Understanding Self-Supervised Learning through BYOL: Bootstrap-your-own-latent

Zhihua Zhang, Ramya Muthukrishnan, Daniel Lee
{zhihuaz, ramyamut, laniel}@seas.upenn.edu

Abstract

In vision, self-supervised learning is an unsupervised method of learning rich feature representations from unlabeled images that can be used for improving performance on downstream tasks. In this work, we explore bootstrap-your-own-latent (BYOL), a recently developed self-supervised learning method that reduces computational expenses and produces state-of-the-art results. We implement a simplified version of the BYOL pipeline and perform experiments to study how various settings affect performance. The results suggest that (1) self-supervised pretraining is a form of regularization to address overfitting; (2) self-supervised learning relies on access to a very large unlabeled dataset; (3) the predictor is critical to BYOL's performance; (4) augmentations must be robust enough for the network to learn meaningful image features and some are much more useful for representation learning than others; and (5) the value of the target update parameter τ is critical to BYOL's success and adding a custom schedule for updating τ during training can boost performance.

1. Introduction

Self-supervised learning harnesses the power of large unlabeled datasets to pre-train models for fine-tuning on downstream tasks. In computer vision, these methods focus on pre-training models on unlabeled images to learn robust feature representations of images that are useful for tasks such as classification, object detection, and segmentation. This is an important and interesting problem because in industry, extensive labeled datasets are difficult to acquire, so unlabeled datasets have potential to significantly boost performance. Many of these methods require pre-training with certain carefully selected settings in order to boost model performance during fine-tuning. Self-supervised learning is still very new, and not much is understood about why some of these methods work. Understanding why they work is important for devising improved methods in the future. In this work, we implement an existing self-supervised approach, bootstrap-your-own-latent (BYOL) [5], and utilize experimentation to understand how BYOL works and which of its attributes are crucial to its success.

2. Related works

Most recent self-supervised approaches are either generative and discriminative. Generative methods model the distribution of the unlabeled images and their learned representations, using generative models such as autoencoders [10] and GANs [4]. However, discriminative methods tend to be less computationally expensive. Most discriminative methods fall under the category of contrastive self-supervised learning.

Contrastive Self-Supervised Learning.

In contrastive self-supervised learning, each iteration of training typically requires positive pairs (two randomly augmented versions of the original image) and negative pairs (two augmented versions of other images), the feature embeddings for which are inputs to a contrastive loss function. The goal is to train the network to learn feature embeddings that distinguish an image from other images. The intuition behind this approach is to encourage consistent representations of the same image under transformations, allowing robust features to be learned. SimCLR [3] and MoCo [6] are two well-established methods that learn features by maximizing the cosine similarity for positive pairs and minimizing the cosine similarity for negative pairs.

BYOL.

BYOL [5] is a gets rid of negative pairs for contrastive self-supervised learning, making training more computationally feasible while achieving state-of-the-art performance. The details are described in the methods section. During training, images are randomly augmented twice to form positive pairs, and BYOL simply minimizes a similarity loss between embeddings of positive pairs.

Previous work [9] has suggested that batch normalization (BN) is necessary for BYOL to perform better than random because gradient flow across batch samples acts as an implicit negative contrastive term. Removing BN can result in mode collapse, which occurs when the loss plummets to

0 because the model has learned to "cheat" by predicting the same embedding for every image, preventing useful representations from being learned.

3. Method

3.1. Dataset

We used the STL10 dataset [1] developed by Stanford that modifies the CIFAR-10 dataset [8] by keeping only 5,000 training examples labeled (500 for each class) and leaving other 100,000 training samples unlabeled. The unlabeled data comes from a similar but different distribution from the labeled data and they both have resolution 96x96.

3.2. Fine-tuning

We use ResNet18 [7] with 10 output classes. To demonstrate the effectiveness of BYOL, we trained two classifiers with the same architecture. One is a supervised-only model; the other uses the self-supervised pre-trained weights.

1. Supervised-only: we only feed labeled training data to train a ResNet18 model from scratch for 25 epochs.

2. Self-supervised and supervised: we first regard ResNet18 model as BYOL encoder and train BYOL for 75 epochs. After that, we fine-tune the BYOL encoder on labeled data for 25 epochs.

