50.039 The Theory and Practice of Deep Learning Y2021

Big Project Report

Group 18: Chow Jia Yi (1003597) | Yuri Kim (1002334) | Guo Ziwei (1003702) | Dong Ke (1003713) | Nguyen Minh Dang (1003615)

Content Overview

Introduction

Our project aims to create a model that would classify a short video clip depicting an action in a predetermined list of actions.

We would attempt the classification by two different methods:

1. Using only the visual component
2. Using both the visual and audio components together

Dataset

The dataset used is the UCF-101 dataset, which is a collection of short video clips from public sources. Each video clip depicts a person, or multiple people, performing an action in different settings and with different instruments. The dataset has 101 classes of actions. The videos are uniformly in 25 frames-per-second (FPS).

For the purpose of our multimodal experiment, we would only use videos that have accompanying audio. Out of the 101 classes, 51 of them have accompanying audio; however, upon automated inspection of the dataset, we found that only 49 of them have audio components for all the videos in the class. As such, both of our experiments will be using only those 49 classes.

Dataset Distribution

The 49 classes consists of ApplyEyeMakeup, ApplyLipstick, Archery, BabyCrawling, BalanceBeam, BandMarching, BlowDryHair, BlowingCandles, BodyWeightSquats, Bowling, BoxingPunchingBag, BoxingSpeedBag, BrushingTeeth, CliffDiving, CricketBowling, CricketShot, CuttingInKitchen, FieldHockeyPenalty, FloorGymnastics, FrisbeeCatch, FrontCrawl, Haircut, Hammering, HammerThrow, HandstandWalking, HeadMassage, IceDancing, Knitting, LongJump, MoppingFloor, ParallelBars, PlayingCello, PlayingDaf, PlayingDhol, PlayingFlute, PlayingSitar, Rafting, ShavingBeard, Shotput, SkyDiving, SoccerPenalty, StillRings, SumoWrestling, Surfing, TableTennisShot, Typing, UnevenBars, WallPushups, WritingOnBoard. A total of 6549 videos were obtained from these 49 classes and the distribution among different classes are shown in Figure 1. The videos are split for training and testing according to the official *Train/Test Splits for Action Recognition* which can be obtained from the download website and the distribution of each splits are shown in Figure 2, 3 and 4. From the graphs, it is observed that the classes are relatively balanced with no large difference in the number of videos between each class. Having a balanced data set would have lesser bias, and hence a more accurate model.

