RLMU: Reinforcement Learning with Self-supervised Human Motion Analysis

Xu Cao **Umang Sharma** **Anant Rai**

Abstract

Unlabelled videos of humans are a large and almost untapped source of data for robots to learn several routine tasks. These could range from simple movements like waving a hand to more complex ones like operating machinery. The problem lies in closing the gap between the variety of domains that a human video could appear in to the domain of the environment the robot interacts with during training. This work follows up on a previous work based on learning via cross domain adaptation and attempts to make improvement to the training mechanisms to obtain a more robust performance on the task of Visual Pushing from [1]. The source code of proposed method RLMU is available at https://github.com/RaiAnant/rl_with_videos.

1 Introduction

Reinforcement learning has proven to be an extremely useful tool to train robots for many applications. However, most reinforcement learning algorithms have poor sample efficiency and it is quite challenging to collect offline data. To overcome this, some recent works have tried combining imitation learning using unlabeled videos for RL training like [2, 3].

Recently, Schmeckpeper .et al [3] showed good performance on various tasks by leveraging offline human observations and online interaction. They applied an inverse model to infer actions for the human observation from domain-invariant representations of the observed images. They obtain these representations they employ an adversarial setup to train an encoder. The signal to this encoder is provided by a discriminator which is tasked to determine the source of the images it is input. To improve the performance of this domain discriminator, the authors also collect some human-robot paired data and use a L2 loss as an additional loss term. While the use of domain-invariant representations has yielded good performance, using a single shared encoder to capture all features from human observations to RL observations makes convergence challenging. Besides, using pair data requires humans to label a new dataset for each task, which is not a scalable solution for reinforcement learning.

To allow agents to learn from human motions, we propose Reinforcement Learning with Motion Understanding and Data Augmentation (RLMU) that incorporates RLV[3], with temporal relation understanding and negative data augmentation. We will introduce two sub-structures in this paper: a Motion Extractor Module (MEM) and a Video Data Generator to implement these ideas. MEM is a pre-trained self-supervised motion understanding network that helps provide additional signal to the encoder for domain adaption, while the Video Data Generator focuses on amplifying the number of samples used during the RL training. The experiment on pushing tasks proves that our architecture with Motion Extractor Module and Video Data Generator is more robust than [3].

The next section reviews several related works on Cross-Domain Reinforcement Learning, Temporal Relation, and data augmentation in RL. Section 3 provides a detailed description of the proposed model: Reinforcement Learning with Motion Understanding (RLMU). Section 4 presents and compares the experimental results of our RLMU with the baseline RLV and the vanilla RL algorithm (without making use of observation videos) used i.e. Soft Actor Critic (SAC). Our conclusions and

Preprint. Under review.

summaries are provided in the final section 5. Note that across this paper we use the terms human videos and observation space interchangeably, as we do agent or robot environment and interaction environment.

2 Related Work

2.1 Cross-Domain Reinforcement Learning

Prior to RLV, several works [4, 5, 6] have employed imitation learning to learn optimal or sub-optimal policies with the help of an expert. However it is quite common for the domain in which the expert operates to be quite different from the one where the agent is trained and deployed. On approach to overcome such domain shifts between observation and interaction environments is to use Domain Adaptation. RLV [3] uses adversarial domain confusion to accomplish this to perform reinforcement learning using human videos. This approach was originally proposed by Tzeng et al.[7]. Other adversarial setups like generative adversarial imitation learning[8], and adaptive adversarial imitation learning[9] have also proven useful for a variety of cross-domain reinforcement learning and imitation learning tasks. These algorithms have been widely used to learn similar representations for different robots having similar behaviors[10].

2.2 Temporal Relation in Observation Space

Temporal relation using videos is widely studied by the computer vision community [11, 12]. Owing to the ambiguity in describing activities at appropriate timescales, it remains a challenging job till date. Before the rise of deep learning, the general approach to this problem included using optical flow to evaluate the scene, or analysis of human poses across video frames[13]. The emergence of CNN[12], LSTM[14], and Transformer[15] architectures have provided significant breakthroughs in this field. However, existing methods still find it difficult to reason about temporal relations in a few-shot learning scenario. In this report, we apply a LSTM-based architecture to pre-train an encoder for human motion understanding.

