Dense Optical Flow Prediction from a Static Image

Jacob Walker, Abhinav Gupta, and Martial Hebert
Robotics Institute, Carnegie Mellon University
{jcwalker, abhinavg, hebert}@cs.cmu.edu

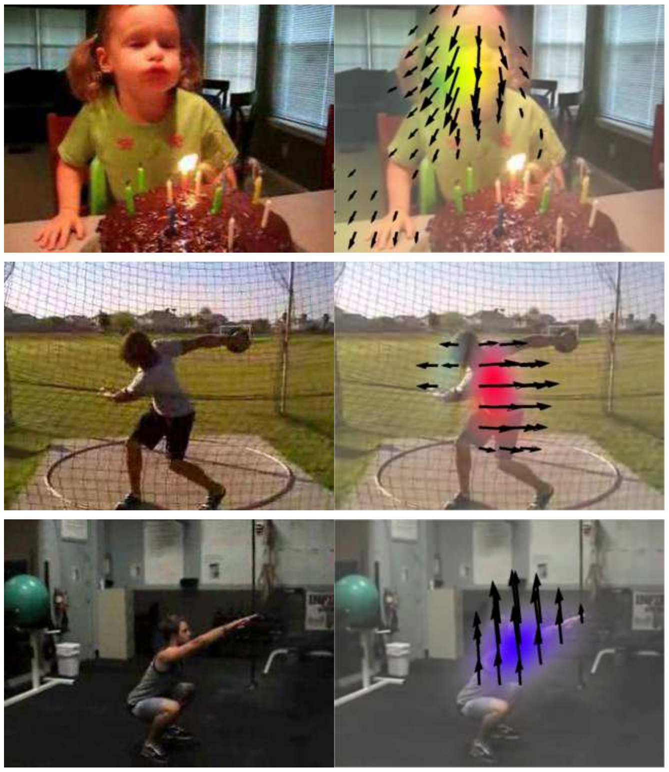
Abstract

Given a scene, what is going to move, and in what direction will it move? Such a question could be considered a non-semantic form of action prediction. In this work, we present a convolutional neural network (CNN) based approach for motion prediction. Given a static image, this CNN predicts the future motion of each and every pixel in the image in terms of optical flow. Our CNN model leverages the data in tens of thousands of realistic videos to train our model. Our method relies on absolutely no human labeling and is able to predict motion based on the context of the scene. Because our CNN model makes no assumptions about the underlying scene, it can predict future optical flow on a diverse set of scenarios. We outperform all previous approaches by large margins.

1. Introduction

Consider the images shown in Figure 1. Given the girl in front of the cake, we humans can easily predict that her head will move downward to extinguish the candle. The man with the discus is in a position to twist his body strongly to the right, and the squatting man on the bottom has nowhere to move but up. Humans have an amazing ability to not only recognize what is present in the image but also predict what is going to happen next. Prediction is an important component of visual understanding and cognition. In order for computers to react to their environment, simple activity detection is not always sufficient. For successful interactions, robots need to predict the future and plan accordingly.

There has been some recent work that has focused on this task. The most common approach to this prediction problem is to use a planning-based agent-centric approach: an object [10] or a patch [24] is modeled as an agent that performs actions based on its current state and the goal state. Each action is decided based on compatibility with the environment and how these actions helps the agent move closer to the goal state. The priors on actions are modeled via transition matrices. Such an approach has been shown to produce impressive results: predicting trajectories of humans in parking lots [10] or hallucinating car movements on streets [24]. There are two main problems with this approach. First, the predictions are still sparse, and the motion is still modeled as a trajectory. Second, and more importantly, these approaches have always been shown to perform in restrictive domains such as parking lots or streets.


(a) Input Image (b) Prediction

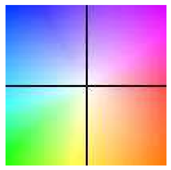


Figure 1: **Motion Prediction.** Consider single, static input images (a). Our method can first identify what these actions are and predict (b) correct motion based on the pose and stage of the action without any video information. We use the color coding from [1] shown on the right.

In this paper, we take the next step towards generalized prediction — a framework that can be learned from tens of thousands of realistic videos. This framework can work in indoor and outdoor environments; it can account for one or multiple agents whether the agent is an animal, a human, or even a car. Specifically, this framework looks at the task of motion prediction — given a static image we predict the dense expected optical flow as if this image were part of a video. This optical flow represents how and where each and every pixel in the image is going to move in the future. However, we can see that motion prediction is more than identifying active agents; it is also highly dependent on context. For example, someone's entire body may move up or down if they are jump-roping, but most of the body will be stationary if they are playing the flute. Instead of modeling agents and its context separately under restrictive assumptions, we use a learning based approach for motion prediction. Specifically, we train a deep network that can incorporate all of this contextual information to make accurate predictions of future motion in a wide variety of scenes. We train our model from thousands of realistic video datasets, namely UCF-101 [21] and the HMDB-51 [13].

Contributions: Our paper makes three contributions. First, we present a CNN model for motion prediction. Given a static image, our CNN model predicts expected motion in terms of optical flow. Our CNN-based model is agent-free and makes almost no assumptions about the underlying scene. Therefore, we show experimental results on diverse set of scenes. Second, our CNN model gives state of the art performance on prediction compared to contemporary approaches. Finally, we also present a proof of concept extension of the CNN model which makes long-range prediction about future motion. Our preliminary results indicate that this new CNN model might indeed be promising even for the task of long-range prediction.

