Weight Decay Scheduling and Knowledge Distillation for Active Learning

Juseung Yun, Byungjoo Kim, and Junmo Kim

KAIST, South Korea
{juseung_yun,junmo.kim}@kaist.ac.kr
byungjoo.kim92@gmail.com

Abstract. Although convolutional neural networks perform extremely well for numerous computer vision tasks, a considerably large amount of labeled data is required to ensure a good outcome. Data labeling is labor-intensive, and in some cases, the labeling budget may be limited. Active learning is a technique that can reduce the labeling required. With this technique, the neural network selects on its own the unlabeled data most helpful for learning, and then requests the human annotator for the labels. Most existing active learning methods have focused on acquisition functions for an effective selection of the informative samples. However, in this paper, we focus on the data-incremental nature of active learning, and propose a method for properly tuning the weight decay as the amount of data increases. We also demonstrate that the performance can be improved by knowledge distillation using a low-performance teacher model trained from the previous acquisition step. In addition, we present a novel perspective of the weight decay, which provides a regularization effect by limiting the number of effective parameters and channels in the convolutional filter. We validate our methods on the MNIST, CIFAR-10, and CIFAR-100 datasets using convolutional neural networks of various sizes.

Keywords: active learning, weight decay, knowledge distillation

1 Introduction

Deep convolutional neural networks (CNNs) have shown a significant in several computer vision tasks [9,34,29]. However, a large amount of data is needed to ensure the desirable performance of such networks. Unfortunately, because the process requires considerable manual effort, it is occasionally difficult to collect a sufficient amount of labeled data. In certain cases, the budget allowed for labeling may be limited. Active learning can be used to mitigate the aforementioned problem. The goal of active learning is to minimize the labeling budget, while achieving maximum performance. In this paper, we focus on pool-based active learning. The process of pool-based active learning starts with completely unlabeled data. While training, the neural network selects a set of data and requests the human annotator for their labels. The network is then trained using these

labels. Iterative training proceeds until it meets certain constraints, such as the labeling budget, or the desired performance. Because the selection of data notably affects the performance, it is important to select those *informative* samples that are most helpful for training a model from unlabeled pools.

In the existing literature, there are three major approaches used to determine an informative sample, namely, uncertainty-based methods [26,27,18,6], distribution-based methods [37], and expected model change methods [5,19,49]. Because it is infeasible and redundant to train neural networks from scratch every time we obtain additional labels, a rounding setting or batch mode setting is widely used, where the neural network requests labels for multiple data at the same time.

The majority of existing active learning methods focus on sampling the *most informative* samples by finding the acquisition functions. However, in our approach, we address the problem by considering the data incremental nature of active learning. Few annotations are accessible during the early stage of training, and the network is easily over-fitted to the annotated data; this degrades the generalization performance. Therefore, a strong regularization is necessary at the beginning of the training to prevent over-fitting. As the training progresses, we obtain additional data, and the total amount of annotated data increases. If we maintain the strong regularization of the first round during the subsequent rounds, the performance decreases because the network cannot learn new information in an efficient manner. The more data we have, the weaker the regularizer needed to ensure that the model can learn sufficient information. Thus, in active learning, the number of data continues to increase and regularization must be performed in a planned manner.

To this end, we propose a novel method for active learning. We control the regularization effect by scheduling the weight decay as the amount of data increases. We show that reducing the weight decay inversely proportional to the amount of data is simple but extremely effective. We also demonstrate that initializing the network parameters randomly and learning from scratch during every round is better than initializing the model trained during the previous round. However, if we re-initialize the network during each round, we are unable to use any information previously trained by the model. As an alternative, we use the method of distilling knowledge to a new model, in which the model from the previous round acts as a teacher. We show that the proposed method is effective in various CNN architectures and datasets.

Contribution. The three main contributions of our study are as follows.

1. We propose a weight decay scheduling method for CNNs with batch normalization during active learning to reduce the weight decay inversely proportional to the number of training data. We also verified that the weight decay method is effective in batch normalization because it regulates the model complexity by adjusting the number of effective parameters and convolutional channels.
2. We show that a network with a low performance and effective capacity can distill useful knowledge to networks with a higher performance and effec-

tive capacity. When training a new model after an acquisition, even distilling knowledge from a previous model shows a better performance than the training alone.
3. We also verified that our method shows an improved performance on the MNIST, CIFAR-10, and CIFAR-100 datasets when using various CNN architectures with batch normalization.

2 Related Work

Active learning. Classic active learning methods can be categorized into the following three types: an uncertainty-based approach [26,27,18], a diversity-based approach [33,4,8], and expected model change-based methods [5,19,39]. In an uncertainty-based approach, entropy [38,30,18], max margin [36], or distance to the decision boundary [43,45] are used as a proxy for the uncertainty. With diversity-based approaches, diverse samples are selected that represent the distribution of an unlabeled data space [33]. The method of an expected model change estimates an expected gradient length or optimal model improvement [5,19]. In addition, active learning methods have been applied in deep neural networks. Through an uncertainty-based approach, Wang *et al.* [46] used entropy in deep neural networks, and demonstrated its effectiveness. Gal *et al.* [6] estimated the uncertainty through a Monte Carlo dropout. Beluch *et al.* [2] used an ensemble of CNNs for estimating uncertainty. Sener *et al.* [37] proposed a distribution-based approach, in which the core-set covering the feature space of the entire unlabeled pool is selected. Fianlly, Yoo *et al.* [49] proposed a loss prediction module and selected samples that are expected to have the highest loss.

However, in most studies, the incremental data characteristics of active learning are not considered and the model is trained using the same hyper-parameters regardless of the increase in the amount of data [13,46,2,37,49]. Some studies have used a method to reduce the weight decay as the amount of data increases [21,6]. However, rather than adjusting the weight decay with the rules, the authors heuristically identified an appropriate weight decay during each round and used CNN without batch normalization.

