Learning and Verification of Task Structure in Instructional Videos

Medhini Narasimhan\textsuperscript{1,2}, Licheng Yu\textsuperscript{2}, Sean Bell\textsuperscript{2}, Ning Zhang\textsuperscript{2}, Trevor Darrell\textsuperscript{1}

\textsuperscript{1}UC Berkeley, \textsuperscript{2}Meta AI

https://medhini.github.io/task_structure

Abstract

Given the enormous number of instructional videos available online, learning a diverse array of multi-step task models from videos is an appealing goal. We introduce a new pre-trained video model, VideoTaskformer, focused on representing the semantics and structure of instructional videos. We pre-train VideoTaskformer using a simple and effective objective: predicting weakly supervised textual labels for steps that are randomly masked out from an instructional video (masked step modeling). Compared to prior work which learns step representations locally, our approach involves learning them globally, leveraging video of the entire surrounding task as context. From these learned representations, we can verify if an unseen video correctly executes a given task, as well as forecast which steps are likely to be taken after a given step. We introduce two new benchmarks for detecting mistakes in instructional videos, to verify if there is an anomalous step and if steps are executed in the right order. We also introduce a long-term forecasting benchmark, where the goal is to predict long-range future steps from a given step. Our method outperforms previous baselines on these tasks, and we believe the tasks will be a valuable way for the community to measure the quality of step representations. Additionally, we evaluate VideoTaskformer on 3 existing benchmarks—procedural activity recognition, step classification, and step forecasting—and demonstrate on each that our method outperforms existing baselines and achieves new state-of-the-art performance.

1. Introduction

Picture this, you’re trying to build a bookshelf by watching a YouTube video with several intricate steps. You’re annoyed by the need to repeatedly hit pause on the video and you’re unsure if you have gotten all the steps right so far. Fortunately, you have an interactive assistant that can guide you through the task at your own pace, verifying each step as you perform it and interrupting you if you make a mistake. A composite task such as “making a bookshelf” involves multiple fine-grained activities such as “drilling holes” and “adding support blocks.” Accurately categorizing these activities requires not only recognizing the individual steps that compose the task but also understanding the task structure, which includes the temporal ordering of the steps and multiple plausible ways of executing a step (e.g., one can beat eggs with a fork or a whisk). An ideal interactive assistant has both a high-level understanding of a broad range of tasks, as well as a low-level understanding of the intricate steps in the tasks, their temporal ordering, and the multiple ways of performing them.

As seen in Fig. 1, prior work \cite{5,4} learns step representations from single short video clips, independent of the task, thus lacking knowledge of task structure. Our model, VideoTaskformer, learns step representations for masked video steps through the global context of all surrounding steps in the video, making our learned representations aware of task semantics and structure.
Our approach for pre-training VideoTaskformer is outlined in Fig. 2. Our framework consists of two steps: pre-training and fine-tuning. During pre-training, VideoTaskformer is trained on weakly labeled data on the pre-training task. For fine-tuning, VideoTaskformer is first initialized with the pre-trained parameters, and a subset of the parameters is fine-tuned using labeled data from the downstream tasks. Each downstream task yields a separate fine-tuned model.

We extend masked language modeling techniques used in BERT [2] and VideoBERT [8] to learn step representations for instructional videos. While BERT and VideoBERT operate on language and visual tokens respectively, VideoTaskformer operates on clips corresponding to steps in an instructional video. By predicting weakly supervised natural language step labels for masked-out clips in the input video, VideoTaskformer learns semantics and long-range temporal interactions between the steps in a task.

**Masked Step Modeling.** Let \( V = \{ v_1, \ldots, v_K \} \) denote the visual clips corresponding to \( K \) steps in video \( V \). The task for pre-training is to predict categorical natural language step labels for the masked-out steps. While we do not have ground truth step labels, we use the weak supervision procedure proposed by [5] to map each clip \( v_i \) to a distribution over step labels \( p(y_i \mid v_i) \) by leveraging the noisy ASR annotations associated with each clip. The distribution \( p(y_i \mid v_i) \) is a categorical distribution over a finite set of step labels \( V \). More details are provided in the Supplemental.

