Combining Multiple Sources of Knowledge in Deep CNNs for Action Recognition

Eunbyung Park, Xufeng Han, Tamara L. Berg, Alexander C. Berg
University of North Carolina at Chapel Hill
{eunbyung,xufeng,tlberg,aberg}@cs.unc.edu

Abstract

Although deep convolutional neural networks (CNNs) have shown remarkable results for feature learning and prediction tasks, many recent studies have demonstrated improved performance by incorporating additional hand-crafted features or by fusing predictions from multiple CNNs. Usually, these combinations are implemented via feature concatenation or by averaging output prediction scores from several CNNs. In this paper, we present new approaches for combining different sources of knowledge in deep learning. First, we propose feature amplification, where we use an auxiliary, hand-crafted, feature (e.g. optical flow) to perform spatially varying soft-gating on intermediate CNN feature maps. Second, we present a spatially varying multiplicative fusion method for combining multiple CNNs trained on different sources that results in robust prediction by amplifying or suppressing the feature activations based on their agreement. We test these methods in the context of action recognition where information from spatial and temporal cues is useful, obtaining results that are comparable with state-of-the-art methods and outperform methods using only CNNs and optical flow features.

1. Introduction

As deep convolutional neural networks (CNNs) have shown remarkable performance for computer vision tasks on static images[11, 2, 15], there has been interest in applying deep CNNs to video generally and to human action recognition in particular. However, for action recognition, the discriminative performance of spatio-temporal features learned by deep CNNs has fallen short compared to accuracy gains seen in the image domain[10, 24]. There are two potential reasons for this: 1) existing datasets are not large or representative enough to learn robust features, and 2) it is difficult to either design or train networks to learn spatial and temporal features simultaneously. In order to overcome these challenges, recent studies have proposed methods to incorporate hand-crafted features in combination with traditional CNN based approaches that operate directly on raw pixels from video frames[18, 28, 13, 27, 25].

Notable progress was made by Simonyan et al[18] who proposed a two-stream network architecture designed to mimic the pathways of the human visual cortex for object recognition and motion recognition. One stream of their approach used a network focused on learning spatial feature from RGB input images (spatial network), while the other stream used a network focused on learning temporal features from optical flow inputs (temporal network). Final predictions were computed as an average of the outputs of these two networks, demonstrating improved performance over single stream approaches. Results indicated that errors made by the two networks were tuned to the particular input feature type, with the temporal network confusing different classes with similar motion patterns and the spatial network confusing different classes with the same depicted objects (e.g. human's face).

Given these results, we believe that the two-stream approach is promising, but could be strengthened by enhancing each network with knowledge from the other stream. For the spatial network, it could be useful to know which parts of the image are moving. For example, the spatial network could be improved by knowing that the hands and hammers are moving (in a hammering activity) so that the network could focus on these portions of the frame rather than other non-essential parts of the frame. On the other hand, for the temporal network, it would be useful to have knowledge about "what" is depicted in the moving parts of a video. For example, if the temporal network knew that the repetitive up-down motion was a hammer, it could more reliably label the video as hammering. Therefore, rather than simply averaging the final outputs from the two networks, we propose to incorporate this knowledge directly into the networks to allow for earlier cross-communication of knowledge between the two streams.

In this paper, we present two ways of combining spatial and temporal cues in deep architectures for action recognition. First, we propose feature amplification, where we use an auxiliary, hand-crafted, feature (e.g. optical flow) to perform spatially varying soft-gating on an intermediate CNN feature map. In particular, we amplify the activations of the

last convolutional layer of a deep network by the magnitude of optical flow, allowing a previously spatial-only network to understand which parts of image are moving before the fully-connected layers are evaluated. Second, we present a spatially varying multiplicative fusion method for combining spatial feature maps (e.g. the last convolutional layers) from multiple CNNs trained on different sources. This method amplifies or suppresses activations based on the agreement between the networks in each part of the image. Experimental results on two popular action recognition datasets, UCF101[21] and HMDB51[12], show that this outperforms previous additive fusion methods.

