Video Representations of Goals
Emerge from Watching Failure

Dave Epstein and Carl Vondrick
Columbia University
aha.cs.columbia.edu

Abstract

We introduce a video representation learning framework that models the latent goals behind observable human action. Motivated by how children learn to reason about goals and intentions by experiencing failure, we leverage unconstrained video of unintentional action to learn without direct supervision. Our approach models videos as contextual trajectories that represent both low-level motion and high-level action features. Experiments and visualizations show the model is able to predict underlying goals and detect when action switches from intentional to unintentional. Although the model is trained with minimal supervision, it is competitive with highly-supervised baselines, underscoring the role of failure examples for learning goal-oriented video representations.

1 Introduction

Consider the person in Figure 1, which shows a man heating a wine bottle with a blowtorch. Even if this action is unconventional, we still perceive the action as rational in the context of the goal (to open the bottle). Evidence suggests that this ability to reason about goals is learned before our second birthday [39, 46], and it plays a key role in children’s rapid development of communicative skills [40] and mental representations of the world [2]. However, despite the importance of this problem, learning visual representations of human goals has remained challenging.

Visual action recognition has largely focused on learning to predict action categories from supervision [3, 19], which indicate how a person acted, and not why they acted. Current video representations do not capture information about the underlying goals. We hypothesize that a key missing piece is the lack of examples demonstrating the failure to achieve goals. Similar to how a child learns about goals by experiencing failure, we leverage a large dataset containing both intentional and unintentional real-world action [6] to learn goal-oriented representations of video.

Figure 1: What is this person’s goal? Although only the action is observable, we are still able to predict the goal behind the action. In this paper, we introduce a model to learn goal-oriented video representations in the form of latent action trajectories.

Preprint. Under review.
Learning goal-oriented video representations: We show an overall view of our approach. First, we embed short clips using a 3D CNN to represent short-term motion features. Then, we run the sequence of CNN embeddings through a stack of Transformers, where they interact with each other to finally form a context-adjusted latent action trajectory. The model is trained end-to-end from scratch, with intentionality and temporal coherence losses (depicted top-left). Points along the resultant trajectory are decoded with fine-tuning into various spaces (top-middle).

We present a video model that learns a trajectory representation of action, and encodes goals as the path of the trajectory. We input entire videos to our model by first dividing them into short clips, which are run through a 3D CNN to learn low-level motion features. We then pass the motion features into a Transformer model, which models relations between different periods in its input, and thus represents entire action as context-aware latent trajectories. The whole model is trained from scratch in an end-to-end manner.

Our experiments show that failure is critical for learning to represent goals. We evaluate our model on three visual tasks for goal prediction. First, we experiment on detecting unintentional action in video, and we demonstrate state-of-the-art performance on this task. Second, we evaluate the representation at predicting goals with minimal supervision, which we characterize as structured categories consisting of subject, action, and object triplets. Lastly, we use our representation to automatically “correct” unintentional action and decode these corrections by retrieving from other videos or generating categorical descriptions.

Our main contribution is an approach that (1) models long video sequences as latent-space trajectories with indirect supervision and, in doing so, (2) learns a goal-directed representation of videos. Since the goals are encoded in the path of the trajectory, we also show how to find minimal adjustments to the path to automatically correct unintentional action in video. The remainder of this paper will describe those contributions in detail. Code, data, and models will be released.

2 Related Work

Recognizing action in video: Previous work explores many different approaches to recognizing action in video. Earlier directions develop hand-designed features to process spatio-temporal information for action recognition [24, 22, 43, 33]. Popular deep learning architectures for images were
To allow the evaluation of models trained to learn a representation of goals in video, we annotate the Oops! dataset as shown. We automatically extract subject-verb-object triples and use these to train a decoder on learned representations. The intentional and unintentional action in the dataset span a diverse range of categories.

extended to operate directly on video by modeling time as a third dimension \cite{13, 3, 36, 26, 19}. To deal with variable-length or long video input, previous work frequently takes one of two approaches: pooling or recurrent networks. However, pooling loses spatial and/or temporal connections between different moments of video. Since recurrent networks are sequential, they require selecting important video features ahead of time, without viewing full context. RNNs are also known to struggle to connect between far-apart inputs, which creates significant challenges in modeling long-term video. To address these drawbacks, we propose a 3D-CNN-Transformer model which allows for short-term, granular motion detection combined with a long-term action representation.