We run supervised-training for 25 epochs because we observed the loss has converged.

3.3. BYOL pre-training

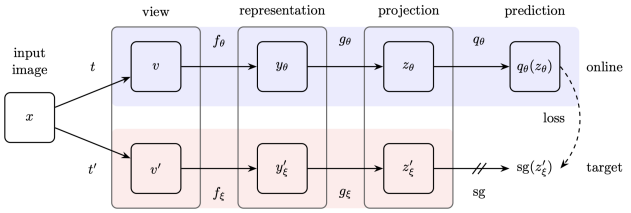

Figure 1. BYOL architecture cited from [5]. BYOL minimizes a similarity loss between $q_\theta(z_\theta)$ and $\text{sg}(z_\xi)$, where θ are the trained weights, ξ are an exponential moving average of θ and sg is stop-gradient. After training, y_θ is used as the image embedding.

BYOL wants to learn a representation y_θ that can be used for downstream tasks. In our case, we use BYOL to pre-train our encoder f_θ, which is a ResNet18 model. BYOL uses two neural networks: *online* and *target* networks. Let the online network predict the representation from the target network. Both networks have an *encoder* f and a *projector* g, but the online network has an additional *predictor* q. This difference aims at breaking the symmetry to avoid mode collapse. Specifically, the online network has parameters $f_\theta, g_\theta, q_\theta$ and the target network uses different weights f_ξ, g_ξ.

During training, given an unlabeled image $x \sim \mathcal{D}$, two image augmentations sampled from different distributions $t \sim \tau$ and $t' \sim \tau'$ are applied to generate two augmented views $v = t(x)$ and $v' = t'(x)$. The target network produces $z'_\xi = g_\xi(y'_\xi) = g_\xi(f_\xi(v'))$ as the target projection and the online network predicts it with $q_\theta(z_\theta) = q_\theta(g_\theta(y_\theta)) = q_\theta(g_\theta(f_\theta(v)))$. They are normalized $\bar{q_\theta}(z_\theta) = \frac{q_\theta(z_\theta)}{\|q_\theta(z_\theta)\|_2}, \bar{z'_\xi} = \frac{z'_\xi}{\|z'_\xi\|_2}$ before calculating the mean squared loss in (1)

$$\mathcal{L}^{BYOL}_{\theta,\xi} = \|\bar{q_\theta}(z_\theta) - \bar{z'_\xi}\|_2^2 = 2 - 2 \cdot \frac{< q_\theta(z_\theta), z'_\xi >}{\|q_\theta(z_\theta)\|_2 \|z'_\xi\|_2} \tag{1}$$

The two networks are optimized differently as shown in (2). We use typical stochastic optimization procedure to update the online network θ. The target network is not updated by gradient-based methods but using a moving average of the online network

$$\begin{aligned} \theta &\longleftarrow optimizer(\theta, \eta, \nabla_\theta \mathcal{L}_{\theta,\xi}) \\ \xi &\longleftarrow \tau\xi + (1-\tau)\theta \end{aligned} \tag{2}$$

The moving average update is the core of BYOL because the target network is not updated in the direction of $\nabla_\xi(\mathcal{L}_{\theta,\xi})$, which prevents mode collapse.

3.4. Implementation Details

Image augmentation We follow the original paper that uses the same set of image augmentations as in SimCLR [2]. First, a random patch of the image is selected with a random horizontal flip, followed by a color distortion, consisting of a random sequence of brightness, contrast, saturation, hue adjustments, and an optional grayscale conversion. Finally Gaussian blur is applied to the patches. The Kornia library was used to apply these augmentations.

Figure 2. Custom patch drop augmentations. We added augmentations where 1-3 random rectangular patches were dropped from each image

For further experimentation, we implemented our own custom patch drop augmentation in which 1-3 randomly

chosen rectangular patches are dropped (made black) for each image (see Figure 2). We also experimented with random perspective transformations, using an existing implementation in the Kornia library.

Architecture We use ResNet18 as encoders for both the online branch f_θ and the target branch f_ξ. The projectors g_θ, g_ξ and the predictor q_θ all employ a two-layer perceptron with Batch normalization and ReLU activation between layers. For simplicity, they share the same architecture that has a projection dimension of 256 and the intermediate dimension of 4096.

