State of Video Understanding Models as Robot Task Verifiers

Alper Canberk

***Abstract*— In this report, we discuss methods for building a robot system that learns low-level robot manipulation primitives from ego-centric videos. To evaluate the feasibility of this approach, we conduct a thorough analysis of the pre-trained action classifier VideoMAE, trained to classify actions on the egocentric Something-Something-V2 dataset. Finally, we discuss our plans on how to move forward with this project.**

I. INTRODUCTION

The development of robust low-level manipulation primitives for robots has become increasingly important with the adoption of large language models (LLMs) as high-level planners for robotic systems. Now that LLMs enable robots to common-sensically decompose long-horizon tasks into smaller, executable actions, the effectiveness of such language-guided systems is limited by the flexibility and success rate of the available manipulation primitives. Previous approaches for acquiring manipulation skills, such as hand-engineering or behavior cloning, rely on human input in the training process and may not scale up sufficiently for open-world applications. In order to overcome these limitations, there is a need for methods that allow robots to learn manipulation skills from abundant internet-scale data sources, such as human videos. However, previous approaches that have attempted to learn from human videos have been constrained to limited objects and limited environments [1] [3].

In this report, we propose a novel approach: a robot can first learn arbitrary 3D manipulation actions in a semantically abstracted, sim2real generalizable way, and then bring context and semantics into these actions using LLMs and VLMs only in the downstream tasks (e.g. the robot can first learn how to "put [something] in [something]", for arbitrary objects, later we can specify what objects should go in these blanks).

As a first step, we evaluate the feasibility of this approach through a thorough analysis of the state-of-the-art action classifier, VideoMAE [2], which was trained on the Something-Something-V2 dataset [4]. Using insights from these analyses, we then discuss how we can move forward in order to enable robots to learn manipulation primitives from ego-centric human video data.

II. PROBLEM DEFINITION

In recent years, there has been growing interest in using video understanding models to enable robots to learn manipulation skills from visual data. However, little research has been conducted on how these models perform in the context of robotic manipulation tasks depending on various environmental factors. In this paper, we aim to address this gap in knowledge by examining the effects of camera

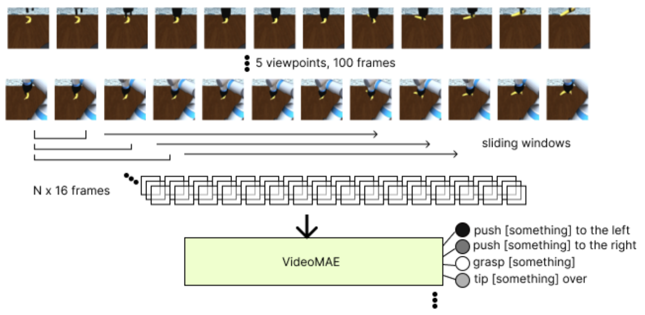


Fig. 1. Batch Evaluation Pipeline.

angle, input video segment, object being manipulated, and background on the ability of video understanding models to accurately classify actions. Additionally, we explore how to best utilize the output of the action classifier in the context of robotic manipulation. Through this analysis, we hope to provide insights into how video understanding models can be effectively applied to enable robots to learn manipulation skills from visual data.

III. APPROACH

In order to study the performance of the action classifier on robot data, we hand-coded manipulation primitives for the actions "Push" and "Grasp" and introduced slight randomization to allow for both successful and unsuccessful actions. These primitives were then generated in simulation and recorded in video form. By analyzing the responses of the action classifier to these videos, we aimed to gain insights into the effects of various environmental factors on its performance. The details of the hand-coded manipulation primitives can be found in the appendix.

A. Simulator

Initially, we utilized the PyBullet physics engine and the Panda Arm in our simulations. However, we later decided to switch to the higher-level simulator iGibson and the Fetch Robot due to the availability of assets and tools. As a result, the majority of our experiments were conducted using iGibson. We later discovered that the physical interactions in iGibson were not entirely accurate, leading us to also conduct an experiment using the MuJoCo physics engine.

