Is Self-Supervision Helpful? Exploring Self-Supervision in Reinforcement Learning and Classification

Prabhat Agarwal, Abhishek Sinha, Kumar Ayush
{prabhat8, a7b23, kayush}@cs.stanford.edu

I. Introduction

Human labels are expensive to collect and hard to scale up. To this end, there has been an increasing research interest to investigate learning representations from unlabeled data. In particular, the image itself already contains structural information which can be utilized. Self-supervised learning approaches attempt to capture this. Self-supervised tasks such as rotation task loss, jigsaw puzzle task loss as auxiliary loss functions have shown to improve classification performance across a range of classification datasets. In this work, we aim to explore the benefits of such self-supervision in two classes of tasks: (a) off-policy reinforcement learning, (b) classification (e.g. detection of forgery in manipulated facial images).

II. Task Definition

Our goal is to explore the effects of self-supervision in classification and off-policy reinforcement learning. To do this, in addition to supervised losses, we consider self-supervised losses based on labeled data $x \rightarrow (\hat{x}, \hat{y})$ that can be systematically derived from inputs x alone. We aim to consider self-supervised tasks like rotation task, jigsaw puzzle task, colorization task or sequence prediction task in our work. The jigsaw task rearranges the input image and uses the index of the permutation as the target label, while the rotation task uses the angle of the rotated image as the target label. A separate function h is used to predict these labels from the shared feature backbone f. Our final loss function will combine the two losses and thus the self-supervised losses act as a data-dependent regularizer for representation learning.

We incorporate some of the aforementioned self-supervision tasks in (a) off-policy reinforcement learning, and (b) classification. An example of self-supervision in each class of task is shown in Figure 1 and 2.

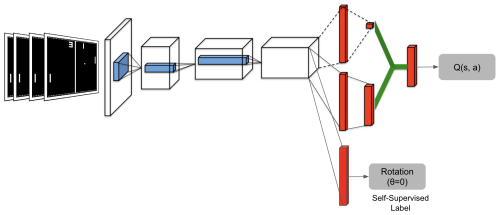


Fig. 1: Self-supervision in Pong with angle of rotation prediction.

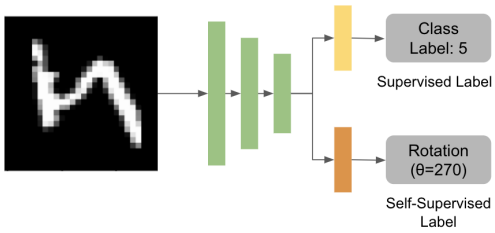


Fig. 2: Self-supervision in classification with angle of rotation prediction.

III. Related Work

There is a rich line of work on self-supervised learning. One class of methods removes part of the visual data (e.g., color information) and tasks the network with predicting what has been removed from the rest (e.g., greyscale images) in a discriminative manner [1], [2]. Su et al. [3] showed that self-supervision improves the transferability of representations for few-shot learning tasks on a range of image classification datasets. They introduced self-supervised tasks such as rotation task loss, jigsaw puzzle task loss as auxiliary loss functions.

IV. Modelling

Our framework augments standard supervised losses with those derived from self-supervised tasks to improve representation learning as seen in Figure 1 and 2.

A. Classification

In classification, we are given a training dataset $\{(x_i, y_i)\}_n^{i=1}$ consisting of pairs of images $x_i \in X$ and labels $y_i \in Y$. A feed-forward CNN $f(x)$ maps the input to an embedding space, which is then mapped to the label space using a classifier g. The overall prediction function from the input to the labels can be written as $g \circ f(x) : X \rightarrow Y$. Learning consists of estimating functions f and g that minimize a loss l on the predicted labels on the training data, along with suitable regularization R over the functions f and g, and can be written as:

$$L_s := \sum_i l(g \circ f(x_i), y_i) + R(f, g) \tag{1}$$

On tasks we assume that the test set consists of novel images from the same distribution as the training set. We use the standard cross-entropy (softmax) loss computed over posterior label distribution predicted by the model and target label.

