Compensating for the Lack of Extra Training Data by Learning Extra Representation

Hyeonseong Jeon[1][0000−0002−5599−6883], Siho Han[2][0000−0001−9303−3714], Sangwon Lee[2][0000−0003−3936−9342], and Simon S. Woo[2][0000−0002−8983−1542]

[1] Department of Artificial Intelligence
[2] Department of Applied Data Science
Sungkyunkwan University, Suwon, S. Korea
{cutz, siho.han, lee1465, swoo}@g.skku.edu

Abstract. Outperforming the previous state of the art, numerous deep learning models have been proposed for image classification using the ImageNet database. In most cases, significant improvement has been made through novel data augmentation techniques and learning or hyper-parameter tuning strategies, leading to the advent of approaches such as FixNet, NoisyStudent, and Big Transfer. However, the latter examples, while achieving the state-of-the-art performance on ImageNet, required a significant amount of *extra training data*, namely the JFT-300M dataset. Containing 300 million images, this dataset is 250 times larger in size than ImageNet, but is publicly unavailable, while the model pre-trained on it is. In this paper, we introduce a novel framework, Extra Representation (ExRep), to surmount the problem of not having access to the JFT-300M data by instead using ImageNet and the publicly available model that has been pre-trained on JFT-300M. We take a *knowledge distillation* approach, treating the model pre-trained on JFT-300M as well as on ImageNet as the teacher network and that pre-trained only on ImageNet as the student network. Our proposed method is capable of learning additional representation effects of the teacher model, bolstering the student model's performance to a similar level to that of the teacher model, achieving high classification performance even without extra training data.

1 Introduction

The success of deep learning has been prominent in the image classification domain [1,2,3,4], partly, yet most importantly, since the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 with the development of the ImageNet database. Even after ILSVRC 2012, ImageNet has been the de facto standard benchmark dataset for image classification. A sizeable body of research [5,6] involved the ImageNet data for the evaluation of their methods in terms of classification accuracy. However, the proposed methods typically relied on convolutional neural networks (CNNs), facing inevitable limitations due to their lack of generalization ability to the ImageNet validation set.

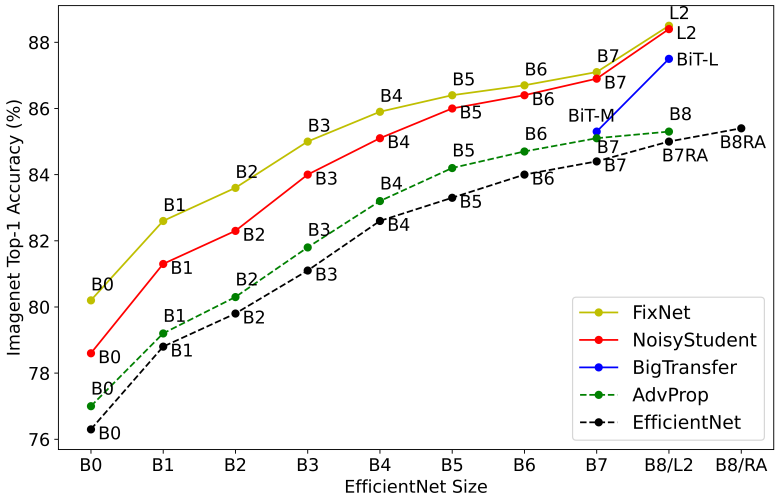

Fig. 1. Top-1 accuracy on ImageNet. Dotted lines represent the performance without extra training data, whereas solid lines represent that with extra training data. RA denotes RandAugment.

