# Actions ∼ Transformations

Xiaolong Wang[1] Ali Farhadi[2,3] Abhinav Gupta[1,3]
[1]Carnegie Mellon University [2]University of Washington [3]The Allen Institute for AI

## Abstract

*What defines an action like "kicking ball"? We argue that the true meaning of an action lies in the change or transformation an action brings to the environment. In this paper, we propose a novel representation for actions by modeling action as a transformation which changes the state of the environment before the action happens (precondition) to the state after the action (effect). Motivated by the recent advancement of video representation using deep learning, we design a Siamese network which models the action as the transformation on a high-level feature space. We show that our model gives improvements on standard action recognition datasets including UCF101 and HMDB51. More importantly, our approach is able to generalize beyond learned action categories and shows significant performance improvement on cross-category generalization on our new ACT dataset.*

## 1. Introduction

Consider the "soccer kicking" action shown in Figure 1. What is the right representation for the recognition of such an action? Traditionally, most research in action recognition has focused on learning discriminative classifiers on hand-designed features such as HOG3D [18] and IDT [43]. Recently, with the success of deep learning approaches, the focus has moved from hand-designed features to building end-to-end learning systems. However, the basic philosophy remains the same: representing action implies encoding the appearance and motion of the actor. But are actions all about appearances and motion?

We argue that the true essence of an action lies in the change or the transformation an action brings to the environments (and most often these changes can be encoded visually). For example, the essence of "soccer kicking" lies in the state change of the ball (acceleration) caused by the leg of the player. What if we try to represent actions based on these changes rather than appearance and motion?

In this paper, we propose representing actions as transformations in our visual world. We argue that current action recognition approaches tend to overfit by focusing on scene context and hence do not generalize well. This is partly because of the lack of diversity in action recognition datasets compared to their object recognition counterparts. In this paper, we overcome this problem by forcing a representation to explicitly encode the change in the environment; the inherent reason that convinced the agent to perform the action. Specifically, each action is represented as a transformation that changes the state of the environment from what it was before the action to what it will be after it. By borrowing the terminology from NLP, we call the state before the action as the precondition state and the state after the action as the effect state. We build a discriminative model of transformation by using a Siamese network architecture

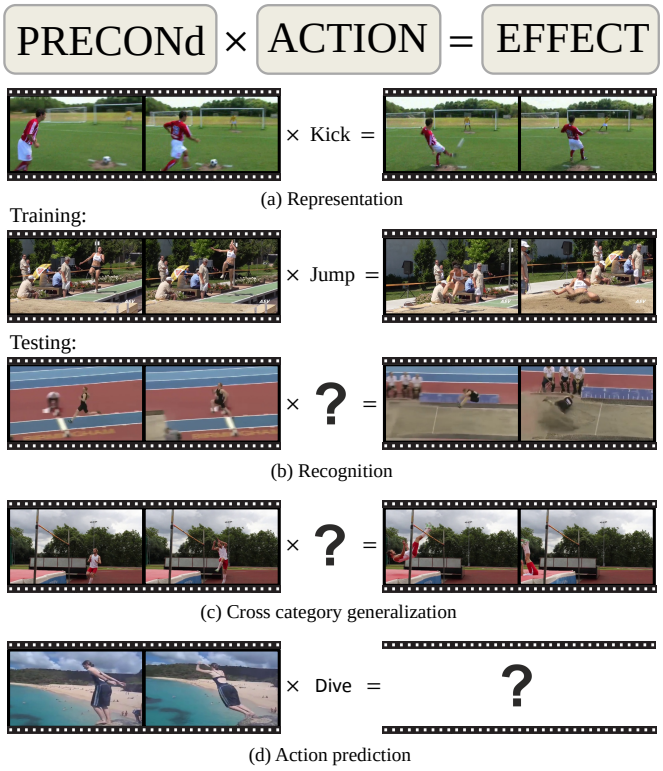


Figure 1: We represent action as the transformation from precondition to effect. (a) For example, the precondition of kicking is the player running towards the ball and the effect is the ball flies away. By using this representation, we can (b) perform action recognition given the training data on long jump, and (c) generalize the classifier to high jump in testing. (d) More interestingly, we can perform visual prediction given the action precondition.

(improved based on [2, 1]) where the action is represented as a linear transformation between the final fully connected layers of the two towers representing the precondition and effect states of an action.

Our experimental evaluations show that our representation is well suited for classical action recognition and can produce state of the art results on standard action recognition dataset such as UCF101 [35]. However, in order to test the robustness of our representation, we also test our model for cross category generalization. While overfit representations would perform competitively on current action recognition datasets (due to lack of diversity), their true test lie in their ability to generalize beyond learned action categories. For example, how would a model learned on "opening a window" generalize to recognize opening the trunk of the car? How about generalizing from a model trained on climbing a cliff to recognize climbing a tree? Our experimental evaluations show that our representation allows successful transfer of models across action categories. Finally, our transformation model can also be used to predict what is about to happen.

Our contributions include: (a) a new representation of actions based on transformations in visual world; (b) state of the art performance on an existing action recognition dataset: UCF101; (c) addressing the cross category generalization task and a new dataset, ACT, consisting of 43 categories of actions, which can be further grouped to 16 classes, and 11648 videos; (d) results on prediction task for our ACT dataset.

## 2. Related Work

Action recognition has been extensively studied in computer vision. Lack of space does not allow a comprehensive literature review (see [28] for a survey).