2. Related Work

There have been many approaches for applying deep learning methods to video classification. 3D CNN was proposed to train networks to learn spatio-temporal features[7, 24]. [10] collected a large scale You-tube video dataset and applied several variants of CNN architectures, including early fusion, late fusion, and 3D CNN. [18] proposed a two-stream network to decouple spatial and temporal features. Some fusion methods based on fully connected layers have been suggested to improve video classification[9, 26]. Recently, efforts to learn long-term temporal features have been proposed to train recurrent neural networks (e.g. LSTM[6]) [1, 26] using hand-crafted features (as opposed to raw pixels) as inputs[25, 27].

Multiplicative interactions have been explored in several places. [23] introduced a bilinear model that has multiplicative interaction between two factors, such as content and styles. [20] proposed a point-wise gated boltzmann machine (PGBM) that can distinguish task-relevant features from distracting task-irrelevant features by introducing switching units that have multiplicative interactions between visible and hidden units. [22] introduced a multiplicative RNN which considers the multiplicative interaction between the current input and previous outputs of hidden units. Additionally, many variants of gated RNNs, such as LSTM, have introduced multiplicative gates to control the amount of information that can be passed to successive layers.

Bilinear CNNs were proposed very recently [14], using two CNNs pre-trained on the ImageNet dataset and fine-tuned on fine-grained image classification datasets. Their combination method extracts features from the last convolutional layer of each network and performs a dot-product between all possible pair-wise interactions to get a feature vector. Then they add one final fully-connected layer on top of this for classification. This means that their model is a linear combination of all possible pair-wise interactions between feature maps. In terms of model capacity, their method is similar to our multiplicative fusion method. However, their method is less scalable because the size of the feature vectors are exponential with respect to the number of feature maps, whereas ours does not depend on the size of feature maps and the number of networks. Additionally, their networks are both trained for the same object classification task, whereas our two-stream approach can incorporate two explicitly different types of features, e.g., spatial and temporal, into the classification process.

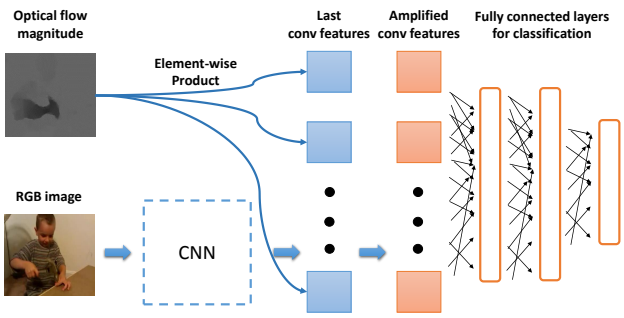
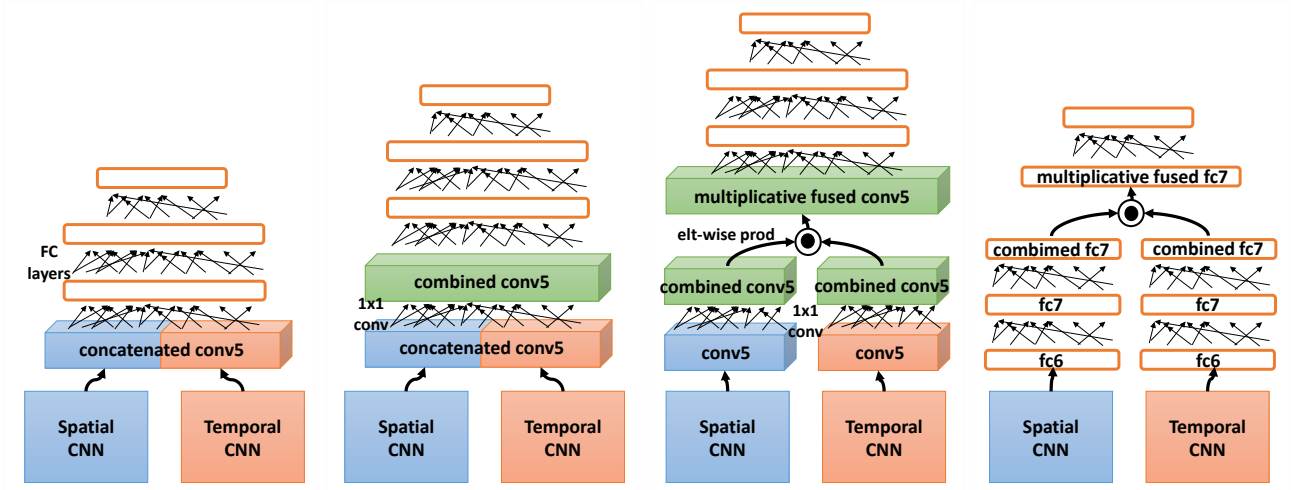


