CZ 4042 - Neural Networks - Project 1

Submitted by

Joshi Chaitanya Krishna (U1522971F)

Part 1 - Classification Problem

Introduction and Dataset

We are given the spam email dataset containing attributes (57 for each data point) obtained from spam emails and normal emails together with their corresponding binary label: "spam" or "not spam". We have to build a model to classify an email as “spam” or “not spam”, given its attributes.

Architecture and Preprocessing

We shall use a deep neural network for the task. The basic model architecture is as follows- an input layer with 57 neurons (number of attributes), one/multiple hidden layers with variable size and activations, and an output layer with 2 neurons which performs softmax regression over the binary class labels. The network learns to minimise categorical cross-entropy loss. We add dropout layers between the hidden layers and employ L2-norm regularization on the model weights to prevent overfitting. We conduct exhaustive experiments to determine the model hyperparameters like number of hidden layers, hidden layer size, activation functions, etc as described in the following sections.

We preprocessed all real-valued attributes in our data to zero mean and unit standard deviation as this is a known technique for improving neural network performance and speed up convergence. Note that mean and standard deviation were only calculated over training data.

All models were trained in an online learning fashion using tensorflow (specifically, tflearn, a high-level API for tensorflow) and results were visualized using tensorboard.

Choosing Hyperparameters

Initial tests were conducted on a two-way data split (80% training set and 20% validation sets) where the data was randomly shuffled before splitting. Various combinations of model hyperparameters were trained on the training set. For each model, accuracy and loss functions were plotted to choose the best model configuration.

Finally, similar tests were conducted on a three-way data split (70% training, 15% validation and 15% testing) after random shuffling. Various model architectures were trained on the training set and evaluated

on the validation set. The model with the best validation performance was trained on data from both training and validation sets. This is the final model to be assessed on the testing set.

Experimental results are described in the following sections.

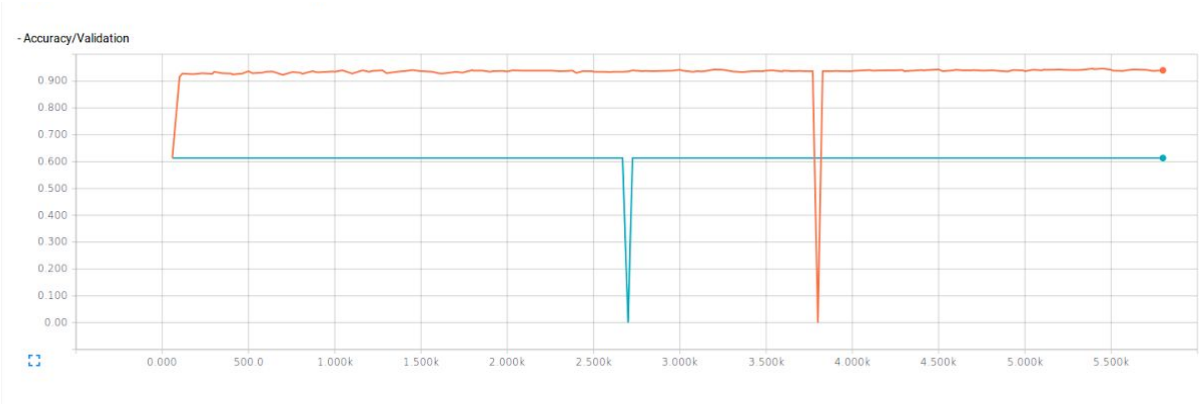
Two-way Data Split

Please note that we could not include all graphs in the report due to space constraints. They can be accessed under img/train-val/<experiment-type>. For each experiment, we found plots of Accuracy (training and validation) and Loss (training and validation).

1. Learning Algorithm

We tested two learning algorithms- vanilla Stochastic Gradient Descent (SGD) and Adaptive Momentum Estimation (Adam). Both were trained to minimise a categorical cross-entropy loss in an online manner.

The graphs show validation set accuracy for two models with 3 hidden layers (50-30-15) using the ReLU activation function and a learning rate of 0.002, trained for 100 epochs each. The orange plot represents the model trained with Adam and the blue plot is the model trained with SGD (with a decay parameter value of 0.96).

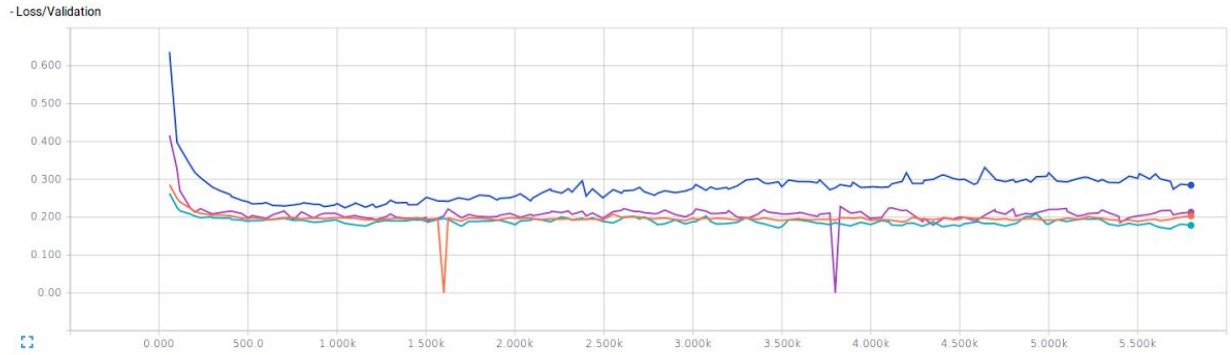


We chose Adam over SGD because, all other hyperparameters being same, the model trained with Adam gave far superior accuracy and lower loss on both training and validation sets. SGD probably gets stuck in a local minima and is unable to improve accuracy beyond 60%.