2. Background

Prediction has caught the interest of the vision community in recent years. Most of research in this area has looked at different aspects of the problem. The first aspect of interest is the output space of prediction. Some of the initial work in this area focused on predicting the trajectory for the given input image [28]. Others have looked at more semantic forms of prediction: that is, predicting the action class of what is going to happen next [5, 15]. However, one of the issues with semantic prediction is that it tells us nothing about the future action beyond the category. One of our goals in prediction is to go beyond classification and predict the spatial layout of future actions. For example, in case of agents such as humans, the output space of prediction can be trajectories themselves [10]. On the contrary, recent approaches have argued for much richer form of predictions even in terms of pixels [24] or the features of the next frame [7, 19].

The other aspect of research in visual prediction looks at the question of selecting the right approach for prediction. There have been two classes of approaches for the temporal prediction. The first is a data-driven, non-parametric approach. In the case of non-parameteric approaches, they do not make any assumptions about the underlying scene. For example, [28] simply retrieves videos visually similar to the static scene, allowing a warping [17] of the matched action into the scene. The other end of the spectrum is parametric and domain-specific approaches. Here, we make assumptions on what are the active elements in the scene whether they may be cars or people. Once the assumption is made, then a model is developed to predict agent behavior. This includes forecasting pedestrian trajectories [10], human-human interactions [7, 15], human expressions through SOSVM [5], and human-object interaction through graphical models [11, 3].

Some of the recent work in this area has looked at a more hybrid approach. For example, Walker et al. [24] builds a data-derived dictionary of rigid objects given a video domain and then makes long-term motion and appearance predictions using a transition and context model. Recent approaches such as [19] and [22] have even looked at training convolutional neural networks for predicting one future frame in a clip [19] or motion of handwritten characters [22].

We make multiple advances over previous work in this paper. First, our self-supervised method can generalize across a large number of diverse domains. While [24] does not explicitly require video labels, it is still domain dependent, requiring a human-given distinction between videos in and outside the domain. In addition, [24] focused only on birds-eye domains where scene depth was limited or non existent, while our method is able to generalize to scenes with perspective. [18] also uses self-supervised methods to train a Structured Random Forest for motion prediction. However, the authors only learn a model from the simple KTH [16] dataset. We show that our method is able to learn from a set of videos that is far more diverse across scenes and actions. In addition, we demonstrate much better generalization can be obtained as compared to the nearest-neighbor approach of Yuen et al. [28].

Convolutional Neural Networks: We show in this paper that a convolutional neural network can be trained for the task of motion prediction in terms of optical flow. Current work on CNNs have largely focused on recognition tasks both in images and video [12, 9, 4, 23, 29, 20, 22]. There has been some initial work where CNNs have been combined with recurrent models for prediction. For example, [19] uses a LSTM [6] to predict the immediate next frame given a video input. [22] uses a recurrent architecture to predict motions of handwritten characters from a video. On the other hand, our approach predicts motion for each and every pixel from a static image for any generic scene.

3. Methods

Our goal is to learn a mapping between the input RGB image and the output space which corresponds to the pre-

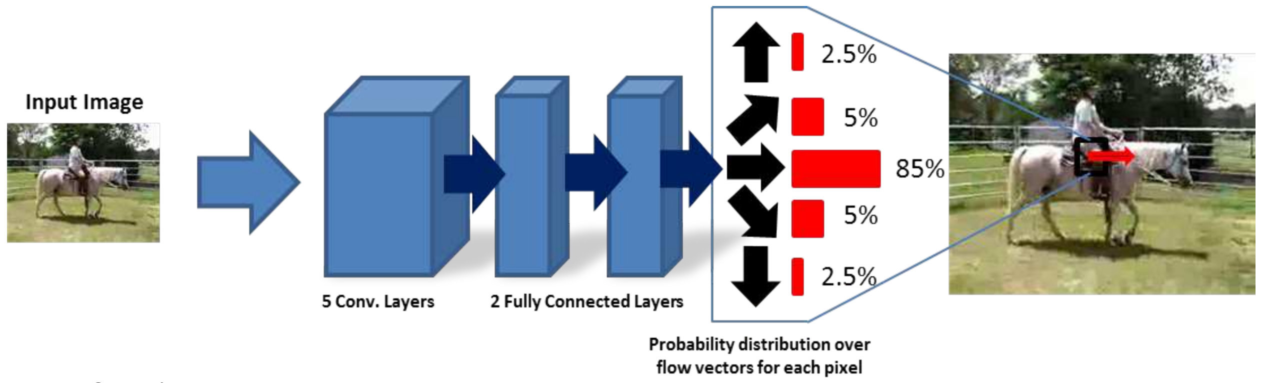


Figure 2: **Overview**. Our network is similar to the standard 7-layer architecture [12] used for many recognition tasks. We take a 200x200 image as input. However, we use a spatial softmax as the final output. For every pixel in the image we predict a distribution of various motions with various directions and magnitudes. We can combine a weighted average of these vectors to produce the final output for each pixel. For computational reasons, we predict a coarse 20x20 output.