Figure 1: Feature amplification with optical flow magnitude

3. Feature Amplification

CNNs have been successful for feature learning and selection that directly affect the discriminative power of classifiers for many computer vision tasks. Especially for the image classification task, where CNNs have outperformed hand-crafted features by a significant margin, extremely good performance has been achieved using only categorical labels for images, leaving the CNNs to learn everything else such as objects, parts of objects, background, etc. It is widely accepted that large-scale datasets and high computational power have played an important role in these achievements. Recent work has shown that CNNs can even recognize and take advantage of characters on signs of store for store front classification in large scale street view datasets without any supervision about where the text is located or what type of characters are depicted[16].

However, collecting large-scale labeled datasets is very expensive. In some cases, such as the video domain, the situation is even more challenging. However, without a large enough dataset, CNNs may fail to learn and extract useful features since they could be easily distracted by noisy and unimportant features. They may also overfit to noisy and small training datasets, making generalization to the real world infeasible. For example, for action recognition datasets in the video domain, such as UCF101, we found out that existing spatial networks can easily be trained to achieve 100% accuracy on training dataset, but with relatively lower accuracy on testing dataset. In addition, training deep and large CNNs on multiple frames for learning



(a) Baseline A (b) Baseline B (c) M-fusion(conv5) (d) M-fusion(fc7)

Figure 2: Various fusion alternatives

spatio-temporal features is also very expensive even with GPU and distributed computing frameworks[10]. Many studies try to reduce computational cost by collapsing temporal information after the first convolutional layer, considering only short-term temporal information (e.g. 10 or 20 frames), or ignoring temporal information completely.

The proposed feature amplification aims to alleviate both problems by introducing additional supervision along with the category label during training of spatial networks (Figure 1). First, we extract features from the last convolutional layer of the spatial network. We then compute optical flow magnitudes given a pair of consecutive frames and resize it to be the same size as the feature maps in the last convolutional layer (13x13 or 14x14 depending on the architecture). Then, we simply perform element-wise product to amplify feature activations and fine-tune the rest of fully connected layers. Optical flow magnitude is scaled from 1 rather than 0 so as not to zero out other parts of the image features which might also be important for classification. By doing this, we are able to reduce the burden of learning temporal features, which results in lower computational cost. Furthermore, it provides hints about which parts of images are important, allowing us to learn better CNNs from smaller datasets.

4. Multiplicative fusion of multiple CNNs

In addition to our feature amplification technique which directly enhances the spatial network CNN, we also develop a method to combine multiple networks. In this paper, we use this to combine networks trained on temporal and spatial information, but this could be applied more generally to combine any given set of networks. One straightforward way of combining multiple networks is to add additional fully connected layers to combine the outputs (or internal layers) of the networks. This has an additive property between different feature activations and has been successfully adopted in several applications, such as similarity metric learning[4]. We, however, found that this approach tends to suffer from overfitting when applied directly to the spatial and temporal networks[18]. Motivated by the promising performance of our feature amplification technique, we propose a new multiplicative fusion method. The proposed method aims to find important features of each network and give a higher score to the prediction only when multiple networks agree with each other. For example, activations of feature maps from convolutional layers will be amplified if both networks produce high activations at the same location, otherwise they will be suppressed.