Only a few approaches managed to overcome these limitations by self-training with a noisy student (NoisyStudent) [7], fixing the train-test resolution (FixNet) [8], or scaling up pre-training (Big Transfer or BiT) [9]. From Fig. 1, we can observe that the latter (solid red, yellow, and blue lines in Fig. 1) approaches achieve higher top-1 accuracy than the state-of-the-art architecture, EfficientNet (black dotted line in Fig 1), using extra training data. NoisyStudent introduced a novel training strategy, that is, self-training with a noisy student; FixNet proposed an effective training method, using the train-test image resolution discrepancy; BiT took a transfer learning approach similar to that used for language modeling, whereby a model pre-trained on a large task is fine-tuned for use in a smaller task. The achievement of high top-1 accuracy of these three methods is attributed to multiple factors, such as a massive number of parameters (NoisyStudent and FixNet each has about 480 million parameters), high resolution images by 800×800 interpolations, AutoAugment [10] by policy gradient agents, and optimized architectures by neural architecture search [11]. Most importantly, however, all three methods exploit vast amounts of extra training data, which, as shown in many cases, such as for CIFAR10 and CIFAR100 [12], and SVHN [13], generally leads to higher classification performance.

Developed by Google for internal use, the JFT-300M dataset [14] consists of 300 million images labeled with 18,291 categories. While offering an impressive

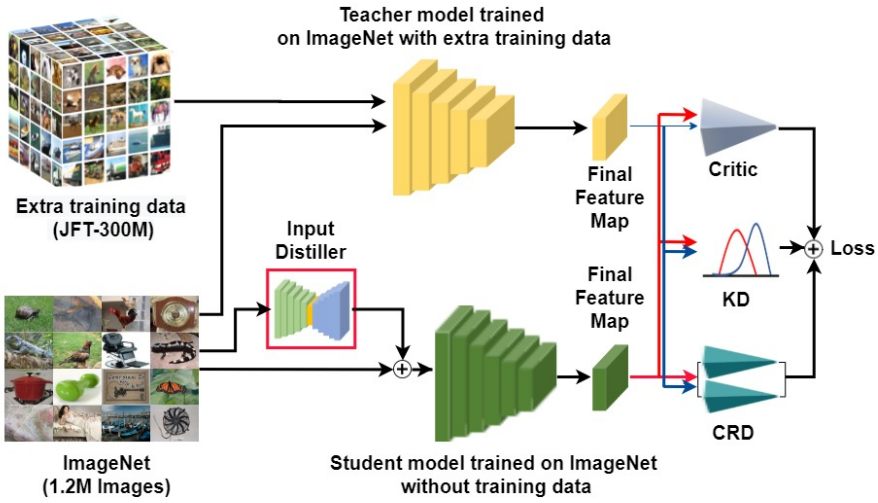

Fig. 2. Overview of our ExRep framework. The input distiller (red box) learns the discrepancy between ImageNet and JFT-300M and the output is fed to the student model. From the final feature maps of the teacher and student models are computed three loss functions — the critic loss, the knowledge distillation (KD) loss, and the contrastive representation distillation (CRD) loss.

amount of 1.2 million images, ImageNet pales in comparison to the JFT-300M dataset, solely considering the number of samples. Nonetheless, the JFT-300M dataset is not publicly available, making it nearly impossible to reproduce the results yielded by approaches using this data. In this work, we aim at achieving a classification performance that is on par with those of the aforementioned models, which used the JFT-300M dataset as extra training data. In particular, based on the hypothesis that the model pre-trained on both ImageNet and JFT-300M (the "teacher model") contains significantly more representational information than that pre-trained only on ImageNet (the "student model"), as demonstrated by the high classification performance of NoisyStudent [7], FixNet [8], and BiT [9], we leverage the discrepancy (the "extra representation") between the teacher and student models; if the student model learns to replicate the representational behavior of the teacher model, then the student model would be able to achieve equally high classification performance on ImageNet as the teacher model.

