2024 Tenth Indian Control Conference (ICC)
9 - 11 December, 2024
IISER Bhopal, India

Complete Task Learning from a Single Human Demonstration by Identifying Fundamental Sub-tasks for Robot Manipulator Execution

Anurag Maurya, Dharmendra Sharma, Archit Sharma, Peeyush Thakur, Sandeep Gupta, Narendra Kumar Dhar, and Laxmidhar Behera

Abstract— **This work tackles the problem of teaching robot manipulators to carry out intricate tasks independently by observing human demonstrations. By breaking down the whole task into smaller basic sub-tasks, a reliable model is developed by associating human actions with equivalent robotic ones more easily so that manipulator actions can be programmed efficiently. To enable real-time sub-task level identification, vision transformer (ViViT), TimeSformer, and VideoMAE models are used separately as encoder architecture and trained using video data to anticipate the sub-task levels. These models are compared, and the accuracy aggregated on different tasks is** 64.36% **(ViViT),** 71.26% **(TimeSformer) and** 81.03% **(VideoMAE). The identified sub-tasks are executed by a robot manipulator using the trajectories learned through the dynamic movement primitives (DMP). Real-time experiments show that this approach greatly enhances the robot's ability to reliably and precisely reproduce complex tasks. The proposed solution also emphasizes how flexible the system is to various task modifications and how it can be used in multiple healthcare, home, and industrial settings. The novelty lies in the notion of finding semantic connections between primitive sequences from video data.**

Index Terms— **Task learning, imitation learning, classification, video, robot manipulator.**

I. INTRODUCTION

Video classification involves categorizing them into predefined classes based on their content. Traditional methods rely heavily on handcrafted features and shallow learning algorithms that do not often capture the complex patterns inherent in video data [1].

Robotic manipulators have the potential to revolutionize a wide range of industries by performing complex tasks with high precision and efficiency [2]. However, programming these robots to execute intricate tasks remains a significant challenge. Traditional methods often rely on pre-defined instructions, which can be inflexible and require extensive programming knowledge. This limitation has driven the need for more intuitive approaches that can harness the natural proficiency of humans in task execution [3]. Human tasks often involve a sequence of nuanced actions that should be accurately captured, interpreted, and translated into robotic maneuvers. Achieving this requires sophisticated techniques capable of real-time processing and adaptation [4].

In this research, we propose a novel approach that utilizes a vision transformer model to address the challenge of learning fundamental sub-tasks from human demonstrations. Vision transformers, known for their robust performance in video analysis, are trained on video data capturing human task executions. Our approach involves capturing high-resolution video data of human demonstrations and training the vision transformer model to identify and segment these demonstrations into discrete sub-tasks.

The **practical motivation** for this paper is to address several key challenges in learning from human demonstrations for robot manipulator execution: i) Precision and Accuracy: Achieving high precision and accuracy in capturing intricate human movements is a primary challenge. ii) Variability and Context Sensitivity: Addressing the variability among individuals and the context sensitivity of human demonstrations is crucial for reliable task execution. iii) Real-time Processing: Developing efficient real-time processing capabilities is essential to ensure quick and accurate interpretation of human actions, especially in dynamic environments. The **contributions** of this paper align with the above motivation. They are: 1) A vision transformer encoder is utilized for learning sub-task levels from human demonstrations for capturing intricate movements, 2) real-time capabilities are integrated to enable efficient and accurate interpretation of human actions. The organization of the rest of the paper is as follows: Section II presents a literature review and problem formulation. Section III contains the proposed methodology, Section IV presents results and relevant discussions, and Section V provides the conclusion.

II. LITERATURE REVIEW AND SOLUTION FORMULATION

A. Existing works

Classifying subtasks in full-task videos can be quite challenging because they involve different factors, such as understanding spatial locations within the frame and temporal locations across the frame, and understanding the object movements in the frames. Numerous contributions from early research [5], [6], [7] to the works presented in [8], [9], [10] and the references therein highlight the important advancements in vision-domain video classification and ongoing studies in human-robot interaction. Feichtenhofer *et. al.* [11] introduced X3D, a family of efficient video models that expands a 2D network into 3D in a progressive manner. This methodology boasts high efficiency with fewer parameters and lower computational cost, showing strong performance on video classification tasks. However, it still requires substantial computational resources and has complexity in model architecture design.

