Inferring Actions from Video Demonstration

Anirudha Paul and Rosella Liu

Introduction

Offline demonstration has been a popular topic in the deep learning community. With vast amounts of video data available on the internet, it's researchers' natural inclination to attempt leveraging them to learn an optimal policy for specific tasks and action spaces. Among all of the research done in this area, *VPT* ("Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos") serves as a unique presence. Proposing the Inverse Dynamics Model architecture, the non-causal nature of the model differs from both the traditional imitation learning policy and behavioral cloning by considering the past and future events and is thus easier and more efficient to train.

Our proposal initially aimed to train a Boston Dynamics Spot with human demonstrations through VPT architecture so the robot can open a door without the assistance of human pixel input, like how it is carried out in the robot's Python SDK. However, due to the high demand for Spot and the lack of transparency of the VPT repository, we ended up shrinking the scope of our project and aiming at building the Inverse Dynamics Model submodule from scratch and test it performance on a different domain by attempting to train it to predict the following action inferred from video frames data collected from Airsim Block Environment in Unreal Engine.

Related Work

Imitation learning methods aim to construct a policy that accurately models the behavior distribution in a dataset of action-observation pairs. These policies need to be causal, conditioning current and past observations only. While labeled demonstrations enable straightforward imitation, alternative methods are required when such labels are unavailable. Some approaches use adversarial objectives to match behaviors from an unlabeled dataset[2], while others leverage latent policies[3] or motion-capture techniques[4]. Pathak et al.[5] and Nair et al.[6] train goal-conditioned policies, while Torabi et al.[7] employ an inverse dynamics model (IDM) and behavioral cloning (BC) in tandem. However, the IDM-BC approach requires the BC model to explore efficiently for effective learning.

Our implemented model is directly inspired by Open AI's *VPT* paper[1], where the non-causal IDM structure is introduced. The motivation of the paper stems from existing semi-supervised imitation learning methods' susceptibility to exploration bottlenecks while aiming to learn with few or no explicit action labels. To generate a pseudo-label for the vast amount of the unlabeled offline demonstration data, the researchers gathered a small amount of labeled data to train an inverse dynamics model (IDM) that predicts the action taken at each timestep in a video. Using

pseudo-labels generated from the IDM, the researchers train the *VPT* foundation model, which mimics the distribution of behavior in the previously unlabeled dataset with standard behavioral cloning at scale, and fine-tune the *VPT* foundation model to downstream tasks with either behavioral cloning or reinforcement learning. However, despite the amazing performance on Minecraft data promised in the paper, when we ran the *VPT* code, in all instances, the model failed to mine diamonds for a long period of time. Although it was our hope that we could reuse the IDM architecture introduced in the paper and tailor it for action prediction in a different domain, we found that the code in the repository only publicizes weights for the model.

Thus, our work focuses on two goals- rewriting the OpenAI's IDM architecture in a way that can be trained using a single GPU and training that model to predict actions taken in unlabeled demonstrations in a different environment and agent other than Minecraft game — in our case, it's the Block Environment on Unreal Engine with Microsoft AirSim agent.

Methods

Confronted by the challenges in the original code, we decided to build the IDM model from scratch and train it on a labeled video frames dataset collected from the Unreal Engine Block environment. For the implementation of the code, we initially followed the original architecture of the IDM proposed in the paper then adjusted it to fit in a single GPU.

The input of the OpenAI's IDM was 128 consecutive image frames, each with dimensions 128 * 128 * 3, and in our version of the architecture, we pass in 32 frames to use less VRAM.

These frames first go through a *3-D* convolution with 128 learnable filters with a temporal kernel width of 5 and spatial kernel widths of 1. This is a non-causal convolution layer, meaning embeddings at time t not only contain pixel values at times $t - 2$, $t - 1$, t but also from time $t + 1$ and $t + 2$. This means the temporal information this layer is capturing contains both past and future information.

The output of this temporal convolutional layer is fed into stacked ResNet blocks. In the original *VPT* paper, each block is comprised of, in the order of:

1. A convolutional layer with 3*3 kernel size, 1-pixel zero padding at the embedding boundary with 1 stride.
2. A 3x3 max pooling with stride 2 and padding 1 such that the embedding width and height becomes half in the output channel.
3. Two classic ResNet blocks, which is basically 2 convolution layer with 3*3 size with a *ReLu* in between.