Weight decay. Weight decay is a regularization method that prevents overfitting by limiting the complexity of parametric models in machine learning [24]. It is also used in modern deep neural networks and acts as a good regularizer [9,14,41]. However, when weight decay is used with batch normalization [16], the output of each layer is re-scaled, and the weights can be scaled using a small factor without changing the network's prediction. Therefore, Van *et al.* [44] suggested that the weight decay does not effectively limit the complexity of the network. Furthermore, Zhang *et al.* [51] indicated that the increase in the effective learning rate leads to a larger gradient noise, which in turn acts as a stochastic regularizer [32,20,17], which the authors argue is reason why a weight decay leads to performance gains even when batch normalization is used.

Knowledge distillation. Knowledge distillation is a method for transferring knowledge to a small student network using a high-capacity model or an ensemble

of multiple models as the teacher. The student is trained to imitate the teacher model in the form of class probabilities [11], feature representation [1,35,22,10], attention map [50] or inter-layer flow [48]. Most existing methods use highly complex or high capacity models as the teacher and aim at obtaining small and fast models. Rometo *et al.* [35] used a student network deeper than that of the teacher. However, the numbers of channels and parameters in the student model are smaller than those of the teacher.

Recently, Xie *et al.* [47] successfully distilled knowledge to a larger student model. However, when training the student model, they used strong regularizers such as RandAugment data augmentation [3], a dropout [42], and stochastic depth [15]. Therefore, the performance gain was predictable to a certain extent. By contrast, our method has a large weight decay for the teacher and a small weight decay for the student. In other words, in our study, we analyze a case in which the student has a more effective capacity and is trained with no additional regularizers such as RandAugment [3].

3 Method

In this section, we describe the active learning problem, along with our proposed method. In Section 3.1, we present an overview of the entire active learning procedure. Next, we introduce our weight decay scheduling method in Section 3.2, and knowledge distillation in Section 3.3.

3.1 Entropy-based Active Learning

We consider a C-class classification problem. In addition, following previous studies [2,37,49,31], we consider active learning in a batch setting, i.e, the neural network requests labels for multiple data at the same time. During initial round, we randomly sample K data points from an unlabeled data pool $\mathcal{D}$, and ask a human oracle to annotate them to create an initial labeled dataset $\mathcal{D}_0^L = \{(\boldsymbol{x}, y)\}^K$, where $\boldsymbol{x} \in \mathcal{X}$ is an input, and $y \in \mathcal{Y} = \{1, 2, ..., C\}$ is its label. The subscript 0 denotes the initial round. Let $\mathcal{D}_0^U = \mathcal{D} \setminus \mathcal{D}_0^L$ be the rest of the unlabeled pool. Subsequently, we train the initial CNN, $f_0(\boldsymbol{x}; \boldsymbol{\theta})$, using the labeled dataset $\mathcal{D}_0^L$ with a weight decay λ_0. After the initial training, we evaluate all data points in the unlabeled pool $\mathcal{D}_0^U$ using an acquisition function $a\left(\mathcal{D}^U, f\right)$. Finding a good acquisition function was not the focus of our study, however, and we therefore used entropy as $a\left(\mathcal{D}^U, f\right)$, which is simple but achieves a good performance [46]. Once the network f outputs a probabilistic class posterior $p(c|\boldsymbol{x}, \boldsymbol{\theta})$ for a sample $\boldsymbol{x}$ over a class c, the entropy $H(p)$ can be calculated as follows:

$$H(p) = -\sum_{c=1}^{C} p(c|\boldsymbol{x}, \boldsymbol{\theta}) log\left(p(c|\boldsymbol{x}, \boldsymbol{\theta})\right). \tag{1}$$

Subsequently, we select K samples with the largest entropy from $\mathcal{D}_0^U$ and request the human annotator for labels. The labeled dataset is then updated and $\mathcal{D}_1^L$

is obtained. Next, we train the CNN $f_1(\boldsymbol{x};\boldsymbol{\theta})$ over $\mathcal{D}_1^L$, and select K samples with the largest entropy from the $\mathcal{D}_1^U = \mathcal{D}\backslash\mathcal{D}_1^L$. We repeat this process until the desired performance is achieved or the labeling budget is exhausted.

3.2 Weight Decay Scheduling

With active learning, the amount of training data in the first round is small and increases as the learning progresses. An over-fitting is likely to occur as the amount of data is initially small, and therefore, a strong regularizer is needed in the initial stages of training. By contrast, the more data we have, the weaker the regularizer needed to ensure that the model can learn sufficient information. If we keep using a strong regularizer through the training, the network suffers during the learning. We propose a method for controlling the degree of regularization by scheduling a weight decay in the network, which contains a batch normalization layer [16].