**Hand-crafted representations** have been conventionally used to describe patches centered at Space Time Interest Points (STIP) [21]. Most succesfull examples are 3D Histogram of Gradient(HOG3D) [18], Histogram of Optical Flow(HOF) [22], and Motion Boundary Histogram [3]. Mid- to high-level representation also used to model complex actions [34, 46, 30, 51, 11]. More recently, trajectory based approaches [43, 26, 24, 42, 14, 27, 20] have shown significant improvement in action recognition.

**Learned representations** using deep learning have recently produced state of the art results in action recognition [13, 17, 32, 44, 40, 41, 23, 8, 49]. Karpathy et al. [17] proposed to train Convolutional Neural Networks(CNNs) for video classification on the Sports-1M dataset. To better capture motion information in video, Simonyan et al. [32] introduced a Two Stream framework to train two separate CNNs for motion and color. Based on the Two Stream representation, Wang et al. [44] extracted deep feature and conducted trajectory constrained pooling to aggregate convolutional feature as video representations. Instead of learning features with 2D CNNs, Tran et al. [41] proposed to learn 3D CNNs, which offers effective and compact deep features.

**Temporal structure** of videos have also shown to be effective in action recognition [39, 6, 37, 10, 29]. For example, Tang et al. [39] proposed an HMM model to model the duration as well as the transitions of states in event video. Fernando et al [6] learned ranking functions for each video and tried to capture video-wide temporal information for action recognition. Recurrent Neural Networks have also been used to encode temporal information for learning video representations [36, 4, 25, 38, 48]. Srivastava et al. [36] proposed to learn video representations with LSTM in an unsupervised manner. Ng et al. [25] proposed to extract features with a Two Stream framework and perform LSTM fusion for action recognition. However, these HMM, RNN and recent LSTM approaches model a sequence of transformation across frames or keyframes; whereas in our framework we model action as a transformation between precondition and effect of action. The location of these frames are latent in our model. Note that our approach also applies a constraint such that precondition and effect frames for an action lie close-by the embedding space.

Most similar to our work is the work of Fathi et. al [5] where change in the state of objects are modeled using hand-crafted features in ego-centric videos of 7 activities. We differ from [5] in that we learn representations that enable explicit encoding of actions as transformations and show that our representations not only produce state of the art generic action recognition results but also allow prediction of the outcome of actions as well as cross-category model generalization.

## 3. Dataset

To study action recognition, several datasets have been compiled. Early datasets (e.g. Weizmann, KTH, Hollywood2, UCF Sports, UCF50) are too small to be used for training ConvNets. Recently, few large-scale video datasets have been introduced (e.g. Sports-1M [17], ActivityNet [9], CCV [16], THUMOS [15] and FGA-240 datasets [38]). Unfortunately, some of these recent datasets have untrimmed videos without localization information for short term actions. The most commonly studied datasets are UCF101 [35] and HMDB51 [19]. The UCF101 dataset lacks the desirable diversity among videos in each class. The HMDB51 dataset does not have enough number of videos compared to UCF101, and some of the videos are hard to recognize. But more importantly, none of these datasets are suitable for our task of examining cross category generalization of actions.

In this paper, we argue for cross-category generalization as a litmus test for action recognition. We argue that the cross-category recognition task should not allow approaches to overfit to action classes based on contextual information. For this task, we propose a dataset, namely ACT dataset. In this dataset, we collected 11648 video

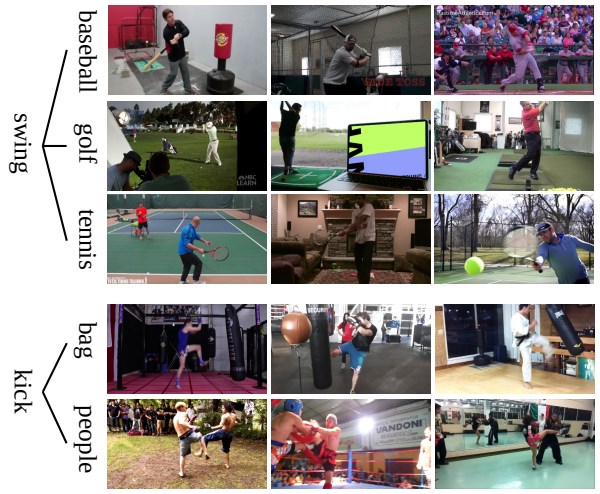


Figure 2: Samples in our ACT dataset. The action classes are arranged in a two-layer hierarchy. For each class, we collected hundreds of video clips with large diversities.

clips with 43 classes. These 43 classes can be further grouped into 16 super classes. For example, we have classes such as kicking-bag and kicking-people, they all belong to the super class kicking; swinging-baseball, swinging-golf and swinging-tennis can be grouped to swinging; climbing-rock, climbing-rope and climbing-tree can be grouped to climbing. Thus the categories are arranged in a 2-layer hierarchy. The higher layer represents super classes of actions such as kicking, swinging and climbing. Each super class has different sub-categories which are the same action under different subjects, objects and scenes. During the dataset collection, we also ensured that we only consider high resolution and diverse videos for each class.

**Dataset collection.** To collect our dataset we used YouTube videos. We used 50 keywords to retrieve videos which belong to one of the 43 classes. For each keyword, we downloaded around 500 high quality videos which have length within 15 minutes. The videos were labeled by a commercial crowd-sourcing organization. We asked the workers to label the starting and ending frames for actions in the video. For each action class, we provided a detailed description and 3 annotation examples from different videos. To increase the diversity of actions, we required the workers to label no more than 8 action clips in each video, and between each clips there should be at least a temporal gap of 40 frames. We also set the temporal length limitations that each annotated clip should have at least 1 second and at most 10 seconds length. As figure 2 illustrates, our dataset has large intra-class diversities.