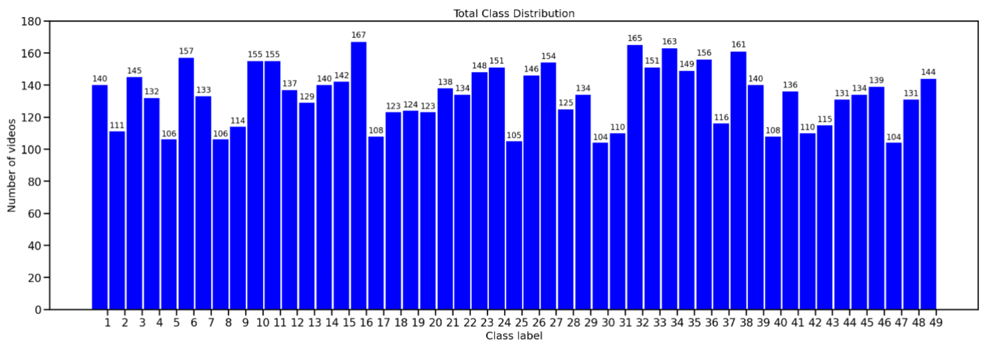


Figure 1: Total class distribution

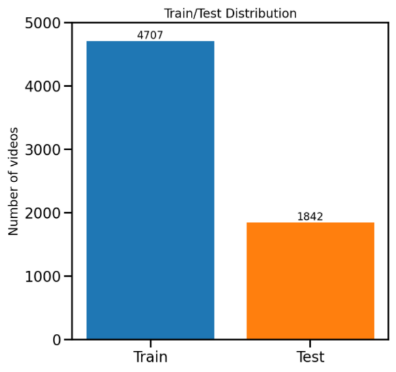


Figure 2: Train and test set distribution

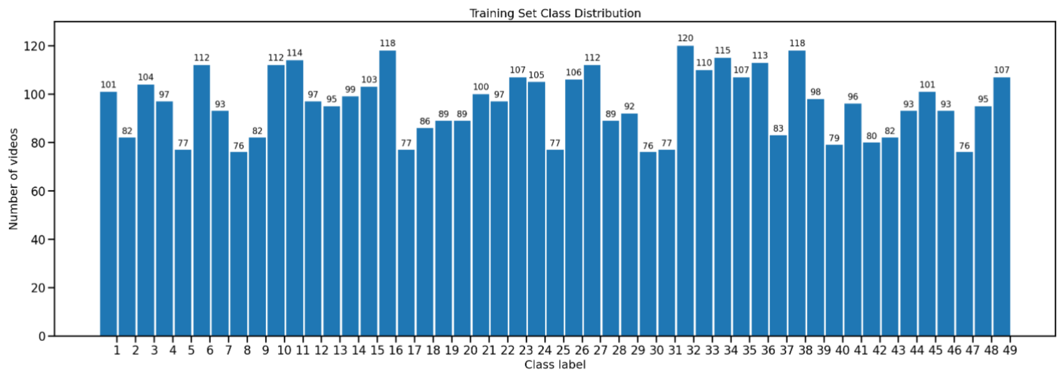


Figure 3: Training set class distribution

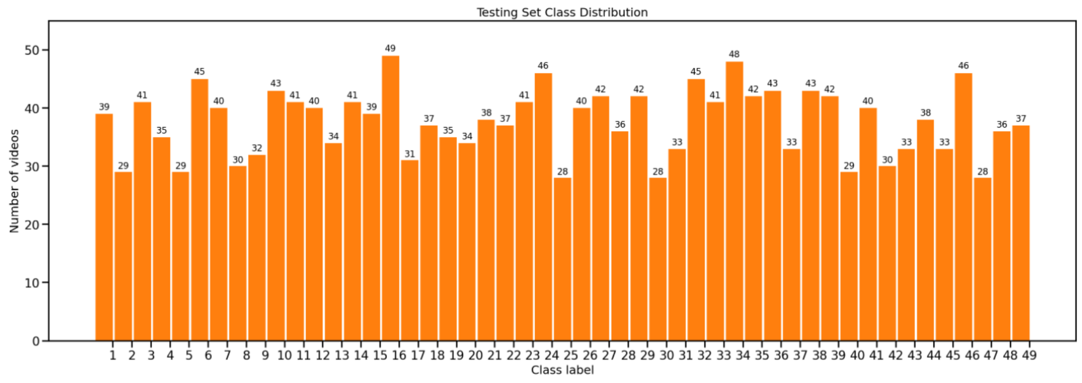


Figure 4: Testing set class distribution

Methods and Architecture

We had proposed two types of methods and architectures.

1. **Using only the visual component**
   In the first model, we used the visual components alone. The videos from the dataset are extracted into frames using the av library. After that, we loaded the dataset using a self-built dataset loader which takes in 50 frames (2 seconds) per video and pad those whose length is shorter than 2 seconds. The frames are loaded and then passed through an encoder to get the frames embeddings, which are passed to an LSTM recurrent NN layer to extract the features. After that, it is passed through a dense layer for final classification.

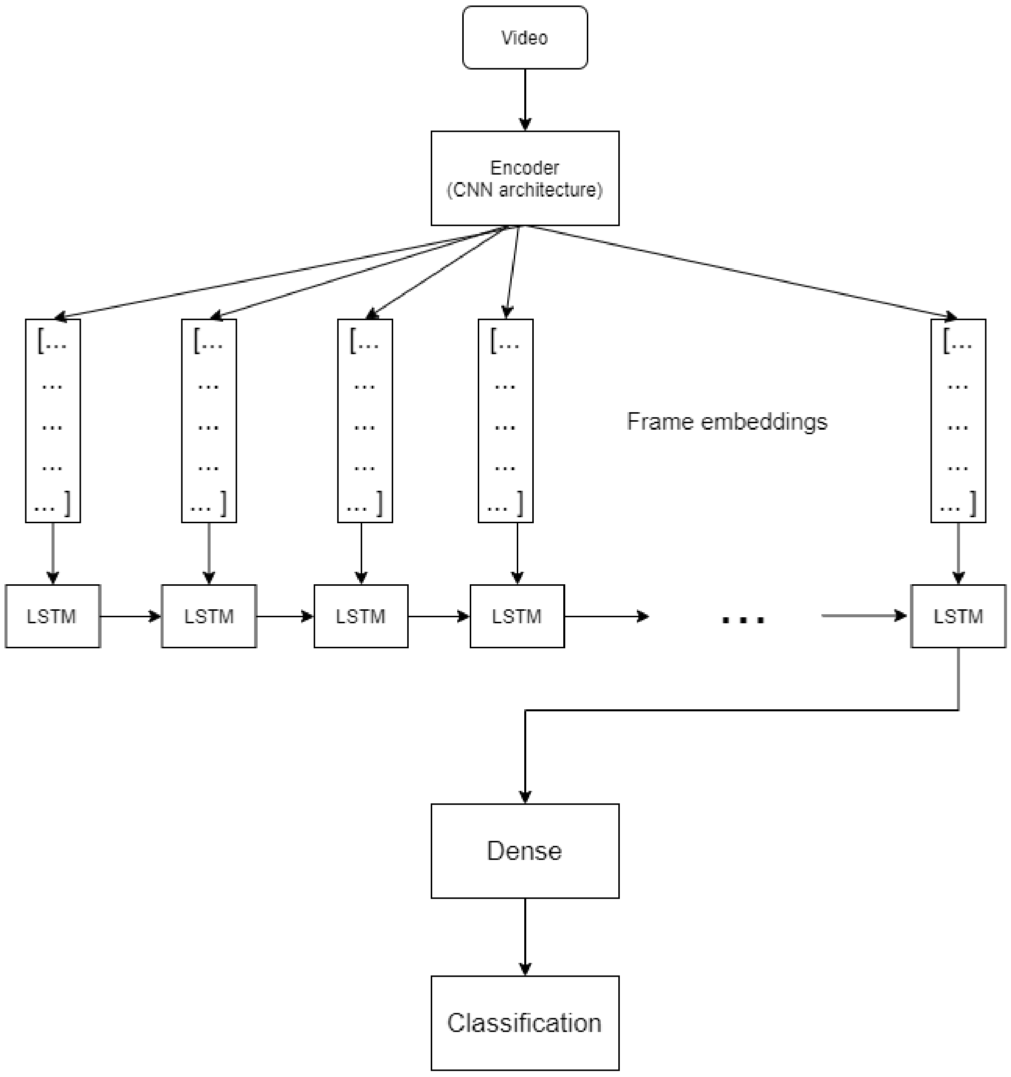


Figure 5: Overall architecture using visual components only

2. Using the visual and audio components together

In the second exploration, we combined both audio and visual components. The audio file is stripped from video using the moviepy library. Then, the extracted audio file is trimmed to 2 seconds to coincide with the video components. For those audio whose length is shorter than 2 seconds, we padded it with silence. After that, we used the librosa library to extract the common audio features for processing which include rms, chroma_stft, spectral_centroid, spectral_bandwidth, spectral_rolloff, zero_crossing_rate, mfcc. These features were concatenated together to form a joint representation of the audio features in the form of tensors.

