Human Action Classification Using Two-Stream Convolutional Neural Networks

Akhil Devarakonda, Rahul Zahroof, Zachary Fisher

***Abstract*—In this paper we present our findings in classifying human action in videos found in the HMDB-51 dataset, capturing scenes from movies and YouTube. A two-stream convolutional neural network was implemented utilizing a pre-trained backbone and fusing to explicitly capture spatial and temporal image information. Our experiments vary the dataset format, network input data type, and fusing strategy to yield promising results that in some cases outperform past methods.**

I. Introduction

Human action recognition in videos is a challenging area of research in computer vision with many real-world applications. A variety of architectures have been studied in this area leveraging the rapid boom of research into convolutional neural networks and adopting it to videos. The existence of pre-trained networks on large image datasets has proven to be particularly useful for deploying CNNs on videos. In this paper, we implement various two-stream CNNs described in [1] and [2], which fuse the outputs of a spatial CNN with a temporal CNN to predict the action the human is performing in the video. The spatial CNN is responsible for spatial information in frames of the video while the temporal stream will learn to recognize action from motion. Both streams will leverage the performance of pre-trained networks on large image datasets.

The main contributions of this paper are to study the benefits and trade-offs of the approaches presented in [1] and [2]. Additionally, we evaluate the performance of a simple fully-connected network whose purpose is to fuse the results from the spatial and temporal streams. In section II, we discuss previous work done in the area of human action recognition, with a focus on the techniques in [1] and [2]. In section III, we review the HMDB dataset preparation, provide details about the spatial and temporal stream CNNs, and discuss the various fusion approaches we tested. Finally, we discuss the experiments performed and report our accuracies on the validation set for the various approaches we implemented.

II. Related Work

For this project, we use the Human Motion DataBase (HMDB) dataset from [4]. The entire dataset contains around 7000 manually annotated videos with 51 action categories of humans performing a wide variety of complex human actions. The dataset contains five types of action categories: facial actions (smiling, chewing), facial actions with object manipulation (eating, drinking), body movements (climbing stairs, handstand), body movements with object interaction (hair brushing, golfing), and body movements for human interaction (hugging , punching). While a good place to start for this project, the HMDB dataset is far from capturing the wide variety of human actions.

Capturing complementary information in video is commonly done with the aid of multi-stream networks. A two-stream convolutional network for action recognition in video that incorporates the complementary information of position from individual frames and motion between frames has shown to be more successful than using either of these features alone [1]. While this paper found advantages in training across multiple datasets, the temporal stream typically outperforms the spatial, and in turn, the two-stream out performs the temporal. The fusion of each stream is approached through averaging and SVM which had comparable performance with a slight lead for the SVM.

An extension of this architecture that has yielded positive results is a two-stream inflated 3D network from [3]. This paper considers several methods to implement 3D ConvNets, which improve action classification accuracy significantly by including spacio-temporal filters. The paper introduces a base network, called the Inflation Inception-v1, which expands the two-stream model from 2D to 3D in a very deep architecture. This implementation, in combination with pre-training from the Kinetics Human Action Video dataset, boosts accuracy on the HMDB-51 dataset to up to 80.2%. This particular method is likely outside this project's scope for training times.

Temporal stream inputs can be more tunable than the standard RGB image frames of the spacial stream due to their time varying component. Optical flow is a common way of extracting motion information but can yield different results depending on the number of stacked optical flows (directly related to the number of frames being sample in each video). For example, a network trained on single-frame optical flow resulted in an accuracy 7% below that of a ten-frame optical flow network [1]. Pre-trained networks designed to receive optical flow data are far rarer than those pre-trained on images. As a result, a stacked grayscale 3-channel image (SG3I) can similarly be used to combine information across multiple frames into a single image to get comparable results [2].

