Video actions classification

Chen Tingting (A0198820W), Chu Yuan (A0119483Y),
Hao Weining (A0198884B), Li Xingxuan (A0198935H)

1. Data Background and Initial Thoughts

This dataset is created by Serre lab from Brown University. Each of the 52 participants conducted 10 actions related to breakfast preparation in 18 different kitchens, for each session, it was recorded by a number of cameras (3 to 5) from different positions. Unlike most existing datasets, those participants conducted those sessions solely by following the receipt without any rehearsal or direction.

The 10 actions are: coffee, orange juice, chocolate milk, tea, bowl of cereals, fried eggs, pancakes, fruit salad, sandwich, scrambled eggs. Each action contains a few subactions, for example, in a coffee preparation action session as shown below, 5 sub-actions were conducted by the participant.

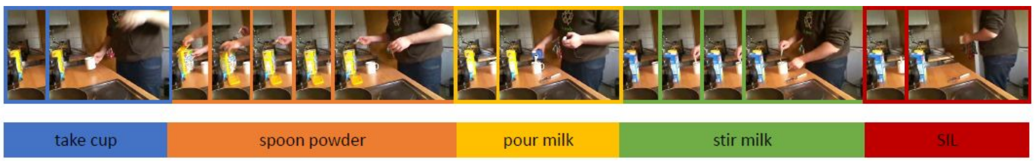


Fig 1: example of sub-actions

There are some common sequences among the subactions. For example, after cutting oranges, the participants will always squeeze oranges; after taking a knife, the participants will either cut oranges, cut fruit or cut bun. Therefore, our model should train on the whole video sesion to keep the sequence information, instead of separating them into subaction sessions as different inputs.

The video features of this project are obtained from the I3D model pre-trained on Kinetics dataset. Based on the scientific proof by Joao Carreira and Andrew Zisserman, similar to ImageNet challenges, pre-training an action classification network on a sufficiently large dataset, will also give a similar boost in performance when applied to a different temporal task or dataset.

I3D gives the best performance boost for action classification network, and its architect design is listed below:

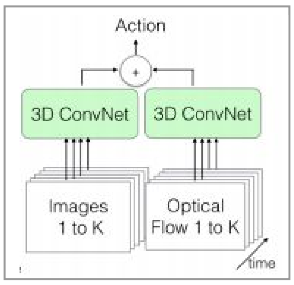


Fig 2: One I3D network trained on RGB inputs, and another on flow inputs which carry optimized, smooth flow information. The two networks are trained separately and averaged their predictions during the test stage.

2. Data Processing

i. Using raw train data and test data

Each video has multiple segments with different numbers of frames, and the number of frames varies a lot, for example in one particular breakfast video, the segment frame length can vary from less than 100 to more than 2000. We assume that the number of frames has a linear correlation with the actual preparation time that one sub-action segment takes. Keeping the data as it is could utilize 100% of the features and information within the segments. However, this will also lead to more training time.

For the same action type, the number of frames could vary a lot in different videos. This might be due to people's different cooking style, the food prepared, and preparation quantity. In order to make sure that this variance in the number of frames will not affect the model too much, we introduced another data transformation method as below, to control the number of frames for each segment in a fixed range.

ii. Using similar sequence length

For each of the segments, select certain frames with equal intervals in between. This method could make sure that all the segments are with a similar number of frames. We assume this transformation will not lose too much training information since it picks the frames across the whole video segment, the main actions within one segment should be kept. It could be similar as playing the video with 2x speed. Sometimes when the original number of frames is too large compared to the number of frames we picked (e.g. from 2000 frames we only picked 60 frames for model training, the play speed of the video is as 30x), the actions might not look as fluent as it was originally. A further improvement could be that when the number of frames exceed a certain threshold, we fix the play speed and take more frames for that segment with particularly long time length.

This method helps reduce the training time since it reduces the input data size. And picking a certain number of frames serves as a method of feature selection. The accuracy seems to improve faster than the model with raw data input. Our assumption here is that dropping certain frames could help reduce the data size without losing too much useful information, so that the model optimizes at a faster rate.

iii. Adding auxiliary frames

To further utilize the known information, we also added one transformation to include the statistics tensors in each segment. Here we tried with the maximum, minimum and mean values of all the frames within one segment. For each segment, in the end we added 3 tensors of size 400. Our assumption is that the transformation could summarize the main features of one segment and train the model in a more efficient manner.

