PDPP: Projected Diffusion for Procedure Planning in Instructional Videos Supplementary Material

Hanlin Wang[1] Yilu Wu[1] Sheng Guo[3] Limin Wang[1, 2, ✉]
[1]State Key Laboratory for Novel Software Technology, Nanjing University, China
[2]Shanghai AI Lab, China [3]MYbank, Ant Group, China

1. Supplementary Material

Our supplementary material consists of the following: Sec. 2 provides the details of our model architecture, training process and dataset curation. Sec. 3 describes the baselines we compare with. Sec. 4 shows the details of training the task classifier in the first learning stage. We talk about another different evaluation protocol used by previous approach [13] and present the performance of our model with this protocol in Sec. 5. Then in Sec. 6, we study how batch size affects the value of mIoU metric and show the results of evaluating mIoU with different batch size on the same model. In Sec. 7, we discuss the importance of introducing prior knowledge that the start/end observations are more related to the first/last actions to our model. Finally, we provide more results, details and visualizations about the ability to model uncertainty of our approach in Sec. 8.

2. Implementation Details

2.1. Details of model architecture

The maintain of our model is the learnable model f_θ, which we implement as a basic 3-layer U-Net [10]. As in [7], each layer in our model consists of two residual blocks [6] and one downsample or upsample operation. One residual block consists of two convolutions, each followed by a group norm [15] and Mish activation function [9]. Time embedding is produced by a fully-connected layer and added to the output of first convolution. We apply a 1d-convolution along the planning horizon dimension as the downsample/upsample operation. Considering that the value of planning horizon is small ($T = \{3, 4, 5, 6\}$), we set the kernel size of 1d-convolution as 2, stride as 1, padding as 0 so the length change of planning horizon dimension keeps 1 after each downsample or upsample.

The input for our model is the concatenation of task class, actions labels and observation features, so the size of feature dimension is $dim = L_c + L_a + L_o$. Here L_c means the number of task classes in the dataset, L_a is the number of different actions in the dataset and L_o is the length of visual features. Our model embeds the input feature with shape $[dim \to 256 \to 512 \to 1024]$ in the downsample process and recover to the initial size in the upsample process as $[1024 \to 512 \to 256 \to dim]$.

For diffusion, we use the cosine noise schedule to produce the hyper-parameters $\{\beta_n\}_{n=1}^{N}$, which denote the ratio of Gaussian noise added to the data at each diffusion step.

2.2. Dataset curation details

Each video in the dataset is annotated with action labels and temporal boundaries. That is, the start time and end time of each action in an instructional video are annotated as $\{s_i, e_i\}_{i=1}^{num}$, where s_i and e_i denote the start and end time of the i_{th} action, num denotes the number of actions in the video. We in this paper follow previous work [1, 2] to extract all action sequences with predicting horizon T $\{[a_i, ..., a_{i+T-1}]\}_{i=1}^{num-T+1}$ from the given video which contains num actions by sliding a window of size T. Thus each action sequence we need to predict can be presented as $\{a_i, a_{i+1}, ..., a_{i+T-1}\}$. We choose the video clip feature at the beginning time of action a_i and clip feature around the end time of a_{i+T-1} as the start observation o_s and goal state o_g, respectively. Specifically, we first round the start and end time of this action sequence to get $\lfloor s_i \rfloor$ and $\lceil e_{i+T-1} \rceil$. Then we choose clip feature starts from $\lfloor s_i \rfloor$ and clip feature ends with $\lceil e_{i+T-1} \rceil$ as o_s, o_g, as shown in Fig. 1. Both clips are 3 seconds long.

2.3. Details of training process

We train our model with a linear warm-up scheme. For different datasets, the training scheme changes due to different scales. In CrossTask$_{Base}$, we set the diffusion step as 200 and train our model for $12,000$ steps with learning rate increasing linearly to 8×10^{-4} in the first $4,000$ steps. Then the learning rate decays to 4×10^{-4} at step $10,000$. In CrossTask$_{How}$, we keep diffusion step as 200 and train our model for $24,000$ steps with learning rate increasing linearly to 5×10^{-4} in the first $4,000$ steps and decays by 0.5 at step $10,000, 16,000$ and $22,000$. In NIV, the diffusion step is 50 and we train $6,500$ steps due to its small size. The