In addition to supervised losses, we consider self-supervised losses L_{ss} based on labeled data $x \rightarrow (\hat{x}, \hat{y})$ that can be systematically derived from inputs x alone. Figure 2 shows an example of the rotation task that uses the angle of the rotated image as the target label. A separate function h is used to predict these labels from the shared feature backbone f and the self-supervised loss can be written as:

$$L_{ss} := \sum_i l(h \circ f(\hat{x_i}), \hat{y_i}) \tag{2}$$

We use rotation task loss as the self-supervised loss in our initial experiments. In this case, the input image x is rotated by an angle $\theta \in \{0°, 90°, 180°, 270°\}$ to obtain $\hat{x}$ and the target label $\hat{y}$ is the index of the angle.

Our final loss function combines the two losses:

$$L := L_s + L_{ss} \tag{3}$$

and thus the self-supervised losses act as a data-dependent regularizer for representation learning.

B. Reinforcement Learning

Reinforcement learning optimizes policies for expected cumulative reward. Reward is delayed and sparse for many tasks, making it a difficult and impoverished signal for end-to-end optimization. To augment reward, we consider self-supervised tasks to provide auxiliary losses. These losses offer instantaneous supervision for representation learning even in the absence of reward. While current results show that learning from reward alone is feasible, pure reinforcement learning methods are constrained by computational and data efficiency issues that can be remedied by auxiliary losses. We aim to explore whether self-supervised pre-training and joint optimization improve policy returns of end-to-end reinforcement learning as shown in Figure 3.

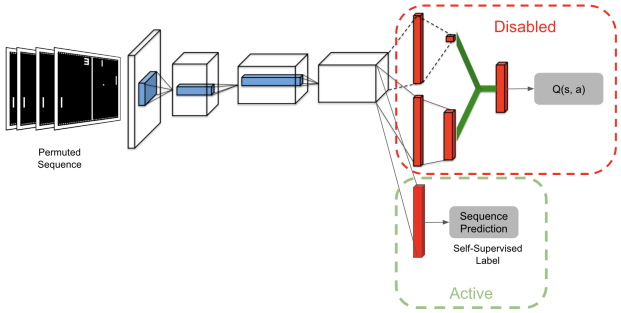
For reinforcement learning tasks, we use two kinds of self-supervision:

- *Image Based Self-Supervision (Rotation)*: Each input state in a sampled batch is rotated by an angle $\theta \in \{0°, 90°, 180°, 270°\}$ to obtain $\hat{x}$. In addition to predicting the $Q(s, a)$, the network is also tasked to predict the angle of rotation of the input state (See Figure 1).
- *Sequence Based Self-Supervision*: The input state to a DQN consists of a concatenation of four consecutive image frames. We set up an auxiliary task where sequence of the four input frames, $x = (f_1, f_2, f_3, f_4)$, is permuted using a permutation α to obtain $\hat{x}$. With input $\hat{x}$, the network is tasked to identify the class of the permutation that when applied to a valid sequence produces the input sequence. We use the following permutations with the corresponding class label:
  - Class 0: $[0, 1, 2, 3], [3, 2, 1, 0]$
  - Class 1: $[0, 2, 1, 3], [3, 1, 2, 0]$
  - Class 2: $[0, 3, 2, 1], [1, 2, 3, 0]$
  - Class 3: $[1, 3, 0, 2], [2, 0, 3, 1]$

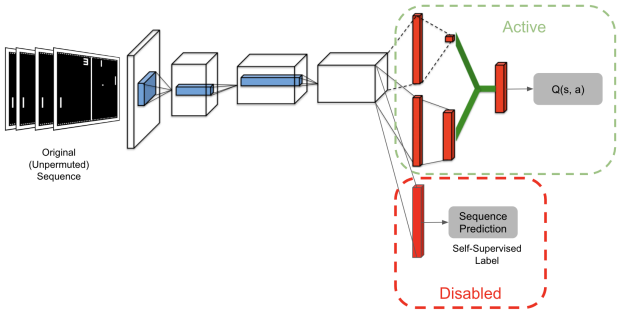
  Since $rev(x)$ and x are both valid input sequences, and $\alpha(rev(x)) = (rev(\alpha))(x)$, we group α and $rev(\alpha)$ in the same class for prediction.