Bertasius *et. al.* [12] applied transformer models popular in NLP to video classification by dividing attention into temporal and spatial components in their paper. This approach captures long-range dependencies and offers a simpler architecture than the 3D CNNs. The drawbacks include

979-8-3315-1721-2/24/$31.00 ©2024 IEEE

the requirement of large datasets and extensive computational resources. The approach in [13] applies self-supervised learning to video classification by masking parts of the video and predicting the missing content using transformer models. Tong *et. al.* in [14] proposed VideoMAE leveraging masked frame prediction to improve data efficiency in video understanding. This framework reduces the need for large labeled datasets and achieves competitive performance with less data.

B. Solution formulation

Our approach decouples the entire execution in two stages. The first stage reads an input video, captures the semantic meaning of action and outputs a sequence of primitives. These primitives are passed into a FIFO data structure to execute them sequentially. The second stage accepts these primitives and utilizes a DMP model to execute the trajectories in real-time.

The subtasks considered in this work are **Reach, Tilt, Retract, Grasp, Release, Rotate, Stir, Wipe,** and **Press**. We define classes for the given subtasks and an extra class to provide padding. Some of the labels are not used in current data but we defined them for sake of completion and future works.

III. Proposed Methodology

Breaking down a large robotic task into smaller subtasks involves segmenting it into discrete steps, each crucial for the overall goal. While CNNs struggle with long-range dependencies in video sequences, transformers [15] excel at this. We propose using an attention-based approach for video encodings, followed by a decoder to generate probability scores for the defined primitives.

We have performed experiments using three different attention-based architectures (TimeSformer [12], ViViT [16], and VideoMAE [14]) for encoding the video and used a simple multi-head-multi-class MLP to decode the hidden states into probability scores over subtask class distribution.

A. Architecture

Our approach uses the videoMAE architecture which utilizes the masked auto-encoding approach, a technique popular in deep learning for unsupervised and semi-supervised learning. Since the demonstrations have a lot of redundancy in consecutive frames in terms of information carried, videoMAE masks some portion of the input data and passes it to a Transformer auto-encoder. The autoencoder thus learns more reliable features through reconstruction. These efficient features can then be utilized to identify subtasks present in the video.

High temporal redundancy can lower the importance of motion information. VideoMAE tackles video-masked modeling inefficiencies with a custom design. Let V be the original video clip with T consecutive frames, each containing $H \times W \times 3$ pixels (height, width, channels). Let Ω denote the set of all possible spatial locations within a frame ($\Omega = \{(i,j) \mid 1 \le i \le H, 1 \le j \le W\}$). Let ρ_{mask} be the masking ratio ($0 \le \rho_{\text{mask}} \le 1$).

It takes downsampled video frames and utilizes cube embedding to create video tokens. Cube embedding can be interpreted as a function f_{cube} that takes a cube of size $C \times H' \times W'$ (temporal dimension C, spatial dimensions H', W') and embeds it into a D-dimensional vector

$$f_{\text{cube}} : (C, H', W', 3) \to \mathbb{R}^D.$$

Now creates $T' = \left\lfloor \frac{T}{C} \right\rfloor$ non-overlapping cubes by dividing the video sequence V into chunks of size C frames.

To reduce redundancy and promote semantic reasoning, a high masking ratio ($90-95\%$) is applied along with temporal tube masking where masked regions span entire frames. The mask variable M is a 3D binary tensor with the same dimensions as a cube (C, H', W'):

$$M \in \{0,1\}^{C \times H' \times W'}.$$

Each element $M[c,i,j]$ indicates masking(Mask/unmask) by 1 or 0, respectively. This involves sampling each element of M independently according to a Bernoulli distribution with parameter ρ_{mask}:

$$M[c,i,j] \sim \text{Bernoulli}(\rho_{\text{mask}}).$$

The backbone uses the vanilla Vision Transformer (ViT) in Figure 1 with **joint space-time attention** to capture spatiotemporal information in the remaining unmasked tokens.

