Bayesian Optimization Meets Self-Distillation

HyunJae Lee* Heon Song* Hyeonsoo Lee* Gi-hyeon Lee Suyeong Park Donggeun Yoo
Lunit Inc.

Abstract

Bayesian optimization (BO) has contributed greatly to improving model performance by suggesting promising hyperparameter configurations iteratively based on observations from multiple training trials. However, only partial knowledge (i.e., the measured performances of trained models and their hyperparameter configurations) from previous trials is transferred. On the other hand, Self-Distillation (SD) only transfers partial knowledge learned by the task model itself. To fully leverage the knowledge gathered from all training trials, we propose the BOSS framework, which combines BO and SD. BOSS suggests promising hyperparameter configurations through BO and carefully selects pre-trained models from previous trials for SD, which are otherwise abandoned in the conventional BO process. BOSS achieves significantly better performance than both BO and SD in a wide range of tasks including general image classification, learning with noisy labels, semi-supervised learning, and medical image analysis tasks. Our code is available at https://github.com/sooperset/boss.

1. Introduction

Convolutional Neural Networks (CNNs) have achieved remarkable success in a wide range of computer vision applications [9, 31, 34]. However, their performance is greatly dependent on the choice of hyperparameters [11]. As the optimal hyperparameter configuration is not known a priori, practitioners often explore the hyperparameter space manually to obtain a better configuration. Despite its time-consuming process, it typically results in sub-optimal performance [7]. Recently, Bayesian optimization (BO) has emerged as a successful approach to hyperparameter optimization, automating the manual tuning effort and pushing the boundaries of performance [6, 24, 45]. BO allows for the effective exploration of multi-dimensional search spaces by suggesting promising configurations based on observations. This technique has achieved state-of-the-art performance for training CNNs [8, 15] and has contributed to improving various applications such as AlphaGo [10].

BO is inherently an iterative process in which a probabilistic prior model is fitted using observations of hyperparameter configurations and their corresponding performances [6, 24]. At each iteration, BO suggests the next configuration to evaluate that is most likely to improve performance. After training the network with the suggested configuration, a new observation is retrieved and used to update the probabilistic model. However, only partial knowledge (i.e., the measured performances of trained models and their hyperparameter configurations) from previous trials is transferred, and the knowledge learned by the network is discarded.

Self-distillation (SD) can also be viewed as a knowledge transfer method. A recent line of research in SD has demonstrated that transferring knowledge from a previously trained model with the identical capacity can improve the performance of the model [2, 16, 36]. If a student network is trained to mimic the feature distribution of a teacher network, then the student could beat the teacher. Allen-Zhu and Li [2] have both theoretically and empirically interpreted this as a similar effect to ensembling various models. Inspired by this property of SD and the iterative nature of BO, we ended up asking if the knowledge inside the network from the previous trials could be used for the next trials during the BO process.

In this paper, we propose a new framework, Bayesian Optimization meets Self-distillation (BOSS), which combines BO and SD to fully leverage the knowledge obtained from previous trials. The overall process of BOSS is illustrated in Figure 1. Following the BO process, BOSS suggests a hyperparameter configuration based on observations that are most likely to improve the performance. After that, it carefully selects pre-trained networks from previous trials for the next round of training with SD, which are otherwise abandoned in the conventional BO process. This process is performed in an iterative manner, allowing the network to persistently improve upon previous trials. To the best of our knowledge, this is the first work that leverages the knowledge of the network learned during the BO process. This
is not a simple combination of two orthogonal methods but we provide solutions to the problem of how to transfer past knowledge (i.e., model parameters, hyperparameters, and performances) appropriately. The suggested solution is that (1) not only the teacher but the student should be initialized from the prior knowledge, and (2) they should be initialized from different previous trials to fully exploit the prior knowledge. Our thorough ablation analysis supports this.