III. Method

A. Dataset

Several methods were employed to curate the dataset for our model. PyTorch's HMDB-51 dataset implementation was modified to accept videos with an ID of 0 and 1 in the split file as training data. The default settings in the dataset class (mode 1) were used for training over the full dataset, using

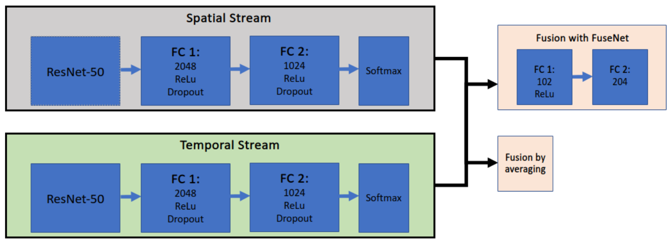

Fig. 1. Network Architectures

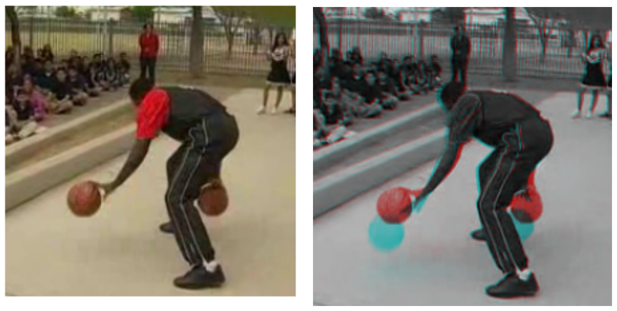

Fig. 2. Single Frame vs. SG3I

every frame in every video as training data. In mode 1, random cropping and random horizontal flipping were used to augment the data. Because of the great amount of points in this dataset, training models in mode 1 took an excessive amount of time (6 hours/epoch for spatial and 10 hours/epoch for temporal). To address this issue, an alternate sampled dataset (mode 2) was curated by sampling the video clips at 6 frames per second and stepping 1 second in between samples. This greatly reduced the size of the dataset, and allowed us to save the dataset in a single tensor to greatly reduce the training time (10 minutes/epoch for spatial/temporal and 1 minute/epoch for FuseNet). In the sampled dataset, the only data augmentation performed was random horizontal flipping.

B. *Spatial Stream*

The spatial stream CNN is essentially responsible for image classification on the still frames of the video. Thanks to the incredible performance of recent large-scale image classification networks, they can be used as a pre-trained backbone upon which the spatial stream can be fine-tuned and further trained for the HMDB dataset.

PyTorch's ResNet-50, pre-trained on the ImageNet dataset, was used as a backbone for the spatial network. The output from ResNet-50 was passed through two fully connected layers with ReLu activation and a dropout ratio of 0.5 to produce a final output from the spatial stream. The spatial stream architecture can be seen in Fig. 1.

C. *Temporal Stream*

The temporal stream consists of the same CNN structure as the spatial's in which a pre-trained ResNet-50 is used as the backbone, followed by two fully connected layers. The architecture can be seen in Fig. 1. Using the HMDB dataset, three images were sampled from each video, converted to grayscale, and stacked together into a 3-channel image. The resulting format represents motion between frames in color and all other static space in grayscale (Fig. 2).

D. *Two-Stream Fusion*

The results from the spatial and temporal streams can be fused to leverage the benefits of each and produce a more accurate prediction of the action being done in the videos. In [1], the authors fuse the streams by trying two methods: averaging and training a multi-class linear SVM. These methods produce similar results, with the SVM showing slightly better accuracy than averaging. In this paper, two simple methods for fusing the results from the spatial and temporal streams were evaluated: averaging and training another network on the two outputs. A simple network, FuseNet, with a hidden fully connected layer using ReLu activation was trained on the outputs of the spatial and temporal streams. It is probable that the spatial or temporal streams are better at predicting certain actions; the goal of FuseNet was to learn these associations and boost the overall prediction accuracy.