The processing of visual components is the same as the previous visual-alone model for extracting frames, encoding, and LSTM. However, the output tensors of the LSTM for visual components are concatenated with the tensors of audio features before passing through the dense layer for classification.

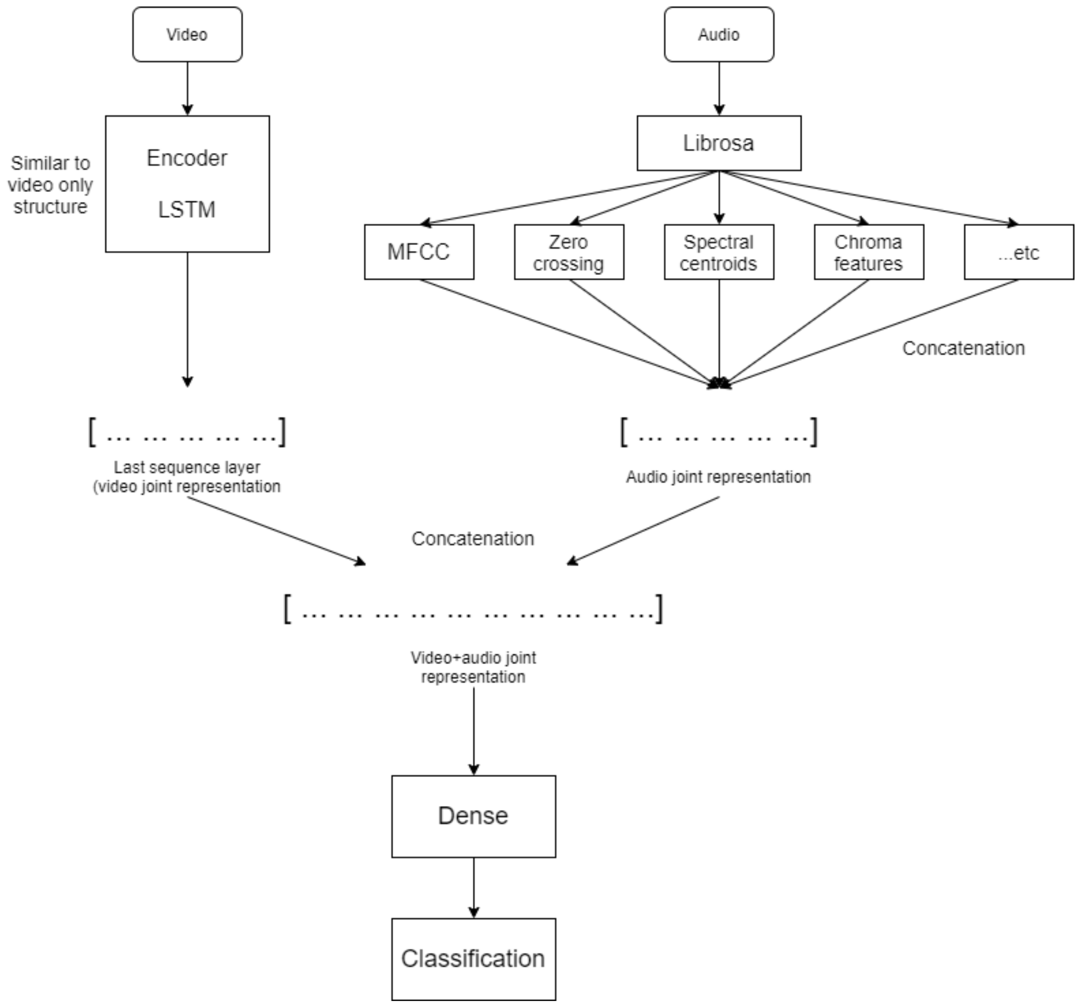


Figure 6: Overall architecture using visual and audio components

Experiments and Results visualization

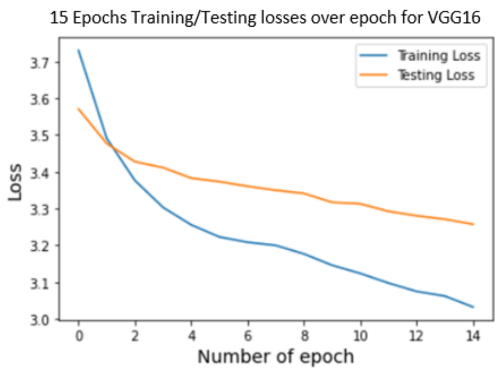
We have used pre-trained models, VGG16, DenseNet121, DenseNet161 and ResNet152, in the encoder to extract feature embedding. Due to limited computational resources, all the models are trained with only 15 epochs, batch size of 4 and Adam as the optimizer. The experiments are classified according to the model architecture they follow, and the results are shown below.

