Report

Roll Number: 2022101027

Name: Debangan Mishra

Task Description

The given task is such that we are given an input image, and we are predicting the sum of the numbers represented in the image. The images are single channel images made by collating 4 MNIST images into one.

Regressor Modelling

If there is no guarantee as to whether the test set contains exactly 4 numbers, the typical classifier approach will likely fail. As a result, we can modify the network to act as a regressor with a single output and the output value being rounded to get an integer representation. Hence, during training, the model will use MSE Loss (typically used for regression tasks) and will round the number to closest integer during inference to calculate accuracy.

Note that as a result of this approach, a low MSE Loss need not translate to a good accuracy. Consider for example, our model predicts 24.6 as an output while the ground truth is 24. In this case, the MSE Loss will be low, but the accuracy need not be as rounding changes the output. Also note that it is not possible to train the model based on rounded outputs as rounding is a step function with zero gradient almost everywhere making it useless for learning.

Classifier Modelling

When we are assuming that the input image will always have 4 numbers as that is the only value appearing in the dataset. In that case, we can model the network as a classifier with the outputs ranging from 0 to 36. For this case, we can use the cross entropy loss which is commonly used for classification task. Empirically we observe that the model performance on the validation split is lower for the classifier than it is for the regressor. This might be because regression operates in the continuous space (just like the input) and this possibly allows for a smoother optimization. It is not possible to verify why the regressor works better than the classifier case.

However, the same behaviour was observed across seeds (0, 10, 42), indicating a consistent behaviour. Also note that early stopping based on validation loss has not been implemented for the models (even though the validation accuracy is higher at some early stages) to ensure the train loss is sufficiently low. Training for further epochs leads to a decrease in the validation accuracy indicating overfitting. Hence, the number of epochs was chosen to be 15. Note that the training accuracy is very high for the classifier approach, but the validation is not, indicating a possibility that this approach is overfitting.

Transfer Learning With VGG

Owing to the success of the regressor approach, a third type of network was created using transfer learning using VGG (a popular CNN network which achieves fairly high accuracy across various tasks). The convolutional backbone and the average pooling layers of the network are frozen and a classifier is added on top which has one output as the previous CNN regressor case. Note that a convolutional layer is also prepended to ensure that the MNIST dataset with one channel is compatible with the VGG network that is trained on 3 channel RGB images. This approach is computationally more expensive, but was tried out to see whether the pretraining of the VGG leads to better results.

It does not provide any particularly significant advantages in the results, but only comes with a much greater overhead in compute. This is not particularly feasible for our scenario.

Some Observations

1. It is seen that the current training approach leads to a possible overfitting of the classifier approach as it has a very high train accuracy, but low validation accuracy.
2. None of the models perform very well. This was expected as a simple CNN network will not be able to see the semantic information between multiple numbers being displayed on the image and the sum output. An object recognition network capable of isolating individual objects, or a sequence-to-sequence approach which analyzes chunks of the images in a logical sequence will perform better.

Running Instructions

1. The trained models cannot be pushed to GitHub due to their high size. The Training code will generate the files and the Inference code will load from the model. The trained files will be provided later as they take a lot of time to generate.
2. Take the data and label files provided in the link of the course project and place them in a folder called "data". The training and inference code should be take care of the remaining.