Optimization We use the Adam optimizer without setting special learning schedule. The learning rate is set as $1e^{-4} \times \frac{batchsize}{128}$ that scales linearly with the batch size. In addition, we use a global weight decay parameter of 10^{-6} following the original paper. For the target network, the exponential moving average parameter τ remains constant 0.999. We use a batch size of 1024 with the mixed precision training. With this setup, training takes around 3 hours over a single NVIDIA A10 GPU.

Linear Accuracy After each epoch of pre-training, we train a linear classifier on the BYOL encoded embeddings for our supervised classification dataset to assess the quality of pre-training over time.

4. Experiments

4.1. Baseline

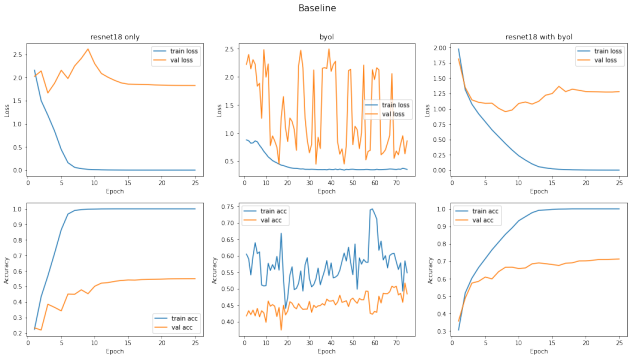

Figure 3. BYOL Baseline

We first train a baseline using the setup described in section 3.4. Supervised-only Resnet18 can achieve an accuracy of 0.551 after 25 epochs and the number increases significantly to 0.713 with BYOL pre-training (see figure 3).

The middle column in figure (3) displays BYOL results and the second row is the linear accuracy described in section 3.4. The BYOL training loss and accuracy oscillate but the validation results keep improving. We suspect the reasons to be: (a) BYOL loss is calculated on randomly augmented views, which is noisy in nature; (b) BYOL accuracy does not improve steadily because our model never sees labels during pre-training; (c) The target network keeps changing, which can be seen as a regularization tool to address overfitting.

The final BYOL accuracy is already comparable to the supervised-only model, proving its powerfulness. However, the unstable training behavior deserves further study.

	no byol	$\frac{1}{8}$ byol	$\frac{1}{4}$ byol	$\frac{1}{2}$ byol	full byol
$\frac{1}{8}$ fine-tune	0.308	0.305	0.359	0.313	0.389
$\frac{1}{4}$ fine-tune	0.390	0.401	0.411	0.482	0.590
$\frac{1}{2}$ fine-tune	0.508	0.497	0.570	0.482	0.650
full fine-tune	0.551	0.577	0.587	0.652	0.713

Table 1. **Validation accuracy** – With varying data set size.

	Supervised-only	Supervised+BYOL
Baseline	0.551	0.713
No Predictor	0.539	0.589
No BatchNorm	0.533	0.676

Table 2. **Validation accuracy** – Removing predictor makes online and target networks have the same architectures. Changes to BatchNorm only apply to projector and predictor.

4.2. Varying data set size

To better understand the effectiveness of BYOL, we vary the size of unlabeled pre-training data set and labeled supervised-training data set to see how much it can help in different situations. The results are shown in table 1 (see figure 4 in appendix for details). We can observe

1. Increasing the size of fine-tuning set always boosts performance as expected.

2. BYOL pre-training with full unlabeled data set can improve the final accuracy by approximately 30%.

However, the benefit of BYOL reduces if we decrease the number of unlabeled samples, especially when the fine-tuning data set is small. For example, using only $\frac{1}{8}$ of the unlabeled data set provides little benefit, and increasing its size does not consistently improve the final accuracy when only a few labeled samples are used.

