MSDS 458, Section 55, Assignment 2

Alan Kessler

Deep Neural Networks for Space and Time

In this analysis, the goal is to analyze a series of simple convolutional neural networks to understand the impact that the various layers have on model performance and interpretation. The analysis starts with an existing framework, examining performance and interpretation as layers are removed. Additionally, the analysis demonstrates the capability of Shapley Additive Explanations (SHAP) implemented in Python. Ultimately, it is the interpretation of the convolutional neural network (CNN) structure and interpreting model results that is more important in this analysis compared to performance itself.

The data used in the analysis is the MNIST database of handwritten digits. The database consists of single digits displayed in grayscale. There are considerable differences between this data and the prior set of alphabet data. The greatest difference is that there are ten classes of output with sixty thousand training observations and ten thousand test observations. The difference being that there are multiple representations of each digit. At a high level, this similar to the perturbed alphabet data but greater importance is given to the model's ability to generalize to new data. Figure 1 shows an example of a single digit. The data is also of a greater resolution at 28 by 28 pixels.

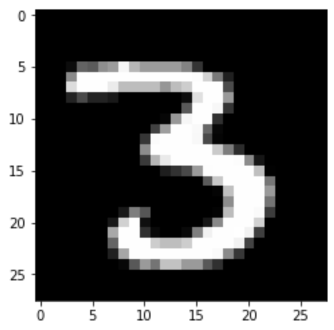

Figure 1 – Example Handwritten Digit

The initial framework is made up for two 3x3 convolutional layers with 32 and 64 features respectively. After these layers a 2x2 max-pooling layer is added, followed by a 0.25 probability dropout layer, a fully connected hidden layer with 128 nodes, and another 0.5 probability dropout layer. The last fully connected layer generates softmax class outputs. The first model uses all of the layers. The next model removes the dropout layers. The third model removes the second convolutional layer. The final model removes 50% of the nodes from the fully connected hidden layer. All of the models are trained using 12 epochs with 128 images per batch. Each model is compared based on the time it takes to evaluate an epoch, the performance on the testing data, and the value of the SHAP interpretations.

It is important for the data scientist to understand the impact and function of the different layers in a deep learning application. While the addition of multiple layers makes that interpretation more difficult, it is still critical to understand the elements of the input data the model is responding to when making a prediction. Shapley Additive Explanations (SHAP) is a method by which to tackle the credit assignment problem for any type of machine learning model. This approach combines the concept of Shapley regression values for feature importance but implements it in a way where the conditional expectation functions are from the original model. This results in a consistent method by which to attribute what aspect of the data or parts of an image result in a given classification. For this data, it can display the parts of the image that contribute to the correct class but also the parts that might indicate that the digit should be a part of a different class (Lundberg & Lee, 2017).

It is possible to visualize the output from each of the convolutional layers as shown in Figures 2 and 3. The features tend to activate in response to the various edges of the digits. For the second layer of features, specific edges activate them which intuitively would lead to better segmentation between classes.