4.1. Formulation

In this section, we introduce a formulation of our proposed fusion method. Suppose we are given two CNNs, for instance, a spatial network and a temporal network. Let matrix $A \in \mathbb{R}^{d \times M}$ and $B \in \mathbb{R}^{d \times N}$ be extracted features of the last convolutional layer from each network. M and N are the number of feature maps which can vary according to the particular CNN architecture (in our experiments 512 for conv5). d is the size of feature maps, which is usually 13×13 or 14×14 (7×7 or 6×6 after last max pooling). $\mathbf{a_i}$ and $\mathbf{b_i}$ are the ith column of each matrix, which corresponds to one feature map. The output of our proposed fusion C is

then:

$$\mathbf{c_k} = \left(\sum_{i=1}^{M} \alpha_{ki}\mathbf{a_i} + \gamma_k\right) \odot \left(\sum_{j=1}^{N} \beta_{kj}\mathbf{b_j} + \delta_k\right) \quad (1)$$

where $\odot$ is element-wise product (this could be replaced with dot product if spatial information is deemed less important), γ and δ are bias terms (we fix them to 1 for our experiments so as to not-zero out other input features). α and β are weights for each feature map and learnable parameters. This can also be extended to fuse fully connected layers. Unlike in the convolutional layer, for the fully connected layer, d is 1, so the dimensions of A and B will be $1 \times M$ and $1 \times N$ respectively.

The α and β will play a key role to select good features in each network, giving higher weights to features that are useful for prediction. K is the only additional hyper-parameter we introduce, where a larger value means larger capacity of the fused networks. From our experiments, we find that 2048 was large enough to successfully fuse the spatial and temporal networks. Note that this method is scalable with respect to the number of networks that we are trying to combine, since the size of the fused network depends on K rather than the number of networks.

The proposed fusion layer can be trained with standard back-propagation and stochastic gradient descent. Taking derivatives of this layer is straightforward and can be easily plugged into many popular CNN software platforms, such as *caffe*[8]. In practice, it can be easily implemented in two steps, 1×1 convolutions followed by element-wise product (Figure 2 illustrates the fusion process).

5. Experiments

5.1. Datasets

UCF101 dataset[21] has been widely used by action recognition studies. There are total 13320 short video clips and each clip present only one (repetitive) action out of the 101 categories of human actions. The HMDB51 dataset[12] is another popular action recognition dataset that was collected from movies and internet videos. The dataset has 6,766 short video clips and 51 action categories. Both datasets provide standard 3 train/test splits and final scores are normally reported as the average of the accuracies obtained for each split. We decode all videos into static images for entire frames with ffmpeg library and store the frames compressed in jpeg files. We resized all frames to a fixed size that makes the smallest side of the frame 256 pixels long.

5.2. Implementation details

Optical Flow. We strictly followed the directions in [18] for fair comparison. Using the pre-decoded images, we compute optical flow between pairs of consecutive frames. For optical flow magnitude, we computed euclidean norm, quantized them into natural numbers, and saved them in jpeg files. For optical flow, we concatenated x and y optical flow into one jpeg file, which reduces the number of file operations when we provide data into the network.

Training. All spatial networks were pre-trained on the ImageNet Challenge classification dataset. All training was performed in the caffe[8] framework with relatively simple modifications. Dropout 0.7 and 0.5 were evaluated and we empirically found that 0.7 ratio gave slightly better results. For every iteration, we applied random flipping and cropping, a standard procedure for data augmentation. Weight-decay and momentum were set to 0.0005 and 0.9 respectively. For training each spatial and temporal network, We followed the learning rate policy in [18]. For fusion networks, batch size was set to 128 and initial learning rate was set to 10^{-3}. After 20K iterations, it was changed to 10^{-4}, then changed again to 10^{-5} after another 20K iterations. For multiplicative fusion networks, we sample one image from the video as an input to the spatial network, and selected 10 consecutive optical flow frames starting at the point we sampled the image for the spatial network.

