Project 1 Technical Report

Tyler C. Johnston

Utah State University

CS 5600

Abstract

The purpose of this project was to test various machine learning architectures for different datasets. Such architectures included Artificial Neural Networks (ANNs), and Convolutional Networks (ConvNets), Long Short-Term Memory (LSTM). The first objective was to test these models three models over time series data. The secondary objective was to test ResNet50 and YOLO (two ConvNet models) over image classification data. Performing these tests provided an understanding of the capabilities and limitations of these architectures. Additionally, each objective utilized different programming frameworks (Keras for the time series predictions, and PyTorch for the image classification) to allow exposure to multiple machine learning libraries.

Part 1: Time Series Predictions

The data used to create the time series predictions for the ANN, ConvNet, and LSTM models were collected on bee hives at USDA research apiary in Tucson, AZ in 2022. The data contained time-aligned temperature and weights, and these models were designed to predict the weight of "hive 2059" from the temperature for the month of June and a time-period of 3.

Model 1: ANN

Dozens of ANN architectures were tested, trained, and re-trained with varying levels of dense layers and neurons, with most of them having a consistent MSE of around .8-1.6. The "relu" activation always resulted in better results, similarly with the "adam" optimizer. Thus, the best model found contained the input layer shaped to match the number of given steps and features and two hidden dense layers with 4 and 6 neurons respectively (with both the "relu" activation and "adam" optimizer). Of course, the dense output layer was created to match the number of features used. The loss hovered around .75 for each of the 2000 epochs and the MSE was 1.05, which was slightly higher than some previously discovered results. However, the trend of the graph was very consistent albeit being slightly offset. Figure 1 illustrates the best model configuration for predicting weights using ANNs.

Model 2: ConvNet

Dozens of ConvNet architectures were testing, trained, and re-trained with varying filter numbers, kernel-sizes, pool-sizes, and dense layers. Like model 1, the "relu" activation and "adam" optimizer nearly always resulted in better results. Thus, the best model found began with convolutional layer having a filter size of 4, a kernel-size of 2, and a max pooling layer with a pool-size of 2 with both the "relu" activation and "adam" optimizer. Next, the model was flattened with two dense layers being utilized: the first contained 4 neurons and the second contained 1 neuron. The loss hovered around .75 for each of the 2000 epochs and the MSE was 1.00, which was slighly higher than some previously

discovered results. However, the trend of the graph was very consistent albeit being slightly offset. Figure 2 illustrates the best model configuration for predicting weights using ConvNets.

Model 3: LSTM

Many LSTM architectures were tested, trained, and re-trained with a varying number of neurons on the dense layer. Again, both the “relu” activation and “adam” optimizer produced the best results. The final model architecture was simple: it consisted of the input layer shaped to match the number of given steps and features and one dense layer with 23 units. Of course, the dense output layer was created to match the number of features used. The loss hovered around .68 for each of the 2000 epochs and the MSE was .99. While this may be considered high, it had a consistent, albeit slightly offset, graph. Figure 3 illustrates the best model configuring for predicting weights using LSTM.

Figure 1.

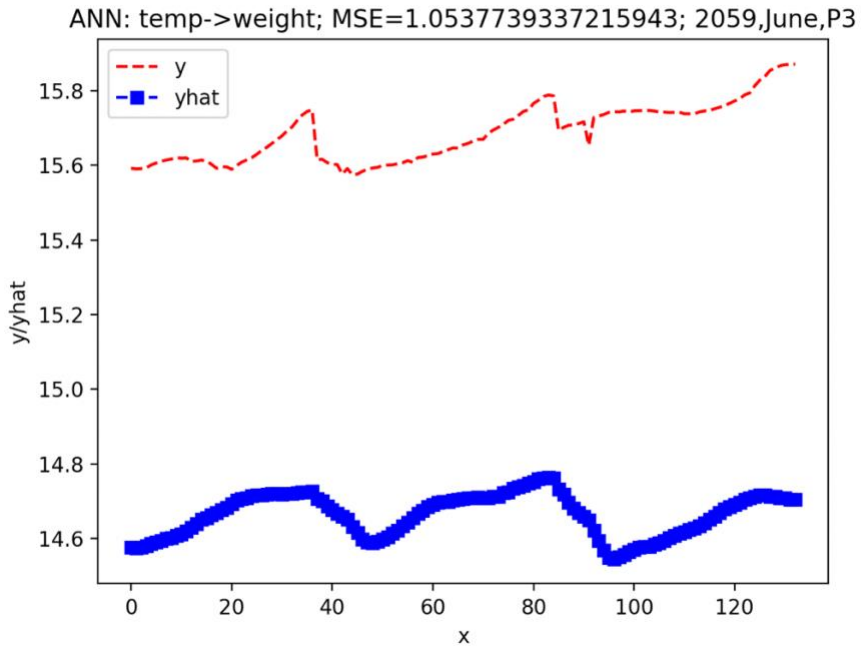


Figure 2.