3. Training and hyperparameter fine-tuning

i. Model Structure

Since the dataset for this project are videos that consist of frames in sequence, it is ideal to start with a RNN model to capture the temporal information. Then we applied fully connected layers to the output of the RNN to generate the predictions for each frame. In order to capture the temporal information better, we decided to use bi-directional LSTM as our RNN part.

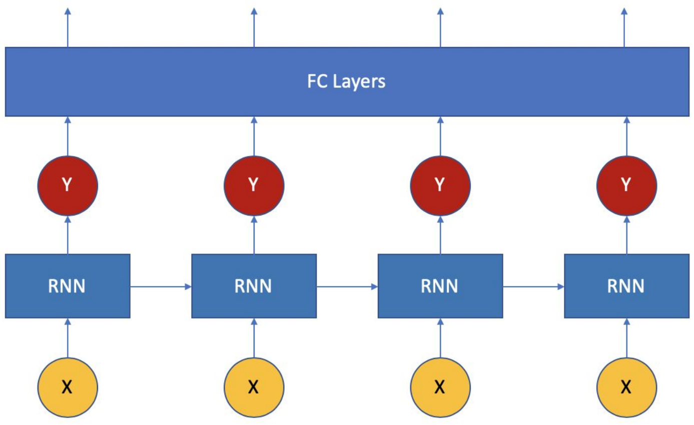


Fig 3: structure of proposed model

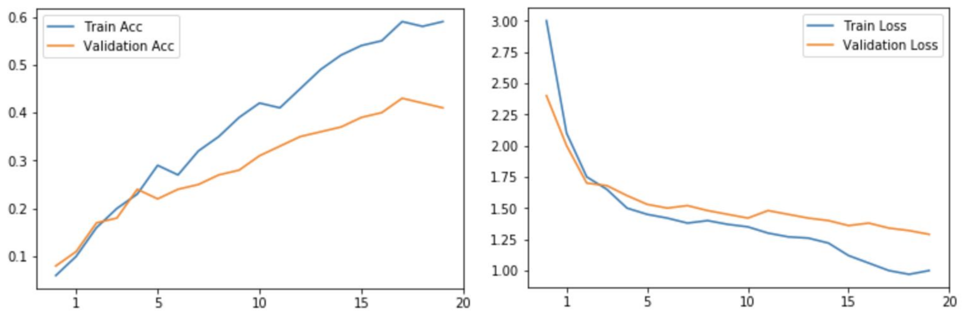
ii. Performance evaluation and fine-tuning

1) Model-level fine-tuning

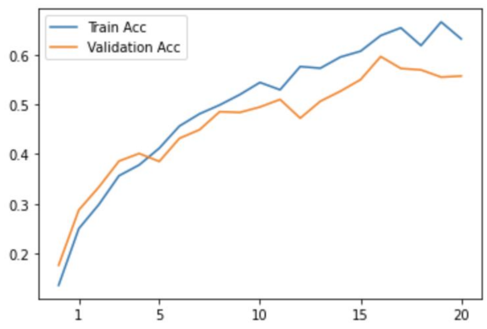
For model level fine-tuning, since we used Bi-LSTM followed by fully connected layers, several techniques were implemented to fine-tune the model, for example, number of Bi-LSTM layers, number and size of fully connected layers, normalization of input of each layer, dropout, etc.

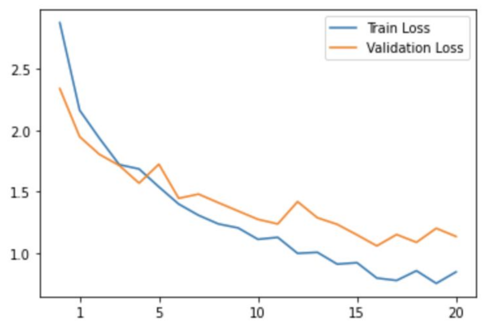
We first trained our model with the entire video sequence as inputs to the RNN model:

a. 4 layers of Bi-LSTM + 256 FC layer + 128 FC layer. After trying this model, we realized that it tends to overfit before actually learning the sequential information between frames. And the final testing result of this model is only 40.3%. (refer to the figure below)