B. Action Classifier

For the video understanding model, we were inspired by the work of Concept2Robot and chose to use a model trained on the Something-Something-V2 dataset. To evaluate the performance of this model, we utilized VideoMAE, the current state-of-the-art action classifier trained on Something-Something-V2. VideoMAE processes 16 frames of 224x224 images as input and returns prediction probabilities for 174 action categories within the Something-Something-V2 dataset.

C. Video Collection Technique

While previous works have used 16 equally spaced frames from an entire trajectory to feed into the action classifier, we collected 16 frames using sliding windows of various widths over the trajectory video. This way, we were able to collect 16 frames from all parts of the video with various amounts of spacing. In addition, we collected the trajectory videos from multiple viewpoints for testing the effect of camera angle on the action classifier. After many 16-frame windows were collected, they were pre-processed and forwarded through VideoMAE in batches (Fig. 1.).

IV. RESULTS

A. Metrics

For each action primitive (push and grasp), we hand-designed both a dense reward metric as well as a success condition (in Appendix). Then, for a given trajectory and a given primitive, we observed the correlation of the dense reward and success condition with respect to the outputs of the action classifier.

1) **Absolute Logit**, which corresponds to the raw activation of an action primitive in the output layer of VideoMAE. Given that the outputs from the final dense layer of VideoMAE are denoted $\mathbf{o}$ and the selected action primitive has index j, the absolute logit corresponds to the value $\mathbf{o}_j$
2) **Prediction Probability (Normalized Logit)** Let the specified action primitive be indexed by j. The prediction probability for this specified action primitive is

$$p_j = \frac{e^{\mathbf{o}_j}}{\sum_i e^{\mathbf{o}_i}}$$

The reason why we measure the prediction probability separately from the absolute logit is that in robot videos, since the robot arm is distinct from a human arm, seeing a robot arm can trigger activations related to action primitives that signify tool use, which in turn lowers the predicted probability of the desired action primitive. (e.g. pushing [something] with [something] as opposed to pushing [something]). In an effort to obtain predictions that are more "individual", we also decided to measure the absolute logit.

B. Grasp Banana in iGibson

Viewpoint	Abs Logit	Normalized Logit
Left	0.48	0.38
Right	0.49	0.29
Top	0.05	0.02
Robot	0.40	0.05
Low	0.87	0.86
Max	0.87	0.86

Correlation Between Task Success and Predicted Metrics

The grasp banana task had a surprising amount of correlation between the success metric and the action classifier predictions. As seen in Figure 4., the low robot view, which is roughly orthogonal to the end-effector movement, activates the output neuron corresponding to "grasp [something]" with a 0.87 correlation with respect to the task successfully. It's also worth noting that the top-down camera view did not activate much because it couldn't clearly see the action. Another important conclusion from this experiment is that the absolute logit predictions (left column of the table) generally seem more correlated with the ground truth metric than the normalized logit.

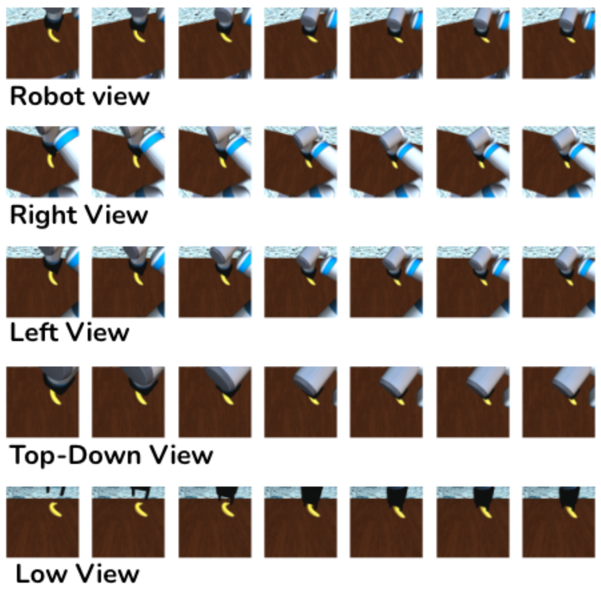


Fig. 2. *iGibson Grasp Banana From Different Views* The same robot trajectory from 5 different angles.

