html).

6 Conclusions

Image classification has seen significant improvements going from older machine learning algorithms to more recent convolutional neural networks. Further improvements to convolutional neural networks include more complicated architectures and applying data transforms. However, changes like larger and more complicated network architectures can slow down training times significantly. In this work, empirical testing was conducted to test the effects of wavelet transforms on data fed into a CNN. A variety of datasets, from simpler to more complex, were tested and several wavelet types were also investigated. Faster learning and a reduction in training times were observed. These transforms also offer smaller, more dense data feature mapping and networks. This can offer a significant improvement in compute resource management. There is also potential for use in de-noising and avoiding over-fitting. However, the wavelet transform application typically did not appear to improve the accuracy of the image classification overall (with some notable exceptions). Future work could include using wavelet transforms in an ensemble with other transforms to improve performance or designing more creative architectures around injecting wavelet transforms into the model. The potential reduction in training times on large datasets may also be significant for researchers with a mismatch between available computing power and training task complexity.

7 Contributions

1. Justin Whitaker: I contributed all of the results with the Butterfly dataset. This includes all preliminary work done on the data itself (not much). Any results obtained on the Butterfly dataset were run and compiled into figures/tables/data by me.

2. Xiang Ma: I initialized the idea of wavelet transform before feeding the data to the ML model. I verified the effectiveness of the idea on the MNIST, EMNIST, and FashionMNIST datasets. The KMNIST results after the wavelet transform are run by me.

3. Andrew Curtis: I contributed the results of the baseline machine learning algorithms and performed other validation. I also contributed to the presentation of results along with the other members.