Learning about intention: Evidence in developmental psychology quantifies why humans perceive intention \cite{2}, how we perceive it \cite{47, 48, 46}, when we begin to do so \cite{27, 28}, and what allows us to infer the goals behind others’ behavior \cite{35}. While these questions have been studied in early stages of child development, the same abilities have remained a challenge for machines in unconstrained situations. One possible reason for this is a lack of good data. We take advantage of incidental signals in realistic videos \cite{6} to learn video representations.

Leveraging adversarial attacks: We use adversarial gradients \cite{10, 23} to find corrections to the trajectory. Previous work studied adversarial attacks in steganography \cite{14, 51}, software bug-finding \cite{34}, generating CAPTCHAs \cite{42} to fool modern deep nets \cite{32}, generating interesting images \cite{37}, creating real-world 3D objects that trick neural networks \cite{50, 1}, and in training models more robust to test-time adversarial attacks \cite{30, 10, 31}. \cite{17} extend this concept to generative models, setting a new image output as a target label and perturbing latent space. In video, \cite{21, 45} introduce various methods to fool action recognition networks, often on a 3D CNN backbone. We instead utilize adversarial attacks to manipulate and correct unintentional action.

3 Method

In this section, we introduce our framework to learn video representations as trajectories, formulate learning objectives, and use the learned representations to predict goals in video.

3.1 Visual Dynamics as Trajectories

The conventional approach to representing video data is to run each clip through a convolutional network and combine clip representations by pooling to run models on entire sequences \cite{7, 12, 9, 49}. However, these methods do not allow for connections between different moments in video and cannot richly capture temporal relationships, which give rise to goal-directed action. While recurrent networks \cite{15} are more expressive, they require compressing history into a fixed-length vector, which forces models to select relevant visual features without viewing full context and makes reasoning about connections between different moments difficult, especially when they are far apart.
Starting from an initial trajectory, we use model gradients as a signal to correct the course of points representing unintentional action. This corrected trajectory is evaluated by decoding into various feature spaces.

Temporal streams of visual input are highly contextual with both short- and long-term dependencies. We will represent video as a contextually-adjusted trajectory of latent representations in a learned space. Figure 2 illustrates this architecture, which has both a motion and action level:

**Motion Level:** First, we separate video into short clips (or tokens) in order to make initial motion-level observations. Let $x$ be a video, and $x_i$ be a video clip centered at time $i$. We estimate the motion-level features $\phi_i = f(x_i)$ where $f$ is a 3D CNN [20].

**Action Level:** Second, we model relations between $\phi_i$ to construct a contextual trajectory $h_i = g_\phi(\phi_i)$ where $g$ is the Transformer [41]. The Transformer architecture is able to capture relations between the input by performing self-attention among tokens in its input sequence, and outputs a contextual representation across the video. The resulting sequence of embeddings induces a trajectory in the form of a sequence of hidden vectors $h_i$, which we can use for different downstream tasks.

### 3.2 Learning with Indirect Supervision

We train the representation with indirect supervision that is accessible at large scales. We use the following two objectives for learning:

**Action Intentionality:** We train the model to temporally localize when action is unintentional. Let $t \in \mathbb{R}$ be the video frame where the action shifts from intentional to unintentional (which we assume is labeled [6]). For each video clip $x_i$, we set the target $y_{\text{fail}}(t) \in \{0, 1, 2\}$ according to whether the labeled $t$ happens before, during, or after the clip $x_i$. The model estimates $\hat{y}_{\text{fail}} = \text{softmax}(w_1^T h_i)$ with a linear projection where $w_1$ is a jointly learned projection matrix to $\mathbb{R}^3$. We train with a cross-entropy loss between $\hat{y}_{\text{fail}}$ and $y_{\text{fail}}$ where the class weight is set to the inverse frequency of the class label to balance training. We label this loss $L_{\text{fail}}$.

