PlaTe: Visually-Grounded Planning with Transformers in Procedural Tasks

Jiankai Sun, De-An Huang, Bo Lu, Yun-Hui Liu, Bolei Zhou, Animesh Garg

Abstract—In this work, we study the problem of how to leverage decision-making videos to facilitate the understanding of human decision-making processes, focusing on training a model with the ability to plan a goal-directed procedure from real-world videos. Learning structured and plannable state and action spaces directly from unstructured videos is the key technical challenge of our task. There are two problems: first, the appearance gap between the training and validation datasets could be large for unstructured videos; second, these gaps lead to decision errors that compound over the steps. We address these limitations with Planning Transformer (PlaTe), which has the advantage of circumventing the compounding prediction errors that occur with single-step models during long model-based rollouts. Our method simultaneously learns the latent state and action information of assigned tasks and the representations of the decision-making process from human demonstrations. Experiments conducted on real-world instructional videos show that our method can achieve a better performance in reaching the indicated goal than previous algorithms. We also validated the possibility of applying procedural tasks on a UR-5 platform. Please see pair.toronto.edu/plate-planner for additional details.

Index Terms—Deep Learning for Visual Perception, Task Planning, Embodied Cognitive Science

I. INTRODUCTION

In this work, we study the problem of how to leverage decision-making videos to facilitate the understanding of human decision-making processes, focusing on training a model with the ability to plan a goal-directed procedure from real-world videos. Learning structured and plannable state and action spaces directly from unstructured videos is the key technical challenge of our task. There are two problems: first, the appearance gap between the training and validation datasets could be large for unstructured videos; second, these gaps lead to decision errors that compound over the steps. We address these limitations with Planning Transformer (PlaTe), which has the advantage of circumventing the compounding prediction errors that occur with single-step models during long model-based rollouts. Our method simultaneously learns the latent state and action information of assigned tasks and the representations of the decision-making process from human demonstrations. Experiments conducted on real-world instructional videos show that our method can achieve a better performance in reaching the indicated goal than previous algorithms. We also validated the possibility of applying procedural tasks on a UR-5 platform. Please see pair.toronto.edu/plate-planner for additional details.

Procedure planning in instructional videos aims to make goal-conditioned decisions by planning a sequence of high-level actions that can bring the agent from current observation to the goal. Planning in instructional videos is a meaningful task since the ability to perform effective planning is crucial for an instruction-following agent. Although learning from instructional videos is natural to humans, it is challenging for the AI system because it requires understanding human behaviors in the videos, focusing on actions and intentions. How to learn structured and plannable state and action spaces directly from unstructured videos is the key technical challenge of our task.

It is crucial for autonomous agents to plan for complex tasks in everyday settings from visual observations. Although reinforcement learning provides a powerful and general framework for decision making and control, its application in practice is often hindered by the need for extensive feature and reward engineering. Moreover, deep RL algorithms are often sensitive to factors such as reward sparsity and magnitude, making well-performing reward functions particularly difficult to engineer. In many real-world applications, specifying a proper reward function is difficult.

In this paper, we proposed a new framework for procedure planning from visual observations. We address these limitations with a new formulation of procedure planning and
novel algorithms for modeling human behavior through a Transformer-based planning network PlaTe. Our method simultaneously learns the high-level action planning of assigned tasks and the representations of the decision-making process from human demonstrations.

We summarize our contributions as follows:

- We proposed a novel method, Planning Transformer network (PlaTe), for procedure planning in instructional videos task, which enjoys the advantage of long-term planning. We introduce some critical design choices that assist in learning cross-modal correspondences and, more importantly, improve the accuracy of generated planning sequences.

- We integrate Beam Search to PlaTe to prevent it from large search discrepancies, and reduce the performance degradation.

- Experimental results show that our framework outperforms the baselines in the procedure planning task on CrossTask, a real-world dataset. We also validated the possibility of applying procedural tasks on a real UR-5 platform.

