Efficient 3-Stream Recurrent Neural Network for Action Classification

Harsha Chivukula
hchivuku@andrew.cmu.edu

Rishi Khanna
rishik@cmu.edu

Pratik Prakash
pratikpr@andrew.cmu.edu

Markus Woodson
mwoodson@andrew.cmu.edu

Abstract

Recurrent and convolutional networks have been extensively applied to image recognition, image segmentation, natural language processing, and other retrieval tasks. In this work we propose and evaluate several deep neural network architectures that combine image and sound information across time in videos to accurately classify human actions. In our approach we integrate detail, motion, and sound information into our classifier by constructing 3 deep networks and pooling their feature output for classification. We evaluate our results with both convolutional and recurrent networks using Long Short-Term Memory (LSTM) cells. Our 3 stream approach gives us a 74% accuracy on the UCF-101 dataset while both not pre-training on a much larger dataset such as Sports1M or THUMOS and keeping computational expense low.

1. Introduction

Convolutional neural networks have had great success in static image recognition problems such as MNIST, CIFAR, and ImageNet Large-Scale Visual Recognition challenge[15,21,28]. By using a deep neural network, we get both a trainable feature extractor and classifier which are both automatically learned. Recently there has been a surge to apply these networks to other problems in fields such as natural language processing and video classification tasks [3, 4, 5, 6].

Video classification tasks introduce 2 new dimensions to the problem that should be exploited: the time and the sound component. Though we now have new information to exploit, the task itself is very computationally expensive as there may be thousands of frames per video, yet not all of them are useful. An approach that has been used before is to only use the raw frames from the video as input into a CNN in order to classify the video. This not only fails to take advantage of the new information we have obtained, but it also completely disregards the time component of the data.

Thus, we predict that a description of a video does not only need to come from the frames but also from the temporal relation between these frames and the sound associated with each video as well. To that end, we propose a 3-fold network approach using both recurrent and convolutional neural networks. Taking inspiration from previous work [4, 5] which split the processing into 2 streams, a fovea and a context stream, we follow a similar pattern. The CNNs extract the local frame information while the recurrent connections calculate the temporal features. We employ Long Short-Term Memory (LSTM) cells in order to implement a recurrent neural network. LSTMs operate similarly to CNNs but are able to integrate information in time.

Naturally we need to incorporate motion information into our model to achieve better performance. There has been success [5] using optical flow to incorporate motion information into the model. The motion flow serves to provide motion information so we can use a lower fps when training on the raw video frames. However the optical flow was integrated over all 3 channels (RGB). To save computation time and keep the accuracy loss minimal, we computed optical flow on grayscaled versions of the videos.

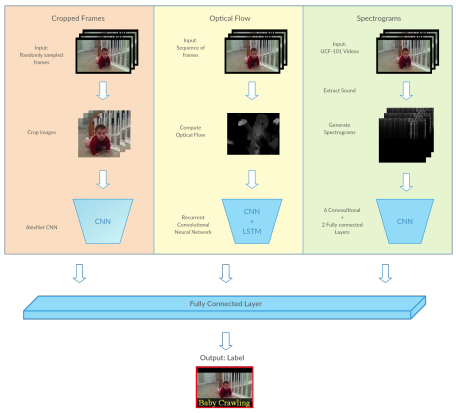

Figure 1. Proposed 3-stream action classification architecture

We incorporated the sound information similarly to the motion and context information by means of a deep neural network. We train on spectrograms of the sound in each video in order to exploit CNN architectures to find complex frequency information that may not be found in a simple ANN. Using the features from sound, motion from optical flow, and context from raw image frames taken from videos, we achieved reasonable performance while still keeping computational intensity modest.

2. UCF-101 Dataset

The UCF-101 Action Recognition Dataset contains 13,320 videos in a total number of 101 action classes divided into five main types covering a wide range of actions: Body-Motion Only, Human-Human Interaction, Human-Object Interaction, Playing Musical Instruments and Sports [1, 4]. We are given three training and testing partitions, to select videos to train and test on, and we follow the given protocol for cross-validation between the three training partitions of videos. Figure 2 shows sample frames from 9 action classes of the UCF-101 dataset [1].