As shown in Fig. 2, we apply knowledge distillation [14] to achieve our learning objective, which is to transfer the knowledge acquired by the teacher model during pre-training to the student model. We introduce an autoencoder-based *input distiller* (red box in Fig. 2) to replicate the effect of training on JFT-300M by feeding the reconstructed output of the input distiller to the student model, which differs from many prior works that directly train the weights of the student model. We also propose a novel adversarial training [15] technique to effectively

train our input distiller, which acts as the generator in Generative Adversarial Networks (GANs). We place the discriminator at the final feature layer of the pre-trained models to distinguish whether the output is from the teacher model or the student model. Finally, we compute the loss function by aggregating the critic loss, the knowledge distillation (KD) loss [14], and the contrastive representation distillation (CRD) [16] loss obtained by comparing the final feature maps of the teacher and student models, as shown in Fig. 2. Consequently, we can effectively represent the additional representation that would have been available had we had direct access to the JFT-300M dataset itself. This training procedure thus renders the model representation more informative, enabling us to achieve high classification performance as if our training set also consisted of JFT-300M; we refer to the integrated, end-to-end framework comprised of these components as ExRep. We also demonstrate that ExRep can be used to train a randomly initialized model without any extra training data and that ExRep is reproducible even if models are trained using randomly initialized weights.

2 Related Work

Image classification. Since the advent of Xception [5], CNN architectures have been deeper and broader: combining ResNet [4] and Xception, ResNeXt [17] achieved high classification performance on ImageNet. Also, the use of optimal CNN blocks, enabled by neural architecture search [11], led to popular models, such as EfficientNet (B0 through B7 or L2, depending on the compound scaling) [6]. Since most approaches exploiting extra training data use EfficientNet as their backbone network, we also use EfficientNet to prepare our teacher and student models for knowledge distillation. However, previous methods also incorporate other methods, such as novel learning strategies or data augmentation techniques, to further improve the ImageNet classification performance beyond the one that can be achieved by the sole use of EfficientNet as the backbone.

Image classification with extra training data. Many prior works utilize large datasets for model representations by weakly supervised learning [18,19,9]. Joulin et al. [19] used the Flickr-100M dataset [20], although the performance was lower than that of approaches using JFT-300M [14] or Instagram-1B [21], which is an internal dataset for Facebook generated from Instagram images. NoisyStudent [7], FixNet [8], and BiT [9] used the JFT-300M dataset, based on weakly supervised learning. NoisyStudent applied self-training, where the teacher model and a bigger and noisy student model guide each other by exchanging feedback. FixNet employed different train-test resolutions, demonstrating that a low train resolution with a fine-tuning of the model at the test resolution improves the test classification accuracy. BiT combined well-organized components and simple heuristic methods from prior works on image classification, using upstream pre-training and downstream fine-tuning (a huge pre-trained model and a tiny fine-tuned model), Group Normalization [22], and Weight Standardization [23]. BiT has BiT-S, BiT-M, and BiT-L as the model size grows. As of now, these methods constitute state of the art for image classification on ImageNet (see

Fig. 1), but are not reproducible using a randomly initialized training model due to the limited access to the JFT-300M dataset; that is, only the model pre-trained on JFT-300M is publicly available instead of the dataset itself.

Image classification without extra training data. Leveraging policy gradient agents for parameter search for image augmentation, AutoAugment (AA) [10] achieved high classification accuracy on ImageNet without extra training data. RandAugment (RA) [24], of which the performance is shown in Fig. 1, reduced the search space of AA, achieving the highest accuracy with the EfficientNet-B8 backbone; however, RA is computationally expensive. Unlike other methods using adversarial examples [25,26], which only focus on model robustness against adversarial attacks, AdvProp [27] aims at improving model generalization as well. They apply Auxiliary Batch Normalization (ABN) [28] for the training of a novel classifier. Based on these prior work, we attempt to further improve the classification performance without using any extra training data.

Knowledge distillation. Applied for the computation of our loss function as well, knowledge distillation [14] (KD) has been proposed to alleviate the computational cost resulting from training a large model or an ensemble of models by transferring the acquired knowledge to a smaller model. The latter student model is trained under the guidance of the large pre-trained teacher model, such that the student model replicates the soft targets of the teacher model. Yim et al. [29] proposed a weight-based knowledge distillation method, where the flow of solution procedure (FSP) matrix, defined by the Gramian matrix of adjacent layers in the same model, is learned during the training phase. This approach is similar to ours in that an additional layer is added, but ours, which is the input distiller, is added to the input images and not to the intermediate layers [29].

