SELF-SUPERVISED LEARNING FOR BINARY NETWORKS BY JOINT CLASSIFIER TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite the great success of self-supervised learning with large floating point networks, such networks are not readily deployable to edge devices. To accelerate deployment of models to edge devices for various downstream tasks by unsupervised representation learning, we propose a self-supervised learning method for binary networks. In particular, we propose to use a randomly initialized classifier attached to a pretrained floating point feature extractor as targets and jointly train it with a binary network. For better training of the binary network, we propose a feature similarity loss, a dynamic balancing scheme of loss terms, and modified multi-stage training. We call our method as *BSSL*. Our empirical validations show that BSSL outperforms self-supervised learning baselines for binary networks in various downstream tasks and outperforms supervised pretraining in certain tasks.

1 INTRODUCTION

Recent years have witnessed great successes in self-supervised learning (SSL) for floating point (FP) networks (Goyal et al., 2021; Tian et al., 2021a; Zbontar et al., 2021; Li et al., 2021a; Cai et al., 2021; Ericsson et al., 2021; Tian et al., 2021b; Ermolov et al., 2021; Tian et al., 2020b; Chen et al., 2020b; Grill et al., 2020; Caron et al., 2020; He et al., 2020) that perform *on par* with or even outperform the supervised pretraining by the help of huge unlabeled data in a number of downstream tasks such as image classification (Abbasi Koohpayegani et al., 2020; Caron et al., 2020), semi-supervised fine-tuning (Grill et al., 2020; Caron et al., 2020; Chen et al., 2020b) and object detection (He et al., 2020). While recent works (Chen et al., 2020b; Grill et al., 2020; Caron et al., 2020; He et al., 2020) from resourceful research groups have shown that the gains from SSL scale up with model size and/or dataset size used for pretraining, there is little work where the resulting pretrained models are small such that we can expedite the AI deployment due to the high efficiency in computational and memory costs, and energy consumption (Coldewey) with the power of unsupervised learning. At the extreme of resource constrained scenarios, binary networks exhibit superior efficiency and the accuracy is being significantly improved (Rastegari et al., 2016; Lin et al., 2017; Liu et al., 2018; 2020; Martinez et al., 2020; Bulat et al., 2020; Kim et al., 2020a; Bulat et al., 2021). Thus, developing an SSL method for binary networks can further accelerate the deployment of models to edge devices for various downstream tasks, yet is seldom explored.

One of the most effective and popular ways of training a binary network is to utilize a pretrained FP network to provide additional supervisory signals via the KL divergence between the softmax outputs from the classifiers of the FP network and binary network (we call it as 'supervised KL div.') (Martinez et al., 2020; Liu et al., 2020; Bulat et al., 2021; 2020). However, this method *requires label supervision* as the classifier of the FP network needs to be pretrained with labeled data to provide meaningful targets for the binary network. Very recently, Shen et al. (2021) claimed to propose an unsupervised representation learning method for binary networks utilizing the supervised KL div. method. Unfortunately, the FP network used in their proposal should be *pretrained with labeled data*, making their method *not applicable to unsupervised learning* where no labeled data is available to pretrain the FP network.

Motivated to extend the supervised KL div. method to the unsupervised scenario where no labeled data is available in any of the training procedure, we propose the first method to specifically train binary networks in the unsupervised manner. We name it as **B**inary **S**elf-**S**upervised **L**earning or **BSSL**. Specifically, we first construct an FP network consisting of a fixed feature extractor pretrained in an SSL manner and a randomly initialized FP classifier. Then, we use the outputs of the randomly initialized FP classifier as pseudo-labels for the binary network and jointly optimize both the FP classifier and the binary network using a KL divergence loss. But the gradients provided by the randomly initialized FP classifier could have unexpectedly large magnitudes especially during early training phase. To alleviate the problem, we additionally propose to enforce feature similarity across both precision, providing stable gradients that bypass the randomly initialized classifier. As the relative importance of the feature similarity loss decreases as the FP classifier gets jointly trained to provide less random targets, we further propose to use a *dynamic balancing strategy* between the KL divergence loss and the feature similarity loss. Finally, we modify the multi-stage training scheme (Martinez et al., 2020) for BSSL to further improve the performance.

In extensive empirical validations with a wide variety of downstream tasks including linear evaluation on ImageNet, semi-supervised fine-tuning on ImageNet with 1% and 10% labeled data, object detection on Pascal VOC, SVM classification and few-shot SVM classification on Pascal VOC07, and transfer learning to diverse datasets such as CIFAR10, CIFAR100, CUB-200-2011, Birdsnap, and Places205, the binary networks trained by our method outperforms the tuned MoCoV2 (Shen et al., 2021) and other SSL methods by large margins.

We summarize our contributions as follows:

- We propose the first true SSL method specific for binary networks
- We propose and show that jointly training a random FP classifier used as targets is effective for binary SSL
- We propose to improve the baseline greatly via a feature similarity loss and dynamic balancing
- The proposed method BSSL outperforms other comparisons by large margins on a wide variety of downstream tasks

2 Related Work

2.1 Self-Supervised Representation Learning

To reduce the annotation cost for representation learning, self-supervised representation learning (SSL) methods including (Goyal et al., 2021; Tian et al., 2021a; Zbontar et al., 2021; Tian et al., 2020b;a; Chen et al., 2020a; He et al., 2020; Tian et al., 2020b; Chen et al., 2020b) and many more have been shown to be effective, with the Info-NCE loss (Oord et al., 2018) being a popular choice for many works. These methods use the instance discrimination task as the pretext task which aims to pull instances of the same image closer and push instances of different images farther apart (Wu et al., 2018; Oord et al., 2018). Different to these methods, (Grill et al., 2020; Caron et al., 2020; Li et al., 2021b; Wei et al., 2021; Fang et al., 2021; Abbasi Koohpayegani et al., 2020) use feature regression with an EMA target (Grill et al., 2020), matching cluster assignments (Caron et al., 2020; Li et al., 2021b), or matching similarity score distributions (Wei et al., 2021; Fang et al., 2021; Abbasi Koohpayegani et al., 2020) as the pretext task. We are most interested in BYOL (Grill et al., 2020), SWAV (Caron et al., 2020), InfoMin (Tian et al., 2020b), and SimCLRv2 (Chen et al., 2020b) as the four state-of-the-art methods as they offer the highest empirical performance and represent SSL methods that are based on the instance discrimination or other tasks. However, while these methods show promising results for large FP models and datasets, they do not consider resource constrained scenarios which are more practical, *e.g.*, models with smaller complexity.

2.2 Binary Networks

Since the advent of XNOR-Net (Rastegari et al., 2016), binary networks have become a popular method to reduce computational and memory costs. Since then, numerous approaches for binary networks (Lin et al., 2017; Liu et al., 2018; 2020; Martinez et al., 2020; Bulat et al., 2020; Kim et al., 2020a; Bulat et al., 2021; Lin et al., 2020; Qin et al., 2020; Han et al., 2020; Meng et al., 2020; Kim et al., 2020b) have been proposed. These approaches include searching architectures for binary networks (Kim et al., 2020a; Bulat et al., 2020) via gradient-based NAS methods to using a specialized activation function (Liu et al., 2020) for binary networks. Note that previous works mostly focused on the supervised training set-up.