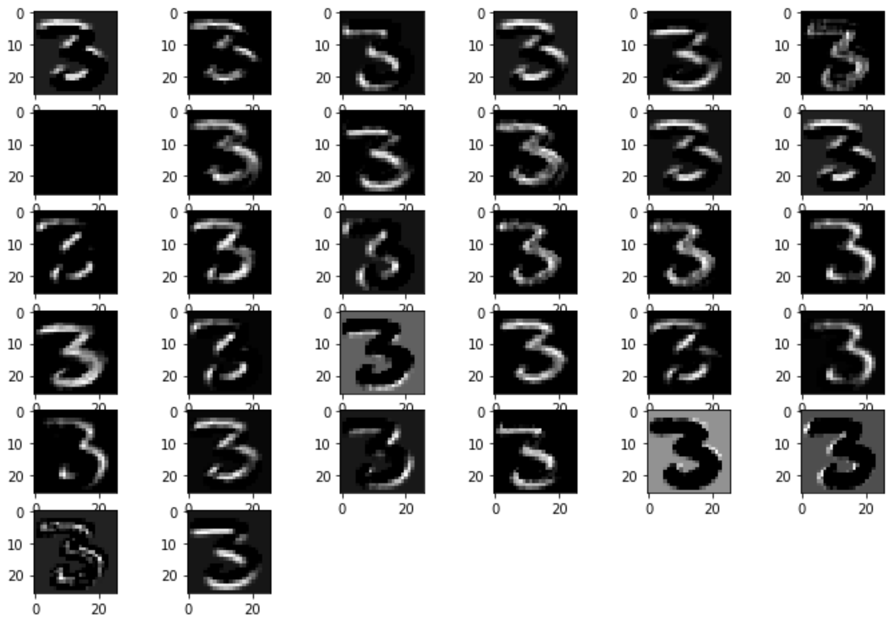

Figure 2 – Full Model First Convolutional Layer Example

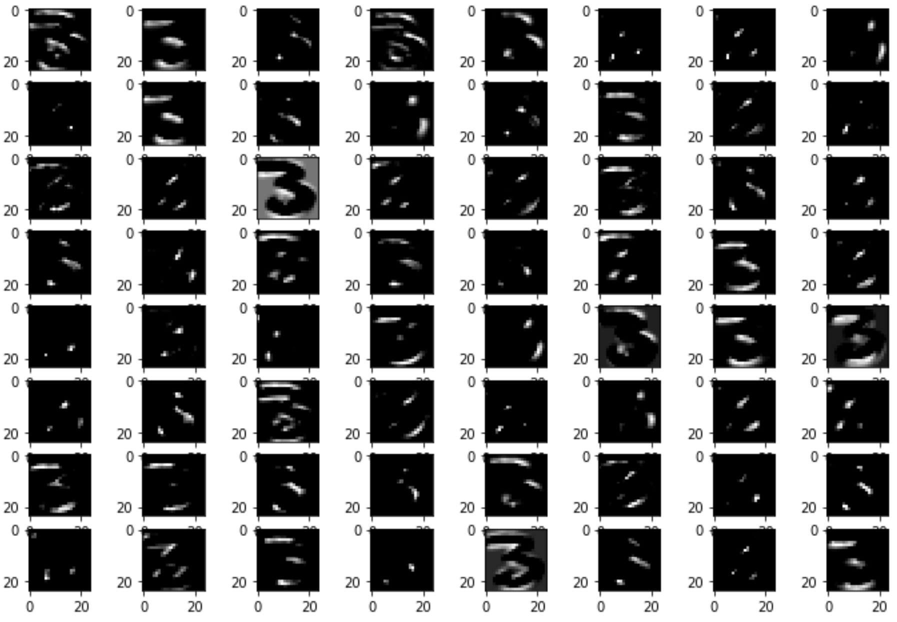

Figure 3 – Full Model Second Convolutional Layer Example

Figure 4 shows a comparison of performance between the four different models. The accuracy for all four is fairly high due to the simplicity of the data but worth observing the relative differences. The presence of the dropout layers, used to prevent overfitting, do in fact improve accuracy with the addition of some computational time. The biggest difference in accuracy and time is the result of removing the second convolutional layer as well. It's clear that having multiple convolutional layers is beneficial but comes at a cost in terms of computation time. Reducing the number of hidden nodes also reduces accuracy but had a minor impact on timing.

Model	µs per Step	Test Accuracy
Full Model	135	99.23%
Without Dropout	125	99.17%
Without Dropout/Second Conv	55	98.74%
Without Dropout/Second Conv/50% Hidden Nodes	52	98.66%

Figure 4 – Performance Comparison

The next point of comparison is between the Shap values for the full model compared to the most reduced version of the model. Figures 5 and 6 display these values across all of the classes for four examples from the testing data. The leftmost column shows the input data, and the following columns show the activation for each class. For example in the first row, the digit is a "9" and in the column representing the digit "4", the part of the "9" that closes the loop on top has a strong negative value indicating that that part of the input made that class unlikely to be the maximum. Given the high levels of accuracy for the two models, the interpretations are not all that different. Figure 6 may show a slightly lower resolution feature attribution but not significantly.