The vanilla ViT consists of the following components:

*1) **Patch embedding**:* An image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ is reshaped into a sequence of flattened 2D patches $\mathbf{x}_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$

$$\mathbf{x}_p = \text{reshape}(\mathbf{x}), \tag{1}$$

where (H, W) is the resolution of the original image, C is the number of channels, (P, P) is the resolution of each image patch, and $N = \frac{HW}{P^2}$ is the resulting number of patches.

Each patch is then embedded into a vector of dimension D using a trainable linear projection

$$\mathbf{E} = \mathbf{x}_p \mathbf{W}_e. \tag{2}$$

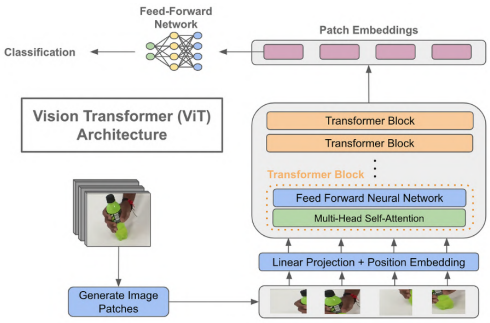


Fig. 1: Block diagram of ViT for pick and place task.

*2) **Position embedding**:* To retain positional information, learnable position embeddings are added to the patch embeddings as

$$\mathbf{z}_0 = [\mathbf{E}_1; \mathbf{E}_2; \ldots; \mathbf{E}_N] + \mathbf{E}_{\text{pos}}. \tag{3}$$

*3) **Multi-head self-attention (MSA):*** The self-attention mechanism computes attention scores based on the queries (**Q**), keys (**K**), and values (**V**) derived from the input **z** as

$$\text{attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}. \tag{4}$$

For multi-head attention, the input **z** is linearly projected h times with different learned projections to form h sets of queries, keys, and values given by

$$\text{MSA}(\mathbf{z}) = [\text{head}_1;\text{head}_2;\ldots;\text{head}_h]\mathbf{W}_o, \tag{5}$$

where MSA is the concatenation of individual head outputs where each head is computed as

$$\text{head}_i = \text{Attention}(\mathbf{Q}_i,\mathbf{K}_i,\mathbf{V}_i). \tag{6}$$

*4) **Feed-forward network (FFN):*** The FFN is applied to each token consisting of two linear transforms with a ReLU activation

$$\text{FFN}(x) = \max(0, x\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2, \tag{7}$$

where b_1 and b_2 are respective biases.

*5) **Transformer encoder:*** The Transformer encoder consists of L layers each with layer normalization, multi-head self-attention, a residual connection, and a FFN as follows:

$$\begin{aligned}\mathbf{z}'_l &= \text{MSA}(\text{LayerNorm}(\mathbf{z}_{l-1})) + \mathbf{z}_{l-1} \text{ and}\\ \mathbf{z}_l &= \text{FFN}(\text{LayerNorm}(\mathbf{z}'_l)) + \mathbf{z}'_l,\end{aligned} \tag{8}$$

where $\mathbf{z}_l$ is the output from l-th encoder layer.

*6) **Classification head:*** After the Transformer encoder, a classification head is applied to the flattened token representations of all patches. We use the videoMAE encoder backbone to obtain hidden states, which are passed through an MLP layer and 10 heads with softmax activations.

$$\mathbf{y}^i = \text{softmax}(\mathbf{z}_{[\text{CLS}]}\mathbf{W}^i_{\text{cls}} + \mathbf{b}^i_{\text{cls}}). \tag{9}$$

Each head is responsible for predicting a sub-action. The

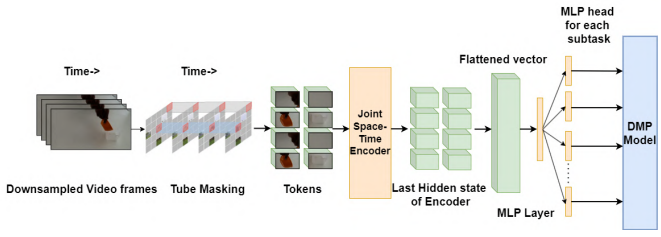


Fig. 2: Pipeline for video classification and primitive execution.