Let \( M \subseteq [1, \ldots, K] \) denote some subset of clip indices (where each index is included in \( M \) with some masking probability \( r \), a hyperparameter). Let \( V_M \) denote a partially masked-out sequence of clips: the same sequence as \( V \) except with clips \( v_i \) masked out for all \( i \in M \).

Let \( f_{VT} \) represent our VideoTaskformer model with parameters \( \theta \). \( f_{VT} \) is composed of a video encoder model \( f_{vid} \) which encodes each clip \( v_i \) independently, followed by a step transformer \( f_{trans} \) operating over the sequence of clip representations, and finally a linear layer \( f_{head} \) (which includes a softmax). The input to the model is an entire video (of size \( K \times L \times H \times W \times 3 \)) and the output is of size \( K \times S \) (where \( S \) is the output dimension of the linear layer).

For the downstream tasks, we extract step-aware representations using \( f_{VT} \) by feeding an unmasked video \( V \) to the model. We then extract the intermediate outputs of \( f_{trans} \) (which are of size \( K \times D \), where \( D \) is the output embedding size).

To predict step labels for masked-out steps at pre-training, VideoTaskformer operates on clips corresponding to steps in an instructional video.  

\[ f_{VT} \]

\[ f_{vid} \]

\[ f_{trans} \]

\[ f_{head} \]

where

- \( f_{vid} \) is a video encoder model,
- \( f_{trans} \) is a step transformer,
- \( f_{head} \) is a linear layer,
- \( S \) is the output size of the linear layer,
- \( D \) is the output embedding size.

**Figure 2:** VideoTaskformer Pre-training (Left). VideoTaskformer \( f_{VT} \) learns step representations for the masked out video clip \( v_i \), while attending to the other clips in the video. It consists of a video encoder \( f_{vid} \), a step transformer \( f_{trans} \), and a linear layer \( f_{head} \), and is trained using weakly supervised step labels. Downstream Tasks (Right). We evaluate step representations learned from VideoTaskformer on 6 downstream tasks.
That is, for each clip \( p \) distribution of step labels.

**Distribution matching loss.** For this objective, we treat the \( S \) steps in \( Y \) as the target distribution for each clip \( v_i \) according to the weak supervision process. That is,

\[
y_i^* = \underset{y \in Y}{\arg \max} \ p(y | v_i).
\]

(1)

We calculate the cross entropy between the predicted and target distributions for each masked out clip, yielding the following expression:

\[
- \log([f_{VT}(V_{i \backslash M}]_j))
\]

(2)

where \( j \) is the index of \( y_i^* \) in \( Y \), i.e., such that \( y_i^* = Y_j \). To get the final training objective for a single masked video \( V_{i \backslash M} \), we sum over all indices \( i \in M \), and minimize with respect to \( \theta \).

**Distribution matching loss.** For this objective, we treat the distribution of step labels \( p(y_i | v_i) \) from weak supervision as the target distribution for each clip \( v_i \). We then compute the KL Divergence between the prediction distribution \( f_{VT}(V_{i \backslash M}) \) and the target distribution \( p(y_i | v_i) \) as follows:

\[
\sum_{j'=1}^S p(Y_{j'} | v_i) \log \frac{p(Y_{j'} | v_i)}{[f_{VT}(V_{i \backslash M}]_{j'})}
\]

(3)

We sum over all \( i \in M \) and minimize with respect to \( \theta \). Following [5], we use only the top-\( k \) steps in \( p(y_i | v_i) \) and set the probability of the remaining steps to 0. Lin et al. [5] show that the distribution matching loss results in a slight improvement over step classification loss. For VideoTaskformer, we find both objectives to have similar performance and step classification outperforms distribution matching on some downstream tasks. We use \( f_{VT} \) as a feature extractor (layer before softmax) to extract step representations for new video segments.

**Downstream Tasks.** To show that the step representations learned by VideoTaskformer capture task structure and semantics, we evaluate the representations on 6 downstream tasks—3 new tasks which we introduce (mistake step detection, mistake ordering detection, and long-term step forecasting) and 3 existing benchmarks (step classification, procedural activity recognition, and short-term step forecasting). We describe the dataset creation details for our 3 new benchmarks in the Supplemental.