Zagoruyko and Komodakis [30] proposed to transfer the neuron responses during the process by which the student model learns to classify images; this work additionally suggests an attention transfer method whereby the activation-based and gradient-based spatial attention maps are matched. The attention-based method has been further generalized by Huang and Wang [31], who proposed to minimize the Maximum Mean Discrepancy (MMD) metric of the distributions of neuron selectivity patterns between the teacher and student models. Heo et al. [32] proposed an activation transfer loss that is minimized when the activation boundaries formed by hidden neurons in the student model coincide with those in the teacher model.

However, most prior works, attempting to minimize the Kullback–Leibler (KL) divergence between the probabilistic outputs of the teacher and student models, tend to overlook important structural knowledge of the teacher model. Motivated by InfoNCE loss [33], Tian et al. [16] thus proposed CRD, which formulates a contrastive learning objective by which the student model is trained to capture additional information in the teacher model's data representation, achieving superior performance to those of previous approaches using attention-based KD [30,31] and initialization-based KD [34,32]. Therefore, in our work, we use an autoencoder-based input distiller, which learns the discrepancy between

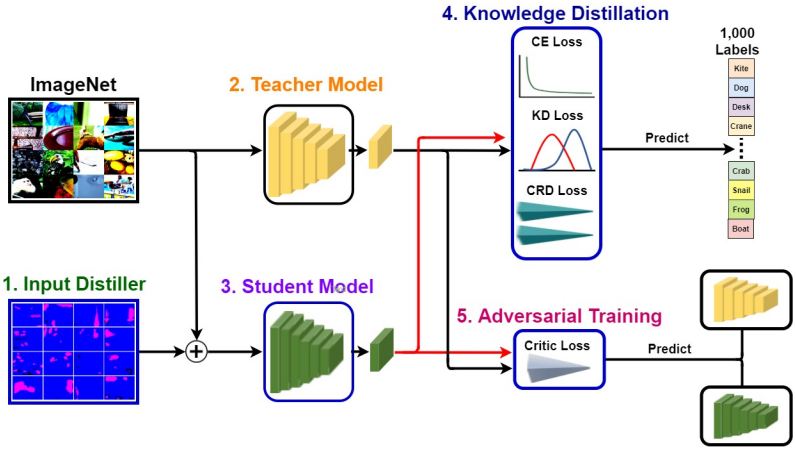

Fig. 3. Illustration of our training pipeline. CE, KD, and CRD denote Cross Entropy, Knowledge Distillation, and Contrastive Representation Distribution, respectively. Models in the two black boxes at the lower right corner represent the teacher model (top) and the student model (bottom). The CE, KD, and CRD losses are used to predict 1,000 labels, whereas the critic loss is used to predict whether the final feature map is from the teacher model or the student model. Inside a blue box is a model with trainable weights, while inside a black box is a model with frozen weights.

the teacher and student models by accounting for the loss computed by the aggregation of the critic loss, the KD loss [14], and the CRD loss [16].

3 Our Method

As shown in Fig. 3, depicting the details of our proposed method ExRep, the first step is to distill the knowledge of the teacher model for the student model. Here, the input distiller is trained using the pre-trained models.

EfficientNet. We implemented EfficientNet-B1 NoisyStudent, our teacher model, and EfficientNet-B1, our student model. Our choice of EfficientNet-B1 as the backbone is attributed to its relatively small number of parameters (7.8M), which enables efficient computation. The weights of the model pre-trained on JFT-300M in PyTorch are available here[3].

3.1 Input Distiller

Our input distiller, shown in Supp. Section C, is a ResNet-based autoencoder comprised of simple residual blocks, each of which consists of convolution (Conv), Instance Normalization (IN), and LeakyReLU layers. In ExRep, the input distiller learns the representational discrepancy between the teacher and student models

[3] https://github.com/rwightman/pytorch-image-models

through knowledge distillation. More specifically, let x_i denote the i^{th} input image and θ_g denote the weights of the input distiller f; then, the output $\tilde{x}_i$ of f with the same dimension as x_i is defined as

$$\tilde{x}_i = f_{\theta_g}(x_i) + x_i, \tag{1}$$

which is then fed into the student model. The output of the input distiller $\tilde{x}_i$ acts as a weight-based noisy sample equivalent to augmented data or adversarial examples.

