P^{3}IV: Probabilistic Procedure Planning from Instructional Videos with Weak Supervision

Supplemental Material

He Zhao[1,2] Isma Hadji[1] Nikita Dvornik[1] Konstantinos G. Derpanis[1,2]
Richard P. Wildes[1,2] Allan D. Jepson[1]
[1]Samsung AI Centre - Toronto, [2]York University
{zhufl, kosta, wildes}@eecs.yorku.ca, {isma.hadji, n.dvornik, allan.jepson}@samsung.com

1. Summary

Our supplemental material is organized as follows: Section 2 provides details on the regularization loss used in our adversarial training. Section 3 elaborates on our inference procedure using the Viterbi algorithm. Section 4 includes a thorough description of our implementation details. We then provide a detailed description of the evaluation protocols in Section 5. We describe the baselines we compare to in Section 6. Finally, in Section 7, we present additional ablation experiments and more visualizations.

2. Regularization loss for the generative model

As mentioned in the main paper, Section. 3.5, in addition to the contrastive and cross-entropy losses, we also use an adversarial loss to train the stochastic component of our model. To effectively capture the different modes present in the data and avoid the notorious mode collapse problem of GANs, we apply the recently proposed latent code normalized distance regularizing loss [6, 16–18] defined as

$$\mathcal{L}_{reg} = -\mathbb{E}_{z_1,z_2}\left[\frac{||h_v(\mathcal{T}(\mathbf{Q}^{z^1},\mathbf{M})) - h_v(\mathcal{T}(\mathbf{Q}^{z^2},\mathbf{M}))||_1}{||z^1 - z^2||_1}\right]. \tag{1}$$

Intuitively, this regularization loss encourages model outputs to be different (in L_1-norm), for different input noise vectors $z^1, z^2 \in \mathcal{N}(0,1)$. Note that we only apply the regularization loss on the outputs of the visual state branch, *i.e.* $h_v(\cdot)$ in the main paper. For stronger supervision, in our implementation we follow previous work [6] and generate $S = 20$ samples using S latent noise vectors, $\{z^i\}_{1:S}$, calculate the value of $\mathcal{L}_{reg}$ for all possible pairs, and select the maximum value to use in the regularization loss $\mathcal{L}_{reg}$. We validate the role of this regularization loss in Sec. 7.

3. Inference with Viterbi algorithm

To find the optimal procedure plan, we use the Viterbi algorithm [15], as discussed in Section 3.5 of the main paper. Traditionally, the Viterbi algorithm is used to find the most probable path in the first-order Markov model of a dynamic system. For a sequence of length T and N_a possible states of the system, the Viterbi algorithm relies on two main inputs: (i) a transition matrix, $A \in \mathbb{R}^{N_a \times N_a}$, capturing the probability of transitioning from one state, a_i, to another, a_j, (ii) an emission matrix, $B \in \mathbb{R}^{T \times N_a}$, describing the probability of each state, a_i, given a set of observations. In our work, we use the marginal state probabilities for each timestep (defined as $\bar{\Pi}$ in Section 3.5 of the main paper) as the emission matrix, B, and estimate the transition matrix, A, directly from the ground truth plans. Specifically, the transition matrix, A, is calculated based on the action co-occurence frequencies in the training data. To calculate the value of $A_{i,j}$, we must find the number of times action a_i is followed by action a_j in the ground truth plans. We then normalize each row of A to sum to 1, by applying L_1-normalization followed by a softmax with temperature $\tau = 1$.

4. Implementation details

Our model uses pre-extracted language and vision features, with a model trained on the HowTo100M dataset for joint text-video embedding [9]. This backbone model embeds both vision and language inputs into 512 dimensional features. For our model, we use a transformer decoder [14] with eight heads, two layers and 128 dimensional hidden states. Note that we provide an ablation on our architecture choices in Sec. 7. Since our pre-trained features are of dimension 512, we embed our initial features using a multilayer perceptron (MLP) with shape $[512 \rightarrow 256 \rightarrow 128]$ interspersed with ReLU nonlinearities. A similar MLP is used to project the ground truth language features, l_i, to the same

dimension. For the memory module, we empirically set the number of memory entries, n, to 128. For the critic, C, we use another three layer MLP with shape $[256 \rightarrow 64 \rightarrow 32]$ with ReLU activations in all layers. The dimension of the noise vector, z, is empirically set to 32. We train our model for 200 epochs with an initial learning rate set to 7×10^{-4} and decayed by 0.65 every 40 epochs. The best performing model on the validation set (*i.e*. randomly collected using 20% training data as mentioned in main paper Section 4.1) is used to report the final test set results.