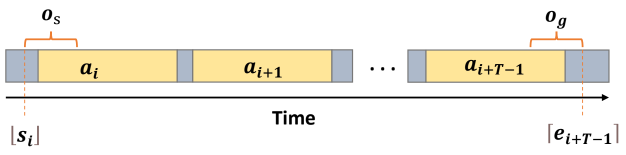

Figure 1. Selection of observations with a given action sequence.

learning rate increases linearly to 3×10^{-4} in the first $4,500$ steps and decays by 0.5 at step $6,000$. The COIN dataset requires much more training steps due to its large scale. We set diffusion step as 200 and train our model for $160,000$ steps. The learning rate increases linearly to 1×10^{-5} in the first $4,000$ steps and decays by 0.5 at step $14,000$ and step $24,000$. Then we keep learning rate as 2.5×10^{-6} for the remaining training steps. The training batch size for all experiments is 256. For the weighted loss in our training process, we set $w = 10$. All our experiments are conducted with ADAM [8] on 8 NVIDIA TITAN Xp GPUs.

3. Baselines

In this section, we introduce the baselines we used in our paper.
- $Random$. This policy randomly selects actions from the available action space in dataset to produce the plans.
- $Retrieval$-$Based$. Given the observations $\{o_s, o_g\}$, the retrieval-based method retrieves the closest neighbor by calculating the minimum visual feature distance in the train dataset. Then the action sequence associated with the retrieved result will be used as the action plan.
- $WLTDO$ [3]. This approach applies a recurrent neural network(RNN) to predict action steps with the given observation pairs.
- $UAAA$ [4]. UAAA is a two-stage approach which uses RNN-HMM model to predict action steps autoregressively.
- UPN [12]. UPN is a physical-world path planning algorithm and learns a plannable representation to make predictions. To produce the discrete action steps, we follow [2] to add a softmax layer to the output of this model.
- DDN [2]. DDN model is a two-branch autoregressive model, which learns an abstract representation of action steps and tries to predict the state-action transition in the feature space.
- $PlaTe$ [13]. PlaTe model follows DDN and uses transformer modules in two-branch to predict instead. Note that the evaluation protocol of PlaTe is different with other models, so we move the comparison with PlaTe to supplementary material, which we will discuss later.
- Ext-$GAIL$ [1]. This model solves the procedure planning problem by reinforcement learning techniques. Similar to our work, Ext-GAIL decomposes the procedure planning problem into two sub-problems. However, the purpose of the first sub-problem in Ext-GAIL is to provide long-horizon information for the second stage while our purpose is to get condition for sampling.
- P^3IV [16]. P^{3}IV is a single-branch transformer-based model which augments itself with a learnable memory bank and an extra generative adversarial framework. Like our model, P^{3}IV predicts all action steps at once during inference process.

4. Details of the first learning stage

In the first learning stage, we need to predict the task class with the given observations $\{o_s, o_g\}$. We use a simple 4-layer MLP model to achieve this and calculate the cross entropy loss for the output of the model and the ground truth task class label to train our model, except for CrossTask$_{Base}$. A two-layer Res-MLP [14] trained with MSE loss is applied to CrossTask$_{Base}$, which can get a better result when $T = 3$. The task classification results in the first learning stage for different models with different training losses on CrossTask$_{Base}$ are shown in Tab. 1.

Choices	T = 3	T=4	T=5	T=6
MLP+Crossentropy	81.94	82.61	83.14	**84.08**
ResMLP+Crossentropy	81.6	82.47	82.77	82.83
ResMLP+MSEloss	**94.38**	**83.64**	**83.37**	83.85

Table 1. Task classification results in the first stage with different training choices for CrossTask$_{Base}$.

5. Evaluation with another evaluation protocol

As talked in Sec. 3, PlaTe [13] used another protocol for evaluation. P^{3}IV [16] later evaluated SR with this protocol to compare with PlaTe. We name this as "protocol 2". In all experiments of the main paper, we follow previous work [1,2,16] to use a 70%/30% split for training/testing and rely on a sliding window to get our learning data. In this section, we evaluate our model with "protocol 2" and compare our performance with previous works that evaluated with this protocol.

The main differences of "protocol 2" are as followed: a) "protocol 2" uses a 2390/360 split for train/test. b) "protocol 2" randomly selects one procedure plan with prediction horizon T in each video for training and testing, rather than relying on a T-$size$ sliding window to consider all procedure plans in each video. c) Given the planning horizon T, "protocol 2" only predicts $T - 1$ actions.

