April 23, 2021

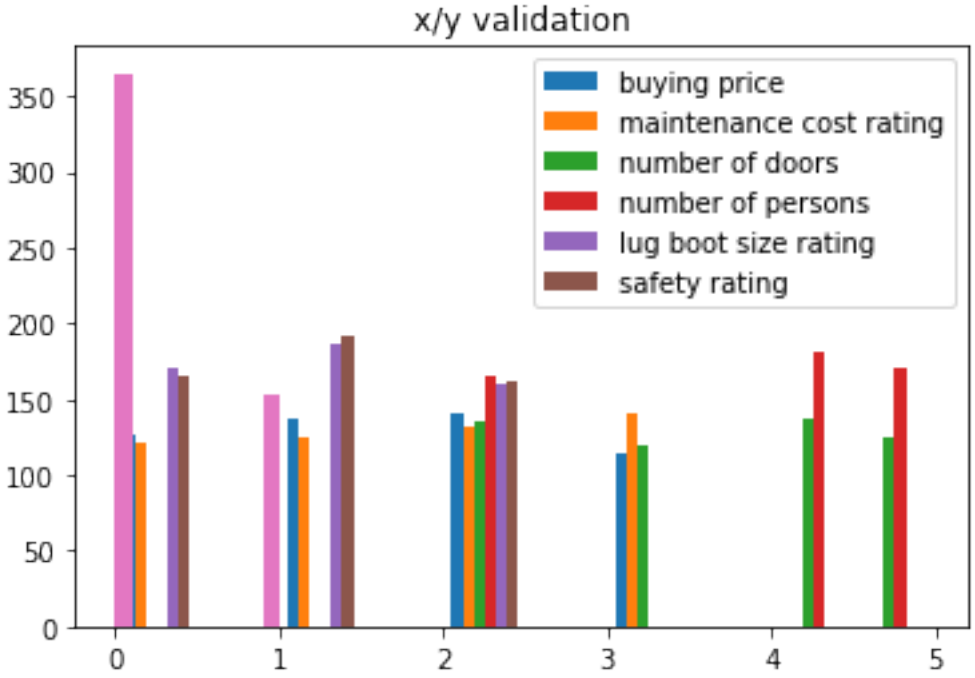
REPORT 2

Steven Guo
cs 4300
University of Missouri-St. Louis
Click Here For Google Colab Source

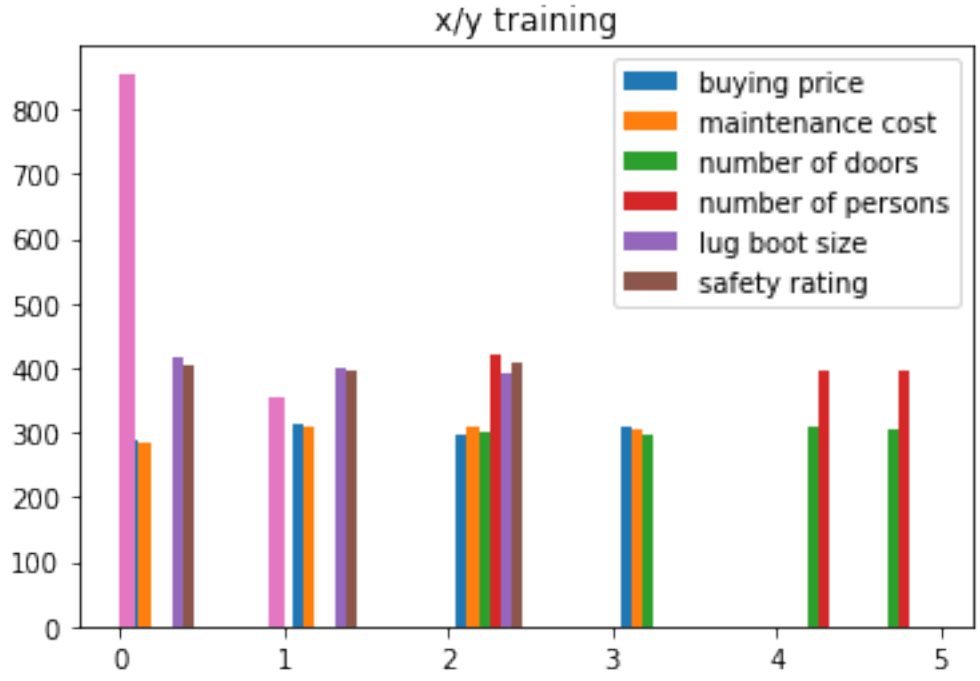
1 Dataset manipulation

I normalized the last column which determines if the car is unacc, acc, good, or vgood. 0,1,2,3 respectively. I made all values not zero equal 1 for logistic regression. Which 0 determines the car to be unacceptable and 1 to be acceptable. Then I find 30 percent of the data and split it into training and validation, 30 percent validation set and 70 percent training.