4.3. Different architecture

We also modify the BYOL architecture to under the importance of each part. Table 2 summarizes some of them

1. Removing the predictor drastically deteriorates performance because it helps break the symmetry between online and target networks. Otherwise, they would be

Augmentation setting	Performance
No augmentations	0.520*
No augmentations on target input	0.635

Table 3. **Removing augmentations** – Removing augmentations on the inputs to 1 or both networks results in a significant decrease in performance. The latter scenario results in mode collapse and performs no better than no self-supervised learning. (* indicates mode collapse)

Augmentation type	Only augmentation	Augmentation removed
Grayscale	0.525*	0.681
Horizontal flip	0.577*	0.671
Color jitter	0.632	0.716
Gaussian blur	0.534*	0.683
Patch selection	0.700	0.623

Table 4. **Effect of existing augmentations** – Some augmentations by themselves prevent mode collapse and significantly improve performance. Removing augmentations results in a decrease in performance to varying degrees, based on the augmentation removed. (* indicates mode collapse)

forced to output the same embedding to make two randomly augmented views look similar.

2. Disabling BatchNorm will also worsen the results. But the impacts are mediocre compared to removing predictor.

The supervised-only accuracy is different across different settings because we train them in parallel to save time and the model can converge to different optimum. However, the differences are small and the comparison is still valid.

4.4. Augmentations

Pre-training was run with no augmentations at all and with only applying augmentations to the input to the target network. The supervised fine-tuning validation accuracies of these experiments are summarized in table 3. They suggest that removing all augmentations results in mode collapse (see Figure 5), since minimizing the similarity between two identical images is a trivial problem, and that removing augmentations does not result in mode collapse but results in less robust feature representations, perhaps because it is an easier problem for the algorithm to solve.

For each of the 5 existing augmentations used in the original implementation, supervised fine-tuning validation accuracy was calculated after self-supervised pre-training (1) only the one augmentation and (2) with the 4 other augmentations. The results are shown in table 4. They suggest that:

1. Random patch selections are robust enough to be used on their own and result in similar performance to using all 5 augmentations.

2. Some augmentations are not robust enough to be used on their own and thus result in mode collapse and poor fine-tuning performance.

3. Color jitter augmentations are robust enough to prevent mode collapse on their own but harm BYOL performance in the presence of other augmentations.

4. In most cases, removing augmentations decreases performance, probably because the problem becomes easier to solve, so the learned features become less robust.

Augmentation type	Only augmentation	All except patch selection	All
Patch drop	0.608*	0.649	0.525
Perspective	0.606*	0.624	0.601

Table 5. **Effect of new augmentations** – The new augmentations that we experimented with did not improve performance and were not robust enough to prevent mode collapse on their own. Combining patch drops with patch selection augmentations demonstrates how sometimes too many augmentations can be harmful. (*indicates mode collapse)

For each of the new augmentations that were not used in the original paper, we calculated fine-tuning validation accuracy for self-supervised pre-training (1) with only the one augmentation; (2) with the augmentation, plus all original augmentations except patch selection; and (3) with the augmentation, plus all original augmentations. The results are summarized in table 5. Taken together, they suggest that:

1. Both augmentations are not robust enough to prevent mode collapse on their own but still significantly improve performance on their own, for unknown reasons.

2. Too many augmentations can be harmful: when patch drops and patch selection augmentations are combined, the augmented images no longer resemble the original images, preventing useful representations from being learned.

3. Replacing patch selection with the new augmentations did not improve performance, suggesting that the existing set of augmentations is better suited to the problem than either of the introduced augmentations.

4.5. Target Network Weight Updates

We first ran experiments on different fixed values of τ, for values $0.1, 0.5, 0.9, 0.995, 0.999, 0.9999$. We show the corresponding validation accuracies in Table 6. We found that: (1) there is a big initial drop in BYOL train loss for small τ, perhaps an indication of mode collapse; (2) 0.1 and 0.5 have best BYOL train loss, but worst validation accuracies (see Figure 6); (3) there is a "sweet spot" for τ: if

τ	Validation Accuracy
0.1	0.640
0.5	0.651
0.9	0.702
0.995	**0.715**
0.999	0.713
0.9999	0.665

Table 6. **Effect of varying τ** – The optimal value for τ lies between 0.995 and 0.999.

Target Drop Schedule	Validation Accuracy
Quarter	0.701
Half	0.683

Table 7. **Effect of dropping the target network** – Dropping the target network during BYOL training does not boost validation accuracy.

