Supplementary Materials

In this section, we describe additional implementation details of our method and provide more qualitative results and comparisons on all the 6 downstream tasks.

S1. Implementation Details

Pre-training. The base video model is a Timesformer [4] model with a ViT backbone initialized with ImageNet-21K ViT pretraining [8]. We pre-train our model on 64 A100 GPUs for 20 epochs which takes 120 hours for all the videos in the HowTo100M dataset. We use a batch size of 64 videos (1 video per GPU), each consisting of 12 segments. To train the model, we use SGD optimizer with momentum and weight decay. The learning rate is set to 0.01 and is decayed using a step learning rate policy of 10% decay at steps 15 and 19. We perform a second round of pre-training for 15 epochs using AdamW [15] with a learning rate of 0.00005.

We use a 15% masking ratio during pre-training. Segment transformer f_{trans} is a two layer transformer with 12 video segments as input. Each segment consists of 8 embedding vectors extracted from a series of 8 adjacent 8-second clips from the input video (spanning a total of 64 seconds). It has a 768 embedding dimnesion and 12 heads, along with learnable positional encodings at the beginning. The WikiHow knowledgebase has 10588 step classes all of which are used for training the network with step classification loss. For obtaining the distant supervision from WikiHow and mapping ASR text to step labels in the WikiHow knowledge base, we follow the setup described in [13].

Fine-tuning. For mistake step detection, mistake ordering detection, long term and short term step forecasting, and procedural activity recognition the input consists of 12 segments from the video. We fine-tune only the segment transformer f_{trans} and the linear head f_{head} using cross entropy loss, while the keeping the base TimeSformer video model f_{vid} as a fixed feature extractor. We use a learning rate of 0.005 with a step decay of 10% and train the network for 50 epochs using sgd optimizer.

For the step classification task, we only fine-tune the linear head, while keeping both the base video model and the 2 layer segment transformer fixed. We use a learning rate of 0.005 with a step decay of 10% and train the network for 50 epochs using sgd optimizer.

S2. Additional Quantitative Results

Activity Recognition. In Tab. T2, we include results for activity recognition on EPIC-KITCHENS-100 by fine-tuning our pre-trained model for noun, verb, and action recognition tasks. We outperform all baselines on noun recognition, and are on par with MoViNet (*Kondratyuk et al., CVPR 2021*) on action recognition.

Model	Action (%)	Verb (%)	Noun (%)
MoViNet	**47.7**	**72.2**	57.3
LwDS: TimeSformer	44.4	67.1	58.1
VideoTaskformer (SC)	**47.6**	70.4	**59.8**

Table T1: **Activity Recognition on EPIC-KITCHENS-100.**

Evaluating on step localization: We evaluate our pre-trained embeddings on the action segmentation task in COIN. Following previous work, we train a linear head on top of our fixed features and predict action labels for non-overlapping 1-second input video segments. LwDS attains 67.6% on this task, and our method achieves 69.1%.

Step labels as input: Our method uses visual features since step labels are not always available during inference. Nevertheless, for the purpose of comparison, we assume we have access to ground-truth step labels during inference and include results for all tasks. The results shown in Tab. T2 are from training a single layer transformer on the COIN train set and evaluating on the test set, i.e. there is no pre-training. As expected, using step labels makes the task much simpler and it outperforms using visual features. However, adding task label information to visual features improves performance significantly for all the tasks.

Task	Step Labels(%)		Visual Features (%)	
	-	w/ Task label	-	w/ Task label
Short term forecasting	65	68	20	49
Long term forecasting	50	53	14	40
Mistake Ordering	80	82	60	65
Mistake Step	64	68	28	33

Table T2: **Step labels vs Visual features.**

S3. Additional Qualitative Results

Step Classification. We compare results from our method VideoTaskformer, to the baseline LwDS [13] in Fig. F1. Since our model was trained on the entire video by masking out segments, it has a better understanding of the relationship between different steps in the same task, i.e. learned representations are *"context-aware"*. As a result, it is better at distinguishing the steps within a task and correctly classifies all the steps in the four examples shown here. LwDS on the other hand incorrectly classifies all of the steps. For reference, we show a keyframe from the correct video step clip corresponding to the incorrect step class chosen by LwDS. The input image clips and the correct clips for the LwDS predictions are closely related and contain similar objects of interest, they correspond to different stages of the task and contain different actions. Since our model learns step representations *"globally"* from the whole video, it is able to capture these subtle differences.