5. Data curation and evaluation protocol

To train and evaluate our model, we use instructional video datasets to construct sequences with plans of various time horizons, T. Each plan contains a pair of {start, goal} visual observations, $\{v_{start}, v_{goal}\}$, and a sequence of ground truth intermediate action labels, $a_{1:T}$.

Data curation. Following previous work [3, 4], for each instructional video, we start by extracting the *ordered* list of all actions, $a_{1:N}$, present in the video. We also collect language descriptions, $l_{1:N}$, corresponding to each action, a_i. In addition, to obtain various start and goal observations, we also extract corresponding visual observations, $v_{1:N}$, by locating the start and end times of each action following previous work [3, 4]. *Note that this information is only used for data curation and not for training as done in previous work*. Each video is therefore described as the ordered set of tuples

$$V = [(a_1, l_1, v_1), \ldots, (a_N, l_N, v_N)], \tag{2}$$

where N is the total number of actions present in the video.

Given the video, V, represented with (2), we curate a set of plans with prediction time horizon T by sliding a window of size $T + 1$, such that $a_{t+1:t+T}$ and $l_{t+1:t+T}$ represent the set of action plans we need to predict, whereas v_t and v_{t+T} represent v_{start} and v_{goal}, respectively.

Evaluation protocols. As mentioned in Section 4.3 of the main paper, there exist two different evaluation protocols on CrossTask, namely "Protocol 1" and "Protocol 2". In all our experiments, we follow previous work [3, 4] and use "Protocol 1", which relies on the sliding window based data curation procedure described above. In addition, "Protocol 1", uses a 70/30 train/test dataset split as mentioned in Section 4.1 of the main paper. For the long horizon planning experiments on CrossTask (*i.e*. Section 4.3 of the main paper), we also adopt an additional protocol, proposed in recent work [12] and referred to as "Protocol 2". The main differences of "Protocol 2" can be summarized in the following three points: (i) Instead of relying on a sliding window to consider all procedure plans of time horizon T for each video, "Protocol 2" randomly selects one procedure plan of horizon T per video. (ii) "Protocol 2" uses 85/15 train/test split. (iii) "Protocol 2" predicts actions, $a_{1:T-1}$, which concretely means that, for a prediction horizon, T, it actually makes $T - 1$ predictions.

6. Baselines

Here, we provide a more detailed description of the procedure planning baselines used in our paper.

- *Random.* As an elementary baseline, we randomly select action steps from the entire vocabulary with equal probability (*i.e*. Uniform distribution) for evaluation.
- *Retrieval-Based.* For each start and goal observation pairing, $\{v_{start}, v_{goal}\}$, in the test set, this method retrieves the nearest neighbor in the train-set based on visual feature similarity. The plan labels associated with the retrieved nearest neighbor is used for evaluation.
- *WLTDO* [5] and *UAAA* [1]. These two frameworks use recurrent neural networks (RNN) [10] for action planning from the visual observation input.
- *Universal Planning Networks (UPN)* [11]. UPN is a physical-world path planning algorithm with a continuous action policy. We follow previous work [4] to modify it via appending a softmax function for the discrete action space.
- *Dual Dynamics Networks (DDN)* [4]. DDN models the state-action transition of procedural plans with a two-branch auto-regressive model. The state branch is modeled with an MLP, while the action branch uses an RNN.
- *PlaTe* [12]. This method simply replaces the models used in the two-branch model of DDN with autoregressive transformer modules.
- *Ext-GAIL* [3]. This method uses reinforcement learning techniques for procedure planning and augments the DDN architecture with a context latent variable.

7. Additional ablation experiments

In addition to the various ablations presented in the main paper, here, we provide additional experiments to evaluate other aspects of our loss, architecture and inference procedure.

7.1. Impact of the regularization loss

In this experiment we further evaluate our model's capability to model plan distributions. For this purpose we follow the evaluation procedure adopted in Section 4.4 of the main paper. The effectiveness of the diversity regularization loss, (1), for our approach is studied in Table 1. It is clear that training without the regularization leads to worse NLL and KL-div values. These results confirm the important role of the regularization in the probabilistic model, which indeed seems to suffer from mode dropping in the absence of such regularization.