**Mistake Detection.** A critical aspect of step representations that are successful at capturing the semantics and structure of a task is that, from these representations, correctness of task execution can be verified. We consider two axes of correctness: content (what steps are portrayed in the video) and ordering (how the steps are temporally ordered). We introduce 2 new benchmark tasks to test these aspects of correctness.

- **Mistake step detection.** The goal of this task is to identify which step in a video is incorrect. Each input consists of a video \( V = \{v_1, \ldots, v_K\} \) with \( K \) steps. \( V \) is identical to some unaltered video \( V_1 \) that demonstrates a correctly executed task, except that step \( v_j \) (for some randomly selected \( j \in [1, \ldots, K]\)) is replaced with a random step from a different video \( V_2 \). The model needs to predict the index \( j \) of the incorrect step in the video.

<table>
<thead>
<tr>
<th>Downstream Model</th>
<th>Base Model</th>
<th>Pre-training Supervision</th>
<th>Pre-training Dataset</th>
<th>Acc (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer</td>
<td>TimeSformer [1]</td>
<td>Supervised: action labels</td>
<td>Kinetics</td>
<td>34.7</td>
</tr>
<tr>
<td>LwDS: Transformer</td>
<td>TimeSformer</td>
<td>Unsupervised: k-means on ASR</td>
<td>HT100M</td>
<td>34.0</td>
</tr>
<tr>
<td>LwDS: Transformer w/ KB Transfer</td>
<td>TimeSformer</td>
<td>Distant supervision</td>
<td>HT100M</td>
<td>39.4</td>
</tr>
<tr>
<td>VideoTF (SC; fine-tuned) w/ KB Transfer</td>
<td>TimeSformer</td>
<td>Unsupervised: NN on ASR</td>
<td>HT100M</td>
<td>35.1</td>
</tr>
<tr>
<td>VideoTF (SC; linear-probe) w/ KB Transfer</td>
<td>TimeSformer</td>
<td>Distant supervision</td>
<td>HT100M</td>
<td>39.2</td>
</tr>
<tr>
<td>VideoTF (DM) w/ KB Transfer</td>
<td>TimeSformer</td>
<td>Distant supervision</td>
<td>HT100M</td>
<td>40.1</td>
</tr>
<tr>
<td>VideoTF (DM) w/ KB Transfer</td>
<td>TimeSformer</td>
<td>Distant supervision</td>
<td>HT100M</td>
<td>41.5</td>
</tr>
<tr>
<td>VideoTF (DM) w/ KB Transfer</td>
<td>TimeSformer</td>
<td>Distant supervision</td>
<td>HT100M</td>
<td>42.4</td>
</tr>
</tbody>
</table>

Table 1: Accuracy of different methods on the short-term step forecasting dataset.

<table>
<thead>
<tr>
<th>Downstream Model</th>
<th>Base Model</th>
<th>Pre-training Supervision</th>
<th>Pre-training Dataset</th>
<th>Acc (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer (ASR text) w/ Task label</td>
<td>MPNet</td>
<td>Supervised: action labels</td>
<td>HT100M</td>
<td>40.1</td>
</tr>
<tr>
<td>Transformer</td>
<td>SlowFast [3]</td>
<td>Supervised: action labels</td>
<td>HT100M</td>
<td>17.0</td>
</tr>
<tr>
<td>Transformer</td>
<td>TimeSformer [1]</td>
<td>Supervised: action labels</td>
<td>HT100M</td>
<td>40.1</td>
</tr>
<tr>
<td>Transformer w/ Task label</td>
<td>TimeSformer</td>
<td>Distant supervision</td>
<td>HT100M</td>
<td>41.3</td>
</tr>
<tr>
<td>LwDS: Transformer w/ Task label</td>
<td>TimeSformer</td>
<td>Distant supervision</td>
<td>HT100M</td>
<td>40.2</td>
</tr>
<tr>
<td>VideoTF (DM)</td>
<td>TimeSformer</td>
<td>Distant supervision</td>
<td>HT100M</td>
<td>46.4</td>
</tr>
<tr>
<td>VideoTF (DM) w/ Task label</td>
<td>TimeSformer</td>
<td>Distant supervision</td>
<td>HT100M</td>
<td>46.4</td>
</tr>
</tbody>
</table>

Table 2: Accuracy of different methods on the long-term step forecasting dataset.
The task is to predict the step labels for the next 5 steps, i.e., the next step given the previous n steps. The steps are randomly permuted with a 50% probability. The model needs to predict whether the steps are ordered correctly or are permuted.

