CV ASSIGNMENT 5

Feature Extraction:

In this section, I have performed the following steps:

1. First I loaded the file 'videos_labels_subsets.txt' and separated the first 3360 folders into train and test based on the third column in the text file.
2. Then I called the VGG pretrained network and got the first two layers with input features as 25088 and output features as 4096.
3. Since the video data was very huge I separated my feature extraction process into batches of 500 videos.
4. In each step, I read the images within each video folder one at a time and saved them into a list.
5. Then I cropped that image from the center and at the four corners to get 5 images corresponding to one input image and then I normalized the images using torchvision.transforms.
6. Then I passed these five images into the pretrained vgg model to get a feature and then I computed the mean for these five features to get a single feature from these five features corresponding to an image within a folder.
7. I appended these features to a list and then wrote this list to a pickle file which stores the features as a list of tensors.

I performed the above steps for 2049 train folders and 951 test folders and then got a final pickle file for train and test data.

Modelling:

Long Short Term Memory networks are used to handle long term dependencies in the input data with the use of gates which modulate the amount of data flowing through the network. The input dimensions that I gave to the LSTM was [4,25,4096] for the training data and [4,1] for the train labels. I have taken 4 as the batch size here.
I used reshape, view and torch.stack for reshaping the tensors into the desired dimension.

I designed the LSTM network by referring to the tutorial mentioned in the python notebook and I have a simple LSTM with an lstm layer and a linear layer.

I tried two configurations of the lstm network:

1. Hidden dimension=256
   The output I got was
   ```
   Epoch: 1 Loss: 3.217
   Epoch: 2 Loss: 3.125
   Epoch: 3 Loss: 3.020
   ```

```
Epoch: 4 Loss: 2.894
Epoch: 5 Loss: 2.750
Epoch: 33 Loss: 0.306
Epoch: 34 Loss: 0.240
Epoch: 35 Loss: 0.168
Epoch: 36 Loss: 0.111
Epoch: 37 Loss: 0.074
Training done in 15.106865974267324 minutes
Accuracy on the train dataset is: 99.75093399750934
Accuracy on the test dataset is: 79.81072555205047
```

2. Hidden dimension=512
   The output was:

```
Epoch: 1 Loss: 3.203
Epoch: 2 Loss: 3.084
Epoch: 3 Loss: 2.941
Epoch: 4 Loss: 2.775
Epoch: 5 Loss: 2.599
Epoch: 32 Loss: 0.452
Epoch: 33 Loss: 0.403
Epoch: 34 Loss: 0.348
Epoch: 35 Loss: 0.296
Epoch: 36 Loss: 0.261
Epoch: 37 Loss: 0.191
Training done in 36.734095601240796 minutes

Accuracy on the train dataset is: 96.13947696139476
Accuracy on the test dataset is: 78.44374342797056
```

I trained both the networks for 37 epochs. We can see that the LSTM with 256 hidden layers performs better than the model with 512 layers and does so in almost half the time.

I could not compare this accuracy with that of SVM due to some shape conversion issues but I believe that SVM should perform better than LSTM in this problem domain as Neural networks are prone to overfitting under some circumstances whereas there is no such vanishing gradient and overfitting problems in the case of SVM.

References:

- Tutorials mentioned in the .ipynb notebook
- Evaluation code reference taken from the previous homework
- VGG Model reference taken from the previous homework
- https://medium.com/datathings/the-magic-of-lstm-neural-networks-6775e8b540cd