**Task design.** We design two types of tasks for our ACT dataset. The first task is standard action classification over 43 categories. The split used for this task included 65% of videos as training and the rest as testing data; resulting in 7574 training videos and 4074 for testing in total. The second proposed task for this dataset is cross-category generalization. For each of the 16 super classes, we consider one of its sub-category as testing and the other sub-categories are used for training. For example, for superclass "swinging", we want to see if the model trained on swinging-baseball and swinging-golf can recognize swinging-tennis as "swinging". The selection for testing sub-categories is performed randomly. We create 3 different random splits for the second task. There are around 7000 training samples and 4500 testing samples in average.

## 4. Modeling Actions as Transformations

Given an input video $X$ consisting of $t$ frames, we denote each frame as $x_i$ and the whole video as $X = \{x_1, x_2, ..., x_t\}$. We make an assumption that the precondition state of an action corresponds to the first $z_p$ frames and the effect of the action can be seen after from $z_e$ till the end. We denote precondition and effect frames as: $X_p = \{x_1 \dots x_{z_p}\}$ and $X_e = \{x_{z_e} \dots x_t\}$. Note that we do not manually define how many frames are used of representing precondition and effect; and therefore $z_p$ and $z_e$ are treated as latent variables, which will be inferred automatically by our model during training and testing.

Instead of representing the precondition and effect by pixels and modeling the transformation in pixel space, we want to represent them using higher-level semantic features (e.g., the last fully connected layer of a ConvNet). The action then corresponds to transformation in higher-level feature subspace. This can be modeled via a Siamese ConvNet as shown in Figure 3. Given the video frames for the precondition and effect states, we feed the frames of precondition state $X_p$ as inputs for the ConvNet on the top and the frames of effect state $X_e$ are fed to the ConvNet on the bottom. For each tower of ConvNet, it computes the features for each frame independently, and then the features are aggregated via average pooling. We add a final $d$-dimensional fully connected layer after average pooling which represent the feature space where precondition, effect and the transformation between the two are modeled. Formally, we use $f_p(X_p)$ to represent the $d$-dimension embedding for the precondition state generated from the network on the top(in Figure 3) given input $X_p$. For the network on the bottom, we represent the embedding for the effect state as $f_e(X_e)$ with the same dimension $d$ given input $X_e$.

Finally, we model the actions as the transformation between these two states. Specifically, we use a linear transformation matrix to model this. For a dataset with $n$ categories of actions, we have a set of $n$ corresponding transformation matrices $\{T_1, ..., T_n\}$ to represent them. Each $T_i$ is a $d \times d$ dimensions matrix. At the training time, suppose an input video $X$ belongs to action category $i$, we first obtain the embedding for the precondition and effect states of the action as $f_p(X_p)$ and $f_e(X_e)$. We then apply the transformation matrix $T_i$ on the embedding of the precondition state as $T_i f_p(X_p)$, which is also a $d$-dimension vector. The objective of learning is making the distance

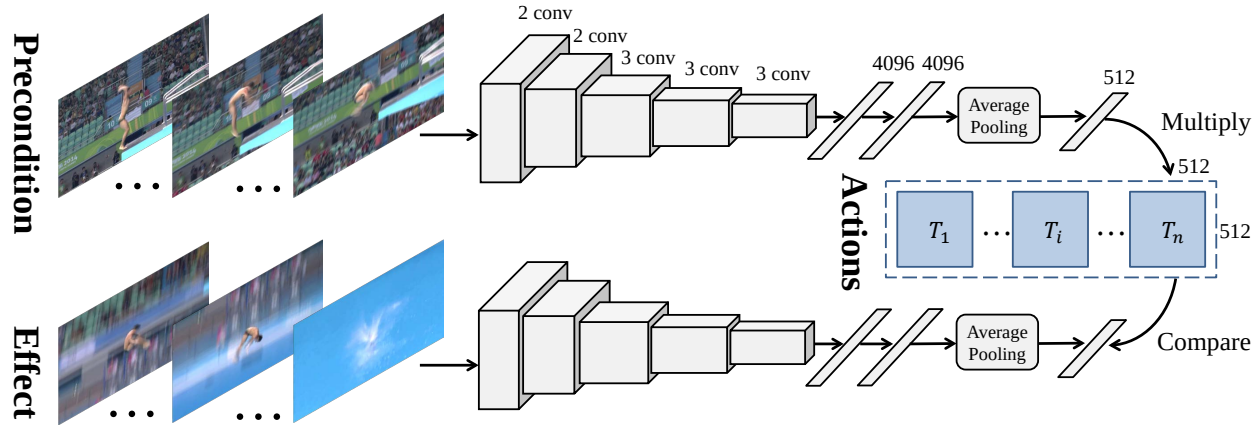


Figure 3: Siamese network architecture. Given a video, we feed the precondition state frames to the top network and effect state frames to the bottom network. Each tower of the ConvNet computes the feature for each frame independently, and aggregates the features via average pooling. The pooling results are fully connected to 512-D embedding outputs. We apply $n$ transformations (actions) on the precondition embedding and compare with the effect embedding to decide the action class.

$D(T_i f_p(X_p), f_e(X_e))$ between the two embeddings small. Note that while training, the gradients are backpropagated throughout the Siamese networks; thus we learn both the embedding space and the transformation simultaneously.