Assume that we train a neural network with N_1 labeled samples to minimize the cross-entropy loss, $\mathcal{L}_{CE}$, and L_2 weight decay. The total loss, $\mathcal{L}$, is then expressed as follows:

$$\mathcal{L} = \mathcal{L}_{CE} + \frac{1}{2}\lambda_1 \left\|\boldsymbol{\theta}\right\|_2^2, \tag{2}$$

where λ denotes the weight decay parameter. Because we train the network with a mini-batch, kN_1 iterations are required for the entire training process. Here, k is a proportional constant, which is determined by the size of the mini-batch and the total number of epochs. The optimizations over $\mathcal{L}_{CE}$ and $\left\|\boldsymbol{\theta}\right\|_2^2$ are closely related. In general, reducing the L_2 norm to a significant extent will not effectively reduce the cross-entropy loss, and vice versa. However, when the network contains a batch normalization layer, it is shown that the weights can be scaled by a small factor without changing the prediction of the network [44,51,28,12]. Formally, let $\boldsymbol{\theta}_l$ be the learnable weights for a convolutional layer. Assume that the output of the layer feeds in to a batch norm layer, and let $\mathbf{BN}\left(\boldsymbol{x};\boldsymbol{\theta}_l\right)$ denote the output of that batch norm layer. Suppose that, as a result of a L_2 penalty term, we scale $\boldsymbol{\theta}_l$ by a factor of $0 < \kappa < 1$. Then, the new output of the batch norm layer is as follows:

$$\mathbf{BN}\left(\boldsymbol{x};\kappa\boldsymbol{\theta}_l\right) = \mathbf{BN}\left(\boldsymbol{x};\boldsymbol{\theta}_l\right). \tag{3}$$

This implies that scaling the weights by κ has a negligible effect on the output. Therefore, we can consider the effect of the weight decay term, $\frac{1}{2}\lambda_1 \left\|\boldsymbol{\theta}\right\|_2^2$, independent of the cross-entropy loss. Because we are using the stochastic gradient descent (SGD) optimizer with an initial parameter θ_0, and learning rate η, the update by the weight decay proceeds as follows:

$$\theta_1 = \theta_0 - \eta\lambda_1\theta_0 = (1-\eta\lambda_1)\theta_0 \tag{4}$$

$$\theta_2 = (1-\eta\lambda_1)\theta_1 = (1-\eta\lambda_1)^2\theta_0 \tag{5}$$

$$\vdots$$

$$\theta_{kN_1} = (1-\eta\lambda_1)^{kN_1}\theta_0. \tag{6}$$

If we train another network with N_2 labeled samples, the weight θ_{kN_2} becomes $\theta_{kN_2} = (1-\eta\lambda_2)^{kN_2}\theta_0$. For an identical effect of the weight decay regularization on the networks, we have the following:

$$(1-\eta\lambda_2)^{kN_2}\theta_0 = (1-\eta\lambda_1)^{kN_1}\theta_0 \tag{7}$$

$$\lambda_2 = \frac{1}{\eta}\left\{1-(1-\eta\lambda_1)^{\frac{N_1}{N_2}}\right\} \tag{8}$$

$$\approx \frac{1}{\eta}\left\{1-\left(1-\frac{N_1}{N_2}\eta\lambda_1\right)\right\} \tag{9}$$

$$= \frac{N_1}{N_2}\lambda_1. \tag{10}$$

In Equation (9), we only approximate the first two terms from the binomial expansion, because η and λ are much smaller than 1. This result shows that when training a CNN with batch normalization, reducing the weight decay inversely proportional to the number of data can approximately have the same effect on the weight. Therefore, we set λ_0 during the initial round and reduce it to a value inversely proportional to the number of data from the second round onward. In addition, because the initial parameters are assumed to be the same when deriving the equation, the model should be randomly initialized during every round. Note that the weight values do not have to be exactly the same, but they only need to be initialized to a distribution of equal size. If we initialize the model using the model trained in the previous round, the effect of the weight decay will be more significant than desired because the weight is already reduced by the weight decay during the previous round.

3.3 Knowledge Distillation

If we re-initialize the model during every round and train from scratch, the information learned by the previous model f_{t-1} is only used to select a new sample and form $\mathcal{D}_t^L$; it is not applied to train the new model f_t. Some previous studies have initialized f_t with f_{t-1} to employ previously learned information[40,31,46,49,21]. However, we found that, without re-initialization, the effect of the weight decay will be so large that the performance will decrease, even if the same weight decay scheduling is used. This is consistent with the findings of Hu *et al.* [13]. We discuss this in more detail in Section 5.1.

As an alternative to initializing f_t to f_{t-1}, we propose a transfer of knowledge of f_{t-1} to f_t using a knowledge distillation technique. We follow the knowledge distillation proposed by Hinton *et al.* [11]. In the first round, we train the model without a teacher network. From the second round onward, we train the student model f_t by distilling knowledge from the teacher model f_{t-1}. We experimented using two cases: distilling all $\mathcal{D}_t^L$ and distilling only $\mathcal{D}_{t-1}^L$ that the teacher model previously used for training. For the former, the total loss $\mathcal{L}_t$ used to train the

student model, f_t, is as follows:

$$\mathcal{L}_t^{KD1} = \frac{1-\alpha}{\left|\mathcal{D}_t^L\right|} \sum_{\boldsymbol{x}\in\mathcal{D}_t^L} \mathcal{L}_{CE}\left(y, \sigma\left(\boldsymbol{z}_t(\boldsymbol{x})\right)\right) + \frac{\alpha T^2}{\left|\mathcal{D}_t^L\right|} \sum_{\boldsymbol{x}\in\mathcal{D}_t^L} \mathcal{L}_{KL}\left(\sigma\left(\frac{\boldsymbol{z}_{t-1}(\boldsymbol{x})}{T}\right), \sigma\left(\frac{\boldsymbol{z}_t(\boldsymbol{x})}{T}\right)\right) + \frac{1}{2}\lambda_t \left\|\boldsymbol{\theta}\right\|_2^2 \tag{11}$$

where $\boldsymbol{z}_t$ is the logits output by the network f_t, T is the temperature, σ is the softmax function, $\mathcal{L}_{KL}$ is the Kullback Leibler (KL) divergence, and $|\cdot|$ represents the cardinality, the latter of which is as follows:

$$\mathcal{L}_t^{KD2} = \frac{1-\alpha}{\left|\mathcal{D}_t^L\right|} \sum_{\boldsymbol{x}\in\mathcal{D}_t^L} \mathcal{L}_{CE}\left(y, \sigma\left(\boldsymbol{z}_t(\boldsymbol{x})\right)\right) + \frac{\alpha T^2}{\left|\mathcal{D}_{t-1}^L\right|} \sum_{\boldsymbol{x}\in\mathcal{D}_{t-1}^L} \mathcal{L}_{KL}\left(\sigma\left(\frac{\boldsymbol{z}_{t-1}(\boldsymbol{x})}{T}\right), \sigma\left(\frac{\boldsymbol{z}_t(\boldsymbol{x})}{T}\right)\right) + \frac{1}{2}\lambda_t \left\|\boldsymbol{\theta}\right\|_2^2 . \tag{12}$$