2. Number of Hidden Layers

We tested for up to four hidden layers. The models used the relu activation function and were trained with Adam at a learning rate of 0.002, trained for 100 epochs each. The graph shows validation loss for various models.

The orange plot is 1 hidden layer (30 neurons), light blue is 2 hidden layers (50-30), purple is 3 hidden layers (50-30-15) and dark blue is 4 hidden layers (50-30-15-10).

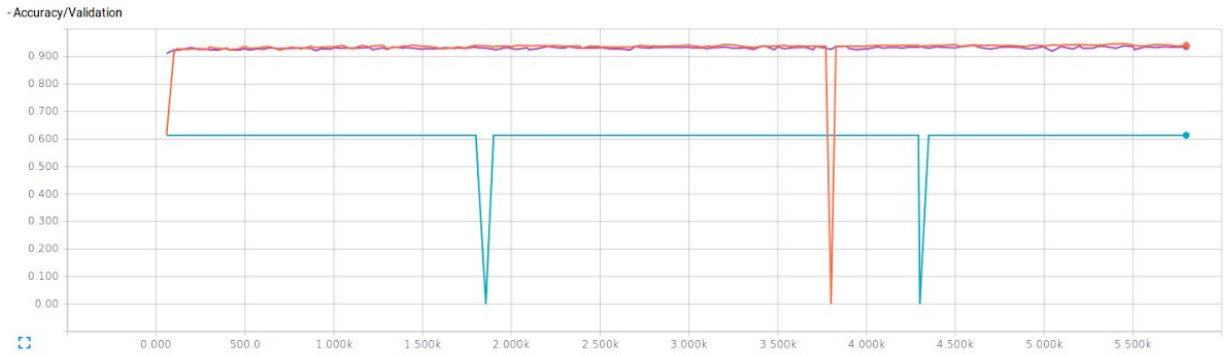


Although all models gave very similar validation accuracy, validation loss is lowest for 2 hidden layers. It is clear that increasing number of hidden layers beyond 2 does not improve the model's prediction capacity despite making it more complicated.

3. Activation Function

We ran test for three different activation functions- tanh, sigmoid and Rectified Linear Units (ReLU). All models had 3 hidden layers (50-30-15) and were trained with Adam at a learning rate of 0.002, trained for 100 epochs each. The graph shows validation accuracy for the models.

The orange plot is ReLU, blue is sigmoid and purple is tanh.



It is obvious that we can eliminate sigmoid as an option. We also see that ReLU gives slightly better validation accuracy than tanh, although tanh has slightly lower validation loss. Although theoretically different functions (tanh squashes real values between -1 and 1), the performance of tanh and ReLU are comparable here.

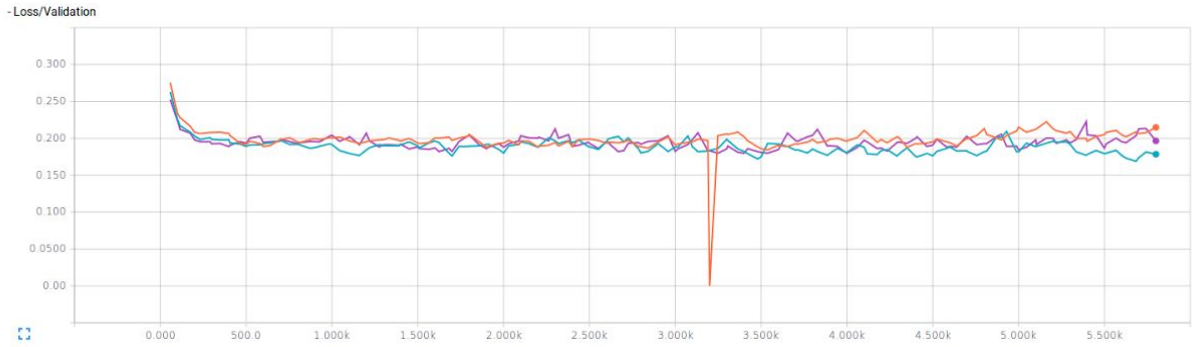
4. Learning Rate

We tested a 2 hidden layer (30-15) model with ReLU activation function, trained with Adam for 100 epochs, at four different learning rates- 0.001 (orange), 0.002 (light blue), 0.005 (purple) and 0.01 (dark blue).

From the plot for training loss, we see that varying the learning rate does not drastically affect time for the model to converge. We choose 0.002 as the optimum learning rate due to slightly higher validation accuracy, as shown by the light blue line peaking above the rest in the graph.

5.Hidden Layer Size

After concluding that 2 hidden layers is the optimum parameter, we tested 3 different combinations of number of neurons in the 2 hidden layers- 40 and 15 (orange), 50 and 30 (blue), and 70 and 40 (purple). The models used ReLU as the activation function and were trained with Adam at a learning rate of 0.002 for 100 epochs each.



Validation accuracies of the larger models are marginally higher than the 40-15 model. The best parameter is 50-30 because it matches the 70-40 model for validation accuracy while having a lower validation loss. Hence, the extra neurons in the hidden layers are not causing drastic changes in model's performance.

Final Model

We trained various models on the training set and used validation accuracy and loss to chose the best model hyperparameters. The final model is a 2 hidden layer (30-15) neural network which uses the ReLU activation function and is trained with Adam at a learning rate of 0.002 for 100 epochs.

We would like to note here that we have not employed any early stopping and continued training for 100 epochs because there was no drastic variation in training or validation accuracy once the models converged. Using dropout and L2-norm regularization helped counter overfitting.

