Action Detection in Videos

50.039 Theory of Deep Learning Big Project

Xie Han Keong 1003876
Jisa Mariam Zachariah 1003638
Amarjyot Kaur Narula 1003084
Xavier Tan De Jun 1003376
Goh Yi Lin, Tiffany 1003674

Introduction

Our project focuses on teaching our machine learning model to develop a fine-grained detailed understanding of basic human actions and to classify them accordingly. This can serve as a backbone for transfer learning for more task-specific machine learning tasks in the future which involves differentiating human actions such as autonomous detection of suspicious activity in CCTV feeds, autonomous threat detection in baby monitoring devices, driverless cars, etc, which can be achieved through fine-tuning higher layers in our pre-trained model.

Our project can be found on Google Drive as well as GitHub.

Model Architecture

1. **Two-Stream (Ours)**

The two-stream model makes use of the spatial and temporal properties of videos. The spatial component of the video lies in the individual frames and it is responsible for the information regarding the objects in the video. The temporal component is responsible for the movement of the objects and the observers across frames. The video recognition is separated into 2 streams through the two stream architecture and it makes use of the fine tuned Resnet50 layer and a simple CNN layer. The fine tuned ResNet50 layer takes in the preprocessed individual video frames as the input and the simple CNN layer takes the optical flow as the input. The outputs from both layers are then combined and passed into an additional fusion layer which will output the final result.

2. **CNN-LSTM**

This model makes use of CNN and LSTM layers. The CNN layers will be used to extract spatial features from each video frame, which will be concatenated and fed into an LSTM layer to extract out temporal features. The result of the CNN and LSTM layers will be passed into a fully-connected layer for classification.

3. **FineCoarse+Fine+Coarse (ImageNet pre-trained)**

This model represents most of the information in video data by using frequency analysis. Fine details in motion from region boundaries are identified by high frequency in the spatio-temporal domain while low frequencies carry highly redundant course information, reducing the efficiency for video models which receive raw RGB frames as input. A Motion Band-pass Module (MBPM)

is used in this model for separating the fine-grained information from coarse information in raw data. The MBPM is embedded into a 2 pathway CNN architecture called FineCoarse network. The efficiency of this model is defined by avoiding the redundant information in the feature space produced by the 2 pathways. One pathway works on downsampled features of low-resolution data and the other works on the finer detailed information from the MBPM. This model currently produces the best output performance with a top 1 accuracy of 57%.

4. **TDN ResNet101 (one clip, center crop, 8+16 ensemble, ImageNet pretrained, RGB only)**

This model adopts the Temporal Difference Network architecture which aims to capture multi-scale temporal information from videos. It creates an efficient temporal module (TDM) by employing a temporal difference operator and comparing its effect on short and long term motion modelling. TDN is designed with 2 levels for difference modelling. For local video modelling, 2D CNNs take as input the temporal difference over consecutive frames and for global motion modelling, temporal difference across segments is used to extract long-range structure for motion feature excitation.

5. **PAN ResNet101 (RGB only, no Flow)**

This model helps to speed up the video recognition process by removing the dependence on optical flow which is time consuming to extract. It uses Persistence of Appearance (PA) which distils the motion information from motion boundaries as opposed to optical flow. It increases efficiency as it only accumulates pixel differences in feature space and is hence 1000 times faster than optical flow modelling. It also uses a global temporal fusion strategy called Various-timescale Aggregation Pooling (VAP) that can adaptively model long range temporal relationships from different time scales. PA and VAP are integrated as a Persistent Appearance Network which models temporal information.

Model Performance

We were unable to complete the training process due to time constraints and thus we are unable to report the validation accuracy and loss as well as visualisations. The current state of the art models used for the Something-Something video classification problem include the following models which rank as the top 3 in terms of performance on the Something-Something V1 dataset (See Table 1).

Dataset

We used the 20BN-Something-Something v2 (SSV2) dataset which can be found at https://20bn.com/datasets/something-something/v2. The SSV2 dataset is a large collection of labeled video clips that show humans performing pre-defined basic actions with everyday objects. Table 2 shows more information about the SSV2 dataset.

The input to our model was video files of humans performing certain basic actions. The output was one out of 174 labels describing the action performed by the human in the input video. Some examples of labels are: “Putting something on a surface”, “Moving something up”, “Covering something with something”, “Pushing something from left to right”, and etc.