2.3 Data Augmentation in Reinforcement Learning

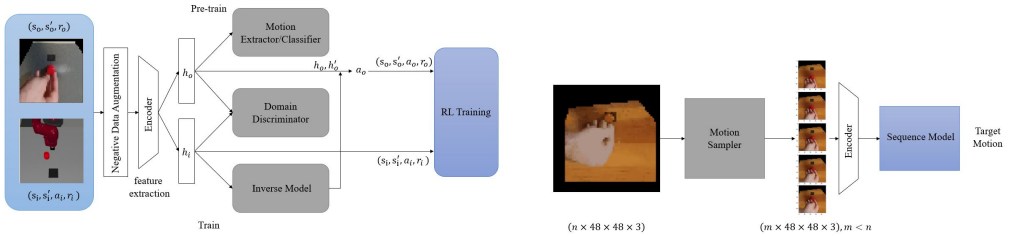
Both DrQ[16, 17] and RAD[18] show that RL algorithms benefit significantly from data augmentation. Their results show that data augmentation can significantly improve efficiency and generalization of most of RL algorithms. Since data augmentation is usually agnostic to the baseline algorithm, any framework can be adapted to incorporate it without much engineering. In our project, we try to explore if sequence data augmentation can help improve the performance of RLV and RLMU.

3 Architecture

Fig.1(left) illustrates the overall framework of Reinforcement Learning with Motion Understanding, which is primarily based on the structure of RLV[3].Our architecture can be divided into five sub-structures: (1) A shared encoder handle domain shift between the human observations and agent observations; (2) Motion Extractor Module for pre-training the encoder; (3) Domain Discriminator to provide signal to the encoder; (4) Inverse Model for predicting action between human observations; (5) Reinforcement learning baseline model. As previously mentioned, we use the Soft actor-critic[19] algorithm as the baseline RL model.

3.1 Motion Extractor Module

Although RLV has a domain discriminator to reduce domain shift between human video and training environment, the simple structure of a single shared encoder leads to a loss in stability. During our experiments, we found that around half of the training seeds do not converge to the maximum rewards. We hypothesize that this is mostly because the encoder fails to learn meaningful encoding if the signal from the discriminator and from the Q function or policy network cannot align well. This could potentially happen when the policy is not good enough for the model to collect good observations in the interaction environment which may throw off signals that the discriminator sends to the encoder



(a) (b)

Figure 1: The proposed RLMU architecture (left) and MEM module (right). Larger figures are shown in Appendix section.

for domain adaptation. To overcome this problem, we propose using a pre-training module called Motion Extractor Module (MEM) and a new self-supervised auxiliary task.

In MEM, we introduce a motion classifier based on a LSTM architecture for pre-training the shared encoder. Fig.1(right) shows the detailed structure of the Motion Extractor Module. MEM takes a fixed size re-sampled sub-sequence from the human videos as input. First each frame of the input is sent to the shared encoder and converted to a sequence of encoded states. The encoded states are then passed to a 3 layer LSTM motion analysis model to perform sequential position predictions. By learning to predict positions of the sub-sequences in the video, the MEM can help the encoder capture relative high-level features at the pre-training stage. Besides, we can use other relevant human video motion datasets to pre-train the encoder. This self-supervised auxiliary task borrows its intuition from Time-Contrastive Networks[20] and Shuffle and Learn[21]. In the future, this model can be combined with sequential domain prediction and can extend to RL training frameworks[22].