Three-way Data Split

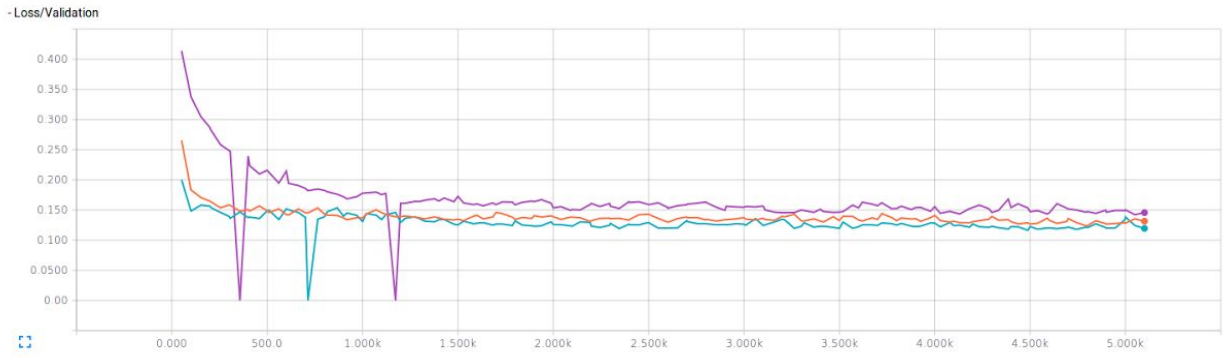
We proceeded with experiments on the three-way data split exactly like the two-way data split. In this case, there is less data to train and validate on, as 15% of the dataset is held out for testing. We get similar results as the two-way split and hence, for brevity, only mention the models tested and show the plots used to choose the best parameters.

We did not train models with different learning algorithms due to Adam being vastly superior to SGD in previous tests. Again, we do not employ an early stopping as we are already accounting for overfitting.

All graphs are accessible at img/train-val-test/<experiment-type>

1. Number of Hidden Layers

Orange- 1 hidden layer (30 neurons), ReLU activation function, trained with Adam at learning rate 0.002 for 100 epochs.
Blue- 2 hidden layer (50-30), ReLU activation function, trained with Adam at learning rate 0.002 for 100 epochs.
Purple- 3 hidden layers (50-30-15), ReLU activation function, trained with Adam at learning rate 0.002 for 100 epochs.

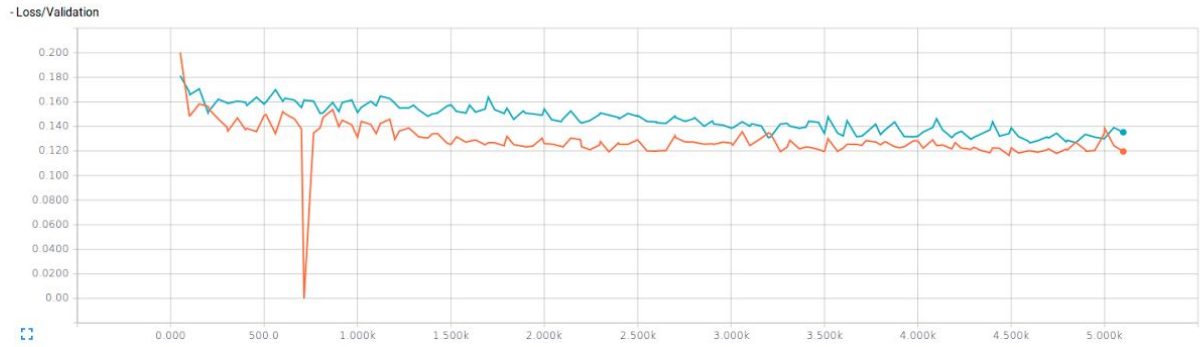


The model with 2 hidden layers has the lowest validation loss.

2. Activation Function

Orange- 2 hidden layer (50-30), ReLU activation function, trained with Adam at learning rate 0.002 for 100 epochs.

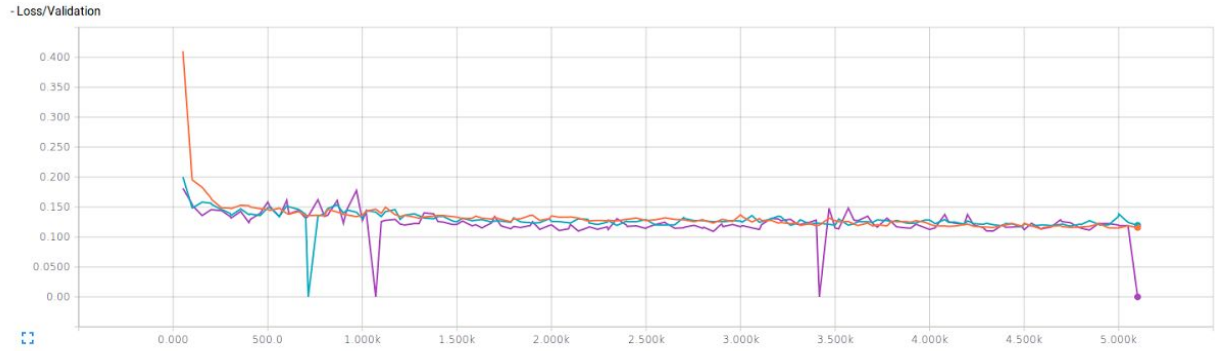
Blue- 2 hidden layer (50-30), tanh activation function, trained with Adam at learning rate 0.002 for 100 epochs.



Using ReLU as the activation function leads to a lower validation loss.