II. RELATED WORK

Self-Attention and Transformer. Transformer-based architectures, eschew the use of recurrence in neural networks and instead trust entirely on self-attention mechanisms to draw global dependencies between inputs and outputs. Self-attention [6] is particularly suitable for procedure planning, which can be seen as a sequence modeling task. Compared with Recurrent Neural Networks (RNNs), long short-term memory (LSTM) [7] and gated recurrent neural networks [8], the advantages of self-attention includes avoiding compressing the whole past into a fixed-size hidden state, less total computational complexity per layer, and more parallelizable computations. In this paper, thanks to Transformers’ computational efficiency and scalability, we explore the possibility of marrying Transformer-based architecture for procedure planning.

Learning to Plan from Pixels. Another related work is learning dynamics models for model-based RL [9], [10], [11]. Recent works have shown that deep networks can learn to plan directly from pixel observations in domains such as table-top manipulation [12], [13], [14], navigation [15], [16], [17], and locomotion in joint space [18]. Universal Planning Networks (UPN) [13] assumes the action space to be differentiable and uses a gradient descent planner to learn representations from expert demonstrations. Prior work [3] proposes the conjugate dynamics model to expedite the latent space learning, but suffers from compounding error. While they attempt to solve the same problem, however there exist some key methodical differences. Mainly, PlaTe uses a transformer model to plan actions, which results in improved success rate, accuracy, and mIoU. Further, even with a good planner, due to the stochastic nature of predictions, planning is often low quality with a single roll-out. We address this issue with a beam search method. The adversarial training scheme proposed by Bi et al. [19] is a complex, multi-stage pipeline. In contrast, we implement an end-to-end transformer architecture, simple but efficient.

Learning from Instructional Videos. The interest has dramatically increased in recent years in understanding human behaviors by analyzing instructional videos [20], [21], [22]. Event discovery tasks such as action recognition and temporal action segmentation [23], [24], [25], [26], state understanding [27], video prediction [28] and video summarization / captioning [29], [20] study recognition of human actions in video sequences. The other works [30] perform egocentric action anticipation model the relationships between past, future events, and incomplete observations. Action label prediction [31], [32] addresses the problem of anticipating all activities within a time horizon. However, the correct answer is often not unique due to the large uncertainty in human actions.

III. OUR METHOD: PLATE

A. Problem Setup

We consider a similar setup to [3]: given the start visual observation $o_t$ and a visual goal $o_g$ that indicates the desired outcome for a particular task. During training, we have access to the observation-action pairs $\{(o_t, a_t)\} \sim \pi_E$ that were collected by an expert attempting to reach the goals. When testing, only the start visual observation $o_t$ and a visual goal $o_g$ are given. Our objective is to plan a sequence of actions $[\tilde{a}_t, \ldots, \tilde{a}_{t+T-1}]$ that can bring the underlying state of $o_t$ to that of $o_g$. $T$ is the horizon of planning, which means the number of task-level action steps the model is allowed to take. Figure 1 shows a goal-oriented plannable example where the intermediate steps of performing a complex planning task are planned.

Our key insight is that the compounding error can be reduced by jointly learning the state and action representation with Transformer-based architecture. As shown in the overall architecture in Figure 2 the procedure planning problem $p(\tilde{a}_{1:T} | o_t, o_g)$ is formulated as maximizing the probability

$$
p(\tilde{a}_{1:T} | o_t, o_g) = \text{Pr}(\tilde{s}_1 | o_t) \text{Pr}(\tilde{s}_T | o_g) \prod_{t=1}^{T} \text{Pr}(\tilde{a}_t | \tilde{s}_t, \tilde{a}_{t-1}, \tilde{s}_{t-1}) \text{Pr}(\tilde{s}_{t+1} | \tilde{s}_t, \tilde{a}_t, \tilde{s}_{t+1}).$$  \hspace{1cm}(1)

In the following sections, we first discuss how to encode the latent semantic representation. Then, we will introduce how to solve the long-term procedure planning task with transformer-based architecture. Lastly, we will discuss how to apply learned representation to solve the procedure planning by integrating Beam Search.

B. Latent Semantic Representation

First, we use the state encoder $\hat{s} = f(o)$ that encodes the visual observation to a latent semantic representation, then added with learnable positional encoding, before they were input into the transformer layers.