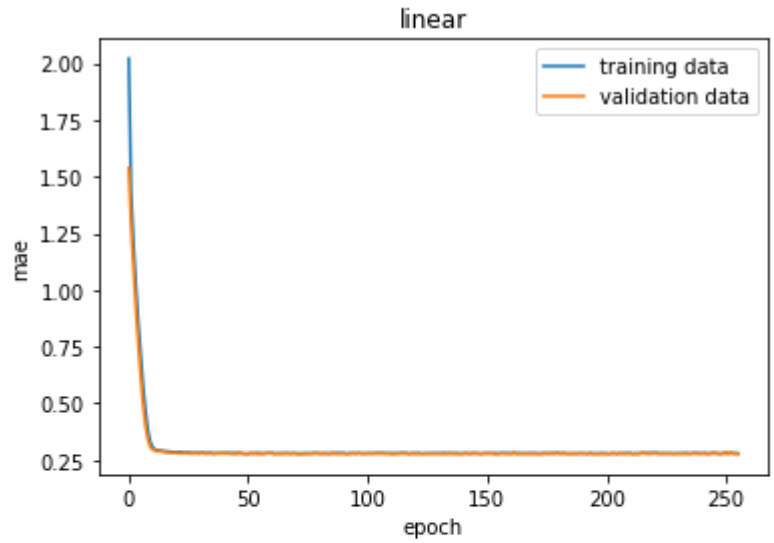
2 Validation Dataset



3 Training Dataset



4 accuracy on the validation set




```
Training
Accuracy: 10.50%
Precision: 56.83%
Recall: 10.50%
F1-score: 0.18
Validation
Accuracy: 9.07%
Precision: 50.33%
Recall: 9.07%
F1-score: 0.15
```

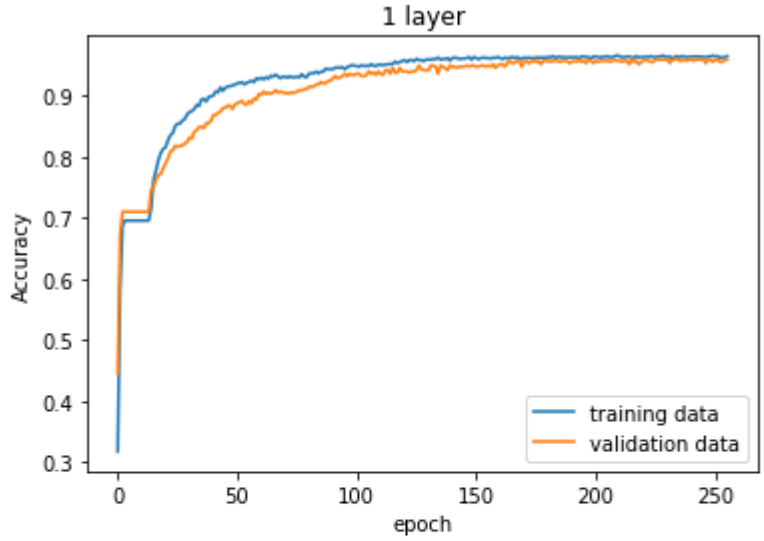
To get this I made a neural network with 6 neurons with linear activation in the first layer and 1 neuron with linear activation in the last layer. The accuracy on the validation set is 10.50 percent

5 Neural network experimentation

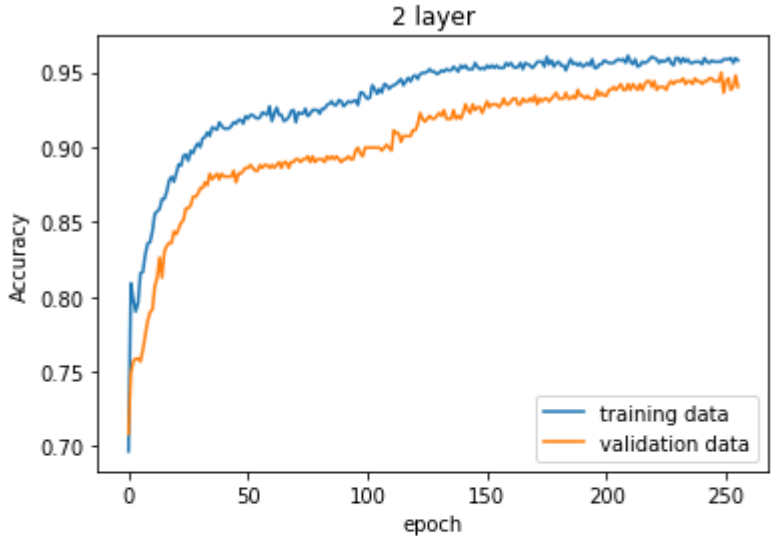
The first layer will have 6 neurons to match the colunms I have. When I add more layers I double the neurons every additional layer. I also use relu activation with the last activation being sigmoid. I tried it with 4 initial neurons and doubling it every additional layer, but it did not like it. I was getting zeros for precision, recall, and f1-scores