3. Learning Rate

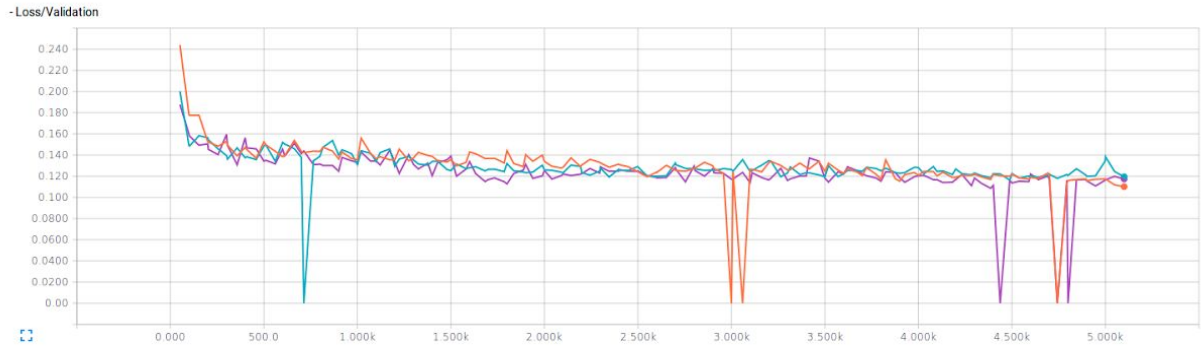
Orange- 2 hidden layer (50-30), ReLU activation function, trained with Adam at learning rate 0.001 for 100 epochs.
Blue- 2 hidden layer (50-30), ReLU activation function, trained with Adam at learning rate 0.002 for 100 epochs.
Purple- 2 hidden layer (50-30), ReLU activation function, trained with Adam at learning rate 0.005 for 100 epochs.



The higher learning rate of 0.005 gave the lowest validation loss.

4. Hidden Layer Size

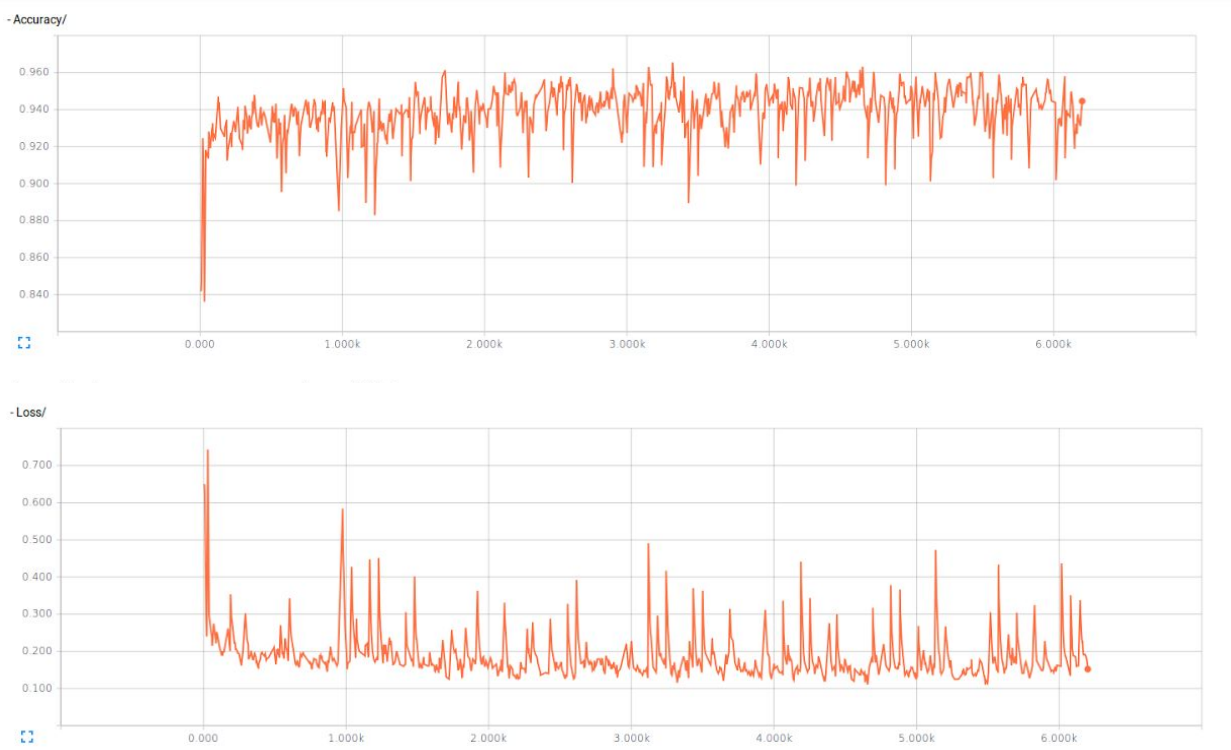
Orange- 2 hidden layer (40-15), ReLU activation function, trained with Adam at learning rate 0.002 for 100 epochs.
Blue- 2 hidden layer (50-30), ReLU activation function, trained with Adam at learning rate 0.002 for 100 epochs.
Purple- 2 hidden layer (70-40), ReLU activation function, trained with Adam at learning rate 0.002 for 100 epochs.



Giving the model's hidden layers more neurons (70-40) leads to slightly lower validation loss.

Final Model

The final model to be evaluated on the test set is a 2 hidden layer (70-40) neural network which uses ReLU activation functions and is trained with Adam at a learning rate of 0.005 for 100 epochs. The following graphs show training accuracy and loss when the model is trained on training and validation set data.



Result

On the test data, the model had an accuracy of 95.80.

Part 2 - Regression Problem

Introduction and Dataset

We are given the California Housing Prices Dataset which contains attributes of housing complexes in California such as location, dimensions, etc, together with their corresponding price. We have to build a model to approximate a property's price given its attributes.

Architecture and Preprocessing

We shall use a deep neural network for the task. The basic model architecture is as follows- an input layer with 8 neurons (number of attributes), one/multiple hidden layers with variable sizes and activations, and a linear output layer with 1 neurons which performs regression. The network learns to minimise the mean-square loss. We employ L2-norm regularization on the model weights to prevent overfitting. We conduct exhaustive experiments to determine the model hyperparameters like number of hidden layers, hidden layer size, activation functions, etc as described in the following sections.

