Perception Assignment

OBJECTIVE

Build a NN that is capable of predicting depth given simply bounding box data for any cone. Validation split is supposed to be 60:40

CUSTOM NN

Pre-processing and Normalization

The coordinates were of the format [Class,x,y,width,height,confidence] where the values of the X and Y coordinates as well as the width and height were already normalized according to the image size. The number of features used as input were tried and new features such as area of the bounding box was also added. Instead of using the coordinates of the centre of the bounding box, the coordinates were modified to use the coordinates of the top left and bottom right points of the box. That gave a better result than using just the centre [x,y] along with the width and height. The length of the diagonal of the bounding box also emerged as an important feature in the process.

Data augmentation techniques such as translating the boxes, scaling them and flipping them to increase the training data so that it generalizes well but there were issues with augmenting the depth values and the model was not able to learn well appropriately.

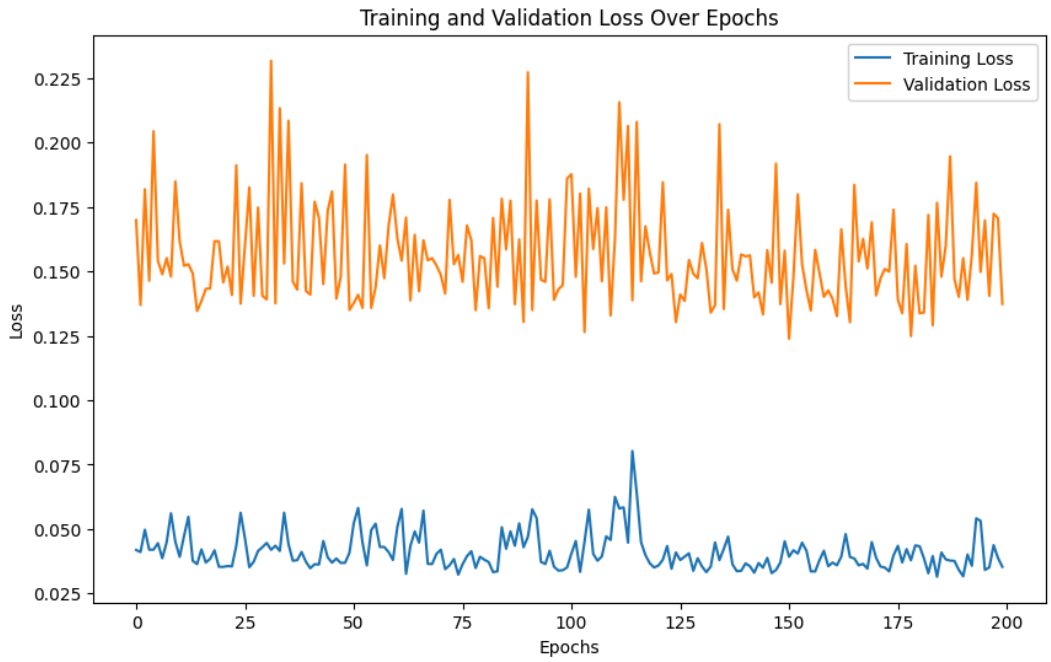
Model architechture

The model consists of linear layers with the LeakyReLU activation function. Since the output is supposed to be the depth value of the corresponding cone, the activation function was chosen accordingly. Other variants of the ReLU function were tried but after adjusting the negative slope value of the LeakyRelu function, it was found to give the best accuracy. Increasing more linear layers weakens the model's ability to generalize well.

AdamW optimizer is used for this purpose. The learning rate needs to be decayed after the first 1200-1500 epochs of training so it was adjusted manually during training. A learning rate scheduler was used initially to try to get an appropriate learning rate.

Training

In the last few epochs, the val loss saturates to around 0.175 and the training loss to 0.045

