Video Classification and Action Recognition

FNU Vivek
Computer Science
North Carolina State University
Raleigh, NC, USA
vvivek@ncsu.edu

Ravindersingh Rajpal
Computer Science
North Carolina State University
Raleigh, NC, USA
rkrajpal@ncsu.edu

I. Introduction

Action Recognition (Video Classification) is a technique to classify a person's action by studying his/her movement. Studying the motion of a person is quite a challenging task due to the abundance of information changing with time during the action. It also depends on the placement of the camera as such camera position can change in a video that changes the frame completely. Various architectures have been proposed over time to achieve successful results. We have performed a comparative study over three such architectures. The classification has been performed over a dataset provided by University of Central Florida (UCF) collected from YouTube. The dataset comprises of 13320 videos from 101 action categories. The architecture used and their respective results are discussed further.

II. Methodology

Before jumping over to various architectures, a sufficient amount of time was spent in understanding and extracting data. We performed data exploration and pre-processing over the dataset which is discussed as follows.

A. Data Exploration and Pre-processing

Along with the videos, a train-test split directory is also provided in UCF101 dataset. The directory has 3 splits namely trainlist01/testlist01, trainlist02/testlist02, trainlist03/testlist03. These text documents are predefined splits created and distributed in order to compare benchmarks by various architectures. We used trainlist01/testlist01 in our project.

Since, the architectures that we decided to implemented worked on frames and optical/dense flows for a video, we had two options to approach this problem. One way was to reduce to overhead of reading videos every time by storing corresponding frames and all details on the disk. Second way was to use the frames and other information on the fly while training. We tried both ways while training one model. Storing the frames and optical flows was the optimal solution. The overhead was to pre-process all of approximately 11k videos in the 'splitfile' before training or testing. While storing the frames, we also re-sized them. As two of the models required 224x224, we consciously decided to keep the processed image as 224x224x3. Initially, we faced a lot of difficulty performing this step due to lack of proper resources e.g 100% CPU utilization was observed on an 8 core machine and extraction time was nearly 3-4 minute per video. Eventually, we set up Google Cloud Platform to perform this task and had our pre-processing completed in around 100 hours on 2 instances each.

B. Architecture 1: Temporal Segment Networks (TSN)

Temporal Segment Networks architecture was proposed in 2016 by Wang et al. [1]. It deals with studying long range temporal information from the video. The architecture proposes to use segments of video for its classification. The authors chose 3 segments and we decided to stick with the same number of segments. There are 3 segments(earlier, middle, later) of a video and we randomly sample a frame(from each segment) for studying the spatial nature of that video and also sampling an optical flow(from each segment) to study the temporal nature of the video. In all, we have 6 convolutional networks for classifying each of the frames and optical flows sampled randomly. After that, we take a consensus (averaged results) of the spatial networks and similarly one for the temporal networks and a final consensus to provide the final prediction(action) for a video.

The authors used Batch Normalization Inception as a single unit of ConvNet in their architecture. Since, the inception network is a very dense network, we decided to perform transfer learning to save the time while training. We used InceptionResnetV2 model provided by Keras as our ConvNet for each segment.

The final results which we are proposing from our implementation is not what we intended to achieve. There is a major flaw in the implementation of Batch Normalization layer of Keras while it is being used in the case of transfer learning. An issue was raised on GitHub on Keras' repository regarding this. So, we had to train our model by keeping the batch normalization layer's trainable property as False which is equivalent to not using batch normalization and thus, achieved poor results. Training time was also quite large because of a very complex model. Increasing learning rate led to underfitting of the model.

C. Architecture 2: 3D ConvNet (C3D)

3D ConvNet was proposed in 2014 by Du Tran et al. [2] This architecture deals with studying of frames along with their time distributed nature in the video. C3D treats 3D convolutional networks as feature extractors and uses deconvolutional layers to interpret model decision. Instead of using

default resized frames of size 224x224x3, in this architecture we scaled down those frames by a scale of 2 (112x112x3). We implemented this network in Keras and this network did not have any inception network in its implementation. Since the model was very dense, we used pretrained weights trained on Sports1M dataset and used them for our application of action recognition by initializing with those pre-trained weights and training the network over again.