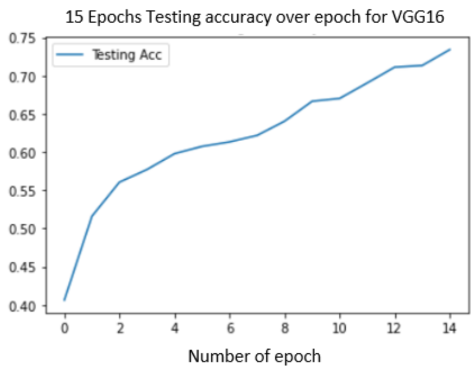
	Method 1		**Method 2**	
Model	**Accuracy**	**Number of epochs trained**	**Accuracy**	**Number of epochs trained**
VGG16	73.4%	15	14.2%	15
ResNet152	64.0%	15	-	-
DenseNet121	69.8%	15		
DenseNet161	76.0%	15		

Table 1: Performance of each of experiments

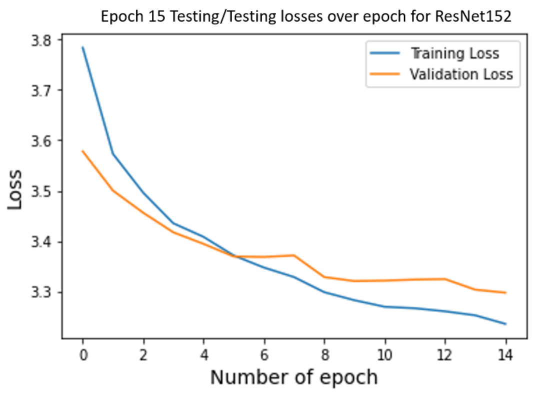
1. Method 1: using visual components only