The remaining question is: how to learn a planning model $p(\tilde{a}_{1:T}, \tilde{s}_{1:T} | \tilde{s}_1, \tilde{s}_2)$ to plan the action sequence and corresponding latent state representation? We assume the underlying process in Figure 1 is a fully observable goal-conditioned Markov
Decision Process \((\mathcal{S}, \mathcal{A}, \mathcal{T})\), where \(\mathcal{S}\) is the state space, \(\mathcal{A}\) is the action space, \(\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}\) is the unknown transition probability distribution. We denote \(h(\hat{a}_t|\hat{s}_t, \hat{a}_{t-1}, \hat{s}_g)\) as the action prediction model conditioned on the current state, previous action, and goal, \(g(\hat{s}_{t+1}|\hat{s}_t, \hat{a}_t, \hat{s}_g)\) as the state prediction model conditioned on the previous state, goal, and previous action. They plan the sequence of actions and hidden states from the initial state to the goal state. In this way, we are able to factorize the planning model \(p(\hat{a}_{1:T}, \hat{s}_{1:T}|\hat{s}_1, \hat{s}_g)\) as:

\[
p(\hat{a}_{1:T}, \hat{s}_{1:T}|\hat{s}_1, \hat{s}_g) = \prod_{t=1}^{T} h(\hat{a}_t|\hat{s}_t, \hat{a}_{t-1}, \hat{s}_g) g(\hat{s}_{t+1}|\hat{s}_t, \hat{a}_t, \hat{s}_g),
\]

where we use the convention that \(\hat{s}_0 = 0\), \(\hat{a}_0 = 0\). The data is in the form of tuples \((s_t, a_t, s_{t+1})\). We assume to start at \(s_1\), and reach a goal \(s_g\) in \(T\) timesteps. We need to generate an action sequence \(a_{1:T}\) of length \(T\), which yields a state trajectory of length \(T + 1\).

### C. Transition Transformer

We propose a transformer-based network architecture that can learn the action-state and generate planning sequences. The overview of this architecture is shown in Figure 2. Our design choices are explained in detail below.

We introduce two cross-modal transformers: the action transformer \(h(\hat{a}_t|\hat{s}_t, \hat{a}_{t-1}, \hat{s}_g)\), which learns the correspondence between previous action feature \(\hat{a}_{t-1}\) and state feature \(\hat{s}_t, \hat{s}_g\) and generates the action prediction \(\hat{a}_t\); the state transformer \(g(\hat{s}_{t+1}|\hat{s}_t, \hat{a}_t, \hat{s}_g)\), which learns the correspondence between the state feature \(\hat{s}_t, \hat{s}_g\) and action feature \(\hat{a}_t\) and generates the future state prediction \(\hat{s}_{t+1}\). The model takes as input the whole sequence during training and all past pairs of state-action during inference, similarly to how transformer-like models usually work.

Specifically, the output of the attention layer, the context vector \(C\) is computed using the query vector \(Q\) and the key \(K\) value \(V\) pair from the input with an upper triangular look-ahead mask \(M\) via

\[
C = \text{Attn}(QK^T + M, \sqrt{D})V,
\]

where \(D\) is the number of channels in the attention layer. The look-ahead-mask \(M\) is a triangular matrix to ensure that the predictions can depend only on the known outputs before.

### D. Beam Search in Procedure Planning

Given a transformer planning model \(P_\theta\) parameterized by \(\theta\) and an input \(x\), which contains the information of current state, previous action step, and goal state, the problem of procedure planning task consists of finding an action sequence \(\hat{a}\) such that \(\hat{a} = \text{argmax}_{a \in A} P_\theta(a|x)\), where \(A\) is the set of all sequences. \(a = \{\hat{a}_1, \ldots, \hat{a}_T\}\) can be regarded as a sequence of tokens from vocabulary \(V\), where \(T\) is the length of the sequence \(\hat{a}\). Then \(P_\theta(\hat{a}|x)\) can be factored as

\[
\log P_\theta(\hat{a}|x) = \log \prod_{t=1}^{T} P_\theta(\hat{a}_t|\hat{s}_1, \hat{a}_1; \ldots; \hat{s}_t, \hat{a}_{t-1}; \hat{s}_g) \\
= \sum_{t=1}^{T} \log P_\theta(\hat{a}_t|\hat{s}_1, \hat{a}_1; \ldots; \hat{s}_t, \hat{a}_{t-1}; \hat{s}_g).
\]