**Network Architecture** We applied the VGG-16 network architecture [33] for both sides of our Siamese network. The VGG-16 network is a 16-layer ConvNet with 13 convolutional layers and 3 fully connected layers. As we mentioned before, we perform forward propagation for each video frame independently. We extract the feature of the second last fully connected layer for each frame, which is a 4096-D vector. In each side of our model, the features are aggregated via average pooling, and we use a final fully connected layer upon these pooling outputs. The dimension of the last embedding layer outputs is $d = 512$. In our model, we don't share the model parameters between two ConvNets. Intuitively, we want to learn different semantic representations for precondition and effect of actions.

**Two Stream Siamese: RGB and Optical Flow as Inputs.** For each input frame, we resize it by keeping the aspect ratio and make the smaller side 256 pixels. To represent the video frames, we follow the same strategy as [32], which represents the frames with RGB images as well as optical flow fields. We train two separate models for RGB and optical flow fields as inputs. For the model using RGB images as inputs, the input size is $224 \times 224 \times 3$. For the model using optical flow as inputs, we represent each frame by stacking 10 optical flow fields extracted from 10 consecutive frames in the video starting from the current one. The optical flow field can be represented by a 2-channel image including horizontal and vertical flow directions. Therefore, the input size is $224 \times 224 \times 20$ as mentioned in [32].

**Implementation Details:** During training and testing, similar to [32], we resample the videos to be 25 frames in length ($t = 25$). Note that the optical flow fields are still computed in the original video without resampling. We also constrain the latent variables $z_p$ and $z_e$, which are the indexes for the end frame of the precondition state and start frame of the effect state such that: $z_p \in [\frac{1}{3}t, \frac{1}{2}t)$ and $z_e \in (\frac{1}{2}t, \frac{2}{3}t]$. In our experiments, we noticed that the choices of these parameters has no significant effect on the performance.

### 4.1. Training

We now introduce the training procedure for our model. Suppose we have a dataset of $N$ samples with $n$ categories $\{(X_i, y_i)\}_{i=1}^N$, where $y \in \{1, ..., n\}$ is the label. Our goal is to optimize the parameters for the ConvNets and transformation matrices $\{T_i\}_{i=1}^n$. We also need to calculate the latent variables $z_p$ and $z_e$ for each sample. Thus, we propose an EM-type algorithm, which performs a two-step procedure in each iteration: (i) learning model parameters and (ii) estimating latent variables .

**(i) Learning model parameters.** We first discuss the optimization of model parameters given the latent variables. Our objective is to minimize the distance between the embedding of the precondition state after transformation and the embedding of the effect state computed by the second ConvNet. This can be written as:

$$\min D(T_y f_p(X_p), f_e(X_e)), \tag{1}$$

where $T_y$ is the transformation matrix for the ground-truth class $y$, $f_p(X_p)$ is the embedding for the precondition of the action and $f_e(X_e)$ is the embedding for the effect of the action. We use cosine distance here such that $D(v_1, v_2) = 1 - \frac{v_1 \cdot v_2}{\|v_1\| \|v_2\|}$ between any two vectors $v_1, v_2$.

To make our model discriminative, it is not enough to minimize the distance between two embeddings given the corresponding transformation $T_y$. We also need to maximize the distances for other incorrect transformations $T_i (i \neq y)$, which equals to minimize the negative of them. Thus, we have another term in the objective,

$$\min \sum_{i \neq y}^{n} max(0, M - D(T_i f_p(X_p), f_e(X_e))), \quad (2)$$

where $M$ is the margin so that we will not penalize the loss if the distance is already larger than $M$. In our experiment, we set $M = 0.5$. By combining Eq.(1) and Eq.(2), we have the final loss for training. It is a contrastive loss given sample $(X, y)$ and latent variables $z_p$ and $z_e$ as inputs. We train our model with back-propagation using this loss.

**(ii) Estimating latent variables.** Given the model parameters, we want to estimate the end frame index $z_p$ of the precondition state and start frame index $z_e$ of the effect state. That is, we want to estimate the latent variables to give a reasonable temporal segmentation for the input video. Since the latent variable only depends on the ground-truth action category; we only use the first term in the loss function, Eq.(1), to estimate $z_p$ and $z_e$ as follows:

$$(z_p^*, z_e^*) = argmin_{(z_p, z_e)} D(T_y f_p(X_p), f_e(X_e)), \quad (3)$$

where $(z_p^*, z_e^*)$ are the estimation results given current model parameters. To estimate these latent variables, we use a brute force search through the space of latent variables. Note that for computation efficiency, we first compute features for all frames, and then a brute force search only requires the average pooling step to be redone for each configuration of $(z_p, z_e)$

**Model pre-training.** We initialize the ConvNets using pre-training and adapt them to action classification with fine-tuning as mentioned in [45]. We transfer the same set of convolutional parameters to both ConvNet towers and randomly initialize the fully connected layers in our model. We summarize the learning procedure as Algorithm 1.

### 4.2. Inference

During inference, given a video $X$ and our trained model, our goal is to infer its action class label $y$ and segment the action into the precondition and effect states at the same time. The inference objective can be represented as,

$$\min_{y, z_p, z_e} D(T_y f_p(X_p), f_e(X_e)). \quad (4)$$

**Algorithm 1** Learning

**Input:**
$N$ videos with $n$ classes $\{(X_i, y_i)\}_{i=1}^N$, iteration number $Iter$.
**Output:**
Parameters of ConvNets and transformation matrices.