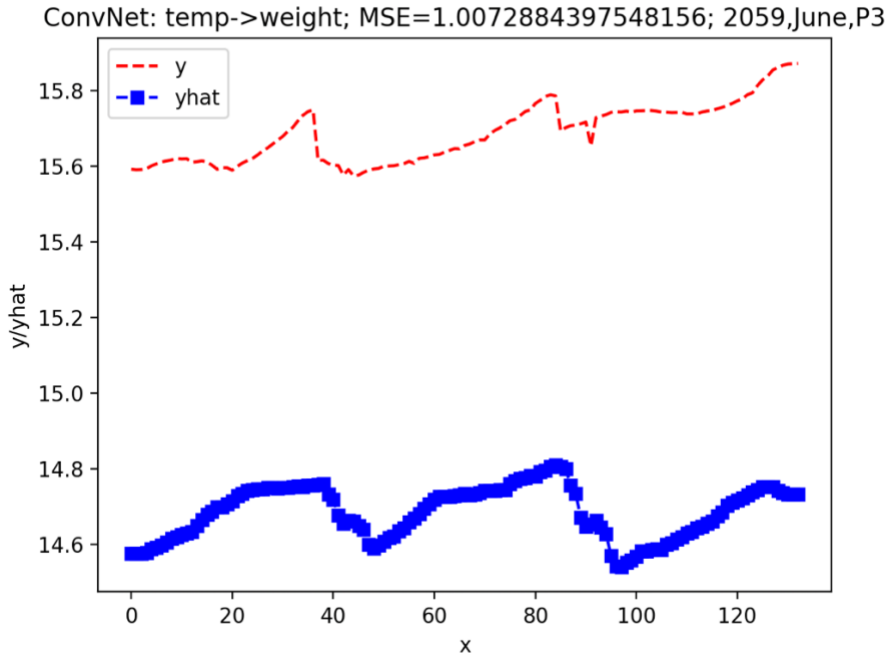
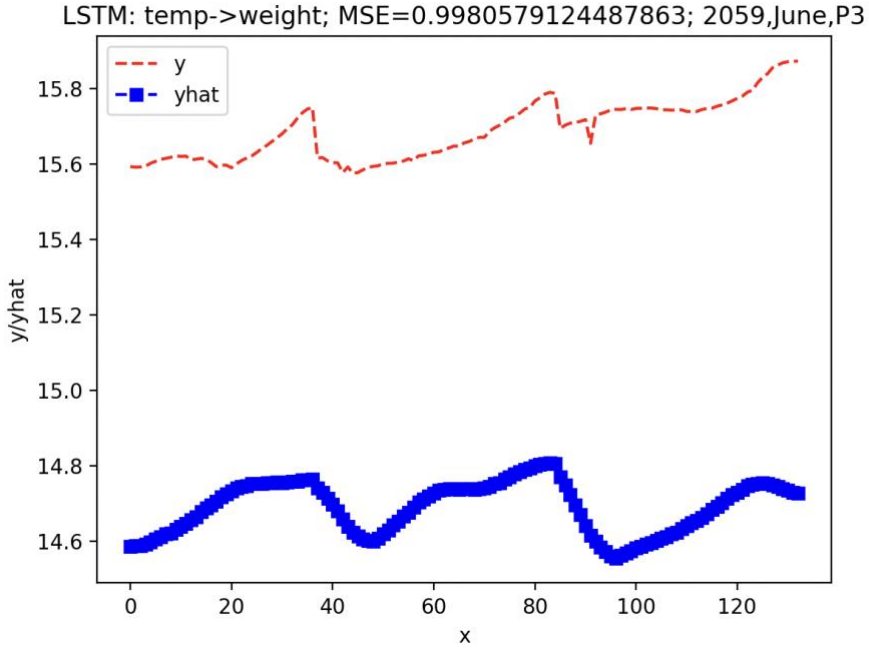


Figure 3.