Metric ↓	Method	$T=3$	$T=4$	$T=5$	$T=6$
NLL	Ours - prob. w/o $\mathcal{L}_{reg}$	5.07	6.61	7.57	8.28
	Ours - prob.	**4.89**	**5.48**	**6.24**	**7.67**
KL-Div	Ours - prob. w/o $\mathcal{L}_{reg}$	2.37	4.45	6.49	7.74
	Ours - prob.	**2.11**	**3.50**	**4.26**	**6.89**

Table 1. The effect of diversity regularization loss on the produced plans in the CrossTask dataset.

emission matrix	SR ↑	mAcc ↑	mIoU ↑
B Uniform	0.12	7.25	<0.01
A Uniform	22.61	46.12	70.24
Ours	**23.34**	**49.96**	**73.89**

Table 2. The effect of estimating transition matrix, A, and emission matrix, B, of *Viterbi* post-processing from data. The results are for prediction horizon, $T=3$, on the CrossTask dataset.

7.2. Impact of the Viterbi post-processing

In the main paper, we have shown that using Viterbi post-processing, while relying on our predicted plan distribution, $\bar{\Pi}$, as an emission matrix, leads to overall better plan predictions (*i.e*. see Table 1 of the main paper). Here, we further study the role of emission, B, vs. transition, A, matrices, in the Viterbi-based plan inference. In particular, to show the importance of using our predicted distribution, $\bar{\Pi}$, as an emission matrix, we substitute it with a uniform matrix, *i.e*. a matrix with all values equal to $\frac{1}{N_a}$, and perform plan inference with a correspondingly modified Viterbi algorithm. In other words, in these settings, the plan inference will be driven purely by the transition matrix, A. We also experiment with setting A to the uniform matrix, and only using B to drive Viterbi's inference. Table 2 shows that our original formulation gives the best results and highlights the importance of estimating the emission and transition matrices properly from data, as handled by our model.

7.3. Impact of the size of the sample set

In the main paper, we provide inference results obtained by sampling $\mathcal{K}=1500$ procedure plans from our probabilistic model. Here, we ablate this parameter and show its influence on how well we fit the ground truth plan distribution. Table 3 shows that larger value of $\mathcal{K}$ generally leads to better results as it can better approximate the true distribution. However, it should be noted that larger value of $\mathcal{K}$ require more computation. We therefore chose $\mathcal{K}=1500$ for our experiments as a reasonable trade-off between performance and computation cost.

7.4. Impact of transformer architecture

Here, we examine the impact of the transformer architecture on the plan prediction performance. To save on the computations, we limit this study to a prediction horizon of $T=3$. Table 4 shows the performance of alternative transformer configurations with different number of layers and/or heads. The model used in our paper – with two layers and eight heads – performs best on CrossTask.

Metric ↓	$\mathcal{K}$	$T=3$	$T=4$	$T=5$	$T=6$
NLL	150	5.03 ±0.21	5.85 ±0.31	6.78 ±0.25	8.73 ±0.43
	500	4.91 ±0.11	5.71 ±0.20	6.57 ±0.13	8.33 ±0.27
	1500	4.89 ±0.10	5.48 ±0.11	6.24 ±0.09	7.67 ±0.18
	2500	4.89 ±0.04	5.45 ±0.06	6.22 ±0.05	7.63 ±0.08
KL-Div	150	2.51 ±0.31	4.20 ±0.44	4.66 ±0.29	7.23 ±0.14
	500	2.48 ±0.24	3.58 ±0.12	4.51 ±0.20	6.94 ±0.10
	1500	2.11 ±0.10	3.50 ±0.06	4.26 ±0.08	6.89 ±0.03
	2500	2.01 ±0.06	3.27 ±0.02	4.18 ±0.02	6.83 ±0.01

Table 3. Ablation study of the number of samples, $\mathcal{K}$, used by our approach for probabilistic inference. We show NLL and KL-Div results with corresponding variances obtained from 20 runs.

layers/heads	SR ↑	mAcc ↑	mIoU ↑
2/1	17.81	40.67	70.85
2/4	20.54	48.32	73.05
2/8	**23.34**	**49.96**	**73.89**
3/4	19.21	45.51	72.47
3/8	14.85	39.48	69.72

Table 4. Ablation study on the number of layers and heads for prediction horizon, $T=3$, with CrossTask.