For the grasp banana task, we also tested various windows and spacings from which to take the 16 frames:

1) 16 frames equally spaced from the beginning to the end
2) the maximum prediction of the 16 frames that begin in the first half of the video, and finish near the end of the episode.
3) the maximum prediction of the 16 frames that finish near the end of the episode.

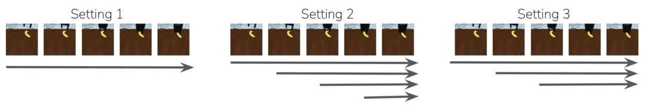


Fig. 3. Three different window selection settings.

Viewpoint	Setting 1	Setting 2	Setting 3
Prediction Confidence	0.87	0.71	0.73

From the results given in the table above, we can conclude that naively feeding in the 16 equally spaced frames into the action classifier gives us the best performance. This is interesting because the actual grasping action happens in the second half of the episode, and one would expect a video in which only the grasping happens to score better. However, it seems that having some padding before grasping helps prediction.

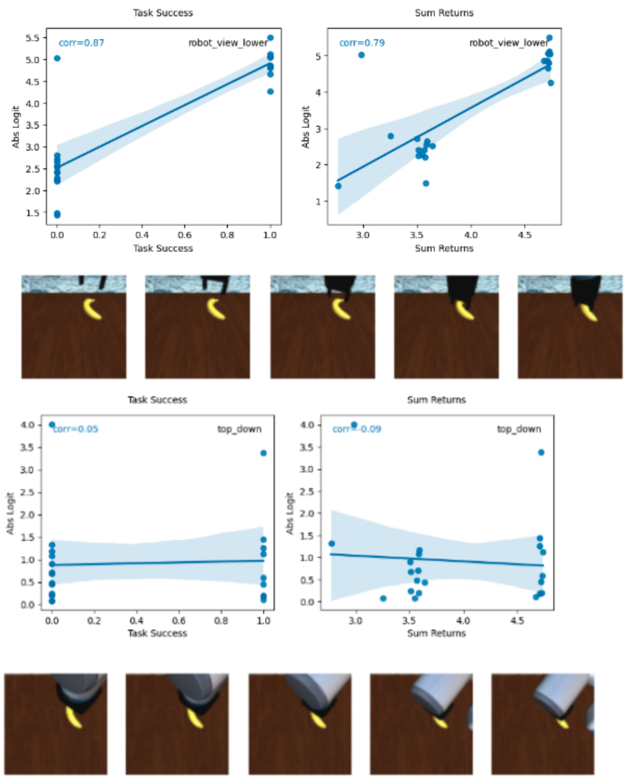
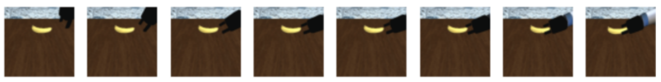


Fig. 4. *Grasp Banana Prediction Based On Viewpoint*

C. *Push Object in iGibson*

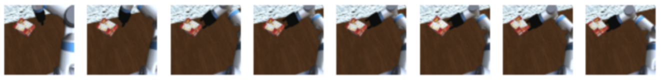
1) *Push Banana From Right to Left:*



Viewpoint	Abs Logit	Normalized Logit
Left	0.03	−0.18
Right	0.43	0.05
Top	0.60	0.29
Robot	0.33	0.05
Low	0.71	0.43
Max	0.71	0.43

Correlation Between Task Success and Predicted Metrics

2) *Push Box From Right to Left:*



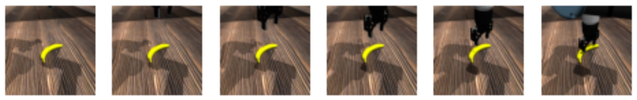
Viewpoint	Abs Logit	Normalized Logit
Left	0.35	0.06
Right	0.64	0.31
Top	0.51	0.49
Robot	0.38	0.23
Low	0.73	0.54
Max	0.73	0.54

Correlation Between Task Success and Predicted Metrics

The correlation results for the push task are quite a bit noisier than the grasping task. Our suspicion is this is due to the fact that an object can be pushed to the left by different amounts and that our hand-defined pushing distance threshold may not reflect that of the model.