τ Schedule	Validation Accuracy
Cosine	0.712
Inverse Cosine	**0.717**

Table 8. **Effect of τ scheduling** – Inverse cosine scheduling outperforms the baseline.

it is too high, training is slow, but if it is too low, proper representations are not learned.

We hypothesized that as long as we get the online network to a point where it's learning viable features, if we drop the target network altogether, we prevent collapse while perhaps speeding up training. To test this, we trained BYOL using the baseline $\tau = 0.999$, and then set it to 0 either a quarter ways through, or halfway through (see Table 7 for validation accuracy results).

We observed that: (1) there was a small spike in BYOL train loss when the target network was dropped (Figure 7) and (2) dropping the target network speeds up the reduction of BYOL loss and increase in training accuracy, but does not correspond to boosted performance in supervised learning. However, we do observe that the resulting validation accuracy is not as bad as when we started off with a low τ.

Finally, we tried out different τ update schedules. The original BYOL paper uses a cosine update schedule, from 0.996 to start up to 1 to end. But given our observations above, we hypothesized that it might be beneficial to start from a higher τ (0.999), and "reverse anneal" down (to $\tau = 0.5$) following a cosine curve (see Table 8 for results).

We concluded that: (1) inverse cosine scheduling outperforms the baseline in terms of validation accuracy; (2) fine-tuning converged much faster when pre-trained with the inverse cosine schedule; (3) the inverse cosine schedule resulted in the best linear train accuracy and the fastest increase in validation accuracy during BYOL training (see Figure 8.

5. Conclusions and Future work

5.1. Conclusions

We have shown that BYOL self-supervised learning is a powerful method to address overfitting and boost performance. Our experiments also indicate the size of the unlabeled and labeled datasets significantly affect the performance improvement, the more the better.

Another important conclusion is that the additional predictor of online network is critical to the success of BYOL because it helps break symmetry to prevent mode collapse.

The augmentations used for contrastive self-supervised learning is critical. The augmentations must be robust enough to prevent mode collapse, but must be reasonable: too many augmentations can prevent the model from learning useful image representations. Some augmentations are suited for particular datasets/supervised tasks more than others; random image patch selection is critical to performance for image classification on STL-10, while color-related augmentations are not. Furthermore, our patch dropping and perspective augmentations are not critical for this task.

The value of the exponential moving average update parameter τ is important: it must be large enough to prevent mode collapse but low enough to allow for reasonable training speed. Dropping the target network during training speeds up training but prevents the model from learning useful representations. We introduce a novel τ update schedule we call *inverse cosine*, and find that it beats the baseline in fine tuning accuracy and training speed.

5.2. Future work

Unlike other researchers, we did not observe mode collapse without batch normalization; it is not understood why this was the case and could be explored further.

We also intended to study the effects of our experiments on transfer learning. Perhaps some of the characteristics of BYOL that we observed as not critical (such as batch normalization) become critical to perform well in a completely different domain. Finally, we hope to study how different augmentations during pre-training affect performance for different downstream tasks, such as object detection.

References

[1] Honglak Lee Adam Coates, Andrew Ng. An analysis of single-layer networks in unsupervised feature learning. In *PMLR*, 2011. 2

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020. 2

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020. 1

[4] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 1

[5] 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, 2020. 1, 2

[6] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. *CoRR*, abs/1911.05722, 2019. 1

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 2

[8] Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009. 2

[9] Pierre H. Richemond, Jean-Bastien Grill, Florent Altché, Corentin Tallec, Florian Strub, Andrew Brock, Samuel Smith, Soham De, Razvan Pascanu, Bilal Piot, and Michal Valko. Byol works even without batch statistics, 2020. 1

[10] Michael Tschannen, Olivier Bachem, and Mario Lucic. Recent advances in autoencoder-based representation learning. *CoRR*, abs/1812.05069, 2018. 1

Appendix

Supplementary Material

A. Code details

The initial code was forked from https://medium.com/the-dl/easy-self-supervised-learning-with-byol-53b8ad8185d. However, a large number of bugs existed in the original code, and much time was spent debugging and fixing the code to demonstrate that the BYOL baseline works. We added a lot of functionality to this base implementation as well. We reorganized the code in a much cleaner way that allowed us to easily run experiments. We added the code to output Tensorboard logs, npy files for plotting, and model checkpoints. We also added code for calculating and evaluating the linear accuracy at each epoch of pre-training with BYOL.