7.5. Alternative uncertainty baselines

Our approach captures the uncertainty of plans by learning a stochastic model (GAN), which allows it to produce distinct procedure plans when seeded with different random noise vectors. On the other hand, one can follow a more naive approach and obtain diverse outputs from a model trained deterministically (*i.e*. no randomness at training), by adding random noise to the system only at inference, *e.g*., injecting random noise in the input or using dropout at inference. Here, we compare our GAN-based formulation to the aforementioned naive approaches for introducing diversity in the predicted plans; results are summarized in Table 5. In particular, we evaluate adding Gaussian random noise of different intensity to the input, *i.e*. $\mathcal{N}(0,\sigma)$, and applying dropout (of probability p) to the hidden representation of the network at inference time. We see that both baselines can increase diversity at the cost of accuracy. The best naive configuration is Ours-deter+dropout ($p=0.3$) which is still inferior to our full GAN-based approach on both deterministic and probabilistic metrics. This result shows the importance of learning the uncertainty model during training, as opposed to simply adding it at inference, confirming that the GAN is useful for modelling uncertainty in our scenario.

7.6. Ablation on Video+Language features

In this study, we choose vision+language features pretrained on HowTo100M because recent work (e.g., [7, 8])

Datasets	SR ↑	mAcc. ↑	mIoU ↑	KL-Div ↓	NLL ↓	MCPrec ↑	MCRec ↑
Ours-prob. (HowTo100M pretrained)	23.34	48.96	73.89	2.11	4.89	36.61	66.13
Ours-prob. (CrossTask pretrained)	10.82	35.11	58.32	3.78	5.56	17.19	12.40
Ours-deter+noise ($\sigma = 1$)	20.02	44.76	72.96	2.37	5.77	24.74	67.21
Ours-deter+noise ($\sigma = 3$)	10.94	42.67	63.63	3.53	7.51	4.290	58.96
Ours-deter+noise ($\sigma = 5$)	5.83	35.53	55.43	5.52	8.53	1.169	37.01
Ours-deter+dropout ($p = 0.1$)	22.27	45.43	73.40	2.34	5.49	33.79	44.17
Ours-deter+dropout ($p = 0.3$)	22.04	43.64	70.97	2.46	5.83	30.61	55.32
Ours-deter+dropout ($p = 0.5$)	20.77	44.89	70.21	2.73	6.40	24.08	64.61
Ours-prob. (Strong Sup.)	24.41	45.17	73.83	2.12	4.71	36.89	62.69

Table 5. (top rows) Results using features extracted from a model pre-trained on HowTo100M vs. features finetuned on CrossTask. (middle rows) Ablation study on two alternative ways for uncertainty modeling. (last row) Strong visual supervision results. Blue text indicates the top performer and red the 2nd-best result. All results are from final evaluation on CrossTask at $T = 3$.

Metric ↓	Method	$T = 3$	$T = 4$	$T = 5$	$T = 6$
NLL	Ours - determinstic	6.37	6.38	7.71	8.95
	Ours - probabilistic	6.37	6.37	7.79	8.84
KL-Div	Ours - determinstic	5.34	6.88	6.75	7.06
	Ours - probabilistic	5.75	6.68	6.84	7.15

Table 6. Evaluation of the plan distributions produced by our probabilistic approach vs. the deterministic variant on COIN.

Metric ↓	Method	$T = 3$	$T = 4$	$T = 5$	$T = 6$
NLL	Ours - determinstic	7.18	7.79	8.49	9.17
	Ours - probabilistic	7.07	7.81	8.36	9.25
KL-Div	Ours - determinstic	5.35	5.60	5.96	8.23
	Ours - probabilistic	4.92	5.76	6.28	9.51

Table 7. Evaluation of the plan distributions produced by our probabilistic approach vs. the deterministic variant on NIV.

show state-of-the-art results with such features on the datasets we are experimenting with (*i.e.*, COIN, CrossTask). Still, it is interesting to consider using a dataset's "native" features as a variation. For this experiment, we use the CrossTask dataset, employ the video features trained on it (provided with the dataset) and train language embeddings (not provided by the dataset) by ourselves; we finetune the language model [8] to align with CrossTask's video features, using contrastive learning. These features are then used to train our model as described in Section 3 in the main paper. Table 5 (top two rows) shows that using the "native" CrossTask features is still inferior to the large-scale pre-training on HowTo100M [9]. This result could be due to the relative scarcity of data in CrossTask that is insufficient to train strong language+vision features.