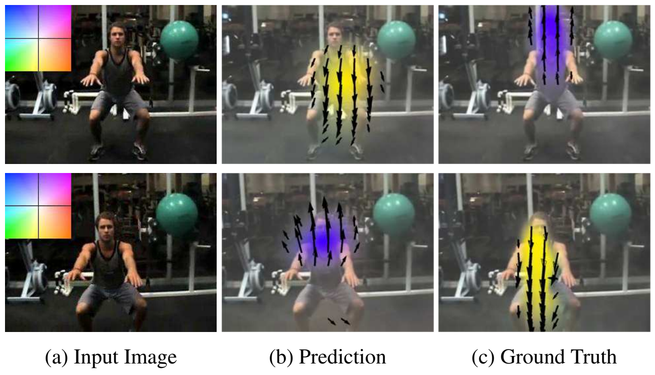


Figure 3: Consider the images on the left. Is the man squatting up or down? The bottom is near completion (or just starting), and the top image is right in the middle of the action. Our dataset contains a large number of ambiguous images such as these. In our evaluation we consider the underlying distribution of movements predicted by our network. It is highly likely that this man is going to move up or down, but unlikely that he will veer off to the left or right.

dicted motion of each and every pixel in terms of optical flow. We propose to use CNNs as the underlying learning algorithm for this task. However, there are a few questions that need to be answered: what is a good output space, and what is a good loss function? Should we model optical flow prediction as a regression or a classification problem? What is a good architecture to solve this problem? We now discuss these issues below in detail.

3.1. Regression as Classification

Intuitively, motion estimation can be posed as a regression problem since the space is continuous. Indeed, this is exactly the approach used in [18], where the authors used structured random forests to regress the magnitude and direction of the optical flow. However, such an approach has one drawback: such an output space tends to smoothen results to the mean. Interestingly, in a related regression problem of surface normal prediction, researchers have proposed reformulating structured regression as a classification problem [26, 14]. Specifically, they quantize the surface normal vectors into a codebook of clusters and then output space becomes predicting the cluster membership. In our work, we take a similar approach. We quantize optical flow vectors into 40 clusters by k-means. We can then treat the problem in a manner similar to semantic segmentation, where we classify each region as the image as a particular cluster of optical flow. We use a soft-max loss layer at the output for computing gradients.

However, at test time, we create a soft output by considering the underlying distribution of all the clusters, taking a weighted-probability sum over all the classes in a given pixel for the final output. Transforming the problem into classification also leads directly to a discrete probability distribution over vector directions and magnitudes. As the problem of motion prediction can be ambiguous depending on the image (see Figure 3), we can utilize this probability distribution over directions to measure how informative our predictions are. We may be unsure if the man in Figure 3 is sitting down or standing up given only the image, but we can be quite sure he will not turn right or left. In the same way, our network can rank upward and downward facing clusters much higher than other directions. Even if the ground truth is upward, and the highest ranked cluster is downward, it may be that the second-highest cluster is also upward. Because the receptive fields are shared by the top layer neurons, the output trends to a globally coherent movement. A discrete probability distribution, through classification, allows an easier understanding of how well our network may be performing.

3.2. Network Design

Our model is similar to the standard seven-layer architecture from [12]. To simplify the description, we denote the convolutional layers as $C(k, s)$, which indicates that there are k kernels, each having the size of $s \times s$. During convolution, we set all the strides to 1 except for the first layer, which is 4. We also denote the local response normalization layer as LRN, and the max-pooling layer as MP. The stride for pooling is 2 and we set the pooling operator size as 3×3. Finally, $F(n)$ denotes fully connected layer with n neurons. Our network architecture can be described as:

$C(96,11) \rightarrow LRN \rightarrow P \rightarrow C(256,5) \rightarrow LRN \rightarrow P \rightarrow C(384,3) \rightarrow C(384,3) \rightarrow C(256,3) \rightarrow P \rightarrow F(4096) \rightarrow F(4096)$. We used a modified version of the popular Caffe toolbox [8] for our implementation. For computational simplicity, we use 200x200 windows as input. We used a learning rate of 0.0001 and a stepsize of 50000 iterations. Other network parameters were set to default. The only exception is that we used Xavier initialization of parameters. Instead of using the default softmax output, we used a spatial softmax loss function from [26] to classify every region in the image. This leads to a $M \times N \times C$ softmax layer, where M is the number of rows, N is the number of columns, and C is the number of clusters in our codebook. We used $M = 20$, $N = 20$, and $C = 40$ for a softmax layer of 16,000 neurons. Our softmax loss is spatial, summing over all the individual region losses. Let I represent the image and Y be the ground truth optical flow labels represented as quantized clusters. Then our spatial loss function $L(I,Y)$ is:

$$L(I,Y) = -\sum_{i=1}^{M\times N}\sum_{r=1}^{C}(\mathbb{1}(y_i = r)\log F_{i,r}(I)) \tag{1}$$

where $F_{i,r}(I)$ represents the probability that the ith pixel will move according to cluster r. $\mathbb{1}(y_i = r)$ is an indicator function.

Data Augmentation: For many deep networks, datasets which are insufficiently diverse or too small will lead to overfitting. [20] and [9] show that training directly on datasets such as the UCF-101 for action classification leads to overfitting, as there is only on the order of tens of thousands of videos in the dataset. However, our problem of single-frame prediction is different from this task. We find that we are able to build a generalizable representation for prediction by training our model over 350,000 frames from the UCF-101 dataset as well as over 150,000 frames from the HMDB-51 dataset. We benefit additionally from data augmentation techniques. We randomly flip each image as well as use randomly cropped windows. For each input, we also mirror or flip the respective labels. In this way we are able to avoid spatial biases (such as humans always appearing in the middle of the image) and train a general model on a far smaller set of videos than for recognition tasks.