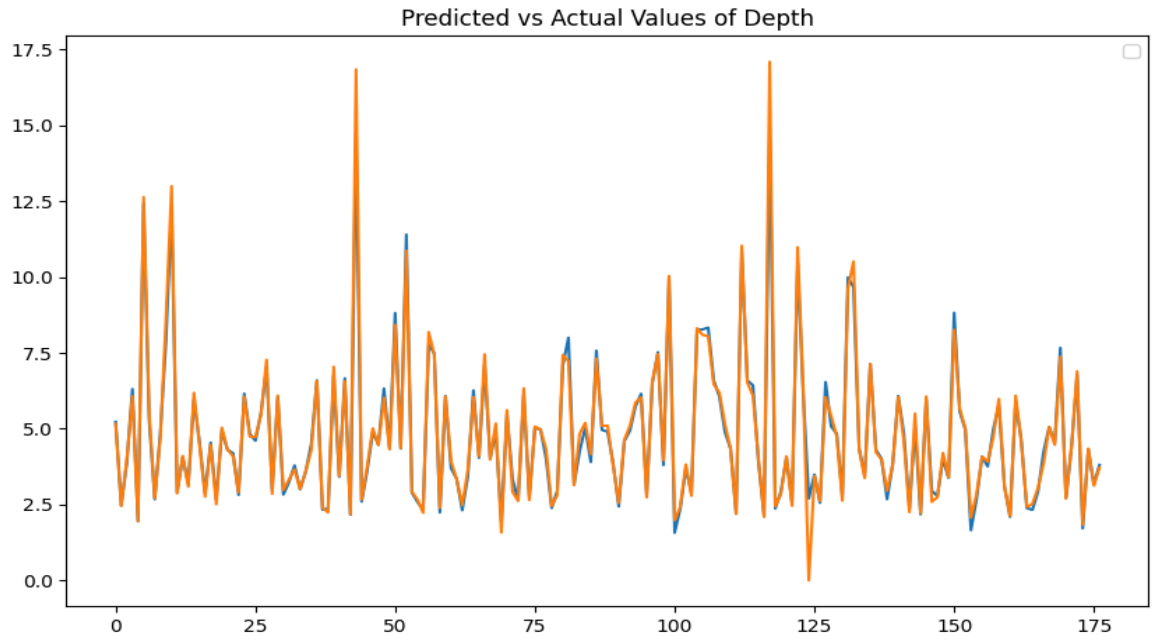


Since it is compulsory for the training : validation split to be 60:40, there is some degree of overfitting in the model. Techniques to overcome it such as regularization were tried and dropout layers are used between the linear layers.

Gradient saturation was not an issue with the activation function used and there was no need to clip the higher gradients as well.

Results

Validation MSE: 0.14839189045704937
Validation RMSE: 0.3852166798790641
Validation MAE: 0.1930382080940203
These are the results of the trained model