model	accuracy	percision	recall	f1-score
1 layer	26.78	18.64	41.85	0.26
2 layer	60	43.08	98.1	0.6
3 layer	58.68	25.19	18.21	0.21
4 layer	77.44	77.46	36.41	0.5
5 layer	78.51	76.21	42.66	0.55
6 layer	74.21	54.73	88.04	0.67
7 layer	75.7	73.72	31.25	0.44
8 layer	70	77.78	1.9	0.04

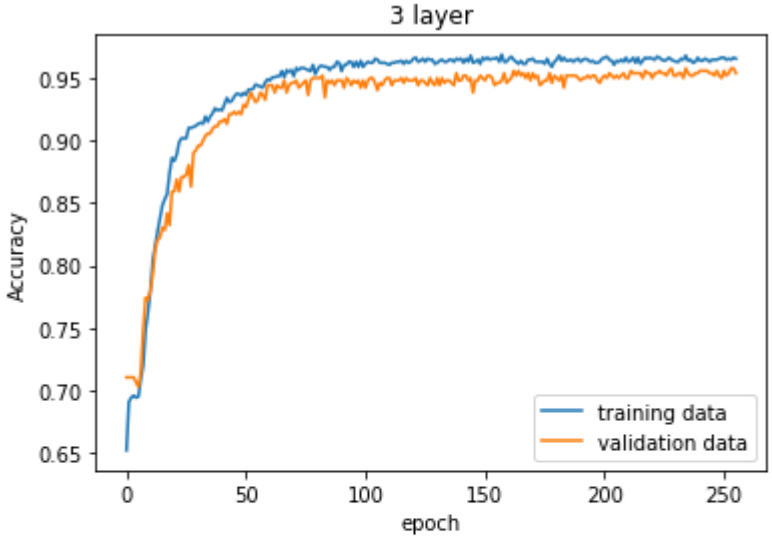
So while I was adding more and more layers, the accuracy got better. However at the 6th layer I noticed that the accuracy was lower than the previous accuracy. So I kept going and found out that 5 layers was the sweet spot. The learning curves for the experiments are below



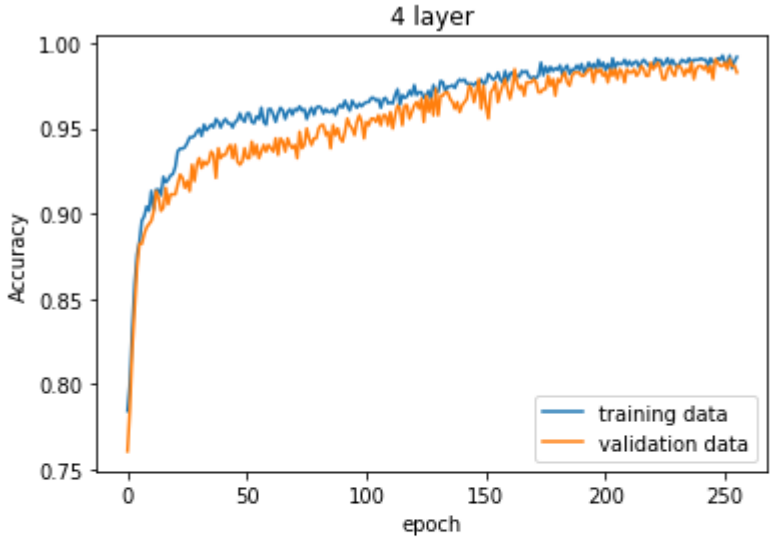

```
Training
Accuracy: 26.78%
Precision: 18.64%
Recall: 41.85%
F1-score: 0.26
Validation
Accuracy: 28.57%
Precision: 20.59%
Recall: 51.33%
F1-score: 0.29
```