Role of the input distiller. Our proposed pipeline comprises three stages: 1) knowledge distillation using the input distiller (Table 1), 2) application of the input distiller to the training of a classifier lacking extra training data (Table **??** in Supp.), and 3) inference without using the input distiller (Table 1). Note that in the first stage, we distill the knowledge acquired during training into the KD and CRD losses for the input distiller to learn the representation effect of the extra training data. Hence the role of the input distiller is not to enhance the effect of knowledge distillation itself, but to capture and later reuse the extra representation to assist the training of another classifier lacking extra training data. Removing the input distiller, we would fail to effectively capture the extra representation due to the insufficient number of trainable parameters.

3.2 Knowledge Distillation Loss

Using the KD function, proposed by Hinton et al. [14], the student model is trained such that its output is similar to that of the teacher model by minimizing the KL divergence. However, our approach differs in that we aim to imitate the teacher model's representation and produce the effect of using extra training data without actually using it by adding trainable weights to the input images.

Let y_i, $\hat{y}_i^T$, and $\hat{y}_i^S$ denote the i^{th} label, the label predicted by the teacher model, and that predicted by the student model, respectively. Then, the respective outputs z^T and z^S of the teacher and student models following activation by a softmax function σ are defined as

$$z^T = \sigma\left(\frac{\hat{y}_i^T}{\tau}\right), \quad z^S = \sigma\left(\frac{\hat{y}_i^S}{\tau}\right), \quad \sigma = \frac{\exp(\hat{y}_i)}{\Sigma_j \exp(\hat{y}_j)}, \tag{2}$$

where τ is the temperature. Using the softmax outputs z^T and z^S in Eq. 2, we first compute the KL divergence loss as follows:

$$\mathcal{L}_{KL} = \tau^2 KL(z^T, z^S). \tag{3}$$

We also compute the CE loss using $\hat{y}_i^S$ and y_i as follows:

$$\begin{aligned} \mathcal{L}_{CE} &= J(\hat{y}_i^S, y_i), \\ J(\hat{y}_i, y_i) &= -y_i log(\hat{y}_i). \end{aligned} \tag{4}$$

We then compute the final KD loss, which ought to be minimized during training, as the sum of the KL divergence loss and the CE loss, each weighted by λ_{KL} and λ_{CE}, respectively, as follows:

$$\underset{\theta_T}{\operatorname{argmin}}\, \mathcal{L}_{KD} = \lambda_{KL}\mathcal{L}_{KL} + \lambda_{CE}\mathcal{L}_{CE}. \tag{5}$$

3.3 Contrastive Representation Distillation Loss

The CRD loss [16] applies contrastive learning, whereby the model learns a closer representation for the pairs of the same class (positive sample pairs) than for those of different classes (negative sample pairs) in ImageNet.

Feature embedding. Let o^T and o^S denote the respective final feature maps of f_{θ_T} (the teacher model) and f_{θ_S} (the student model) before the logit layer. The embeddings of o^T and o^S through the linear transformation functions l_{θ_S} and l_{θ_T}, e^T and e^S are normalized using the $L2$ norm as follows:

$$\begin{aligned} e^T &= l_{\theta_T}(o_i^T)/\Sigma_i \|l_{\theta_T}(o_i^T)\|_2^2, \\ e^S &= l_{\theta_S}(o_i^S)/\Sigma_i \|l_{\theta_S}(o_i^S)\|_2^2, \\ o_i^T &= f_{\theta_T}(\tilde{x}_i), \quad o_i^S = f_{\theta_S}(\tilde{x}_i). \end{aligned} \tag{6}$$

The linear transformation functions l_{θ_T} and l_{θ_S} in Eq. 6 are updated by iterative gradient descent.