In the original paper, the researchers stacked three ResNet blocks together. However, due to our lack of computation power, we decreased the number of ResNet stacks in our implementation from 3 to 2.

After processing the frames with ResNet stacks and flattening the output channel, the data is finally fed into two frame-wise dense layers with 256 output activations and 4096 output activations. In the original paper, the result is fed into four subsequent non-causal residual transformer blocks. Each block consists of an unmasked attention layer, with 32 attention heads and a surrounding residual connection that skips this layer. As part of our effort to shrink our model and reduce computational complexity, we used one residual transformer block instead of four.

The Transformer processed embedding is then again passed through two frame-wise dense layers. In "VPT," they also used a residual connection to skip past these dense layers, but we chose to avoid this.

Finally, independent dense layer heads for each action are pulled from the final embedding – a 2 class on /off categorical parameterized with a softmax for each available key. In our case, three keys - representing W (move front), A (pan left), and D (pan right) - the actions taken in each frame in order to move to the next frame.

Experiments

The dataset that we used for both training and validation are collected from the Unreal Engine block environment paired with Microsoft Airsim drone simulation. When controlling the drone in the environment, the script of the project will take a frame of the action sequence when either "w" (move front), "a" (pan left), "d" (pan right) keys are triggered and record the action of the human-controlled drone. We store the labeled data with a timestamp in each frame's file name. We then wrote a script to take random action in the world to automate the data collection process. In total, we collected 43808 frames of data for training purposes and 1280 frames of data for validation purposes.

Though the original authors have used data augmentation techniques using PyTorch transforms library, we did not use those techniques in our experiment. Besides this, in terms of training scheme, the researchers at OpenAI used an *ADAM* optimizer with linear learning rate decay. In the paper, the parameters of the model are set to 0.003 for learning rate of 0.003 and 0.01 for weight decay of. The researchers trained the network for 20 epochs on 32 A100 GPUs for 4 days.

Initially, we tried to follow their setup of hyperparameters and optimizers. Some major differences between our approaches include the input size — we were feeding 32 frames at a time instead of 128 frames to lower the computation cost. We used cross-entropy loss between

the final softmax layer and one-hot encoded true action. For training, we only used a single RTX 3080ti with 12 GB VRAM and restricted the training time to under two hours.

However, this setup generated a very disappointing result. The loss was not decreasing at all with these hyperparameters.An attempt to resolve this by increasing the learning rate resulted in exploding gradient behavior. In addition to tuning the hyper parameter, we also changed the loss function by switching to SGD optimizer with a learning rate of 0.001, which generates much better training and validation performance. Along with the decrease of the loss, the accuracy in our validation also started increasing.

So we conducted three separate tests - first two tests, we used only 12800 frames instead of our full test dataset; for our final test, we used our full dataset of 43808 frames to train the IDM model.

The loss curve and accuracy graph of these tests can be seen below.

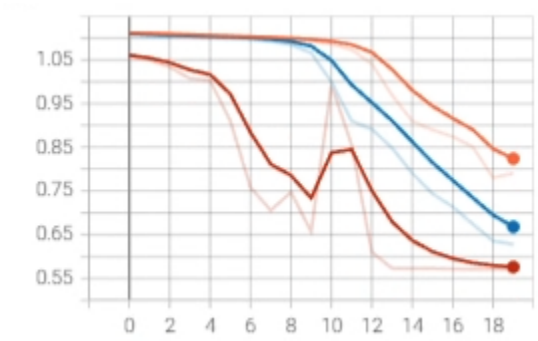

Figure 1 - Loss curve on test data during training in each epoch
(Orange and Blue line represents the test with 12800 frames
and Red represents the test using full training dataset)

-

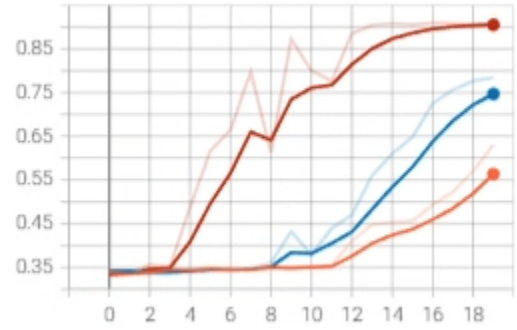

Figure 2 - Accuracy curve during training on validation data in each epoch
(Orange and Blue line represents the test with 12800 frames
and Red represents the test using full training dataset)

As seen in Figure 2, even with a small amount of training data, the accuracy reaches around 70% after the end of 20 epochs. And the performance improvement is also proportional to the amount of training data. As we increased the training dataset size, the accuracy also increased significantly and reached 90%.