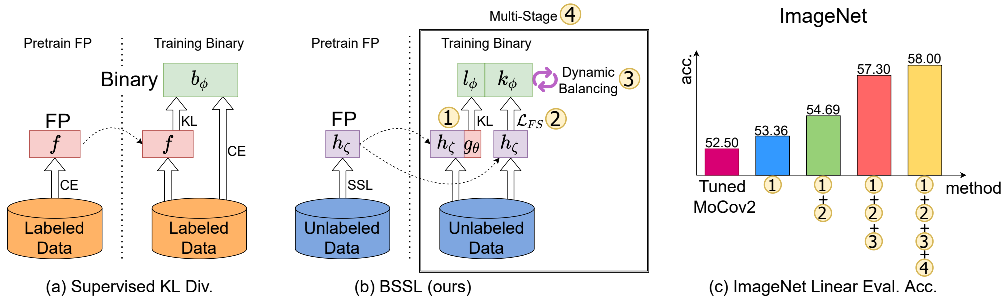


Figure 1: Supervised KL div. method (Martinez et al., 2020; Liu et al., 2020), proposed BSSL (uses no supervision), and linear evaluation accuracy of ablated models on ImageNet. Shen et al. (2021) tuned MoCov2 for binary networks and we show it ('Tuned MoCov2' in (c)) as a reference. Our baseline (1) already outperforms it by a noticeable margin and gain by the full model (+5.5%) is significant, given that the downstream task is on a large scale dataset, ImageNet.

Among many proposals, two recent works stand out as the state-of-the-art on binary networks due to their strong empirical performance: ReActNet (Liu et al., 2020) and High-Capacity Expert Binary Networks (HCEBN) (Bulat et al., 2021). (Liu et al., 2020) proposes to learn channel-wise thresholds for the binarization process by introducing RSign and RPReLU activation functions. (Bulat et al., 2021) proposes to learn multiple experts per convolution layer, inspired by conditional computing. They also explore effective ways to increase the representation capacity of binary networks without incurring an increase in total operation count (OPs). We use ReActNet as our backbone because of its efficiency and high accuracy; it achieves 69.4% Top-1 accuracy on the ImageNet with 0.87×10^8 OPs while HCEBN achieves 71.2% Top-1 accuracy with 1.36×10^8 OPs.

Key components of the recent advances in the field of binary networks are the 'supervised KL div.' method and the multi-stage training scheme (Liu et al., 2020; Martinez et al., 2020). The supervised KL div. method uses a pretrained FP network to provide targets for the KL div. loss in training binary networks. The multi-stage training scheme trains a binary network in multiple stages, where more and more parts of the network are binarized. Very recently, (Shen et al., 2021) proposed to utilize the supervised KL div. method to train binary networks in an SSL manner. However, the FP network used in their proposal is pretrained with labeled data which makes the proposed method inapplicable to the unsupervised scenario (Please refer to Sec. A.5 for details). Note that they also report results using tuned MoCov2 for binary networks, which we have comparisons with.

In contrast, our work resides in the intersection of SSL and binary networks, a field seldom explored.

3 Approach

The supervised KL div. method is an effective and well-known method to train binary networks (Martinez et al., 2020; Liu et al., 2020) with labeled data. But, as we are interested in the self-supervised learning with no access to labeled data at any time during training, the supervised KL div. is not applicable because we need labeled data to train the FP classifier. Here, we propose a self-supervised learning method for binary networks by a knowledge transfer mechanism from networks that extends the supervised KL div. method to the unsupervised scenario. We illustrate the supervised KL div. method (Martinez et al., 2020; Liu et al., 2020) and our proposal in Fig. 1.

Specifically, instead of using softmax outputs from a fixed pretrained classifier with labeled data, we propose to use softmax outputs from *a randomly initialized classifier* that is jointly trained with the binary network using the KL divergence loss. As the supervision from the untrained classifier makes gradients with unexpectedly high magnitudes, we subdue gradients by proposing an additional feature similarity loss across precisions. To improve the performance further, we propose to use a dynamic balancing scheme between the loss terms and employ multi-stage learning (Martinez et al., 2020) for better learning efficacy.

3.1 Knowledge Transfer from Jointly Trained Classifier with No Labels

Grill et al. (2020) show that even when a randomly initialized exponential moving average (EMA) network is used as the target network, the online network improves by training with it. One reason for the improvement could be that the randomly initialized EMA target network is also updated in an EMA manner during training, improving the target network gradually. Inspired by that, we consider whether a randomly initialized classifier attached to a pretrained FP feature extractor can be used as a *pseudo-label generator* for training binary networks. To gradually improve the classifier, we jointly train only the classifier and the binary network. Thus, we do not need any labeled data in pretraining the FP feature extractor nor when we jointly train the FP classifier and the binary network.

The joint training of randomly initialized classifier is depicted in ① in Fig. 1-(b). Specifically, a FP network $f(\cdot)$ is decoupled into $h_\zeta(\cdot)$, the pretrained and fixed FP feature extractor, and $g_\theta(\cdot)$ the randomly initialized and trainable classifier. We use the outputs of $g_\theta(\cdot)$ as pseudo-labels for training the binary network $b_\phi(\cdot)$. Formally, our objective is to minimize the KL divergence between the outputs of $g_\theta(\cdot)$ and $b_\phi(\cdot)$ as:

$$\min_{\theta,\phi} \mathbb{E}_{x\sim\mathcal{D}}\left[\mathcal{L}_{KL}(g_\theta(h_\zeta(x)), b_\phi(x))\right], \tag{1}$$

where x is a sample from the dataset $\mathcal{D}$ and $\mathcal{L}_{KL} = D_{KL}(\cdot\|\cdot)$ is the KL divergence between the outputs of $g_\theta(\cdot)$ and $b_\phi(\cdot)$. However, the softmax outputs from the classifier would be close to random early on, thus immediately using the outputs from a random classifier as the only target for the binary network could result in noisy gradients.

3.2 Stabilize Gradients by Feature Similarity across Precisions

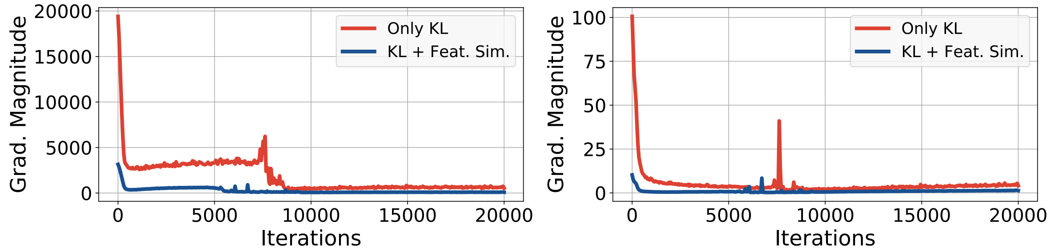
Note that $g_\theta(\cdot)$ will be updated quickly by the joint training, especially in the early learning phase. As the binary classifier uses the quickly changing $g_\theta(\cdot)$ as a target to transfer knowledge from, the binary classifier might receive large gradients. To address the potentially undesirably large gradients caused by the randomly initialized classifier being the only target, we propose to augment an additional loss term that bypasses the classifier in addition to gradient clipping (Zhang et al., 2019; Chen et al., 2020c). We call it as *feature similarity loss*. The new loss provides supervisory signals to the binary feature extractor from the feature extractor of the FP network not from the randomly initialized classifier. Since the feature extractor of the FP network is pretrained and fixed, the learned feature vectors by the FP network serves as stationary and stable targets as opposed to the softmax output from the randomly initialized classifier.

