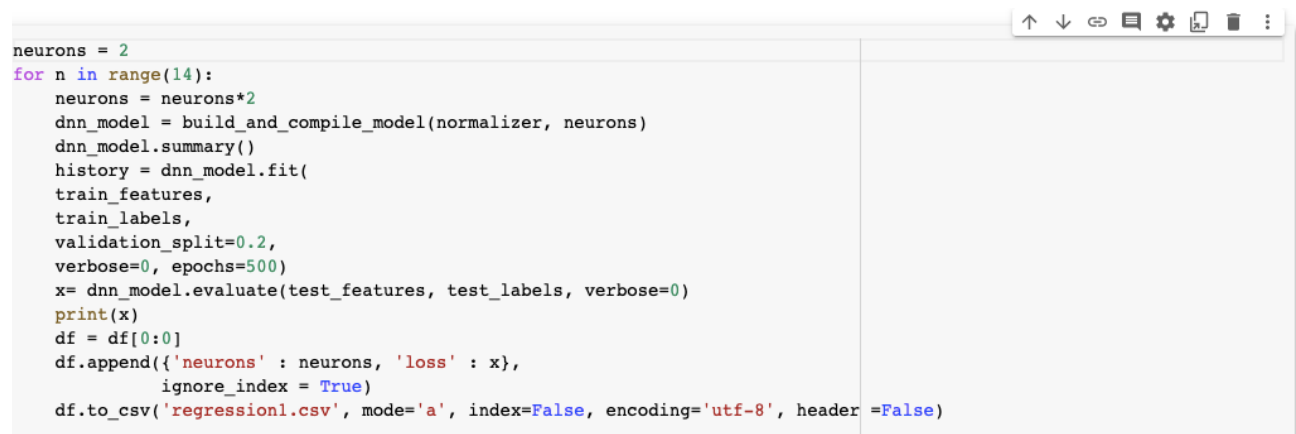


# Assignment 2: Regression

Sunayana Gupta  
1222389090  
[sgupt279@asu.edu](mailto:sgupt279@asu.edu)

In this assignment, we had to vary the number of dense hidden layers and the number of neurons per hidden layer to see and understand why changes are occurring when we changed these numbers in the TensorFlow neural network given to us for the given regression task.

I ran all the neural networks given in the reference material and decided to perform the experiment on the Deep neural network with multiple inputs as that will predict the median housing value considering all the input factors. So first I created a logic that iterated with 1 hidden layer and neurons in that hidden layer varying from 2 to 16384 in multiple of 2 that is 2, 4, 8, etc. in the below fashion:



```
neurons = 2
for n in range(14):
    neurons = neurons*2
    dnn_model = build_and_compile_model(normalizer, neurons)
    dnn_model.summary()
    history = dnn_model.fit(
        train_features,
        train_labels,
        validation_split=0.2,
        verbose=0, epochs=500)
    x= dnn_model.evaluate(test_features, test_labels, verbose=0)
    print(x)
    df = df[0:0]
    df.append({'neurons' : neurons, 'loss' : x},
              ignore_index = True)
    df.to_csv('regression1.csv', mode='a', index=False, encoding='utf-8', header =False)
```

Figure 1 Logic to iterate over the the model with varying number of layers

In the model we had 4 parameters to vary, namely the number of layers, number of neurons, the number of epochs and the learning rate. I tried training multiple neural networks and fixated on 500 epochs as that was making the error stable in the error and number of epochs graph on all the models.

I decided to perform the neurons varying experiment along with varying the learning rate. So I ran the above loop with two sets of the learning rate, 0.01 and 0.1, and below are the graphs showing the results of the same.