We evaluate the effectiveness of BOSS with various computer vision tasks, such as object classification [21], learning with noisy labels [20], and semi-supervised learning [3]. In addition, we also evaluate it with medical image analysis tasks, including medical image classification and segmentation. Our experimental results demonstrate that BOSS significantly improves target model performance, outperforming both BO and SD. Furthermore, we conduct comprehensive analysis and ablation studies to further investigate the behavior of BOSS.

In summary, the main contributions of this paper are:
1. We present BOSS framework that fully harnesses the knowledge from various models by leveraging the benefits of both BO and SD.
2. Exhaustive evaluation experiments demonstrate the efficacy of our framework, as it results in significant performance improvements across diverse scenarios.
3. In-depth analysis and ablation studies provide essential insights into how to transfer prior knowledge effectively for CNNs.

2. Related Work

**Bayesian Optimization.** Bayesian optimization (BO) is a global optimization method for black-box functions [6]. Given a set of hyperparameters $\lambda$ and an objective function $\Phi$ (e.g., accuracy), BO models the conditional probability $p(\Phi(\lambda)|\lambda)$ with a surrogate model. In each iteration, it fits the surrogate model given observations then uses the acquisition function to determine which configuration to evaluate next. By suggesting probable candidates based on observations, it has shown remarkable performance on various hyperparameter optimization tasks [6, 24, 45]. While other methods [38, 6] models $p(\Phi(\lambda)|\lambda)$, the Tree-structured Parzen Estimator (TPE) [6] models $p(\lambda|\Phi(\lambda))$ by partitioning hyperparameter density function into good and bad groups with respect to their corresponding objective values. It has linear complexity on the number of observations and enables scalable hyperparameter optimization. There have been various approaches to improve BO by incorporating prior knowledge. A line of works aims to transfer knowledge learned from other datasets through meta-learning. Multi-objective TPE [50] extends TPE to transfer knowledge from other tasks considering the similarity between tasks. Surrogate-Based Collaborative Tuning [5] leverages the knowledge from various datasets by integrating the surrogate ranking algorithm into BO. On the other hand, Prior-guided Bayesian Optimization [47] allows domain experts to transfer their knowledge into BO in the form of priors. Our framework is different from these methods in that these utilize prior knowledge from external sources to better estimate the surrogate model, while we utilize the prior knowledge from the given task to enhance the training of the target model. It means that these methods can be applied orthogonally to our framework.

**Self-Distillation.** Knowledge distillation (KD) is a model compression method that involves transferring the knowledge of a large teacher model to a small student model while maintaining performance. The original work by Hinton et al. [22] proposed distilling knowledge by matching the softmax distribution of the teacher and student models. Since then, various methods have been introduced to improve the knowledge transfer process. Self-Distillation (SD) is a special form of KD where the teacher and student networks have identical architecture. Born-Again Networks (BAN) [16] demonstrated that when training the student to match the output distribution of the teacher with the identical architecture, it could outperform the teacher. Furthermore, they showed that performing multiple rounds of BAN could further improve the performance where the trained student is set to be a new teacher in the following round. The ef-
fectiveness of SD has been theoretically explained by the “multi-view” hypothesis introduced by Allen-Zhu and Li, who showed that self-distillation performs an implicit ensemble of various models [2]. Empirical evidence from Pham et al. [36] suggests that SD encourages the student to find flatter minima, leading to better generalization. In this work, we identify that SD can be an effective method for propagating the task knowledge learned in early stages of BO, to late stages of BO. This combination of the SD and BO processes is key to yielding a high-performing model, which we validate experimentally.

3. Method

In this section, we introduce BOSS which combines BO and SD to leverage the knowledge learned in the training trials. We first provide a brief introduction to BO and SD then introduce BOSS in detail.