Testing. We also adopted the approach in [18] for testing. Given a video, we sample 25 frames with equal temporal spacing between them. For each frame, we computed 10 CNN features by cropping and flipping four corners and the center of the frame. The average of total 250 scores was used for final prediction.

	UCF101		HMDB51	
models	base	amp	base	amp
T[18]	80.3	–	47.3	–
S[18]	70.6	**75.0**	35.4	**38.0**
S(VGG19)[19]	78.8	**81.0**	40.3	**44.9**
S + T	85.0	**85.5**	50.6	**51.2**
S(VGG19) + T	87.8	**88.5**	50.1	**54.5**

Table 1: Classification accuracy on the UCF101 and HMDB51 dataset with amplified spatial network (for 3 standard train/test splits). '+' indicates taking the average of the l2-normalized softmax scores of two networks[18], S and T stands for spatial and temporal networks respectively, and 'amp' indicates the use of our feature amplification method.

5.3. Results of feature amplification

We adopt two widely used CNN architectures for the spatial networks, one from [18] and the other from very deep CNNs [19], both pre-trained on 1000 categories from the ImageNet Challenge dataset. We then fine-tune the fully connected layers using our optical-flow based feature amplification approach. Table 1 shows that this simple trick

Figure 3: Effects of feature amplification in UCF101 dataset. Top row shows sample images from videos where our amplified spatial networks provide a correct activity classification while the original spatial network's predictions are incorrect (from left to right, predictions by the amplified network vs original network are ['jumping jack', 'boxing punching bag'], ['tennis swing', 'trampoline jumping'], ['playing daf', 'drumming'], and ['head massage', 'mopping floor']). Bottom row shows example images from the incorrect activity class predicted by the original spatial network. The middle row visualizes the optical flows y-axis images corresponding to top row images.

achieves significant improvement in accuracy of the spatial network. Rows 2 and 3 of the table show performance for the [18] and [19] networks respectively, demonstrating improvements from an average of 70.6% accuracy to 75% accuracy(35.4% accuracy to 38% for HMDB51) for the baseline vs feature amplified versions of the [18] network and 78.8% to 81.0%(40.3% accuracy to 44.9% for HMDB51) for the [19] network. Subsequently this also results in superior two-stream classification accuracy when combined with the temporal network (base vs amp columns of rows 4 and 5 respectively).[1]

We also perform a qualitative study to analyze effects of the amplified spatial network. First, we select some classes where the amplified spatial network provides the correct answer while the original network produces the wrong answer. Then, we study why the original spatial networks failed to produce the right answer.

Figure 3 shows some examples in UCF101 dataset. In the first column, the original spatial network predicts the video to show the action 'boxing punching bag' rather than 'jumping jack' since there are several punching bags in the image in question. However, the amplified spatial network is able to predict the right answer because it focuses on the moving portions of the image rather than the punching bags. We observe similar patterns in other examples. In the second column, the amplified spatial network can focus on the human and tennis racket while the original network misclassifies the action as 'trampoline jumping' since the trampoline and tennis court look visually similar. In the third column, there are drums in the sample image, confusing the original spatial networks, but not the amplified network. Lastly, the amplified spatial network is able to focus on the hands and head in the last column, enabling a correct classification of 'head massage'.

5.4. Results of multiplicative fusion

We evaluate our multiplicative fusion method compared to baseline fusion methods. Figure 2 shows various alternatives. First, baseline A represents a straight way of combining two CNNs, putting a fully connected layer on top of concatenated conv5 features followed by a few of fully connected layers for classfication. For baseline B, we combine networks as a linear combination of concatenated conv5

[1]Pretrained spatial and temporal networks of [18] are not publicly available. We did our best to reproduce their result and put the numbers from our version, which are slightly lower than original performance. In addition, [18] applied multitask learning approach witn combined both UCF101 and HMDB51 datasets when they trained networks for HMDB51 dataset for better accuracy while we only used HMDB51 dataset.