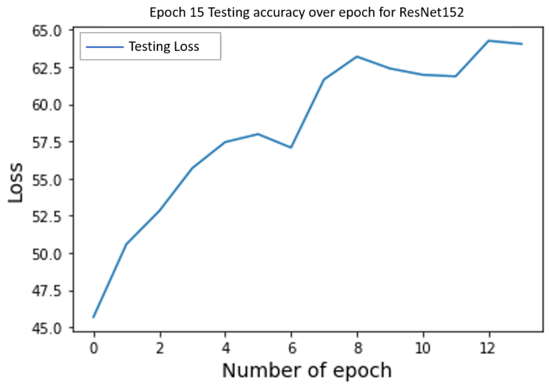
1.1 VGG16



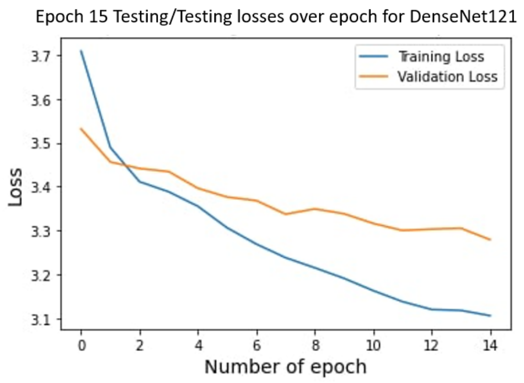


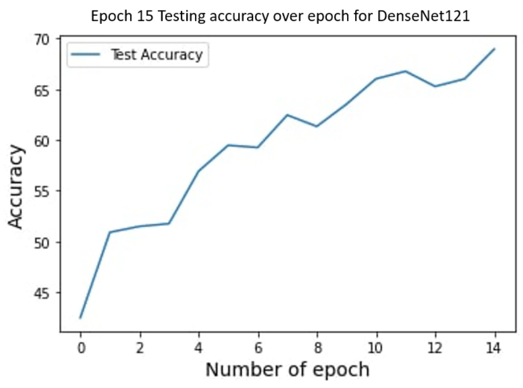
1.2 ResNet152



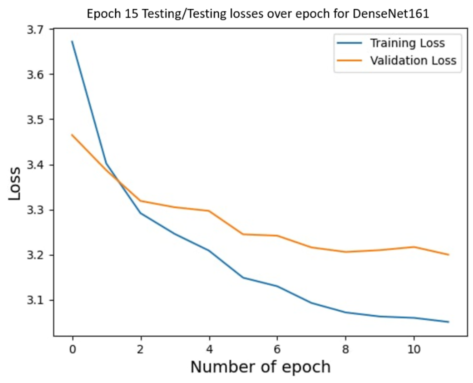


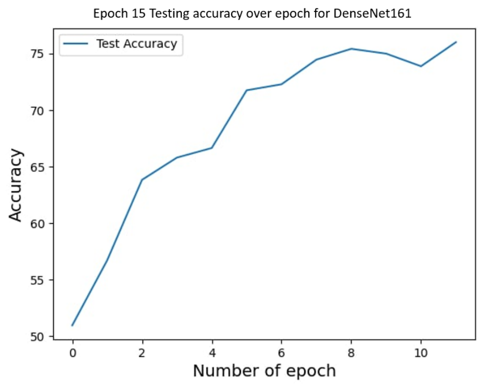
1.3 DenseNet121





1.4 DenseNet161

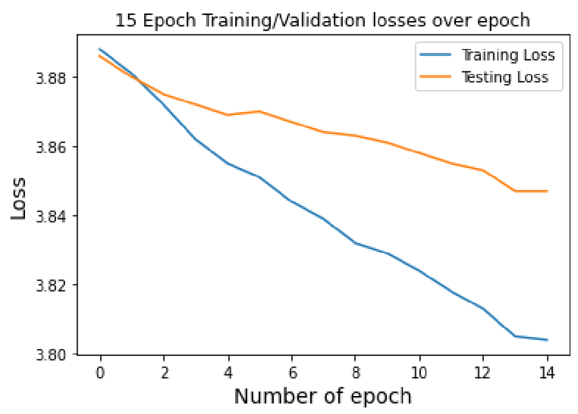


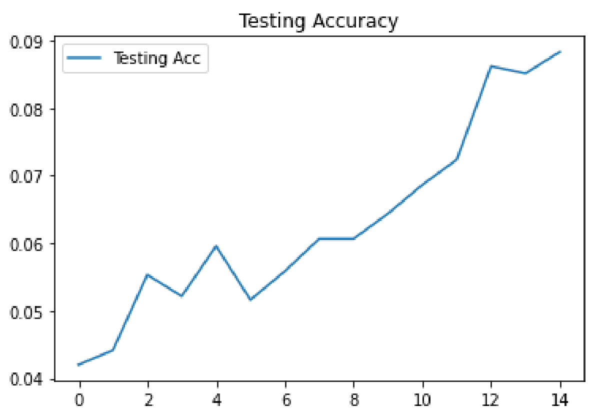


2. Method 2: using visual components and audio components

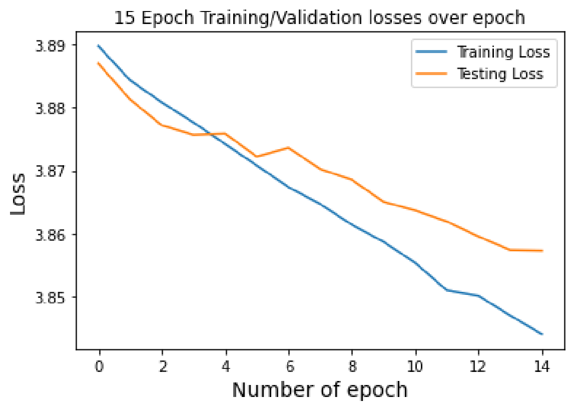
Due to time and hardware constraints, we could only run multimodal experiments for VGG16 and DenseNet121 encoders

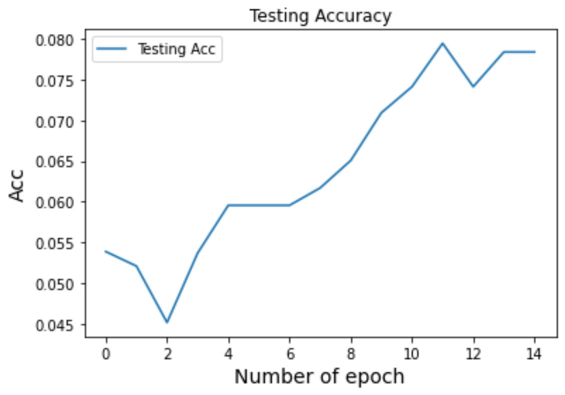
2.1 VGG16 with audio component





2.2 DenseNet121 with audio component





Discussions

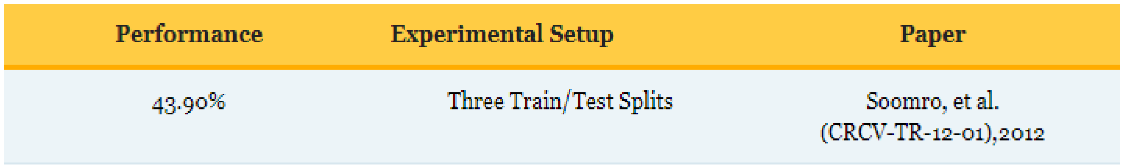
For the video only analysis, it is observed that DenseNet161 performs the best with the highest test accuracy of 76.0% at the 15th epoch and this is reasonable because deeper layers are generally better able to capture features embeddings and DenseNet also handles gradient vanishing issues.

The multimodal analysis method of using both the video and audio component did not perform as expected, with extremely low accuracy both in training and testing. Although it was improving over time, it did not increase past a certain threshold. Our hypothesis was that the low accuracy is attributed to the mismatch between video and audio; as we manually inspect a few of the videos, we found that not all of them retain the original audio of the recording, but replaced with background music instead. Moreover, many video clips have similar audio of only ambient noise, or random chatter, which do not distinctively reflect the action portrayed.

Comparison with state-of-the-art models

We have compared our model with several state-of-the-art models that use the same dataset.

- In the official website that publishes the UCF-101 dataset, the performance for the model using the dataset is revealed. In the paper (Soomro, 2012), the videos are splitted into four types and the accuracy for the predefined action types are: Sports (49.40%), Playing Musical Instrument (42.04%), Human-Object Interaction (36.62%), Body-Motion Only (37.64%), Human-Human Interaction (42.66%).
  In general, Table 2 shows the accuracy from the paper.