3.1. Bayesian Optimization

BO is an iterative process that aims to optimize an objective function $\Phi$ with respect to a set of hyperparameters $\lambda$ [24]. At each iteration $t$, BO builds a surrogate model to approximate $\Phi$ based on the observations from the previous iterations, denoted by $\Lambda_{t-1} = \{(\lambda_0, \Phi(\lambda_0)), \ldots, (\lambda_{t-1}, \Phi(\lambda_{t-1}))\}$. BO then uses an acquisition function $\mu(\lambda|\Lambda_{t-1})$ to select the next hyperparameter configuration $\lambda_t$ to evaluate. The acquisition function balances exploration and exploitation, with the most common choice being the Expected Improvement (EI) [25]:

$$
\mu(\lambda|\Lambda) = \mathbb{E}(\max\{0, \Phi(\lambda) - \Phi_{\text{max}}\}|\Lambda),
$$

where $\Phi_{\text{max}}$ is the best observed function value. The hyperparameter configuration at iteration $t$ is selected as $\lambda_t = \arg\max_{\lambda} \mu(\lambda|\Lambda_{t-1})$. After evaluating $\Phi(\lambda_t)$, the new observation is added to the previous ones, $\Lambda_t = \Lambda_{t-1} \cup \{(\lambda_t, \Phi(\lambda_t))\}$, and the surrogate model is updated.

While many surrogate models including the Gaussian process (GP) directly models $p(\Phi(\lambda)|\lambda)$, the Tree-structured Parzen Estimator (TPE) [6] models $p(\lambda|\Phi(\lambda))$ which is defined by two functions:

$$
p(\lambda|\Phi(\lambda)) = \begin{cases} 1(\lambda), & \Phi(\lambda) > \Phi^* \\ g(\lambda), & \Phi(\lambda) \leq \Phi^* \end{cases}.
$$

Here, $1(\lambda)$ is the “good” density formed by observations that the performance was higher than a threshold $\Phi^*$. $g(\lambda)$ is the “bad” density formed by the remaining observations. Bergstra et al. [6] claims that maximizing EI is proportional to maximizing the ratio of $1(\lambda)/g(\lambda)$. Hence, on iteration $t$, TPE suggests a configuration $\lambda_t$ that maximizes this ratio. Despite its simplicity, TPE has outperformed traditional surrogate models such as GP [38].

3.2. Self-Distillation

Furlanello et al. [16] proposed the Self-Distillation (SD) technique for transferring knowledge from a teacher network to a student network with the same architecture. Given a neural network $f$ initialized with random parameters $\theta$, SD first obtains the parameters of the teacher model $\theta^T$ by minimizing the task loss $L_{gt}(f(X; \theta), Y)$ with respect to $\theta$. SD then obtains the parameters of the student model $\theta^S$ by minimizing a loss with respect to $\theta$ that balances the task loss $L_{gt}$ and the distillation loss $L_{dt}$ as follows:

$$
\alpha L_{gt}(f(X; \theta), Y) + (1 - \alpha) L_{dt}(f(X; \theta), f(X; \theta^T)) \tag{3}
$$

where $\alpha$ is a hyperparameter for balancing the two losses.

In various image classification tasks, the cross-entropy loss is used to define the task loss $L_{gt}$ as $L_{gt}(f(X; \theta), Y) = -\sum Y \log f(X; \theta)$, where $\sigma$ is the softmax function. For the distillation loss, in conventional self-distillation, $L_{dt}$ is defined using the Kullback-Leibler (KL) divergence loss. However, recent work [28] has shown that the mean squared error (MSE) between logits from the student and teacher outperforms the KL divergence loss. The MSE distillation loss is defined as $L_{dt}(f(X; \theta), f(X; \theta^T)) = ||f(X; \theta^T) - f(X; \theta)||^2$.

3.3. BOSS Framework

BOSS integrates SD into the BO process in order to incorporate the knowledge learned in former trials. While the proposed framework is applicable to various BO and SD methods (see Section 5), we opt for TPE [6] due to its low computational complexity and scalability to diverse search spaces. In addition, we utilize MSE for our distillation loss as it has been shown to yield better performance than KL divergence while requiring fewer hyperparameters [28]. The overall procedure of BOSS is summarized in Algorithm 1.

