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.

3.3. Implementation Details

Step labels from Weak Supervision. To train VideoTaskformer, we require step annotations, i.e., step labels with start and end timestamps in the video, for a large corpus of instructional videos. Unfortunately, this is difficult to obtain manually and datasets that provide these annotations, like COIN and CrossTask, are small in size (∼10K videos). To overcome this issue, the video speech transcript can be mapped to steps in WikiHow and used as a weak form of supervision [13]. The intuition behind this is that WikiHow steps are less noisy compared to transcribed speech.

The WikiHow dataset contains a diverse array of articles with step-by-step instructions for performing a range of tasks. Denote the steps across all T tasks in WikiHow as $s = \{s_1, \ldots, s_N\}$, where s_n represents the natural language title of the nth step in s, and N is the number of steps across all tasks in WikiHow. Each step s_n contain a lengthy language-based description which we denote as y_n.

Consider a video with K sentences in the automatic speech transcript denoted as $\{a_1, \ldots, a_K\}$. Each sentence is accompanied by a $\{start, end\}$ timestamp to localize it in the video. This yields K corresponding video segments denoted as $\{v_1, \ldots, v_K\}$. Each video segment v_i is a sequence of F RGB frames having spatial resolution $H \times W$. To obtain the step label for a segment v_i, the corresponding sentence in the transcript a_i is used to find the distribution of the nearest steps in the WikiHow repository. Following [13], we approximate this distribution using a textual similarity measure sim between y_n and a_i:

$$P(y_n|v_i) \approx \frac{\exp\left(\text{sim}(a_i, y_n)\right)}{\sum_{n'} \exp\left(\text{sim}(a_i, y_{n'})\right)}. \quad (4)$$

The authors of [13] found it best to search over all the steps across all tasks (i.e., all y_n), rather than the set of steps for the specific task referenced in the video. The similarity function sim is formulated as a dot product between language embeddings obtained from a pre-trained language model.

Language model. To compare WikiHow steps to the transcribed speech via the sim function, we follow the same setup as in Lin *et al.* [13]. For a fair comparison to the baseline, we use MPNet (paraphrase-mpnet-base-v2) to extract sentence embeddings $\in \mathbb{R}^{768}$.

Video model. VideoTaskformer is a TimeSformer model with a two-layer transformer. Following [13], the TimeSformer is initialized with ViT [8] pre-trained on ImageNet-21K, and is trained on subsampled clips from HowTo100M (with 8 frames sampled uniformly from each 8-second span).

We include additional implementation details in the Supplemental.

4. Datasets and Evaluation Metrics

Pre-training. For pre-training, we use videos and transcripts from the HowTo100M (HT100M) [17] dataset and steps from the WikiHow dataset [4]. HT100M contains 136M video clips from 1.2M long narrated instructional videos, spanning 23k activities such as "gardening" and "personal care." The WikiHow dataset contains 10,588 steps collected from 1059 WikiHow articles which are sourced from the original dataset [11].

Evaluation. All evaluation benchmarks use videos and step annotations from the COIN dataset [25]. COIN consists of 11,827 videos related to 180 different tasks and provides step labels with start and end timestamps for every video. We use a subset of 11,104 videos that were available to download.

As described in Sec. 3.2, we introduce 3 new benchmark tasks in this work: mistake step detection, mistake ordering detection, and long-term step forecasting.

Mistake Step Detection. For creating the mistake step detection dataset, for every video in the COIN dataset, we randomly replace one step with a step from a different video. The network predicts the index of the mistake step. We use the same train/validation/test splits as in COIN and report average accuracy of predicting the mistake step index on the test set.

Mistake Ordering Detection. We synthetically create the mistake ordering detection dataset by randomly shuffling the ordering of the steps in a given video, for 50% of the videos and train the network to predict whether the steps are in the right order or not. While creating the dataset, we repeatedly shuffle the input steps until the shuffled "mistake" order is different from the original valid order. Additionally, we compare the shuffled "mistake" order across all the videos in the same task, to ensure it doesn't match any other video's correct ordering of steps. Despite these two pre-processing checks, there might be noise in the dataset. We report average prediction accuracy on the test split.

Long-term step forecasting. Given a video clip corresponding to a single step, long-term step forecasting involves pre-