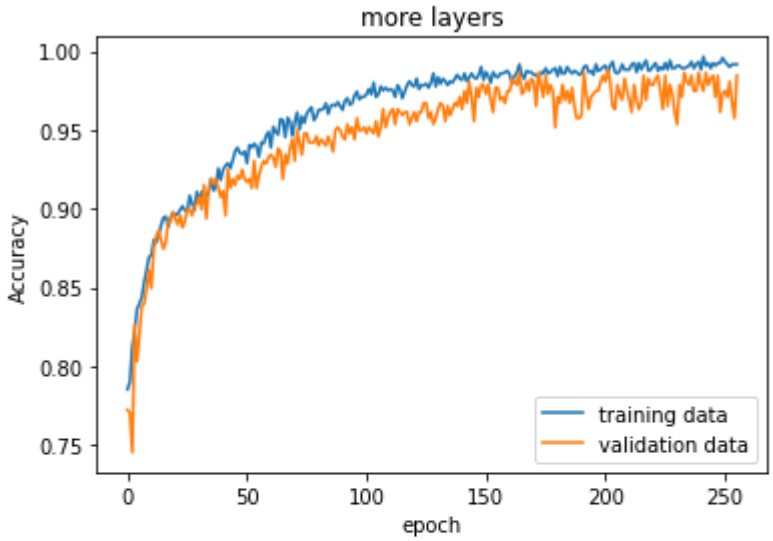

```
Training
Accuracy: 60.00%
Precision: 43.08%
Recall: 98.10%
F1-score: 0.60
Validation
Accuracy: 58.11%
Precision: 40.62%
Recall: 96.67%
F1-score: 0.57
```



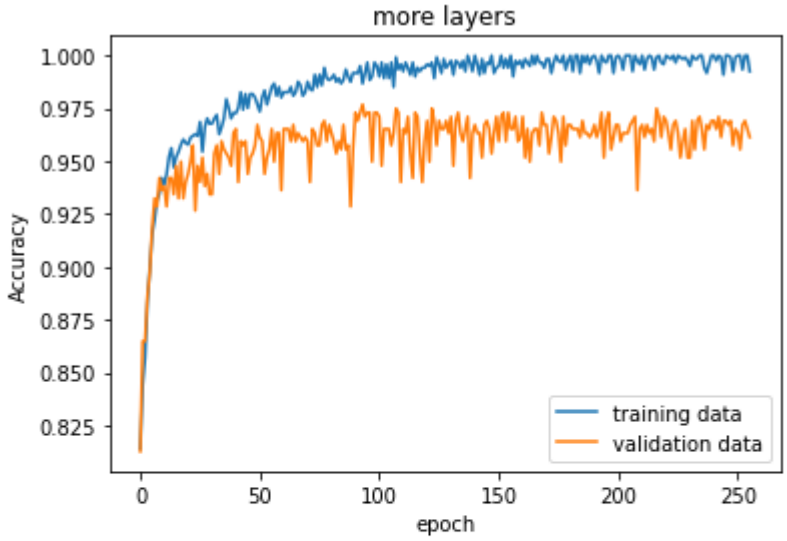

```
Training
Accuracy: 58.68%
Precision: 25.19%
Recall: 18.21%
F1-score: 0.21
Validation
Accuracy: 61.39%
Precision: 26.42%
Recall: 18.67%
F1-score: 0.22
```



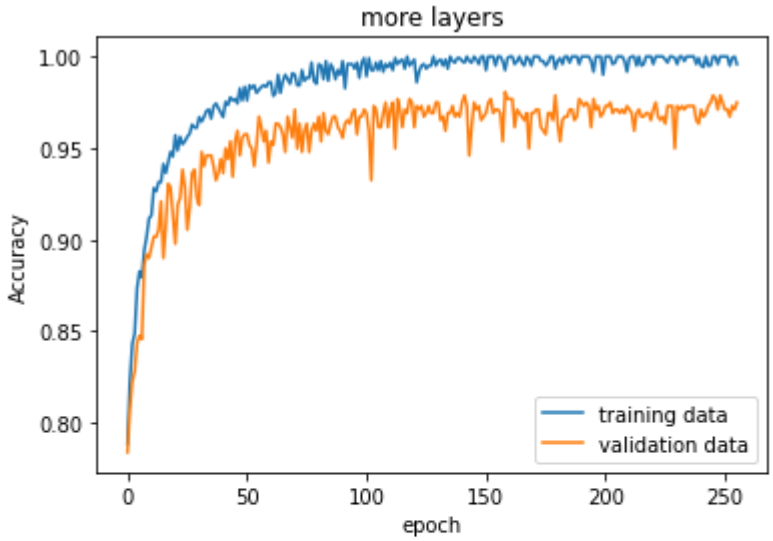

```
Training
Accuracy: 77.44%
Precision: 77.46%
Recall: 36.41%
F1-score: 0.50
Validation
Accuracy: 75.87%
Precision: 68.12%
Recall: 31.33%
F1-score: 0.43
```