We can see that the correlation numbers for the banana and the box are slightly different, but this difference seems to be minimized in the "Low View", which was also the best view for the grasping task. This shows that picking a good viewpoint for grading may be important for robustness against objects of different scales.

D. *Grasp Banana in MuJoCo*



Viewpoint	Abs Logit	Normalized Logit
Left	−0.11	−0.11
Right	0.73	0.73
Top	0.24	0.24
Robot	0.10	0.10
Low	0.14	0.14
Max	0.73	0.73

Correlation Between Task Success and Predicted Metrics using Darkwood Background

Viewpoint	Abs Logit	Normalized Logit
Left	0.14	0.07
Right	0.43	0.49
Top	0.22	0.27
Robot	0.03	−0.10
Low	0.48	0.65
Max	0.48	0.65

Correlation Between Task Success and Predicted Metrics using Lightwood Background

Viewpoint	Abs Logit	Normalized Logit
Left	−0.00	−0.13
Right	0.59	0.72
Top	0.16	0.25
Robot	0.28	0.19
Low	0.39	0.46
Max	0.49	0.72

Correlation Between Task Success and Predicted Metrics using White Background

The correlation between the ground truth metrics and the predicted metrics from MuJoCo is slightly worse than that of iGibson, regardless of the background. This could be due to factors such as the robot arm positioning of the camera (no table), robot shadows, and complex background pattern. In this regard, the MuJoCo environment needs some more tuning until we can train a useful policy on it.

In this setting, the significance of picking a good viewpoint seems to be amplified even further (with the left view having a much higher correlation than all the other viewpoints).

V. TAKEAWAYS

1) **What's the effect of the camera angle?** The camera angle that we use for the prediction can have a significant impact on the performance. In the simulation, the best way to find this camera is to try all cameras and take the maximum per trajectory. However, in the real world, it might be expensive or not feasible to have that many cameras around the robot. In this case, a method for finding the best camera angle would be extremely useful.
2) **What's the effect of feeding in different parts of the video?** As long as the prediction window is at a reasonable location in the video, the difference is very insignificant. Therefore, the simplest and the most general choice is to take the frames from the entire trajectory.
3) **What's the effect of the object being manipulated?** Although changes in the object can result in slight differences in prediction correlations, this difference is reasonable. In addition, the magnitude of the absolute logit predictions for the banana and the box are distributed very closely, which makes us optimistic about this method's robustness against variation in the objects that are being manipulated.
4) **How can we best use the outputs of the action classifier?** As we suspected from the beginning, isolating the desired action primitive by using its "absolute logit" generally gives higher correlations than simply using the predicted probabilities, which takes into account all 174 action primitives.
5) **What's the effect of the background?** The experiments from MuJoCo demonstrate that having an appropriate background can lead to nontrivial changes in the GT vs. Predicted Metric correlations. For this reason, going forward, it will be important that we pick the right background for the task. Our initial hypothesis is that backgrounds that are simpler lead to better results, but further investigation is needed in this direction.

VI. NEXT STEPS

A. Tasks

The three tasks I'd like my system to be able to tackle are push, grasp, and open drawer. Although Concept2Robot "succeeds" on these tasks, the environments for these tasks are in a configuration such that the robot can accomplish them by performing a single linear motion. We would like my method to go beyond the limited environment constraint. Although these tasks are basic, each of them has their nuances

1) Push: Pushing is an extremely primitive task because it doesn't require much precision. However, the nuance of pushing resides in the customizability of the pushing direction. An object could be pushed right, left, or even towards some arbitrary object on the table. In addition, pushing might be tricky to get right with an action classifier because it's not clear how much pushing the action classifier considers the most pushing. Is there a threshold at which if an object is pushed more than that, the classifier confidence decreases? What happens if the robots keep pushing object A towards object B but does not stop near object B but keeps pushing object B with object A? Questions such as these may require additional constraints to the problem definition, which would in turn limit the generality of our framework. Finding the right balance will most certainly be challenging.