We evaluate our model with "protocol 2" on CrossTask and compare the results with the previous best approach, considering both short and long horizons. Specifically, we use CrossTask$_{Base}$ to compare with PlaTe for shorter horizons and CrossTask$_{How}$ to compare with P^{3}IV for longer horizons. In this way we align the visual features used in

Model	T = 3	T=4	T=5	T=6
CrossTask$_{Base}$	81.71	83.63	84.65	84.53
CrossTask$_{How}$	91.78	93.31	93.50	93.75

Table 2. Task classification results with protocol2.

Models	T = 3			T = 4		
	SR↑	mAcc↑	mIoU↑	SR↑	mAcc↑	mIoU↑
PlaTe [13]	16.00	36.17	65.91	14.00	35.29	55.36
Ours$_{Base}$	**33.61**	**50.83**	49.86	**24.17**	**51.48**	54.33

Table 3. Evaluation results with protocol2 on CrossTask. Prediction horizon set to $T = \{3, 4\}$. Note that we compute IoU on every action sequence and take the mean as mIoU.

different approaches with our model to conduct a fair comparison. The task classification accuracy results for $T = \{3, 4, 5, 6\}$ with "protocol 2" are provided in Tab. 2. Tab. 3 shows the results of our model with short horizons, and Tab. 4 shows the results of SR metric with longer prediction horizons. Note that we compute mIoU by calculating the mean of every IoU for a single antion sequence rather than a mini-batch. We can see that our method keeps the top performance for all prediction horizons.

Models	T = 3	T = 4	T = 5	T = 6
	SR↑	SR↑	SR↑	SR↑
P^{3}IV [16]	24.4	15.8	11.8	8.3
Ours$_{How}$	**53.06**	**35.28**	**21.39**	**13.33**

Table 4. Evaluation results of SR with protocol 2 on CrossTask. Prediction horizon set to $T = \{3, 4, 5, 6\}$.

6. Impact of batch size on mIoU

As we discussed in the main paper, previous approaches calculate the IoU value on every mini-batch and take their mean as the final mIoU. However, the batch size value for different methods may be different, which results in an unfair comparison. In this section, we study the impact of batch size on mIoU, which can illustrate the importance for the standardization of computing mIoU.

We use our trained models to compute the mIoU metric on CrossTask with different evaluation batch size. Planning horizon is set to $\{3, 4, 5, 6\}$. The results are shown in Tab. 5, which validate our thought and show the huge impact of batch size on mIoU. The value of mIoU evaluated on the same model can vary widely as batch size changes, so comparing mIoU with different evaluation batch size has no meaning. To address this problem, we standardize the way to compute mIoU as setting evaluation batch size to 1 at inference time.

	Batch size	T = 3	T=4	T=5	T=6
Ours$_{Base}$	1	58.95	56.99	56.32	57.51
	32	68.03	67.14	67.10	**70.48**
	64	**71.46**	**69.64**	**67.39**	69.31
	128	71.01	67.26	64.53	63.19
Ours$_{How}$	1	66.57	65.13	65.32	65.38
	32	75.21	77.07	78.56	78.59
	64	79.74	81.74	**81.73**	**80.88**
	128	**80.50**	**82.32**	81.41	78.64

Table 5. Evaluation results of mIoU with different batch size on CrossTask.

7. Role of prior knowledge

In this section, we study the role of leveraging a prior knowledge that the start/end observations are more related to the first/last actions for our model.

Inspired by [5], we establish a baseline not using this prior knowledge by tiling the observations and task class conditions to the output of the U-Net encoder before decoding. 1×1 convolution is applied to reduce the channel dimension of observation features and class conditions. Then the features and conditions are replicate k times(k is the horizon length of the U-Net encoder output) and concatenated along the channel dimension. We conduct experiments on CrossTask and the results are shown in Tab. 6, which demonstrates that the introduced prior knowledge is quite useful to the learning process with visual features provided by CrossTask(Ours$_{Base}$), while not good when better visual features are provided(Ours$_{How}$).