The discrepancy gap is the difference in log-probability between the most likely token and the chosen token \[33, 34\]. At time step \(t\), the discrepancy gap is

\[
\max_{a \in A} \log P_\theta(\hat{a}_t|\hat{s}_1, \hat{a}_1; \ldots; \hat{s}_t, \hat{a}_{t-1}; \hat{s}_g) - \log P_\theta(\hat{a}_t|\hat{s}_1, \hat{a}_1; \ldots; \hat{s}_t, \hat{a}_{t-1}; \hat{s}_g).
\]
Algorithm 1 PlaTe: Planning Inference Phase

Input: sequence \( x \), Beginning Of Sequence \( \text{EOS} \), End Of Sequence \( \text{EOS} \), buffer \( B \), buffer size \( \mathcal{N} \), score function \( \text{score}(\cdot, \cdot) \), maximum sequence length \( n_{\text{max}} \), maximum beam size \( k \), planning model \( p(\hat{s}_t, \hat{d}_t|\hat{s}_{t-1}, \hat{d}_{t-1}, \hat{s}_T) \)

Output: searched sequence \( B_{\text{max}}(\cdot) \)

\[
B_0 \leftarrow \{<\text{BOS}>, \ldots \}
\]

for \( t \in \{1, \ldots, n_{\text{max}} - 1\} \) do

\[
B \leftarrow \emptyset
\]

for \( w, (\hat{s}_{t-1}, \hat{d}_{t-1}, \hat{s}_T) \in B_{t-1} \) do

if \( \hat{s}_{t-1, \text{last}}() = \text{EOS} \) then

\[
B, \text{add}(<w, (\hat{s}_{t-1}, \hat{d}_{t-1}, \hat{s}_T)>)
\]

continue

end if

\[
(\hat{s}_t', \hat{d}_t') \leftarrow p(\hat{s}_{t-1}', \hat{d}_{t-1}', \hat{s}_T) \]

\[
w \leftarrow \text{score}(\hat{s}_t', \hat{d}_t', \hat{s}_T)
\]

\[
B, \text{add}(<w, (\hat{s}_t', \hat{d}_t', \hat{s}_T)>)
\]

end for

\[
B_t \leftarrow B, \text{top}(k)
\]

end for

return \( B_{\text{max}}(\cdot) \)

To avoid long-term procedure planning from significant search discrepancies, we introduce the discrepancy-constrained Beam Search during the inference phase of procedure planning. The action log-probability output by action prediction model \( h(\cdot) \) is used as the score function. The inference algorithm is shown as Algorithm 1. In this way, we can reduce the performance degradation.

E. Learning

As shown in Figure 3, we have three main components to optimize: state encoder \( f \), action prediction model \( h \), and state prediction model \( g \). We refer to the expert trajectory as \( \tau^E = \{(s_t^E, a_t^E)\} \) and predicted trajectory \( \tau = \{(s_t, a_t)\} \) as state-action pairs visited by the current planning model.

We optimize by descending the gradient in Equation 6

\[
\min_{\theta} \sum_{t=1}^{T} ||s_t - s_t^E||_2 + \text{CE}(a_t, a_t^E)
\]

where CE is the cross-entropy loss. In training, \( T \)-step sequence is output once. In testing, single-step inference is made with Beam Search.

IV. EXPERIMENTS

In our experiments, we aim to answer the following questions: (1) Is PlaTe efficient and scalable to procedure planning tasks? (2) Can PlaTe learn to plan on the interactive environment? To answer Question 1, we evaluate PlaTe on CrossTask, a real-world offline instructional video dataset. We show procedure planning with our algorithm performs better on the CrossTask dataset than previous methods. To answer Question 2, we evaluate our method on a real-world UR-5 robot arm platform.

A. Experimental Setup

Datasets. We first evaluate PlaTe on an instructional video dataset CrossTask [21]. For real-world UR-5 experiments, we collect a UR-5 Reaching Dataset which consists of 100 trajectories (2150 first-person-view RGB image and corresponding action pairs) as a training set and evaluate on a real UR-5 platform.

