Capstone Research Final Report:
Multivariate LSTM Fully Convolutional Networks for Time Series Classification

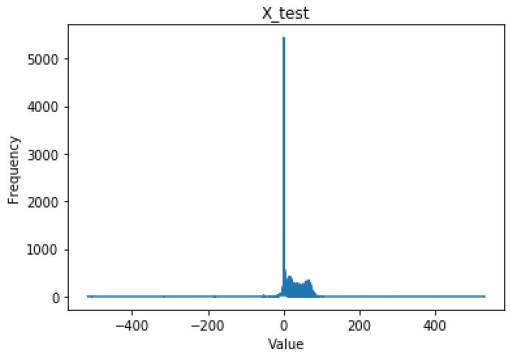
Introduction and Progress

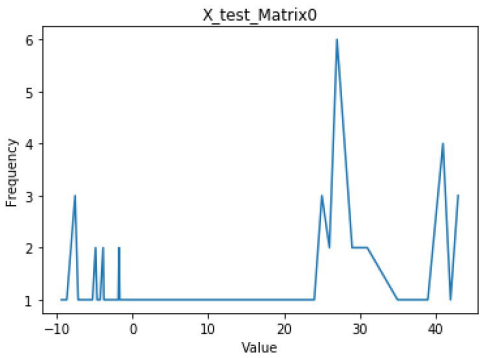
For my capstone research project, I conducted research on the multivariate LSTM fully convolutional networks for time series classification. Throughout the semester, I studied, analyzed, and developed a multi-layer feedforward neural network using Tensorflow while analyzing the model of MLSTM-FCN with the Keras deep learning library. In addition, different models of MLSTM-FCN, including LSTM-FCN and ALSTM-FCN, were also trained and compared on 10 different datasets with adding layers, throughout which the values of accuracy were recorded and analyzed. Finally, a new data set was given and analyzed through several computational methods.

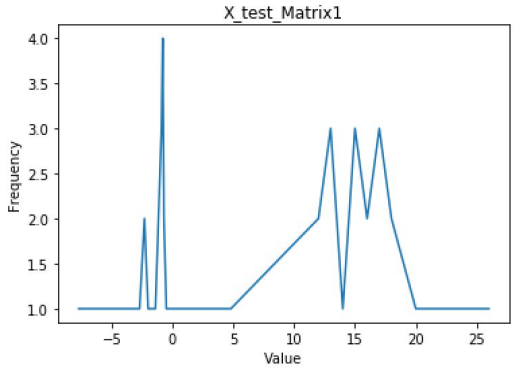
During the training progress of the 10 different datasets, the final accuracy varied from 0.4844 to 0.9892 under different conditions. In the first place, different models were tested regarding the same dataset, including LSTM-FCN and ALSTM-FCN, whose results varied within a range of 0.1. For the LSTM-FCN model, after analyzing the japanese vowels dataset, the ozone layer dataset, the EEG dataset, and the lp1-4 dataset, it could be concluded that adding layers would generally improve the accuracy for MLSTM FCN model, along with a subtle tendency of decrease in accuracy as more layers are added (but not to the depth of decreasing overall accuracy since the first layer was added). In addition, the accuracy could be decreasing due to the possibility of model overfitted.

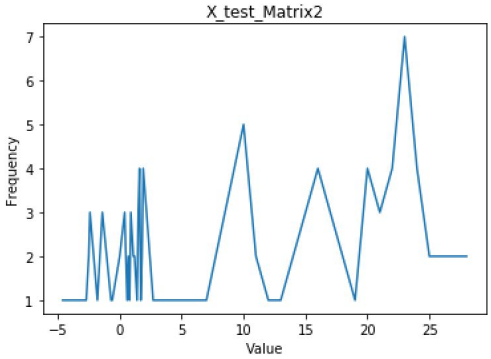
However, when adding more layers (one by one) on the two different models, fluctuations occurred: for the EEG dataset, with the MALSTM STN model, the accuracy appears to decrease when adding more layers but kept constant when adding two more layers. For the lp1 dataset, however, when adding 1 to 3 more layers, the accuracy increased, whereas when two more layers were added, the accuracy dropped to the same as the original one.