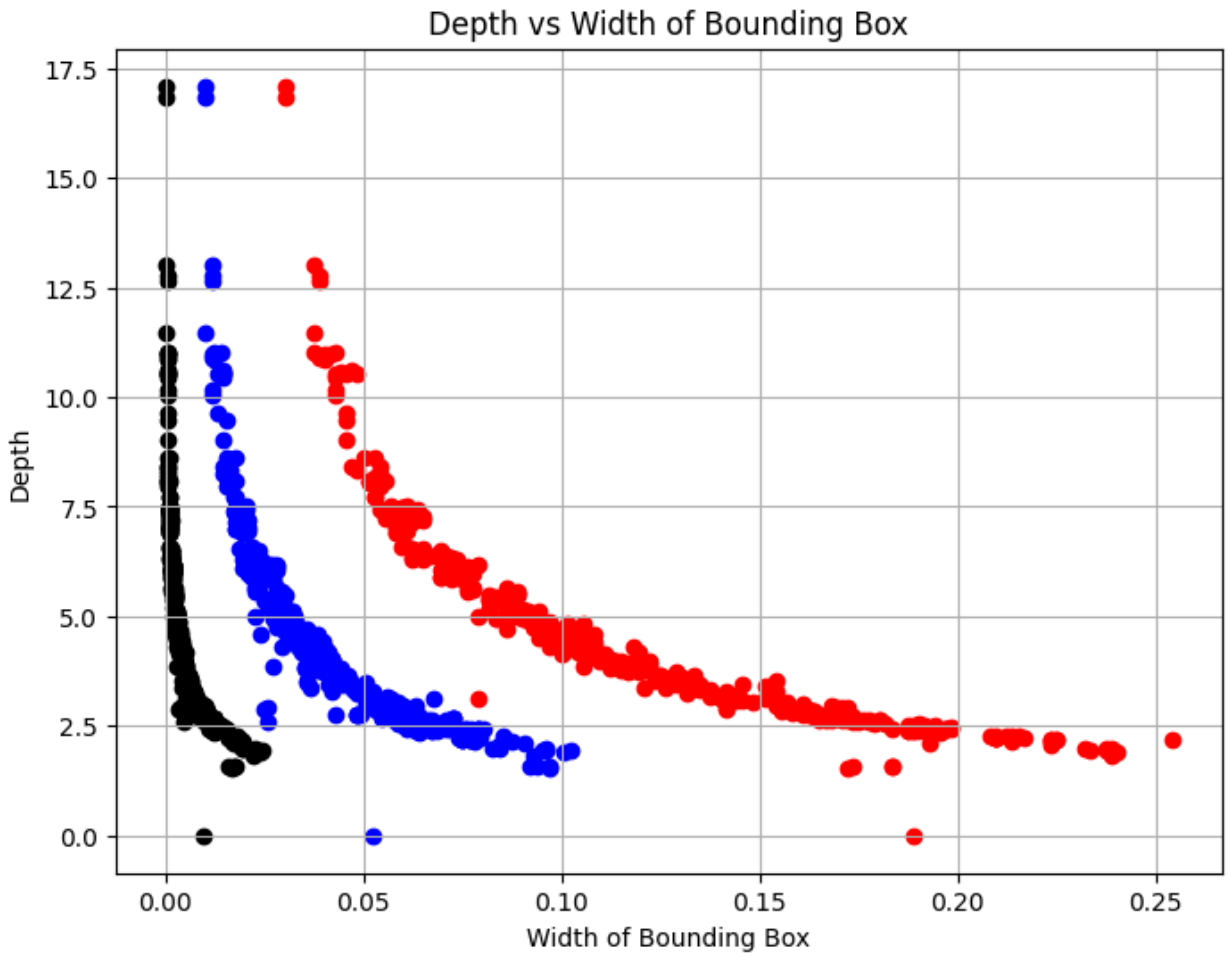


The model does a good job at predicting the depth values.

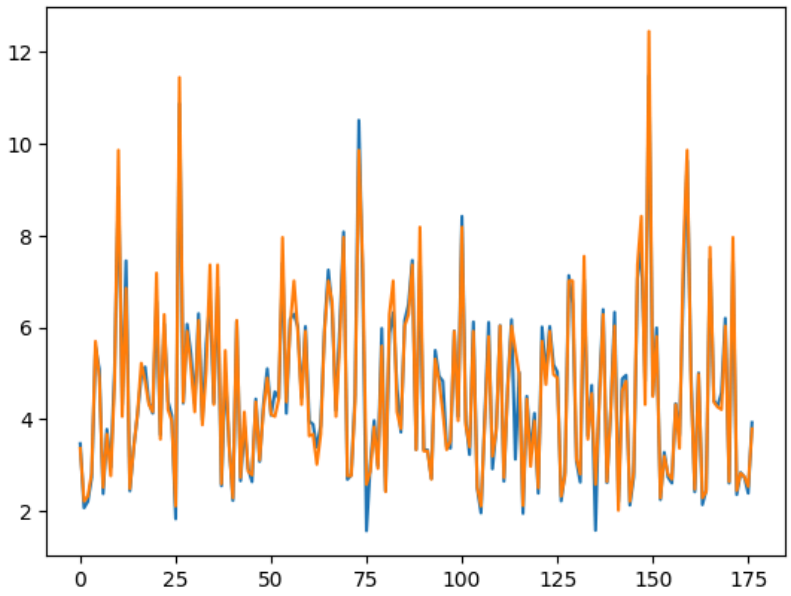
It takes the model around 51 milli-seconds to predict the depth for a single input provided (Measured on Apple Silicon M2 CPU). The time taken is quite significant and this is one of the major drawback of using a neural network for this purpose. Model optimization and boosting techniques can be used to reduce computational time and trade-off some accuracy.

Curve-fitting

The graphs of the depth with the height, width and areas were all following a inverse relation as expected and it is indicated in the graph as well. I tried to find a inverse and square-inverse curve to the plot. Using multivariate regression had issues and the model was unable properly as it requires more data and there was a lot of overfitting in the model. The features were also redundant.



After trying out fitting the curves for the width, height and areas for the depth, fitting the inverse curve with the width of the bounding box, the best result is achieved.



Plot of predicted vs actual values

The validation MSE : **`0.11978.`**

Curve fitting achieves a better result at prediction than the neural networks and the time taken is 0.23 milli-seconds (almost 90x faster than the custom neural network)