	T=3			T=4		
	SR↑	mAcc↑	mIoU↑	SR↑	mAcc↑	mIoU↑
Tile$_{Base}$	23.40	50.37	55.14	14.13	45.80	55.03
Ours$_{Base}$	**26.47**	**55.35**	**58.59**	**15.40**	**49.42**	**56.99**
Tile$_{How}$	36.21	**65.12**	**66.79**	**22.31**	**59.00**	**66.20**
Ours$_{How}$	**37.20**	64.67	66.57	21.48	57.82	65.13

Table 6. Ablation study on the role of prior knowledge.

8. Additional study on modeling uncertainty

In the main paper, we study the probabilistic modeling ability of our model on CrossTask$_{How}$ and show that our diffusion based model can produce both diverse and accurate plans. Here we provide more details, results and visualizations about modeling the uncertainty in procedure planning by our model.

Details of evaluating uncertainty modeling. For the *Deterministic* baseline, we just sample once to get the plan since the result for *Deterministic* is certain when observations and task class conditions are given. For the *Noise* baseline and our diffusion based model, we sample 1500

	T=3		T=4	
	KL-Div↓	NLL↓	KL-Div↓	NLL↓
Deterministic	5.40	5.49	5.13	5.26
Noise	4.92	5.00	5.04	5.17
Ours	**4.85**	**4.93**	**4.62**	**4.75**

Table 7. Evaluation results of the plan distributions metrics on NIV.

	T=3			T=4		
	SR↑	Prec↑	Rec↑	SR↑	Prec↑	Rec↑
Deterministic	27.94	29.63	27.44	25.43	26.64	24.08
Noise	25.73	26.87	**38.37**	22.84	23.05	31.89
Ours	**31.25**	**31.78**	33.09	**26.72**	**29.10**	**33.08**

Table 8. Evaluation results of diversity and accuracy metrics on NIV.

action sequences as our probabilistic result to calculate the uncertain metrics. Furthermore, in order to efficiently complete the process of 1500 sampling, we apply the DDIM [11] sampling method to our model, with which one sampling process can be completed with 10 steps(accelerating the sampling for CrossTask and COIN by 20 times and NIV by 5 times). Note that the multiple sampling process is only required while evaluating probabilistic modeling and our model can generate a good plan just by sampling once.

Uncertainty modeling results on other datasets. We here provide the uncertainty modeling results on NIV(Tab. 7,Tab. 8) and COIN(Tab. 9,Tab. 10). Different with CrossTask, we here find the diffusion process still helps for NIV, but harms the performance on COIN. We suspect the reason for this is that data scales and variability in goal-conditioned plans of these datasets are different. To verify our thought, we calculate the average number of distinct plans with the same start and goal observations in these datasets, as in [16]. The results in Tab. 11 show that the variability in CrossTask is the much larger than the other two datasets. And longer horizons can bring more diverse plans for all datasets. Thus our diffusion based approach performs best on CrossTask with longer horizons T = 4, 5, 6. For NIV, we hypothesize that our model can fit this small dataset well thus adding noises to the learning does not harm the accurancy of planning much. With our diffusion based method, the model makes a good trade-off between the diversity and accuracy of planning, resulting in the best results. However, for the large COIN dataset, our model can not fit it really well and introducing noises to our model just makes the learning harder.

	T=3		T=4	
	KL-Div↓	NLL↓	KL-Div↓	NLL↓
Deterministic	**4.52**	**5.46**	**4.43**	**5.84**
Noise	4.55	5.50	4.52	5.92
Ours	4.76	5.71	4.62	6.03

Table 9. Evaluation results of the plan distributions metrics on COIN.

	T=3			T=4		
	SR↑	Prec↑	Rec↑	SR↑	Prec↑	Rec↑
Deterministic	**27.96**	**34.35**	27.40	**19.98**	**30.65**	**19.63**
Noise	18.49	25.67	**29.82**	12.58	22.25	19.32
Ours	21.33	28.03	23.49	14.41	24.83	16.28

Table 10. Evaluation results of diversity and accuracy metrics on COIN.

Datasets	T=3	T=4	T=5	T=6
CrossTask	4.02	7.82	10.92	11.51
NIV	1.31	1.50	-	-
COIN	1.74	2.52	-	-

Table 11. Average number of paths with the same start and goal states across multiple horizons and datasets.