Negative sampling. Since the negative sample space is significantly larger than the positive sample space (999 classes v. 1 class), negative sampling is used for an efficient computation of negative sample pairs. A negative sample x_j is defined such that $y_i \neq y_j$. Tian et al. [16] demonstrated that negative sampling leads to higher classification performance than when using intra-class sampling. We define the score function s, the range of which falls within 0 and 1, based on negative sampling for the discriminator, which estimates the class probability as follows:

$$s(e^T, e^S) = \frac{\exp(e^T e^S/\tau)}{\exp(e^T e^S/\tau) + (N/M)}, \tag{7}$$

where N is the number of negative sample pairs and M is the dataset cardinality.

Contrastive loss. Let C denote a random variable equal to 0 when the input pairs are positive samples and 1 when they are negative samples. Since s is a score function, we treat the contrastive loss as a binary CE term, where the number of negative samples N is multiplied by the positive sample variable ($C = 0$) to offset the effect of larger samples among negative ones:

$$f_{CRD} = \mathbb{E}_{s(e^T, e^S|c=1)}[\log s(e^T, e^S)] + N\mathbb{E}_{s(e^T, e^S|c=0)}[\log(1 - s(e^T, e^S)], \tag{8}$$

where f_{CRD} denotes the CRD function. The CRD loss is then defined as:

$$\underset{\theta_T}{\operatorname{argmin}} \min_{o} \mathcal{L}_{CRD} = -\lambda_{CRD} f_{CRD}, \tag{9}$$

where λ_{CRD} is a balancing parameter. Our final CRD loss term accounting for feature embedding and negative sampling in Eq. 9 is denoted by $\mathcal{L}_{CRD}$.

3.4 Adversarial Training Loss

Adversarial training was first introduced in GANs [15]. In our work, we use adversarial training to provide extra representation to the student model. Similarly

Algorithm 1 Adversarial training algorithm, where m is the minibatch size.

Require: a target data $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$.
for each epoch **do**
  Sample minibatch X_m, $Y_m = \{(x_i, y_i), ..., (x_m, y_m)\}$
  Input distiller $\tilde{X_m} = f_{\theta_g}(X_m)$
  o_m^T, $o_m^S = f_{\theta_T}^T(X_m)$, $f_{\theta_S}^S(\tilde{X_m})$
  Update the critic by ascending the stochastic gradient:
    $\nabla_{\theta_d} \frac{1}{m} \Sigma_{i=1}^m [\log D(o_m^T) + \log(1 - D(o_m^S))]$.
  Update the input distiller by descending the stochastic gradient:
    $\nabla_{\theta_g} \frac{1}{m} \Sigma_{i=1}^m \log(1 - D(G(o_m^T)))$.
end for

to a discriminator, we introduce a critic that decides whether the final feature map is from the teacher model or the student model, while the input distiller plays the role of a generator. Let f_{θ_g} and f_{θ_d}, referred to as G and D, denote the input distiller and the critic model, respectively. From Eq. 1 and Eq. 6, we obtain the final feature map o^S of the student model as follows:

$$\begin{gathered} V(D,G) = \mathbb{E}_{x \sim P(x)}[\log D(o^T)] + \mathbb{E}_{\tilde{x} \sim P(\tilde{x})}[\log(1 - D(o^S)], \\ o^S = f_{\theta_s}(G(x) + x). \end{gathered} \tag{10}$$

Then, the critic loss $\mathcal{L}_Critic$ is defined as follows:

$$\min_G \max_D \mathcal{L}_{Critic} = \lambda_{Critic} V(D,G), \tag{11}$$

where λ_{Critic} is a balancing parameter. The critic loss $\mathcal{L}_{Critic}$ in Eq. 11 enables the input distiller to generate a representation much closer to that of the teacher model from that of the student model. Although our critic objective is similar to the projected gradient descent (PGD) [26], where examples generated by G attempt to deceive D through imitation of the teacher model's representation, ours differs in that our perturbation has trainable weights obtained from the input distiller.