	UCF101		HMDB51	
methods	basic	VGG19	basic	VGG19
baseline A	75.3	82.2	37.4	36.9
baseline B	76.0	81.0	39.2	38.7
m-fuse(conv5)	**82.1**	**84.4**	**51.6**	**52.7**
m-fuse(fc7)	**83.4**	**87.6**	**52.0**	**53.3**

Table 2: Classification accuracy of fusion networks on UCF101(split1) and HMDB51(split1). The 'basic' results use the spatial network from [18], 'VGG19' results use [19]. Note that using the more advanced spatial model already significantly improves performances, but our proposed fusion combination techniques improve both versions of the networks. See Fig. 2 and Sec.5.4

models	UCF101	HMDB51
S + T	85.0	50.6
S(VGG19) + T	87.8	50.1
S + T + m-fusion	86.0	52.7
S(VGG19) + T + m-fusion	88.3	54.4
S(amp) + T + m-fusion	**88.9**	**56.2**
S(amp) + T + m-fusion(fc7)	**89.1**	**54.9**

Table 3: Classification accuracy on UCF101 and HMDB51 using various combinations of spatial (S) and temporal (T) streams. The baseline S and T implementations were trained following [18]. 'm-fusion' stands for multiplicative fused network and 'amp' uses the VGG19 spatial network with feature amplification. Combining our amplification technique for gating the spatial network with multiplicative fusion in the last convultional layer or in fc7 led to the best results. Details in Sec.5.4.

features, which can be easily implemented in the form of 1x1 convolution plus a few of fully connected layers for classification. M-fusion(conv5) and M-fusion(fc7) are our proposed multiplicative fusion methods on conv5 and fc7 features respectively.

Table 2 shows performance of various fusion methods. For UCF101 dataset, we achieve performance gains from 75.3% (baseline A) or 76% (baseline B) to 82.1% and 83.4% for our proposed fusion methods on the network from [18]. For the network from [19], we see improvements from 82.2% (baseline A) or 81% (baseline B) to 84.4% and 87.6% by our proposed fusion. This is potentially because our method can avoid overfitting due to the fact that only pairs of features that two network have agreed upon can contribute to the final classification. In other words, many of the features that could lead to do overfitting were effectively suppressed by the multiplicative operation. Note that the over-fitting problem for baseline fusion methods with HMDB51 dataset are very serious because of the smaller size of the dataset. With our multiplicative fusion technique, we can achieve significant performance gains from 39.2% to 52.0% on the network from [18], 38.7% to 53.3% on the network from [19].

We also performed a qualitative study to analyze the impact of multiplicative fusion networks in similar way in section 5.3. Figure 4 shows some examples in UCF101 dataset. In the first row, the correct action class was 'Archery', which is very difficult to identify since the person and bow are far away in the image making them quite small in size. Thus, the spatial net predicted 'CliffDiving' which has a similar background(the third column in the first row). The temporal net also confused this example with 'GolfSwing' since the motion pattern of arrows and golf clubs are similar(the second column in the first row). In addition, both networks are very confident in their beliefs, so the two-stream network using averaging based fusion also provided the wrong answer. Our method was able to predict the correct activity with high belief since it could effectively suppress the background feature activations and amplify feature activations for the arrow. The second row shows another example. Here both the temporal and spatial networks predict the same incorrect answer ('HighJump' vs the correct answer of 'JavelinThrow') with high confidence. Therefore, the averaging based fusion also produces this incorrect answer. However, our method predicts 'HighJump' with strong belief by effectively selecting the moving pole as one of the important features.

However, We noticed that many action classes are easily classified by either the static visual information or motion information alone. Therefore, we simply performed a weighted average of the m-fusion, spatial, and temporal network predictions together (empirically, 2:3:4 was good ratio for all experiments) to make our final prediction. This provides superior results to the two-stream networks (Table 3).