**Step Forecasting.** Here we test the network’s capabilities in anticipating future steps given one or more past clips of a video.

- **Short-term forecasting.** Consider a video \( V = \{v_1, \ldots, v_K \} \) where \( v_i \) denotes a step, and \( V \) has step labels \( \{y_1, \ldots, y_K \} \), where \( y_i \in Y \), the finite set of all step labels in the dataset. Short-term forecasting involves predicting the step label \( y_{n+1} \) given the previous \( n \) segments \( \{v_1, \ldots, v_n\} \) [5].

- **Long-term step forecasting.** Given a single step \( v_i \) in a video \( V = \{v_1, \ldots, v_K \} \) with step labels \( \{y_1, \ldots, y_K \} \), the task is to predict the step labels for the next 5 steps, i.e., \( \{y_{i+1}, y_{i+2}, \ldots, y_{i+5}\} \). This task is particularly challenging since the network receives very little context—just a single step—and needs to leverage task information learned during training from watching multiple different ways of executing the same task.

**Procedural Activity Recognition.** The goal of this task is to recognize the procedural activity (i.e., task label) from a long instructional video. The input to the network is all the \( K \) video clips corresponding to the steps in a video, \( V = \{v_1, \ldots, v_K\} \). The task is to predict the video task label \( t \in T \) where \( T \) is the set of all task labels for all the videos in the dataset.

**Step Classification.** In this task, the goal is to predict the step label \( y_i \in Y \) given the video clip corresponding to step \( v_i \) from a video \( V = \{v_1, \ldots, v_K\} \). No context other than the single clip is given. Therefore, this task requires fine-grained recognition capability, which would benefit from representations that contain information about the context in which a step gets performed.

For all of the above tasks, we use the step and task label annotations as supervision. We show the “zero-shot” performance of VideoTaskformer by keeping the video model \( f_{vid} \) and the transformer layer \( f_{trans} \) fixed and only fine-tuning a linear head \( f_{head} \) on top of the output representations. Additionally, we also show fine-tuning results where we keep the base video model \( f_{vid} \) fixed and fine-tune the final transformer \( f_{trans} \) and the linear layer \( f_{head} \) on top of it. The network is fine-tuned using cross-entropy loss with supervision from the step labels for all downstream tasks.

We provide implementation details of our method and describe the datasets and evaluation metrics in the supplemental.

3. Experiments

We evaluate VideoTaskformer (VideoTF) and compare it with existing baselines on 6 downstream tasks: step classification, procedural activity recognition, step forecasting, mistake step detection, mistake ordering detection, and long-term forecasting. Here we report results on mistake detection 3 and forecasting tasks Tab. 2 and 1. The datasets, metrics, baselines, ablations, and additional results are included in the Supplemental.

For short-term forecasting in Tab. 1, we achieve a 3% improvement over LwDS and our unsupervised pre-training using NN with ASR outperforms previous unsupervised methods. We also note that linear-probe outperforms baselines in Tab. 1. VideoTF achieves a strong improvement of 5% over LwDS on the long-term forecasting task 2, 4% on mistake step detection 3, and 4% on mistake ordering detection 3. Adding task labels improves performance on all three tasks.

4. Conclusion

In this work, we introduce a new video model, VideoTaskformer, for learning contextualized step representations through masked modeling of steps in instructional videos. We also introduce 3 new benchmarks: mistake step detection, mistake order detection, and long term forecasting. We demonstrate that VideoTaskformer improves performance on 6 downstream tasks, with particularly strong improvements in detecting mistakes in videos and long-term forecasting. Our method opens the possibility of learning to execute a variety of tasks by watching instructional videos; imagine learning to cook a complicated meal by watching a cooking show.
Acknowledgements. We would like to thank Suvir Mirchandani for his help with experiments and paper writing. This work was supported in part by DoD including DARPA’s LwLL, PTG and/or SemaFor programs, as well as BAIR’s industrial alliance programs.

References