Labeling: We automatically label our training dataset with an optical flow algorithm. We chose the publicly available implementation of DeepFlow [27] to compute optical flow. The UCF-101 and the HMDB-51 dataset use realistic, sometimes low-quality videos from a wide variety of sources. They often suffer from compression artifacts. Thus, we aim to make our labels somewhat less noisy by taking the average optical flow of five future frames for each image. The videos in these datasets are also unstabilized. [25] showed that action recognition can be greatly improved with camera stabilization. In order to further denoise our labels, we wish to focus on the motion of objects inside the image, not the camera motion. We thus use the stabilization portion of the implementation of [25] to automatically stabilize videos using an estimated homography.

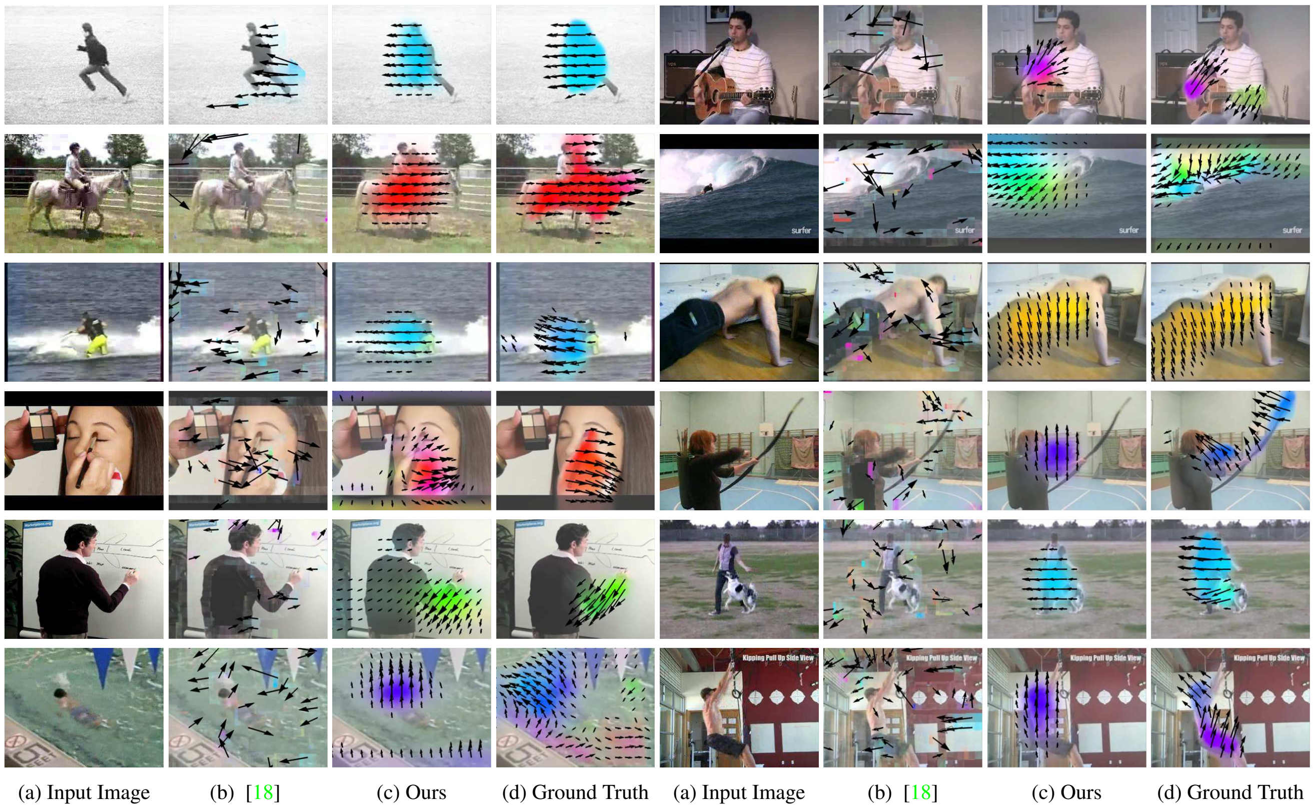
4. Experiments

For our experiments, we mostly focused on two datasets, the UCF-101 and HMDB-51, which have been popular for action recognition. For both of these datasets, we compared against baselines using 3-fold cross validation with the splits specified by the dataset organizers. We also evaluated our method on the KTH [16] dataset using the exact same configuration in [18] with DeepFlow. Because the KTH dataset is very small for a CNN, we finetuned our UCF-101 trained network on the training data. For training, we subsampled frames by a factor of 5. For testing, we sampled 26,000 frames per split. For our comparison with AlexNet finetuning, we used a split which incorporated a larger portion of the training data. We will release this split publicly. We used three baselines for evaluation. First we used the technique of [18], a SRF approach to motion prediction. We took their publicly available implementation and trained a model according to their default parameters. Because of the much larger size of our datasets, we had to sample SIFT-patches less densely. We also use a Nearest-Neighbor baseline using both fc7 features from the pre-trained AlexNet network as well as pooled-5 features. Finally, we compare unsupervised training from scratch with finetuning on the supervised AlexNet network.