Overfitting and regularization. It turns out that multiplicative fusion works as well as a regularizer. When it comes to regularization of deep CNNs, weight-decay and dropout[5] are easily applicable and commonly used. We have tried to make our baseline methods avoid overfitting by using these techniques, but even with aggressive weight decay and dropout we still observed low performance on the testing dataset.

Finetuning. Our proposed methods is a simple multiplication between a linear combination of previous layers. One may argue that the linear combination layer before the multiplication may not be necessary since previous layers should already perform this function (e.g. the convolution operation can be interpreted as linear combination with convolutional filters). However, from our experimental results, fine-tuning the layers before the fusion layer was not very

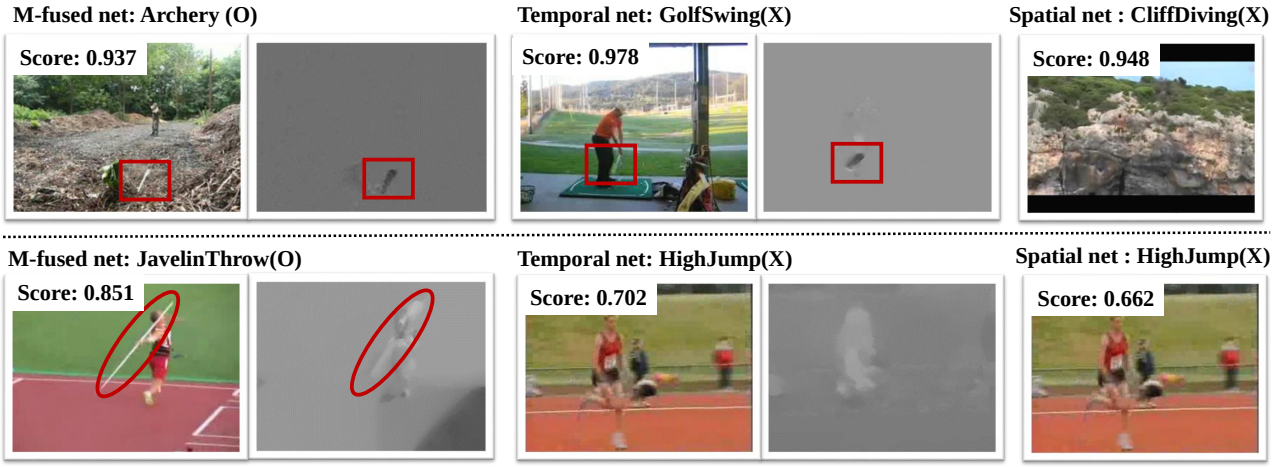


Figure 4: Examples of the results with multiplicative fusion network in UCF101 dataset.

helpful. Therefore, we fine-tuned the layers only after the fusion layer for all of our experiments, which is the case where the linear combination layer plays a very important role. Normalization after or before the fusion layer might help to fine-tune all the way down to the first convolutional layer. We also might get benefits from training whole fusion networks from the scratch rather than fine-tuning from the pre-trained networks.

5.5. Performance comparison to other methods

models	accuracy
Twostream with extra data[18]	86.9
Twostream with extra data, SVM fusion [18]	88.0
Twostream, regularized fusion[26]	88.4
Twostream, regularized fusion , LSTM[26]	91.3
CNN, IDT, FV[28]	89.6
CNN, optical flow, LSTM [27]	88.6
IDT, FV, temporal scale invariance[13]	89.1
CNN, IDT, FV, trajectory, SVM[25]	91.5
Ours	89.1

Table 4: Performance comparison to state-of-art results on UCF101. Our results are the best of the two stream approaches that do not add an extra LSTM stage, and compares favorably to the state of the art that adds many additional, somewhat complex, stages to processing.

Table 4 shows state-of-art methods on UCF101 dataset. Among the methods [18, 26] that only depend on convolutional networks, we achieved the best result. Also note that we didn't use any extra data that might lead us to have better result, e.g. multi-task learning for training the temporal network[18]. Another straightforward way to improve performance would be to combine various hand-crafted features with CNN features, which might also be helpful for our method[28]. Several recent works have also considered longer temporal information while our CNN feature only contains short temporal information. For example, [26] trained an LSTM over entire frames of video, [27] proposed various pooling methods on LSTMs, and [13] considered temporal scale invariance. All of these could also incorporated directly into our method.

