Unsupervised Representation Learning in Atari

Shekar Ramaswamy
Brown University

Lawrence Huang
Brown University

Kendrick Tan
Brown University

Tyler Jiang
Brown University

Abstract

In this study, we reproduced the paper "Unsupervised Representation Learning in Atari" as part of the NeurIPS 2019 Reproducibility Challenge. The original paper presents ST-DIM, or Spatiotemporal Deep Infomax, which is an unsupervised method for learning useful state representations. ST-DIM attempts to learn an encoding that maximizes shared information between sequential game frames. The authors propose that this method allows for more effective learning of meaningful environmental features, such as the game state variables of Atari games. We review the authors' goals, claims, and results. We also discuss our process, challenges encountered, and results of our work.

1 Introduction

The motivation behind the paper comes from the desire to learn representations that can capture high-level features in the environment in an unsupervised manner, which the authors claim will:

1. Enable agents to transfer knowledge between tasks.
2. Enable learning with fewer interactions.

ST-DIM is designed to do well on this task since it ignores low-level details by maximizing "predictive information," or the mutual information between consecutive states. The authors proposed a benchmark based on the degree to which underlying high-level features of the environment can be recovered from a linear transformation of the learned representation. For each state variable in each game, a linear probe is trained to predict the ground truth value of that variable. Then, the representation is evaluated based on how accurately the probes predict those values (via mean F1 score). The authors showed that ST-DIM performed better at this task than a suite of other unsupervised state representation learning methods.

We discuss our attempt at reproducing this paper, challenges faced along the way, and the results we recorded. We will also reflect on the main arguments of the paper and clarify implementation details that were not fully addressed.

2 Related Work

ST-DIM builds on the main ideas explained in the DIM (Deep InfoMax) paper. DIM is an unsupervised technique for producing useful representations. DIM is a method that both estimates and maximizes mutual information between input data and learned representations. It summarizes the features of an image at a global and local scale. Then, it trains a discriminator to maximize mutual information between a given image's global representation and local patches. This learns

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

representations that encode relevant local features as well as the overall content of the image. It is also able to prioritize either global or local information, depending on what the task desires.

ST-DIM builds upon this method by incorporating both a spatial and temporal element to DIM, which the authors argue are crucial elements of learning high-level features of a representation. Understanding the idea of mutual information as well as how DIM was structured and implemented was necessary for reproducing ST-DIM. The DIM paper turned out to be one of the main resources we used for understanding the ST-DIM paper and how to reproduce it. It also gave us a better understanding of the authors' motivation behind incorporating both spatial and temporal elements to DIM.

The authors chose to use InfoNCE as the mutual information estimator for ST-DIM. It was shown in the DIM paper that InfoNCE experimentally outperformed other mutual information estimators on large state spaces, such as those found in Atari 2600 games. Understanding how InfoNCE was related to ST-DIM was crucial to our reproduction.

3 Objectives

Our main goal while attempting this reproducibility challenge was to evaluate whether the paper's ST-DIM representation is better able to predict game state variables through a linear transformation when compared to other unsupervised methods. In particular, we focused on replicating Tables 2 and 3, which outlined the Probe F1 scores across categories for various games, with data collected by random agents.

We also discuss whether their proposed benchmark is more effective at (1) transferring knowledge between tasks (in the same environment) and (2) learning with fewer interactions.

4 Experimental Methodology

In this study, we focused primarily on evaluating whether ST-DIM was more effective than a Random-CNN in Pong. Initially, we intended to replicate more methods across more games, however, we made the decision to stick to Pong after considering our limited time and compute resources. We address this tradeoff in greater detail in the "Reproducibility cost" and "Reflections" sections.

Our overall experience replicating Table 2 was mixed. The paper was very helpful in providing architecture and hyperparameter details, especially with respect to the encoder and probe. This removed a key pain point in the replication, which we found valuable working on a limited time frame. However, we found that the main contribution, ST-DIM, could have been explained better in that it was not particularly clear how it was derived from the original DIM. We go into more detail in the dedicated ST-DIM section.

Our framework of choice was PyTorch.

4.1 Data collection

To collect data, we set up OpenAI gym environments to simulate the Atari games. We followed the data preprocessing procedures as laid out in Table 5 of the ST-DIM paper, using the corresponding OpenAI baselines wrappers to accomplish grayscale conversion, no-op resets, etc. We then utilized the authors' RAM labeling dictionaries from the Atari Annotated RAM Interface (AtariARI) to ensure consistent data labels for each frame of data we collected. Duplicate episodes between the training and testing sets were removed, and variables with less than an entropy of 0.6 were excluded from the linear probing task. Where specified, we collected the same number of frames for encoder training and linear probing. However, it was not mentioned in the original paper how the frames for encoder training were split into training, validation, and testing splits.