maximum number of sub-actions that can be predicted is 10 (10 is chosen for experimental purposes, any arbitrary number of heads can be chosen). During the forward pass, we take probability distribution from softmax and return the index of the element with the highest value. The output from the decoder is an array of size 10 where each element represents the sub-action perceived. The array is then passed through a look-up table to get the sub-action being performed. Once we receive the sub-task labels from the model as shown in Table I, we utilize a library of predefined trajectories generated by the DMP model for executing these sub-actions1. Figure 2 illustrates the pipeline used.

Labelling scheme for individual tasks	
Task	**Label**
Pick and Place	Reach–Grasp–Reach–Release–Retract
Pick and Pour	Reach–Grasp–Reach–Tilt–Reach–Release–Retract
Mix	Reach–Grasp–Reach–Stir–Reach–Release–Retract
mopping	Reach–Grasp–Wipe–Release–Retract

TABLE I: Sub-action labels for different tasks.

B. Learning sub-tasks trajectories using DMP

The trajectory evolution for each sub-task can be learned through demonstration or manual programming to produce human-like movements [17][18]. It is described using a set of equations with specific hyperparameters, considering the position, velocity, and acceleration data for each joint of the manipulator. This dataset trains a DMP model that simplifies the movements into basic expressions, facilitating effective trajectory generation for the manipulator. It is a mathematical framework for generating smooth movement patterns for both periodic and discrete (point-to-point) movements [19].

The DMP equation for each manipulator's joint to describe the trajectory is [20]

$$\tau\ddot{x} = \alpha(\beta(g-x) - \tau\dot{x}) + f(x), \tag{10}$$

where τ represents the temporal scaling factor for movement duration; x is position, $\dot{x}$ is velocity and $\ddot{x}$ is the acceleration of the system; g is the goal or target position, and α is the gain parameter for convergence speed towards the goal. A larger α value results in faster convergence toward the target position, while a smaller α value leads to slower convergence. β controls the manipulator's spring-like behaviour for controlling the trajectory's shape, and $f(x)$ is the forcing function. The following steps are important for learning the trajectory evolution.

1) Forcing function: The forcing function shapes the manipulator's trajectory based on the collected data, ensuring precise and accurate movement for optimal performance. It is a guiding control signal, directing the system's motion to adhere to specific trajectory, speed, and orientation requirements. Mathematically, it is expressed as:

$$f(s) = \frac{\sum_{i=1}^{N} w_i \psi_i(s)}{\sum_{i=1}^{N} \psi_i(s)}, \tag{11}$$

where w_i represents the weight of basis function ψ_i.

2) Basis function: The basis function in DMP enables adaptability to various types of movement, enhancing flexibility and generalization. The function should be smooth to avoid jerky movements, making it an ideal choice for applications that require precise and controlled movements. We use the Gaussian Basis function in DMP [21], which is given as follows:

$$\psi_i = \exp\left(-\frac{(s-c_i)^2}{2\sigma^2}\right), \tag{12}$$

Where: - ψ_i is the i^{th} Gaussian basis function in (12), c_i is its center, σ determines the spread or width of the normal distribution, and σ is calculated as $\frac{1}{\sqrt{2h}}$, where h is the width of the basis function.

3) Weight Estimation: Basis function weight determines the impact of each basis function on the trajectory. The higher the weight, the greater the influence on the trajectory. We use a locally weighted regression method, according to the locally weighted regression algorithm, w_i can be calculated by the following equation:

$$w_i = \frac{\zeta^\top \Phi f_t}{\zeta^T \Phi \zeta}, \tag{13}$$

where variable ζ depends on the phase variable, initial position ($y0$), and goal position (g). Φ is a diagonal matrix where each diagonal element corresponds to the value of the basis function ψ for a specific time step. The desired (or target) forcing function f_t is required for computing the basis function weight. To achieve this goal, we need to learn the target forcing function. To learn the target forcing function [21], the function f_t in equation (13) is calculated as

$$f_t = \tau^2 \ddot{x} - \alpha(\beta(g-x) - \tau\dot{x} \tag{14}$$

4) Execution and adaptation: Once, the necessary parameters mentioned in the above steps are computed, the desired trajectory is generated by executing the DMP equation (10). The system behaviour is monitored, and the hyperparameters α, β, α_x, N and α_c are updated if needed to reduce errors or improve the trajectory.