3.2 Data Augmentation: Video Data Generator

We try to use data augmentation techniques to improve the stability of our framework. The observation space involves all transitions that contribute towards a positive direction of motion. This does not fully represent the states the agent sees in the interaction environment. To overcome this we generate some "negative" augmentations of the observation space videos. First we try to generate videos with frames in reversed order and add them to the dataset. We find that using this augmentation does not impact the performance of the baseline model. Based on observing the agent in the interaction environment during its initial training phases, we came up with a new method to augment these videos. We call this technique "negative jitter" augmentation. The technique is described as following,

- Set a jitter value, j (we used j = 2,5,7 or 10 frames for our experiments)
- Pick a frame from the total set of frames in the original video as a start frame (we select a frame that lies between the first 10-25
- After copying frames prior to the selected start frame as it is, repeatedly copy the next j frames in original and reversed order for the remaining length of the video.

The technique proposed generates videos that mimic the initial phases of training the agent where it just moves back and forth at various positions and does not progress towards the target location. Using this technique showed us improvements over the base line, however these models were quite unstable. Our assumption is that this happened due to the increase in the number of samples without any change to the number of samples actually having a large reward. Thus, we scaled down the number of negative augmentations we add by half and this led to an improvement over the baseline as we show in figure 2.

The reward scheme used was to have a large reward at the final transition and small rewards for every other step. We tried to modify this by adding a small negative reward for all transitions that happen during the reversed frame ordering portion of our jitter. The intuition is that reversed order in these videos would mean motion in a direction opposite to the target and hence must be penalized. However, we fail to obtain convergence using this scheme. We hypothesize that this is due to the difference in the rewards distribution introduced between the observation and interaction environments. Using a

large enough batch size might be one way to resolve such an issue, but we were are yet to try such experiments.

3.3 Joint Optimization

The original RLV has 3 losses which are jointly optimized. One is the domain-adaptation loss (Eq 3) which consists of the paired-loss (Eq 2) and the discriminator-loss. The rest are the action prediction loss (Eq 1) and loss incurred by the RL algorithm.

$$\mathcal{L}_a(\mathbf{a}_{int}, \mathbf{h}_{int}, \mathbf{h}'_{int}, \theta) = \left\| \mathbf{a}_{int} - f_{inv}\left(\mathbf{h}_{int}, \mathbf{h}'_{int}; \theta\right) \right\|^2 \tag{1}$$

$$\mathcal{L}_{pair}(\mathbf{s}_{obs,pair}, \mathbf{s}_{int,pair}, \psi) = \left\| f_{enc}(\mathbf{s}_{int,pair}; \psi) - f_{enc}(\mathbf{s}_{obs,pair}; \psi) \right\|^2 \tag{2}$$

$$\mathcal{L}_{da}(\mathbf{s}_{int}, \mathbf{s}_{obs}, \mathbf{s}_{int,pair}, \mathbf{s}_{obs,pair}, \psi, \phi) = \mathcal{L}_{discrim}(\mathbf{s}_{int}, \mathbf{s}_{obs}, \psi, \phi) + \mathcal{L}_{pair}(\mathbf{s}_{obs,pair}, \mathbf{s}_{int,pair}, \psi) \tag{3}$$

The RLV framework optimizes all the losses together during the training curriculum. When combined, the joint optimization looks like the following:

$$\min_{\psi,\theta} \sum_{\substack{\{\mathbf{s}_{obs}, \mathbf{s}'_{obs}\} \in \mathcal{D}_{obs}, \\ \{\mathbf{s}_{int}, \mathbf{a}_{int}, \mathbf{s}'_{int}\} \in \mathcal{D}_{int}, \\ \{\mathbf{s}_{int,pair}, \mathbf{s}_{obs,pair}\} \in \mathcal{D}_{pair}}} c_2 \mathcal{L}_{\text{RL}} + c_1 \mathcal{L}_a(\mathbf{a}_{int}, f_{enc}(\mathbf{s}_{int}; \psi), f_{enc}(\mathbf{s}'_{int}; \psi), \theta) + c_3 \max_{\phi} \mathcal{L}_{da}(\mathbf{s}_{int}, \mathbf{s}_{obs}, \mathbf{s}_{int,pair}, \mathbf{s}_{obs,pair}, \psi, \phi) \tag{4}$$