4.2 Base encoder architecture

The base encoder was a CNN architecture used as a basis for the Random-CNN and ST-DIM methods. The architecture for this encoder was laid out clearly in Figure 4 of Appendix C in the original paper. The only issue we ran into was determining the number of output channels for each layer, which happened to be the number above the top right corner of each layer in the diagram. We note that it may be worth adjusting the diagram to make it clearer that those numbers refer to the number of output channels.

For the Random-CNN task, we initialized the CNN with random weights and froze them by ensuring that the losses do not backpropagate through the CNN. This was done through PyTorch's `torch.no_grad()` functionality.

4.3 Encoder trained with ST-DIM

ST-DIM is a flexible method that can be applied to different encoders. In this case the authors utilized the same CNN structure as the base encoder. Therefore, we were able to reuse the same code for the base encoder and write ST-DIM on top of it as a training method. The main challenges during implementation were creating the batches of consecutive frame pairs and converting the InfoNCE loss formulas from equations (2) and (3) in the original paper into code. The pairs in each batch had to be nonconsecutive because it was assumed that each image in the was only consecutive to other image paired with it. Once the InfoNCE loss was calculated, it was backpropogated to adjust the weights of the base CNN encoder. The bilinear layer for the score function was implemented by applying a linear layer to one feature vector and matrix multiplying the result with the other feature vector.

4.4 Probe

The probe consisted of a linear layer that took in the encoding output as input and outputted a value ranging from 0 to 255, as each state variable is represented as a byte. There is one probe for each state variable. Replicating the probe was straightforward, given that it consisted of the singular linear layer in the unsupervised case and the base encoder plus the same linear layer in the supervised case.

4.5 Reproducibility cost

The authors mentioned that they used a computational cluster with multiple V100 and P100 GPU's. Each of their machines had 8 cores to collect data in parallel. For our experiments, we only had one K80 GPU on a Google Cloud Platform (GCP) instance and our personal laptops. Data collection and training was also done in series. It took between 6 and 8 hours to collect data, train an ST-DIM encoder, train a linear probe, and collect testing results using our GCP instance. For simpler methods such as the Random CNN and supervised linear probe, we found that training on just our personal laptops was feasible. We did not have access to the same level of computational resources, which greatly limited our ability to review the many methods and games covered in the original work.

5 Results

We replicated three representation learning methods for the Atari game Pong. To collect our results, we calculated the F1 score and accuracy for our linear probe's predictions of each state variable. We then averaged across the ground state variable categories as defined by the authors' benchmark. The F1 score was used as a heuristic to determine how well ground truth variables could be recovered from the representation through a linear transformation.

Table 1: Average F1 scores of representation learning method per category for Pong (according to the proposed benchmark).

Category	Random-CNN	ST-DIM	Supervised
Small location	**0.66**	0.60	0.79
Agent location	**0.49**	0.43	0.44
Other location	**0.60**	0.58	0.70
Score/clock/lives/display	0.88	**0.98**	0.52
	0.66	**0.65**	0.61

Table 2: Average accuracy of each representation learning method per category for Pong (according to the proposed benchmark).

Category	Random-CNN	ST-DIM	Supervised
Small location	**0.64**	0.58	0.76
Agent location	**0.46**	0.41	0.40
Other location	**0.57**	**0.56**	0.69
Score/clock/lives/display	0.87	**0.98**	0.48
	0.64	**0.63**	0.58

6 Analysis and Discussion of Findings

Overall, our results were inconclusive as to whether ST-DIM is better than Random-CNN at predicting Pong's state variables. Due to the length of training time and our limited compute resources, we were only able to complete a single run on each of the Random-CNN and ST-DIM. We were also only able to complete two supervised encoder runs. As a result, we focus our discussion on the results we achieved, how we collected them, and the issues which could have arisen.

6.1 F1 Scores

Rather surprisingly, the Random-CNN in our experiments performed significantly better than the Random-CNN trained on Pong in the paper. More specifically, our linear probe trained on the Random-CNN's representation achieved an average F1 score of 0.66, while the one in the paper was 0.17 (Table 2). It was also surprising how well the Random-CNN and ST-DIM performed on the "Score/clock/lives/display" category, with ST-DIM achieving an accuracy of 0.98 and Random-CNN achieving an accuracy of 0.87.

The high accuracy of the Random-CNN likely stems from our data collection step, since the data collected was solely from a random agent. This could have resulted in test sets which are generally uniform and easy to predict representations for. It is possible that the authors performed additional pruning steps that made their data less uniform than the the data we collected, which caused their scores to be lower.

