A0254358X

`Epoch 7, train error 0.9168, val error 1.1375. Train acc = 0.8547, val acc = 0.7848. Time cost 1.66 min.\n`

Techniques used to achieve the results:

1. Modelling using CNNs
    1. I have used 4 CNN layers combined with batchnorm and dropouts. Adding the second pair of layers improved accuracies by around 3-4%. Adding further layers only led to a very minor increase (0.5%) in validation accuracy for same number of epochs, suggesting not a whole lot of benefit of doing so.
2. Data Augmentation:
    1. Flipping images laterally - and changing the signs of corresponding angular velocity and `dlm` to create a new data point.
    2. Randomly rotating images by 10 degrees either side.
    3. Also tried cropping upper 25% of image since it doesn't contribute to the image - but it did not lead to any major change in results
    These techniques led to a 1-2% improvement in validation accuracy.
3. Regularization and normalization:
    1. Using Dropouts with probability 20% after Maxpool2d
    2. Using batchnorm to normalize inter batch variances and thus stabilize and speedup training
    Without dropout validation accuracy was around 2-3% lower.
4. Parameter values:
    1. `lr=0.001` with AdamW optimizer and `weight_decay=0.001` and chosen architecture gives the best results for a 10 epoch run
        1. reducing the LR values to 0.0009 or 0.0008 did not lead to any major improvement in final validation accuracy even when run till 20 epochs.
        2. Increasing LR value to 0.0015 or above led to worse validation accuracies suggesting that the model was not converging properly
    2. Increasing epochs increases validation accuracy till about `11` epochs, after which is start falling slightly, indicating overfitting.
    3. Lowering the batch size to `32` improved the validation accuracy slightly (from ~77% to 79%)

On covariate shift in behavior cloning

The core reason for this is that the training data distribution is different from the real world distribution - which is always going to be the case for finite amount of data. When the model encounters something out of distribution at test time, it often results in bizzarre "unexpected" actions and therefore lower accuracy on test.

The first step in battling covariate shift is to get a highly representative sample - that in imitation learning would mean that the expert demonstrates all sorts of situations along with the actions to be taken in them.

Another way is to bridge the gaps in the distribution of training data through generative models or using a simulator - or preventing learning those gaps by using proper regularization.