IV. Results and Discussions

A. Experimental Setup and Implementation details

In the system setup, we will go over all of the software and hardware configurations that are used to carry out the task.

1) Software: We use ROS and Kinova libraries [22] to control the robot manipulator and Kinect camera. For training, we utilize Hugging Face Transformers and PyTorch, with cross-entropy loss aggregated across all heads for better performance. The loss is backpropagated through the MLP decoder. Initially, we freeze the encoder (pretrained videoMAE on Kinetics-400) for decoder convergence, then unfreeze it for fine-tuning. We use the ADAM optimizer and cosine annealing for learning rate scheduling.

2) Hardware: A 7-DOF Kinova Robot Manipulator (Fig. 3) is used for executing different tasks. A visual sensor (i.e., camera) on its top captures the video of human gestures in real time. The background remains as usual. Real-time inference has been performed with a Gigabyte GeForce GT 710 2GB GPU clock speed of 954 MHz. Model Training has been done on a single GPU server with 4 Nvidia A4000 16 GB GPUs.

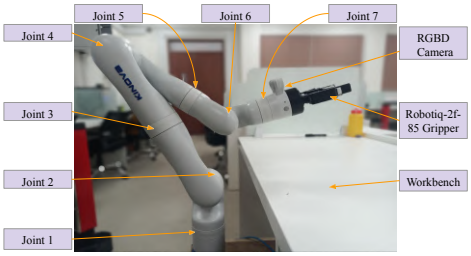


Fig. 3: 7-DoF Kinova Gen3 Robotic Manipulator.



(a) Human demonstration of pick and place task.



(b) Robot execution of pick and place task.

Fig. 4: Pick and place task.

B. Experimental Results

The experiments were done for four different tasks. The tasks were performed in different orientations and different table backgrounds. These experiments were conducted in low background clutter to avoid misinterpretation of semantic relationships in the environment. In-the-wild generalization is to be explored in our future works using environment segmentation.

1) Pick and place task: Pick and Place tasks (Fig. 4) are very fundamental for evaluating a manipulation policy. Our model gives good results in successfully identifying the primitives in the correct order for this task.

2) Pick and pour task: This task (Fig. 5) involves pouring a liquid from one container to another container. The motion involves a trajectory similar to Pick and place except for a tilt motion. Our model confuses these two tasks sometimes and therefore leads to a decrease in the accuracy. **Example:** Ground truth label: Reach–Grasp–Reach–Tilt–Reach–Release–Retract, Predicted Label: Reach–Grasp–Reach–Reach–Release–Retract



(a) Human demonstration of pick and pour task.

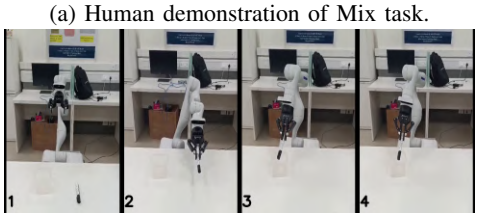


(b) Robot execution of pick and pour task.

Fig. 5: Pick and Pour task.



(a) Human demonstration of Mix task.

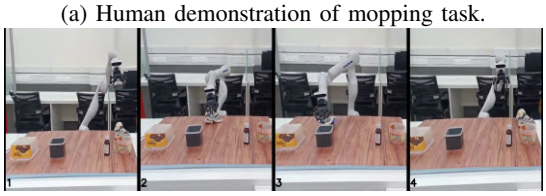


(b) Robot execution of Mix task.

Fig. 6: Mix task.



(a) Human demonstration of mopping task.



(b) Robot execution of mopping task.

Fig. 7: mopping task.