**Temporal Consistency:** We also train the model to learn temporal dynamics with a self-supervised consistency loss [12, 29, 8, 44, 18, 5]. Let $y_{\text{mp}} = 1$ indicate the sequence is consistent. We predict whether the input sequence is temporally consistent with $\hat{y}_{\text{mp}} = \sigma(w_2^T h_0)$ where $w_2$ is a jointly learned projection to $\mathbb{R}$. We train with the binary cross-entropy loss between $y_{\text{mp}}$ and $\hat{y}_{\text{mp}}$. We label this loss $L_{\text{mp}}$ (next sequence prediction).

We create inconsistent sequences as follows. For each video sequence in the batch, we bisect the sequence into two parts at a random index with probability $p_{\text{split}} = 0.5$. For these sequences, we perturb one or both of the video segments with probability $p_{\text{perturb}} = 0.5$. When perturbing, we swap the order of the two sequences with probability $p_{\text{swap}} = 0.3$, otherwise we pick a randomly sized subsequence from another video sequence in the batch to replace one of the two segments.

**Training:** To train our model, we set the overall loss as $L = L_{\text{fail}} + \lambda L_{\text{mp}}$, where $\lambda$ is a hyperparameter controlling the importance of the coherence loss. We set $\lambda = 0.5$ to balance the magnitudes of the losses. We sample sequences of one-second long clips, run each clip $x_i$ through the entire model, and calculate the gradients. We optimize the loss with stochastic gradient descent. At inference time, we run entire continuously-sampled videos through our model. For additional details, please see the Supplementary Material.
### 3.3 Completing Goals by Auto-Correcting Trajectories

We use this learned representation in order to complete the goals of people in the scene [27, 38]. However, since the model is trained with indirect supervision, the trajectories $h$ are not supervised with goal states. We propose to formulate goal completion as a latent trajectory prediction problem. Given an observed trajectory of unintentional action $h$, we seek to find a new, minimally modified trajectory $h'$ that is classified as intentional. By analogy to how word processors auto-correct a sentence, we call this process action auto-correct. We illustrate this process in Figure 4.

We find this correction in feature space, not pixel space, to yield interpretable results. We find $\phi$ with goal states. We propose to formulate goal completion as a latent trajectory prediction problem. With the initial part of the original trajectory, we minimize:

$$\min_{\phi'} J$$

where

$$J = \max (0, L_{\text{mp}}^{\text{mp}}(\phi') - L_{\text{mp}}^{\text{mp}}(\phi)) + \lambda \sum_{t} L_{\text{mp}}^{\text{fail}}(\phi'_t)$$

(1)

where $L$s are the original loss functions but with target labels $y^\text{fail}$ overridden to be the intentional class, and $\lambda = 2$ is a scalar to balance the two terms. We only modify $\phi$ on the clips which the model classifies as unintentional in the first place, which we denote $\phi^\text{fail}_{t:T}$. The coherence loss is also truncated by its original value, causing the optimization to favor a trajectory that is no less temporally coherent than the original one.

To solve this optimization problem, we use the iterative target class method [23], which repeatedly runs the input through the model and modifies it in the direction of the desired loss. For every $\phi_t$ corresponding to a clip where action is unintentional, we repeat a gradient attack step towards the target $y^\text{fail}_t = 0$. The complete update is:

$$\phi^{k+1}_{t:T} = \text{clip} \left[ \phi^k_{t:T} - \alpha \text{sign} (\nabla_{\phi_t:T} J), \phi_{t:T} \pm \epsilon \right]$$

(2)

where $\phi^0_{t:T} = \phi$, $\epsilon = 1$ is a small scalar. We repeat this process until the network is “fooled” into classifying the input as intentional action, for at most $k_{\text{max}}$ iterations or until $\arg\max y^\text{fail}_t = 0$. Once the halting condition is satisfied, we run the modified $\phi'$ vectors through the model, yielding a trajectory of corrected action $h'$ that encodes successful completion of the goal. This trajectory can be read out into various spaces (Section 5.1).