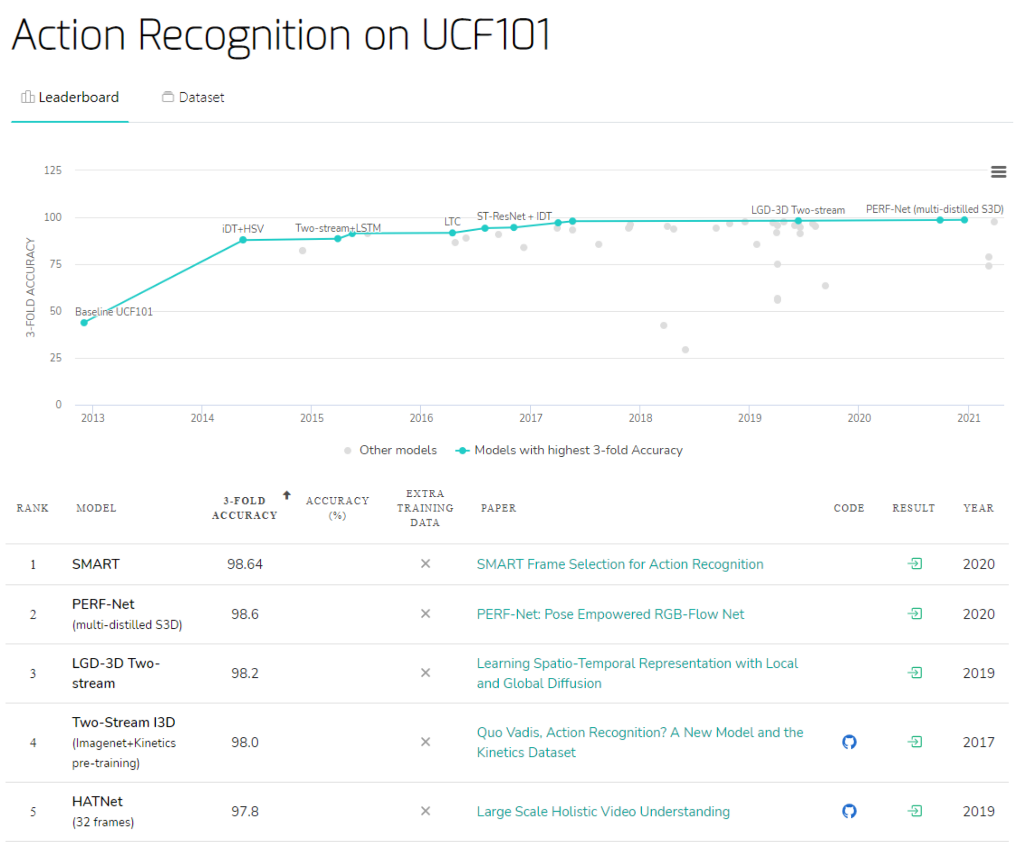
  

Performance	Experimental Setup	Paper
43.90%	Three Train/Test Splits	Soomro, et al. (CRCV-TR-12-01),2012

  Table 2: Accuracy from the paper

  They extracted the Harris 3D corners and computed 162 dimensional HOG/HOF descriptors for each. Utilizing three train/test splits, a SVM was trained using the histogram vectors of the training set. They employed a nonlinear multiclass SVM with histogram intersection kernel and 101 classes each representing one action.
  To compare with our models in method 1, we have a higher accuracy because CNN encoders and LSTM, which is more suitable for video due to their recurrent nature, are used instead of SVM for classification.

- In addition, we have also compared our model with the leaderboard models for action recognition using UCF101 dataset (Papers with Code - UCF101 Benchmark (Action Recognition), n.d.) and some of the best performing models are listed below.

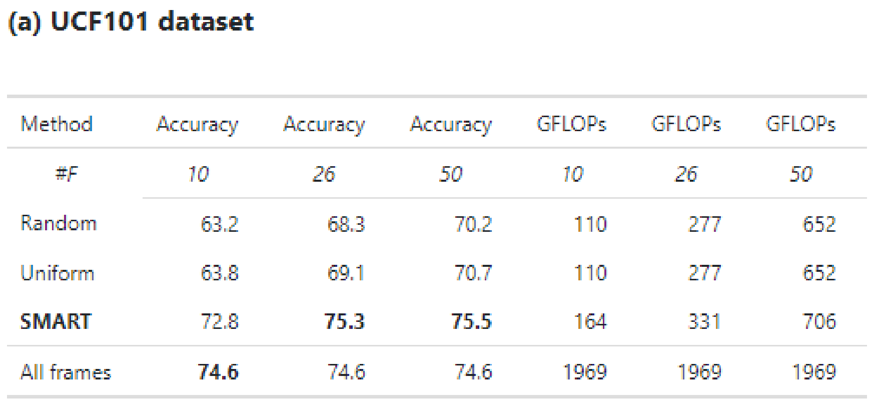


RANK	MODEL	3-FOLD ACCURACY	ACCURACY (%)	EXTRA TRAINING DATA	PAPER	CODE	RESULT	YEAR
1	SMART	98.64		×	SMART Frame Selection for Action Recognition			2020
2	PERF-Net (multi-distilled S3D)	98.6		×	PERF-Net: Pose Empowered RGB-Flow Net			2020
3	LGD-3D Two-stream	98.2		×	Learning Spatio-Temporal Representation with Local and Global Diffusion			2019
4	Two-Stream I3D (Imagenet+Kinetics pre-training)	98.0		×	Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset			2017
5	HATNet (32 frames)	97.8		×	Large Scale Holistic Video Understanding			2019

Figure 7: Leaderboard from Paper with Codes

It can be seen that the top performing models have much higher accuracy as compared to our own model. Besides the hardware constraints that would be a cause of the difference, the difference in model architecture also contributes to their higher accuracy.

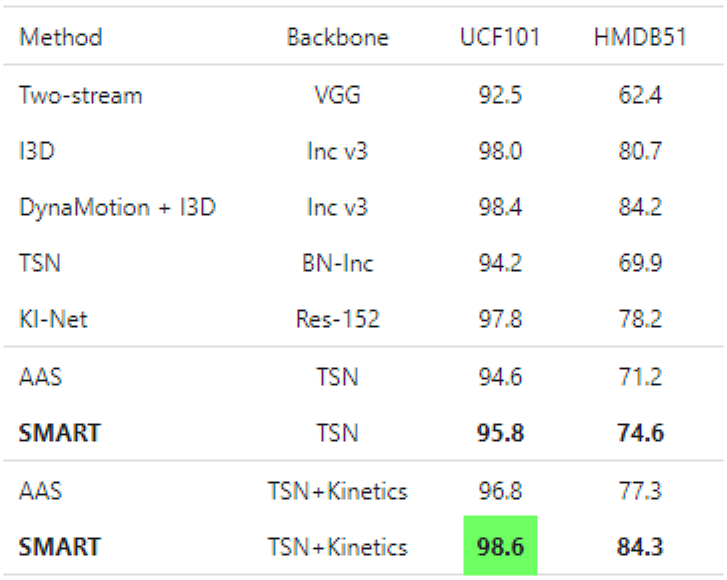
For example, for the first SMART model, their output using UCF101 dataset without the pre-processing with the input is on par with our best performing model around 76%.