Model initialization, $i = 0$.
**repeat**
1. Forward propagation and feature computation for each frame.
2. Search latent variables with Eq.(3).
3. Calculate the joint loss using Eqs.(1)and (2) and perform back-propagations.
4. $i = i + 1$.

**until** $i = Iter$

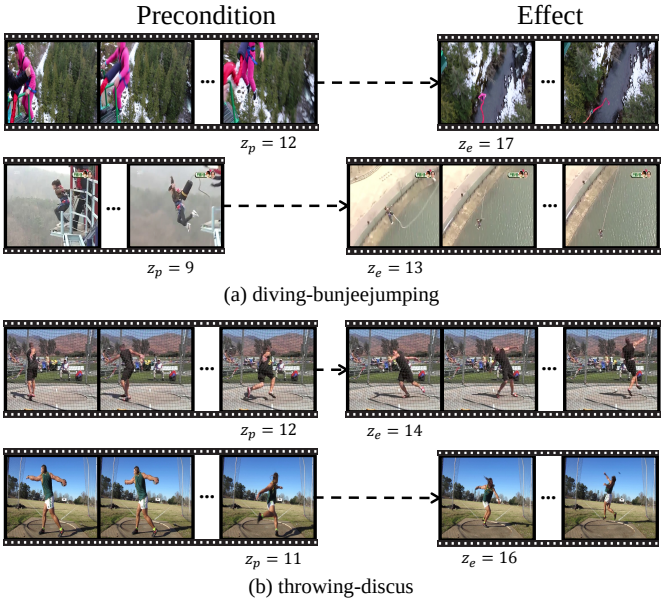


Figure 4: Temporal segmentation results after inference. $z_p$ and $z_e$ are the latent variables indexing the end frame for precondition and start frame for effect. Our model can provide reasonable temporal alignment of states between different videos in the same category.

More specifically, we first calculate the ConvNet feature before the average pooling layer for all the frames. Note that the first half of the frames are fed into the network for the precondition state and the second half of the frames are fed into the network for the effect state. We do a brute force search over the space of $(y, z_p, z_e)$ to estimate the action category and segmentation into precondition and effect. We visualize reasonable segmentation results for precondition and effect states during inference as Figure. 4.

**Model fusion.** We perform the inference with the models using RGB images and optical flow as inputs separately. For each video, we have $n$ distance scores from each model. We fuse these two sets of scores by using weighted average. As suggested by [45] we weigh the flow twice as much as the RGB ConvNet.

## 5. Experiment

In this section, we first introduce the details of datasets and our experimental settings. Then we provide quantitative and qualitative results on different datasets.

**Datasets.** We evaluate our method on three datasets, including UCF101 [35], HMDB51 [19] and our ACT dataset. UCF101 dataset contains 13320 videos with 101 classes. The HMDB51 dataset is composed of 6766 videos with 51 action categories. For both datasets, we follow the standard evaluation criteria using 3 official training/testing splits. Our ACT dataset has 11648 video clips from 43 action classes and 16 super classes. For this dataset, we conduct our experiments on two tasks. The first task is standard action classification of 43 categories. The second task is to

| Method | RGB | Optical Flow | Fusion |
|---|---|---|---|
| Two Stream [32] | 73.0% | 83.7% | 88.0% |
| Two Stream (VGG16) [45] | 78.4% | 87.0% | 91.4% |
| Ours | **80.8%** | **87.8%** | **92.4%** |

Table 1: Average accuracies on UCF101 over 3 splits.

| Method (for split1) | RGB | Optical Flow | Fusion |
|---|---|---|---|
| Two Stream (VGG16) [45] | 79.8% | 85.7% | 90.9% |
| Two Stream First Half | 80.0% | 84.7% | 90.0% |
| Two Stream Second Half | 79.5% | 84.8% | |
| Ours | **81.9%** | **86.4%** | **92.0%** |

Table 2: Comparing method training different networks for different parts of the videos on UCF split1.

test the generalization ability of our model. We conduct the experiments using 3 training/testing splits for 16-class classification. For each super class, one sub-category is used for testing and the others for training.

**Implementation Details.** We first pre-train our networks using softmax loss on action classfication as the Two Stream baseline [45] and then transfer the convolutional parameters to our model. For HMDB51 dataset, as the number of training videos are relatively small (around 3.6K), we borrowed the ConvNet trained on UCF101 dataset with softmax loss to initialize our model as [44, 32]. For the ACT dataset, we did not use UCF101 or HMDB51 during pre-training. Note that there are two types of inputs for our model: RGB and optical flow. We trained two different models for different inputs. For the model using RGB images as input, we set the initial learning rate as $10^{-5}$ and reduce it by 10 times after every $10K$ iterations and stop at $30K$ iterations. For the model using optical flow, the initial learning rate is set to $10^{-5}$. We reduce learning rate by order of 10 for every $20K$ iterations and stop at $50K$ iterations. We set the size of the batch to 50 videos. To compute the optical flow, we apply the TVL1 optical flow algorithm [50] and discretize the values to the range of $0-255$.

### 5.1. Experimental Results.

**UCF101 dataset.** The accuracies over 3 training/testing splits of UCF101 are shown in Table 1. The Two Stream [32] method using an 8-layer CNNs achieves $88.0\%$. By replacing the base network with VGG-16 architecture, [45] obtains $91.4\%$ accuracy. We use [45] as a baseline for comparisons. Our model outperforms both models using RGB images and optical flow fields. After fusion of two models, we achieve the state of the art performance of $92.4\%$. See Table 4 for more comparisons.