In other words, goals are the adversarial examples [10] of failed action – instead of viewing adversarial examples as a bug, we view them as a feature [16].

### 4 Unintentional Action and Goals Dataset

Similar to how children learn about goals by perceiving failed attempts at executing them [28], we hypothesize examples of failure are crucial for learning to discriminate between action and goal. We use the recently released Oops! dataset [6], which is a large collection of videos containing

---

1 We found $k_{\text{max}} = 25$, $\alpha = 0.03$, $\epsilon = 1$ to be reasonable values.
Figure 5: **Decoding the Trajectories:** After estimating the decoder, we read out triplets from different parts of videos. The first row shows intentional action, and the decoder predicts the goal. The second row shows unintentional action, and the decoder now predicts the failure instead. The final row shows unintentional videos that have been auto-corrected, and the decoder returns to predicting goals, suggesting the auto-correct procedure shifts the failed trajectories towards successful ones.

intentional and unintentional action, to train and evaluate our models. Videos in this dataset are annotated with the moment at which action becomes unintentional.\(^2\) Figure 3 shows some example frames. We also use the Kinetics dataset [4] to evaluate models, since it contains a wide range of successful action.

**Goal Annotation:** Established action datasets in computer vision [11, 25] contain annotations about person and object relationships in scenes, but they do not directly annotate the goal, which we need for evaluation of goal prediction. We collect unconstrained natural language descriptions of a subset of videos in the Oops! dataset (4675 training videos and 3404 test videos), prompting Amazon Mechanical Turk workers\(^3\) to answer “What was the goal in this video?” as well as “What went wrong?” We then process these sentences\(^4\) to detect lemmatized subject-verb-object triples, manually correcting for common constructions such as “tries to X” (where the verb lemma is detected as “try”, but we would like “X”). The final vocabulary contains 3615 tokens. Figure 3 shows some example annotations. Detailed statistics for processed SVO triples are provided in the Supplementary Material. We use SVO triples to evaluate the video representations.

5 **Experiments**

We experiment with our model on two tasks: recognizing intentional action, and predicting goals. We train our method from scratch on a dataset of unintentional action [6].

5.1 **Experimental Setup**

**Baselines:** We evaluate the 3D CNN from [6] which is trained from scratch on the action intentionality loss (Section 3.2). We also evaluate a 3D CNN pre-trained on Kinetics action recognition, which is frozen unless indicated otherwise. We also compare to a frozen, randomly initialized network (denoted “Scratch”). We finally consider several ablations of our model. To evaluate each representation, we freeze them and implement different decoders described below.

\(^2\)In addition to the ground truth annotations provided by [6], we run their pretrained model on the unlabeled portion of the training set and collect pseudo-ground-truth, which we found improves performance.

\(^3\)with \(> 10k\) approvals at \(\geq 99\%\) rate

\(^4\)Using the Spacy.io natural language library
Figure 6: **Retrievals from Auto-corrected Trajectories:** We show the nearest neighbors from auto-corrected action trajectories, using our proposed method and a linearization baseline. The retrievals are computed across both the Oops! and Kinetics datasets. The corrected representations yield corrected trajectories that are often embedded close to the goal.