We preprocessed all real-valued attributes in our data to zero mean and unit standard deviation as this is a known technique for improving neural network performance and speed up convergence. Note that mean and standard deviation were only calculated over training data. We also divide all the prices by 1000 to deal with smaller numbers in the mean-square loss. This does not affect the results of the model as the predicted price can again be multiplied by 1000 to get the real prediction.

All models were trained in an online learning fashion using tensorflow (specifically, tflearn, a high-level API for tensorflow) and results were visualized using tensorboard.

Choosing Hyperparameters

Initial tests were conducted on a two-way data split (80% training set and 20% validation sets) where the data was randomly shuffled before splitting. Various combinations of model hyperparameters were trained on the training set. For each model, accuracy and loss functions were plotted to choose the best model configuration.

Finally, similar tests were conducted on a three-way data split (70% training, 15% validation and 15% testing) after random shuffling. Various model architectures were trained on the training set and evaluated on the validation set. The model with the best validation performance was trained on data from both training and validation sets. This is the final model to be assessed on the testing set.

Experimental results are described in the following sections.

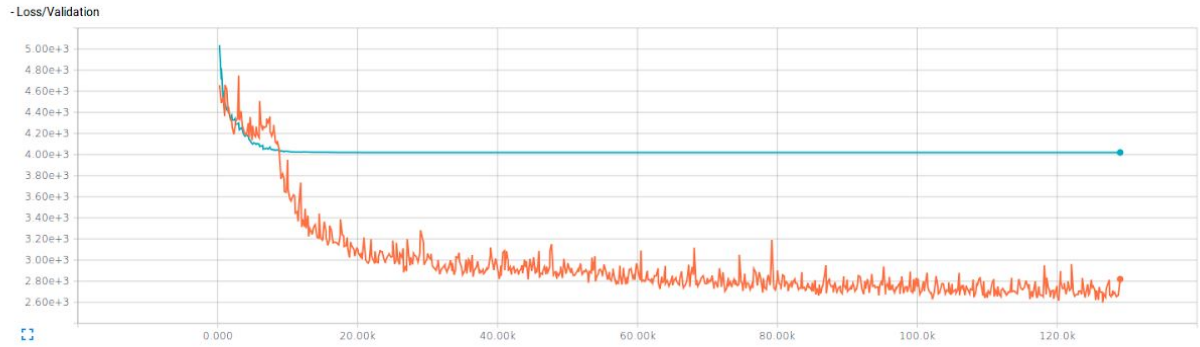
Two-way Data Split

All training and validation loss graphs are accessible at img/train-val

1. Learning Algorithm

We tested two learning algorithms- vanilla Stochastic Gradient Descent (SGD) and Adaptive Momentum Estimation (Adam). Both were trained to minimise mean square loss in an online manner.

The graphs show validation loss for two models with 2 hidden layers (30-15) using the ReLU activation function and a learning rate of 0.02, trained for 500 epochs each. The orange plot represents the model trained with Adam and the blue plot is the model trained with SGD (with a decay parameter value of 0.96).

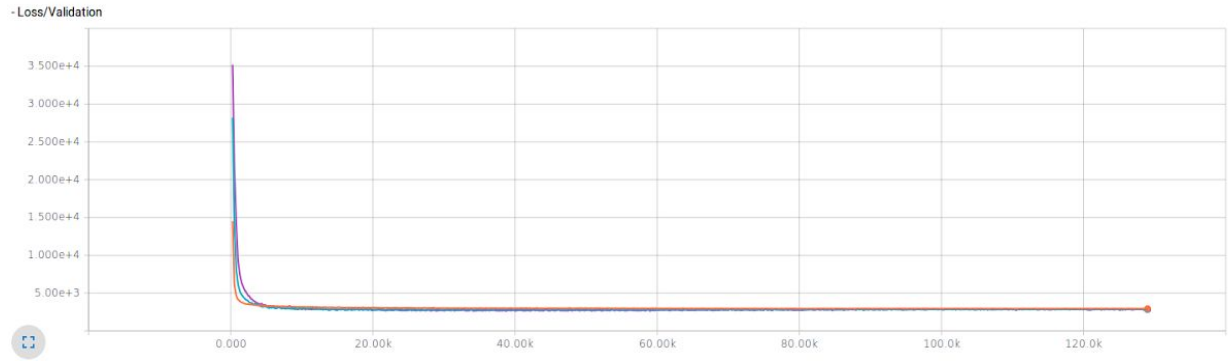


It is clear that using Adam leads to lower validation loss.

2. Number of Hidden Layers

We tested for up to four hidden layers. The models used the sigmoid activation function and were trained with Adam at a learning rate of 0.02, trained for 500 epochs each. The graph shows validation loss for various models.

The orange plot is 1 hidden layer (30 neurons), light blue is 2 hidden layers (30-15) and purple is 3 hidden layers (40-20-10).

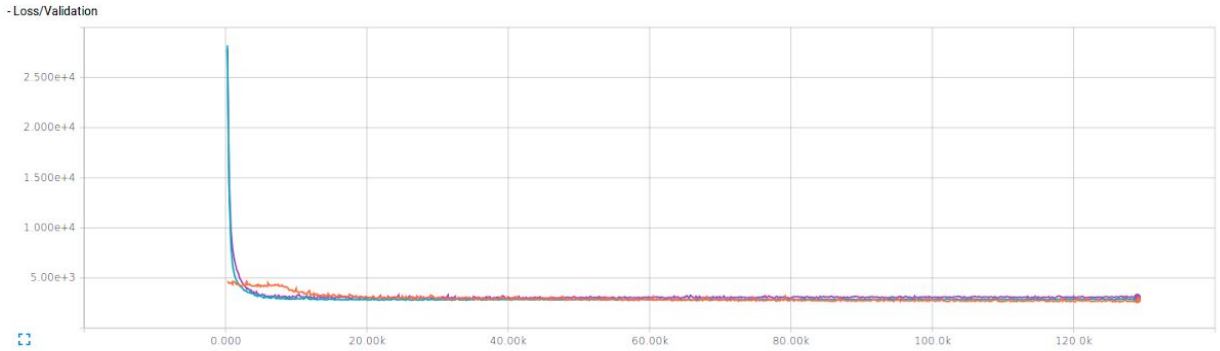


There appears to be no difference in validation loss when increasing or decreasing number of hidden layers. There is slight benefit in using more than one hidden layer but no apparent improvement when increasing from two to three layers. Hence, we shall choose to have 2 hidden layers in our model.