Visualizations for uncertainty modeling. In Figures Fig. 2 to Fig. 5, we show the visualizations of different plans with the same start and goal observations proceduced by our diffusion based model CrossTask$_{How}$ for different prediction horizons. In each figure, the images denote the start and goal observations, the first row denotes the ground truth actions(rows with ”GT”), the last row denotes a failure plan(rows with ”Failure”) and the middle rows denote multiple reasonable plans produced by our model, respectively. Here the reasonable plans are plans that share the same start and end actions with the ground truth plan and exist in the test dataset.

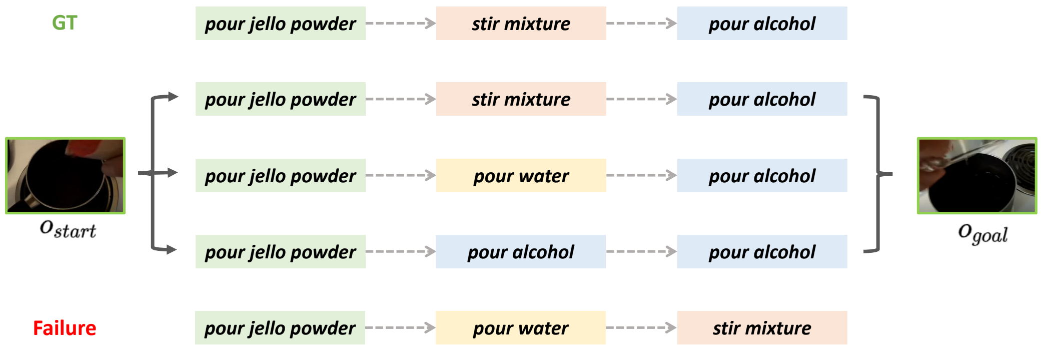

Figure 2. Visualization of diverse plans produced by our model with horizon $T = 3$.

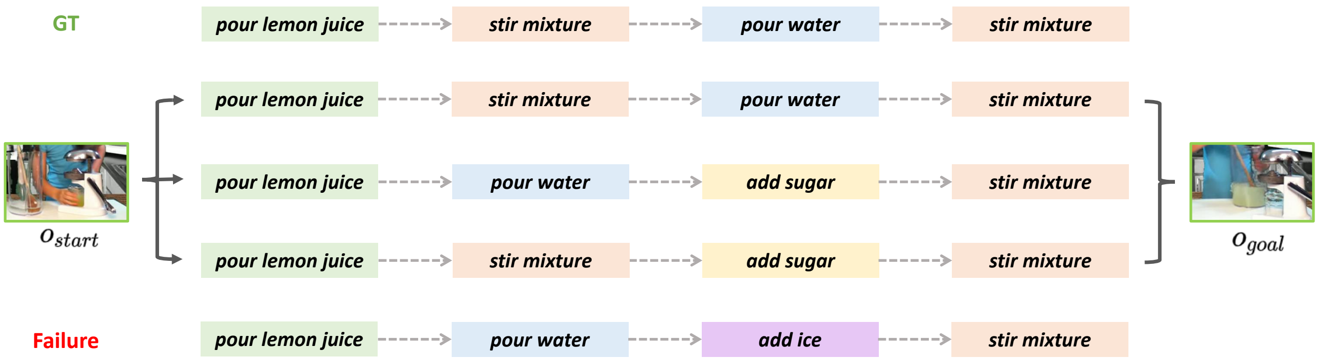

Figure 3. Visualization of diverse plans produced by our model with horizon $T = 4$.

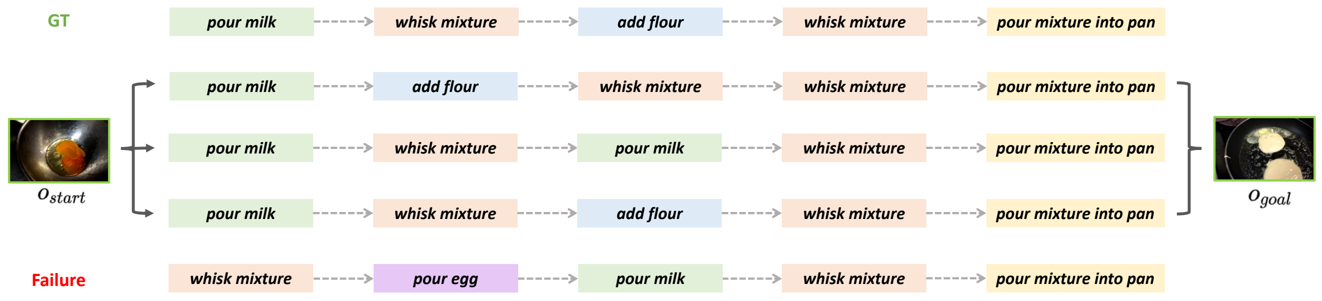

Figure 4. Visualization of diverse plans produced by our model with horizon $T = 5$.