2) Grasp: A successful grasp requires a nice trajectory toward the object that needs to be grasped, a precise and secure grasp, and a pull. The most challenging aspect of this is getting precise grasping, which makes successful grasps very sparse. Luckily, the action classifier seems to be able to recognize if an object is picked up correctly by observing the object with respect to the arm motion after the grasp is performed. This means that efficient exploration of the action space is crucial for succeeding in this task.

3) Open Drawer: The monumental task I'd like my system to be able to tackle is opening a drawer. This is not only an impressive task, but it also highlights the importance of visual feedback. Whereas the supervision for the push and grasp tasks can be described visually or using the center of mass coordinate of an object, opening the drawer results in a much more complicated visual result that heuristics cannot trivially recognize. Unfortunately, opening a drawer is the most complicated task of all because it merges the progressive nature of pushing (how much to pull the drawer handle) with the sparsity of grasping.

B. *Training a Policy*

All three of the tasks that we would like to accomplish can be done with actions in the end-effector space. Therefore, going forward I'd like to adopt the action space of PerAct. To even further constrain this action space we can simplify it to predicting two consecutive end-effector actions. Given the two waypoints, the robot will move to the first waypoint, then the second one, and back to the first one. This formulation captures grasping, pushing, and opening drawer.

As for the state space, we would like to use a 3D representation due to the Sim2Real effectiveness of such methods. We would like the robot to first learn actions in simulation using very simplified and segmented 3D visuals with solid colors while receiving supervision from the action classifier that observes the real scene from a 2D window. When applying this system in the real world, we can parse a real scene into the segmented 3D format that the robot sees to achieve high Sim2Real generalization.

C. *Viewpoint Selection*

The analyses that we have run show that camera viewpoint has a strong impact on the strength of the supervision we get from a given video. If we would like this project to go in a direction where it can train autonomously in the real world, then we might certainly need to position a camera well so that it can clean supervision. Along this line, we propose the following ideas

1) **Observe the given action from a direction that's orthogonal to the direction of the action.** The reason why the lower viewpoint did so well on the iGibson results we presented in this report is that the lower view was positioned almost orthogonally to the direction of the end effector movement. Therefore, it observed the highest magnitude of movement. This can be used as a simple yet effective heuristic to guide camera position.
2) **Try to maximize the similarity of the initial frame with the initial frames from the Something-Something-V2 dataset** Assuming that we get the best supervision from videos with initial frames that are similar to the initial frames from the Something-Something-V2 dataset, here's what we could do to capture these similarities.

   We can first obtain a bunch of non-egocentric images from the internet that are somewhat similar in environment to the environments in the Something-Something-V2 dataset. Then we could try to learn a representation that pushes the initial images in Something-Something-V2 closer together and the different pairs further apart.

   If different action categories also happen to differ in terms of the initial frame, we can come up with a different weight pulling together similar vs different action primitives.

   If the dataset we're working with provides some information about the camera viewpoint with respect to the action being performed in the video, and it happens that we need our representation to capture information about the camera viewpoint, then we could also add an auxiliary loss for reconstructing the camera parameters from the given representation.

   After we train this representation, we can sample a few different viewpoints before we start any robot action, score each viewpoint according to various metrics (e.g. distance to the nearest point in the latent space with the given action primitive label), and then pick the best candidate out of those viewpoints. We could then proceed onto the training of the action primitive.

References

[1] Shao, Lin, Toki Migimatsu, Qiang Zhang, Karen Yang, and Jeannette Bohg. "Concept2Robot: Learning Manipulation Concepts from Instructions and Human Demonstrations." In Proceedings of Robotics: Science and Systems (RSS), 2020.

[2] Tong, Zhan, et al. "Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training." arXiv preprint arXiv:2203.12602 (2022).

[3] Chen, Annie S., Suraj Nair, and Chelsea Finn. "Learning generalizable robotic reward functions from" in-the-wild" human videos." arXiv preprint arXiv:2103.16817 (2021).

[4] Goyal, Raghav, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fründ, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. "The 'Something Something' Video Database for Learning and Evaluating Visual Common Sense." In IEEE/CVF International Conference on Computer Vision, 2017.