Figure 2. Sample frames from videos of 9 UCF-101 classes

UCF-101 contains videos of length 10-15 seconds on average, and the videos of each class are subdivided into 25 groups of 4-7 videos each. The videos have a fixed frame rate of 25 FPS and frame resolution of 320 x 240 pixels[12]. All the videos also preserve audio, which we plan to leverage in our next steps to form a spectrogram network to learn feature information from the sound of each video as well. UCF-101 is the most challenging action recognition dataset because it features several pattern recognition challenges encountered in video classification. Furthermore, it contains a much larger number of unconstrained clips from YouTube than other action classification datasets[12].

3. Previous Work

Traditional approaches to video recognition have used hand-crafted features such as Histogram of Oriented Gradients (HOG) and Histogram of Optical Flow (HOF) to obtain both spatial and temporal information [15]. CNNs are able to automatically learn such features for action classification in videos, and perform better than traditional approaches [4]. [4] suggests two approaches, namely feature pooling and recurrent neural networks.

Fovea and Context: A topic of interest described by [13] is the pre-processing of frames to speed up computation. In pre-processing, the frames were split into two streams - a fovea stream and a context stream. The fovea stream acted as the human eye's fovea and cropped the middle area of each frame. The context stream subsampled the original image to the same resolution as the fovea stream. The reasoning behind this was the low resolution image could still represent motion information just as well and the middle cropped image would represent the important features of the frame due to camera bias.

Motion and Fusion: To account for the loss of motion information due to processing only one frame per second for a reduced computational intensity, the use of optical flow images computed over adjacent frames is incorporated. Long Short Term Memory (LSTM) is used to counter the problem of standard recurrent neural networks of learning spatio-temporal features over long time periods, by using memory units to alter internal state [4]. [4] uses AlexNet and GoogLeNet to process individual video frames, and we follow the AlexNet CNN architecture due to it having lesser parameters to deal with and a shorter time to train, despite GoogLeNet outperforming AlexNet. These suggested approaches in [4] yield the highest performance measured for the the Sports-1M benchmark when incorporating LSTMs in both image frames as well as optical flow.

The usage of different CNN network structures to describe time information in videos was described by [13]. The authors investigated four different kinds of CNN fusion techniques in an attempt to fuse information over the time domain. The baseline used was a simple single frame network representing a single frame of time. Next, they implemented a technique called Late Fusion where two single frame networks were built 15 frames apart and then merged before the first fully connected layer. Individually the single frame networks were not able to describe any motion but after the first fully connected layer the motion could be described by comparing the outputs of the two networks. The third technique used was called Early Fusion. This was done by modifying the first layer of the single frame network to accept a fourth dimension, T, that represented the number of consecutive frames. This comes at the cost of reduced dimensionality in the actual image. This technique performed the worst excluding the baseline. The last tech-

nique that was implemented was called Slow Fusion. This involved the combination of the Early and Late fusion techniques where each successive layer has access to more and more time information.

Multiple streams: The idea of ensembling multiple networks to obtain a combined feature vector has generally yielded significantly better results. In order to accurately conduct video action classification, both spatial and temporal features should be considered. As a result, [5] proposes a two-stream convolutional neural network which incorporates both spatial and temporal networks. In the temporal network, the team also performs optical flow computations on the raw video frames in order to retrieve the temporal features from the videos. In the spatial network, the team used a CNN on the raw video frames to retrieve spatial features. After retrieving features from both the spatial and temporal dimensions, the team averaged the class score from both streams to classify the action. Fusion by averaging the results of the two streams resulted in an accuracy of approximately 85% [5].

4. Approach

The approach we used in constructing our solution was motivated by the biological principles described in [13]. We construct three deep neural networks representing detail, speech, and motion, and then we pool the features generated by these networks to derive the final classification.

In the human eye, the fovea centralis is responsible for the sharpest, and most focused vision. The first network we construct applies this principle by first pre-processing the frames by cropping the central region then using this as input to the network. We chose to model this network using an AlexNet CNN architecture because AlexNet has a history of performing well on ImageNet[14]. We also test a recurrent architecture using LSTM on the cropped images to introduce time information. We hypothesize that this will lead to moderate improvement in performance as our motion information should pre-dominantly come from optical flow as described below.

Next we attempt to incorporate the sound information of the videos into our model. The sound is extracted from the videos and an image representing the spectrogram is created. We considered implementing this network using an AlexNet architecture however we hypothesize it will sufficient to use a simpler model of a CNN. The spectrogram features are unlikely to be as complex as features that would be found in an image classification network, thus a complex network like AlexNet will not be necessary. A CNN approach is preferred so we can exploit CNN architectures to find complex frequency information that might not be found in a simple ANN.