Specifically, we use the cosine distance between the feature vectors from the FP and binary feature extractors as the feature similarity loss; $\mathcal{L}_{FS}(v_1, v_2) = 1 - \frac{\langle v_1, v_2\rangle}{\|v_1\|_2 \cdot \|v_2\|_2}$ for smoothness and a bounded nature to prevent large gradients. The cosine distance (or $1-$the cosine similarity) is widely used in numerous representation learning literature (Grill et al., 2020; Xiao et al., 2021; He et al., 2020; Chen et al., 2020a). Augmenting the cosine distance to the KL divergence loss, we can write our new objective as:

$$\min_{\theta,\phi} \mathbb{E}_{x\sim\mathcal{D}}\left[(1-\lambda)\mathcal{L}_{KL}(g_\theta(h_\zeta(x)), l_\phi(k_\phi(x))) + \lambda\mathcal{L}_{FS}(h_\zeta(x), k_\phi(x))\right], \tag{2}$$

where the binary network $b_\phi(\cdot)$ is also decoupled into $k_\phi(\cdot)$ the binary feature extractor and the classifier $l_\phi(\cdot)$, λ is a static balancing factor, and $\mathcal{L}_{FS}(\cdot,\cdot)$ is the feature similarity loss.

We empirically observe the gradient of the binary classifier and feature extractor with and without $\mathcal{L}_{FS}$ in Fig 2. Note that with only KL, the gradients of the binary classifier are extremely large; it starts at roughly $20,000$ then drops to roughly $3,000$ for some iterations and finally drops to a reasonably small value starting at roughly $9,000$ iterations. In addition, there is a surge in gradient magnitude at around $7,500$ iterations. The binary feature extractor also shows a similar trend where the gradients exhibit a sudden spike at around $7,500$ iterations. Both very high magnitudes of the gradients at the start and the sudden spike, occurring after some iterations, harm training stability. However, as shown in the figure, addition of the proposed $\mathcal{L}_{FS}(\cdot,\cdot)$ significantly reduces the gradient magnitudes of the binary classifier and the feature extractor at early iterations as well as reducing surges throughout the training, empirically validating the effectiveness our proposal.



(a) Gradient magnitudes of the classifier

(b) Gradient magnitudes of the feature extractor

Figure 2: Gradient magnitude for the binary classifier (a) and the binary feature extractor (b) during early training with and without $\mathcal{L}_{FS}$. With only KL, the gradients of the classifier is extremely large and this carries over to the feature extractor. Additionally, we observe intermediate spikes for both the classifier and the feature extractor. The addition of $\mathcal{L}_{FS}$ significantly lowers the gradient magnitudes of the classifier as well as the feature extractor at early iterations. Additionally, the surges in gradient magnitudes are also subdued.

3.3 Dynamic Balancing of λ

As g_θ is gradually updated and provides less random targets, $\mathcal{L}_{FS}$ becomes less important. Thus, we additionally propose a *dynamic balancing* strategy to replace the static balancing factor λ in Eq. 2 by a smooth cosine annealing similar to how Grill et al. (2020) annealed the momentum value as:

$$\lambda(t) = \lambda_T - (\lambda_T - \lambda_0) \cdot (\cos(\pi t/T) + 1)/2, \tag{3}$$

where $\lambda_0 = 0.9$, $\lambda_T = 0.7$, T is the maximum training iteration and t is the current training iteration. Thus, $\lambda(t)$ will start at 0.9 then gradually decay to 0.7. In other words, the cosine distance is emphasized more at the beginning and gradually less emphasized as the learning progresses.

With all the components, our final objective can be written as:

$$\min_{\theta,\phi} \mathbb{E}_{x\sim\mathcal{D}}[(1-\lambda(t)) \cdot \mathcal{L}_{KL}(g_\theta(h_\zeta(x)), l_\phi(k_\phi(x))) + \lambda(t) \cdot \mathcal{L}_{FS}(h_\zeta(x), k_\phi(x))]. \tag{4}$$

3.4 Multi-Stage Training for BSSL

The multi-stage training scheme (Martinez et al., 2020; Bulat et al., 2020; Liu et al., 2020) is known to be effective in training binary networks. In the multi-stage training method, the binary network is trained with only the activations being binarized in the first stage. The trained weights are then used as initial values for training the network in a fully binarized setting in the second stage. Unfortunately, we cannot directly use the same strategy as the binary networks would converge too quickly due to having good initial values and the FP classifier g_θ cannot converge as quickly as the binary network, providing low quality targets.

Thus, we propose to modify the multi-stage training scheme for BSSL which jointly optimizes a randomly initialized classifier. Specifically, we load the weights of g_θ obtained during the first stage when training the binary network in the second stage. Rather, g_θ is also given good initial points so that it too can converge quickly and provide high quality targets.

With all the proposals, we describe the details of our proposed **BSSL** in Alg. 1

4 Experiments

Pretraining. Following Xiao et al. (2021); Lee et al. (2021); Chuang et al. (2020); Kalantidis et al. (2020); Zhao et al. (2021); Wang & Isola (2020), we use ImageNet (Krizhevsky et al., 2012) for pretraining. We provide results of pretraining with ImageNet100 (Tian et al., 2020a) in Sec. A.2.

Downstream Tasks. We use 1) linear evaluation on ImageNet, 2) semi-supervised fine-tuning on ImageNet, 3) object detection on Pascal VOC, 4) image classification and few-shot image classification using SVM on VOC07, and 5) transfer learning via linear evaluation on frozen backbone on

Algorithm 1 Self-Supervised Learning for Binary Networks (BSSL)

function STAGE 1($\mathcal{D}$,t, ζ,h_ζ, g_θ, k_ϕ, l_ϕ) ▷ Only Binarized Activations
 $h_\zeta \leftarrow \zeta$ ▷ Load pretrained weights for h_ζ
 $x = \text{RandomSelect}(\mathcal{D})$ ▷ Sample $x \sim \mathcal{D}$
 $v_1, v_2 = h_\zeta(x), k_\phi(x)$ ▷ Feature vectors v_1, v_2
 $p_1, p_2 = g_\theta(v_1), l_\phi(v_2)$ ▷ Softmax Probabilities p_1, p_2
 $\mathcal{L}_{\zeta,\theta,\phi} = \text{AugmentedLoss}(v_1, v_2, p_1, p_2, t)$
 $\theta \leftarrow \text{Optimizer}(\nabla_\theta \mathcal{L}_{\zeta,\theta,\phi}, \eta)$ ▷ Update θ
 $\phi \leftarrow \text{Optimizer}(\nabla_\phi \mathcal{L}_{\zeta,\theta,\phi}, \eta)$ ▷ Update ϕ
 return θ, ϕ
end function
function STAGE 2($\mathcal{D}$,t, ζ,θ, ϕ,h_ζ, g_θ, k_ϕ, l_ϕ) ▷ Fully Binarized
 $h_\zeta, g_\theta, k_\phi, l_\phi \leftarrow \zeta, \theta, \phi$ ▷ Load pretrained weights for $h_\zeta, g_\theta, k_\phi, l_\phi$
 $x = \text{RandomSelect}(\mathcal{D})$ ▷ Sample $x \sim \mathcal{D}$
 $v_1, v_2 = h_\zeta(x), k_\phi(x)$ ▷ Feature vectors v_1, v_2
 $p_1, p_2 = g_\theta(v_1), l_\phi(v_2)$ ▷ Softmax Probabilities p_1, p_2
 $\mathcal{L}_{\zeta,\theta,\phi} = \text{AugmentedLoss}(v_1, v_2, p_1, p_2, t)$
 $\theta \leftarrow \text{Optimizer}(\nabla_\theta \mathcal{L}_{\zeta,\theta,\phi}, \eta)$ ▷ Update θ
 $\phi \leftarrow \text{Optimizer}(\nabla_\phi \mathcal{L}_{\zeta,\theta,\phi}, \eta)$ ▷ Update ϕ
 return k_ϕ
end function
function AUGMENTEDLOSS(v_1, v_2, p_1, p_2, t)
 $\mathcal{L}_{KL} = \mathcal{D}_{KL}(p_2 \| \| p_1)$ ▷ KL Divergence
 $\mathcal{L}_{FS} = 1 - \frac{\langle v_1, v_2 \rangle}{\|v_1\|_2 \cdot \|v_2\|_2}$ ▷ Cosine Distance
 $\lambda(t) = \lambda_T - (\lambda_T - \lambda_0) \cdot (\cos(\pi t/T) + 1)/2$ ▷ Dynamic Balancing Eq. 3
 $\mathcal{L} = (1 - \lambda(t) \cdot \mathcal{L}_{KL} + \lambda(t) \cdot \mathcal{L}_{aug}$ ▷ Final Loss Eq. 4
 return $\mathcal{L}$
end function