In order to perform SD, a teacher network is required to train a student network. However, the absence of any network, in the beginning, poses a cold start problem. To address this issue, a warm-up phase is introduced, which is similar to the regular BO process but with the difference that the trained CNNs are recorded for use in subsequent phases. In the warm-up phase, the neural network is initialized with random parameters $\theta$ and a hyperparameter $\lambda$ is suggested using the acquisition function $\lambda = \arg\max_{\lambda} \mu(\lambda|\Lambda)$ in Equation 1. Training images $X$ and their corresponding labels $Y$ are then used to obtain new parameters $\theta^*$ for $\lambda$ by minimizing the task loss $L_{gt}$. Any task-specific loss can be used such as the cross-entropy loss used in image classification tasks. After training, the performance of the neural network $\Phi(\lambda)$ is computed and the observation $(\lambda, \Phi(\lambda))$ is
Algorithm 1: Optimization Process for BOSS

**Input:** Training data \((X, Y)\); number of total trials \(N\); number of warm-up trials \(W\); number of candidates \(K\); neural network \(f\); balance parameter \(\alpha\).

**Initialize:** Sets of observations \(\Lambda_0, \Lambda'_W = \emptyset\); set of parameters \(\Theta_0 = \emptyset\).

**Phase 1: Warm-up**

for \(t = 1, 2, \ldots, W\) do

- Draw \(\theta_t\) from the standard normal distribution
- Find \(\lambda_t\) that maximizes \(\mu(\lambda_t|\Lambda_{t-1})\)
- Obtain \(\theta_t^*\) for \(\lambda_t\) that minimizes \(L_{gt}\) w.r.t \(\theta_t\):

\[
\theta_t^* = \arg\min_{\theta_t} L_{gt}(f(X; \theta_t), Y)
\]

- Update \(\Lambda_t: \Lambda_t = \Lambda_{t-1} \cup \{\lambda_t, \Phi(\lambda_t)\}\)
- Update \(\Theta_t = \Theta_{t-1} \cup \{\theta_t^*\}\).

end

**Phase 2: BOSS Training**

for \(t = W + 1, W + 2, \ldots, N\) do

- Randomly select \(\theta_t^S, \theta_t^T \in \text{TOP-}\(K\)(\(\Theta_{t-1}\))
- Find \(\lambda_t\) that maximizes \(\mu(\lambda_t|\Lambda_{t-1})\)
- Obtain \(\theta_t^*\) for \(\lambda_t\) that minimizes Eq. 3 w.r.t \(\theta_t^S\):

\[
\theta_t^* = \arg\min_{\theta_t^S} \alpha L_{gt}(f(X; \theta_t^S), Y) + (1 - \alpha)L_{dt}(f(X; \theta_t^S), f(X; \theta_t^T))
\]

- Update \(\Lambda_t: \Lambda_t' = \Lambda_{t-1} \cup \{\lambda_t, \Phi(\lambda_t)\}\)
- Update \(\Theta_t = \Theta_{t-1} \cup \{\theta_t^*\}\).

end

**Output:** \(\theta_{\text{best}} \in \Theta_N\) showing the best performance.

added to a set of observations \(\Lambda\). Also, the obtained parameters \(\theta^\ast\) are added to a set of parameters \(\Theta\).

In the second phase where BOSS training is performed, an empty set is initialized as a new observation set \(\Lambda'\), because the training scheme differs from the warm-up phase and the optimal configuration is likely to be different. Given the set of parameters \(\Theta\) and the pre-defined number of candidates \(K\), the top-\(K\) candidates are selected according to the performances in \(\Theta\). Among them, two are randomly chosen again to define a student and a teacher. We initialize the teacher \(\theta^T\) and student \(\theta^S\) with different parameters as we expect some benefit from aggregating different knowledge from different networks. The effect of this scheme is elaborated further in Section 5.2. After the initialization, the hyperparameter \(\lambda\) is suggested in the same procedure as in the warm-up phase. Unlike the warm-up phase, \(\theta\) is trained to minimize the loss in Equation 3. Once training is complete, the performance and trained parameters are added to \(\Lambda'\) and \(\Theta\), respectively. This procedure is repeated until the budget \(N\) is exhausted. Despite potential concerns that the utilization of pre-trained weights could interfere with BO’s modeling, the previous study [27] has demonstrated that BO is sufficiently robust to the variability of initial weights by randomly initializing them at each trial.