The last network we construct characterizes the time information of the videos. The optical flow information is derived based on movement in the video. There are two key components of this network that still need to be finalized. Firstly, when representing the optical flow information it is common to use all 3 RGB channels. The main purpose of this approach is to gain context in the motion. But as we are getting our context from our fovea stream we hypothesize it would be sufficient to compute optical flow on grayscale frames. Using this approach we also save computation time and space as our input only has 1 channel now. To encourage the network to find generalized features we sample the video at different frame-rates. Sampling too frequently could potentially lead to learning of features that are too specific so it is likely we will sample at intervals between 6-30 frames apart as described in [4]. The architecture of choice for this stream will be recurrent using LSTM to emphasize the temporal information in optical flow.

For the final step, each of the fully connected layers in the three networks were removed resulting in the output of the networks to be the features that were learned. The three networks were pooled together and all of the outputs were connected to another set of fully connected layers. This new pooled network is what generated the final classification.

5. System Info and Issues

These models were all run on a desktop CPU with a Intel i5-4670K 3.40Ghz CPU, a Nvidia GTX 980 GPU, and 12gB of RAM. This is obviously a rather small system to run deep learning models on compared to much larger systems at other labs. With such a small system we inevitably encountered problems in both the data size and the model complexity. Since one of the goals of the project was to reduce model and computational complexity, the system size was not too much of an issue.

The dataset size remained an issue even after reducing model complexity. For instance, in our fovea stream we take 70 frames per video which results in an overall training size of 4.9GB. Unfortunately this does not fit in GPU memory, so we have to train in batches. We used batch training with 5000 images per batch because it fit comfortably in GPU memory but didn't result in large training times in loading data to and from device memory.

The Neon deep learning framework[2] allowed us to implement batch training for images rather easily. However, Neon did not support training batches of sequences, nor did it support using inference on the sequence level instead of on the frame by frame level. To overcome this we had to edit a bit of the Neon source code by adding a metric for sequence classification and changing the batch iteration code.

6. Optical Flow

Optical flow is important as it encodes the pattern of motion of objects in videos and we use it to obtain the tempo-

ral information in our second motion stream. Optical flow can be represented in multiple ways such as optical flow stacking, trajectory stacking, bi-directional optical flow and mean flow subtraction [5]. The main objective of optical flow is to compute a flow field estimating the motion of pixels in two consecutive image frames [19]. The videos are first sampled at a fixed frames per second (fps), and displacement vectors are calculated between the pairs of consecutive frames t and t+1 [5]. The displacement vectors define how a particular point in frame t moves from frame t to t+1. Motion across a sequence of frames for a particular point is represented by stacking the flow fields.

In our approach, we sampled the videos between 4 fps and 6 fps to obtain a fixed number of successive frames for each video. Optical flow was calculated using 30 successive frames. We leveraged OpenCVs functionality for calculating dense optical flow to get the optical flow images. We used a simple approach for the input to the recurrent neural network by passing in the raw optical flow images. Dense optical flow for all the points in the frame is computed in OpenCV using Gunner Farneback's algorithm, as explained in [21]. The Farneback algorithm estimates displacement by using polynomial expansion to approximate the neighborhood of each pixel [21]. A 2-channel array with optical flow vectors is obtained and the magnitude and direction are calculated, which correspond to the Hue value and Value plane of the images.

7. Convolutional Neural Networks

Convolutional neural networks are a subset of artificial neural networks that use a special architecture well suited for image classification [16]. They incorporate three basic ideas: local receptive fields, shared weights and pooling. A local receptive field is the input window (N x N) that is used for convolution with the input image. In other words, the hidden layer neurons are calculated by sliding the local receptive field across the input image using some predetermined stride length. Each mapping from the input layer to the hidden layer is known as a feature map. CNNs are unique because they use the same (N x N) weights to calculate each of the neurons in the hidden layer, so each neuron in the hidden layer detects the same exact feature but at different locations in the image. Hence, the weights are shared among the neurons. A convolutional layer typically consists of several feature maps.

Usually, a pooling layer immediately follows a convolutional layer. Pooling takes the feature maps calculated in the convolutional layer and outputs a condensed feature map. This paper uses Max Pooling which scans each feature map using a (n x n) window and only outputs the neuron that yields the maximum activation. This saves computation time by further reducing the total number of parameters needed in later layers [17].

The last step in a CNN is a fully connected layer that uses all the neurons in the previous layer to determine which of the 101 classes in UCF-101 the image should belong.