To further analyse our results and factor out the contributions of the appearance of the precondition and effect segments, we perform an ablative analysis on the first split of UCF101. We train different networks using softmax loss for different segments of the videos independently. For each video, we cut it by half and train one network using the first half of frames and another network are trained using the second half. In total we train 2 RGB ConvNets and 2 optical flow ConvNets. For model fusion, we average the results from these 4 networks. We show the results in Table 2. Compared to the baseline method [45], we find that the performance decreases most of the time and the fusion model is $0.9\%$ worse than the baseline. This is mainly due to the decrease in training samples leads to over-fitting. This analysis confirms that our model is learning something beyond the appearance of the precondition and effect segments.

| Method | RGB | Optical Flow | Ave Fusion |
|---|---|---|---|
| Two Stream [32] | 40.5% | 54.6% | 58.0% |
| Two Stream (VGG16) | 42.2% | 55.0% | 58.5% |
| Ours | **44.1%** | **57.1%** | **62.0%** |

Table 3: Average accuracies on HMDB51 over 3 splits.

**HMDB51 dataset.** For HMDB51, we also report the average accuracies over 3 splits in Table 3. As a baseline, we apply Two Stream method [32, 45] with VGG-16. The baseline method is better than [32] with average fusion. However, it is not as good as the best results ($59.4\%$) in [32] using SVM for model fusion. We show that our method has $1.9\%$ gain for models using RGB images and $2.1\%$ gain for model using optical flow. After model fusion, our method reach $62\%$ accuracy, which is $3.5\%$ better than the baseline.

More interestingly, we show that our method and the Two Stream method are complimentary to each other by combining the classification results from two methods. Note that we use the same initialization for convolution layers in our model and the Two Stream model. Since our outputs are distances and the Two Stream outputs are probabilities, we convert the distances outputs to probabilities. To do this, we extract the precondition embedding after groundtruth transformation and the effect embedding on training data. We concatenate the two embeddings and train a softmax classifier. In testing, we apply the same inference method as before and use the new trained softmax classifier to generate the probabilities, which gives the same performance as before. We average the outputs from two methods with equal weights as our final result. As Table 4 shows, by combining our method and the Two Stream method we have $1.4\%$ boost, which leads to $63.4\%$ accuracy.

**ACT dataset.** We evaluate on two tasks in this dataset as proposed in the Dataset section. Motivated by the recent literatures [25, 48, 4] in using LSTM to model the temporal information of videos, we also conduct experiments using the LSTM baseline for comparison besides the Two Stream baseline. All the ConvNets are based on VGG-16 architecture. For this LSTM approach, we first perform forward propagation with Two Stream model on each of the 25 frames and extract the feature from the fully connected layer before the classification outputs. The features are fed into LSTM to generate the classification scores. During training, we train the LSTM and Two Stream models jointly.

The first task is the standard action classification over 43 classes. As Table 5 illustrates, the LSTM method gives

| | RGB | | | | Optical Flow | | | | Fusion | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | Split1 | Split2 | Split3 | Average | Split1 | Split2 | Split3 | Average | Split1 | Split2 | Split3 | Average |
| Two Stream | 48.3% | 53.2% | 49.8% | 50.4% | 53.5% | 56.8% | 56.3% | 55.5% | 59.6% | 66.8% | 61.8% | 62.7% |
| LSTM+Two Stream | 47.7% | 53.4% | 50.4% | 50.5% | 55.3% | 57.2% | 58.2% | 56.9% | 60.0% | 67.3% | 61.9% | 63.1% |
| Ours | 50.7% | 55.2% | 52.3% | **52.7%** | 57.4% | 59.6% | 60.3% | **59.1%** | 62.5% | 70.1% | 65.4% | **66.0%** |

Table 6: Accuracies over 3 splits for the 2nd task on ACT dataset.

| HMDB51 | | UCF101 | |
|---|---|---|---|
| STIP+BoW [19] | 23.0% | STIP+BoW [35] | 43.9% |
| DCS+VLAD [12] | 52.1% | CNN[17] | 65.4% |
| VLAD Encoding [47] | 56.4% | IDT+FV [43] | 85.9% |
| IDT+FV [43] | 57.2% | IDT+HSV [26] | 87.9% |
| Two Stream [32] | 59.4% | Two Stream [32] | 88.0% |
| IDT+HSV [26] | 61.1% | LSTM with Two Stream [25] | 88.6% |
| TDD+FV [44] | 63.2% | TDD+FV [44] | 90.3% |
| Two Stream (VGG16) by us | 58.5% | Hybrid LSTM [48] | 91.3% |
| Ours | 62.0% | Two Stream (VGG16) [45] | 91.4% |
| Ours+Two Stream | **63.4%** | Ours | **92.4%** |

Table 4: Comparison to the state of the art results.

| Method | RGB | Optical Flow | Fusion |
|---|---|---|---|
| Two Stream | 66.5% | 70.5% | 78.0% |
| LSTM+Two Stream | 68.5% | 71.4% | 78.3% |
| Ours | **69.3%** | **73.0%** | **80.1%** |

Table 5: Accuracies for the 1st task on ACT dataset.

boosts on both RGB and optical flow inputs compared to the Two Stream method, however the results after fusion only has 0.3% improvement. On the other hand, our method reach 80.1%, 2.1% higher than the baseline.