<table>
<thead>
<tr>
<th>Method</th>
<th>Subject Top 1</th>
<th>Subject Top 5</th>
<th>Verb Top 1</th>
<th>Verb Top 5</th>
<th>Object Top 1</th>
<th>Object Top 5</th>
<th>Average Top 1</th>
<th>Average Top 5</th>
<th>All three Top 1</th>
<th>All three Top 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kinetics [3]</td>
<td>26.79</td>
<td>72.34</td>
<td>27.33</td>
<td>52.67</td>
<td>36.01</td>
<td>64.64</td>
<td>30.04</td>
<td>63.22</td>
<td>2.07</td>
<td>16.46</td>
</tr>
<tr>
<td>3D CNN [6]</td>
<td>29.44</td>
<td>72.72</td>
<td>26.42</td>
<td>50.36</td>
<td>44.71</td>
<td>57.89</td>
<td>33.52</td>
<td>60.32</td>
<td>2.86</td>
<td>13.85</td>
</tr>
<tr>
<td>Scratch</td>
<td>23.67</td>
<td>55.73</td>
<td>22.74</td>
<td>45.44</td>
<td>44.82</td>
<td>52.67</td>
<td>30.41</td>
<td>51.28</td>
<td>1.42</td>
<td>8.72</td>
</tr>
<tr>
<td>Our Model</td>
<td><strong>34.31</strong></td>
<td><strong>74.50</strong></td>
<td><strong>29.72</strong></td>
<td><strong>54.17</strong></td>
<td><strong>44.95</strong></td>
<td><strong>58.16</strong></td>
<td><strong>36.32</strong></td>
<td><strong>62.27</strong></td>
<td><strong>3.32</strong></td>
<td><strong>14.39</strong></td>
</tr>
<tr>
<td>Chance</td>
<td>0.14</td>
<td>&lt;0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 2: Comparison of Representations:** To evaluate how well representations encode goals, we freeze them and estimate a linear projection to predict labelled subject-verb-object triples.

**Retrieval:** This decoder does not require further training and performs nearest-neighbor retrieval among one-second long clips in the test sets for the Oops! and Kinetics datasets. While we do not learn a representation using Kinetics, we include it in retrieval to see if auto-corrected actions match with successfully executed goals in Kinetics rather than failed attempts (see Section 5.3). This decoder maintains a lookup table of all clip representations and computes the k-nearest neighbors from different videos using cosine distance.

**Categorization:** We also implement a decoder using the textual labels we gathered on the videos. Here, the task is to describe the goals of the input video using the SVO triplets. We train a decoder to predict the main goal for clips with intentional action, and predict what went wrong for clips with unintentional action. The estimated decoder will describe the video with goal-aware descriptions, for example “athlete wins game”, which is a goal, and not “woman throws ball”, which is an action. We train a linear layer to output a vector for subject, verb, and object. To account for synonyms, we use BERT word embeddings [5], calculating scores using dot product and running them through softmax and a cross-entropy loss.

**5.2 Unintentional Action Detection**

We first evaluate how well the model is able to detect and localize when action deviates from its goal. We use labels from the test set in [6] as the ground truth. We process entire videos with our model, sampling continuous one-second clips as tokens, and take the predicted localization as the center of the clip with maximum probability of failure. We also classify each clip according to its label (intentional, transitional, or unintentional). We show results in Table 1. On the former task, our model is competitive with fine-tuning a fully-supervised Kinetics CNN, despite using less data and less supervision. On classification, our network outperforms the Kinetics network by 14%, showing that representing videos as contextual trajectories is effective.

**5.3 Goal Prediction**

We next evaluate the model at predicting goal descriptions. We train a decoder on the trajectory to read out subject, verb, object triplets. In training, if sentences have more than one extracted SVO, we randomly select one as ground truth. In testing, we average-pool predictions among all clips with intentional action and unintentional action separately and take the maximum over all
Figure 7: Analyzing the Representation: We probe the learned trajectories. (a) shows the neurons with highest correlation to the words in the SVO vocabulary, along with their top-5 retrieved clips. Neurons that detect intentions across a wide range of action and scene appear to emerge, despite only training with binary labels on the intentionality of action. (b) We show six randomly sampled video trajectories in t-SNE space, before and after auto-correct, superimposed over the embeddings for intentional and unintentional action. Visualizations suggest our approach tends to adjust unintentional action in the direction of successful, intentional action.