After training and comparisons of 10 different datasets of the model were conducted, dataset analysis was performed on the new dataset named cut data. The dataset included X and Y training files in npy format, which was dealt with as numpy arrays during the research. The following are the graphs generated by analyzing the frequencies of values for each 3 matrices in X test and X train, respectively:

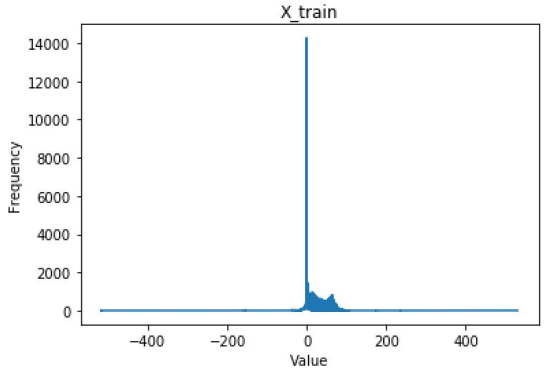

X_test
Frequency
Value
5000
4000
3000
2000
1000
0
−400
−200
0
200
400

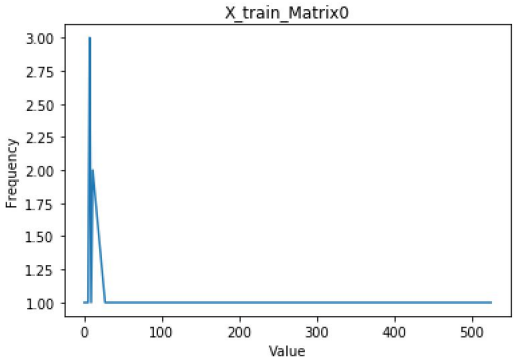

X_test_Matrix0
Frequency
Value
6
5
4
3
2
1
−10
0
10
20
30
40

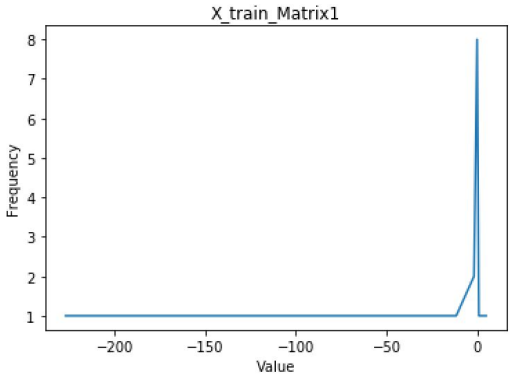

X_test_Matrix1
Frequency
Value
4.0
3.5
3.0
2.5
2.0
1.5
1.0
−5
0
5
10
15
20
25

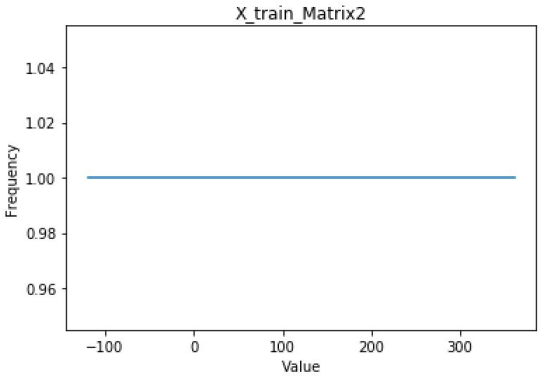

X_test_Matrix2
Frequency
Value
7
6
5
4
3
2
1
−5
0
5
10
15
20
25

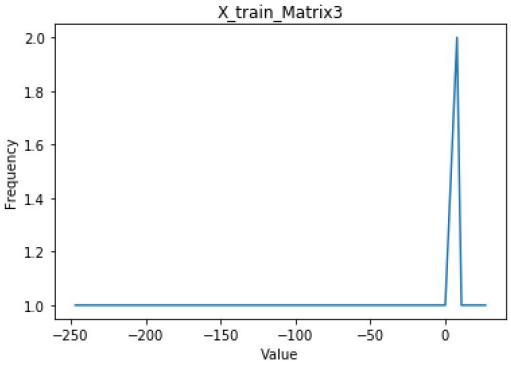
The following are the graphs generated by analyzing the frequencies of values for each 3 matrices in X train, respectively:











Regarding X train variables, the shape of the data included 1844 matrices with each element taking up 2 lines and each line 225 arrays in the datatype of float64. Therefore, the number of elements in a numpy array of X train was 1844 * 2. The same calculations were applied to X test variables, which had a shape of (791, 2, 225) interpreted as the number of arrays being 225, each element 2 lines, each line 225 arrays in the datatype of float64. Therefore, the number of elements in a numpy array of X test was 791 * 2. Y train variables were solved by allowing pickles to be true to the np.load argument, resulting in a shape of 1844 elements with each line one element in the datatype of object. Y test variables were concluded to have a shape of 791 elements, with each line containing one element in the datatype of object as well.