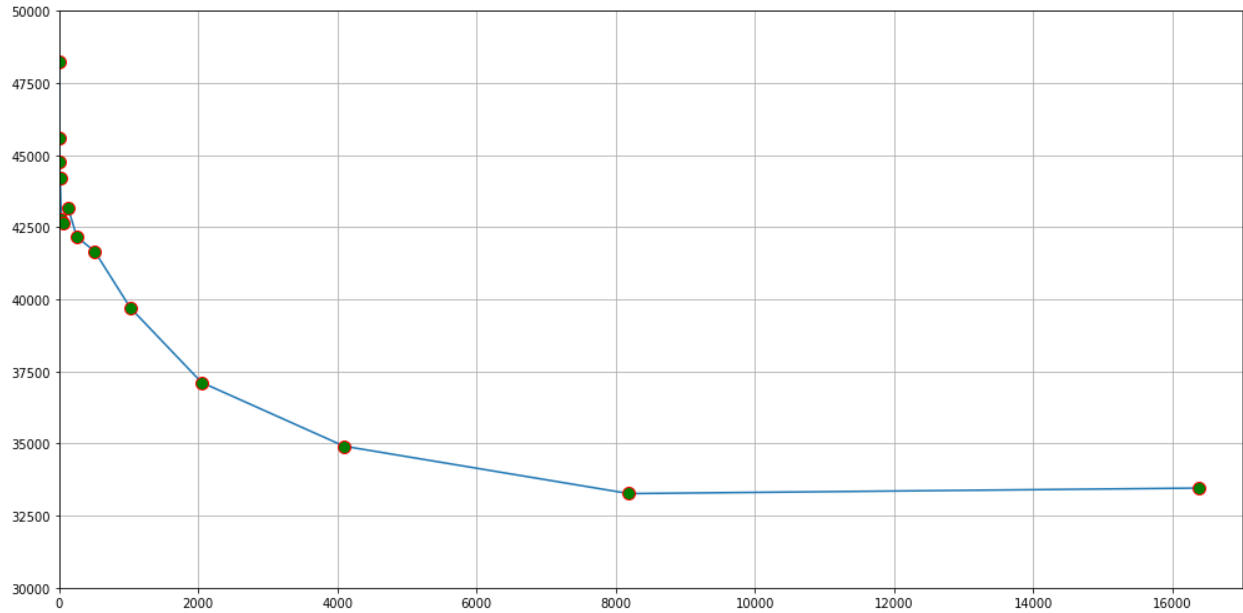


Figure 2 Graph showing Error vs number of neurons for lr 0.1 with one hidden layer

As we can see in this graph the error decreased in a steep manner as we increased the number of neurons from 2 to 128. This is because the model could make out a proper relation below 64-128 neurons. Post 128 neurons also the error continued to decrease but the rate at which the rate was decreasing decremented a little as the model was reaching to its the most optimum number of neurons that were possible for a single hidden layer. As we can see the error did not decrement a lot going from 8192 neurons to 16384 as the model started reaching overfitting.