CIFAR10, CIFAR100, Birdsnap, CUB-200-2011, and Places205 for downstream tasks. We strictly follow the evaluation protocols of the respective downstream tasks (Goyal et al., 2019; He et al., 2020; Chen et al., 2020b;a). More details can be found in Sec. A.1.

Implementation Details. For our binary network for all experiments, we use the official implementation of the ReActNet-A (Liu et al., 2020). Our models are trained with the LARS optimizer (You et al., 2017) with a batch size of $2,048$ for 200 epochs on ImageNet with the learning rate set as 0.3. For the FP network, we use a ResNet50 pretrained using MoCov2 (He et al., 2020) on ImageNet for 800 epochs. All codes for pretraining and downstream tasks will be publicly released.

Baselines. As mentioned in Sec. 2, the proposed method from (Shen et al., 2021) cannot be applied to the unsupervised scenario and hence is not a fair comparison. Instead, we establish baselines to compare our BSSL method by pretraing a ReActNet-A with either BYOL (Grill et al., 2020), SWAV (Caron et al., 2020), tuned MoCov2 (Shen et al., 2021), or supervised pretraining. Note that we mainly compare to other SSL methods and discuss additional comparisons to supervised pretraining in Sec. A.3. Number of epochs used in pretraining is kept the same (200) for all methods.

4.1 RESULTS ON DOWNSTREAM TASKS

We denote the best results in each table in **bold.**

Linear Evaluation. We conduct linear evaluation (top-1) on ImageNet and summarize the results in Table 1. Once the binary feature extractor is pretrained, it is frozen and only the attached classifier is trained for classification. As shown in the table, BSSL outperforms other SSL methods by at least $+5.5\%$ and up to $+14.69\%$ top-1 accuracy, possibly because it utilizes the knowledge from the FP network effectively. Interestingly, BSSL even outperforms supervised binary network methods such as XNOR-Net (51.20%) (Rastegari et al., 2016).

Method	Linear Eval.	Semi-Supervised Fine-tuning 1% Labels		10% Labels	
	Top-1 (%)	Top-1 (%)	Top-5 (%)	Top-1 (%)	Top-5 (%)
Supervised	64.10	42.96	69.10	53.07	77.40
SWAV (Caron et al., 2020)	49.41	24.66	46.57	33.83	57.81
BYOL (Grill et al., 2020)	49.25	23.05	43.90	34.66	58.78
Tuned MoCov2 (Shen et al., 2021)	52.50	22.96	45.12	31.18	55.64
BSSL (Ours)	**58.00**	**35.53**	**61.02**	**43.25**	**68.82**

Table 1: Linear evaluation (top-1) and semi-supervised fine-tuning (1% labels or 10% labels) on ImageNet after pretraining. BSSL outperforms all other SSL methods by large margins across for both the linear evaluation and semi-supervised fine-tuning.

Semi-Supervised Fine-Tuning. We conduct semi-supervised fine-tuning (top-1 and top-5) and summarize the results in Table 1. We fine-tune the entire network on the labeled subset (1% or 10%) from ImageNet. BSSL outperforms other SSL baselines by large margins across all metrics; at least $+10.87\%$ top-1 accuracy and $+14.45\%$ top-5 accuracy on the 1% labels setting and $+8.59\%$ top-1 accuracy and $+10.04\%$ top-5 accuracy on the 10% labels setting, respectively. Interestingly, the gain by BSSL to other SSL methods in semi-supervised fine-tuning is much larger than the gain in the linear evaluation, implying that BSSL is more beneficial in tasks with limited supervision as also discussed by Goyal et al. (2021).

Method	mAP (%)	AP50 (%)	AP75 (%)
Supervised	38.22	68.53	37.65
SWAV	37.22	67.47	35.91
BYOL	36.92	67.13	35.65
Tuned MoCov2	37.42	67.30	36.37
BSSL (Ours)	**41.00**	**70.91**	**41.45**

Table 2: Object detection (mAP, AP50 and AP75) on Pascal VOC after pretraining. BSSL outperforms all the compared methods including supervised pretraining.

Object Detection. We conduct object detection (mAP (%), AP50 (%) and AP75 (%)) on Pascal VOC and summarize the results in Table 2. Once the feature extractor is pretrained, we use the pretrained weights as initial weights for the detection framework and fine-tune the entire detection framework. Note that object detection task is hard for binary networks (Wang et al., 2020), especially with the settings of (He et al., 2020); even the supervised pretraining only achieves 38.22% mAP on Pascal VOC. Nonetheless, BSSL outperforms all other methods in all three metrics. We believe the one of the reasons for the gain is that BSSL utilizes a FP network trained in an SSL manner that mostly learned low- and mid-level representations (Zhao et al., 2021) which help object detection.

SVM Image Classification. We conduct SVM classification (mAP (%)) and summarize results for both the few-shot and full-shot ('Full') settings on VOC07 in Table 3. For the few-shot results, the results are averaged over 5 runs. The number of shots k is varied from 1 to 96.

For the few-shot setting, BSSL outperforms all other SSL methods by roughly $+6\%$ to $+10\%$ mAP depending the number of shots. Noticeably, BSSL performs very close to the supervised pretraining regardless of the number of shots. Similar to the semi-supervised fine-tuning, BSSL shows strong performance in tasks with limited supervision such as the few-shot classification (Goyal et al., 2021). In the full-shot setting, BSSL outperforms other SSL methods by at least $+6.26\%$ mAP and performs on par with the supervised pretraining. In both settings, representations learned with ImageNet by BSSL is still effective on a different dataset such as VOC07, potentially due to BSSL using a FP network to obtain more general targets (low to mid level representations) (Zhao et al., 2021).

Transfer Learning. We summarize the results of transfer learning by the linear classification (top-1) task on various datasets in Table 4. We use two types of datasets, *i.e.*, object-centric and scene-centric, to test the knowledge transferability of learned representations across domains. Specifically, as we use ImageNet (object-centric) for pretraining, we evaluate methods in the transfer scenario to object centric datasets such as CIFAR10, CIFAR100, CUB-200-2011 and Birdsnap, and to a scene centric dataset such as Places205. Once we pretrain the binary feature extractor with ImageNet, the feature extractor is frozen and only the attached classifier is trained on the target datasets.

