HW5 PROJECT REPORT

OBJECTIVE:

The objective of this is learning Recurrent Neural Network(RNN) and in particular LSTM. We were suppose to do action recognition using LSTM on a dataset called UCF101.

INTRODUCTION:

Neural networks have a drawback that it cannot store previous information .For example, if you need some information from a step 2 of an algorithm you have to perform the operation from scratch as neural network does not store this information for you.
Recurrent neural networks were designed to resolve this issue.These networks perform with loops which allows information to persist.But RNN works only when you need recent information .

LSTM is a special case of RNN.In LSTM, there is a corresponding hidden state which can contain information from various points earlier in the sequence.

DATASET:

Dataset is called UCF101 which consists of 101 actions/classes and for each action there are 145 samples.Each sample is a video which is sampled to 25 images from each video.And then these samples are divided into test and training sets.

IMPLEMENTATION:

We divided this into 2 steps:
1. Feature Extraction
2. Modelling

1. Feature Extraction

Following are the steps of feature extraction:
We are doing action recognition for only 25 classes .

Dividing into training and test sets:
a. Read the data from the annos text file and store it into a list. The annos file contains the name of the folder, labels and subset.
b. Now divide the data(folder name and tables) into training and testing samples based on the subset value , i.e. if subset =1 put it into the training set and otherwise into the testing set.

Pretrained network

We’ll use vgg as our pretrained network to extract the features.
The pretrained network has the following features:

```
Sequential(
  (0): Linear(in_features=25088, out_features=4096, bias=True)
)
```

Extracting features from vgg:

1. First step is to normalize each of the images using transforms.Normalize
2. Now all the images are of the size 256x340 but vgg takes in nxn as input.So instead of resizing we divided the image into 5 nxn image and take the average of the features of these 5 images as out final feature.Now once we get the features we append the features and the labels in 2 lists.
3. Now I stacked the every 25 images into another list and then divided it into batch size of 8.

So the shape of the one feature was [8,25,4096]
4. Similary I divided the labels into a batch_size of 8 with shape of one tensor as [8,1].
5. Similarly all the steps are performed for test data.
6. And all these features and labels are stored into pickle files in order to reduce the work off extracting it every time .

2. Modelling

Firstly the shape of the features is changes as is needed by the LSTM as input.
So the final shape of one tensor of features is [25,8,4096]
And that of label is [8].

Then these features and labels are given as input to the lstm.

LSTM NETWORK:

<u>LSTM model 1</u>

```
LSTM(
  (lstm): LSTM(4096, 256)
  (hiddenlabel): Linear(in_features=256, out_features=25, bias=True)
)
```

Number of hidden layers:1
Number of hidden dimensions: 256
Num_classes:25
Epochs:35
Loss:0.160
Optimizer:SGD
Training accuracy: 99.79%

Testing accuracy:77.49%

LSTM model 2:

```
LSTM(
  (lstm): LSTM(4096, 256)
  (hiddenlabel): Linear(in_features=256, out_features=25, bias=True)
)
```

Number of hidden layers:1
Number of hidden dimensions: 512
Num_classes:25
Epochs:35
Loss:0.291
Optimizer:SGD
Training accuracy: 95.3%
Testing accuracy:78.5%

EVALUATION:

The same features and labels are also passed through a SVM to compare the accuracy of LSTM.

Training accuracy of SVM : 100%
Testing accuracy of SVM: 86.11%

So the accuracy of SVM is better than that of both my LSTM models.

REFERENCES:

1. http://colah.github.io/posts/2015-08-Understanding-LSTMs/
2. https://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html#sphx-glr-beginner-nlp-sequence-models-tutorial-py