3. Activation Function

We ran test for three different activation functions- tanh, sigmoid and Rectified Linear Units (ReLU). All models had 2 hidden layers (30-15) and were trained with Adam at a learning rate of 0.02, trained for 500 epochs each. The graph shows validation loss for the models.

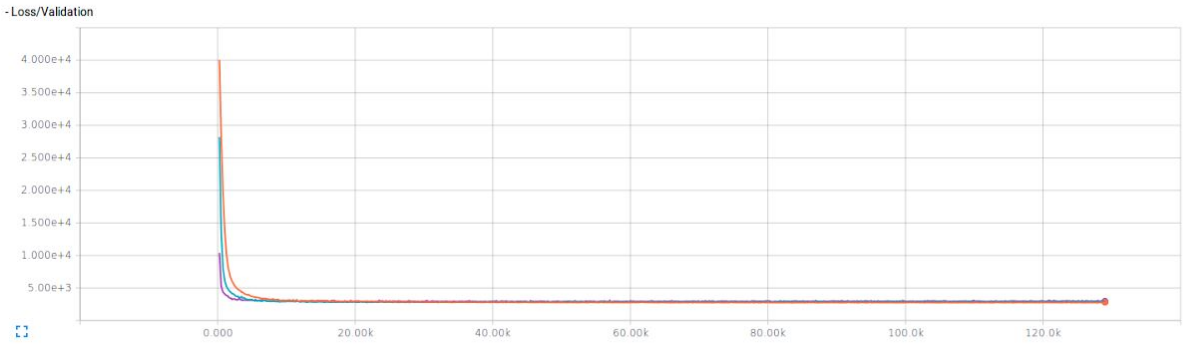
The orange plot is ReLU, blue is sigmoid and purple is tanh.



From the graph, we see that although sigmoid and tanh converge faster, ReLU has the lowest validation loss upon convergence.

4. Learning Rate

We tested a 2 hidden layer (30-15) model with sigmoid activation function, trained with Adam for 500 epochs, at three different learning rates- 0.01 (orange), 0.02 (light blue) and 0.05 (purple).

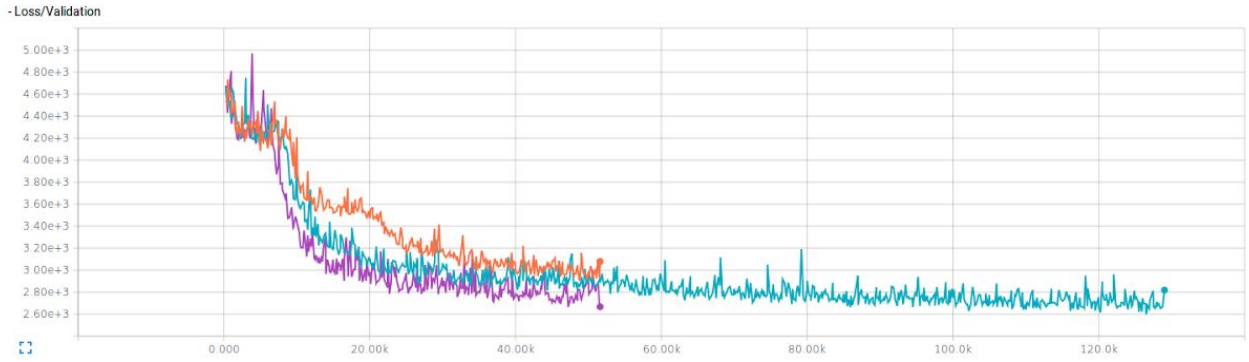


Again, there appears to be no difference in validation loss after the models converge (larger learning rate converges faster). We choose to use 0.02 in our model as it is between the other two and hence relatively ‘safer’.

5. Hidden Layer Size

After concluding that 2 hidden layers is the optimum parameter, we tested 3 different combinations of number of neurons in the 2 hidden layers- 20 and 10 (orange), 30 and 15 (blue), and 50 and 20 (purple). The models used ReLU as the activation function and were trained with Adam at a learning rate of 0.02 for 500 epochs each.

Please note that I took a screenshot of the graph before it could load fully. Please excuse my mistake! The models can be reloaded on tensorboard to see the complete graph,



We see that adding more neurons to our hidden layer leads to lower validation loss and is hence very beneficial for model performance.

Final Model

We trained various models on the training set and used validation loss to chose the best model hyperparameters. The final model is a 2 hidden layer (50-20) neural network which uses the ReLU activation function and is trained with Adam at a learning rate of 0.02 for 500 epochs.

We would like to note here that we have not employed any early stopping and continued training for 500 epochs because there was no drastic variation in training or validation accuracy once the models converged. Using L2-norm regularization helped counter overfitting.