BSSL outperforms all SSL methods on the object-centric datasets with particularly large margins in CUB-200-2011 and Birdsnap. It implies that the representations learned using BSSL transfers well across multiple object-centric datasets. Interestingly, for the scene-centric dataset (Places205),

Method	k = 1	k = 2	k = 4	k = 8	k = 16	k = 32	k = 64	k = 96	Full
Supervised	29.28± 0.94	36.46± 2.97	49.67± 1.20	56.99± 0.67	64.68±0.89	70.08± 0.58	73.49± 0.53	74.96± 0.17	77.47
SWAV	22.97 ± 1.21	27.91± 2.37	37.91±1.11	44.5± 1.51	52.79± 0.81	59.15± 0.62	64.38± 0.59	66.72± 0.19	71.23
BYOL	23.45 ± 0.76	28.04± 2.40	38.09± 1.07	44.69± 1.66	51.5± 0.90	57.44±0.24	62.07± 0.28	64.37± 0.13	69.16
Tuned MoCov2	22.12 ± 0.74	27.45 ± 2.06	36.81 ± 0.82	43.19 ±1.4	51.93 ±0.84	57.95 ± 0.62	63.07 ± 0.43	65.15 ±0.05	69.73
BSSL (Ours)	**29.20±1.51**	**36.14 ±2.15**	**48.49 ± 1.08**	**55.12 ± 1.59**	**62.36 ± 1.01**	**67.70 ± 0.3**	**72.1 ± 0.39**	**74.06 ± 0.18**	**77.49**

Table 3: SVM classification (mAP) for the few-shot and full-shot settings on VOC07 after pretraining. BSSL outperforms all other SSL methods by large margins and performs on par with supervised pretraining on both settings. The number of shots (k) is varied from 1 to 96. We report the averaged performance over 5 runs with the standard deviation.

Method	Object-Centric				Scene-Centric
	CIFAR10	CIFAR100	CUB-200-2011	Birdsnap	Places205
Supervised	78.30	57.82	54.64	36.90	46.38
SWAV	75.78	56.78	36.11	25.54	**46.90**
BYOL	76.68	58.18	38.80	27.11	44.62
Tuned MoCov2	78.29	57.56	33.79	23.37	44.90
BSSL (Ours)	**78.32**	**58.20**	**44.41**	**34.00**	46.20

Table 4: Transfer learning (top-1) on either object-centric or scene-centric datasets after pretraining. CIFAR10, CIFAR100, CUB-200-2011, and Birdsnap are used as the object-centric datasets while Places205 is used as the scene-centric dataset. BSSL outperforms all other SSL baselines on the object-centric datasets and performs similar to other methods on Places205.

we observed that the transfer learning performance for various methods exhibits marginal difference including supervised pretraining. It is expected as ImageNet is object-centric, *i.e.*, transferring knowledge to a scene-centric dataset may suffer from domain gap which results in similar performance across the methods.

4.2 Ablation Studies

We use linear evaluation (top-1) on ImageNet for all our ablation studies.

Components of BSSL. We summarize results for an ablation study of the components of BSSL in Table 5. We number each components, following the convention in Fig. 1. As shown in the table, every component in BSSL contributes to a non-trivial gain given that the study is conducted with a large scale dataset. While dynamic balancing provides the largest gains, it only makes sense when it is used with the added feature similarity loss $\mathcal{L}_{FS}$. Note that the addition of $\mathcal{L}_{FS}$ stabilizes the gradients (see Sec. 3.2) and a dynamic balancing of $\mathcal{L}_{FS}$ captures the changing importance of $\mathcal{L}_{FS}$ stabilizing the gradients even more effectively, resulting in large gains. Interestingly, using just the randomly initialized classifier as targets we outperform tuned MoCov2 (Shen et al., 2021), the best performing SSL baseline in linear evaluation excluding BSSL.

Method	① Rand. Init. Cls.	② Feat. Sim. Loss	③ Dyn. Bal.	④ Multi-Stage	Top-1 (%)
Tuned MoCov2	-	-	-	-	52.50
①	✓	✗	✗	✗	53.36
①+②	✓	✓	✗	✗	54.69
①+②+③	✓	✓	✓	✗	57.30
①+②+③+④ (=BSSL)	✓	✓	✓	✓	**58.00**

Table 5: Ablation studies on the various proposed components of BSSL. ① refers to using a randomly initialized classifier as targets. ② 'Feat. Sim.' refers to feature similarity loss (Eq. 2). ③ 'Dyn. Bal.' refers to using the dynamic balancing. ④ refers to using the tuned multi-stage training. Each step of improving BSSL contribute to a non-trivial performance gain as the evaluation is done with the ImageNet dataset. Also, using only ① already outperforms 'Tuned MoCov2', the state-of-the-art SSL baseline for linear evaluation (Shen et al., 2021).

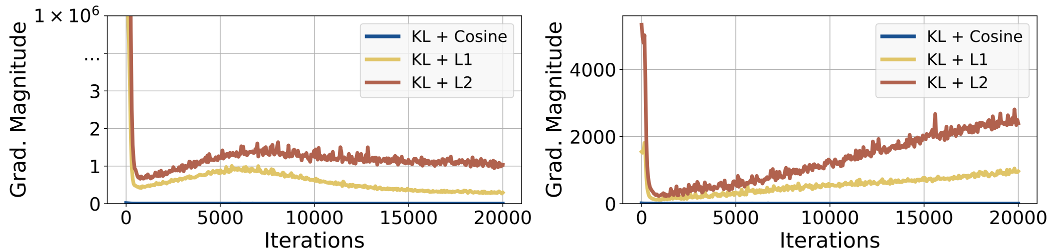
Choice of Feature Similarity Loss. We further investigate the choices of the feature similarity loss ($\mathcal{L}_{FS}$ in Eq. 4) in Table 6. Besides the cosine distance used in BSSL, we compare the L_1 and L_2

$\mathcal{L}_{FS}$	Bounded	Top-1 (%)
L_1: $\|v_1 - v_2\|_1$	✗	51.46
L_2: $\|v_1 - v_2\|_2$	✗	50.28
Cosine: $1 - \frac{\langle v_1, v_2 \rangle}{\|v_1\|_2 \cdot \|v_2\|_2}$	✓	**58.00**

Table 6: L_1, L_2, and the cosine distances are compared. The cosine distance is by far the best choice amongst the three, as is supported by our intuition that a bounded loss term would be better as the augmented loss term.

$\lambda(t)$	Top-1 (%)
$\lambda(t) = 0.7$	55.83
$\lambda(t) = \begin{cases} 1, & \text{if } t < T_{max}/2 \\ 0, & otherwise \end{cases}$	55.60
Eq. 3	**58.00**

Table 7: Comparison of dynamic balancing functions. The Eq. 3 (smooth annealing) is the best amongst the three choices. The $\lambda(t) = 0.7$ does not capture the dynamic nature of the balancing factor and the inverse shifted Heaviside step function disrupts the training midway.



(a) Gradient magnitudes of the classifier (b) Gradient magnitudes of the feature extractor

Figure 3: Gradient magnitude of (a) binary classifier and (b) the binary feature extractor during early training for various choices of $\mathcal{L}_{FS}$ such as the cosine, L_1, and L_2 distances. Both L_1 and L_2 distances show very high gradients at the beginning in the classifier, especially L_2. Moreover, L_1 and L_2 distances exhibit potential gradient explosions in the feature extractor. The proposed cosine distance shows none of these trends that harm training efficacy.

distances. We believe that as both the L_1 and L_2 distances are not bounded, they may potentially cause problem of gradient explosion leading to the worse performance unlike the cosine distance.

The cosine distance outperforms L_1 and L_2 by large margins. In Fig. 3, we illustrate gradient magnitudes of both the classifier and feature extractor when using cosine, L_1, or L_2 distances as $\mathcal{L}_{FS}$. L_1 and L_2 distances show very high gradients early on in the classifier, especially L_2 where the gradients start off at 1×10^6. Even more importantly, L_1 and L_2 distances show signs of gradient explosions in the feature extractor with L_2 suffering more severely. In contrast, the proposed cosine distance exhibits small and subdued gradients for both the classifier and the feature extractor.