However we suggest to remove the paired-loss from the domain adaption part. Instead, to improve the performance without having the paired loss, we suggest adding the MEM loss (Eq 6).

$$\mathcal{L}_{da}(\mathbf{s}_{int}, \mathbf{s}_{obs}, \psi, \phi) = \mathcal{L}_{discrim}(\mathbf{s}_{int}, \mathbf{s}_{obs}, \psi, \phi) \tag{5}$$

$$\mathcal{L}_{mem}(\mathbf{s}_{obs}[m : m+k], \beta, \psi) = -\sum_{i=1}^{N} C_i log f_{mem}(f_{enc}(\mathbf{s}_{obs}[m : m+k]; \psi); \beta) \tag{6}$$

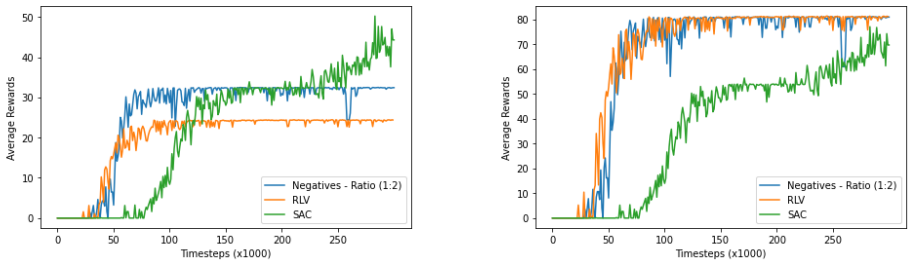
We jointly optimize the domain adaptation loss, inverse model loss and MEM loss, and the optimization objective of the chosen baseline algorithm, according to the following objective:

$$\min_{\psi,\theta,\beta} \sum_{\substack{\{\mathbf{s}_{obs}, \mathbf{s}'_{obs}\} \in \mathcal{D}_{obs}, \\ \{\mathbf{s}_{int}, \mathbf{a}_{int}, \mathbf{s}'_{int}\} \in \mathcal{D}_{int}, \\ \{\mathbf{s}_{obs}[m:m+k]\} \in \mathcal{D}_{obs}}} c_1 \mathcal{L}_{\text{RL}} + c_2 \mathcal{L}_a + c_3 \mathcal{L}_{mem} + c_4 \max_{\phi} \mathcal{L}_{da}(\mathbf{s}_{int}, \mathbf{s}_{obs}, \mathbf{s}_{int,pair}, \mathbf{s}_{obs,pair}, \psi, \phi) \tag{7}$$

We also perform hyper-parameter search to make the process more stable with the new architecture and data that we propose.

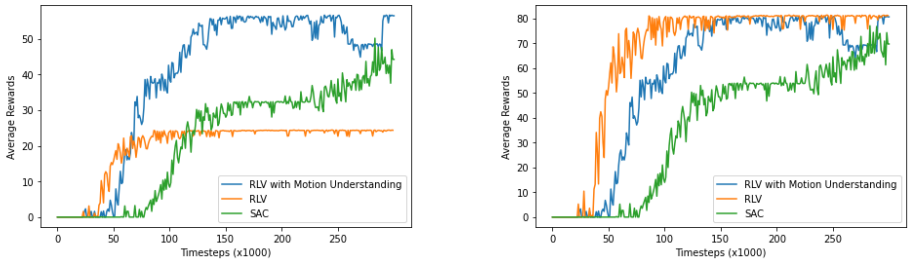
4 Experiments and Results

The environment we use for our experiments is visual pushing[1]. All models were trained on RTX 8000 and v100 GPUs on the NYU HPC Greene cluster. Each experiment was repeated using 10 different random seeds. The performance across all random seeds was averaged to generate the plots in fig.2 and fig.3. Since half of the random seeds do not converge for the RLV framework, we also draw plots similar to [3], where they only show results from seeds that converge. For each plot we also compare with the baseline RL algorithm, which is the vanilla Soft-Actor Critic algorithm (does not use sub-optimal human videos).



