MASTERING ATARI WITH BARLOW TWINS

Youngwoong Cho, Ha Young (Rosemary) Cho
The Cooper Union for the Advancement of Science and Art
New York
{cho4, cho10}cooper@edu

ABSTRACT

Reinforcement learning has achieved remarkable successes in many applications. However, the absence of constraints regarding the hidden states incurs less efficiency during the training phase. Recently, there have been several attempts to incorporate the self-supervised learning methods to enhance the learning capability of the model-based reinforcement learning algorithm. However, most of the self-supervised representation learning methods require a large batch size, which becomes more challenging as the size of the visual domain increases. In this work we present the *BarlowZero* algorithm which, by integrating the self-supervised learning algorithm via redundancy reduction, achieves the superhuman performance in visually complex and challenging domains with a reasonably smaller batch size. We evaluate our model on *Breakout*, and the model outscores the model-based reinforcement learning model without self-supervised framework.

***K*eywords** Reinforcement learning · Self-supervised learning · Redundancy reduction

1 Introduction

Model-based reinforcement learning has achieved remarkable successes in many visually challenging applications. One of the most notable achievement was made by *MuZero*, where the agent is able to learn without any prior knowledge of the underlying dynamics. However, the algorithm of *MuZero* does not pose any explicit constraints on the way it encodes the raw observations to the hidden states. This frequently hinders the model from learning the targets efficiently.

Recently, several attempts were made to account for this issue via integrating the self-supervised learning method for the representation learning. However, most of the works require large batch size, since the self-supervised learning techniques require large batch size in order to perform better. This occasionally poses a challenge to the practical applications where the GPU memory is limited.

Also, most of the self-supervised learning techniques involve asymmetric network design and a `no_grad` operation in order to prevent the model from collapsing. This further induces restriction to the selection of the hyperparameters such as learning rate or weight decay.

Therefore, we focus on the ability of the model-based reinforcement learning algorithm to be able to learn efficiently with a smaller batch size and a network that does not involve asymmetric design. In this work, we propose *BarlowZero*, a model-based reinforcement learning algorithm which can effectively learn with smaller batch size and allows flexible choices of hyperparameters.

2 Related Work

2.1 MuZero

Our work is built ontop of the *MuZero*[1] algorithm. *MuZero* is a model-based reinforcement learning algorithm that learns policy $\mathbf{p}$ based on Monte-Carlo Tree Search (MCTS) algorithm. *MuZero* algorithm consists of three models: representation function h_θ, dynamic function g_θ, and prediction function f_θ. The representation function encodes the observation o_t to the hidden state s_t, which is used by the dynamics function for the prediction of the reward r_t and the

next hidden state s_{t+1}, and by the prediction function for the prediction of the value v_t and the policy $\mathbf{p}_t$.

$$\begin{aligned} s_t &= h_\theta\left(o_t\right) \\ r_t, s_{t+1} &= g_\theta\left(s_t, a_t\right) \\ v_t, \mathbf{p}_t &= f_\theta\left(s_t\right) \end{aligned} \tag{1}$$

After *MuZero* plays against itself to generate a hypothetical trajectories $a_1, ..., a_k$ given past observations $o_1, ..., o_t$, an MCTS algorithm is utilized as a planning algorithm. The loss function is designed to minimize the error between the predicted values for reward r_t, value v_t, and policy $\mathbf{p}_t$, and the observed values for reward u_t, value z_t, and policy π_t. An L2 regularization term is also added.

$$l_t\left(\theta\right) = \sum_{k=0}^{K} l^r\left(u_{t+k}, r_t^k\right) + l^v\left(z_{t+k}, v_t^k\right) + l^p\left(\pi_{t+k}, \mathbf{p}_t^k\right) + c||\theta||^2 \tag{2}$$

Notice that although there are two distinct functions that generates the hidden state; i.e., the representation function h_θ and the dynamics function g_θ, the hidden state is not part of the loss term. This might incur inconsistency between the hidden state generated by h_θ and g_θ, which may provide noisy and inefficient training signals to the model.