As BOSS training proceeds, the top-\(K\) models are updated with the models that incorporate the knowledge of previous trials. These models are then utilized to transfer the knowledge in the follow-up iterations. By iteratively updating the candidates with enhanced networks, BOSS consistently improves performance upon previous trials.

4. Experiments

In this section, we present a comprehensive evaluation of BOSS on a wide range of problems and datasets. We compare it with several other methods in our comprehensive evaluation, including Baseline (conventional training with hyperparameters tuned by human experts), SD (Self-Distillation with MSE distillation loss [28]), Random (Random search for hyperparameter optimization [7]), and BO (Bayesian Optimization with TPE [6]). Motivated by the setting in [32], we define our search space to include learning rate \(l\), momentum \(1 - m\), weight decay \(w\) and batch size \(b\) where \(l, m\) are sampled from \(\log[10^{-3}, 1]\), \(w\) from \(\log[10^{-5}, 10^{-2}]\), and \(b\) from \([64, 256]\). We use an implementation of TPE from the Optuna framework [1] and conduct 128 trials with parallel execution of 8 trials. \(K\) and \(W\) are set to 8 and 32 respectively which will be further explored in Section 5.1.

In order to ensure a fair comparison with BOSS, we conduct multiple rounds of SD until it reaches saturation and report the highest accuracy obtained among all the rounds. Specifically, we continue the SD process until there is no further improvement in performance for three consecutive rounds. We set the hyperparameter \(\alpha = 0.5\) for SD, which we find to yield the best performance among \(\alpha \in \{0.25, 0.5, 0.75\}\) on CIFAR-100 [29] with the VGG-16 [44] architecture. However, we empirically observe that the final performance does not vary significantly when the SD process is continued until saturation.

4.1. Object Classification

We first evaluate the effectiveness of BOSS in object classification tasks. Publicly available implementations\(^2\) of classification networks and training procedures are utilized to fairly compare it with the human baseline. We use three standard datasets: CIFAR-10 and CIFAR-100 [29], which are comprised of 50,000 training and 10,000 test images of 10 and 100 object classes, respectively, and Tiny ImageNet

\(^2\)https://github.com/bearpaw/pytorch-classification
which includes 100,000 training and 10,000 validation images across 200 object classes. Consistent with previous works [32, 53], we resize the images in the Tiny ImageNet dataset to 32x32 pixels to align with the training procedure of the CIFAR datasets.

In our experiments, the networks are trained using the stochastic gradient descent (SGD) optimizer for 164 epochs on a single GPU. The baseline hyperparameters include a learning rate of 0.1, a weight decay of 5e-4, a momentum of 0.9, and a batch size of 128. We follow the standard practice [21] for training-time data augmentation by zero-padding each image with 4 pixels, randomly cropping it to the original size, and performing evaluations on the original images.

**Performance on different datasets.** We train the VGG-16 architecture [44] on the aforementioned three datasets. Table 1 compares the top-1 accuracy with the average and the 95% confidence interval over 5 repetitions. It demonstrates that BOSS exhibits considerably higher accuracy than other methods across all tested datasets. While random search succeeds to improve the performance of the baseline, BO further boosts the performance by adaptively suggesting probable configurations. SD also achieves enhanced performance compared to the baseline as expected. However, the effectiveness of SD and BO varies across datasets. On the other hand, BOSS consistently improves the performance by a large margin, leveraging the advantages of both methods.

**Comparison of diverse methods.** We compare the performance of BOSS with other methods in terms of the number of trials. Figure 2 illustrates the performance curves of Random, BO and BOSS along with the Baseline and SD on CIFAR-100 with VGG-16. Throughout the optimization process – except for the initial warm-up phase – BOSS consistently exhibits superior performance over other methods. Initially, all methods display similar performance due to the lack of adequate observations, and BO outperforms random search by exploring promising configurations using previous observations with an increasing number of trials. However, after a certain point, the performance improves at a slow pace. In contrast, BOSS persistently improves upon previous trials by leveraging prior knowledge as a stepping stone to push the performance boundary. This suggests that BOSS’s efficacy is not easily saturated, making it worthwhile for future investigation.