6. Conclusion and Future Work

In this paper, we proposed new ways of combining knowledge in convolutional networks for action classification. Simple feature amplification for spatial networks using optical flow features yeilded significant improvement in accuracy over the original spatial networks. In addition, we proposed a multiplicative fusion approach to combine multiple CNNs, which also demonstrated better performance compared to normal additive fusion with fully connected layers. Lastly, using deeper and larger networks, which is a straightforward way to improve performance, also worked well as expected. When we combine all of these ideas together, we achieve superior results on UCF101 and HMDB51 datasets compared to previously proposed two-stream CNNs.

As commercial depth sensors become easily available, understanding the visual world via RGB-D images has received a lot of attention. State-of-art object detection and semantic segmentation methods for RGB-D data have extensively used multiple stream convolutional networks[3,

15]. Each stream takes static images and hand-crafted depth feature images as the inputs respectively, and the simple averaging late fusion approach was used for final prediction. Given the promising evidence in this paper, we believe that our proposed method could also improve performance on RGB-D data.

Even if each network is trained on the same input modality, it is known that each network converges to different local minima. Since each local minima has slightly different knowledge, it has been shown that performance increases when combining multiple networks together with simple late fusion approach. For example in the ILSVRC image classification challenge[17], all winning methods have used CNN ensemble approaches. As future work, we plan to apply our methods to multiplicatively combine multiple CNNs for the image classification task.

References

[1] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, June 2015.

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, June 2014.

[3] S. Gupta, R. Girshick, P. Arbelaez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *ECCV*, September 2014.

[4] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *CVPR*, June 2015.

[5] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, July 2012.

[6] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, November 1997.

[7] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *TPAMI*, January 2013.

[8] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, June 2014.

[9] Y. Jiang, Z. Wu, J. Wang, X. Xue, and S. Chang. Exploiting feature and class relationships in video categorization with regularized deep neural networks. *arXiv preprint arXiv:1502.07209*, Feburary 2015.

[10] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, June 2014.

[11] A. Krizhevsky, S. Ilya, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, December 2012.

[12] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, November 2011.

[13] Z. Lan, M. Lin, X. Li, A. G. Hauptmann, and B. Raj. Beyond gaussian pyramid: Multi-skip feature stacking for action recognition. In *CVPR*, June 2015.

[14] T. Lin, A. RoyChowdhury, and S. Maji. Bilinear CNN models for fine-grained visual recognition. *arXiv preprint arXiv:1504.07889*, April 2015.

[15] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, June 2015.

[16] Y. Movshovitz-Attias, Q. Yu, M. C. Stumpe, V. Shet, S. Arnoud, and L. Yatziv. Ontological supervision for fine grained classification of street view storefronts. In *CVPR*, June 2015.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

[18] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, December 2014.

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. May 2015.

[20] K. Sohn, G. Zhou, C. Lee, and H. Lee. Learning and selecting features jointly with point-wise gated boltzmann machines. In *ICML*, June 2013.

[21] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, December 2012.

[22] I. Sutskever, J. Martens, and G. Hinton. Generating text with recurrent neural networks. In *ICML*, June 2011.

[23] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural Compututation*, June 2000.

[24] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3D: generic features for video analysis. *arXiv preprint arXiv:1412.0767*, December 2014.

[25] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, June 2015.

[26] Z. Wu, X. Wang, Y.-G. Jiang, H. Ye, and X. Xue. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. *arXiv preprint arXiv:1504.01561*, April 2015.

[27] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, June 2015.

[28] S. Zha, F. Luisier, W. Andrews, N. Srivastava, and R. Salakhutdinov. Exploiting image-trained CNN architectures for unconstrained video classification. *arXiv preprint arXiv:1503.04144*, March 2015.