Afterward, the five-number analysis was also performed on the X and Y test and train variables. In addition to the computational analysis, the data analysis process included checking the number of nans by marking out every nan element throughout the matrices. The number of unique vectors was also checked using numpy libraries. Through plotting the graph of the number of frequencies of different values of X and Y test and train parameters for each matrix, the range of X and Y and the distribution of the variables for the dataset was analyzed.

Literature review

The study on the neural network includes topics of understandings of neurons, input, output, bias, weight, activation function, loss function, feed-forward neural network, backpropagation, and gradient descent optimization. Neurons, serving as basic units of the neural networks, structure in layers to form different networks, of which multi-layer perceptron is a common type composed of an input layer, a target layer, and a hidden layer (Brownlee, 2016). A neural network can be thought of as a regression model on a set of derived inputs (the hidden units), with the intercept as the bias estimate and the slope as the weight estimate. When there is a single hidden layer, the hidden units can be thought of as regressions applied to linear combinations of the input variables. The hidden unit regressions include an activation function which is a mathematical transformation that is applied to the input layer. A neural network can approximate virtually any association between the inputs and the target. Besides, the gradient descent optimization includes performing optimization and is by far the most common way to optimize neural networks. Putting it together, in order to train a neural network, the steps entail picking a network architecture (connectivity pattern between neurons), deciding the number of input units and output units, and allowing reasonable default on the number of hidden layers.

Regarding MLSTM-FCN, the topics extend to the convolution layer, batch normalization layer, max pooling operation for temporal data, and the difference between epoch, batch size, and iterations. Several notions are clarified in this research, including definitions that one epoch is when an entire dataset is passed forward and backward through the neural network only once, one batch is the total number of training examples present in a single batch, and iterations are the number of batches needed to complete one epoch (Sharma, 2017). Due to the fact that the more hidden layers the better, the number of layers added in the research models tend to increase regularly one by one.

Future work

Although the conclusions of the influence of the different number of layers on different MLSTM FCN models were initially established, further testing and training on various datasets are needed. Also, different forms of gradient descent optimization algorithms could be applied to the models, such as Adam and Nesterov accelerated gradient (Karim, 2018). Besides, different activation functions could also be applied to the MLSTM FCN models to study the effect of change of activation functions on the final accuracy of the model of different datasets.

Bibliography

Maynard-Reid, Margaret. (2018, April 20). Anaconda, Jupyter Notebook, Tensorflow and Keras for Deep Learning. *Medium.* Retrieved from: https://medium.com/@margaretmz/anaconda-jupyter-notebook-tensorflow-and-keras-b91f381405f8

Brownlee, Jason. (2016, June 7). Binary Classification Tutorial with the Keras Deep Learning Library. *Machine Learning Mastery.* Retrieved from: https://machinelearningmastery.com/binary-classification-tutorial-with-the-keras-deep-learning-library/

Brownlee, Jason. (2019, January 28). Loss and Loss Functions for Training Deep Learning Neural Networks. *Machine Learning Mastery.* Retrieved from: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/

Gupta, Tushar. (2017, January 5). Deep Learning: Feedforward Neural Network. *Medium.* Retrieved from: https://towardsdatascience.com/deep-learning-feedforward-neural-network-26a6705dbdc7

Sharma, Sagar. (2017, September 23). Epoch vs Batch Size vs Iterations. *Medium.* Retrieved from: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9

Karim, Raimi. (2018, November 22). 10 Gradient Descent Optimisation Algorithms. *Towards Data Science.* Retrieved from: https://towardsdatascience.com/10-gradient-descent-optimisation-algorithms-86989510b5e9