4.1. Evaluation Metrics

Because of the complexity and sometimes high level of label ambiguity in motion prediction, we use a variety of metrics to evaluate our method and baselines. Following from [18], we use traditional End-Point-Error, measuring the Euclidean distance of the estimated optical flow vector from the ground truth vector. In addition, given vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, we also measure direction similarity using the cosine similarity distance: $\frac{\mathbf{x}_1^T\mathbf{x}_2}{\|\mathbf{x}_1\|\|\mathbf{x}_2\|}$ and orientation similarity (angle taken on half-circle): $\frac{|\mathbf{x}_1^T\mathbf{x}_2|}{\|\mathbf{x}_1\|\|\mathbf{x}_2\|}$. The orientation similarity measures how parallel is predicted optical flow vector with respect to given ground truth optical flow vector. Some motions may be strictly left-right or up-down, but the exact direction may be ambiguous. This measure accounts for this situation.

We choose these metrics established by earlier work. However, we also add some additional metrics to account for the level of ambiguity in many of the test images. As



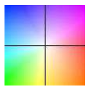


Figure 4: Qualitative results from our method for the single frame model. While [18] is able to predict motion in the KTH dataset (top left), we find our network strongly outperforms the baseline on more complex datasets. Our network can find the active elements in the scene and correctly predict future motion based on the context in a wide variety and scenes and actions. The color coding is on the right.

[18] notes, EPE is a poor metric in the case where motion is small and may reasonably proceed in more than one possible direction. We thus additionally look at the underlying distribution of the predicted classes to understand how well the algorithm accounts for this ambiguity. For instance, if we are shown an image as in Figure 3, it is unknown if the man will move up or down. It is certainly the case, however, that he will not move right or left. Given the probability distribution over the quantized flow clusters, we check to see if the ground truth is within the top probable clusters. For the implementation of [18], we create an estimated probability distribution by quantizing the regression output from all the trees and then, for each pixel, we bin count the clusters over the trees. For Nearest-Neighbor we take the top-N matched frames and use the matched clusters in each pixel as our top-N ranking. We evaluate over the mean rank of all pixels in the image. Following [18], we also evaluate over the Canny edges. Because of the simplicity of the datasets in [18], Canny edges were a good approximation for measuring the error of pixels of moving objects in the scene. However, our data includes highly cluttered scenes that incorporate multiple non-moving objects. In addition, we find that our network is very effective at identifying moving vs non-moving elements in the scene. We find that the difference between overall pixel mean and Canny edges is very small across all metrics and baselines. Thus, we also evaluate over the moving pixels according to the ground-truth. Moving pixels in this case includes all clusters in our codebook except for the vector of smallest magnitude. While unfortunately this metric depends on the choice of codebook, we find that the greatest variation in performance and ambiguity lies in predicting the direction and magnitude of the active elements in the scene.

4.2. Qualitative Results

Figure 4 shows some of our qualitative results. For single frame prediction, our network is able to predict motion in many different contexts. We find that while [18] is able to make reasonable predictions on the KTH, qualitative performance collapses once the complexity and size of the dataset increases. Although most of our datasets consist of human actions, our model can generalize beyond simply detecting general motion on humans. Our method is able to successfully predict the falling of the ocean wave in the second row, and it predicts the motion of the entire horse in the first row. Furthermore, our network can specify motion depending on the action being performed. For the man

UCF-101

Method	EPE	EPE-Canny	EPE-NZ
SRF [18]	1.30	1.23	3.24
NN pooled-5	2.31	2.20	4.40
NN fc7	2.24	2.16	4.27
Ours-HMDB	1.35	1.26	3.26
Ours	**1.27**	**1.17**	**3.19**

—	Dir	Dir-Canny	Dir-NZ
SRF [18]	.004	.000	-.013
NN pooled-5	-.001	-.001	-.067
NN fc7	-.005	-.006	-.060
Ours-HMDB	0.017	0.007	0.032
Ours	**.045**	**.025**	**.092**

—	Orient	Orient-Canny	Orient-NZ
SRF [18]	.492	.600	.515
NN pooled-5	.650	.650	.677
NN fc7	.649	.649	.672
Ours-HMDB	.653	.653	.672
Ours	**.659**	**.657**	**.688**

—	Top-5	Top-5-Canny	Top-5-NZ
SRF [18]	79.4%	81.7%	10.0%
NN pooled-5	77.8%	79.5%	20.0%
NN fc7	78.3%	79.9%	18.8%
Ours-HMDB	88.7%	90.0%	60.6%
Ours	**89.7%**	**90.5%**	**65.0%**

—	Top-10	Top-10-Canny	Top-10-NZ
SRF [18]	82.2%	84.4%	17.2%
NN pooled-5	83.2%	85.3%	32.9%
NN fc7	84.0%	85.4%	32.3%
Ours-HMDB	95.6%	95.9%	88.8%
Ours	**96.5%**	**96.7%**	**90.9%**

Table 1: Single-image evaluation using the 3-fold split on UCF-101. Ours-HMDB represents our network trained only on HMDB data. The Canny suffix represents pixels on the Canny edges, and the NZ suffix represents moving pixels according to the ground-truth. NN represents a nearest-neighbor approach. Dir and Orient represent direction and orientation metrics respectively. For EPE, less is better, and for other metrics, higher is better. With the exception of Orient-NZ against both NN features, all differences against our model are significant at the 5% level with a paired t-test.