8. Long Short-Term Memory (LSTM)

LSTM is a type of recurrent neural network(RNN) architecture that allows for inclusion of memory within the network. The structure of the network allows for it to learn from a series of inputs which makes it well suited for processing video frames, speech, and other types of time-series data[20].

The main unit in a typical LSTM is the memory cell which is able to remember values for arbitrary amounts of time. These cells are built with with four main components, the cell value, input gate, output gate, and forget gate. For some input vector **x**, the hidden layer $\mathcal{H}$ is then calculated as follows where W is the weight matrix and b_j is the bias in j:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t1} + b_i)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)$$
$$c_t = f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o)$$
$$h_t = o_t tanh(c_t)$$

The value c's emission is determined by the output gate and the cell value is maintained unless it is either increased by the input gate or decreased by the forget gate[4].

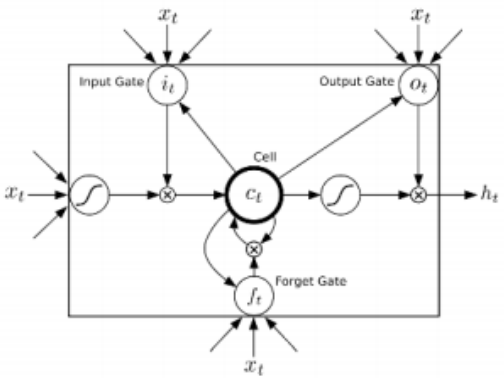

Figure 3. Example of a LSTM memory cell[4]

9. Training

All versions of the fovea stream model were trained with gradient descent with a learning rate of .01 in conjunction with momentum of 0.9 and a weight decay of $(\frac{1}{250})^{1/3}$ which are values taken from the original AlexNet architecture. Both the motion and spectrogram streams are tained

with AdaDelta[9] optimization. We hypothesize that using gradient descent with momentum would probably lead to better performance but due to time constraints we were not able to optimize the hyperparameters as we would have liked and AdeDelta gives rather good performance for not having to tune any parameters. To save time in the training process we loaded parameters from a pre-trained AlexNet model to initialize our fovea stream network.

Each of the 3 models was trained with a pre-defined dataset which is the reccomended method as defined in the UCF-101 website[1]. The fovea stream was trained with both initialization with a pre-trained AlexNet architecture and from scratch to compare performance and computation time. 70 frames were taken randomly from each video in the training set of videos to train the fovea stream. In this way we are reducing the action classification task to a simpler object recognition task which has had great success in recent years[14, 8]. We also randomly transform the input images to increase generalization of the model.

10. Inference

Inference on the fovea stream was done at a frame-by-frame level. Once again, we took 70 random frames from each of the testing videos and then measured the performance per frame. In this way we reduced the action recognition problem to a simple image recognition problem. We then determined the hit-1 and hit-5 across all of the frames. Hit-1 is a measure of overall accuracy and hit-5 is a measure of accuracy given our target class is in the the top 5 predicted outputs. We use a similar inference process for the spectrogram images; the only difference is the amount of images we test on.

Inference for the optical flow stream is done on a video level instead of on the frame-by-frame level. Each frame in the optical flow sequence is fed in temporal order and we get a prediction at each frame. We tried several approaches to combine the predictions from all frames into 1, but in the end, the best performing method was to take the most common prediction among all the frames in the video. Hit-1 and hit-5 are then calculated on the predictions at the video level.

We made sure to note computation time in the training set for all of the networks. We did not measure testing time as inference was rather quick (on the order of a few seconds) compared to the training time.

11. Results

We evaluate our 3 streams independent and combined against the UCF-101 dataset with the goal of investigating not only performance but computational expense. Little testing was done with differing parameters or different fps or frame counts for the archtectures due to time constraints.

Table 1. Results

Method	**Hit@1**	**Hit@5**	**Computation Time (sec)**
Conv pooling + AlexNet	70.4	89.0	N/A
Conv pooling + GoogLeNet	71.7	90.4	N/A
Image and Optical Flow Conv pooling	72.4	90.8	N/A
Image and Optical Flow LSTM	73.1	90.5	N/A
AlexNet Fovea Stream	44	68.7	2339
Fovea Stream	45	69.3	6351
CRNN-RGB	32.1	57.4	11920
CRNN-Grayscale	29.3	56	5105
Spectrogram Network	3	7	50
3-Stream	54	73.2	120

Table 1 compares our results to other popular architectures and approaches.