For the second task, we want to examine the cross category generalization ability of our model. We perform 16-class classification on 3 different splits. For each class, one sub-category is left out for testing and the other sub-categories are used for training. As Table 6 shows, compared to the Two Stream basline, we have 2.3% improvements in models using RGB images as inputs and 3.6% improvements using optical flow fields as inputs in average. After fusion two models with different inputs, we have 66.0% accuracy, 3.3% higher than the baseline. The LSTM model also gives improvement based on the Two Stream model, however the gain is small. This experiment shows that our model works better in category generalization.

## 5.2. Visualization

By just showing the results in numbers does not give a full understanding of why our method is doing better. What is our model learning? In which way our model is doing better? To answer these questions, we perform several visualization tasks on our ACT dataset.

**Nearest neighbor test.** In this experiment, we want to visualize in what cases our method is doing better via nearest neighbor on the features, which are extracted by the models trained in the second task of the ACT dataset. To extract the feature with our model, we perform inference on the video and concatenate the precondition embedding after transformation with the effect embedding. For the baseline Two Stream method, we extract the second last fully connected layer feature and compute the average feature over the frames in the video. For both methods, we extract the feature for the RGB and optical flow models and then average them. As shown in Figure 5, given the query from testing samples on the left, we show the retrieved videos from the training videos. In the middle is our result and the retrieval results of Two Stream method are on the right. For instance, on the first row, the query is a baby pushing a cart, we can retrieve the video of a man is pushing a car, while the Two Stream method gets the baby crawling on the floor as the nearest neighbor. As the Two Stream method tries to match videos with similar appearance and motions, by modeling the action as changes give us more semantically related retrieval results.


Figure 5: Nearest neighbor test. Given the query videos on the left, our method can retrieve semantically related videos in the middle while the Two Stream method retrieve videos with similar appearance and motion on the right.

**Visualization of what the models focus on.** In this experiment, we visualize what the models focus on during inference using similar techniques as [31, 7]. We select one frame from the precondition state and another frame from the effect state, we feed these two RGB images into two

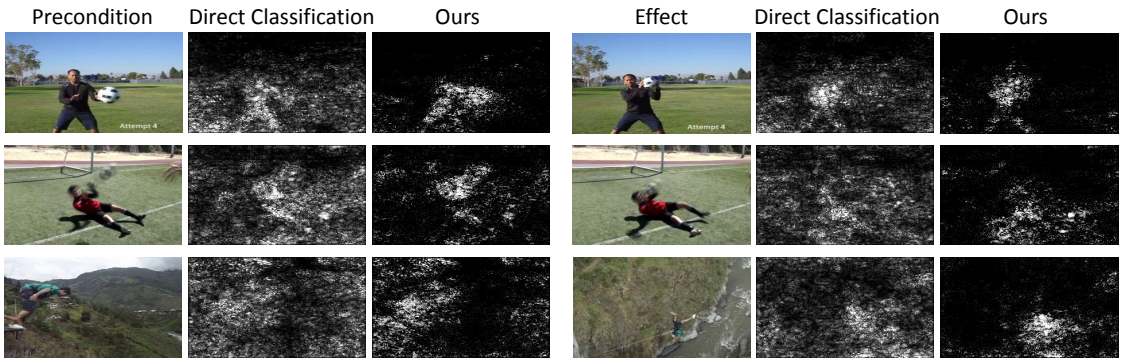


Figure 6: Given the precondition and effect frames as inputs, we visualize the magnitude of gradients via back-propagation. While the direct classificaiton method (network trained with RGB images in Two Stream method) are sensitive to the object and scenes, our model focus more on the changes.


Figure 7: Prediction results. Given the precondition frames of test videos on the left, we retrieve the effect frames of training samples. In the middle are the retrieval results with the same class as the query, on the right are results from different classes.

towers of our model. We calculate the transformation loss given the video label and perform back-propagation. We visualize the gradients passed to the input image by showing the magnitude of the gradients. This visualization shows which parts of the image force the network to decide on an action class. Similar visualization is also applied on the RGB network from the Two Stream method by using softmax loss. As Figure 6 shows, given the precondition and effect frames, our model focus more on the changes happened to the human while the direct classification model (RGB network from Two Stream) perform classification based on the entire scene. In the first row, our model focus on the man and the ball when the ball is flying towards to him. As the man catches the ball, our model has high activations on the man with the ball. In the second row, our model can also capture locations where the ball is flying towards the man. Interestingly, our model notices the change of the feet and the shadow. In both cases, the direct classification model fires on the whole scene.

**Visual prediction.** We use the model trained on the first task of the experiment on the ACT dataset(43-class classification) to perform inference on the training and testing data. Given the precondition embedding after transformation on the testing video, we retrieve the effect embedding from the same class among the training data. If our model is actually learning the transformation, then it should be able to find effects frames that are similar to what we expect from applying the transformation to the precondition frames. We also repeat the same experiment by retrieve the effect embedding from different classes. We visualize the results in Figure 7. In each row, the first two images are the begin and end frame of precondition segments of testing videos. The two images in the middle are the retrieval results with the same label as query. The two images on the right are the retrieval results with different class label as query. We show that our method can retrieve reasonable and smooth prediction results. For example, when asked our model to predict what is going to happen when a man jumpes from a dive, it retrieves getting into the pool as within category results, and getting into the lake as cross category results. This is another signal that shows that our model is in fact learning actions as tranformations from preconditions to effects.

## 6. Conclusion

We propose a novel representation for action as the transformation from precondition to effect. We show promising action recognition results on UCF101, HMDB51 and ACT datasets. We also show that our model has better ability in cross category generalization. We show qualitative results which indicate that our method can explicitly model an action as change or transformation it brings to the environment.

