dicting the step class label for the next 5 consecutive steps. If there are fewer than 5 next steps we append NULL tokens to the sequence. We compute classification accuracy as the number of correct predictions out of the total number of predictions, ignoring NULL steps. We again use the same splits in the COIN dataset.

Additionally, we evaluate on 3 existing benchmarks: ***step classification*** [25] - predicts the step class label from a single video clip containing one step, ***procedural activity recognition*** [25] - predicts the procedure/task label given all the steps in the input video, and ***short-term step forecasting*** [13] - predicts the class of the step in the next segment given as input the sequence of observed video segments up to that step (excluded).

5. 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. Results are on the datasets described in Sec. 4.

5.1. Baselines

We compare our method to state-of-the-art video representation learning models for action/step recognition. We fine-tune existing models in a similar fashion to ours on the 6 downstream tasks. We briefly describe the best performing baseline, Learning with Distant Supervision (LwDS) [13].

• **TimeSformer (LwDS) [13].** In this baseline model, the TimeSformer backbone is pre-trained on HowTo100M using the Distribution Matching loss (but without any masking of steps as in our model). Next, a single-layer transformer is fine-tuned on top of the pre-trained representations from the base model for each downstream task.

• **TimeSformer w/ KB transfer (LwDS) [13].** For procedural activity recognition and step forecasting, the LwDS baseline is modified to include knowledge base transfer via retrieval of most relevant facts from the knowledge base to assist the downstream task. We also include results by adding the same KB transfer component to our method, referenced as w/ KB Transfer.

• **Steps from clustering ASR text.** As an alternative to the weak supervision from WikiHow, we introduce an unsupervised baseline that relies only on the transcribed speech (ASR text) to obtain steps. [18] introduced an approach to segment a video into steps by clustering visual features along the time axis. It divides the video into non-overlapping segments and groups adjacent video segments together based on a similarity threshold. We adopt a similar approach but in the text space. We compute sentence embeddings for the ASR sentences and group adjacent sentences if their similarity exceeds the average similarity of all sentences across the entire video. We include ablations with different thresholds in the Supplemental.

5.2. Ablations

We evaluate our design choices by ablating different components of our model.

• **Base model.** We report results for different base video models for pre-training: S3D [16], SlowFast [10], TimeSformer [4] trained on HT100M, and TimeSformer trained on Kinetics. For short-term step forecasting, procedural activity recognition, and step classification, the results are from [13].

• **Loss function.** For pre-training VideoTF, we test both the loss functions, Step Classification (SC), and Distribution Matching (DM) described in Sec. 3.

• **Modalities.** For mistake step detection and long-term forecasting tasks, we tried replacing video features with ASR text during fine-tuning. The base model is a language model for embedding sentences in the ASR text and is kept fixed. The ASR text embeddings for all the segments of the video are fed as input to the downstream model, a basic single-layer transformer, which is fine-tuned to each of the tasks.

• **Task label.** For mistake detection and long-term forecasting tasks, we include the task name, e.g. *"Install a Ceiling Fan"*, as input to the downstream model. We compute the sentence embedding of the task label and append it to the list of video tokens fed as input to the model. This domain knowledge provides additional context which boosts the performance on these challenging downstream tasks.

• **Linear-probe vs Fine-tuning.** In linear-probe evaluation, only the f_{head} layer is fine-tuned to each downstream task and in the fine-tuning setting, all the layers of the segment transformer f_{trans} are fine-tuned.

5.3. Results

Quantitative Results. We compare our approach to several baselines on all downstream tasks. For all the downstream tasks, the downstream segment transformer is fine-tuned, except for linear-probe where we keep our pre-trained model fixed and only train a linear head on top of it for each downstream task.

On the step classification task in Tab. 1, VideoTF with step classification loss outperforms LwDS [13] by 2%, indicating that step representations learned with global context also transfer well to a task that only looks at local video clips. In procedural activity recognition (Tab. 2), we see that distribution matching loss works slightly better than step classification loss and our fine-tuned model achieves 1% improvement over the best baseline. For short-term forecasting in Tab. 3, we achieve a 3% improvement over