Part 2: Image Classification

The image data used to train the ResNet50 and YOLO ConvNet models are from the "BEE4" dataset, which are images obtained by on-hive video traffic monitors. The classifications were to determine if an image contained "bee" or "non-bee" image data.

Model 1: ResNet50

The ResNet50 model originally contained 3 epochs using the "adam" optimizer and the "CrossEntropyLoss" loss function. Keeping the epochs the same and changing the optimizer to others such as "SGD" or "RMSprop" (while adjusting their respective parameters) did not yield results nearly as good as the "adam" optimizer, so this was kept the same. Additionally, changing the loss function to "MSELoss", "SmoothL1Loss", and "HingeEmbeddingLoss" would not compile due to size mismatches between the predicted outputs and the target labels. Thus, I purely relied on modifying the number of epochs. Initially, after the initial 3 epoch run, I attempted 6 epochs. This resulted in a slightly improved loss and accuracy. However, this took an extremely long time to run, especially on the CPU of my old laptop. I later attempted 8 epochs and let my laptop sit for a good while, and I saw a slightly improved final training accuracy and validation accuracy with a miniscule loss decrement. If I were to own a GPU and could test a larger number of epochs, surely the accuracy would increase even more, provided overfitting does not occur. A summary of these results is found on Table 1.

Model 2: YOLO

The YOLO model originally contained 3 epochs, the "SGD" optimizer, a learning rate of 1e-3, a batch size of 64, and a "CrossEntropyLoss" function, and this yielded very poor results, such as a validation accuracy of 54.20%. When I attempted to slightly increase or decrease the learning rate, the results either had a miniscule change or dramatically decreased in accuracy. However, simply swapping the optimizer from "SGD" to "adam" drastically improved the results, and raising the epochs from 3 to 30 showed an extremely high improvement (such as a validation accuracy of 97.60%). Thus, the learning

rate, the batch size, and the loss function were left alone, but the optimizer was changed to “adam” the epochs were increased to 75. This showed a validation accuracy of 98.40% - a near perfect result. A summary of these findings is found on Table 2.

Table 1:

Metric	3 Epochs	6 Epochs	8 Epochs
Final Training Loss	0.1209	0.1004	0.0958
Final Training Accuracy	95.30%	96.55%	96.60%
Final Validation Loss	0.1591	0.1602	0.1368
Final Validation Accuracy	93.40%	93.70%	94.55%

Table 2:

Metric	Default Configuration	Medium Configuration	Best Configuration
Epochs	3	30	75
Optimizer	SGD	Adam	Adam
Final Training Loss	0.7183	0.0501	0.0193
Final Training Accuracy	56.44%	98.53%	99.27%
Final Validation Loss	0.7875	0.0779	0.0534
Final Validation Accuracy	54.20%	97.60%	98.40%

Concluding Remarks

This technical report examined the performance of various machine learning architectures on distinct datasets. The first part analyzed time series data using Artificial Neural Networks (ANNs), Convolutional Networks (ConvNets), and Long Short-Term Memory (LSTM) models. The data, derived from bee hives, highlighted how different neural network models could predict hive weights based on temperature. The second part focused on image classification with ResNet50 and YOLO ConvNets, with an emphasis on tuning loss functions, optimizers, and the number of epochs to get a good training/validation loss and accuracy.

In general, increasing the number of epochs showed better results, but over-doing this can cause overfitting, so one must be cautious when taking this approach. The “relu” activation and “adam” optimizer showed consistently better results than other optimizers and activations throughout this project. This likely indicates the data being used correlated well with these structures, but it should be advised that if one were to use a different dataset, they may have to do some fine-tuning to get a proper result (which could implicate “relu” and “adam” should not be used).

In Part 1, while the shape of the graphs was fairly consistent with the actual data, this was slightly off-set for each model. Not only so, but the prediction’s peaks and troughs do not line up with the real data in a 1 to 1 manner. This indicates that the model architectures could use some continued improvements, or the data should be expanded past June 2022 (Period 3).

In Part 2, the ResNet50 model may need further improvements. While the original requirements section indicated to only change model parameters and not the underlining architecture, modifcations of this architecture may allow for further improvements. Additionally, increasing the number of epochs (by utilizing a GPU) most likely would have showcased much better results. The YOLO model showed the best results out of any model in this report, which indicates that the model architecture and the chosen parameters were a great fit for the datasets being used.