E. *Implementation*

A batch size of 128 was used for training spatial and temporal streams. FuseNet models were trained with a batch size of 64. All networks were trained using cross entropy loss; mode 1 stream networks were trained using stochastic gradient descent (SGD) with a learning rate of 10^{-2} and momentum of 0.9. The same parameters were used for FuseNet, except it was trained with an Adam optimizer. For mode 1, the spatial stream and temporal streams were trained for 10 and 15 epochs, respectively. FuseNet was trained for 6 epochs. In mode 2, the spatial stream was trained for 445 epochs using both SGD and AdamW optimizers, temporal for 278 epochs with the SGD optimizer, and FuseNet for 14 epochs with the AdamW optimizer and a learning rate of 10^{-3}. All models were trained using Google Colab Pro.

IV. Experiments

A. *Spatial Stream*

Two spatial stream networks were each trained on the full and sampled datasets according to the details specified in the Implementation subsection. For the full dataset, training was done using mostly identical hyperparameters to the ones reported in [1]. This yielded very good results, which are discussed in the Results section.

B. *Temporal Stream*

The first approach to preparing data for the temporal stream relied on stacking optical flow displacement fields between consecutive frames within a video. An explicit description of what the network should focus on, like optical flow, has been been shown to improve performance. Optical flow information between two frames consists of two channels describing the magnitude and direction of flow at each point in the frame.

An approximate visualization of a single optical flow field is shown in Figure 3; this approach abstracts the content of the original video far more than SG3I since motionless content is not represented at all and the moving regions lose visual definition.

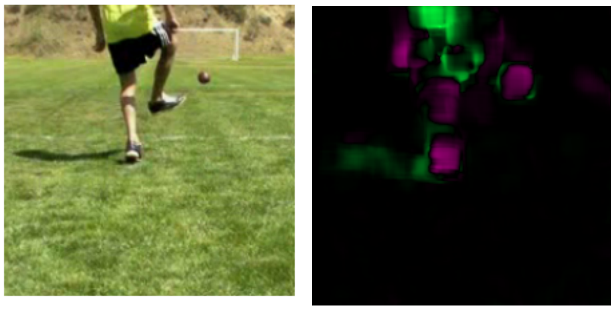

Fig. 3. Optical Flow Visualization

We attempted stacking this information across 10 frames as well as 3 frames per video and fed the information into two different architectures. The first was prescribed in the paper ”Two-Stream Convolutional Networks for Action Recognition in Videos” with 5 convolutions, 2 fully connected layers, and a softmax (with ReLU activation and pooling as prescribed) [1]. The second architecture was our adaption of ResNet-50 to mimic the spatial stream with additional changes needed to the input convolutional dimension based on the optical flow format. In both of these approaches, training failed to yield predictions discernibly better than random guesses from the network. The paper describes performing training across multiple datasets at computational scales beyond our own and the ResNet-50 is of course not pre-trained on optical flow data. We conclude that these factors were the probable causes for these failures and were unable to find a suitable pre-trained network for optical flow. These findings lead to implementing SG3I as outlined in the methods sections to leverage the ResNet backbone and seek novelty.

C. *Two-Stream Fusion*

As described in the Methods section, two approaches were tried for fusing the results from the spatial and temporal streams, namely fusion by averaging and fusion by training a simple network on the results from both streams. Both approaches were implemented for the models trained on the full and sampled datasets.