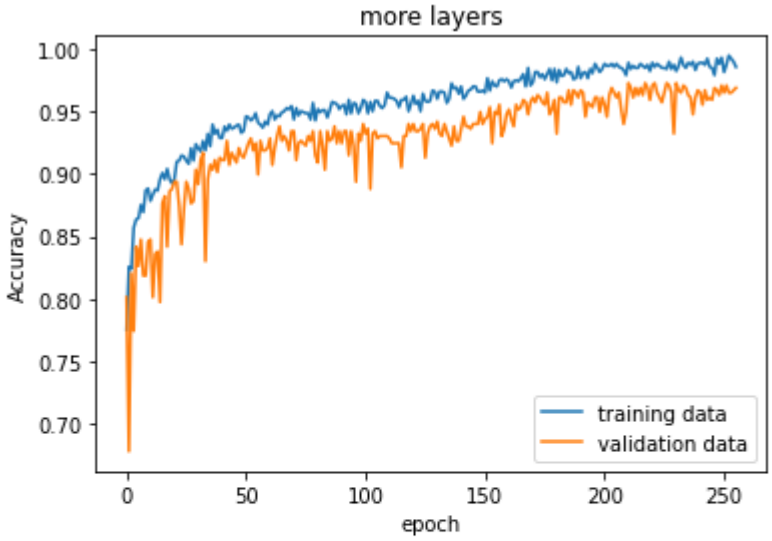

```
Training
Accuracy: 78.51%
Precision: 76.21%
Recall: 42.66%
F1-score: 0.55
Validation
Accuracy: 77.03%
Precision: 72.46%
Recall: 33.33%
F1-score: 0.46
```




```
Training
Accuracy: 74.21%
Precision: 54.73%
Recall: 88.04%
F1-score: 0.67
Validation
Accuracy: 70.85%
Precision: 49.82%
Recall: 90.67%
F1-score: 0.64
```




```
Training
Accuracy: 75.70%
Precision: 73.72%
Recall: 31.25%
F1-score: 0.44
Validation
Accuracy: 73.36%
Precision: 60.34%
Recall: 23.33%
F1-score: 0.34
```

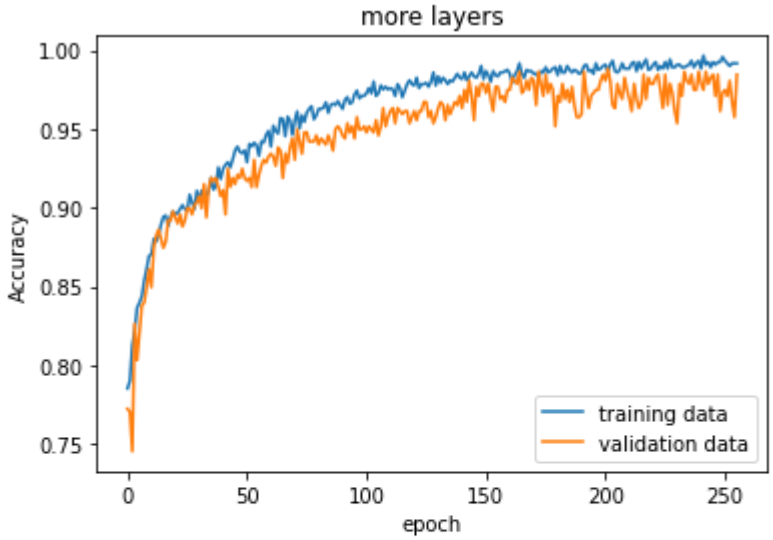



```
Training
Accuracy: 70.00%
Precision: 77.78%
Recall: 1.90%
F1-score: 0.04
Validation
Accuracy: 71.04%
Precision: 50.00%
Recall: 1.33%
F1-score: 0.03
```

6 modifying activation functions

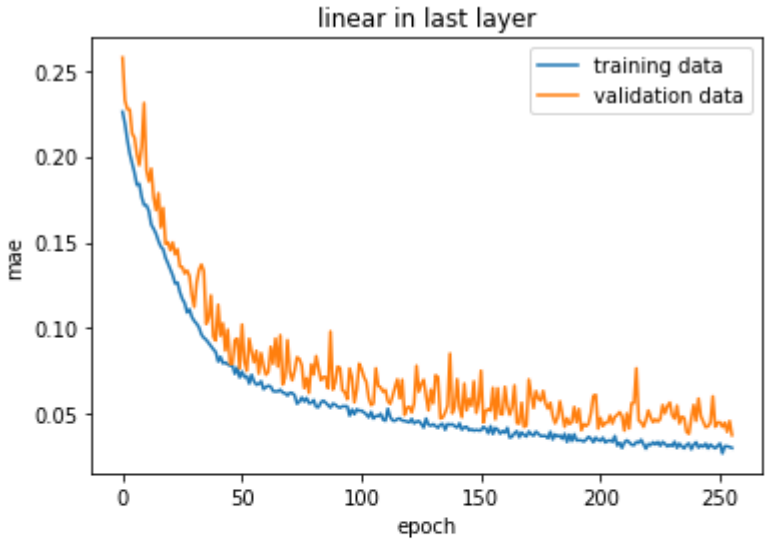
Here I ran some experiments involving activations between linear and sigmoid. This is the result. All experiments at this point is run with values from the base case. The only thing that will be changed is the activation

model	accuracy	precision	recall	f1-score
base	26.78	18.64	41.85	0.26
linear(last)	70.41	49.58	70.41	0.58
linear(all)	65.45	83.83	65.45	0.66
sigmoid(last)	41.57	33.62	100	0.5
sigmoid(all)	69.59	0	0	0