3) Mix task: Mix task (Fig. 6) involves mixing a liquid using a spatula or a spoon. Our model predicts the sub-tasks of this long horizon task with an accuracy of 84.00%. The stir motion is semantically very different from other subtasks and hence is easily captured.

4) Mopping task: This task (Fig. 7) involves the movement of a piece of cloth on a table to and fro to clean the surface. This task is particularly difficult since it involves repeated motions which can be very difficult for the model to understand. Our model sometimes confuses the mopping task with multiple instances of Reach and Release due to overlapping motion patterns. **Example:** Ground Truth label: Reach–Grasp–Wipe–Release–Retract, Predicted Label: Reach–Grasp–Reach–Release–Reach–Release–Retract

We have used multiple objects for the same demonstration and video augmentations to provide generality across various demonstrations. The video augmentations used are changes in brightness and contrast, the addition of blur and noise, and horizontal and vertical flipping. We have used a standard $4:1$ split for training and testing data. The accuracy for each task using different encoder architectures is shown in Table II. Table III shows the overall accuracy for each architecture.

We have also verified the semantic correctness of the model weights obtained after the training. The model should be able to predict distinct embeddings for individual sub-tasks after training on the full-task videos only. We collected around 40 samples each for individual sub-tasks. In the Fig. 8, t-distributed Stochastic Neighbor Embedding (t-SNE)

Results			
Encoder	**Task**	**Number of videos**	**Accuracy**
TimeSformer	Pick and Place	50	88.00 %
	Pick and Pour	50	76.00 %
	Mix	50	58.00 %
	mopping	24	54.16 %
ViViT	Pick and Place	50	76.00 %
	Pick and Pour	50	68.00 %
	Mix	50	46.00 %
	mopping	24	70.83 %
VideoMAE	Pick and Place	50	92.00 %
	Pick and Pour	50	66.00 %
	Mix	50	84.00 %
	mopping	24	83.33 %

TABLE II: Accuracy for individual tasks over different encoders.

Accuracy aggregated over tasks	
Encoder Architecture	**Accuracy**
TimeSformer	71.26 %
ViViT	64.36 %
VideoMAE	81.03 %

TABLE III: Accuracy aggregated over all tasks. Ablations have been shown for different encoder architectures.

[23], the last hidden states of the encoder model, we observe that the model can generate meaningful embeddings of the individual sub-tasks. This supports our hypothesis of training full-task videos to obtain primitive segments. We mapped the DMP trajectories for individual joint states of the robot and verified the execution in real time. Figure 9 shows a plot of joint angles to time for the pick and place task.

C. Discussion

The major contribution of this work is the idea of decoupling a complex long-horizon robot execution into a set of primitives that can be easily performed and understanding the information in the videos. We tested three different encoder architectures: VideoMAE, TimeSformer, and ViViT. This ablation supports the effectiveness of masking techniques for video understanding, which allows for selective focus

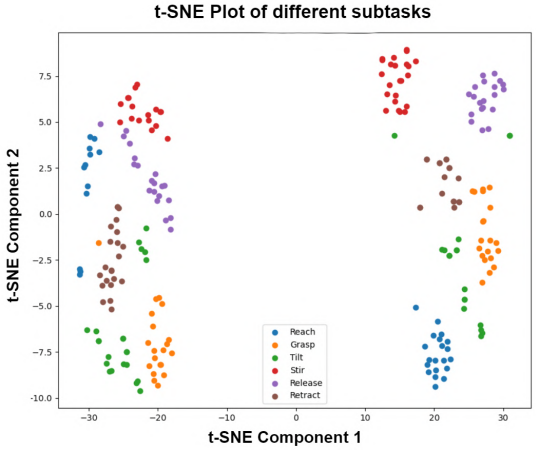


Fig. 8: t-SNE plot of sub-task embeddings.