2.2 EfficientZero

EfficientZero[2] algorithm attempts to ensure the consistency between the hidden states generated by f_θ and g_θ via self-supervising representation learning method. *EfficientZero* tries to minimize the difference between the output s_{t+1} from the representation function f_θ and $\hat{s}_{t+1}$ from the dynamics function g_θ by adopting the SimSiam[3] self-supervised framework. SimSiam is a self-supervised learning method where the difference between the embeddings of the two augmented views of the same image are minimized. *EfficientZero* uses the negative cosine similarity as a metric to quantify the similarity between s_{t+1} and $\hat{s}_{t+1}$.

2.3 Barlow Twins

Barlow Twins[4] is a self-supervised learning method where, similar to SimSiam framework, the difference between the two augmented views of the same image are minimized. *Barlow Twins* differs from Simsiam in a sense that it uses a empirical cross-correlation matrix as a similarity metric. *Barlow Twins* computes the empirical cross-correlation matrix of the embeddings of the two augmented views of the same image, and makes it close to the identity matrix. The loss function is as follows:

$$\mathcal{L}_{BT} \triangleq \sum_i \left(1 - \mathcal{C}_{ii}\right)^2 + \lambda \sum_i \sum_{j \neq i} \mathcal{C}_{ij}^2 \tag{3}$$

where λ is a positive constant and $\mathcal{C}$ is the cross-correlation matrix between two embeddings.

This architecture not only allows the model to learn efficiently with small batches but also results in a symmetric network design which does not require predictor network or a `no_grad` operation.

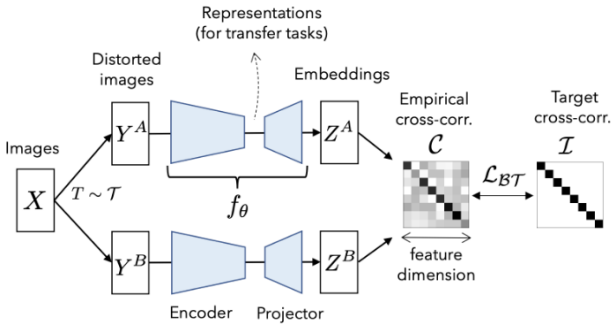

Figure 1: Barlow Twins' objective function measures the empirical cross-correlation matrix between the embeddings of the two augmented views of the same image, and tries to make this matrix close to the identity.

Table 1: Performance comparison between BarlowZero and EfficientZero.

EfficientZero		
Max. score	Mean score	Min. score
465.0	377.4	113.0
BarlowZero		
Max. score	Mean score	Min. score
427.0	351.1	274.0

3 BarlowZero

The integration of the self-supervised frameworks has increased the performance of the model-based reinforcement learning. However, most of the self-supervised learning methods, such as SimSiam[3], SimCLR[5] and BYOL[6], require large batch size – SimSiam requires batch size of 512, SimCLR 4096, and BYOL 4096. Such large batch size might not be possible in many practical applications due to the limitation of the computational resources. Also, the aforementioned self-supervised frameworks either require a `no_grad` operation, which causes additional overhead during the training phase, or adopt an asymmetric network architecture, which requires a specially designed optimizers such as LARS[7] to stabilize the training.

We propose *BarlowZero*, which utilizes the *Barlow Twins* framework for the self-supervised representation learning, that allows to train the model-based reinforcement learning model with smaller batch size and without `no_grad` operation or asymmetric network design.

Since *BarlowZero* takes advantages of *Barlow Twins*, it does not require a large batch size nor specially crafted optimizer. *Barlow Twins* can be trained with a batch size of 64, which is a reasonable size to be trained on a single GPU, and can be trained with conventional optimizers such as SGD or Adam.

4 Experiment

In this section, we aim to compare the performance of *BarlowZero* based on the 32 test runs on Breakout. Additionally, we provide the ablation tests on the loss coefficients, learning rates, projection head sizes.