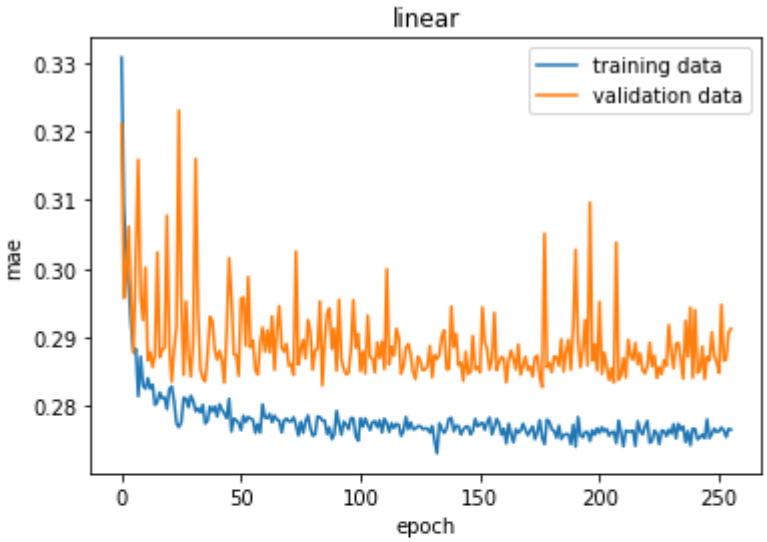

```
Training
Accuracy: 78.51%
Precision: 76.21%
Recall: 42.66%
F1-score: 0.55
Validation
Accuracy: 77.03%
Precision: 72.46%
Recall: 33.33%
F1-score: 0.46
```

This is my base case I picked out of my experiments. This resulted in the best accuracy so far so I picked it.



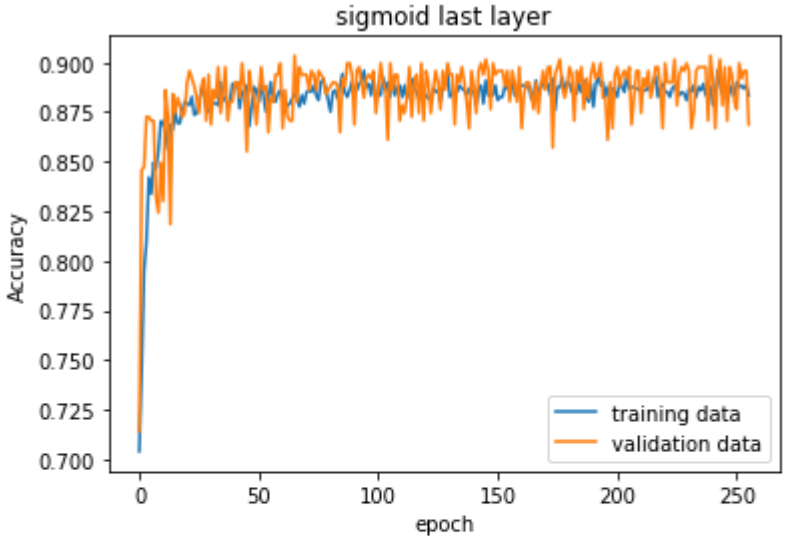

```
Training
Accuracy: 87.44%
Precision: 87.32%
Recall: 87.44%
F1-score: 0.87
Validation
Accuracy: 84.17%
Precision: 83.73%
Recall: 84.17%
F1-score: 0.84
```

With this neural network, the first 4 layers are sigmoid activation and the last is linear.