The overall framework for knowledge distillation is the same as that of [11]. However, Hinton *et al.* [11] focused on compressing and speeding up the model by transferring knowledge of a larger model to a smaller model. By contrast, our goal was to identify knowledge distillation from the teacher model, which shows a reduced performance and has a low effective complexity. Note that reducing the KL divergence with a lower performing teacher can interfere with the training and cause a performance degradation as compared to when training alone without teacher.

4 Experiments

In this section, we first verify the method of adjusting a weight decay inversely proportional to the number of training data (Section 4.1). We also evaluated the proposed active learning method on the MNIST [25], CIFAR-10, and CIFAR-100 [23] datasets using CNNs of various architectures (Section 4.2 $\sim$ 4.4).

Comparison targets. We initialize a labeled dataset $\mathcal{D}_0^L$ through random sampling. For each method, we repeat the same experiment five times with different initially labeled images. In addition, for a fair comparison, all methods use the same random seed for each of the five experiments. We consider three acquisition methods for comparison:

- Random indicates a random sampling regardless of the active learning method.
- Entropy is the most frequently compared method in active learning [2,6,37,49]. We use entropy for all of our methods because the acquisition function is not our focus.
- In LL [49], samples that are expected to have the highest loss are selected using the loss prediction module. To the best of our knowledge, this method achieves a state-of-the-art performance on the CIFAR-10 dataset.

Table 1: Accuracy(%) of (a) 5-layer CNN with various fixed weight decays on the MNIST dataset where the number of training data varies from 200 to 10,000 and (b) Resnet18 with various fixed weight decays on the CIFAR-10 dataset where the number of training data varies from 1,000 to 50,000.

(a) MNIST

WD	Number of labeled images					
	200	500	1k	2k	5k	10k
0.2	92.37	94.86	95.64	95.81	93.84	94.50
0.1	**92.46**	95.75	96.75	96.94	96.67	96.82
0.04	91.82	**96.30**	97.12	97.68	98.17	98.26
0.02	91.00	96.03	**97.29**	97.96	98.54	98.71
0.01	89.97	95.06	97.25	**98.03**	98.58	98.94
0.004	89.94	94.50	96.70	97.91	**98.70**	98.97
0.002	89.25	94.25	96.42	97.63	98.60	**98.98**
0.001	89.67	94.18	96.17	97.45	98.54	98.90

(b) CIFAR-10

WD	Number of labeled images					
	1k	2k	5k	10k	20k	50k
0.02	52.44	54.25	33.03	19.58	17.66	10.00
0.01	**62.84**	71.80	75.83	74.60	71.91	61.75
0.005	58.46	**73.93**	81.50	84.06	83.47	83.37
0.002	54.12	70.83	**83.15**	87.54	90.42	92.05
0.001	53.30	67.63	83.05	**88.26**	92.00	94.48
0.0005	52.1	65.34	81.37	88.13	**92.23**	95.18
0.0002	49.47	63.56	79.23	86.56	91.39	**95.26**
0.0001	50.18	64.06	78.50	85.76	90.91	94.84

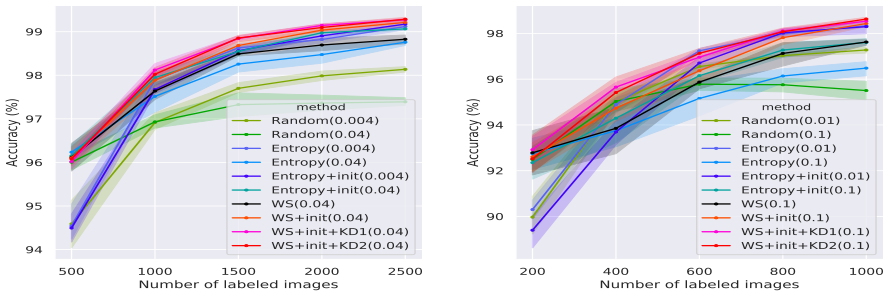
We also consider distilling all $\mathcal{D}_t^L$ and distilling only $\mathcal{D}_{t-1}^L$ learned by the teacher, which we denote as KD1 and KD2, respectively.

4.1 Results for Weight Decay VS the Number of Data

Table 1 shows the test accuracy (%) of the networks according to the fixed weight decay at each number of labeled training data. We use a 5-layer CNN for MNIST, and Resnet18 for the CIFAR-10 dataset. The implementation details are described in Sections 4.2 and 4.3. For each number of data, bold values represent the highest accuracy; the corresponding weight decay decreases inversely proportional to the number of data. The more data we have, the weaker the regularizer we need to ensure that the model can learn a sufficient amount of information. If the strong weight decay of the first round is maintained in the subsequent rounds, the performance deteriorates. The optimal weight decay changes depending on the number of data. Thus, to reach the target performance with the minimum number of data, it is necessary to continuously adjust the optimal weight decay. For this reason, we propose a scheduling weight decay; a fixed weight decay is unsuitable for active learning.