4.1 Environments

Atari Breakout We primarily assess the performance of *BarlowZero* on Atari Breakout, a reinforcement learning API developed by openAI gym. Specifically, we use `BreakoutNoFrameskip-v4` environment. The observation space is given as $(210, 160, 3)$, and the action space is given as $(18,)$. Reward is given when a brick is destroyed, whose quantity varies by its color.

4.2 Results

The performance of the *BarlowZero* was compared with *EfficientZero* by testing on 32 Breakout games. The results are shown in Table 1. Although *BarlowZero* achieved the lower maximum score than *EfficientZero*, it has a greater minimum score, which evinces that *BarlowZero* shows more stable gameplay.

4.3 Ablation study

We tested the effect of various hyperparameters such as loss coefficients, learning rate, and projection head size, and optimizer.

Loss coefficients In order to assign approximately even amount of training signal to each loss, the loss coefficients are adjusted. We trained with the loss coefficients that are proposed by *EfficientZero*, and observed that the consistency loss tends to converge most rapidly, which impedes the training signals from reaching value loss, reward loss, and policy loss. This is due to the inherent nature of the cross-correlation matrix loss of having a relatively large positive value as opposed to the negative cosine loss, a similarity metric used by the *EfficientZero*, which has a range of $[-1, 1]$. Thus, reducing the consistency loss coefficient to 0.001 resulted in better training results. We further figured out that

increasing the policy loss coefficient to 10 helps the model to learn the policy better. Figure 2 demonstrates the effect of reducing the consistency loss..

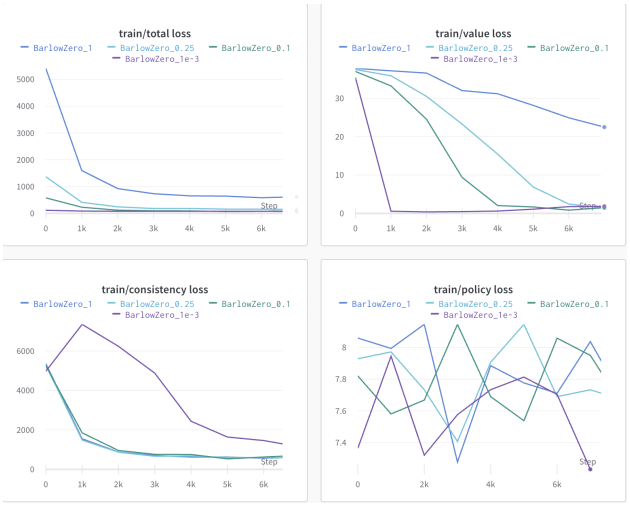

Figure 2: Effect of changing the consistency loss. The less the consistency loss, the faster the rest of the losses converge during the initial training phase. The optimal coefficient for consistency was empirically determined to be 0.001. The value loss coefficient and policy loss coefficient are set to be 1.0 and 10.0, respectively.

Learning rate SGD with different initial learning rates are tested. We tested for the initial learning rate of 0.1, 0.01, and 0.001, while keeping the consistency loss to be 0.001. It is empirically found that the initial learning rate of 0.01 was the optimal, as it can be seen from figure 3.

Also, different learning rate schedulers were tested. Specifically, a SGD optimizer with step scheduler as proposed by the *EfficientZero* ad a cosine annealing learning rate scheduler with $\eta = 0.001$ are compared. The cosine annealing learning rate scheduler was found to find the optima more precisely in the longer run. See figure 4.