Implementation Details. We compare with the baselines in [3] with the metrics such as success rate, accuracy, and mIoU. We use the Transformer architecture [5] as the transition model with 8 self-attention layers and 8 heads. The transition model is two-headed: one for action prediction the other for state prediction. Let \( L_s \) be the hidden dimension for state and \( L_a \) be the embedding dimension for action. The state encoder in our model is two fully-connected layers with [64, \( L_a \)] units in each layer and Leaky-ReLU as non-linearity function. We encode \((o_t, a_t)\) to be \( L_a \)-dim \( s_t, a_t \) is encoded (in transformer terminology: "tokenized") to be a \( L_a \)-dim embedding and concatenated together with \( s_t \) to be provided as input to the transformer. The size of the input of the transformer is \( L_s + L_a \) and the size of the output of the transformer is also \( L_s + L_a \). \( s_t+1 \) is the first \( L_a \)-dim features split from transformer output. \( a_t+1 \) is decoded from the rest of transformer output. In our experiments, \( L_s = 32 \) and \( L_a \) is the total number of possible actions. For CrossTask experiments, \( L_a = 105 \). One-hot vectors are used for this action classification purpose. During training, all models are optimized by Adam [35] with the starting learning rate of \( 10^{-4} \). We train our model for 200 epochs on a single GTX 1080 Ti GPU.

B. Evaluating Procedure Planning on CrossTask

First, we choose a real-world instructional video dataset CrossTask [21] to conduct our experiments. CrossTask comprises 2,750 videos (212 hours). Each video depicts one of the 18 primary long-horizon tasks. To test the trained agent’s generalization capability, for the videos in each task, we randomly divide the videos in each task into 70%/30% splits for training and testing. Each video can be regarded as a sequence of images \( V_i \) (where \( i \) is the index of frames) that have annotated with a sequence of action labels \( a_j \) and each action starts at frame index \( s_j \) and ends at frame index \( e_j \). Same as the setup of [3]: we choose frames around the beginning of the captions \( V_i \cdot \delta/2 \leq s_j \leq \delta/2 \) as \( a_i \), caption description \( a_i \) as the semantic meaning of action, and images nearby the end \( V_i \cdot \delta/2 \leq e_j \leq \delta/2 \) as the next observation \( o_t+1 \). Here, \( \delta \) controls the duration of each observation, and we set \( \delta = 2 \) for all data we have used in our paper. Our state-space \( \mathcal{S} \) is the pre-computed features provided in CrossTask: each second of the video is encoded into a 3,200-dimensional feature vector, which is a concatenation of the I3D [36], Resnet-152 [37], and audio VGG features [38]. The action space \( \mathcal{A} \) is constructed by enumerating all combinations of predicates and objects, which provides 105 action labels and is shared across all 18 tasks. Our method is suitable for modeling longer trajectories, but we restrict the experiments to horizontal lengths \( T = 3 \sim 4 \) to maintain a consistent comparison with state-of-the-art methods.
Recall that in procedure planning, given the start and goal observations \( o_1 \) and \( o_g \), the agent needs to output a valid procedure \( \{a_1, \ldots, a_T\} \) to reach the specified goal. As illustrated in Table I, as instructional videos’ action space is not continuous, the gradient-based planner of UPN cannot work well. By introducing Beam Search, our PlaTe has a better performance in terms of success rate, accuracy, and mIoU. By designing a model with transformer-based components, we show that our model outperforms all the baseline approaches on real-world videos.

In Figure 3, we visualize some examples of the predicted procedure planning results on CrossTask, where the task is to Make Pancake. Our model is able to predict a sequence of actions with correct ordering. Specifically, the most challenging step in Make Pancake is the “add flour” and “add sugar” step, where visual differences are not significant, and it can only be inferred from context and sequence relationships.

C. Evaluating Procedure Planning on Real Robot

Previous Procedure Planning research has rarely reported experimental results in real robots. There remains a gap between offline training and real-world applications. To validate the possibility of applying procedural tasks in the real environment, we conduct experiments on "Reaching a block” (cf. Figure 4) using a Universal Robot UR5 system. While this task is easy in simulation, it can be difficult for real robot [39]. Our agent learns to achieve tasks by imitating expert demonstrations. After the RGB-D camera is installed in front of the manipulator, the observation space includes the raw RGB images at the current position and goal position. The action space includes None, Up, Down, Left, Right, Forward, Backward. UR5 Reacher consists of episodes of interactions, where each episode is \( T \) time steps long. The fingertip of UR5 Reacher is confined within a 3-dimensional 0.7m x 0.5m x 0.4m boundary. The robot is also constrained within a joint-angular boundary to avoid self-collision.