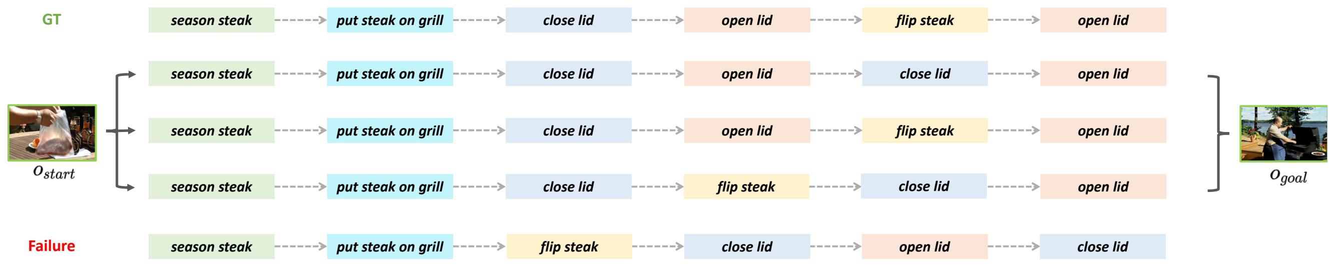

Figure 5. Visualization of diverse plans produced by our model with horizon $T = 6$.

References

[1] Jing Bi, Jiebo Luo, and Chenliang Xu. Procedure planning in instructional videos via contextual modeling and model-based policy learning. In *ICCV*, pages 15591–15600. IEEE, 2021. 1, 2

[2] Chien-Yi Chang, De-An Huang, Danfei Xu, Ehsan Adeli, Li Fei-Fei, and Juan Carlos Niebles. Procedure planning in instructional videos. In *ECCV (11)*, volume 12356 of *Lecture Notes in Computer Science*, pages 334–350. Springer, 2020. 1, 2

[3] Kiana Ehsani, Hessam Bagherinezhad, Joseph Redmon, Roozbeh Mottaghi, and Ali Farhadi. Who let the dogs out? modeling dog behavior from visual data. In *CVPR*, pages 4051–4060. Computer Vision Foundation / IEEE Computer Society, 2018. 2

[4] Yazan Abu Farha and Juergen Gall. Uncertainty-aware anticipation of activities. In *ICCV Workshops*, pages 1197–1204. IEEE, 2019. 2

[5] Ruohan Gao and Kristen Grauman. Co-separating sounds of visual objects. In *ICCV*, pages 3878–3887. IEEE, 2019. 3

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE Computer Society, 2016. 1

[7] Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 9902–9915. PMLR, 2022. 1

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015. 2

[9] Diganta Misra. Mish: A self regularized non-monotonic neural activation function. *CoRR*, abs/1908.08681, 2019. 1

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI (3)*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015. 1

[11] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *CoRR*, abs/2010.02502, 2020. 4

[12] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 4739–4748. PMLR, 2018. 2

[13] Jiankai Sun, De-An Huang, Bo Lu, Yun-Hui Liu, Bolei Zhou, and Animesh Garg. Plate: Visually-grounded planning with transformers in procedural tasks. *IEEE Robotics Autom. Lett.*, 7(2):4924–4930, 2022. 1, 2, 3

[14] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, and Hervé Jégou. Resmlp: Feedforward networks for image classification with data-efficient training. *CoRR*, abs/2105.03404, 2021. 2

[15] Yuxin Wu and Kaiming He. Group normalization. In *ECCV (13)*, volume 11217 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2018. 1

[16] He Zhao, Isma Hadji, Nikita Dvornik, Konstantinos G. Derpanis, Richard P. Wildes, and Allan D. Jepson. P^3iv: Probabilistic procedure planning from instructional videos with weak supervision. In *CVPR*, pages 2928–2938. IEEE, 2022. 2, 3, 4