7.7. Comparison to strong supervision

Our framework also can be trained with strong supervision, *i.e.* simply swapping the language supervision signals, $\{l_i\}$, in Eq. 5 of the main manuscript with visual ones, $\{v_i\}$; see Sec. 5 in this supplement. We show the results resulting from this setting in Table 5 (bottom row). Accessing the full annotations can modestly improve or hurt the performance, depending on the metric. Thus, our weakly supervised model performs comparably, while being cheaper to train.

7.8. Additional probabilistic evaluation

In the main paper, we evaluated our probabilistic model on the CrossTask dataset by measuring the Negative Log-likelihood (NLL) and KL divergence (KL-Div). Here, we further provide such results for COIN (in Table 6) and NIV (in Table 7). Interestingly, different from the results obtained on CrossTask (*i.e.* Table 6 in the main paper), we observe that there is no significant difference between our approach and the deterministic counterpart on COIN and NIV. We suspect that this happens because the NIV and COIN datasets are lacking variability in goal-conditioned plans. To verify that hypothesis, we conduct a quantitative evaluation of such variability on COIN and NIV. In Table 8, we show the average number of distinct plans that can connect the same start and goal observations for each dataset and prediction horizon. It is clear that the plans in CrossTask are much more diverse than those present in the other two datasets. In particular, CrossTask has the largest average number of distinct plans for each time horizon. Notably, for all datasets, it seems longer horizons tend to have larger variability. We believe that this is reasonable because any variations of intermediate steps would result in different paths, which highlights the importance of adopting a probabilistic point of view for procedure planning.

Datasets	$T = 3$	$T = 4$	$T = 5$	$T = 6$
CrossTask	3.26	6.76	8.40	9.29
COIN	1.51	1.93	2.25	2.35
NIV	1.03	1.07	1.28	1.29

Table 8. The average number of unique paths that share the same start and goal across multiple horizons and datasets.

7.9. Additional visualizations

In Figures 1 to 4, we provide additional visual examples of plans produced by our model for different prediction horizons. We also show failure cases and discuss potential reasons for such behaviour in corresponding captions. Note that the top two rows in each figure are successful predictions and the bottom row is the failure case. In each row of images, the first and last images denote the start and goal observations respectively, and the very next row shows the action labels predicted from our approach (*i.e*. rows beginning with "Sample"). For failure cases, we show the corresponding ground truth plan (*i.e*. rows beginning with "GT") for better understanding. In addition, we also show the (unseen) intermediate visual observations just for clarity of presentation. Our approach does not use them for training and/or testing.

8. Attribution of assets

This research was made possible thanks to the following assets.
- CrossTask dataset [19]. Our study follows the rules from the official license and uses the video URLs and annotations from this website: https://github.com/DmZhukov/CrossTask. Our usage of CrossTask is limited to this academic work.
- COIN dataset [13]. We have signed and submitted the official licence agreement from URL: https://coin-dataset.github.io/. Our study uses the provided videos/annotations and follows the rules from their license.
- NIV dataset [2]. We follow the license agreement and use videos/annotations from URL: https://github.com/jalayrac/instructionVideos.
- MIL-NCE pre-trained model [9]. We follow the license agreement and use the provided pre-trained model weights from URL: https://github.com/antoine77340/S3D_HowTo100M.

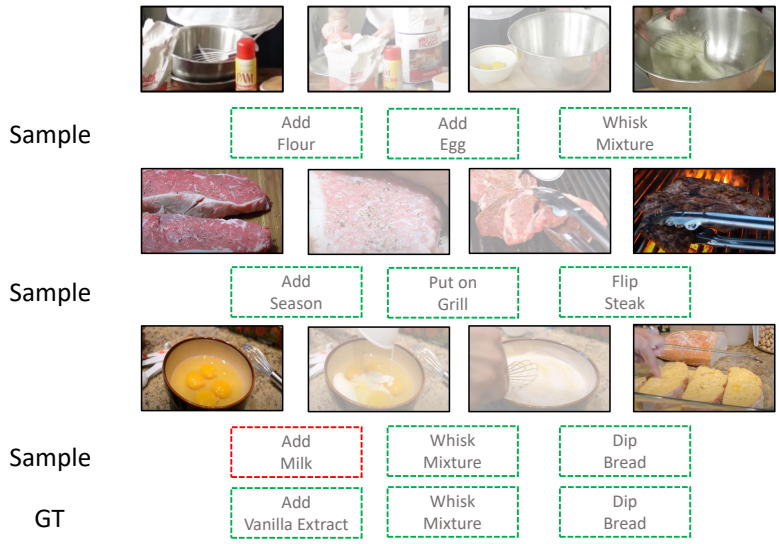

Figure 1. Visualization of two successful outputs (top two rows) and one failure output for $T = 3$ (bottom row) on CrossTask. Our approach can produce decent plans when the given start and goal observations are clear. However, we do notice some failures when the steps deviate from the usual expected actions for a certain plan, as shown in the bottom row.