**Acknowledgement**: This work was supported by ONR N00014-13-1-0720, NSF IIS-1218683, NSF IIS-IIS- 1338054, and Allen Distinguished Investigator Award. This work was also partially supported by ONR MURI N000141010934, ONR MURI N000141612007 and gift from Google. The authors would like to thank Yahoo! and Nvidia for the compute cluster and GPU donations respectively.

## References

[1] J. Bromley, I. Guyon, Y. LeCun, E. Sackinger, and R. Shah. Signature verification using a siamese time delay neural network. *NIPS*, 1993. 2

[2] S. Chopra, R. Hadsell, , and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. *CVPR*, 2005. 2

[3] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*, 2006. 2

[4] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 2, 6

[5] A. Fathi and J. M. Rehg. Modeling actions through state changes. In *ICCV*, 2013. 2

[6] B. Fernando, E. Gavves, J. O. M., A. Ghodrati, and T. Tuytelaars. Modeling video evolution for action recognition. In *CVPR*, 2015. 2

[7] C. Gan, N. Wang, Y. Yang, D.-Y. Yeung, and A. G. Hauptmann. Devnet: A deep event network for multimedia event detection and evidence recounting. *CVPR*, 2015. 7

[8] G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, 2015. 2

[9] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles. Activitynet: A large-scale video benchmark for human activity understanding. *CVPR*, 2015. 2

[10] H. Izadinia and M. Shah. Recognizing complex events using large margin joint low-level event model. *ECCV*, 2012. 2

[11] A. Jain, A. Gupta, M. Rodriguez, and L. S. Davis. Representing videos using mid-level discriminative patches. In *CVPR*, 2013. 2

[12] M. Jain, H. Jegou, and P. Bouthemy. Better exploiting motion for better action recognition. *CVPR*, 2013. 7

[13] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *TPAMI*, 2013. 2

[14] Y. Jiang, M. Lim, C. Zheng, and A. Saxena. Learning to place new objects in a scene. *IJRR*, 31(9), 2012. 2

[15] Y.-G. Jiang, J. Liu, A. R. Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. Thumos challenge: Action recognition with a large number of classes. http://crcv.ucf.edu/THUMOS14/, 2014. 2

[16] Y.-G. Jiang, G. Ye, S.-F. Chang, D. Ellis, and A. C. Loui. Consumer video understanding: A benchmark database and an evaluation of human and machine performance. In *ICMR*, 2011. 2

[17] 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. 2, 7

[18] A. Klaser, M. Marszalek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 2008. 1, 2

[19] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T.Serre. Hmdb: A large video database for human motion recognition. In *ICCV*, 2011. 2, 5, 7

[20] Z. Lan, M. Lin, X. Li, A. G. Hauptmann, and B. Raj. Beyond gaussian pyramid: Multi-skip feature stacking for action recognition. In *CVPR*, 2015. 2

[21] I. Laptev. On space-time interest points. *IJCV*, 64, 2005. 2

[22] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 2

[23] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, 2011. 2

[24] P. Matikainen, M. Hebert, and R. Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *ICCV Workshops*, 2009. 2

[25] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 2, 6, 7

[26] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *CoRR*, /abs/1405.4506, 2014. 2, 7

[27] X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked fisher vectors. In *ECCV*, 2014. 2

[28] R. Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010. 2

[29] M. Rohrbach, M. Regneri, M. Andriluka, S. Amin, M. Pinkal, and B. Schiele. Script data for attribute-based recognition of composite activities. *ECCV*, 2012. 2

[30] S. Sadanand and J. J. Corso. Action bank: A high-level representation of activity in video. In *CVPR*, 2012. 2

[31] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, /abs/1312.6034, 2013. 7

[32] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 2, 4, 6, 7

[33] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 4

[34] Y. Song, L.-P. Morency, and R. Davis. Action recognition by hierarchical sequence summarization. In *CVPR*, 2013. 2

[35] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, /abs/1212.0402, 2012. 2, 5, 7

[36] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. *CoRR*, /abs/1502.04681, 2015. 2

[37] C. Sun and R. Nevatia. Active: Activity concept transitions in video event classification. *ICCV*, 2013. 2

[38] C. Sun, S. Shetty, R. Sukthankar, and R. Nevatia. Temporal localization of fine-grained actions in videos by domain transfer from web images. In *ACM Multimedia*, 2015. 2

[39] K. Tang, L. Fei-Fei, and D. Koller. Learning latent temporal structure for complex event detection. In *CVPR*, 2012. 2

[40] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *ECCV*, 2010. 2

[41] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 2

[42] H. Wang, A. Klaser, C. Schmid, and L. Cheng-Lin. Action recognition by dense trajectories. In *CVPR*, 2011. 2

[43] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 1, 2, 7

[44] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015. 2, 6, 7

[45] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. *CoRR*, /abs/1507.02159, 2015. 5, 6, 7

[46] Y. Wang and G. Mori. Hidden part models for human action recognition: Probabilistic vs. max-margin. *TPAMI*, 2011. 2

[47] J. Wu, Y. Zhang, and W. Lin. Towards good practices for action video encoding. *CVPR*, 2014. 7

[48] 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. *CoRR*, /abs/1504.01561, 2015. 2, 6, 7

[49] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative cnn video representation for event detection. *CVPR*, 2015. 2

[50] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l1 optical flow. *29th DAGM Symposium on Pattern Recognition*, 2007. 6

[51] J. Zhu, B. Wang, X. Yang, W. Zhang, and Z. Tu. Action recognition with actons. In *ICCV*, 2013. 2