**Scalability to various architectures.** We demonstrate the scalability of BOSS with respect to a variety of CNN architectures including AlexNet [30], VGG-11 [44] and ResNet-20/56 [21] on the CIFAR-100 dataset. As shown in Table 2, BOSS consistently outperforms all other methods across all tested architectures. Notably, even for small networks such as ResNet-20, BOSS achieves superior performance compared to the baseline of larger ResNet-56 with more than three times as many parameters. This finding suggests that BOSS enables efficient network design by achieving better performance with fewer parameters. Moreover, it further brings additional performance improvement to large networks, implying that BOSS is not simply substituting the performance gain by increasing the capacity of the CNN architecture.

**Performance with enhanced baseline.** Pham et al. [36] criticize the existing literature on SD, saying that the per-
performance of reported baselines is not fully saturated. In such cases, the gains achieved through distillation may be invalidated if the baseline is better optimized. Following their suggestion, we incorporate the advanced data augmentations of AutoAugment [12] and Cutout [14]. We train VGG-16 on CIFAR-100 using the aforementioned regularizations and report the top-1 validation accuracy in Table 3. As anticipated, the updated data augmentations yield a substantial performance improvement over those in Table 1. Nonetheless, BOSS consistently outperforms other methods by a considerable margin, indicating that the improvements achieved by BOSS are not hindered by better training schemes such as advanced regularization methods.

### 4.2. Learning with Noisy Labels

We further demonstrate the benefit of BOSS in noisy label settings where incorrect labels exist in training data. In real-world scenarios, the presence of noisy labels often hinder the performance of CNN models [46]. To address this challenge, previous research [2, 26] has demonstrated that employing distillation methods could mitigate the impact of noisy labels by incorporating distinct aspects from multiple teachers to the student. BOSS builds upon this approach by iteratively transferring different knowledge obtained from previous trials.

We conduct experiments to show the robustness of BOSS by training VGG-16 on CIFAR-100 following the training procedure in Section 4.1. Table 4 shows the top-1 accuracy with respect to varying noise ratios and types where symmetric noise refers to the situation where all labels are mixed with the same probability of noise, while asymmetric noise flips labels based on similar class pairs. As expected, the performance of the baseline drops significantly compared to training with the clean dataset in Table 1. While both SD and BO improve the performance of the baseline to some extent, they are sensitive to noise and exhibit performance decreases under large noise conditions. On the other hand, BOSS consistently achieves significant performance improvement regardless of noise levels and types which suggests that the negative influence of label noise can be greatly mitigated by the proposed framework.

<table>
<thead>
<tr>
<th>Method</th>
<th>Symmetric</th>
<th>Asymmetric</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20%</td>
<td>40%</td>
</tr>
<tr>
<td>Baseline</td>
<td>64.79</td>
<td>53.34</td>
</tr>
<tr>
<td>SD</td>
<td>68.93</td>
<td>59.58</td>
</tr>
<tr>
<td>BO</td>
<td>65.47</td>
<td>57.52</td>
</tr>
<tr>
<td>BOSS (Ours)</td>
<td><strong>70.34</strong></td>
<td><strong>65.50</strong></td>
</tr>
</tbody>
</table>

Table 4. Comparison with Different Noise Rates on CIFAR-100

### 4.3. Semi-Supervised Learning

We evaluate BOSS in the context of semi-supervised learning (SSL), where a large amount of unlabeled data is available. Previous works [3, 37, 48, 51] have demonstrated that transferring past knowledge via self-training is an effective approach for utilizing unlabeled data. However, it generally does not exploit the knowledge from various models. We investigate whether BOSS could further enhance the performance by incorporating the knowledge of multiple high-performing models yielded during BO process.