Table 1: State of the art models and their performance

Rank	Model	Year	Top-1 Accuracy (%)	Top-5 Accuracy (%)
1	FineCoarse+Fine+Coarse (ImageNet pre-trained)	2021	57.0	83.7
2	TDN ResNet101 (one clip, center crop, 8+16 ensemble, ImageNet pretrained, RGB only)	2020	56.8	84.1
3	PAN ResNet101 (RGB only, no Flow)	2020	55.3	82.8

Table 2: Information about the SSV2 dataset

Total download size	19.6GB
File format	.webm (encoded with VP6 codec)
Total number of video files	220847
Number of videos in training set	168913
Number of videos in validation set	24777
Number of videos in test set	27517
Number of labels	174

This is a brief description of the steps that we took to prepare the data.

1. **extract_data.ipynb**. Unzip the download the dataset and organize the data into train/test directories and subdirectories for the respective classes
2. **copy_data.ipynb**. Select and copy a subset of the data into ssv2_mini
3. **preprocess_data.ipynb**. Process the raw data using transform functions and save the processed data to respective subdirectories. Our processing steps included:
    - **Converting video to images (spatial data)**. We applied a random crop and normalisation to the images.
    - **Converting video to optical flow data**. We converted the video frames to Optical Flow data and applied a random crop (See Figure 1).

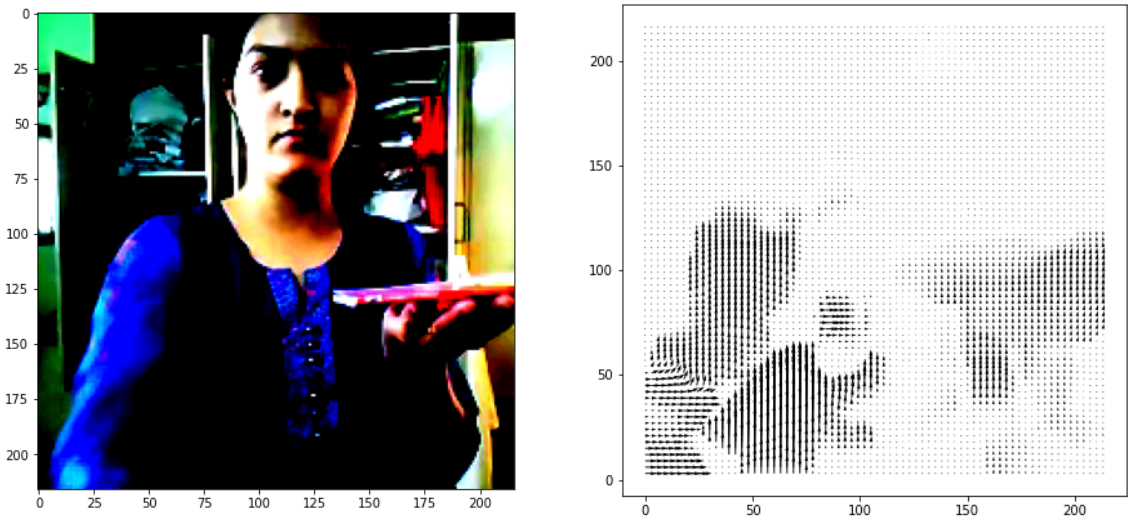

Figure 1: Preprocessed video frame and quiver plot of optical flow

Training Process

For training, we loaded the respective preprocessed data and used it to train the different models (train_models.ipynb). The steps we took for each model are as follows.

- **Fine-tuned ResNet (Two Stream)**. It takes in a preprocessed single frame of the video as the input and output the possible 174 classes that belong to the input.

  Fine-tuned ResNet was trained with a batch size of 32, learning rate of 0.001 and weight decay of 0.000001. Our loss function is cross entropy loss and our optimiser is Adam with betas of (0.9, 0.999). We trained for 10 epochs and subsequently 50 epochs. However, the training loss and the validation accuracy was stagnant at approximately 5.16 and 0.005 respectively as shown in Figure 1.

- **Temporal CNN (Two Stream)**. It is a convolution layer which has progressively decreasing layer size. It takes in the optical flow as its input and outputs the classes according to the optical flow.