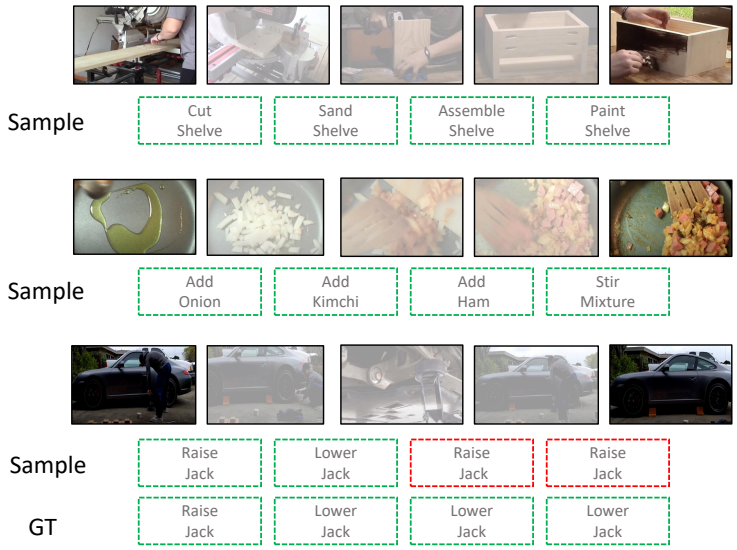

Figure 2. Visualization of two successful outputs (top two rows) and one failure output for $T = 4$ (bottom row) on CrossTask. Note that in the failure case, the start and goal observations are not clearly distinguishable. For example, in the procedure of *Jack a Car* shown in the bottom row, it is hard to reason about the transitions between the start and goal observations when their difference is so subtle.

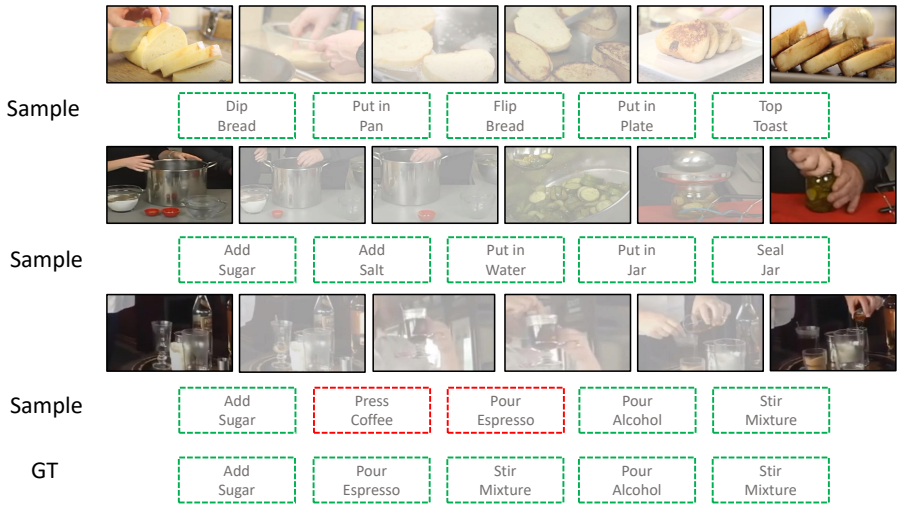

Figure 3. Visualization of two successful outputs (top two rows) and one failure output for $T = 5$ (bottom row) on CrossTask. In the failure case depicted here, we notice that our model still produces a plausible plan.