4.2 Results for Active Learning on MNIST

Implementation details. For the MNIST dataset, we use a 5-layer CNN, which is composed of four convolutional layers followed by stride-2 max pooling and the last fully connected layer. Each convolutional layer has 32, 64, 64, and 64 channels. Following [9], we used three consecutive operations for the convolutional layers: a 3×3 convolution with a stride of 1 without padding, batch normalization [16], and a rectified linear unit (ReLU) [7]. For all methods, we train models for 100 epochs with the initial learning rate 0.01 decayed by a factor of 0.1 at



(a) $K = 500$ (b) $K = 200$

Fig. 1: Performance comparison on MNIST

50 epochs and use no data augmentation. The batch size is set to 32. For distillation, we use an α of 0.5 and T of 6. We test five rounds and sample K images every round. The CNN is trained using $5K$ data during the final round. We use two budget settings, i.e., $K = 500$ and $K = 200$, for the experiments.

Results. Figure 1 shows the results. For a fixed weight decay, the value in parentheses for each method represents the weight decay value. For the weight decay scheduling method, which we note as WS, the number in parentheses represents the initial value. The method of random re-initialization in each round is denoted by init. Shaded areas represent 95% confidence intervals by performing 1000 bootstrap resampling. During the initial round, the methods with WS show outstanding performance because of their well-tuned weight decay. As the weight decay decreases, the performance gap is reduced. When the number of labeled images is 2,500, Entorpy(0.004) and Entorpy+init(0.004) show 99.12% and 99.17% respectively. WS+init+KD1 and WS+init+KD2 show 99.27% and 99.29% respectively, and they outperform all the other methods regardless of labeling budget. The two distillation methods show similar performance and there is no significant difference in trend.

4.3 Results for Active Learning on CIFAR-10

Implementation details. For the CIFAR-10 dataset, we use Resnet18 [9] and Densenet100 [14]. For Resnet18, we train the models for 350 epochs with an initial learning rate 0.1 decayed by factor of 0.1 at epochs 150 and 250. The batch size is set to 128. For Densenet100, we train for 300 epochs with an initial learning rate 0.1 decayed by factor 0.1 at epochs 150 and 225. The batch size is set to 64. We use the standard augmentation setting such as resizing, cropping, and flipping for both architectures. For distillation, we use an α of 0.5 and T of 6. We test ten rounds and sample K images every round. The CNN is trained with $10K$ data during the final round. We experiment on two budget settings, i.e., $K = 1000$ and $K = 250$.

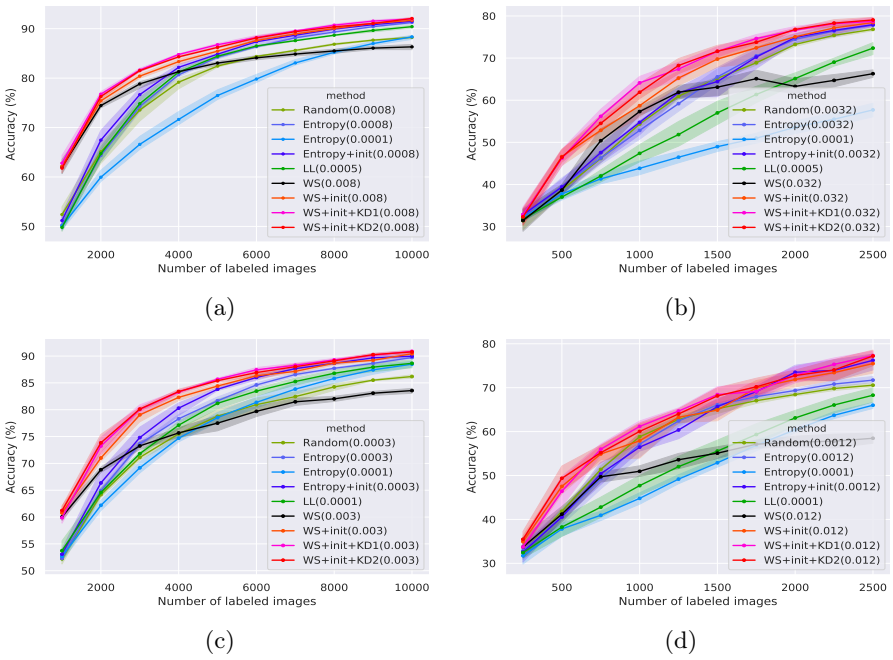
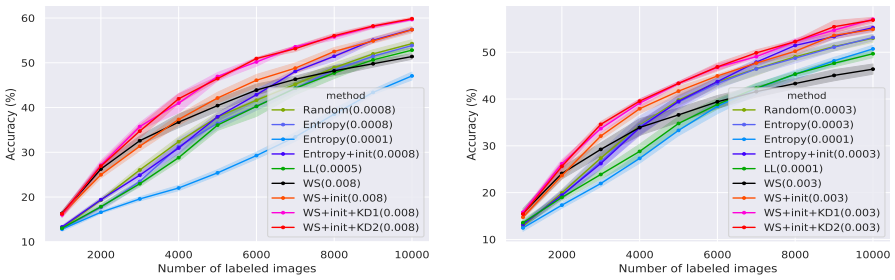


Fig. 2: Performance comparison on CIFAR-10: (a) Resnet18, $K = 1000$, (b) Resnet18, $K = 250$, (c) Densenet100, $K = 1000$, and (d) Densenet100, $K = 250$

Results. Except for Entropy(0.0001) and LL, the methods are set to have the same weight decay during the last round. If we set the weight decay of Entropy to the same initial value of the WS method, the CNN does not converge as the round proceeds. This occurs because an excessive weight decay interferes with the model training. Here, 0.0001 is the value used in the original paper on both Resnet [9] and Densenet [14]. For the LL method [49], we also follow the weight decay value from the original paper.