  We try two variants of training with this self-supervision:

  1) **Sequence-Online**: In this case, we alternate between $Q(s, a)$ prediction and sequence prediction during training (See Figure 3).
  2) **Sequence-Pretraining**: The network is first pre-trained with the self-supervised loss. The pre-trained network is then fine-tuned over the Q-learning loss.



(a) The sequence of frames in the input state is permuted and the network is tasked to predict the correct sequence. Note that the $Q(s, a)$ prediction module is disabled.



(b) The sequence of frames in the input state is not changed during $Q(s, a)$ prediction. Note that the sequence prediction module is disabled.

Fig. 3: Self-supervision in RL using sequence prediction. We incorporate self-supervision in Dueling Deep Q-Learning algorithm. In this variant, we alternate between $Q(s, a)$ prediction and sequence prediction during training.

V. Experiments

A. Dataset and Network Architecture

Classification: We perform our preliminary experiments on MNIST. The shared backbone network consists of 2 CNN layers followed by a max-pooling layer of size 2×2 and a fully connected layer of size 128. The class prediction module is a fully connected layer of size 10 corresponding to the 10 classes. We use rotation task for self-supervision in this case and the rotation prediction module consists of a fully connected layer of size 4 corresponding to the 4 angles of rotation ($\theta \in \{0°, 90°, 180°, 270°\}$). See TABLE IV for the architectural details.

Reinforcement Learning: We perform our preliminary experiments on Pong (an Atari game). We use the open AI gym simulator for the game. The input states are four consecutive frames resized to 84×84 and converted to grayscale before being fed to the network. The action space for the game is discrete corresponding to 6 different actions. We incorporate self-supervision in state-of-the-art Dueling Deep Q-Learning algorithm to estimate $Q(s, a)$ value for a $(state, action)$ pair. Our network consists of 3 CNN layers, followed by 3 fully connected layers in the shared backbone followed by task specific layers for $Q(s, a)$ prediction and auxiliary task prediction (See Figure 1 and 3).

Model	Accuracy
MNIST	99.19
MNIST with Self-Supervision	98.45

TABLE I: Overall classification accuracy of MNIST (on test data) when the network is trained with and without self-supervision.

B. Evaluation Metric

Classification: We use classification accuracy on the test set to assess the performance of the model.

Reinforcement Learning: We use Mean Score in 100 episodes for evaluating the performance of the trained RL agent.

VI. Discussion

For MNIST classification task, the training loss and test accuracy plots are shown in Figure 5 and the maximum test accuracy is written in TABLE I. Self-supervision loss (via rotation task) does not seem to be improving the performance for MNIST classification but rather degrades the performance of the network. We posit that by adding the rotation task, we have increased the complexity. Also, since digits like 6 and 9, 2 and 5 are 180° rotations of each other, our method has introduced noise in the dataset. Hence, the classification accuracy for these classes has decreased which can be seen in TABLE II. The accuracy for classes 2, 5, 6, and 9 are worse for the model with self-supervised training via rotation task.

Class	MNIST	MNIST-Rotation
0	99.8	99.39
1	99.65	99.56
2	**99.32**	**98.93**
3	99.6	98.61
4	98.88	98.68
5	**98.99**	**97.98**
6	**98.96**	**97.7**
7	99.12	97.67
8	98.97	98.25
9	**98.51**	**97.52**

TABLE II: Class-wise accuracy on MNIST test dataset

Agent	Mean score in 100 episodes
Random	-21
DQN	19.2
Human [4]	-3
World Record [5]	21.0
Rotation	19.4
Sequence-Online	18.2
Sequence-Pretraining	19.4

TABLE III: Mean Score in 100 episodes for Pong for various agents.

Figure 4 presents the plot of mean reward with the number of training episodes for different methods in the game of Pong. TABLE III shows the best mean score in 100 episodes for different methods. We find that adding self-supervision losses results in either comparable or slightly better performance than the baseline. An interesting observation we find is that Sequence-Online method performs worse than Sequence-Pretraining method. This suggests that performing multi-task with the two losses is a bit difficult and thus further work needs to be done for it. We believe that with further tuning of hyper-parameters, we can achieve better results.