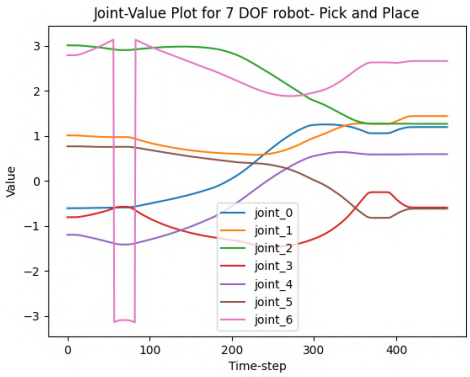


Fig. 9: Joint angles from DMP model for Pick and Place

on relevant parts of the input data. The ablation can be seen in the Table III. The TimeSformer backbone uses Spatio-temporal attention only whereas videoMAE utilizes the masked autoencoding paradigm along with spatio-temporal attention. The masking approach enforces the model to capture more meaningful and robust representations of the video. Trained in a self-supervised manner on the extensive Kinetics-400 dataset, VideoMAE achieves highly generalized representations. This paradigm of self-supervision and masking is the reason behind improved performance over TimeSformer and ViViT.

The ViViT model uses a two-stage hierarchical attention mechanism, applying spatial attention first and then temporal attention. While ViViT performs better for videos with diverse actions like dancing or swimming, our dataset involves more similar actions, requiring joint spatiotemporal attention, which makes TimeSformer perform better. We also tried LSTM-based and attention-based decoders, but they failed to converge. The current model is trained for up to 10 subtasks in a sequence, but it can be extended for more subtasks with a larger dataset.

V. Conclusion

We propose a method for learning from video demonstrations by separating video recognition and low-level execution. The video recognition module segments demonstrations into primitives, which are then executed by a DMP model. This meta-learning framework enables learning new sequences by breaking them into subtasks, with data masking showing promising results. The framework, therefore, enables the identification of a sequence of primitives for a new task from just one or a few demonstrations. However, this is constrained by the number of primitives available in the training data. Improvements can be made in handling semantically similar actions and adding a feedback mechanism to the DMP module to retry failed executions. Another potential enhancement is generating object-action pairs from long-horizon demonstrations for better representations. Expanding the set of primitives will also be explored in future work to enhance the flexibility of the approach.

References

[1] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Comput. Surv.*, vol. 50, no. 2, p. 35, 2017.

[2] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human-robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, 2018.

[3] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[4] A. Chella, H. Dindo, and I. Infantino, "A cognitive framework for imitation learning," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 403–408, 2006.

[5] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[6] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," 2015, pp. 4694–4702.

[7] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," vol. 9912, 10 2016.

[8] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4724–4733.

[9] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," 2018, pp. 7794–7803.

[10] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6201–6210.

[11] C. Feichtenhofer, "X3d: Expanding architectures for efficient video recognition," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 200–210.

[12] G. Bertasius, H. Wang, and L. Torresani, "Is space-time attention all you need for video understanding?" 02 2021.

[13] D. Tu, W. Sun, X. Min, G. Zhai, and W. Shen, "Video-based human-object interaction detection from tubelet tokens," 06 2022.

[14] L. Wang, B. Huang, Z. Zhao, Z. Tong, Y. He, Y. Wang, Y. Wang, and Y. Qiao, "Videomae v2: Scaling video masked autoencoders with dual masking," June 2023, pp. 14 549–14 560.

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

[16] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, "Vivit: A video vision transformer," 2021, pp. 6816–6826.

[17] Y. Bai, F. Li, M. Zhao, W. Wang, Y. Li, and R. Song, "Robot manipulation skill learning based on dynamic movement primitive," in *2022 12th International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, 2022, pp. 579–584.

[18] S. Ambhore, "A comprehensive study on robot learning from demonstration," in *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, 2020, pp. 291–299.

[19] A. Ude, B. Nemec, T. Petrić, and J. Morimoto, "Orientation in cartesian space dynamic movement primitives," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 2997–3004.

[20] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, pp. 328–373, 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:2431443

[21] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *CoRR*, vol. abs/2102.03861, 2021. [Online]. Available: https://arxiv.org/abs/2102.03861

[22] Kinova Robotics . (2020) Ros packages for kinova® kortex™ robotic arms. [Online]. Available: https://github.com/Kinovarobotics/ros-kortex

[23] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.