B. Additional figures

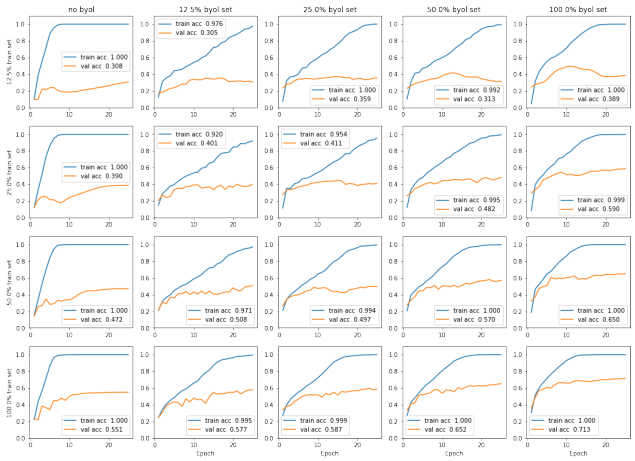

Figure 4. BYOL varying dataset size

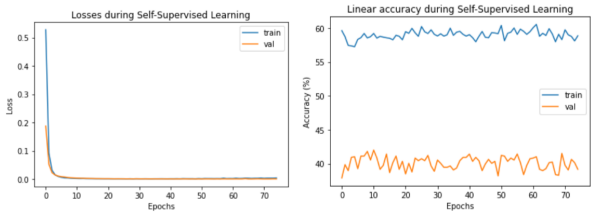

Figure 5. Example of mode collapse. When the augmentations are not robust enough, the network collapses by learning to predict the same output embedding for every image, and the loss plummets during pre-training.

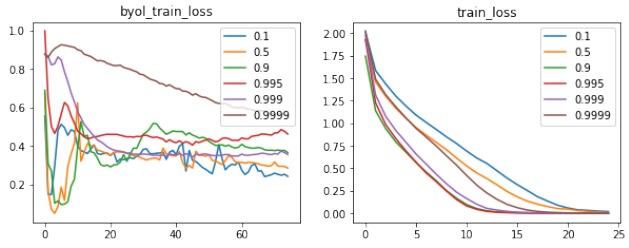

Figure 6. Loss curves for fixed τ target updates. Left: training loss during BYOL pretraining. Right: Fine tuning train loss.

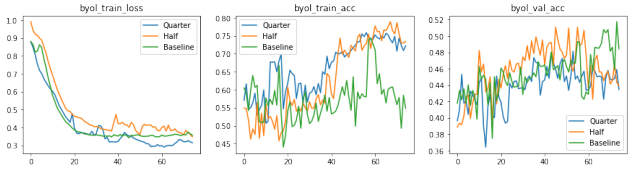

Figure 7. Loss and accuracy curves for fixed τ plus dropping the target at quarter/half (versus baseline).

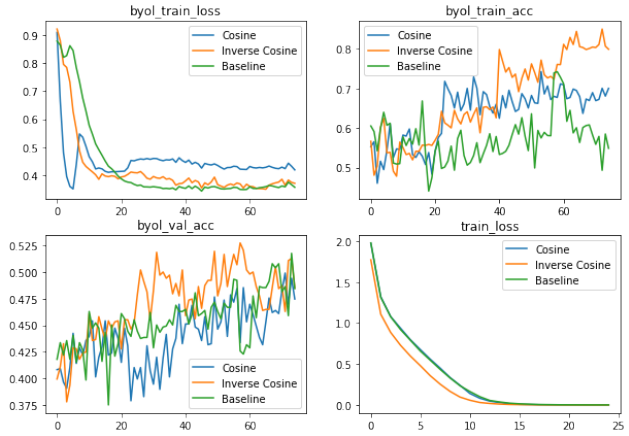

Figure 8. Loss and accuracy curves for cosine and inverse cosine τ update schedules (verses baseline).