Figure 2 shows the results. In Figure 2 (a), WS+init (0.008) and Entropy+init (0.0008) are 91.68% and 91.51%, respectively, during the last round. In addition, in Figure 2 (c), WS+init (0.003) and Entropy+init (0.0003) are 89.96% and 90.43% respectively during the last round. Note that the WS+init and Entropy+init methods have the same weight decay value during the last round. Nevertheless, the performance of WS+init is better than that of Entropy+init for all four cases. This shows that, although the same acquisition function is used, a well-tuned CNN can select more informative samples. Therefore, appropriately scheduling the weight decay in each round seems important for the sampling, as well as for the performance of the current model. In addition, by simply adjusting the weight decay and applying a random initialization on each round, Entropy+init(0.0008) shows a better performance than LL(0.0005).



(a) (b)

Fig. 3: Performance comparison on CIFAR-100 for $K = 1000$: (a) Resnet18, and (b) Densenet100

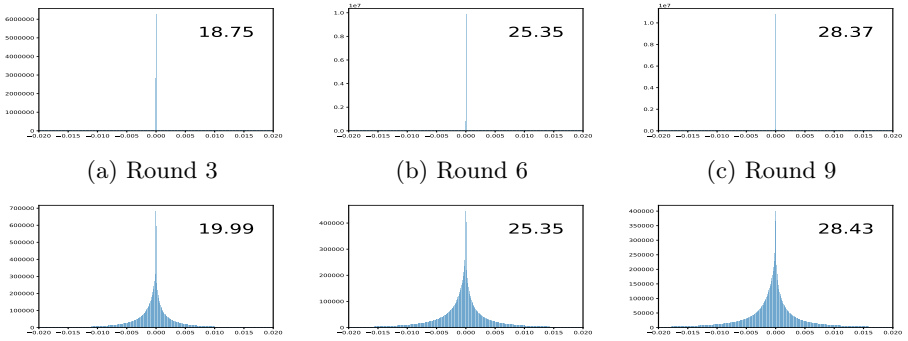
In Figure 2 (d), Random achieves a better performance than Entropy during the early stage. During the third round, Random (0.0012) and Entropy (0.0012) achieve rates of 51.37% and 49.31% respectively. However, during the last round, Entropy (0.0012) shows a better performance. It appears that, if the number of training data is too small, applying only difficult samples hinders the training.

In most cases, initializing with a pre-trained model from the previous round degrades the performance. In particular, WS suffers from a significant degradation in the performance. We discuss this phenomenon in Section 5.1.

4.4 Results for Active learning on CIFAR-100

Implementation details. For the CIFAR-100 dataset, we also use Resnet18 [9] and Densenet100 [14]. All settings are the same as with CIFAR-10, except for K. If we set K to 250, there might be some classes that are not selected during the initial round, and thus we only experiment using $K = 1000$.

Results. Figure 3 shows the result. There is an inconsistency in the behaviors of the methods when switching from CIFAR-10 to CIFAR-100. During the last round, WS+init (0.008) and Entropy+init (0.0008) achieve results of 57.38% and 57.43%, respectively (Figure 3 (a)). In addition, WS+init (0.003) and Entropy+init (0.0003) achieve rates of 54.88% and 55.23%, respectively (Figure 3 (b)). The accuracy of WS+init is worse than that of Entropy+init during the last round. Even Random shows a better result than Entropy. For the CIFAR-100 dataset, the accuracy of the CNN is low and the number of training data per class is smaller than that of CIFAR-10; and thus it seems that applying only difficult samples hinders the training. However, the best performance is shown when WD scheduling and knowledge distillation methods are used together (WD+init+KD).



(a) Round 3 (b) Round 6 (c) Round 9

(d) Round 3 (e) Round 6 (f) Round 9

Fig. 4: Histogram of convolutional weights in Resnet18 during the active learning process. The number in each sub-figure represents the L2-norm of the weights: (a)∼(c) WS(0.008), and (d)∼(f) WS+init(0.008)

5 Analysis

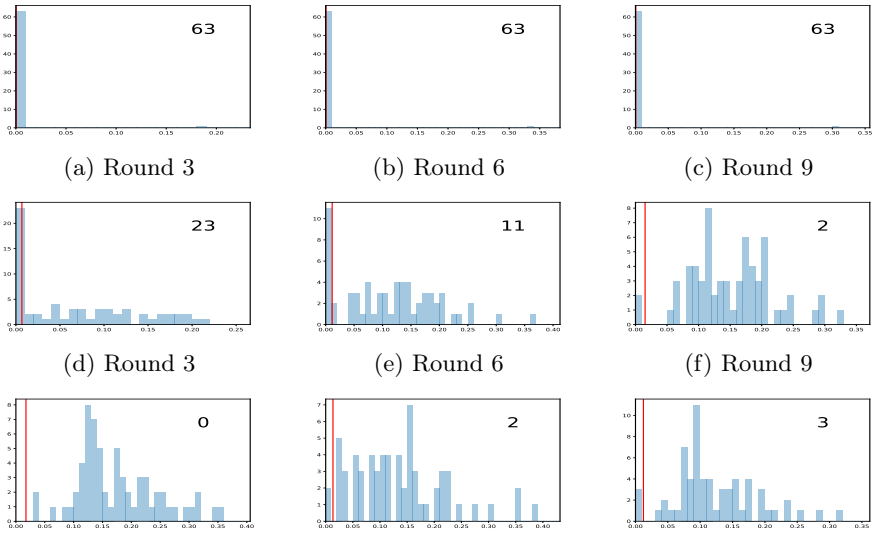
In this section, we analyze why the model initialized to the model in the previous round shows a worse performance than the model trained from scratch. (Section 5.1). We also analyze how the distillation parameter α affects the distillation performance (Section 5.2).