We leverage one of the state-of-the-art SSL method [3] as a baseline and adopt the official implementation\(^3\). We opt for 13-CNN architecture [4] which is mainly explored in the paper. Table 5 presents the top-1 accuracy of CIFAR-100 dataset with 4,000 and 10,000 labeled data. It is worth noting that the baseline is trained with carefully tuned hyperparameters, advanced regularization techniques, and long training iterations. Nevertheless, BOSS consistently improves the accuracy with a meaningful margin. This result indicates that BOSS could generate a positive synergy with existing SSL algorithm, complementing it by leveraging the knowledge from various models.

### 4.4. Medical Image Analysis Tasks

The significance of medical image analysis tasks lies in their direct impact on patient outcomes, making their performance a critical concern. One approach to enhance performance in these tasks is to employ multiple models, like model ensembles [23, 19]. However, this can be limited by resource constraints in real-world environments such as hospitals. To tackle this challenge, researchers have turned to HPO techniques such as BO, despite their substantial costs, to improve model performance [39, 17]. Therefore, in this section, we show the efficacy of BOSS on two critical medical image analysis tasks.

The first task we evaluate is breast cancer classification using mammograms, which are x-ray images of the breast used for early cancer detection. Accurate mammographic breast cancer classification is essential, being the primary detection method and the second most common cancer in women. We use two datasets for breast cancer classification with mammograms: the publicly available Chinese Mammography Database [13] (CMMD, 826 exams), and the in-house dataset sourced from the European

\(^3\)https://github.com/EricArazo/PseudoLabeling
Union (INH, 6,994 exams). We split each dataset into training sets and validation sets in a 7:3 ratio. We use ResNet-34 to classify mammograms into two categories: cancer or non-cancer. The mammograms are resized to 1920×1280 and applied with various geometric and photometric augmentations, such as translate, rotate, shear, flip, and brightness/contrast adjustment. The model’s performance is evaluated using the Area under the ROC Curve (AUC)\(^4\).

Next, we address the task of nuclei instance segmentation, which involves identifying and segmenting individual nuclei in histopathology images. This task is particularly crucial in medical research, as it can assist in the diagnosis and treatment of diseases like cancer. To evaluate the proposed method, we conduct experiments on two datasets: CoNSeP \(^1\) (24,319 instances) and CPM-17 \(^2\) (7,570 instances). To ensure a fair comparison, we train the HoVer-Net \(^3\) architecture, which is a standard architecture in the task, using the official repository and training schemes. The hyperparameter configuration recommended in the original implementation serves as the baseline. We use the multi-class Panoptic Quality (mPQ) \(^4\) metric to evaluate the performance of the model, with higher scores indicating better performance.

Table 6 shows that BOSS outperforms both the baseline and BO methods, achieving enhanced performance on both tasks. This demonstrates that BOSS can efficiently utilize the limited medical image data without any additional overhead at test time via an implicit ensemble of multiple models and superior hyperparameter configuration. It is consistent with the previous findings that have shown model ensembles \(^5\) and hyperparameter optimization \(^6\) to be effective when data is limited. Moreover, the result further shows that the advantage of BOSS is not restricted to classification tasks but could be readily extended to instance segmentation tasks as well. Further extension of BOSS to other tasks such as object detection \(^7\) or generative modeling \(^8\) remains as future work.

### 5. Ablation Study and Analysis

This section presents ablation studies and analytical experiments to investigate the design choices of BOSS algorithm. We conduct these studies using the same experimen-

---

\(^4\) The exam-level AUC \(^9\) is calculated, and only exams containing a complete set of 4-view images are considered.

\(^5\) Applied with various geometric and photometric augmentations, such as translate, rotate, shear, flip, and brightness/contrast adjustment.