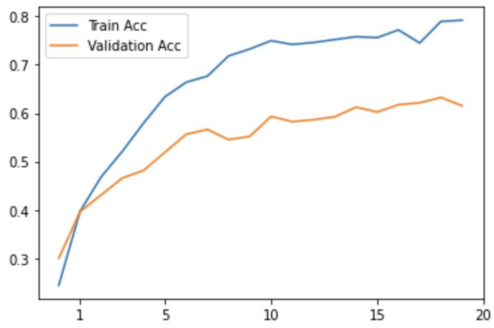


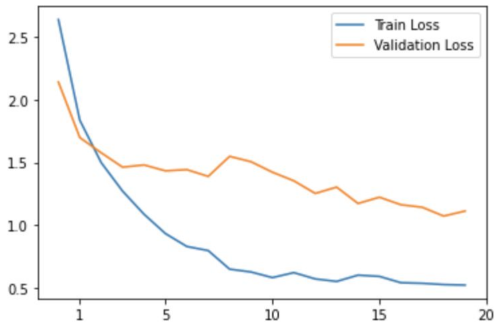
b. 3 layers of Bi-LSTM + 128 FC layer (with dropout=0.2). In order to prevent overfitting which occurred in the above model, we added in the dropout layer and reduced the number of Bi-LSTM layers from 4 to 3, as well as the number of FC layers into 1. The model gives us a better result however still can be optimized. And the final testing result of this model is 53.6%. (refer to the figure below)





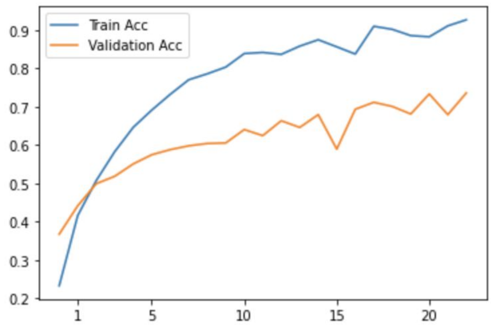
c. 3 layers of Bi-LSTM + Normalization + 128 FC layer (with dropout=0.2). In order to speed up learning and lead to faster convergence, we implemented instance normalization in between Bi-LSTM layer and FC layer. Furthermore, normalization can also reduce the bias of the input data. And the model leads to a better result with a testing accuracy of 59.8%.

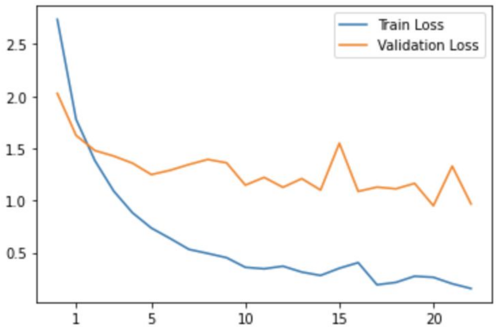




The above three tested networks utilized all the training data as inputs, which made the training process relatively slow. For our second approach, we then decided to feed only a certain number of frames from each segment to the model as inputs:

d. 3 layers of Bi-LSTM + Normalization + 128 FC layer (with dropout=0.2). We chose to extract 60 frames plus 3 auxiliary frames of maximum, minimum and mean values from every segment of the video as our new input data for both training and validation. This gave us a validation score over 70% (refer to the figure below) and a testing accuracy over 65%





2) Hyperparameter level fine-tuning

Besides experimenting with various models, we also tried to fine-tune hyperparameters, such as learning rate and dropout rate. The following table summarizes our selection of hyperparameters:

Learning Rate	0.0001
Dropout Rate	0.2
Epoch	20

3) Model Selection

To properly train and validate our model, we decided to use 80% of labelled data for training and the rest for validation. We will select the best model according to the performance on the validation set.

When we got all the hyperparameters that could give us relatively good results, we fine-tuned the model utilizing 100% of training data instead of the original 80%, to increase our training size.

4. Post Processing

Experimenting with various models, we believe that the models are learnt to make the correct predictions more commonly and incorrect predictions are rather scattered. Thus, post-processing is conducted to select the most commonly predicted result for each segment among top 10 best predictions of all of the models we experimented.

5. Group members contribution

Chen Tingting : Data background and model study, data processing, report writing
Chu Yuan : Hyperparameter fine-tuning, ensembling model, data processing, report writing
Hao Weining : Model selection and fine-tuning, hypermeter fine-tuning, report writing
Li Xingxuan : Model fine-tuning, ensembling results from multiple models, report writing