In the task of Reaching, the robot is required to start at the current observation and then move to a goal observation. Since the controller of the robot is imperfect, we consider a reach to be successful if the robot reaches within 5cm of the block. The UR5 robotic arm is controlled by human volunteers to reach the target, and thus 100 offline expert demonstrations

| Method  | UR5 Prediction Length | \( T = 3 \) | \( T = 4 \) |
|---------|----------------------|-------------|-------------|
| Random  | <0.01               | <0.01       | <0.01       |
| RB [29] | 32                   | 26          |             |
| RL [3]  | 50                   | 44          |             |
| WLTDO [18] | 42                | 38          |             |
| UAAA [32] | 44                 | 40          |             |
| UPN [13] | 44                   | 38          |             |
| DDN [3]  | 52                   | 46          |             |
| PlaTe (Ours) | 60              | 52          |             |
trajecories are generated. All methods are evaluated on a real UR5 robotic arm for 50 episodes. As shown in Table III, our approach outperforms the other methods. Even though other baselines have succeeded in the offline dataset, real robotic reaching lags far behind human performance and remains unsolved in the field of robot learning.

V. CONCLUSION

To conclude, we propose a cross-modal transformer-based architecture to address the procedure planning problem, which can capture long-term time dependencies. Moreover, We propose to enhance the transformer-based planner with Beam Search. Finally, we evaluate our method on a real-world instructional video dataset. The results indicate that our method can learn a meaningful action sequence for planning and recover the human decision-making process. We also validated the possibility of applying procedural tasks on a real UR-5 platform.

ACKNOWLEDGMENT

Animesh Garg is supported in part by CIFAR AI Chair at the Vector Institute for AI and NSERC Discovery Award.

REFERENCES

[1] M. Beetz, R. Chatila, J. Hertzberg, and F. Pecora, “Ai reasoning methods for robotics,” in Springer Handbook of Robotics, 2016.

[2] J. Sun, H. Sun, T. Han, and B. Zhou, “Neuro-symbolic program search for autonomous driving decision module design,” in Proceedings of the 2020 Conference on Robot Learning, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 21–30. [Online]. Available: https://proceedings.mlr.press/v155/sun21a.html

[3] C.-Y. Chang, D.-A. Huang, D. Xu, E. Adeli, L. Fei-Fei, and J. C. Niebles, “Procedure planning in instructional videos,” in ECCV, Springer, 2020, pp. 334–350.

[4] J. Sun, L. Yu, P. Dong, B. Lu, and B. Zhou, “Adversarial inverse reinforcement learning with self-attention dynamics model,” IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 1880–1886, 2021.

[5] J. Huang, S. Xie, J. Sun, Q. Ma, C. Liu, D. Lin, and B. Zhou, “Learning a decision module by imitating driver’s control behaviors,” in Proceedings of the Conference on Robot Learning (CoRL) 2020.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, “Attention is all you need,” in NeurIPS, vol. 30. Curran Associates, Inc., 2017.

[7] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., 1997.

[8] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequencing model,” arXiv:1412.3555, 2014.

[9] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, “Learning latent dynamics for planning from pixels,” in ICML, 2019.

[10] K. Fang, Y. Zhu, A. Garg, S. Savarese, and L. Fei-Fei, “Dynamics learning with cascaded variational inference for multi-step manipulation,” arXiv:1910.13395, 2019.

[11] B. Amos, S. Stanton, D. Yarats, and A. G. Wilson, “On the model-based stochastic value gradient for continuous reinforcement learning learning,” in IJAC, 2021, pp. 6–20.

[12] T. Kurutach, A. Tamar, G. Yang, S. J. Russell, and P. Abbeel, “Learning plannable representations with causal infogap,” in NeurIPS, vol. 31. Curran Associates, Inc., 2018, pp. 8733–8744.