V. Results

Network	Accuracy
Spatial Stream	50.5%
Temporal Stream	54.2%
Fusion by averaging	57.2%
FuseNet	52.9%

TABLE I
Full Dataset (Mode 1) Accuracy Results

Network	Accuracy
Spatial Stream	53.2%
Temporal Stream	46.3%
Fusion by averaging	55.5%
FuseNet	55.3%

TABLE II
Sampled Dataset (Mode 2) Accuracy Results

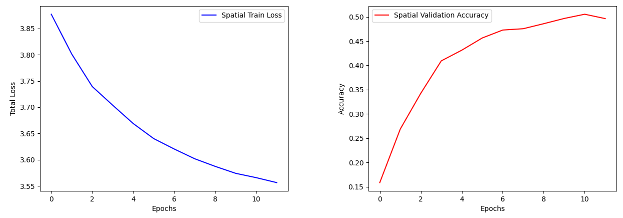

Fig. 4. Spatial Stream Performance (Mode 1)

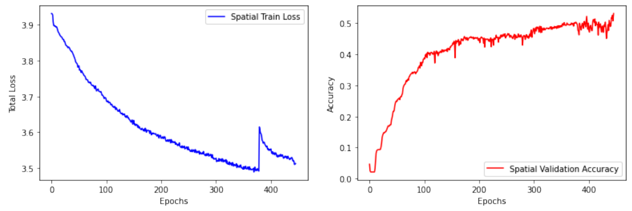

Fig. 5. Spatial Stream Performance (Mode 2)

A. *Spatial Stream*

The mode 1 spatial stream accuracy results after 10 epochs of training proved to be surprisingly good compared to the results obtained in [1]. As shown in Table I, the spatial stream was able to obtain 50.5% accuracy (at epoch 10) which is significantly higher than the 40.5% obtained by the authors in [1]. We suspect this is largely due to the superb pre-trained performance of ResNet-50 used as a backbone for the spatial stream since most of our hyperparameters were similar to the ones used by [1]. The mode 2 results were even better (53.2% as shown in Table II); this discrepancy is likely due to switching optimizers from SGD to AdamW at around 385 epochs, which increased losses but improved accuracy. This switch can be clearly seen in Fig. 5.

B. *Temporal Stream*

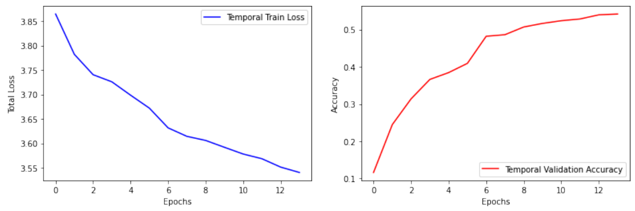

Fig. 6. Temporal Stream Performance (Mode 1)

The mode 1 temporal accuracy of 54.2% was achieved after 14 epochs on the HMDB-51 dataset. The mode 2 temporal

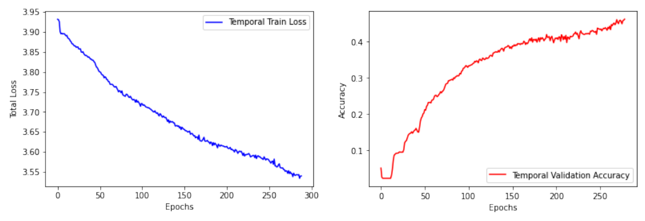

Fig. 7. Temporal Stream Performance (Mode 2)

accuracy was 46.3% after 278 epochs and unlike the spatial stream, saw a lower validation accuracy when using the sampled dataset. The videos were pre-processed using the SG3I format instead of optical flow; however, it is noteworthy that an optical flow approach on the same dataset (without a ResNet backbone) only obtained a 46.6% accuracy as shown in [1]. When compared to other SG3I implementations, our method performed similarly to the 55.1% seen in [2]. While Kim and Won's paper trained on a single SG3I frame per video like ours, their baseline testing included 10 SG3I frames per video instead of our single SG3I frame testing. This advantage of including more information per video is a likely explanation for the small accuracy difference at the cost of 10 times as much input data to the network.