5.1 Effect of Weight Decay

Figure 4 shows a histogram of the weights in convolutional layers of ResNet18. The number in each sub-figure represents the L2-norm of the weights. The experimental setting is the same as that in Section 4.3 where the labeling budget K is 1000. The upper column represents the result when the model is initialized to the model trained in the previous round (WD(0.008)), and the lower column represents the results when the model is randomly initialized during every round (WD+init(0.008)). The norm of WS(0.008) is similar to the norm of WS+init(0.008) even though most of the weights are distributed close to 0. In other words, the norm of the weight is very unevenly distributed, and a small number of weights with large norms have much more influence in determining the overall prediction. This implies that the actual network capacity is smaller than the actual number of parameters. This may cause the performance degradation.

With a similar analysis, we found that another reason for the performance degradation is the weight decay causing the weight of the batch norm to be sparse, resulting in a reduced number of effective convolutional channels.[1] In

[1] This analysis is applicable when weight decay is applied to the batch norm weight. Pytorch implementation of several CNN models also gives weight decay to the weight of the batch norm.



(a) Round 3 (b) Round 6 (c) Round 9

(d) Round 3 (e) Round 6 (f) Round 9

(g) Round 3 (h) Round 6 (i) Round 9

Fig. 5: Histogram of batch norm weights in ResNet18 during the active learning process. We show only the first batch norm weights of the first residual block. The red vertical line represents the average divided by 10 and the number in each sub-figure represent the number of weights that are less than the average divided by 10: (a)∼(c) WS(0.008), (d)∼(f) WS+init(0.008), and (g)∼(i) Entropy+init(0.0008)

other words, although the weight of the convolution is re-scaled at the batch norm layer, the weight decay can still limit the complexity by reducing the number of actual filters. Figure 5 shows a histogram of batch the norm weights for the following three cases: WS(0.008), WS+init(0.008), and Entropy+init(0.0008). The figure shows the absolute value of the weights and we show only the first batch norm weights of the first residual block. The red vertical line in each sub-figure represents the average divided by 10 and the number in each sub-figure are the number of weights that are smaller than the mean of the weights divided by 10. In the case of WS(0.008), the impact of the weight decay is great, thereby causing most of the weight of the batch norm to be close to zero (Figure 5 (a)∼(c)), as well as causing a reduction in the weight of the convolutional filter. Weight values that are much smaller than the average value refer to the corresponding filter, which has little impact on the prediction, thereby reducing the actual complexity of the model. Therefore, a model without random re-initialization is significantly affected by the weight decay, and the poor performance appears to be due to the low complexity. By contrast, in the case of WS+init(0.008), the number of batch norm weights at near zero decreases (Figure 5 (d)∼(f)) as the

Table 2: Performance comparison according to knowledge distillation parameter α on CIFAR-10 using Resnet18

Method	α	Number of labeled images									
		1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
WS+init	·	62.17	75.40	80.40	83.33	85.45	87.60	88.93	90.03	91.07	91.68
WS+init+KD1	0.1	**62.57**	75.49	80.74	83.72	85.71	87.70	89.46	90.26	91.28	91.90
WS+init+KD1	0.5	61.27	**76.24**	**81.85**	**84.77**	86.91	88.07	89.52	90.57	91.53	91.96
WS+init+KD1	0.9	62.55	74.18	80.88	84.59	**86.99**	**88.60**	**89.81**	**90.86**	**91.66**	**92.18**

round proceeds and the number of training data increases. Through weight decay scheduling, the model is trained to have a small number of effective channels when the number of training data is small, and to have a large number of effective channels when the number of training data is large. If random initialization is applied during every round but weight decay scheduling is not applied, we can see that the number of effective channels does not increase in proportion to the number of data (Figure 5 (g)∼(i)). In other words, regardless of the round, the effective capacity is similar, which means that an over-fitting can occur during the early rounds.

5.2 Effect of Knowledge Distillation Parameter

In this section, we analyze how the knowledge distillation parameter affects the performance. In our experiments, the temperature T does not show a significant difference in performance. We compare the performance when α changes. Table 2 shows the result. A small α gives more weight to the cross-entropy loss and a large α gives more weight to the distillation loss. In the early rounds during which the difference in performance between the teacher and student is large, it is better to provide similar weights to the cross-entropy and distillation loss. However, because the training proceeds and the performances of the teacher and student are similar, a large α value shows a better performance. For a small α, there is no significant performance improvement even after knowledge distillation.

6 Conclusion

We showed that reducing the weight decay inversely proportional to the number of data is simple but effective, and we that even a low-performance teacher can distill knowledge in an active learning setting. The experimental results show that our method outperforms the baseline methods. In addition, we presented a new perspective of how the weight decay regularizes a convolutional neural network, which contains a batch normalization layer. The weight decay provides a regularization effect by limiting the number of parameters and effective channels of the convolutional layers.