\(^6\) https://github.com/vqdang/hover_net

---

![Figure 3. Top-1 accuracy of BOSS on CIFAR-100 with varying the number of (left) warm-up trials and (right) candidates.](image)

<table>
<thead>
<tr>
<th>BO</th>
<th>PT Student</th>
<th>PT Teacher</th>
<th>Top-1 Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>76.48</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>76.93</td>
</tr>
<tr>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>77.18</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>77.69</td>
</tr>
</tbody>
</table>

Table 7. Comparison of different choices for utilizing pretrained weights on CIFAR-100 with VGG-16. “PT” stands for “Pretrained”.

---

The exam-level AUC \(^9\) is calculated, and only exams containing a complete set of 4-view images are considered.

\(^7\) Applied with various geometric and photometric augmentations, such as translate, rotate, shear, flip, and brightness/contrast adjustment.

\(^8\) https://github.com/vqdang/hover_net
to even greater performance gains. This indicates that the knowledge of teacher and student could generate a positive synergy.

5.2. Effect of Pretrained Weight

In light of our observations that utilizing pretrained weights for both student and teacher networks leads to a improved performance, we note that the Noisy Student [51] reported no improvement in performance when initializing both models simultaneously. They claim that this approach may sometimes lead to getting stuck in a local optima, resulting in inferior performance compared to training the student model from scratch. We hypothesize that this contrasting result could be due to the fact that they used the same pretrained weight for both teacher and student. Since the student model has already inherited all the knowledge that could be learned from the teacher, distilling knowledge with the teacher may not bring additional performance gains.

A recent study by Allen-Zhu and Li [2] revealed that different networks trained with distinct random seeds learn different knowledge. Motivated by this observation, we train VGG-16 on CIFAR-100 with 8 different seeds following the same procedure as the baseline in Section 4.1 and utilize them to initialize teacher and student networks. For all possible combinations of teacher and student networks, we conducted a single-round self-distillation, comparing the top-1 accuracy of the trained models between identical and asymmetric initialization. The identical models refer to initializing both teacher and student with the same pretrained model, while asymmetric models refer to initializing them with different models. The results are presented in Figure 4, which shows boxplots of top-1 accuracy with different initialization types. As expected, utilizing distinct pretrained models for the teacher and student network leads to better performance. Our analysis suggests that it is crucial to initialize the student and teacher network with different models to benefit from warm-starting the student model.

5.3. Different Bayesian Optimization Methods

We verify the effectiveness of BOSS with respect to various BO methods. We perform additional experiments with Gaussian Process (GP) [38] and Sequential Model-based Algorithm Configuration (SMAC) [24]. We utilize publicly available implementations [33] of these methods and adopt their default settings. Table 8 presents the results of our experiments, showing that all BO methods improve the performance of the baseline (i.e. 74.43). BOSS further brings considerable performance improvement in all methods, demonstrating its effectiveness across different BO algorithms.

5.4. Various Distillation Techniques

We finally evaluate the scalability of BOSS by examining its performance with different distillation techniques. Specifically, we further investigate the efficacy of BOSS with three popular distillation methods: Kullback-Leibler (KL) divergence [22], FitNets [41], and Attention [54]. We follow the suggestion in the original papers for the additional hyperparameters introduced by each distillation loss. Table 9 shows the results of the various distillation methods. We observe that SD successfully enhances the performance of the baseline (i.e. 74.43) across all tested distillation methods, and BOSS further improves the performance significantly. These findings demonstrate that the benefit of BOSS is not limited to specific distillation methods but can be extended to various techniques.

6. Conclusion

In this paper, we present BOSS framework, which combines Bayesian Optimization (BO) and Self-Distillation (SD) to fully leverage the model, hyperparameter configuration, and performance knowledge acquired during the BO process, across all trials. Through extensive experiments in various settings and tasks, we demonstrate that BOSS achieves significant performance improvements, that is consistently better than standard BO or SD on their own. BOSS
does not impose any additional overhead at test time and is versatile in that it is applicable to different BO and SD methods. Based on the presented evidence, we believe that the concept of marrying BO and SD is a powerful approach to training models, that should be further explored by the research community.

References


[48] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. NeurIPS, 2017. 6


[51] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In CVPR, 2020. 6, 8