HMDB-51

Method	EPE	EPE-Canny	EPE-NZ
SRF [18]	1.23	1.20	3.46
NN pooled-5	2.51	2.49	4.89
NN fc7	2.43	2.43	4.69
Ours-UCF	1.30	1.26	3.49
Ours	**1.21**	**1.17**	**3.45**

—	Dir	Dir-Canny	Dir-NZ
SRF [18]	.000	.000	-.010
NN pooled-5	-.008	-.007	-.061
NN fc7	-.007	-.005	-.061
Ours-UCF	.016	.011	.003
Ours	**.016**	**.012**	**.030**

—	Orient	Orient-Canny	Orient-NZ
SRF [18]	.461	.557	.495
NN pooled-5	.631	.631	.644
NN fc7	.630	.631	.655
Ours-UCF	.634	.634	.664
Ours	**.636**	**.636**	**.667**

—	Top-5	Top-5-Canny	Top-5-NZ
SRF [18]	81.9%	83.6%	13.5%
NN pooled-5	76.3%	77.8%	14.0%
NN fc7	77.3%	78.7%	13.5%
Ours-UCF	89.4%	89.9%	60.8%
Ours	**90.2%**	**90.5%**	**61.0%**

—	Top-10	Top-10-Canny	Top-10-NZ
SRF [18]	84.4%	86.1%	22.1%
NN pooled-5	82.9%	84.0%	23.9%
NN fc7	83.6%	84.4%	23.2%
Ours-UCF	95.8%	95.9%	87.6%
Ours	**95.9%**	**95.9%**	**87.5%**

Table 2: Single-image evaluation using the 3-fold split on HMDB-51. Ours-UCF represents our network trained only on UCF data. The Canny suffix represents pixels on the Canny edges, and the NZ suffix represents moving pixels according to the ground-truth. NN represents a nearest-neighbor approach. Dir and Orient represent direction and orientation metrics respectively. For EPE, less is better, and for other metrics, higher is better. With the exception of EPE-NZ against SRF, all differences against our model are significant at the 5% level with a paired t-test.

playing guitar and the man writing on the wall, the arm is the most salient part to be moved. For the man walking the dog and the man doing a pushup, the entire body will move according to the action.

4.3. Quantitative Results

UCF101 and HMDB: We show in tables 1 and 2 that our method strongly outperforms both the Nearest-Neighbor and SRF-based baselines by a large margin on most metrics. This holds true for both datasets. Interestingly, the SRF-based approach seems to come close to ours based on End-Point-Error on all datasets, but is heavily outperformed on all other metrics. This is largely a product of the End-Point-Error metric, as we find that the SRF tends to output the mean (optical flow with very small magnitude). This is consistent with the results found in [18], where actions with low, bidirectional motion can result in higher EPE than predicting no motion at all. When we account for this ambiguity in motion in the top-N metric, however, the difference in performance is large. The most dramatic differences appear over the non-zero pixels. This is due to the fact that most pixels in the image are not going to move, and an algorithm that outputs motion that is small or zero over the entire image will appear to perform artificially well without taking the moving objects into account.

KTH (DeepFlow)

Method	EPE	EPE-Canny	EPE-NZ
[18]	0.21	0.19	1.72
Ours	**0.19**	**0.18**	**1.17**

—	Orient	Orient-Canny	Orient-NZ
[18]	.30	.32	.75
Ours	**.67**	**.67**	**.90**

—	Top-5	Top-5-Canny	Top-5-NZ
[18]	93.9%	94.4%	2.3%
Ours	**99.0%**	**99.0%**	**98.0%**

Table 3: We compare our network fine-tuned on the KTH dataset to [18]. Orient represents orientation metric. NZ and Canny are non-zero and Canny pixels. All differences are significant at the 5% level with a paired t-test.

Pretrained vs. From Scratch

Method	EPE	EPE-Canny	EPE-NZ
Pretrained	1.19	1.12	3.12
From Scratch	1.28	1.21	3.21

—	Orient	Orient-Canny	Orient-NZ
Pretrained	.661	.659	.692
From Scratch	.659	.658	.691

—	Top-5	Top-5-Canny	Top-5-NZ
Pretrained	91.0%	91.1%	65.8%
From Scratch	89.9%	90.3%	65.1%

Table 4: We compare finetuning from ImageNet features to a randomly initialized network on UCF-101. Orient represents orientation metric. NZ and Canny are non-zero and Canny pixels.

Stabilization

Method	EPE	EPE-Canny	EPE-NZ
Unstabilized	1.35	1.28	3.60
Stabilized	1.49	1.42	3.61

—	Orient	Orient-Canny	Orient-NZ
Unstabilized	.641	.641	0.664
Stabilized	.652	.652	0.698

—	Top-5	Top-5-Canny	Top-5-NZ
Unstabilized	88.9%	89.2%	63.4%
Stabilized	88.3%	88.8%	59.7%

Table 5: We also compare our network trained with and without camera stabilization on a split of HMDB. Orient represents orientation metric. NZ and Canny are non-zero and Canny pixels.

KTH: For KTH in table 3, [18] is close to our method in EPE and Orientation, but Top-N suffers greatly because it often output vectors of correct direction but incorrect magnitude. On absolute levels our method seems to perform well on this simple dataset, with the network predicting the correct cluster will over 98% of the time.