(a) (b)

Figure 2: Comparison of performance when using our proposed "negative jitter" augmentations with baseline RLV and SAC algorithms. (a) shows all seeds used in our experiments, (b) shows only the seeds which converged.



(a) (b)

Figure 3: Comparison of performance when using our proposed motion understanding module with baseline RLV and SAC algorithms. (a) shows all seeds used in our experiments, (b) shows only the seeds which converged.

From the fig.2, we can conclude that even though the rate of convergence on good seeds has not changed from the standard RLV performance (plot (a)), training with video data augmentation module does help in getting more consistent results when trained over a large number of seeds (plot (b)).

The performance of RLMU goes down compared to the standard RLV (fig.3) with paired loss (plot (b)), when we only consider the seeds that result in non-zero return values in training. But when comparing the performance across all the seeds (plot (a)), we see significant improvement over RLV. As RLMU does not use the additional paired data, our experiments show that RLMU is more robust than RLV with pair data. Even SAC performs much better in average than RLV in this regard.

One of the intriguing observations was that the RLV architecture has a poor convergence rate. A lot of seeds fail to produce non-zero rewards after 300 epochs of training. We trained RLV on 10 random seeds and only 5 of them provided non-zero average rewards when evaluating. The paired data helps-in providing some stability but is still not enough as is evident from our results. Our proposed approach does help alleviate some of these instability issues, but they are still not as stable as standard SAC training which obtains convergence in about 70%-80% seeds. Even though convergence of RLV is much faster than SAC, its instability makes it a very unreliable approach for now.

All experiments and results presented above are reproducible and to do so, we have shared the seeds and csv files for these mentioned results in our GitHub repo.

5 Conclusion and Discussion

Reinforcement learning with domain adaption is particularly difficult. For successful adversarial domain adaptation, a good secondary signal apart from the discriminator is required to provide a good

initialization to the encoder in at least one of the domains. However the secondary signal in RLV is provided by the RL framework, which is quite weak compared to other approaches that typically use a supervised setup alongside the discriminator for domain adaption. Hence, it is easy to see that there is a cyclic dependency which prohibits the encoder from learning good representations. The RL algorithm requires good feature encodings to have more meaningful trajectories and exploration.However, to learn good encodings for both human and robot environments, the discriminator model needs some assistance from RL to bootstrap the process. This issue is further compounded by the fact that the RLV uses a single encoder for both environments. In [7], the authors discuss how having single encoder can lead to a poorly conditioned optimization problem since the same network has to handle images from two different domains.

We try to overcome the instability by using the Motion Extractor Module to pre-train the encoder using a sequence of human observations. MEM helps breaking the cyclic dependency since it offers a more reliable signal that helps in getting a good start to learn meaningful latent state features. In the future, we plan to implement more optimization methods including (1) Using a transformer architecture for the encoder and motion extractor modules; (2) Have interpretability of the encodings; (3) Build multiple encoder structures.

References

[1] Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering, 2019.

[2] Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. *arXiv preprint arXiv:2108.05877*, 2021.

[3] Karl Schmeckpeper, Oleh Rybkin, Kostas Daniilidis, Sergey Levine, and Chelsea Finn. Reinforcement learning with videos: Combining offline observations with interaction. *arXiv preprint arXiv:2011.06507*, 2020.

[4] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[5] Stephane Ross and J Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014.

[6] Pratyusha Sharma, Lekha Mohan, Lerrel Pinto, and Abhinav Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. In *Conference on robot learning*, pages 906–915. PMLR, 2018.

[7] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017.

[8] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29:4565–4573, 2016.

[9] Yiren Lu and Jonathan Tompson. Adail: Adaptive adversarial imitation learning. *arXiv preprint arXiv:2008.12647*, 2020.