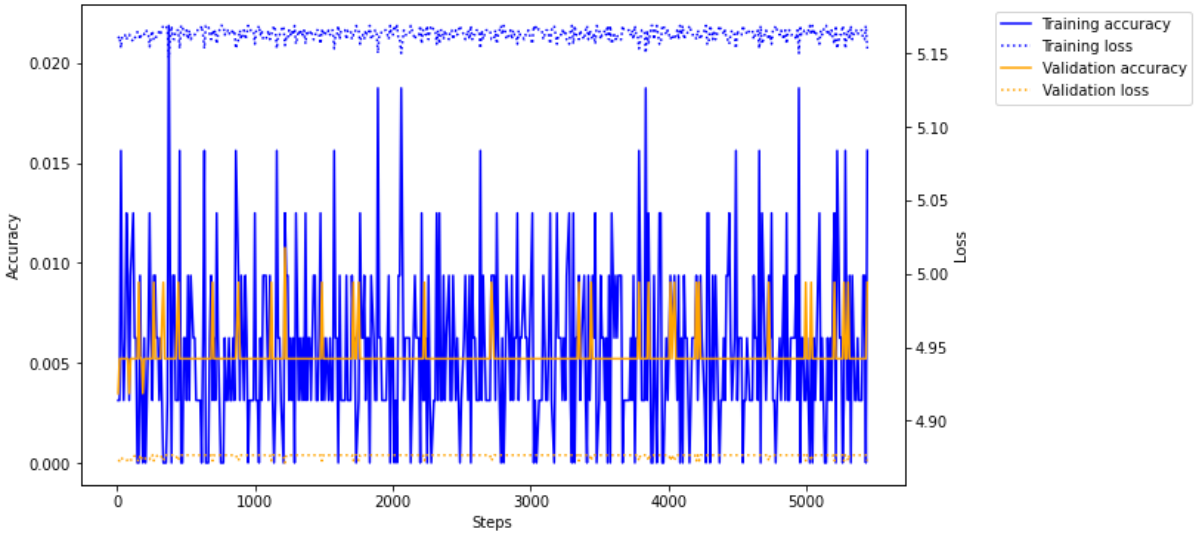

Figure 2: Training loss and accuracy, validation loss and accuracy over time steps

Implementation

Please run the following Jupyter Notebooks to recreate the results as previously described. To perform preprocessing of our dataset, run step 1 to 3. To perform training of our model and obtain results on the validation set, run step 4.

Table 3: Steps to running our models and their functionalities

Steps	File	Function
1	extract_data.ipynb	Unzip and categorises videos into its relevant folders
2	copy_data.ipynb	Select a subset of the main dataset
3	preprocess_data.ipynb	For each video, extract a single frame for training of Fine-tuned ResNet Model, and also extract a sequence of 10 frames to obtain its optical flow data for training of our Temporal CNN Model.
4	train_models.ipynb	Load preprocessed data and train the models.

Project Management & Contribution

Table 4: Contributions of each team member

Name	Contribution
Goh Yi Lin, Tiffany	Data Preprocessing, Report
Xavier Tan De Jun	Dataset procurement, Training Functions, Report
Xie Han Keong	Source code and notebooks, Report
Jisa Mariam Zachariah	Two-stream Model, Temporal CNN Model, Report
Amarjyot Kaur Narula	Data Preprocessing, Report

References

Papers with code - something-something v1 benchmark (action recognition). (n.d.). Retrieved May 03, 2021, from https://paperswithcode.com/sota/action-recognition-in-videos-on-something-1

Simonyan, K., & Zisserman, A. (2014, November 12). Two-Stream convolutional networks for action recognition in videos. Retrieved May 03, 2021, from https://arxiv.org/abs/1406.2199

Woodfrog. (n.d.). Woodfrog/actionrecognition. Retrieved May 03, 2021, from https://github.com/woodfrog/ActionRecognition

Papers with Code - PAN: Towards Fast Action Recognition via Learning Persistence of Appearance. PAN: Towards Fast Action Recognition via Learning Persistence of Appearance | Papers With Code. (n.d.). https://paperswithcode.com/paper/pan-towards-fast-action-recognition-via.

Papers with Code - TDN: Temporal Difference Networks for Efficient Action Recognition. TDN: Temporal Difference Networks for Efficient Action Recognition | Papers With Code. (n.d.). https://paperswithcode.com/paper/tdn-temporal-difference-networks-for.

Papers with Code - Action Recognition. Action Recognition | Papers With Code. (n.d.). https://paperswithcode.com/task/action-recognition-in-videos?page=8.

Papers with Code - Video Classification with FineCoarse Networks. Video Classification with FineCoarse Networks | Papers With Code. (n.d.). https://paperswithcode.com/paper/video-classification-with-finecoarse-networks.