Project Report on Breakfast Dataset Action Classification

The following report outlines the approach followed by me on Breakfast dataset action classification and also the subsequent analysis on the same. The report also states some possible reasons for the said results and hence analyzes the result section.

Main Approach

The main architecture of the model that is followed in our experiments can be seen in `model.py` as the `new_model` class. Also, the workflow for the same can be seen in `train.py` file. **Method:** 1. All the frame features in a video are passed through the `encoder_bilstm` to encode each frame feature to have more better representation. After this, these new features are passed as a stack of features through `segment_bilstm`, with each stack belonging to a particular sub-action segment. 2. Once the segment features are obtained, a `max_pool` operation is conducted on top of these to get a feature vector per segment which is further passed through fully connected layers for final prediction of the sub-action

Some of the hyperparameters pertaining to the main architecture are as follows: - `n_layers` : number of layers in the two lstms used (`encoder_bilstm`, `segment_bilstm`) - `hidden_dim` : dimension for the hidden states in `encoder_bilstm` - `segment_hidden_dim` : dimension for the hidden states in `segment_bilstm` - `epochs` : number of epochs for training the model (which set to 20 because of the limited amount of gpu available and time constraints)

Apart from this, in further experiments for modifying the main network we have also used `GRUs` in place of `LSTM` in both encoder and segment LSTMs. Also for initializing the hidden states in both `encoder_bilstm` and `segment_bilstm`, we have used two different ways, namely, initializing them as zero vectors and initializing them using orthogonal weights (which was done because of some previous works also doing the same).

Results

The following table shows the test accuracy of the model with different hidden state initialization of `encoder_bilstm` and `segment_bilstm`. Correponding to these, the results are shown for different hidden state dimensions. *(here abbreviations: `hidden_dim = h_dim`, `segment_hidden_dim = sh_dim`)*

Table 1: For when `n_layers = 1`

Hidden Layer Initialization	h_dim= 160, sh_dim= 100	h_dim= 150, sh_dim= 100	h_dim= 130, sh_dim= 100	h_dim= 160, sh_dim= 90	h_dim= 150, sh_dim= 90	h_dim= 160, sh_dim= 110	h_dim= 150, sh_dim= 110
`Zero Initialization`	34.43	34.12	33.81	34.67	34.92	33.61	33.32
`Orthogonal Initialization`	32.23	32.97	33.12	34.22	**35.59**	32.79	33.15

Table 2: For when `n_layers = 2` (Not all ablations from Table 1 were launched because of heavier computational requirement and more time taken in its training)

Hidden Layer Initialization	h_dim= 160, sh_dim= 100	h_dim= 150, sh_dim= 90	h_dim= 160, sh_dim= 90
`Zero Initialization`	37.22	37.12	37.25
`Orthogonal Initialization`	38.86	**39.23**	38.81

For `n_layers = 3` , a model with `h_dim = 160` and `sh_dim = 100` with `zero_initialization`, however it underperformed severely with an accuracy of around *20%* on the test dataset.

From this, we see that, we can clearly see that the best performing model comes when: `n_layers:2`, `h_dim=150`, `sh_dim=90`.

Also as can be seen from the architecture in the file `model.py`, the `ClassPredictor` also has two hyperparameters as the dimension of `hidden_1` and `hidden_2` fc layers, which are set to 120 and 80, respectively. Some other values for these particular hyperparameters were also considered like ((110, 100), (60, 70)) but it showed worse `validation_performance` than the above-stated models and hence were discarded and not trained on test dataset.

Apart from this, we would also like to state that all of these models(present in Table 1 and Table 2) used `AdaGrad` optimizer during training

The above line is stated with much importance because a peculiar thing noticed during training the above models was the lack of their performance when using other optimizers such as `SGD with momentum` and `Adam`. The test performance shown by `SGD` and `Adam` were respectively *22%* and *14%* as compared to `AdaGrad` performance of *34.43%*, with all experiments conducted on same settings. The adaptive gradient capability of `AdaGrad` seemingly worked miles better than other optimizers in this particular problem statement.

Below are also some of the results for the case when we use `GRU` model instead of `LSTM` in this setting:

Table 3: For when `n_layers = 1`

Hidden Layer Initialization	h_dim= 160, sh_dim= 100	h_dim= 150, sh_dim= 90
`Zero Initialization`	35.17	34.87
`Orthogonal Initialization`	35.43	**35.83**

Table 4: For when `n_layers = 2`

Hidden Layer Initialization	h_dim= 160, sh_dim= 100	h_dim= 150, sh_dim= 90
`Zero Initialization`	36.75	37.69
`Orthogonal Initialization`	37.86	**38.24**

Hence we see `GRU` model giving a considerably close performance as to that of `LSTM` model and hence serves as a viable replacement of the same.

Alternative Approach

This was another approach that was used for solving the problem and the workflow for the same can be seen in `alternate_train.py` . The model architecture can be seen `model.py` as the `BiLSTM_model` and `BiGRU_model` class. **Method:** 1. In this, we have an already created dataset for `segment_features` (stacked I3D frame features) along with their sub-action class label, as different files. Hence we consider these stacks individually rather than consider one video individually in the previous method. 2. We create a batch of such `segment_features` and pad them to pass them through `segment_bilstm` . On the outputs of these, a `max_pool` operation is conducted to get a feature vector per segment, which is further passed through fc layers for final sub-action classification.

However, the performance on these was considerably lower as compared to the Main Approach in the same settings. It showed performances such as *14.5%* as compared to main approach's performance of *33.23%* in similar settings.

Hence we can clearly find this particular approach to be detrimental to the model's performance.