(a) UCF101 dataset

Method	Accuracy	Accuracy	Accuracy	GFLOPs	GFLOPs	GFLOPs
#F	*10*	*26*	*50*	*10*	*26*	*50*
Random	63.2	68.3	70.2	110	277	652
Uniform	63.8	69.1	70.7	110	277	652
SMART	72.8	**75.3**	**75.5**	164	331	706
All frames	**74.6**	74.6	74.6	1969	1969	1969

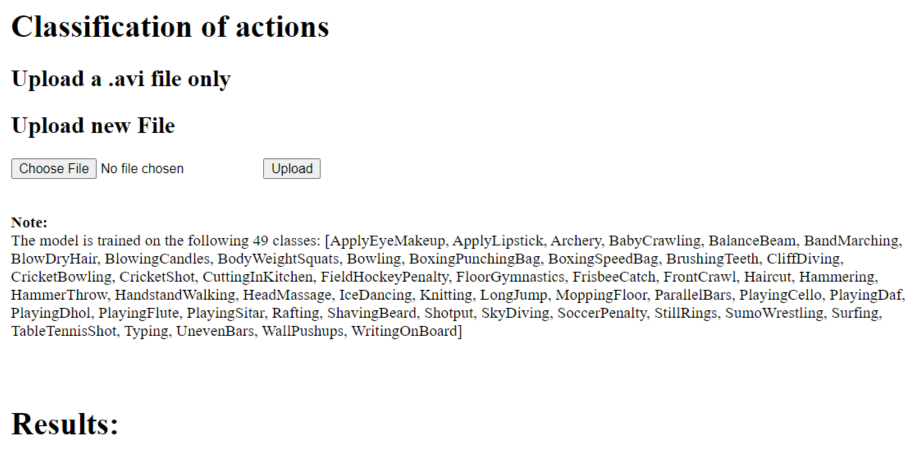
However, when they extend SMART as a pre-processing step, the accuracy soars to 98.6%.



Method	Backbone	UCF101	HMDB51
Two-stream	VGG	92.5	62.4
I3D	Inc v3	98.0	80.7
DynaMotion + I3D	Inc v3	98.4	84.2
TSN	BN-Inc	94.2	69.9
KI-Net	Res-152	97.8	78.2
AAS	TSN	94.6	71.2
SMART	TSN	**95.8**	**74.6**
AAS	TSN+Kinetics	96.8	77.3
SMART	TSN+Kinetics	**98.6**	**84.3**

User Interface and Testing Results

We have built a simple user interface (UI) for users to input their own video clips. In this UI, we have loaded the weights of our model that was trained with DenseNet161 as the encoder.
After setting up the environments which can be found at the last section of the report, the users can select a video clip and upload it for prediction as shown in Figure 8.



Classification of actions

Upload a .avi file only

Upload new File

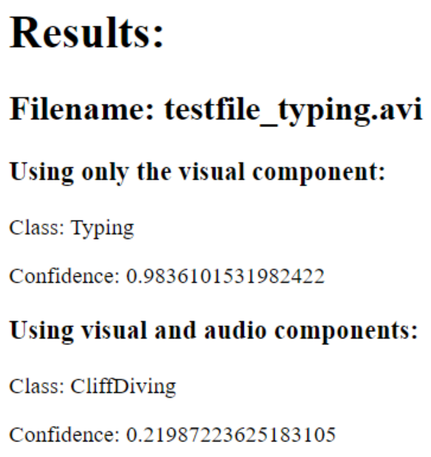
Note:
The model is trained on the following 49 classes: [ApplyEyeMakeup, ApplyLipstick, Archery, BabyCrawling, BalanceBeam, BandMarching, BlowDryHair, BlowingCandles, BodyWeightSquats, Bowling, BoxingPunchingBag, BoxingSpeedBag, BrushingTeeth, CliffDiving, CricketBowling, CricketShot, CuttingInKitchen, FieldHockeyPenalty, FloorGymnastics, FrisbeeCatch, FrontCrawl, Haircut, Hammering, HammerThrow, HandstandWalking, HeadMassage, IceDancing, Knitting, LongJump, MoppingFloor, ParallelBars, PlayingCello, PlayingDaf, PlayingDhol, PlayingFlute, PlayingSitar, Rafting, ShavingBeard, Shotput, SkyDiving, SoccerPenalty, StillRings, SumoWrestling, Surfing, TableTennisShot, Typing, UnevenBars, WallPushups, WritingOnBoard]

Results:

Figure 8: General interface of our GUI

Generally, the model using the visual components only predicts relatively well with high. There are some interesting examples that we would like to share.

Firstly, we filmed one of our group members typing on his laptop and uploaded it for prediction. The prediction result is accurate with high confidence as shown in Figure 9.



Results:

Filename: testfile_typing.avi

Using only the visual component:

Class: Typing

Confidence: 0.9836101531982422

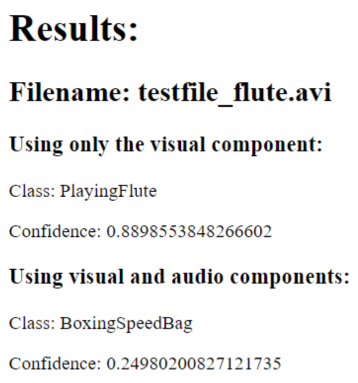
Using visual and audio components:

Class: CliffDiving

Confidence: 0.21987223625183105

Figure 9: Result obtained with our own video

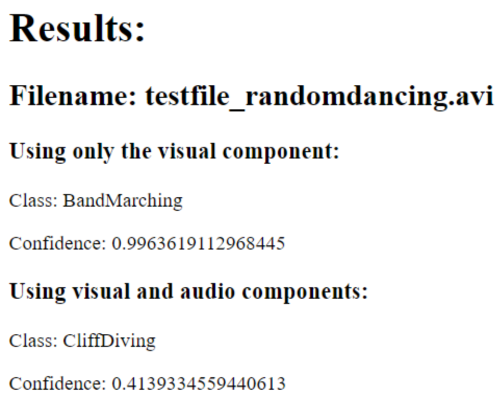
Secondly, we uploaded a video of a lady playing flute. Our model is able to predict the video correctly with an accuracy of around 90.0% as shown in Figure 10. It is worth noting that most of the flute videos in the dataset uses metal flute but the uploaded video is a wooden flute. Despite the difference in material, our model is still able to classify the wooden flute video correctly.



```
Results:
Filename: testfile_flute.avi
Using only the visual component:
Class: PlayingFlute
Confidence: 0.8898553848266602
Using visual and audio components:
Class: BoxingSpeedBag
Confidence: 0.24980200827121735
```

Figure 10: Result obtained with our custom video

Thirdly, we have chosen a video whose action is not in the 49 classes that we have trained. The video consists of a group of people dancing together but our model predicts it as BandMarching as shown in Figure11. This misclassification can be due to the model capturing the feature that multiple people are making movements together.



```
Results:
Filename: testfile_randomdancing.avi
Using only the visual component:
Class: BandMarching
Confidence: 0.9963619112968445
Using visual and audio components:
Class: CliffDiving
Confidence: 0.4139334559440613
```

Figure 11: Result obtained with our custom video

Manual to Re-create the Models, Outputs, and the UI

Architecture 1:

1. Obtain our version of dataset, UCF_49 from
   https://drive.google.com/file/d/18e6TwtREHLS2rLMXBKZx3MBm1JxjzUVF/view?usp=sharing
   and the train_test_split from
   https://drive.google.com/file/d/1_uBpXEo4Kf2QJYw7D-YO_inn_e1Q-Cuv/view?usp=sharing

2. Git clone the entire github repository from
   https://github.com/Zzziwei/50.039-The-Theory-and-Practice-of-Deep-Learning-Y2021

3. Unzip the UCF_49 and train_test_split in the same folder where the repository is cloned.
   In the terminal, run *python extract_frames.py*. Wait for about 2 hours to get all the frames.
   Alternatively, download and unzip the frames from
   https://drive.google.com/file/d/1tjTB_TK53-UtP3l2o7pVMeNhVL_HA7Hz/view?usp=sharing.
   Make sure that the frames are in the same as the train_test_split.

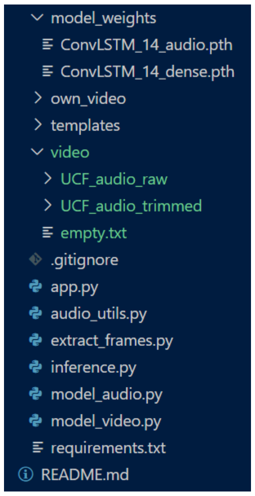
4. Before running any model, make sure that you have UCF_49-frames, train_test_split and visual_component.ipynb in the same directory. Open the jupyter notebook, visual_component.ipynb, and run the codes accordingly for training and testing.

Architecture 2:

1. Step 1-3 is the same as the architecture 1.
2. Download the pickle file from
   https://drive.google.com/file/d/12wsyIfR8-Ub2CBsOpTQ8-YY1__aKI-oM/view?usp=sharing or alternatively run the audio processing python file to get the pickle.
3. Open the jupyter notebook, visual_and_audio_component.ipynb and run the codes accordingly for training and testing.

User Interface:

1. Upon cloning the entire repository in Step 2 of Architecture 1, the codes for the UI component will also be stored locally in the folder called app.
2. Navigate into the *app* folder. Download the model weights from https://drive.google.com/file/d/1IfCClipIXsHxU4x5_tUxLPbtH1-j2noD/view?usp=sharing because the weights of the model are too large to be uploaded to github. Unzip and make sure that the app folder contains all of the files and folders shown in Figure



```
model_weights
  ConvLSTM_14_audio.pth
  ConvLSTM_14_dense.pth
own_video
templates
video
  UCF_audio_raw
  UCF_audio_trimmed
  empty.txt
.gitignore
app.py
audio_utils.py
extract_frames.py
inference.py
model_audio.py
model_video.py
requirements.txt
README.md
```

Figure 12: Files and folders in app folder

3. Set-up for the UI locally.
   a. Create a virtual environment using
   ```
   python -m venv venv
   ```
   b. Activate the virtual environment
   ```
   # Activate the virtualenv (OS X & Linux)
   $ source venv/bin/activate

   # Activate the virtualenv (Windows)
   $ venv\Scripts\activate
   ```
   c. Navigate out in the *app* folder. In the *app* folder, install the necessary libraries using
   ```
   pip install -r requirements.txt
   ```
4. Run the application using
   ```
   python app.py
   ```
5. Choose a video clip with .avi format and upload, the result would be shown. The custom video used in the User Interface and Testing Results section can be found in the *own_video* folder.

Github link: https://github.com/Zzziwei/50.039-The-Theory-and-Practice-of-Deep-Learning-Y2021

References

1. Soomro, K. (2012, December 3). UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. ArXiv.Org. https://arxiv.org/abs/1212.0402
2. Papers with Code - UCF101 Benchmark (Action Recognition). (n.d.). Action Recognition on UCF101. Retrieved May 2, 2021, from https://paperswithcode.com/sota/action-recognition-in-videos-on-ucf101