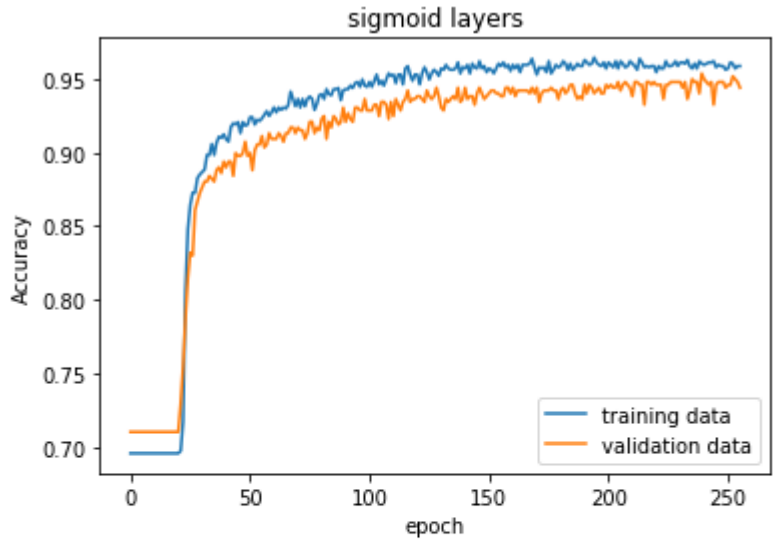

```
Training
Accuracy: 65.45%
Precision: 83.83%
Recall: 65.45%
F1-score: 0.66
Validation
Accuracy: 64.29%
Precision: 84.01%
Recall: 64.29%
F1-score: 0.65
```

With this neural network, all the layers are linear.




```
Training
Accuracy: 41.57%
Precision: 33.62%
Recall: 100.00%
F1-score: 0.50
Validation
Accuracy: 44.02%
Precision: 35.49%
Recall: 99.38%
F1-score: 0.52
```

With this neural network, the first 4 layers are linear activation and the last is sigmoid.



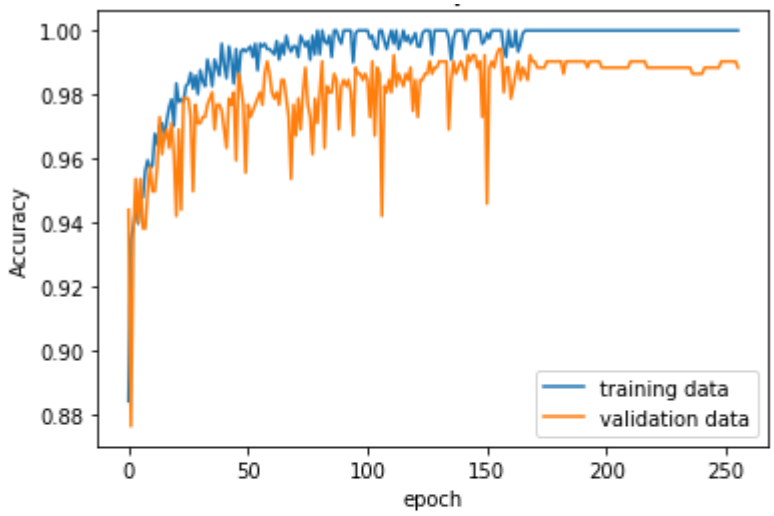

```
Training
Accuracy: 69.59%
Precision: 0.00%
Recall: 0.00%
F1-score: 0.00
Validation
Accuracy: 71.04%
Precision: 0.00%
Recall: 0.00%
F1-score: 0.00
```

With this neural network, all the layers are sigmoid.

7 Conclusion of the Neural network experimentation

Performance wise the linear networks achieved a higher accuracy than the sigmoid activations.

8 Overfit



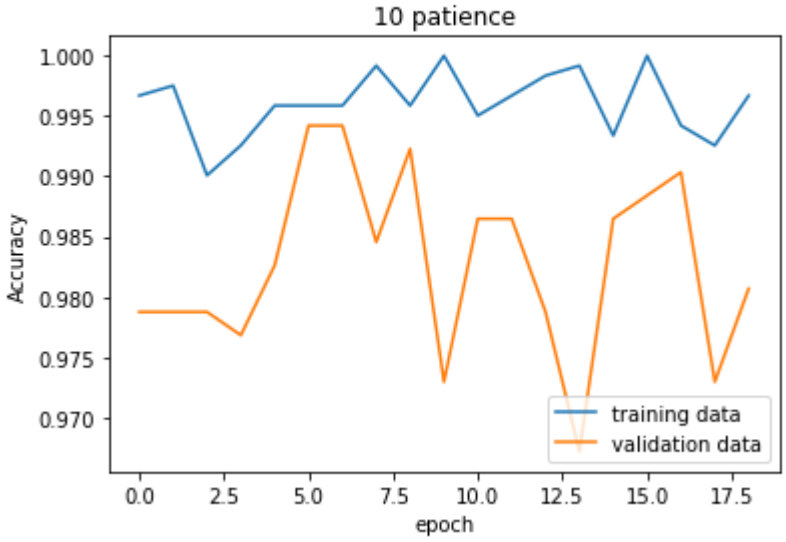

```
Training
Accuracy: 90.74%
Precision: 79.57%
Recall: 92.46%
F1-score: 0.86
Validation
Accuracy: 91.12%
Precision: 82.02%
Recall: 91.25%
F1-score: 0.86
```

The architecture that gives the best accuracy is when all layers are using relu activation except the last layer which is sigmoid activation. The neurons are 60, 120, 240, 480, 960 respectively. I increased the number of neurons in each layer by a factor of 10 to overfit my base model.