References

1. Ba, J., Caruana, R.: Do deep nets really need to be deep? In: Advances in neural information processing systems. pp. 2654–2662 (2014)
2. Beluch, W.H., Genewein, T., Nürnberger, A., Köhler, J.M.: The power of ensembles for active learning in image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9368–9377 (2018)
3. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space. arXiv preprint arXiv:1909.13719 (2019)
4. Elhamifar, E., Sapiro, G., Yang, A., Shankar Sasrty, S.: A convex optimization framework for active learning. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 209–216 (2013)
5. Freytag, A., Rodner, E., Denzler, J.: Selecting influential examples: Active learning with expected model output changes. In: European Conference on Computer Vision. pp. 562–577. Springer (2014)
6. Gal, Y., Islam, R., Ghahramani, Z.: Deep bayesian active learning with image data. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 1183–1192. JMLR. org (2017)
7. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. pp. 315–323 (2011)
8. Guo, Y.: Active instance sampling via matrix partition. In: Advances in Neural Information Processing Systems. pp. 802–810 (2010)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
10. Heo, B., Kim, J., Yun, S., Park, H., Kwak, N., Choi, J.Y.: A comprehensive overhaul of feature distillation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1921–1930 (2019)
11. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
12. Hoffer, E., Banner, R., Golan, I., Soudry, D.: Norm matters: efficient and accurate normalization schemes in deep networks. In: Advances in Neural Information Processing Systems. pp. 2160–2170 (2018)
13. Hu, P., Lipton, Z.C., Anandkumar, A., Ramanan, D.: Active learning with partial feedback. arXiv preprint arXiv:1802.07427 (2018)
14. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
15. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: European conference on computer vision. pp. 646–661. Springer (2016)
16. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
17. Jastrzębski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., Storkey, A.: Three factors influencing minima in sgd. arXiv preprint arXiv:1711.04623 (2017)
18. Joshi, A.J., Porikli, F., Papanikolopoulos, N.: Multi-class active learning for image classification. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 2372–2379. IEEE (2009)

19. Käding, C., Rodner, E., Freytag, A., Denzler, J.: Active and continuous exploration with deep neural networks and expected model output changes. arXiv preprint arXiv:1612.06129 (2016)
20. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: Generalization gap and sharp minima. arXiv preprint arXiv:1609.04836 (2016)
21. Khodabandeh, M., Deng, Z., Ibrahim, M.S., Satoh, S., Mori, G.: Active learning for structured prediction from partially labeled data. arXiv preprint arXiv:1706.02342 (2017)
22. Kim, J., Park, S., Kwak, N.: Paraphrasing complex network: Network compression via factor transfer. In: Advances in Neural Information Processing Systems. pp. 2760–2769 (2018)
23. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
24. Krogh, A., Hertz, J.A.: A simple weight decay can improve generalization. In: Advances in neural information processing systems. pp. 950–957 (1992)
25. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
26. Lewis, D.D., Catlett, J.: Heterogeneous uncertainty sampling for supervised learning. In: Machine learning proceedings 1994, pp. 148–156. Elsevier (1994)
27. Lewis, D.D., Gale, W.A.: A sequential algorithm for training text classifiers. In: SIGIR'94. pp. 3–12. Springer (1994)
28. Li, H., Xu, Z., Taylor, G., Studer, C., Goldstein, T.: Visualizing the loss landscape of neural nets. In: Advances in Neural Information Processing Systems. pp. 6389–6399 (2018)
29. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3431–3440 (2015)
30. Luo, W., Schwing, A., Urtasun, R.: Latent structured active learning. In: Advances in Neural Information Processing Systems. pp. 728–736 (2013)
31. Meyer, B.J., Drummond, T.: The importance of metric learning for robotic vision: Open set recognition and active learning. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 2924–2931. IEEE (2019)
32. Neelakantan, A., Vilnis, L., Le, Q.V., Sutskever, I., Kaiser, L., Kurach, K., Martens, J.: Adding gradient noise improves learning for very deep networks. arXiv preprint arXiv:1511.06807 (2015)
33. Nguyen, H.T., Smeulders, A.: Active learning using pre-clustering. In: Proceedings of the twenty-first international conference on Machine learning. p. 79 (2004)
34. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
35. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: Fitnets: Hints for thin deep nets. arXiv preprint arXiv:1412.6550 (2014)
36. Roth, D., Small, K.: Margin-based active learning for structured output spaces. In: European Conference on Machine Learning. pp. 413–424. Springer (2006)
37. Sener, O., Savarese, S.: Active learning for convolutional neural networks: A core-set approach. arXiv preprint arXiv:1708.00489 (2017)
38. Settles, B., Craven, M.: An analysis of active learning strategies for sequence labeling tasks. In: Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing. pp. 1070–1079 (2008)

39. Settles, B., Craven, M., Ray, S.: Multiple-instance active learning. In: Advances in neural information processing systems. pp. 1289–1296 (2008)
40. Shen, Y., Yun, H., Lipton, Z.C., Kronrod, Y., Anandkumar, A.: Deep active learning for named entity recognition. arXiv preprint arXiv:1707.05928 (2017)
41. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
42. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research **15**(1), 1929–1958 (2014)
43. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. Journal of machine learning research **2**(Nov), 45–66 (2001)
44. Van Laarhoven, T.: L2 regularization versus batch and weight normalization. arXiv preprint arXiv:1706.05350 (2017)
45. Vijayanarasimhan, S., Grauman, K.: Large-scale live active learning: Training object detectors with crawled data and crowds. International journal of computer vision **108**(1-2), 97–114 (2014)
46. Wang, K., Zhang, D., Li, Y., Zhang, R., Lin, L.: Cost-effective active learning for deep image classification. IEEE Transactions on Circuits and Systems for Video Technology **27**(12), 2591–2600 (2016)
47. Xie, Q., Hovy, E., Luong, M.T., Le, Q.V.: Self-training with noisy student improves imagenet classification. arXiv preprint arXiv:1911.04252 (2019)
48. Yim, J., Joo, D., Bae, J., Kim, J.: A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4133–4141 (2017)
49. Yoo, D., Kweon, I.S.: Learning loss for active learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 93–102 (2019)
50. Zagoruyko, S., Komodakis, N.: Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. arXiv preprint arXiv:1612.03928 (2016)
51. Zhang, G., Wang, C., Xu, B., Grosse, R.: Three mechanisms of weight decay regularization. arXiv preprint arXiv:1810.12281 (2018)