Cross Dataset: As both the UCF101 and HMDB dataset are curated by humans, it is important to determine how well our method is able to generalize beyond the structure of a particular dataset. In table 1 we show that training on HMDB (Ours-HMDB) and testing on UCF101 leads only to a small drop in performance. Likewise, training on UCF101 (Ours-UCF101) and testing on HMDB in table 2 shows little performance loss.

Pretraining: We train our representation in a self-supervised manner, using no semantic information. However, do human labels help? We compared finetuning from supervised, pretrained features trained on ImageNet to a randomly initialized network trained only on self-supervised data. The pretrained net has been exposed to far more diverse data, and the network has been trained on explicity semantic information. However, we find in table 4 that the pretrained network yields only a very small improvement in performance.

Stabilization: How robust is the network to the effects of camera motion? We explicitly stabilized the camera in our training data in order for the network to focus on moving objects and not camera motion itself. In table 5 we compare a network trained on data with and without stabilization. We test on stablilized data, and we find even without camera stabilization that the difference in performance is small.

5. Multi-Frame Prediction

Until now we have described an architecture for predicting optical flow given a static image as input. However, it would be interesting to predict not just the next frame but a few seconds into future. How should we design such a network?

We present a proof-of-concept network to predict 6 future frames. In order to predict multiple frames into the future, we take our pre-trained single frame network and output the seventh feature layer into a ”temporally deep” network, using the implementation of [2]. This network architecture is the same as an unrolled recurrent neural network with some important differences. On a high level, our network is similar to the unfactored architecture in [2], with each sequence having access to the image features and the previous hidden state in order to predict the next state. We replace the LSTM module with a fully connected layer as in a RNN. However, we also do not use a true recurrent network. The weights for each sequence layer are not shared, and each sequence has access to all the past hidden states. We used 2000 hidden states in our network, but we predict at most six future sequences. We attempted to use recurrent architectures with the publicly available LSTM implementation from [2]. However, in our experiments they always regressed to a mean trajectory across the data. Our fully connected network has much higher number of parameters than a RNN and therefore highlights the inherent difficulty of this task. Due to the much larger size of the state space, we do not predict optical flow for each and every pixel. Instead, we use kmeans to created a codebook of 1000 possible optical flow frames, and we predict one of 1000 class as output as each time step. This can be thought of as analogous to a sequential prediction problem similar to caption

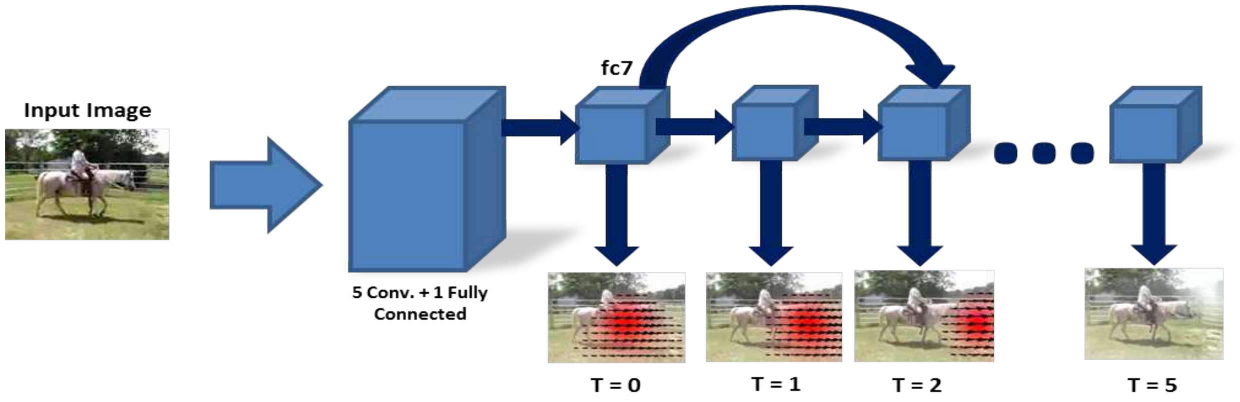


Figure 5: **Overview**. For our multiframe prediction, we predict entire clustered frames of optical flow as a sequence of frames. We take the learned features for our single frame model as our input, and we input them to a series of six fully connected layers, with each layer having access to the states of the past layers.