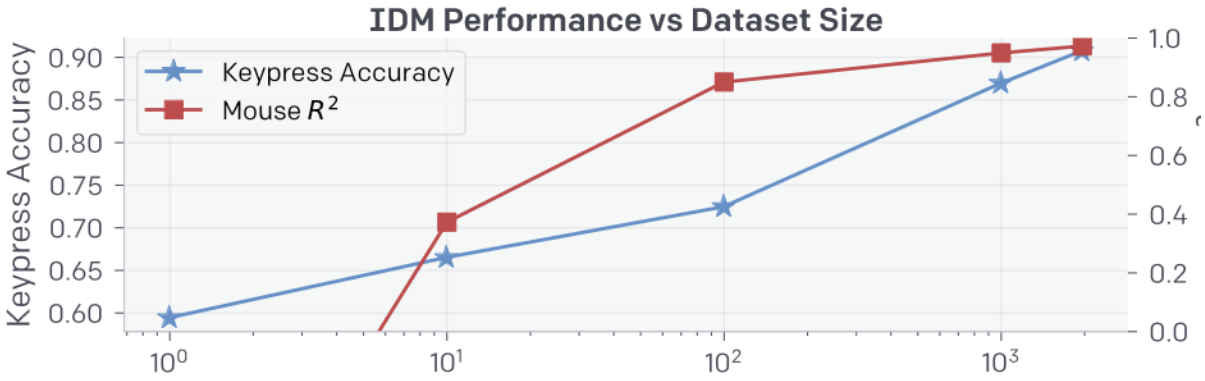

Figure 3 - IDM Key Press accuracy in original VPT paper[1]

This result is statistically significant because it is on par with the accuracy the authors of the original VPT paper achieved, as seen in figure 3. The result implies that if the input data describes a simpler world, we can scale down the data and computational requirements for training a reasonable IDM. In our case, we used only 43808 frames compared to 141264000 frames (1962 hours of 20Hz video) used by the original authors.

Conclusions

We aimed to check the viability of the Inverse Dynamic Model structure in domains other than the Minecraft environment. Our experiments show that this is indeed a viable method to label actions in time series data. Another significant conclusion we can draw is that if the domain is simpler, we don't necessarily require the vast amount of data and computation power used in "VPT". However, some hyperparameter tuning and small architectural changes are required for the domain shift. The vanilla architecture of IDM cannot be directly deployed for action prediction for every domain.

The code can be found in this Github repository (https://github.com/anirudha-ani/VPTAirSim)

References

[1] Baker, B., Akkaya, I., Zhokov, P., Huizinga, J., Tang, J., Ecoffet, A., … Clune, J. (2022). Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), Advances in Neural Information Processing Systems (Vol. 35, pp. 24639–24654). Retrieved from https://proceedings.neurips.cc/paper_files/paper/2022/file/9c7008aff45b5d8f0973b23e1a22ada0-Paper-Conference.pdf

[2] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. Advances in neural information processing systems, 29, 2016.

[3] Ashley Edwards, Himanshu Sahni, Yannick Schroecker, and Charles Isbell. Imitating latent policies from observation. In International conference on machine learning, pages 1755–1763. PMLR, 2019.

[4] Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. Sfv: Reinforcement learning of physical skills from videos. ACM Transactions On Graphics (TOG), 37(6):1–14, 2018.

[5] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A. Efros, and Trevor Darrell. Zero-shot visual imitation. In ICLR, 2018.

[6] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. pages 2146–2153, 05 2017. doi: 10.1109/ICRA.2017.7989247

[7] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. arXiv preprint arXiv:1805.01954, 2018.

Division of labor

Anirudha Paul took care of environment setup, architectural design, programming responsibility and developed the test suit. He also conducted tests with different changes and hyperparameter tuning to achieve the final result. He revised the final draft of the report and introduced the test result figures.

Rosella Liu actively contributed in cross-checking the viability of the architectural changes needed to fit the project within limited computational setup and verified the code against the original paper description, Airsim Unreal Engine setup on the MacOS environment wrote a data collection script for unreal engine camera rolling, and the first draft of the final report, refined the completed text.