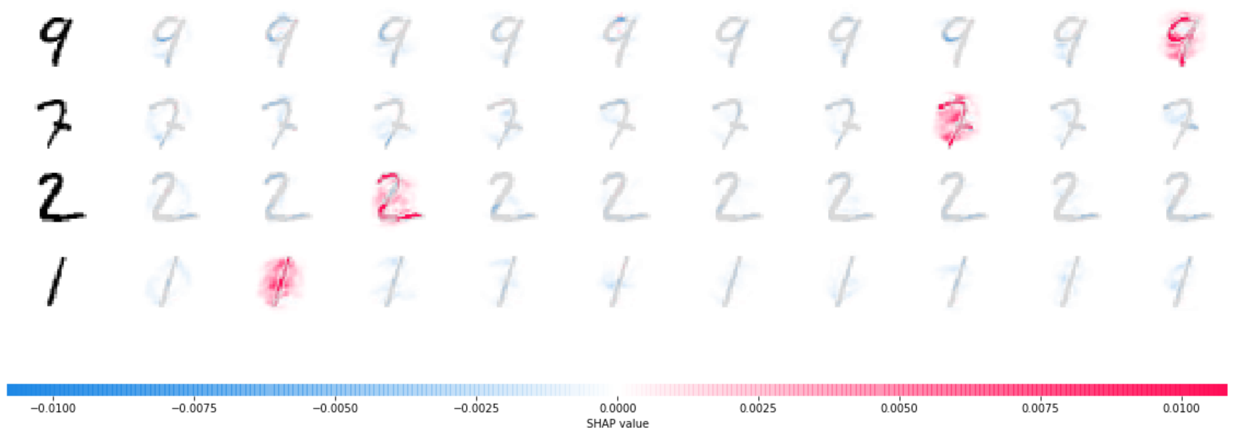

Figure 5 – Full Model Shap Values

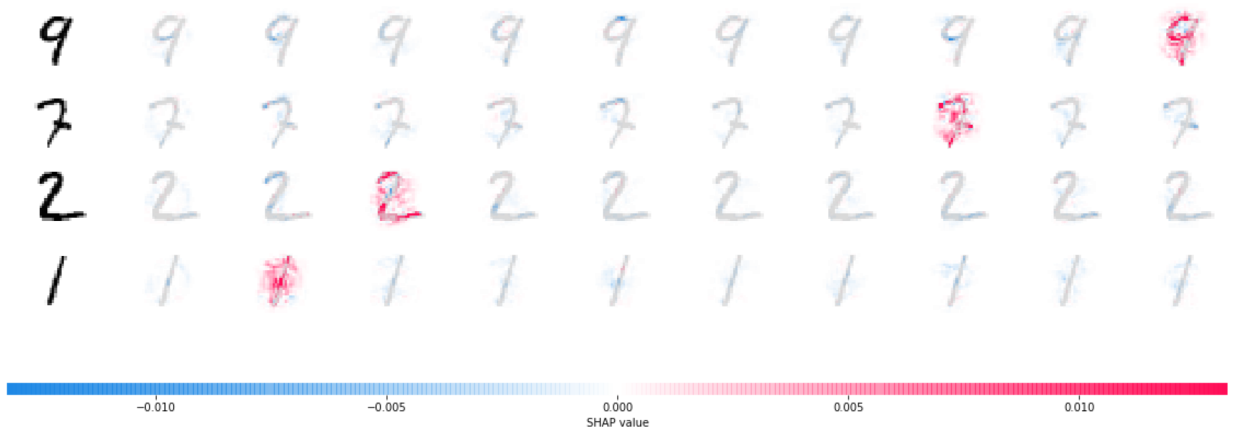

Figure 6 - Fully Reduced Model Shap Values

From a programming standpoint, the model is entirely developed using Keras using the Tensorflow backend. The initial framework is from examples provided by Keras (2018). Numpy and matplotlib modules are used to visuzalize the convolutional layer output. These visualizations are based on an example by Katariva (2017). The modeling is considerably more efficient when incorporating GPUs. The script used in the analysis is executed on a p2.xlarge AWS instance that includes a GPU. The SHAP module is used to produce the Shapley values for model interpretation.

After fitting a series of convolutional neural networks to the MNIST data, it is clear that incorporating convolutional layers adds computation time but also significantly increases accuracy. Additionally, Shap values can help indicate why a given prediction was made by a neural network, but for two similar neural networks, the difference in output may not be large.

References

Katariya, Y. (2017, March 18). Visualizing Filters Python3 Theano Backend. Retrieved October 28, 2018, from https://github.com/yashk2810/Visualization-of-Convolutional-Layers/blob/master/Visualizing Filters Python3 Theano Backend.ipynb

Keras. (2018, February 22). Mnist_cnn.py. Retrieved October 28, 2018, from https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems* (pp. 4765-4774).