To study the temporal information of the video, we stacked up frames in a time-distributed manner to feed to our model. C3D focused on spatial appearance in first few frames and tracked the motion in the subsequent frames. Due to limitations of hardware, we could not stack up more than 16 frames per video for training. Once the model was trained, we used this model for prediction of a video and have provided corresponding results. The long range temporal modeling was still a problem. Also, this network is very dense and hence computationally expensive to train.

D. Architecture 3: Two Stream Inflated 3D ConvNet (I3D)

This architecture takes off from where C3D left. This architecture was proposed by Carreira et al. [3] in 2017. I3D deals with continuing the approach of C3D alongwith using optical flows in another similar C3D architecture and taking a consensus in the end. The spatial stream input has frames stacked in the time dimension instead of single frames as in basic two stream architectures. In essence, I3D combines 3D based models into 2-stream architecture. We used pre-trained weights trained on Imagenet and Kinectics dataset for this network in our implementation. We trained 2 separate networks for frames and flows by initializing respective weights. While testing, we performed averaging over the predictions by using frames and flows separately. The simple averaging of the predictions was based on argmax of the softmax output.

I3D used inflated inception module which takes benefit of 2D pre-trained weights. This is basically repeating the 2D weights in 3-dimensions. The two different architectures Frames and Flows are trained separately. Flow architecture required stacking up the flow for 3 dimensions, hence, was very computationally heavy. It required 2552 seconds per epoch to train. Due to time constraints we trained until the Flow architecture obtained 75% validation accuracy.

III. Results

TABLE I
ACCURACY METRIC

DNN Architecture	Accuracy
3D ConvNet	83.2%
Two-Stream Inflated 3D ConvNet	74.5%
Temporal Segment Networks	38.1%

On the testing set the accuracy was in tandem with what we achieved on validation set. Our train/validation/test split was 60:20:20. Training accuracy was the highest for C3D as observed on Test set. I3D implementation is two fold, as we have two different architecture for Frames and Flows. The validation accuracy for Frames was 88.4% and 75.2% for Flows. It is worth noting that our implementation of these architectures have been trained until we obtained validation accuracy above 75% as it can be seen in Fig. 1-3. This decision was taken because training time for any model was very high and incurred significant costs on Google Cloud Platform when using GPUs. The accuracy reported for TSN here is after 25 hours of training that got us to 200 epochs only. We believe, training TSN on more computational power will yield very good results as evident in the literature.

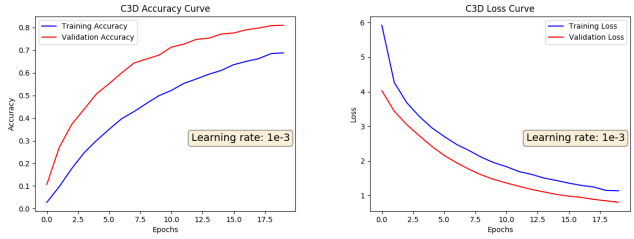

Fig. 1. Curves for C3D

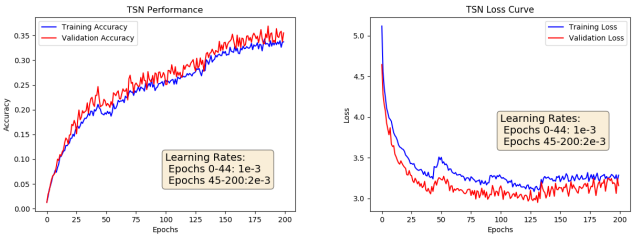

Fig. 2. Curves for TSN

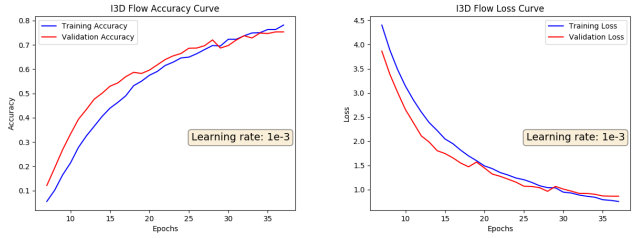

Fig. 3. Curves for I3D Flow

References

[1] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool, "Temporal segment networks for action recognition in videos," *CoRR*, vol. abs/1705.02953, 2017.

[2] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 4489–4497. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2015.510

[3] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4724–4733, 2017.