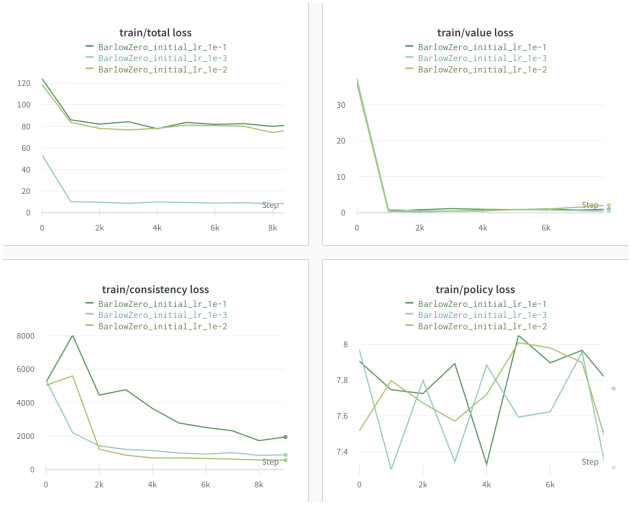

Figure 3: Effect of changing the initial learning rate. The optimal learning rate that makes all losses converge evenly and rapidly is empirically determined to be 0.01.

Projection head *Barlow Twins* empirically found that when the number of the projection head increases, the self-supervising performance of the model increases. We adopted this idea and compared the performance of the *BarlowZero* models with the projection head of 1024 to 2048. Although the training could not be completed due to the time constraint, the loss curve exhibits the possibility that the model with greater number of projection head might lead to better performance. See figure 5.

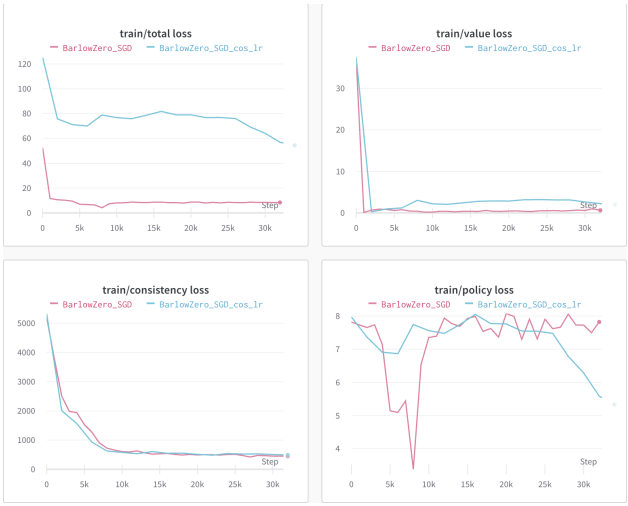

Figure 4: Effect of changing the learning rate scheduler. Cosine annealing learning rate scheduler helped the model to find the local optima more precisely than the step scheduler as proposed by *EfficientZero*, as it can be seen from the stable and rapid decrease in the policy loss. This resulted in more scoring of the Breakout game.

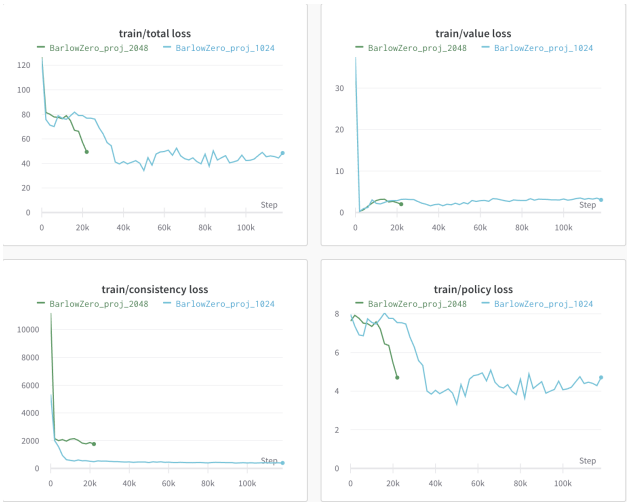

Figure 5: Effect of changing the number of projection head. Although the training was not completed due to time constraint, the model with greater number of projection head tends to have faster loss convergence.

5 Discussion

In this work, we propose a novel model-based reinforcement learning algorithm that leverages the self-supervising representation learning framework. *BarlowTwins* could achieve a performance that is better than *MuZero* and comparable to that of *EfficientZero*. Although *BarlowTwins* didn't achieve a performance distinguishable to that of *EfficientZero*, there are plenty of possibility to further improve the performance of the model, including running experiments on different types of optimizers, batch sizes, and backbone architecture. We plan to keep explore the discovery and extend the research so that *BarlowZero* can perform better than other self-supervision guided model-based reinforcement learning. We also plan to extend the environment from Breakout to other Atari games to find out the types of the game that *BarlowZero* can excel.