6.2 Supervised Learning Graph

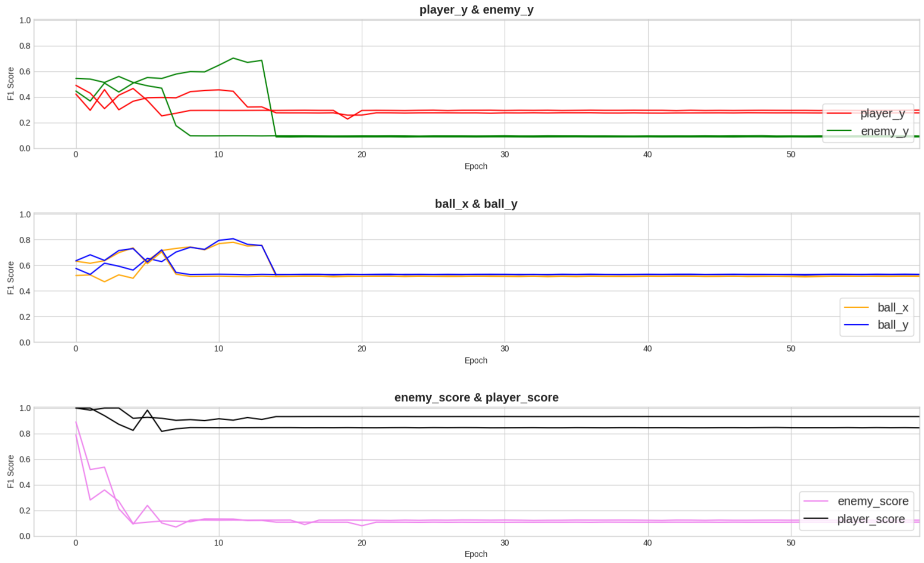

Figure 1: Validation F1 scores for supervised learning on Pong (2 separate runs).

As seen from the graph, the F1 scores for each category all experience a sharp decline after 5 or so epochs. This result was very strange to us, as it seemed likely that the fully supervised approach would at least outperform the Random CNN encoder. Although we would want to perform additional experiments before arriving at a conclusion, one explanation for this behavior is that Pong was simple enough that after a few epochs, our network grossly overfit to the training data and subsequently drove our validation F1 scores down.

7 Reflections

In this section, we discuss in detail our experiences working with the data, ST-DIM, and the probe. In particular, we outline areas which we feel that could be improved and addressed, and provide some additional ideas that may strengthen the paper.

7.1 Data verification

For the extraction of RAM state variables from the Arcade Learning Environment (ALE), we decided to use the authors' code, since they had already manually identified state variables from the RAM state. To ensure that the authors were extracting the correct information from the RAM state, we decided to look over the data to see if the labels in their code matched the data that was being extracted. We observed a few labels which seemed inaccurate given the observed game state.

For example, in *Montezuma's Revenge*, there was a variable "player direction," which should take on one of two values, left and right. We noticed, however, when printing the state variables and observing the game, that the player direction variable took on at least six values.

When examining the state variables in *Space Invaders*, we found that the "number of lives" variable did not reflect the number of lives the agent had at all, since it remained the same when an agent died. Sometimes the variable would even increase throughout the course of an episode. Similarly in *Breakout*, there were 30 different labels named 'block_bit_size', none of which differed in value at any point. Here, we saw 30 variables that were included as state variables, but were filtered out by the entropy threshold after every episode, rendering them useless for any result.

What we felt about these observations, although having minimal effect on our results, was that perhaps the authors did not label all key state variables correctly. We recommend that the authors talk more explicitly about how and why they chose the different state variables.

7.2 ST-DIM

7.2.1 Paper clarifications

The first challenge we encountered while implementing ST-DIM was generating data batches. Section 2.1 of the original paper explained that the batches had to contain pairs of consecutive frames, with each pair considered a "positive" example and images from different pairs (i.e. nonconsecutive frames) as "negative" examples. The significance of this was not clear to us until we reviewed the DIM paper, as the concepts of "negative" and "positive" examples are more thoroughly reviewed in that work. Referring the reader specifically to the paper in this section would have clarified the intention behind this batching choice.

Additionally, when extracting local features for the local-local InfoNCE loss, the authors mentioned that they chose to extract an intermediate layer of the base encoder. It was specified that each local patch was $1/16^{th}$ of the total observation size. Given that the base encoder architecture was laid out clearly in Appendix C, simply marking which layer on the diagram was used for the local patches would have been a more straightforward explanation of this implementation detail.

7.2.2 Necessary background clarification

Figure 2 of the original paper was also intended as a summary of the ST-DIM training process, but we found the diagram was too removed from the implementation to be specifically helpful. The use of the arrows to indicate which patches were compared was not particularly helpful. Instead, we found figure 3 from the DIM paper to be much more helpful for re-implementation. This figure clearly solidified the difference between global infomax and local infomax by showing convolution diagrams and describing how the global and local representations were extracted. Once we became familiar with the concepts from DIM, it became much more clear how ST-DIM was built on top of DIM by sampling subsequent frames and using those as the positive and negative examples explained in the DIM paper.