C. Two-Stream Fusion

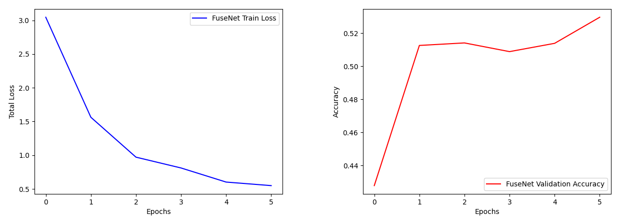

Fig. 8. FuseNet Performance (Mode 1)

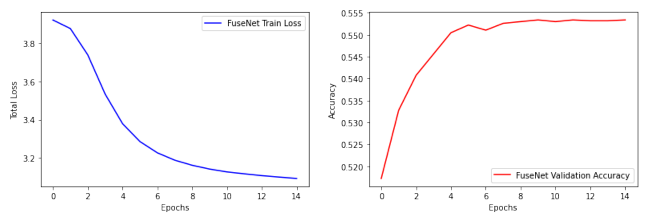

Fig. 9. FuseNet Performance (Mode 2)

Fusion by averaging resulted in higher accuracies than by training FuseNet for both types of datasets. For the full dataset, FuseNet's accuracy was actually lower than the temporal stream accuracy. The FuseNet trained over the sampled dataset performed almost as well as averaging. This is because the FuseNet architecture was modified to remove the hidden layer, essentially performing a weighted average over the label space for both streams. This combated overfitting experienced in the FuseNet trained over the full dataset by reducing the complexity of the network and resulted in higher accuracies, although still not as good as simple averaging.

We believe FuseNet (especially on the full dataset) would have achieved better results than averaging if we had more time for hyperparameter tuning since training a network should weight the two stream outputs for each class, and this should yield better accuracy than just simple averaging.

VI. Conclusions and Future Work

The highest accuracy was obtained by training spatial and temporal streams on the full dataset and fusing the two stream outputs by averaging. This resulted in a final accuracy of 57.2%, which outperforms the same experimental results on the sampled dataset. Unfortunately, this was still lower than the accuracies achieved in both [1] and [2].

We identified several potential reasons why the accuracies of the trained FuseNet models were lower than the averaged results. The first of these is likely related to overfitting. As the training loss decreased and the accuracy over training increased in the FuseNet, validation loss and accuracy quickly plateaued and would sometimes worsen after further training. Removing the hidden layer proved to be an effective method of improving accuracies, but overfitting was still observed relatively quickly in the training process (after about 14 epochs). The AdamW optimizer proved to converge quicker than the SGD optimizer, and sometimes produced better accuracies. Using this method of optimization for all streams may have also improved our performance.

Another potential method of improving results would be to train the spatial, temporal, and fusion components simultaneously. This was not done due to time constraints, but would likely result in slight overall improvements.

The temporal stream resulted in lower-than-expected accuracies for the sampled dataset. To better diagnose this issue, the frames-per-second sample rate and sample-to-sample step size of the dataset could have been modified to include more motion in each SG3I stack. These parameters could have also been better leveraged to lengthen the training dataset for mode 2. Also, testing SG3I vs. optical flow methods in the temporal stream would yield another interesting comparison.

References

[1] Simonyan, Karen, and Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos." Advances in neural information processing systems 27 (2014): 568-576.

[2] J. Kim and C. S. Won, "Action Recognition in Videos Using Pre-Trained 2D Convolutional Neural Networks," in IEEE Access, vol. 8, pp. 60179-60188, 2020, doi: 10.1109/ACCESS.2020.2983427.

[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), Honolulu, HI, 2017, pp. 4724-4733, doi: 10.1109/CVPR.2017.502.

[4] Kuehne, H. et al. "HMDB: A Large Video Database for Human Motion Recognition." IEEE, 2011. 2556–2563. Web. 11 Apr. 2012. © 2012 Institute of Electrical and Electronics Engineers.

[5] Zhou, Brady et al. "Does computer vision matter for action?" Science Robotics 4 (2019): n. pag.

VIDEO:https://drive.google.com/file/d/
1GHzxXxoHO4vXfCMQrzpZL70kYLtiYO3p/view?usp=
sharing