MEMBERS- Kapil Iyer, Vikram Sivashankar

TEAM- KVK

Report for MLab Project

Architecture: *What model class did you choose, and why? Did you experiment with different architectures? How did they perform?*

We experimented with a convoluted neural network with 4 convoluted layers followed by 5 linear layers for our model. This is because the images are of very high dimensions, so the filters help scale them down. Further, filters help extract local information, because they are applied to the neighborhood of every pixel. This included going from 3 to 8 to 40 to 160 to 200 channels as the hidden layer numbers; the kernel was 3x3 with padding of 1 to maintain size. In linearization, the matrix was brought from dim (200*16*16) to 1024 to 256 to 64 to 16 to 1, outputting a 1-dimensional object. We used RELU activation functions after testing out arctangent functions as well as receiving poorer performance.

Training: *How did you train your model? What losses did you use and why? What hyperparameters did you tune and what results did you see? How can you explain your choice of hyperparameters?*

We used a batch size of 150. We decided to use the MSELoss, which penalizes the squared mean absolute error of the dataset, and an Adam optimizer. The hyperparameters we tuned were the learning rate. After adjusting it a few times, we decided to settle on a learning rate of 0.01. We also added in a weight decay of 0.0001 (incorporate weights into the loss calculation to prevent overfitting).

A prominent error we ran into was the excessive use of memory while running our code. We often had our code return the error of RAM being completely used up, which led to us having to restart the entire running of the program. We later found that this was because the

size of the matrix in the convolutional neural network layers were too big, and so we modified those until storage errors were minimized.

We essentially kept changing the numbers and trained over 2500 epochs until we saw that training loss was low, then corrected for overfitting until both the training loss and validation loss were small and close together.

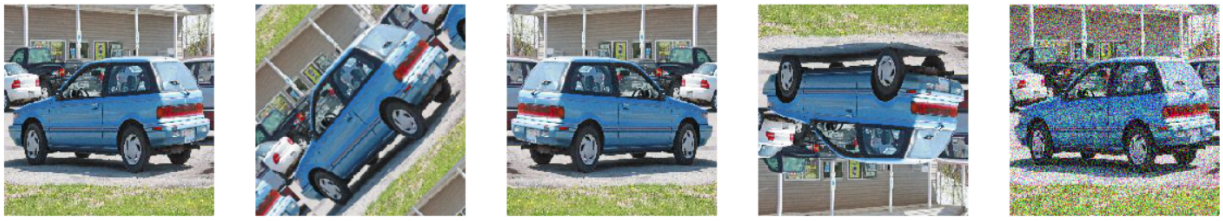
Results: *Did you validate or test your model? Explain your choice of validation and test set and describe strategies to reduce overfitting, if present. Did you conduct any error analysis? Can you hypothesize an explanation for the errors?*

Over the course of our project, we ran into reversal errors. Due to limited experience in dataframes, we ran into a few syntactical errors while extracting data from the CSV file. Further, there were several errors that were caused by the lack of the use of self. function while referring to a variable within the same class, which was later fixed.

The data set we were provided with was relatively small, with 3 convoluted layers and 3 linear layers. In order to make sure that we can minimize the error of overfitting, we decided to try and use data augmentation. This helps us enlarge the data set, as well as help generalize the data for our model. Examples of data augmentations include a 45° rotation, left-to-right flip, up-down flip and adding random noise; examples of these augmentations on a car image (the first image) are pictured below respectively from left to right.

MEMBERS- Kapil Iyer, Vikram Sivashankar

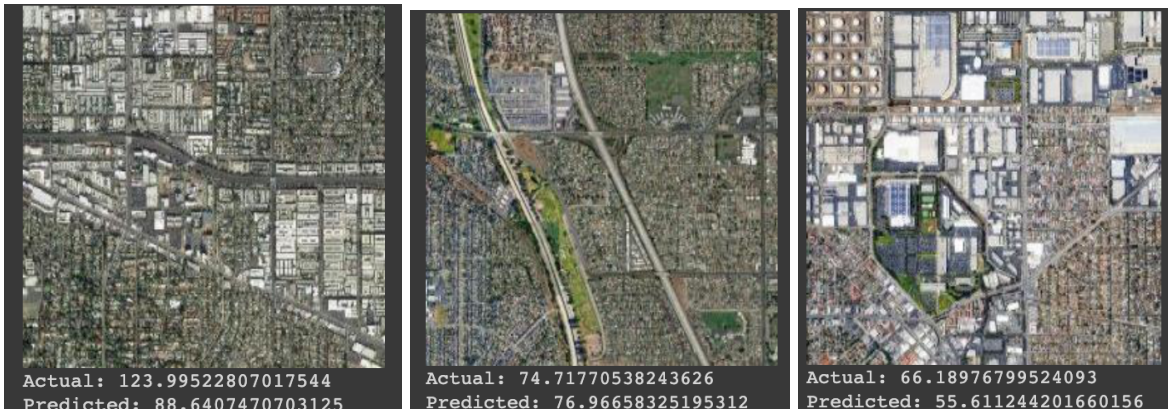
TEAM- KVK



In addition to trying out data augmentation, we used dropout layers at each step with 50% dropped out neuron. We ended up just going with the dropout layers, as this actually yielded less validation loss. This is perhaps because we did not effectively choose the best sort of transformations in the data augmentation, so we tried other augmentations until we found something which worked.

Finally, the model worked from (avg_train_loss, avg_val_loss) = (2188, 19798) to (avg_train_loss, avg_val_loss) = (3, 5) on epoch 2499 (the 2500'th epoch).

Testing out some predictions with a random selection of images, these were the results:



When we plotted loss curves on the final model, we noticed that the plot of training loss decreased to a point of stability. We also noticed that the plot of validation loss decreases to a point of stability and has a small gap with the training loss. Therefore, there was not a big overfitting nor underfitting problem. With this good fit, as well as high target accuracy, we decided to go with this model.