9 callbacks

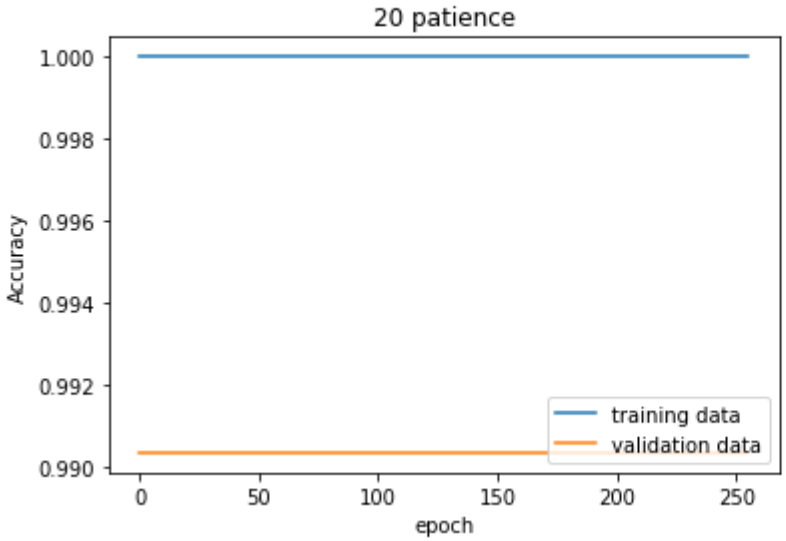
These are callbacks with 10, 20, and 100 patience.Each will have 256 epoch, and batch size 8.

```
Epoch 00019: val_loss did not improve from 0.00995
Epoch 00019: early stopping
```



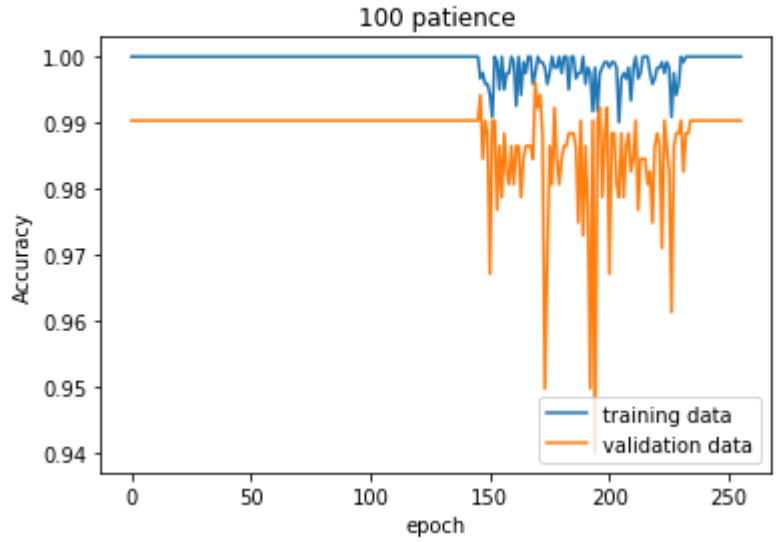
In this learning curve with 10 patience, the val loss didn't improve in about 19 epochs. The callback function stopped the training early because the val loss wasn't improving. If it has gone through 19 epochs and the val loss still hasn't improved then it would be a waste to let it continue because it wouldn't have improved anyways.

```
Epoch 00256: val_loss improved from 0.18448 to 0.18442, saving model to carData.csv
```



In this learning curve with 20 patience, the val loss improved every epoch and therefore it wasn't halted by the callback function.

```
Epoch 00256: val_loss did not improve from 0.04964
```



In this learning curve with 100 patience, the val loss improved every epoch up until the 149th epoch. After the 149th epoch, the val loss never improved until the 197th epoch and after the 197th it didn't improve again.

10 evaluation

model	accuracy	precision	recall	f1-score
base	26.78	18.64	41.85	0.26
linear(last)	70.41	49.58	70.41	0.58
linear(all)	65.45	83.83	65.45	0.66
sigmoid(last)	41.57	33.62	100	0.5
sigmoid(all)	69.59	0	0	0
overfitting	90.74	79.57	92.46	0.86

Essentially everything performed horribly except for overfitting. Not sure what is going on with all layers having sigmoid activation. I assume there were some errors involving that. The model that had all linear layers did okay. The model that had sigmoid as its last layer had a 100 in recall so the positives were identified correctly.