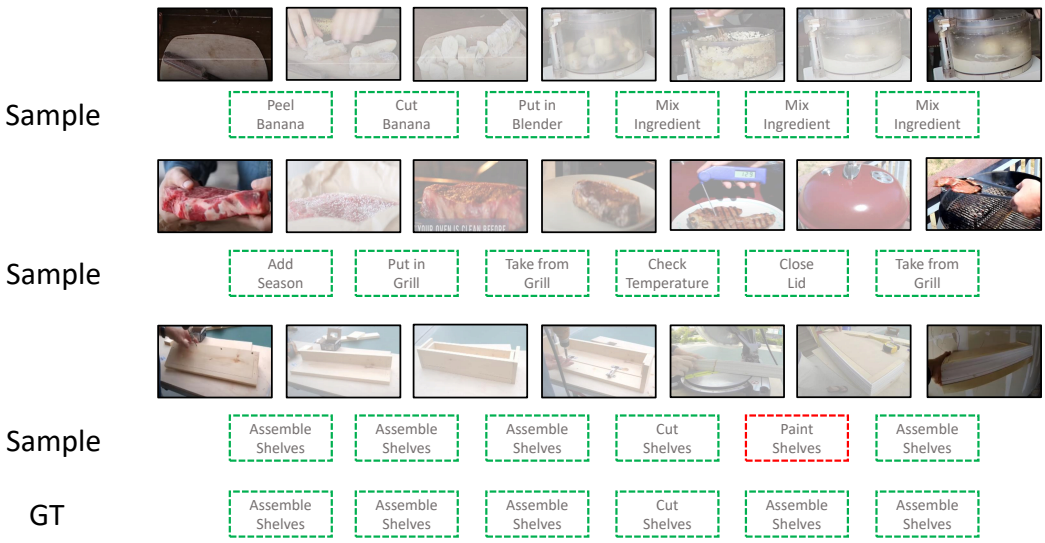

Figure 4. Visualization of two successful outputs (top two rows) and one failure output for $T = 6$ (bottom row) on CrossTask. Note that the ground truth of the failure case depicted here contains a sequence of repetitive actions. Importantly, notice that while the goal observation depicts a change in color, suggesting the action of *Paint Shelves* as predicted by our model, the ground truth seems to ignore that matter.

References

[1] Yazan Abu Farha and Juergen Gall. Uncertainty-aware anticipation of activities. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019. 2

[2] Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Josef Sivic, Ivan Laptev, and Simon Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[3] Jing Bi, Jiebo Luo, and Chenliang Xu. Procedure planning in instructional videos via contextual modeling and model-based policy learning. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 2

[4] Chien-Yi Chang, De-An Huang, Danfei Xu, Ehsan Adeli, Li Fei-Fei, and Juan Carlos Niebles. Procedure planning in instructional videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2

[5] Kiana Ehsani, Hessam Bagherinezhad, Joseph Redmon, Roozbeh Mottaghi, and Ali Farhadi. Who let the dogs out? Modeling dog behavior from visual data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[6] Shaohui Liu, Xiao Zhang, Jianqiao Wangni, and Jianbo Shi. Normalized diversification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[7] Huaishao Luo, Lei Ji, Botian Shi, Haoyang Huang, Nan Duan, Tianrui Li, Jason Li, Taroon Bharti, and Ming Zhou. UniVL: A unified video and language pre-training model for multimodal understanding and generation. *arXiv preprint arXiv:2002.06353*, 2020. 3

[8] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-end learning of visual representations from uncurated instructional videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3, 4

[9] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. HowTo100M: Learning a text-video embedding by watching hundred million narrated video clips. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019. 1, 4, 5

[10] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. 2

[11] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In *International Conference on Machine Learning (ICML)*, 2018. 2

[12] Jiankai Sun, De-An Huang, Bo Lu, Yun-Hui Liu, Bolei Zhou, and Animesh Garg. PlaTe: Visually-grounded planning with transformers in procedural tasks. *arXiv preprint arXiv:2109.04869v1*, 2021. 2

[13] Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. COIN: A large-scale dataset for comprehensive instructional video analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 5

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 1

[15] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967. 1

[16] Dingdong Yang, Seunghoon Hong, Yunseok Jang, Tianchen Zhao, and Honglak Lee. Diversity-sensitive conditional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018. 1

[17] Ye Yuan and Kris Kitani. DLow: Diversifying latent flows for diverse human motion prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 1

[18] He Zhao and Richard P Wildes. On diverse asynchronous activity anticipation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 1

[19] Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David Fouhey, Ivan Laptev, and Josef Sivic. Cross-task weakly supervised learning from instructional videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 5