Table 3: Evaluating Autocorrection: We show effect of auto-correct on SVO decoder predictions (top 5 accuracy and rank assigned to the correct triple). Our model shifts probability mass from unintentional to intentional SVOs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Intentional SVO</th>
<th>Unintentional SVO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Δ Acc.</td>
<td>Δ Rank</td>
</tr>
<tr>
<td>Kinetics [3]</td>
<td>+0.4</td>
<td>-0.3</td>
</tr>
<tr>
<td>3D CNN [6]</td>
<td>+0.3</td>
<td>-0.3</td>
</tr>
<tr>
<td>Ours (linearized)</td>
<td>+0.6</td>
<td>-0.5</td>
</tr>
<tr>
<td>Ours (adversarial)</td>
<td>+1.6</td>
<td>-3.3</td>
</tr>
</tbody>
</table>

sentence SVOs. Each video clip has two pooled predictions: one for intentional action, and one for unintentional action. Table 2 shows our model obtains better top-1 accuracy on all metrics than baselines, including the Kinetics-pretrained model, and is competitive on top-5 accuracy.

5.4 Analysis of Learned Representation

To evaluate how action and goals are embedded in the trajectory representation, we find the minimal “auto-correction” to the unintentional action sequences and probe them. As a comparison, we implement a simple baseline where we linearly extrapolate the trajectory of observed intentional action: if the unintentional action in a sequence of clips \( \{x_i\}_{i=0}^n \) begins at clip \( j \), we extend the trajectory for a clip \( x_k \in \{x_j, \ldots, x_n\} \) by setting \( h_k = h_j + (k-j)h_j-h_0 \).

Figure 6 shows examples of nearest neighbor retrievals of the corrected latent vectors, computing over the Oops! and Kinetics test sets. Despite not training on Kinetics, our representation can adjust video trajectories such that their nearest neighbors are goals being successfully executed. Next, we examine the effects of auto-correction on the frozen SVO decoder. Table 3 shows, for all models, rankings of intentional action SVOs increase and those of unintentional SVOs decrease, but the changes are greatest for our model. Figure 5 visualizes the output of the decoder on the auto-corrected actions.

We finally probe the model’s learned representation to analyze how the learned trajectories are encoded. We measure Spearman’s rho correlation between the activation of neurons in the output vectors \( h \in \mathbb{R}^{512} \) and words in the SVO vocabulary. Each video is an observation containing neuron activations and an indicator variable for whether each word is present in ground truth. Many neurons have significant correlation, and we show the top 3 in Figure 7a, along with the top 5 clips that activate them most. These neurons appear to discover common actions in the Oops! dataset, despite being trained without any action labels. We also visualize trajectories of some videos using t-SNE (Figure
Our model often adjusts trajectories from unintentional action to the region of embedding space with Kinetics videos, shown in the figure as “at goal” action.

6 Conclusion

We introduce an approach to represent videos as contextual trajectories in a learned latent space, leveraging the Transformer architecture. By encoding action as a trajectory, we are able to perform several different tasks, such as decoding to categorical descriptions or manipulating the trajectory. Our experiments show that learning from failure examples, not just successful action, is crucial for learning rich visual representations of goals.

Acknowledgements

We thank Dídac Surís, Mia Chiquier, Amogh Gupta, Ruoshi Liu, Ishaan Chandrafreya, and Boyuan Chen for helpful comments. Funding was provided by DARPA MCS, NSF NRI 1925157, and an Amazon Research Gift. We thank NVidia for donating GPUs.

Broader Impact

Human action recognition is critical for situational awareness applications in robotics, healthcare, and security, which may potentially have a large practical impact on society. For example, predicting the goals of actions could enable machines to better assist and communicate with people. A key limitation in our experiments is that we leverage publicly available video data, which is likely biased to Western cultures. Consequently, the learned representation likely encodes a Western definition of success, which may not generalize to other demographic areas.

References


