Deep Learning for Computer Vision (DA621): Assignment-2

Name: Pradnesh Prasad Kalkar
Roll: 190101103

Note: For reproducing the results of the code, please upload the notebook to google colab and before running the notebook, please **'change runtime type' to GPU.** The code is written primarily using PyTorch.

Notebook Summary, Observations and Results:

Q1.

1. The Caltech101 dataset images were downloaded from torchvision datasets. After the split, the training set contains 5552 images, validation set contains 1389 images and the test set contains 1736 images.
2. In this question, I used VGG16 model pretrained on ILSVRC for transfer learning. For achieving this, I first changed the classification head to predict 101 class logits instead of 1000 (from ImageNet). This layer was vgg16_model.classifier[6] in the code. I just trained this layer and froze the parameters of all the other layers (so as to avoid destroying any of the information they contain during future training rounds). The training in this step was done using an Adam optimizer with initial learning rate of 0.005 with an exponential learning rate scheduler and for 6 epochs. The model corresponding to the best validation accuracy (91.79%) was obtained with training for 3min 40sec.
3. After this step, I decided to unfreeze all the fully connected layers and the last CONV layer in the model; and then finetune (train) the model on a very low learning rate to improve the validation accuracy. This can potentially achieve meaningful improvements, by incrementally adapting the pretrained features to the new data. It will force the weights to be tuned from generic feature maps to features associated specifically with the dataset. Also, the reason for selecting just the last few layers for finetuning is that the initial layers (which learn the low level features) are very generic to vision tasks and so dont need to be retrained. Whereas, the later layers (which learn the high level features) become more specialised towards the dataset/task at hand. Thus, the goal of fine-tuning here is to adapt these specialized features to work with the new dataset, rather than overwrite the generic learning. Finetuning was done for 4 epochs using Adam optimizer with a fixed learning rate of 5e-5 (very low). After this step, the best validation accuracy was observed to be (92.87%) which is an improvement over the previous one. This step required 2min 51sec.

4. In the second part of the question, I used the VGG16 model with randomly initialised weights. And as it is obvious, it was necessary to train the whole model (all layers) this time since the model had no previous knowledge. So, the model was trained with the changed classifier head as done previously under similar training settings for 5 epochs. The training required much more time (7min 3sec) since all the parameters were being trained. Also, the model performance was very poor - the best validation accuracy came out to be 9.58%.
5. On the test set, the trained model with transfer learning obtained an accuracy of 91.42% and the trained model with random initialization obtained an accuracy of 7.83%. This clearly shows the advantage of transfer learning as compared to training from scratch especially when the dataset size is less.
6. To summarise, the performance of the model with transfer learning was much better than the one trained with random initialisation. Also, the transfer learning model required fewer iterations to converge since it already has some knowledge from a similar task (ILSVRC). The model with random initialisation will take a long time to converge. The design choices are discussed in points 2 and 3 above.

Q2.

1. For creating the dataset, sequences were randomly created using uniform distribution with variable sequence length (from min_seq_len of 2 to max_seq_len of 10). A total of 10000 sequences were created with 6400 for training, 1600 for validation and 2000 for testing.
2. All the models were trained for 3 epochs (around 6400*3 = 19200 iterations) using SGD since the sequences were of different lengths. So, a batch size of 1 was used. Since this is a regression task, MSE loss was employed. All the models were trained using Adam optimizer with a fixed learning rate of 0.003 and the loss was examined every 500 iterations. LSTM and GRU took more time to train as compared to Elman network due to the complexity involved in them. Models with weights corresponding to the best (minimum) validation losses were obtained.
3. The performance of these models were compared using two popular metrics used for regression models - MSE (Mean Sequared Error) and MAE (Mean Absolute Error). As per the observed metrics on the test set, the GRU and LSTM were found to be almost equal with GRU slightly better than LSTM. Both the GRU and LSTM were much better than the Elman network. And the baseline performance was very very worse than all the 3 which was expected as it had no intelligence involved.