We can attribute our worse performance to a few factors. For one we did not follow the exact training and testing procedure as others. The reccomended method is to find 3-fold accuracy across 3 different training and test sets. To keep things simple for this project we only evaluated performance on 1 testing set. Another difference between our approach and others is the reduction of computation time. Most other methods used either full frames in the "fovea" stream or when calculating optical flow, used a higher fps, more frames, or all 3 color channels instead of grayscale. All of these factors would lead to better performance with more computation time as the cost.

12. Conclusion

We presented a 3-stream deep learning architecture that efficiently performs action classification tasks by utilizing a combination of convolutional and recurrent neural network architectures. By extracting features from sound, motion from optical flow and context from cropped image frames, we are able to capitalize on all 3 dimensions of videos, resulting in reasonable performance while keeping overall complexity and computation time modest.

Our results indicate that computing optical flow on grayscaled videos over the default 3 color channels led to a negligible decrease in accuracy while cutting computation time in half. This confirms our initial hypothesis that motion features are not influenced by color because motion is captured by integrating the differences in pixels for the raw sequence of frames over time. Furthermore, we discovered the UCF-101 dataset is not well equipped to handle sound extraction. Since only half the classes in the dataset had sound, our spectrogram stream performed worse than we had expected. Without more experimentation on a more capable dataset, we are unable to conclude whether sound features play a significant role in action classification.

In the future we hope to train our architecture on a much larger dataset that contains sufficient training data with respect to all 3 dimensions: context, motion and sound. This enables us to adequately evaluate whether our 3-stream approach is able to compete and surpass other state-of-the-art

action classification architectures. Furthermore, we plan to subsample the motion stream. We hypothesize that computing the difference between pixels over time for optical flow should not depend on resolution of videos, while further reducing the computation time spent training the recurrent neural network. We also hope to try incorporating 3D convolutional neural networks into our architecture because more recent papers have had great success with this technique [18]. 3D CNNs are better suited at video classification tasks than 2D CNNs because they are adept at learning features that model both context and motion simultaneously, while retaining a simple architecture.

References

[1] UCF101 - Action Recognition Data Set, `http://crcv.ucf.edu/data/UCF101.php`

[2] Neon Deep Learning Framework `http://neon.nervanasys.com`

[3] A. Graves, A. Mohamed and G. E. Hinton. Speech Recognition with Deep Recurrent Neural Networks.

[4] J. Y. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, G. Toderici. Beyond Short Snippets: Deep Networks for Video Classification.

[5] K. Simonyan and A. Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos.

[6] A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *ICML*, pages 17641772, Beijing, China, 2014. 2

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, pages 10971105, Lake Tahoe, Nevada, USA, 2012. 1, 2, 3

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich. Going Deeper with Convolutions

[9] M. D. Zeiler. Adadelta: An Adaptive Learning Rate Method

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CoRR*, abs/1409.4842, 2014. 1, 3, 4

[11] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818833, Zurich, Switzerland, 2014. 1, 3

[12] K. Soomro, A. R. Zamir and M. Shah. UCF101: A Dataset of 101 Human Action Classes From Videos in The Wild. CRCV-TR-12-01, November, 2012.

[13] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar and L. Fei-Fei. Large-scale Video Classification with Convolutional Neural Networks. In *CVPR*, 2014.

[14] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." Web.

[15] I. Laptev, M. Marszaek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In Proc. CVPR, pages 18, Anchorage, Alaska, USA, 2008. 2, 3

[16] Zeiler, Matthew D., and Rob Fergus. "Visualizing and Understanding Convolutional Networks." Computer Vision ECCV 2014 Lecture Notes in Computer Science (2014): 818-33. Web.

[17] Nielsen, Michael. "Deep Learning." Neural Networks and Deep Learning. N.p., Aug. 2015. Web. 10 Dec. 2015.

[18] Tran, Du, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. "Learning Spatiotemporal Features with 3D Convolutional Networks." (n.d.): n. pag. 7 Oct. 2015. Web. 10 Dec. 2015.

[19] C. Zach, T. Pock, and H. Bischof. A Duality Based Approach for Realtime TV-L Optical Flow. 2007.

[20] H. Sak and A. W. Senior and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. Proc. Interspeech, pp338-342, Singapore, Sept. 2014

[21] G. Farneback. Two-frame Motion Estimation Based on Polynomial Expansion. 2003.