Choice of the Dynamic Balancing Function. We compare (1) a constant function, $\lambda(t) = 0.7$, (2) an inverse shifted Heaviside step function, $\lambda(t) = \begin{cases} 1, & \text{if } t < T_{max}/2 \\ 0, & otherwise \end{cases}$ and (3) a smooth annealing using the cosine function (Eq. 3) used in BSSL, as choices for the dynamic balancing function in Table 7. The constant function does not capture that the importance of $\mathcal{L}_{FS}$ can change as learning progresses, leading to poor results. The inverse shifted Heaviside step function abruptly changes the balancing factor mid-training, disrupting the training and leads to poor performance. In contrast, the proposed smooth annealing function captures the dynamic nature of the importance of the feature similarity loss while smoothly changing the balancing factor, resulting in the best performance.

5 Conclusion

We propose BSSL, the first SSL framework specific for binary networks by jointly training the FP classifier and the binary network, extending the supervised KL div. method to the unsupervised scenario. We propose a feature similarity loss, dynamic balancing of the losses, and a tuned multi-stage training to improve BSSL. We conduct extensive empirical validations with five different downstream tasks with seven datasets. In all downstream tasks, BSSL consistently outperforms existing SSL baselines by large margins and sometimes supervised pretraining. We further investigate the contributions of the proposed components by various ablations studies.

Ethics Statement

AI models with binary weights and activations would significantly expedite the deployment of AI for edge devices such as robotics agents and surveillance systems, and our proposed method improves its accuracy for wide deployment of AI to resource constrained users. We believe that it helps democratizing the AI to wider range of users but at the same time, once the edge AI is easily deployable by the proposed method, the system may potentially be used for monitoring unwanted mass populations, which exploits private information such as identity, clothing information and personal attributes (*e.g*., age, gender and *etc*.) could be obtained by adversaries. Although the proposed method has *no intention* to allow such problematic cases, the method may be exposed to such threats. Relentless efforts should be made to develop mechanisms to prevent such usage cases in order to make the easily deployable machine learning models safer and enjoyable to be used by humans.

Reproducibility Statement

We take the reproducibility of the research very seriously and solemnly promise to release all codes, containers (*e.g*., Docker) that includes running environments and learned models of pretraining and downstream tasks in a public repository.

References

Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and Hamed Pirsiavash. Compress: Self-supervised learning by compressing representations. In *NeurIPS*, 2020.

Adrian Bulat, Brais Martínez, and Georgios Tzimiropoulos. Bats: Binary architecture search. In *ECCV*, 2020.

Adrian Bulat, Brais Martinez, and Georgios Tzimiropoulos. High-capacity expert binary networks. In *ICLR*, 2021.

Zhaowei Cai, Avinash Ravichandran, Subhransu Maji, Charless Fowlkes, Zhuowen Tu, and Stefano Soatto. Exponential moving average normalization for self-supervised and semi-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 194–203, June 2021.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020a.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. In *NeurIPS*, 2020b.

Xiangyi Chen, Steven Z Wu, and Mingyi Hong. Understanding gradient clipping in private sgd: a geometric perspective. *Advances in Neural Information Processing Systems*, 33, 2020c.

Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. In *NeurIPS*, 2020.

Devin Coldewey. Xnor's saltine-sized, solar-powered ai hardware redefines the edge. `https://techcrunch.com/2019/02/13/xnors-saltine-sized-solar-powered-ai-hardware-redefines-the-edge`.

Linus Ericsson, Henry Gouk, and Timothy M. Hospedales. How well do self-supervised models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5414–5423, June 2021.

Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning. In *International Conference on Machine Learning*, pp. 3015–3024. ICML, 2021.

R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. LIBLINEAR: A library for large linear classification. *JMLR*, 2008.

Zhiyuan Fang, Jianfeng Wang, Lijuan Wang, Lei Zhang, Yezhou Yang, and Zicheng Liu. {SEED}: Self-supervised distillation for visual representation. In *ICLR*, 2021.

Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. In *ICCV*, 2019.

Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, et al. Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*, 2021.

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. In *NeurIPS*, 2020.

Kai Han, Yunhe Wang, Yixing Xu, Chunjing Xu, Enhua Wu, and C. Xu. Training binary neural networks through learning with noisy supervision. In *ICML*, 2020.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. In *NeurIPS*, 2020.

Dahyun Kim, Kunal Pratap Singh, and Jonghyun Choi. Learning architectures for binary networks. In *ECCV*, 2020a.

Hyungjun Kim, Kyungsu Kim, Jinseok Kim, and Jae-Joon Kim. Binaryduo: Reducing gradient mismatch in binary activation network by coupling binary activations. In *ICLR*, 2020b.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

Kibok Lee, Yian Zhu, Kihyuk Sohn, Chun-Liang Li, Jinwoo Shin, and Honglak Lee. i-mix: A strategy for regularizing contrastive representation learning. In *ICLR*, 2021.

Bin Li, Yin Li, and Kevin W. Eliceiri. Dual-stream multiple instance learning network for whole slide image classification with self-supervised contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14318–14328, June 2021a.

Junnan Li, Pan Zhou, Caiming Xiong, and Steven Hoi. Prototypical contrastive learning of unsupervised representations. In *ICLR*, 2021b.

Mingbao Lin, Rongrong Ji, Zihan Xu, Baochang Zhang, Yan Wang, Yongjian Wu, Feiyue Huang, and Chia-Wen Lin. Rotated binary neural network. In *NeurIPS*, 2020.

Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *NeurIPS*, 2017.

Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *ECCV*, 2018.

Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *ECCV*, 2020.

Brais Martinez, Jing Yang, Adrian Bulat, and Georgios Tzimiropoulos. Training binary neural networks with real-to-binary convolutions. In *ICLR*, 2020.

Xiangming Meng, Roman Bachmann, and Mohammad Emtiyaz Khan. Training binary neural networks using the bayesian learning rule. In *ICML*, 2020.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Haotong Qin, Ruihao Gong, Xianglong Liu, Mingzhu Shen, Ziran Wei, Fengwei Yu, and Jingkuan Song. Forward and backward information retention for accurate binary neural networks. In *CVPR*, 2020.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016.

Zhiqiang Shen, Zechun Liu, Jie Qin, Lei Huang, Kwang-Ting Cheng, and Marios Savvides. S2-bnn: Bridging the gap between self-supervised real and 1-bit neural networks via guided distribution calibration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2165–2174, 2021.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *ECCV*, 2020a.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In *NeurIPS*, 2020b.

Yonglong Tian, Olivier J Henaff, and Aaron van den Oord. Divide and contrast: Self-supervised learning from uncurated data. *arXiv preprint arXiv:2105.08054*, 2021a.

Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs. *ICML*, 2021b.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*, 2020.

Ziwei Wang, Ziyi Wu, Jiwen Lu, and Jie Zhou. Bidet: An efficient binarized object detector. In *CVPR*, 2020.

Chen Wei, Huiyu Wang, Wei Shen, and Alan Yuille. {CO}2: Consistent contrast for unsupervised visual representation learning. In *ICLR*, 2021.

Zhirong Wu, Yuanjun Xiong, X Yu Stella, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018.

Tete Xiao, Xiaolong Wang, Alexei A Efros, and Trevor Darrell. What should not be contrastive in contrastive learning. In *ICLR*, 2021.

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

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. 2021.

Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. *arXiv preprint arXiv:1905.11881*, 2019.