We also check whether self-supervision leads to learning of good feature representations. We plot the image attribution map corresponding to the sequence prediction branch after pre-training the network with only the self-supervised loss. To generate the attribution map, we use Smooth Grad [6], which computes map by performing squared average of the gradients with respect to the input for various noisy inputs. The attribution map is shown in Figure 6b. It can be seen that the model has learnt to identify relevant regions in the input image using just the self-supervision loss. As self-supervision loss leads to learning of good features, we believe that augmenting it with the Q-learning loss can lead to faster learning. We will investigate this further in future work.

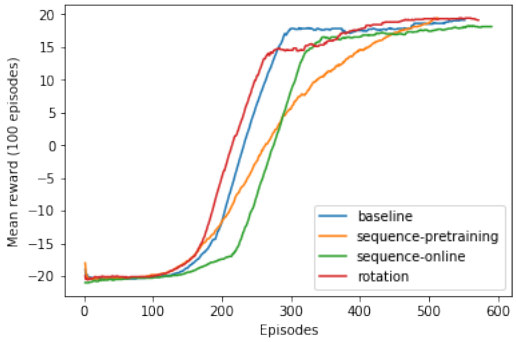


Fig. 4: Mean reward in Pong with number of episodes.

VII. Conclusion and Future Work

This project aims to augment standard supervised losses with those derived from self-supervised tasks to improve representation learning in reinforcement learning and classification. We have currently explored one image-level self-supervision technique (i.e. rotation angle prediction task) in both classification and reinforcement learning, and one sequence-based self-supervision technique (i.e predicting the permutation) in reinforcement learning. In future, we aim to explore other self-supervision techniques such as jigsaw puzzle task and image colorization, for image-based self-supervision and predicting the time of flow for sequence-based self-supervision. Our work will be a stepping stone for future research in the domain of using self-supervision in Reinforcement Learning and Classification.

REFERENCES

[1] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *European Conference on Computer Vision*, pages 577–593. Springer, 2016.

[2] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

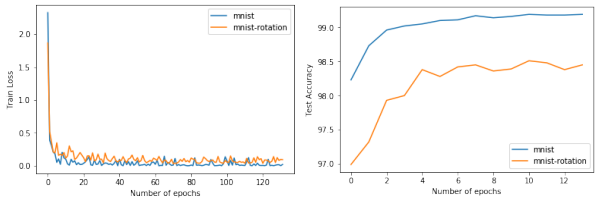
[3] Jong-Chyi Su, Subhransu Maji, and Bharath Hariharan. Boosting supervision with self-supervision for few-shot learning. *arXiv preprint arXiv:1906.07079*, 2019.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[5] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. Is deep reinforcement learning really superhuman on atari? *arXiv preprint arXiv:1908.04683*, 2019.

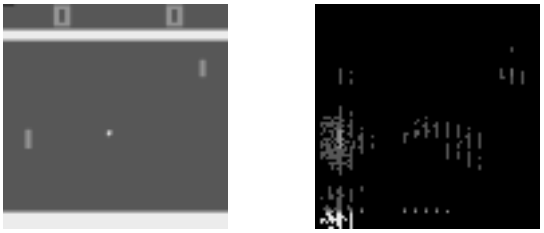
[6] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

APPENDIX



(a) Training Loss (b) Test set accuracy

Fig. 5: Training loss and Test accuracy for MNIST Classification with and without self-supervision (Rotation Task).



(a) Input frame (b) Attribution map

Fig. 6: (a) Input state, and (b) the corresponding Image attribution Map. Here the network has been pre-trained with the self-supervised loss only and Q-learning is not performed. We can see that just with self-supervised training only, the network is able to learn useful features.

Layer 1: Convolution Input: 28x28; Output: 26x26x32 Kernel size: 3x3 No. of Filters: 64, stride: 1 Padding: SAME Activation: Relu
Layer 2: Convolution Input: 26x26x32; Output: 24x24x64 Kernel size: 3x3 No. of Filters: 64, stride: 1 Padding: SAME Activation: Relu
Layer 3: Pooling Max Pooling, Padding: SAME Kernel size: 2 Output: 12x12x64
Layer 4: Fully Connected layer Input: 9216; Output: 128 Activation: Relu
Layer 5a (Classification): Fully Connected layer Input: 128; Output: 10
Layer 5b (Rotation Prediction): Fully Connected layer Input: 128; Output: 4

TABLE IV: Network architecture for MNIST Classification