The paper also required an understanding of mutual information, which we were not very familiar with. We reviewed the CPC paper to better understand the InfoNCE loss. However, a version of the paper that was later uploaded to arXiv included a more detailed explanation of how this loss was specifically calculated in ST-DIM. This concrete formula and explanation greatly helped us translate the abstract concept of "mutual information" into a specific implementation.

Overall, it was essential for us to understand the DIM paper and the concept of mutual information before beginning our replication. We were initially unclear about the depth of prior knowledge required, thus we would recommend that the paper be more explicit about what background is needed.

7.3 Probe

7.3.1 Paper clarifications

While the probe in general was straightforward to implement, the authors do mention that they used a LR scheduler and early stopping. Following the general theme of the paper in that it provides many of the hyperparameters, it would have been beneficial to also have access to both the scheduler and early stopping parameters. However, we acknowledge that this is a small point and is highly unlikely to have significantly altered the final result.

7.3.2 Potential ways to improve probe accuracy

We propose two potential ideas for an ablation study that would have potential to improve the probe F1 scores, although at the cost of compute and implementation time. First, it has been shown that even adding an additional layer of neurons, i..e one fully connected hidden layer, can greatly increase performance of a network. It may have been a good idea to try a more complex probe structure on the same task and compare results.

Secondly, although the probe has potential to predict values from 0 to 255, we noticed that many of the state variables can ever possibly take on a few of these values in this range. It would be interesting to see if probes could be constructed on a per state variable basis, so that their ranges were limited exactly to the total number of forms a given variable could take. Note that this assumes knowledge of the range beforehand. However, such a tight bound may have increased accuracy of the probe prediction.

7.4 Accuracy Calculations

7.4.1 Multiclass averaging

With respect to table 2, the author's method of weighted multiclass averaging seems to have disproportionately accounted for the accuracy of larger, easier to predict objects such as score while discounting categories that would be more likely to have a lower accuracy. For example, score is likely to be easier to predict than small localization, and there are likely to be many more small localizations. However, when the final accuracy is calculated, these two accuracies are weighted equally, which brings up the overall average.

For example, if there was just one score object to predict, and it was predicted with 90% accuracy, and 5 small localizations on average were predicted with 50%, the accuracy under the authors method would be 70%, while 57% would be more representative. We are not arguing for a complete per object accuracy metric, but we felt that this should be addressed and improved with a better class accuracy weighting scheme.

7.4.2 Random vs. PPO agent

In the original paper's data, we noticed that in some cases ST-DIM trained on the data collected by the random agent performed better than ST-DIM trained on the data collected by the PPO agent. This is a discrepancy worth diving into, as it raises a few key questions. In particular, is it that the data collected by the random agent is more uniform than that of the PPO agent, so that the learned representation is better? For cases like Breakout, the F1 score for the random agent is over 0.2 greater than the F1 score for the PPO agent. It is likely that the frames collected by the random agent look very similar, as the game was probably lost in the first few seconds. In which case, the results obtained in Table 6 are more likely to be representative of a full game. We argue that more importance should have been placed on investigating the differences between these data collection approaches.

7.5 Reflecting on the benchmark

As mentioned earlier, the authors chose a benchmark based on a representation's ability to capture ground truth state variables. The paper asserts that performance on this task is useful way to measure the usefulness of representations because the ability to represent state variables is helpful for knowledge transfer. This was a claim that we believe needs further supporting evidence. To truly determine the value of the benchmark for learning, agents playing Atari games could be built on top of the representation techniques measured in the ST-DIM paper. Comparing the performance of agents using ST-DIM's learned representations against the performance of agents using other representation learners would be a more direct measure of each method's usefulness for learning. These results could then be compared against the original benchmark to determine if there is a correlation.

8 Conclusion

ST-DIM is an intuitive solution that has a chance to greatly improve state of the art representations not only in the Atari domain. Maximizing mutual information across time frames is an intuitive idea and provides useful input to a visual encoding system. However, we believe that its full potential is not yet fully explored with this paper, and we have commented on this throughout our analysis of ST-DIM.

Overall, due to time and computational resource limitations, we were only able to focus on the important pieces of the original results. Our own obtained results were inconclusive overall, since we would require many repeated runs to observe a definitive trend. Our main contributions are identifying points of ambiguity in background knowledge required and architecture details, as well as providing our own commentary on their benchmark and their method of calculating F1 and accuracy metrics.

References

[1] Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. Unsupervised state representation learning in Atari. arXiv preprint arXiv:1906.08226, 2019.