[10] Zhao-Heng Yin, Lingfeng Sun, Hengbo Ma, Masayoshi Tomizuka, and Wu-Jun Li. Cross domain robot imitation with invariant representation. *arXiv preprint arXiv:2109.05940*, 2021.

[11] De-An Huang, Vignesh Ramanathan, Dhruv Mahajan, Lorenzo Torresani, Manohar Paluri, Li Fei-Fei, and Juan Carlos Niebles. What makes a video a video: Analyzing temporal information in video understanding models and datasets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7366–7375, 2018.

[12] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 803–818, 2018.

[13] David A Forsyth and Jean Ponce. *Computer vision: a modern approach*. Pearson,, 2012.

[14] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[16] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

[17] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

[18] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020.

[19] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[20] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[21] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer, 2016.

[22] Dejing Xu, Jun Xiao, Zhou Zhao, Jian Shao, Di Xie, and Yueting Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10334–10343, 2019.

A Appendix

Optionally include extra information (complete proofs, additional experiments and plots) in the appendix. This section will often be part of the supplemental material.

Figure

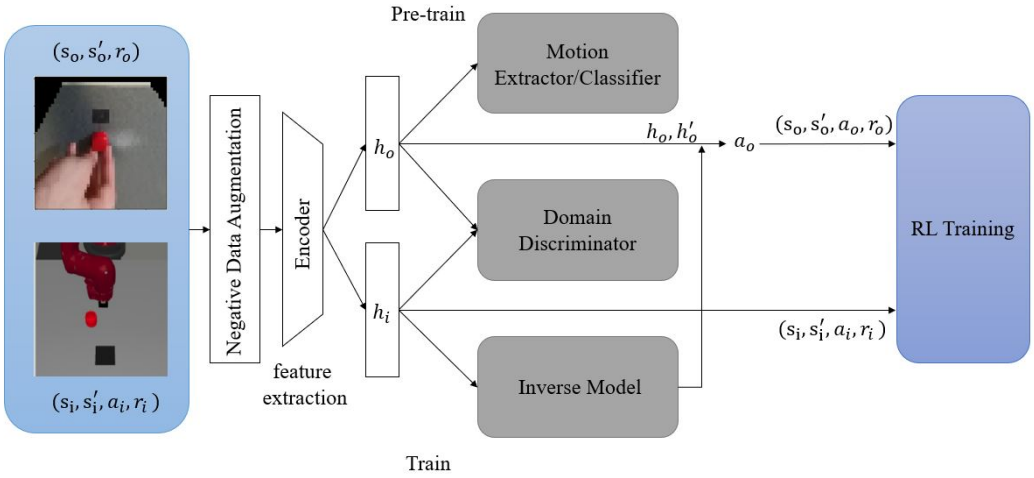


Figure 4: The proposed RLMU architecture.

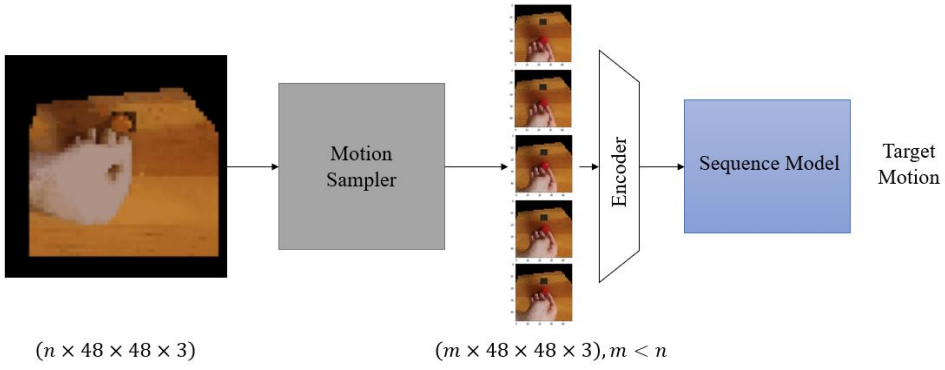


Figure 5: Motion Extractor Module.