Nanxuan Zhao, Zhirong Wu, Rynson W. H. Lau, and Stephen Lin. What makes instance discrimination good for transfer learning? In *ICLR*, 2021.

A APPENDIX

A.1 DETAILS ON DOWNSTREAM TASK CONFIGURATIONS

We present detailed configurations for each downstream task. We strictly follow the experimental protocols from (Xiao et al., 2021; Chen et al., 2020b;a; Goyal et al., 2019; He et al., 2020).

Linear Evaluation. Following (Goyal et al., 2019), we attach a linear classifier (a single fully-connected layer followed by a softmax) on top of the frozen backbone network and train only the classifier for 100 epochs using SGD. The initial learning rate is set to 30 and multiplied by 0.1 at epoch 60 and 80. The momentum is set to 0.9 with no weight decay. The classifier is trained on the target datasets.

Semi-Supervised Fine-Tuning. Following (Chen et al., 2020b;a; Xiao et al., 2021), we attach a linear classifier (a single fully-connected layer followed by a softmax) on top of the backbone network and fine-tune the backbone as well as the linear classifier using SGD for 20 epochs. Different initial learning rates are used for the backbone and the linear classifier where we select one from $\{0.1, 0.01, 0.001\}$ for the backbone learning rate and we multiply either $\{1, 10, 100\}$ to the backbone learning rate for the linear classifier learning rate. We found the performance for different pretraining methods to vary considerably for the different learning rate configurations and hence we sweep all the 9 combinations described above and use the best configuration for each method for fine-tuning.

The momentum is set to 0.9 with a weight decay of 0.0005 and the learning rates for the backbone and the classifier are multiplied by 0.2 at epochs 12 and 16. For fine-tuning, only 1% or 10% of the labeled training images that are randomly sampled from the target datasets are used. The entire validation set is used for evaluation.

Object Detection. Following (He et al., 2020), we use the Faster R-CNN object detection framework. The Faster R-CNN framework is implemented using detectron2. We use Pascal VOC 2007 and Pascal VOC 2012 as the training dataset and test on the Pascal VOC 2007 test set. We use the pretrained weights as the initial weights and fine-tune the entire detection framework. We use the exact same configuration file from (He et al., 2020).

SVM Image Classification. Following (Goyal et al., 2019), we first extract the features from the backbone network and apply average pooling to match the feature vector dimension to be 4,096. Note that (Goyal et al., 2019) uses ResNet50 as a backbone and extracts features after each residual block to report the best accuracy. In contrast, we are based on ReActNet and extract features from the last layer as we found that to perform the best.

The feature vector dimension is 4,096 instead of 8,192 as in (Goyal et al., 2019) because of the backbone architecture difference. With the extracted features, we use the LIBLINEAR (Fan et al., 2008) package to train linear SVMs. For the regular classification, we use the 'trainval' split of VOC07 dataset for training and evaluate on the 'test' split of VOC07 dataset. We report the mAP for the regular classification. For the few-shot classification, we use the 'trainval' split of VOC07 dataset in the few-shot setting for training and evaluate on the 'test' split of VOC07 dataset. The number of shots k (per class) is varied from 1 to 96. We report average mAP over five independent samples of the training data along with the standard deviation for the few-shot classification.

Transfer Learning. We perform linear evaluation on various target datasets. For object-centric datasets, we follow (Goyal et al., 2019) and attach a linear classifier (a single fully-connected layer followed by a softmax) on top of the frozen backbone network and train only the classifier for 100 epochs using SGD. The initial learning rate is set to 30 and multiplied by 0.1 at epoch 60 and 80. The momentum is set to 0.9 with no weight decay. The classifier is trained on the target datasets. For scene-centric datasets, we follow (Goyal et al., 2019) and modify the configuration from the object-centric dataets. Namely, we train the classifier for 28 epochs using SGD. The learning rate is set as 0.01 initially and is multiplied by 0.1 at every 7 epochs. The momentum is set to 0.9 with weight decay set as 0.00001. The classifier is trained on the target datasets.

Pretrain on	Method	Top-1 (%)	mAP (%)
ImgNet100	Supervised	76.54	64.77
	InfoMin	45.38	47.32
	SimCLRv2	61.4	61.36
	SWAV	71.50	64.34
	BYOL	71.08	64.58
	BSSL (Ours)	**77.02**	**70.50**

Table 8: Linear evaluation (top-1) and image classification using SVM (mAP) on the ImageNet100 and VOC07 datasets after pretraining on ImageNet100 are shown. BSSL performs the best compared to SSL baselines and even outperforms the supervised pretraining. The best result among SSL methods for each task is shown in **bold**.

Pretrain on	Method	1% Labels		10% Labels	
		Top-1 (%)	Top-5 (%)	Top-1 (%)	Top-5 (%)
ImgNet100	Supervised	63.10	86.24	75.40	92.16
	InfoMin	21.68	46.74	32.06	59.82
	SimCLRv2	43.78	72.28	60.06	84.9
	SWAV	42.62	70.78	58.74	85.34
	BYOL	42.70	70.22	62.32	86.44
	BSSL (Ours)	**63.05**	**84.02**	**73.10**	**91.08**

Table 9: Semi-supervised fine-tuning with either 1% labels or 10% labels on the ImageNet100 dataset after pretraining on ImageNet100 are shown. BSSL is the best performer in all metrics compared to SSL baseline and performs close to the supervised pretraining. The best result among SSL methods for each setup is shown in **bold**.

Pretrain on	Method	k = 1	k = 2	k = 4	k = 8	k = 16	k = 32	k = 64	k = 96
ImgNet100	Supervised	22.18 ± 1.31	28.77 ± 1.97	36.59 ± 1.61	43.67 ± 0.93	50.61 ± 0.62	55.75 ± 0.43	59.39± 0.20	60.88± 0.41
	InfoMin	14.12± 0.23	17.07±0.93	20.76± 0.91	24.75 ± 0.27	29.9 ±0.73	35.12±0.52	39.2 ±0.31	41.90± 0.22
	SimCLRv2	17.97 ± 0.56	22.87 ±2.0	30.48 ± 1.02	34.98 ±1.58	42.9 ±1.03	48.81 ±0.67	53.87 ±0.48	56.21 ±0.25
	SWAV	21.70 ± 0.95	25.88 ± 2.39	34.36 ± 1.59	40.15 ± 1.30	46.19 ±1.09	51.97 ± 0.74	56.96 ± 0.65	59.43 ± 0.31
	BYOL	19.77 ± 0.41	24.1 ± 2.34	32.47 ± 1.10	38.33 ± 1.58	45.40 ± 0.86	51.70 ± 0.57	56.69 ±0.50	59.27± 0.21
	BSSL (Ours)	**25.03 ± 1.53**	**29.90 ± 1.98**	**38.95 ± 1.21**	**45.67 ± 1.75**	**52.56 ± 0.77**	**58.00 ±0.40**	**63.10 ± 0.28**	**65.05 ± 0.09**

Table 10: Few-shot image classification using SVM (mAP) on the VOC07 dataset after pretraining on ImageNet100 are shown. The number of shots (k) is varied from 1 to 96 and the average over 5 runs with the standard deviation are reported. BSSL outperforms all methods including the supervised pretaining. The best result among SSL methods for each shot is shown in **bold**.

A.2 Additional Results for Pretraining on ImageNet100

For a more comprehensive evaluation of BSSL, we present additional results for pretraining on the ImageNet100 dataset (Xiao et al., 2021). We also add InfoMin (Xiao et al., 2021) and SimCLRv2 (Chen et al., 2020a) with the ReActNet backbone as comparisons.