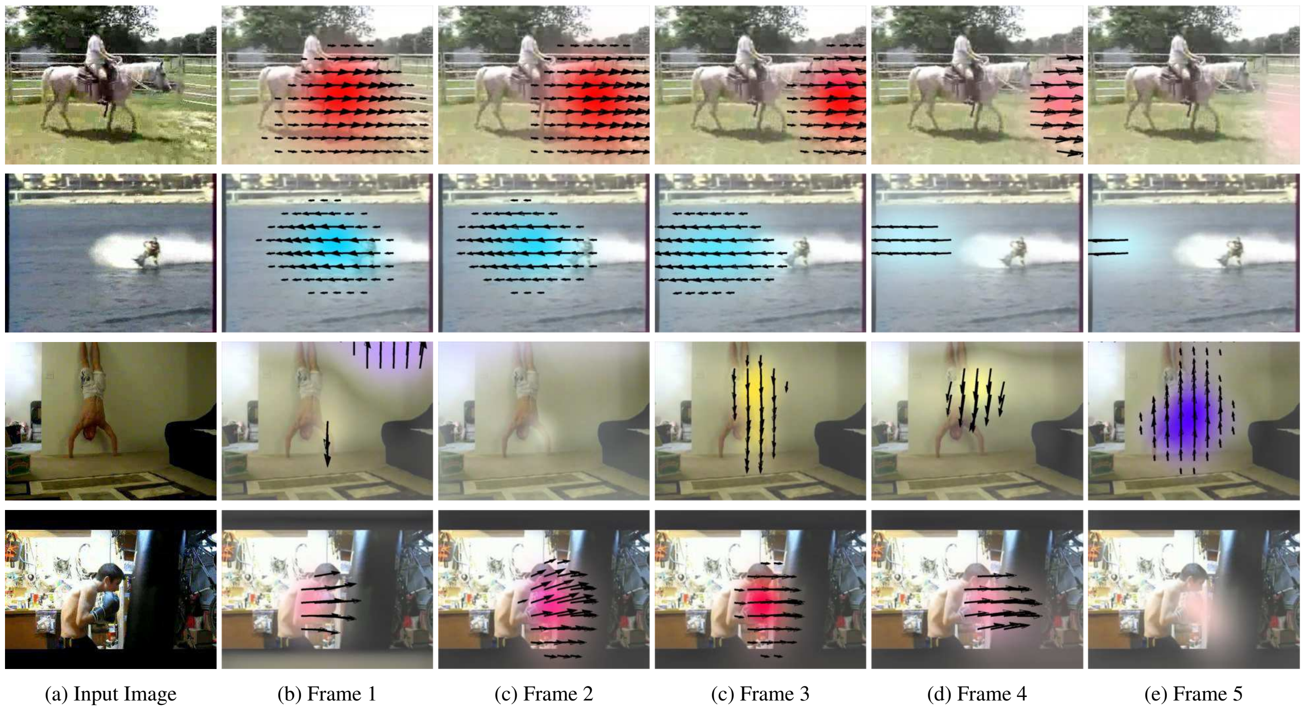


Figure 6: Qualitative results for multi-frame prediction. The five rows represent predictions from our multi-frame model for future frames. Our extension can predict optical flow over multiple frames.

generation. Instead of a sequence of words, our "words" are clusters of optical flow frames, and our "sentence" is an entire trajectory. We used a set number of sequences, six, in our experiments with each frame representing the average optical flow of one-sixth of a second.

6. Conclusion

In this paper we have presented an approach to generalized prediction in static scenes. By using an optical flow algorithm to label the data, we can train this model on a large number of unlabeled videos. Furthermore, our framework utilizes the success of deep networks to outperform contemporary approaches to motion prediction. We find that our network successfully predicts motion based on the context of the scene and the stage of the action taking place. Possible work includes incorporating this motion model to predict semantic action labels in images and video. Another possible direction is to utilize the predicted optical flow to predict in raw pixel space, synthesizing a video from a single image.

Acknowledgements: We thank Xiaolong Wang for many helpful discussions. We thank the NVIDIA Corporation for the donation of Tesla K40 GPUs for this research. In addition, this work was supported by NSF grant IIS1227495.

References

[1] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 92(1):1–31, 2011. 1

[2] 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. 7

[3] D. Fouhey and C. L. Zitnick. Predicting object dynamics in scenes. In *CVPR*, 2014. 2

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

[5] M. Hoai and F. De la Torre. Max-margin early event detectors. *IJCV*, 107(2):191–202, 2014. 2

[6] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 2

[7] D.-A. Huang and K. M. Kitani. Action-reaction: Forecasting the dynamics of human interaction. In *ECCV*. 2014. 2

[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*, 2014. 4

[9] 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, 4

[10] K. Kitani, B. Ziebart, D. Bagnell, and M. Hebert. Activity forecasting. In *ECCV*, 2012. 1, 2

[11] H. S. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. In *RSS*, 2013. 2

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 2, 3

[13] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011. 2

[14] L. Ladickỳ, Z. Bernhard, and M. Pollefeys. Discriminatively trained dense surface normal estimation. 3

[15] T. Lan, T.-C. Chen, and S. Savarese. A hierarchical representation for future action prediction. In *ECCV*. 2014. 2

[16] I. Laptev, B. Caputo, et al. Recognizing human actions: A local SVM approach. In *ICPR*, 2004. 2, 4

[17] C. Liu, J. Yuen, and A. Torralba. Sift Flow: Dense correspondence across scenes and its applications. *PAMI*, 2011. 2

[18] S. L. Pintea, J. C. van Gemert, and A. W. Smeulders. Déjà vu: Motion prediction in static images. In *ECCV*. 2014. 2, 3, 4, 5, 6, 7

[19] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra. Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604*, 2014. 2

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

[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*, 2012. 2

[22] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using LSTMs. In *ICML*, 2015. 2

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*. 2015. 2

[24] J. Walker, A. Gupta, and M. Hebert. Patch to the future: Unsupervised visual prediction. In *CVPR*, 2014. 1, 2

[25] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, Sydney, Australia, 2013. 4

[26] X. Wang, D. F. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *CVPR*. 2015. 3, 4

[27] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. In *ICCV*, 2013. 4

[28] J. Yuen and A. Torralba. A data-driven approach for event prediction. In *ECCV*, 2010. 2

[29] N. Zhang, M. Paluri, M. Ranzato, T. Darrell, and L. Bourdev. Panda: Pose aligned networks for deep attribute modeling. In *CVPR*, 2014. 2