Three-way Data Split

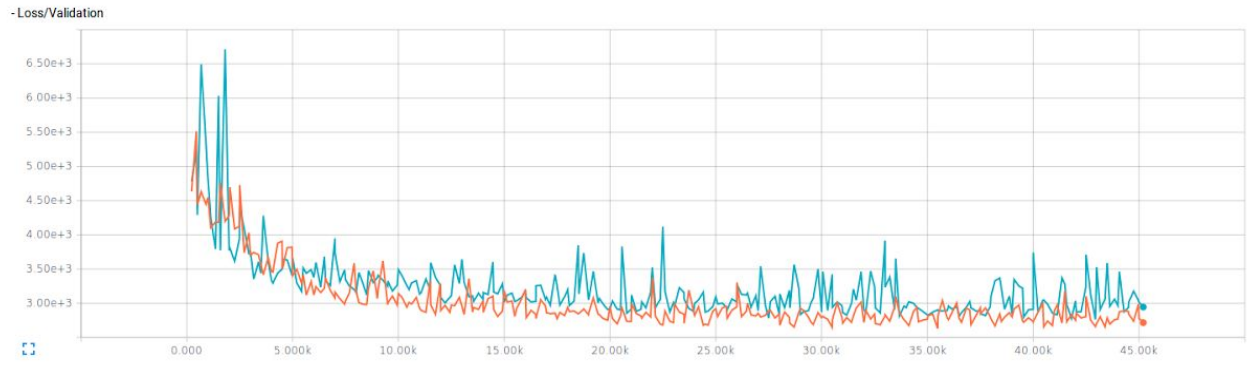
We proceeded with experiments on the three-way data split exactly like the two-way data split. In this case, there is less data to train and validate on, as 15% of the dataset is held out for testing. We get similar results as the two-way split and hence, for brevity, only mention the models tested and show the plots used to choose the best parameters.

We did not train models with different learning algorithms due to Adam being vastly superior to SGD in previous tests. We stopped training at 200 epochs instead of 500 due to time constraints and submission deadlines approaching! Again, we do not employ an early stopping conditions as we are already accounting for overfitting.

All graphs are accessible at img/train-val-test

1. Number of Hidden Layers

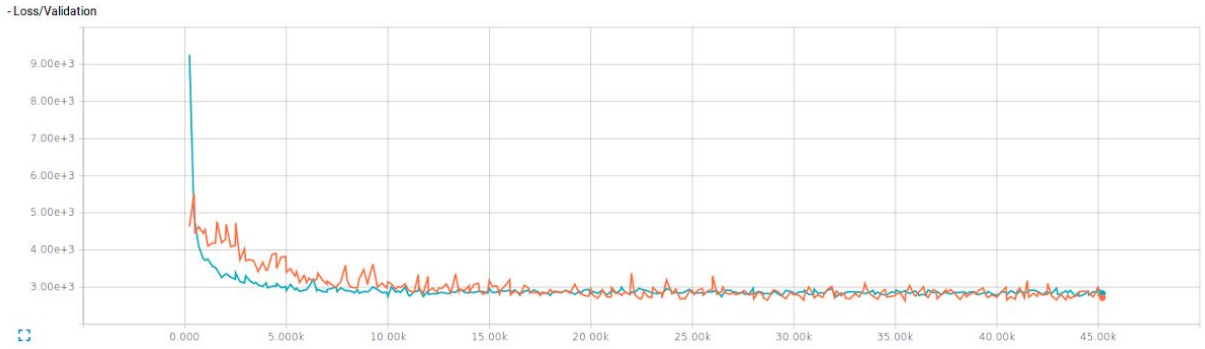
Orange- 2 hidden layer (50-20), ReLU activation function, trained with Adam at learning rate 0.05 for 200 epochs.
Blue- 3 hidden layer (60-30-15), ReLU activation function, trained with Adam at learning rate 0.05 for 200 epochs.



Using 2 hidden layers leads to lower validation loss.

2. Activation Function

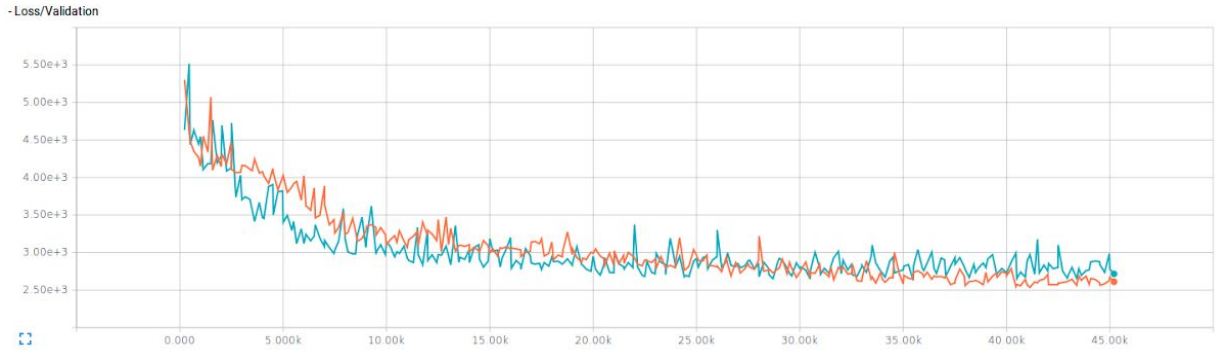
Orange- 2 hidden layer (50-20), ReLU activation function, trained with Adam at learning rate 0.05 for 200 epochs.
Blue- 2 hidden layer (50-20), sigmoid activation function, trained with Adam at learning rate 0.05 for 200 epochs.