[13] A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and C. Finn, “Universal planning networks: Learning generalizable representations for visuomotor control,” in ICML, 2018.

[14] X. Chen, Z. Ye, J. Sun, Y. Fan, F. Hu, C. Wang, and C. Lu, “Transferable active grasping and real embodied dataset,” in 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 3611–3618.

[15] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, “Curiosity-driven exploration by self-supervised prediction,” in ICML, 2017.

[16] Qiu, L. Chen, X. Gu, F. P.-W. Lo, Y.-Y. Tsai, J. Sun, J. Liu, and B. Lo, “Ego-centric human trajectory forecasting with a wearable camera and multi-modal fusion,” arXiv:2111.00993, 2021.

[17] B. Pan, J. Sun, H. Y. T. Leung, A. Andonian, and B. Zhou, “Cross-view semantic segmentation for sensing surroundings,” IEEE Robotics and Automation Letters, vol. 5, no. 3, pp. 4867–4873, 2020.

[18] K. Ehsani, H. Bagherinezhad, J. Redmon, R. Mottaghi, and A. Farhadi, “Who let the dogs out? modeling dog behavior from visual data,” in CVPR, 2018, pp. 4051–4060.

[19] J. Bi, J. Luo, and C. Xu, “Procedure planning in instructional videos via contextual modeling and model-based policy learning,” in ICCV, 2021, pp. 15 611–15 620.

[20] L. Zhou, C. Xu, and J. Corso, “Towards automatic learning of procedures from web instructional videos,” in AAAI, 2018.

[21] D. Zhukov, J.-B. Alayrac, R. C. Cinbis, D. Fouhey, I. Laptev, and J. Siciv, “Cross-task weakly supervised learning from instructional videos,” in CVPR, 2019, pp. 3537–3545.

[22] B. Pan, J. Sun, W. Lin, L. Wang, and W. Lin, “Cross-stream selective networks for action recognition,” in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2019, pp. 454–460.

[23] D.-A. Huang, L. Fei-Fei, and J. C. Niebles, “Connectionist temporal modeling for weakly supervised action labeling,” in ECCV, 2016.

[24] V. Blakis, C. Paxton, D. Fox, A. Garg, and Y. Artzi, “A persistent spatial semantic representation for high-level natural language instruction execution,” in CoRL, 2021.

[25] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese, “Neural task programming: Learning to generalize across hierarchical tasks,” in ICRA. IEEE, 2018, pp. 3795–3802.

[26] D.-A. Huang, S. Nair, D. Xu, Y. Zhu, A. Garg, L. Fei-Fei, S. Savarese, and J. C. Niebles, “Neural task graphs: Generalizing to unseen tasks from a single video demonstration,” in CVPR, 2019, pp. 8565–8574.

[27] J.-B. Alayrac, I. Laptev, J. Siciv, and S. Lacoste-Julien, “Joint discovery of object states and manipulation actions,” in ICCV, 2017.

[28] W. Yu, W. Chen, S. Yin, S. Easterbrook, and A. Garg, “Modular action concept grounding in semantic video prediction,” 2020.

[29] C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid, “Videobert: A joint model for video and language representation learning,” in ICCV, 2019, pp. 7464–7473.

[30] L. Zhou, C. Xu, and J. Corso, “Towards automatic learning of procedures from web instructional videos,” in AAAI, 2018.

[31] F. Sener and A. Yao, “Zero-shot anticipation for instructional activities,” in Proceedings of the 2020 Conference on Robot Learning, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 21–30. [Online]. Available: https://proceedings.mlr.press/v155/sun21a.html

Fig. 4: Qualitative results of procedure planning on UR5.
[35] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 
ICLR, 2015.
[36] J. Carreira and A. Zisserman, “Quo vadis, action recognition? a new 
model and the kinetics dataset,” in CVPR, 2017.
[37] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image 
recognition,” in CVPR, 2016.
[38] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, 
R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, 
R. J. Weiss, and K. Wilson, “Cnn architectures for large-scale audio 
classification,” in ICASSP, 2017.
[39] S. Gu, E. Holly, T. Lillicrap, and S. Levine, “Deep reinforcement 
learning for robotic manipulation with asynchronous off-policy updates,” 
in IEEE ICRA, 2017.