Acknowledgments

This research was supported by The Cooper Union for the Advancement of Science and Art. We thank Christopher Curro for valuable comments on the experiments.

References

[1] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Lillicrap, and David Timothy Silver. Mastering atari, go, chess and shogi by planning with a learned model. In *Nature*, pages 604–609, 2020.

[2] Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. *arXiv preprint arXiv:2111.00210v2*, 2021.

[3] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566v1*, 2020.

[4] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arXiv:2103.03230v3*, 2021.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709v3*, 2020.

[6] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent a new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733v3*, 2020.

[7] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888v3*, 2017.

Appendix

5.1 Model and hyperparameters

Model *BarlowZero* follows the overall network architecture of *Muzero* and *EfficientZero*. It consists of three models – namely, representation model, dynamics model, and prediction model.

The architecture of the representation model is as follows:

- 1 convolution with stride 2 and 32 output planes
- 1 residual block with 32 planes
- 1 residual downsample block with stride 2 and 64 output places
- 1 residual block with 64 planes
- Average pooling with stride 2
- 1 residual block with 64 planes
- Average pooling with stride 2
- 1 residual block with 64 planes

The kernel size is 3×3 for all layers. Also, batch normalization and ReLU activation are applied after the convolutional layer and the average pooling layer.

The architecture of the dynamics network is as follows:

- Concatenation of the input states and the actions into 65 planes
- 1 convolution with stride 2 and 64 output planes
- Residual link to add up the output and the input states
- 1 residual block with 64 planes

Batch normalization is applied after the convolutional layer, and ReLU activation is applied after the residual link.

The architecture of the value prediction model is as follows:

- 1 1×1 convolution and 16 output planes
- Flattening
- LSTM with 512 hidden dimensions

- 1 FC layer and 32 output dimensions
- 1 FC layer and 601 output dimensions

Batch normalization and ReLU activation is applied after the convolution layer, LSTM layer and the first FC layers.

The architecture of the reward and policy prediction networks are the same except for the outputs:

- 1 residual block with 64 planes
- 1 1×1 convolution and 16 output planes
- Flattening
- 1 FC layer and 32 output dimensions
- 1 FC layer and D output dimensions

Batch normalization and ReLU activation are applied after the convolution layer and the first FC layers.

The architecture of the projection layer that is applied to the hidden states are as follows:

- 3 FC layers with output dimension of 1024

First two FC layers are followed by batch normalization and ReLU activation, and the last FC layer is followed by the batch normalization layer only.

Hyperparameters In this section we list parameters that are used for the *BarlowZero* model with highest performance to our knowledge.

Table 2: Hyperparameters for BarlowZero on Atari game

Parameter	Setting
auto td steps ratio	0.3
max moves	3000
history length	400
discount	0.997^4
training steps	120000
batch size	64
num unroll steps	5
td steps	5
stacked observations	4
lstm hidden size	512
lstm horizon len	5
reward loss coeff	1
value loss coeff	0.25
policy loss coeff	10
consistency loss coeff	0.001
optimizer	SGD
scheduler	CosineAnnealingLR(training steps)
weight decay	0.0005
momentum	0.9
initial learning rate	0.01
priority prob alpha	0
priority prob beta	0.4
prioritized replay eps	1e-06
proj hid	1024
proj out	1024
pred hid	512
pred out	1024
bn mt	0.1
blocks	1
channels	64
reduced channels reward	16
reduced channels value	16
reduced channels policy	16
resnet fc reward layers	[32]
resnet fc value layers	[32]
resnet fc policy layers	[32]
observation shape	(12, 96, 96)
augmentation	['shift', 'intensity'],
revisit policy search rate	0.99