Using ReLU as activation function leads to lower validation loss**.**

3. Learning Rate

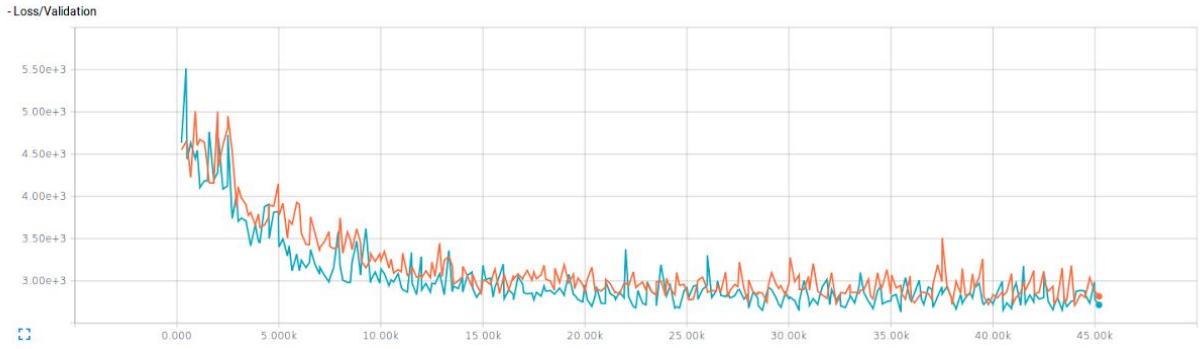
Orange- 2 hidden layer (50-20), ReLU activation function, trained with Adam at learning rate 0.02 for 200 epochs.
Blue- 2 hidden layer (50-20), ReLU activation function, trained with Adam at learning rate 0.05 for 200 epochs.



Using learning rate of 0.02 compared to 0.05 leads to slower convergence but lower validation loss.

4. Hidden Layer Size

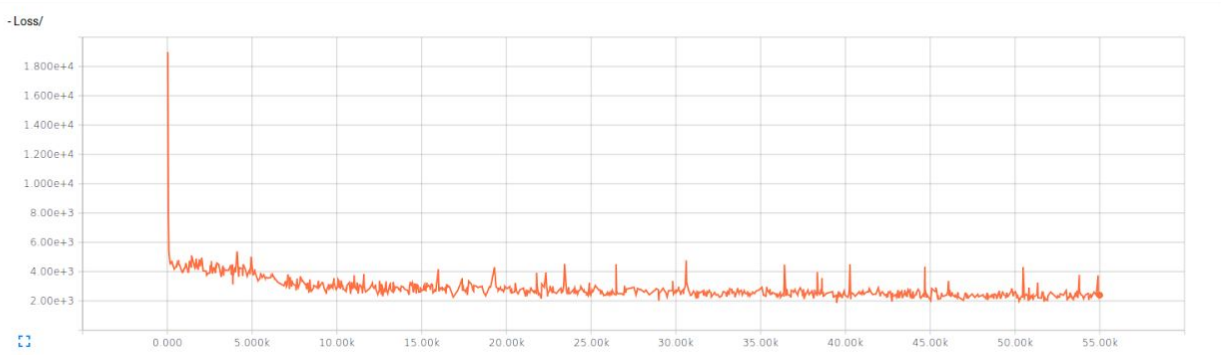
Orange- 2 hidden layer (30-15), ReLU activation function, trained with Adam at learning rate 0.05 for 200 epochs.
Blue- 2 hidden layer (50-20), ReLU activation function, trained with Adam at learning rate 0.05 for 200 epochs.



Giving the model's hidden layers more neurons (50-20) leads to lower validation loss.

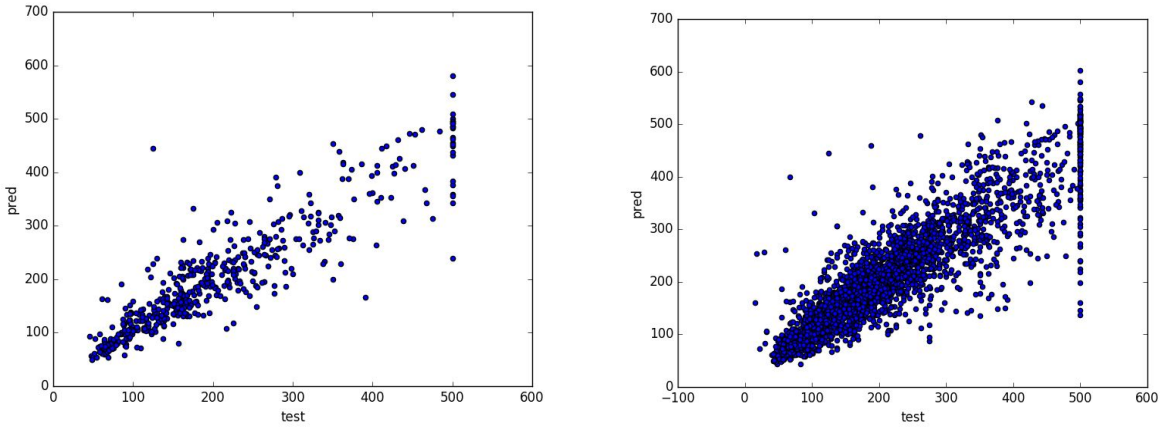
Final Model

The final model to be evaluated on the test set is a 2 hidden layer (50-20) neural network which uses ReLU activation functions and is trained with Adam at a learning rate of 0.02 for 200 epochs. The following graphs show training loss when the model is trained on training and validation set data.



Result

On the test data, the model had a final mean-square loss of 2594.4 (Training loss was 2098.7).



The figures show test data prices vs predicted prices (multiply by 1000 for actual prices) for (1) first 500 data points from test set, and (2) entire test set.

From the fact that the graph is saturated around the line y = x, we can say that our model is decent at predicting the correct prices.

As price increases, the points on the graph move further away from y = x. This shows that the model is better at predicting prices of cheaper property. This corresponds with real life observations- high-end property prices show great fluctuation while prices of cheaper property are considered easier to model mathematically.

Comments about Code

Apart from three python scripts (for two-way split, three-way split and final model training) for each part of the project, I have provided the tensorboard summaries for all the models used in the report. They can be used to visualize the graphs that I have used to select model hyperparameters. Also provided are tflearn model checkpoints to load the all the models (after training completes) used in the report. Please proceed according to tflearn documentation to test various models.