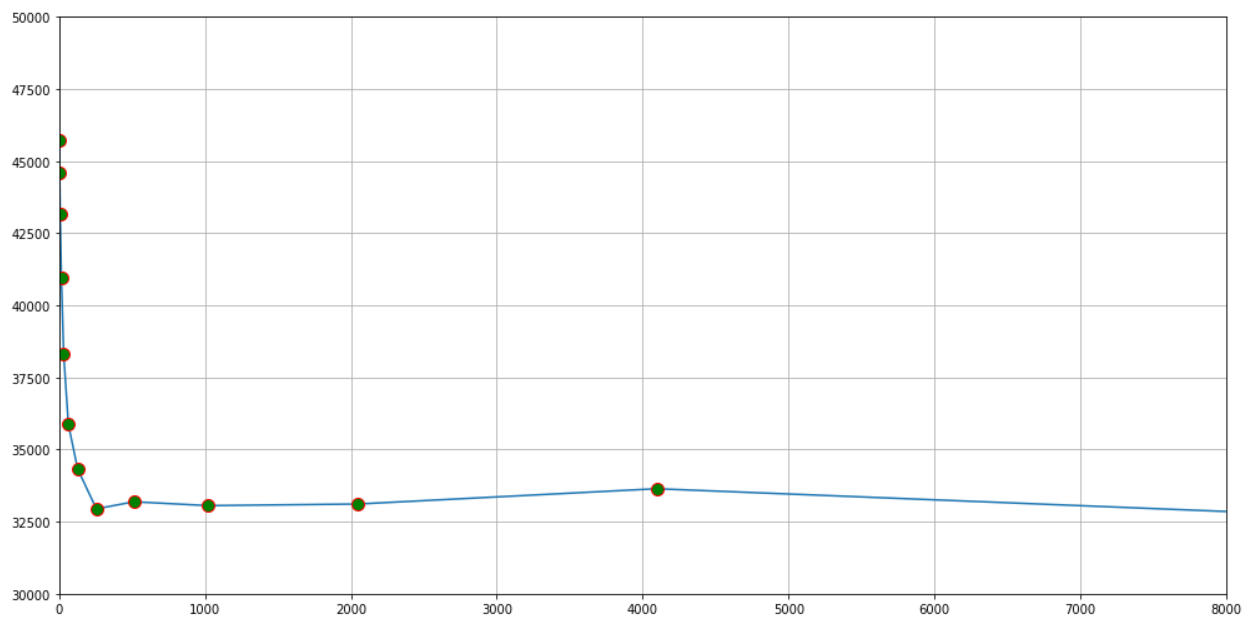
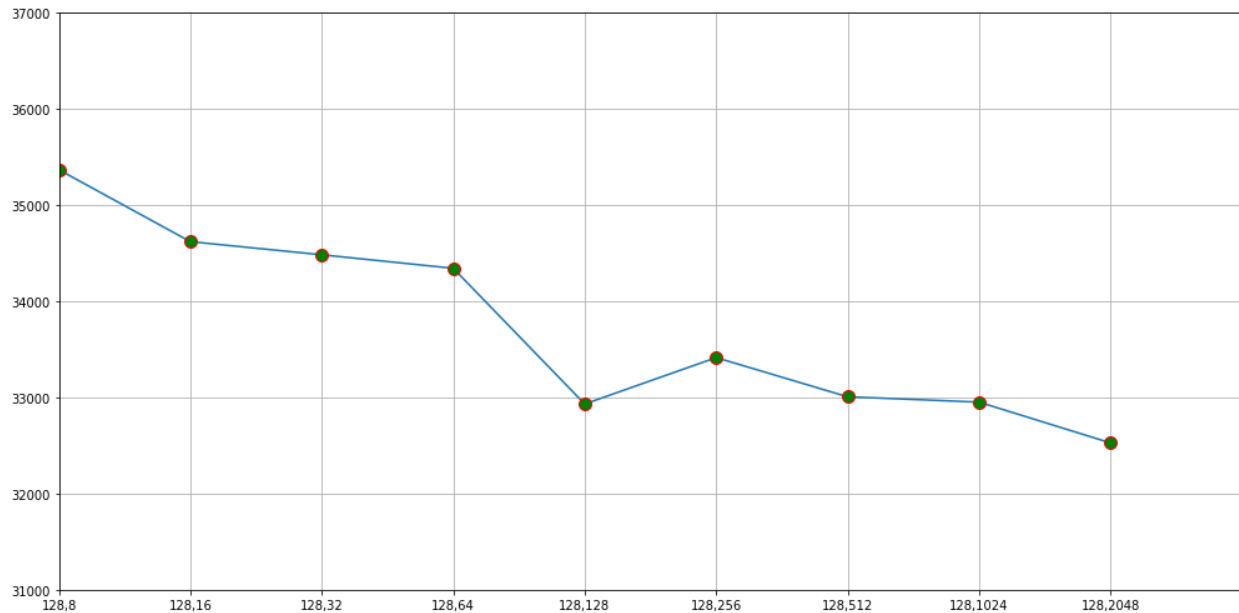


Figure 3 Graph showing Error vs number of neurons for lr 0.01 with one hidden layer

This figure shows some interesting results as when the learning rate was 0.1 then the error took a lot many neurons to reach the optimum error of around 35000 but in 0.01 learning rate it quickly decreased even with 64 neurons and the rate with which the error decreased is even more in this case. And at 4096 neurons only it reached some stability. This analysis was for one hidden layer network, now we will move to 2 layer network. Since 0.01 learning rate networks gave better and more efficient results, I conducted the rest of the experiments only on 0.01 learning rate.