Linear Evaluation and SVM Image Classification. As shown in Table 8, BSSL outperforms both InfoMin and SimCLRv2 by large margins on the two tasks evaluated, *i.e.*, over 30% for InfoMin and 14% for SimCLRv2 on 'linear evaluation' and over 11% for InfoMin and 7% for SimCLRv2 on 'SVM.' BSSL also outperforms other baselines including the supervised pretraining. InfoMin and SimCLRv2 perform poorly compared to other SSL baselines as well.

Semi-Supervised Fine-tuning. As shown in Table 9, BSSL outperforms InfoMin and SimCLRv2 by large margins (*e.g.*, over 40% for InfoMin and almost 20% for SimCLRv2 in top-1 accuracy in 1% label setting). BSSL outperforms other SSL baselines by large margins as well. Interestingly, SimCLRv2 performs similarly to SWAV or BYOL for this particular task, possibly because of the deeper projection layer, which is used only in SimCLRv2, being better for semi-supervised learning.

Few-Shot Learning. The few-shot learning results are summarized in Table 10. Again, BSSL outperforms both InfoMin and SimCLRv2 by large margins across all metrics. BSSL also outperforms other baselines including the supervised pretraining in all shots.

A.3 Discussion on Comparison to Supervised Pretraining

Following the previous literature (Zbontar et al., 2021; Goyal et al., 2021; Tian et al., 2021a; Grill et al., 2020; Caron et al., 2020; He et al., 2020), we used the same amount of labeled and unlabeled data for supervised pretraining or BSSL. As such, BSSL outperforms supervised pretraining on the object detection task but not on all the other tasks we showed. This is not surprising as using the same amount of unlabeled data as the labeled data was not the design goal of SSL methods. Rather, SSL methods are built on the fact that a much larger unlabeled data is available for pretraining than labeled data. Thus, the loss of information from the lack of supervision can be made up with more quantity. However, few works (He et al., 2020) have shown such a comparison as it is practically difficult to utilize a larger unlabeled dataset (*e.g.* IG-1B) than the often used labeled ImageNet for supervised pretraining. To emulate a similar comparison where a larger unlabeled data is used for SSL compared to the labeled data used for supervised pretraining, we show comparisons of supervised pretraining on labeled ImageNet100 Xiao et al. (2021) and BSSL on unlabeled ImageNet. We mainly conduct downstream tasks where the target dataset changes such as SVM classification on Pascal VOC, SVM few-shot classification on Pascal VOC, and object detection on Pascal VOC.

We first summarize the results for the SVM classification and object detection in Table 11. Among the compared methods, BSSL shows the best performance for both tasks by large margins, outperforming supervised pretraining. Note that when a larger unlabeled data is used, SSL methods start outperforming supervised pretraining. This implies that if a larger unlabeled data is available SSL methods have an advantage over supervised pretraining.

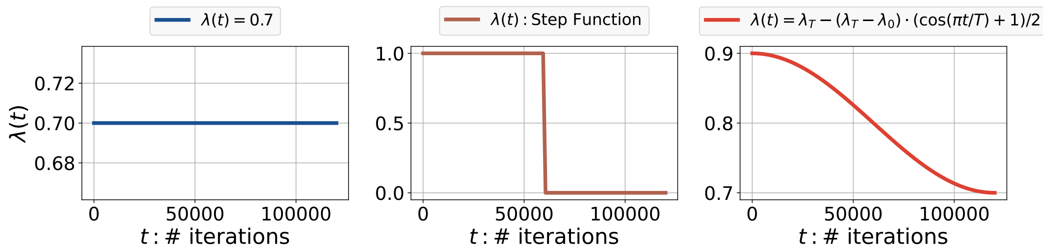
Pretrain On	Method	SVM Cls.	Object Detection		
		mAP (%)	mAP (%)	AP50 (%)	AP75 (%)
ImageNet100	Supervised	64.77	27.74	55.20	23.61
ImageNet	SWAV	71.23	37.22	67.30	35.91
ImageNet	BYOL	69.16	36.92	67.13	35.6
ImageNet	Tuned MoCov2	69.73	37.42	67.30	36.37
ImagetNet	BSSL (Ours)	**77.49**	**41.00**	**70.91**	**41.45**

Table 11: SVM classification (mAP) and object detection (mAP, AP50, AP75) for supervised pretraining on ImagetNet100 and SSL pretraining on ImageNet are shown. BSSL outperforms all other methods by large margins for both tasks. The best result for each metric is shown in **bold**.

Pretrain on	Method	k = 1	k = 2	k = 4	k = 8	k = 16	k = 32	k = 64	k = 96
ImgNet100	Supervised	22.18 ± 1.31	28.77 ± 1.97	36.59 ± 1.61	43.67 ± 0.93	50.61 ± 0.62	55.75 ± 0.43	59.39± 0.20	60.88± 0.41
ImagetNet	SWAV	22.97 ± 1.21	27.91± 2.37	37.91±1.11	44.5± 1.51	52.79± 0.81	59.15± 0.62	64.38± 0.59	66.72± 0.19
ImagetNet	BYOL	23.45 ± 0.76	28.04± 2.40	38.09± 1.07	44.69± 1.66	51.5± 0.90	57.44±0.24	62.07± 0.28	64.37± 0.13
ImagetNet	Tuned MoCov2	22.12 ± 0.74	27.45 ± 2.06	36.81 ± 0.82	43.19 ±1.4	51.93 ±0.84	57.95 ± 0.62	63.07 ± 0.43	65.15 ±0.05
ImagetNet	BSSL (Ours)	**29.20±1.51**	**36.14 ±2.15**	**48.49 ± 1.08**	**55.12 ± 1.59**	**62.36 ± 1.01**	**67.70 ± 0.3**	**72.1 ± 0.39**	**74.06 ± 0.18**

Table 12: Few-shot image classification using SVM (mAP) on the VOC07 dataset for supervised pretraining on ImagetNet100 and SSL pretraining on ImageNet are shown. The number of shots (k) is varied from 1 to 96 and the average over 5 runs with the standard deviation are reported. BSSL outperforms all methods including the supervised pretaining. The best result among SSL methods for each shot is shown in **bold**.

We then summarize the results for the SVM few-shot classificationin Table 12. Among the compared methods, BSSL shows the best performance for all shots by large margins, outperforming supervised pretraining. Note that even when a larger unlabeled data is used, other SSL methods only perform on par with supervised pretraining. Thus, when a larger unlabeled data is available other SSL methods do gain performance, but the performance of BSSL improves drastically more than other SSL. This implies that BSSL has advantages over other SSL methods when a larger unlabeled data is available for pretraining.



(a) Fixed (b) Step Function (c) Smooth Annealing

Figure 4: Plots for the various $\lambda(t)$ compared in Table 7 are shown.

A.4 Plots of Various $\lambda(t)$ in Table. 7

In Fig. 4, we plot the (1) fixed value, (2) an inverse shifted Heaviside function (a step function), and (3) the smooth annealing Eq. 3 to visualize the various choices of $\lambda(t)$ compared in Table 7.

A.5 Discussion about Shen et al. (2021)

While (Shen et al., 2021) claims to have used an SSL method to pretrain the FP model, we checked the official code and this is not the case. The FP model's pretrained weights include the classifier weights and the entire FP model is frozen during training, hence the pretrained classifier weights are used with no joint training as opposed to BSSL. Thus, to train the classifier's weights labeled data would have needed to be used, making (Shen et al., 2021) not applicable for the unsupervised case. On the other hand, the tuned MoCov2(Shen et al., 2021) results are applicable to the unsupervised case and we compare with this extensively.