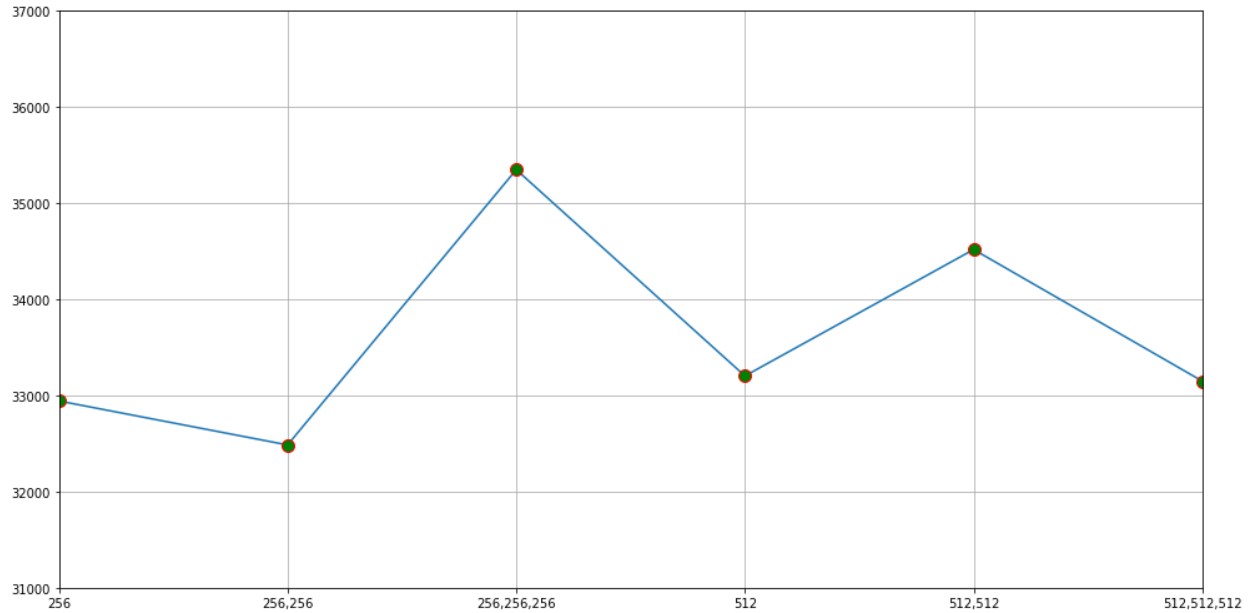
Below is the graph that we got for 2 hidden layer networks:



The starting error for 2 hidden layer networks (35369.48828) was much better than for 1 hidden layer network (43156.0586). This is because in the 2-layer network feature extraction of the dataset is more efficient as more non-linear combinations of the data are possible because of the way the neural networks are structured.

The 32500 mark for error is reached with 128,2048 neurons only in 2 layer network, but in 1 layer network, 8192 neurons were needed to reach the mark of 32500. Hence 2 hidden layer network proved to be more efficient as compared to the 1 later networks.

Now we will move to the 3 later networks:



From this graph we can observe that when we increased the number of hidden layer from one to two then the error decreased for 256 neurons but it increased to 35349 when we again increased it from 2 to 3 which is again due to overfitting. But for 512 it kind of remained constant in 33500 to 34500 range. From this we can conclude that 256,256 is the best combination for the current dataset with a learning rate of 0.01. For this dataset 2 hidden layers are enough for good training of the model.

Below are the URLs of public repository where I have uploaded the results and code

### Code:

I have uploaded the code in the below public repository. Ran this in google colab.

<https://github.com/sunayana17/CSE598IDL/blob/master/regression/RegressionOnlyFinalModel.ipynb>

### Results:

I have uploaded the results in github repo:

Results for 1 hidden layer:

<https://github.com/sunayana17/CSE598IDL/blob/master/regression/lr001.csv>

Results for 2 hidden layer:

<https://github.com/sunayana17/CSE598IDL/blob/master/regression/layer2.csv>

Results for 3 hidden layer:

<https://github.com/sunayana17/CSE598IDL/blob/master/regression/layer3.csv>