Final loss function. Our final loss function $\mathcal{L}_{Final}$ is computed as the sum of $\mathcal{L}_{KD}$, $\mathcal{L}_{CRD}$, and $\mathcal{L}_{Critic}$ as follows:

$$\mathcal{L}_{Final} = \mathcal{L}_{KD} + \mathcal{L}_{CRD} + \mathcal{L}_{Critic}. \tag{12}$$

Practical implementation of critic loss. The implementation of $\mathcal{L}_{Critic}$ is presented in Algorithm 1. Note that in practice, we update D and G sequentially: D is updated first and then G is updated to deceive D. To perform this, we split $\mathcal{L}_{Final}$ into $\mathcal{L}_{Critic}$ and the remaining terms; D is updated separately, while G is updated based on $\mathcal{L}_{KD}$ and $\mathcal{L}_{CRD}$, while fixing the critic loss. Algorithm 2 shows the entire pipeline of ExRep, where stages 1 to 5 represent the procedures depicted in Fig. 3.

Algorithm 2 Pseudo code of ExRep, where n is the ImageNet size and (x_i, y_i) is the i^{th} input image and label.

Require: ImageNet data $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, the input distiller f_{θ_g}, the teacher model f_{θ_T}, the student model f_{θ_S}, the critic model $f_{\theta_{Critic}}$ and the weights θ_{CRD} of $\mathcal{L}_{CRD}$.

1: Input Distiller: $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ is fed to f_{θ_g}, and generates $\{\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_m\}$.
Input: Data (x_i, y_i)
Output: Data $(\tilde{x}_i, y_i)$
Objective: $\tilde{x}_i = f_{\theta_g}(x_i) + x_i$.

2: Teacher Model: The input images are fed to f_{θ_T}.
Input: Data (x_i, y_i)
Output: Final feature map o^T and prediction $\hat{y}_i^T$

3: Teacher Model: $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ added to the output of f_{θ_g} is fed to f_{θ_S}.
Input: Data $(\tilde{x}_i, y_i)$
Output: Final feature map o^S and prediction $\hat{y}_i^S$

4: Knowledge Distillation: Using o^T, o^S, $\hat{y}_i^T$, and $\hat{y}_i^S$, compute $\mathcal{L}_{CE}$ and $\mathcal{L}_{KL}$.
Input: Final feature maps o^T, o^S, and predictions $\hat{y}_i^T$, $\hat{y}_i^S$
Output: $\mathcal{L}_{KD}$
Objective: $\text{argmin}_{\theta_T} \mathcal{L}_{KD} = \lambda_{KL}\mathcal{L}_{KL} + \lambda_{CE}\mathcal{L}_{CE}$.

5: Adversarial Training: The critic model decides whether the final feature map is from the teacher model or the the student model.
Input: Final features o^T, o^S
Output: $\mathcal{L}_{Critic}$
Objective: $\min_G \max_D \mathcal{L}_{Critic} = \lambda_{Critic} V(D, G)$.

3.5 Inference and Application

During training, the input distiller is also trained based on the CE, KD, CRD, and Adv losses. During inference, the input distiller is not used; only the student model is evaluated on the validation set of ImageNet. We can also apply ExRep to other datasets, such as CIFAR10, to train a novel classifier in that dataset, using the input distiller pre-trained during training. We refer to this phase as the application stage, which is explained in Supp. Section A.

4 Experimental Results

In this section, we provide the details of our experiments, including the datasets, the baseline and pre-trained models, and the results. Our training process is presented in detail in Supp. Section D.

4.1 Datasets

ImageNet 2012. ImageNet [35] contains around 1.2 million images labeled with 1,000 categories. To standardize all images with different sizes, we resize them

to 224×224 and 240×240 using bicubic interpolation in EfficientNet-B0 and EfficientNet-B1, respectively. For efficient negative sampling, we use a memory buffer as in [16]. The classification performance is evaluated on the ImageNet validation set.

4.2 Baselines

EfficientNet. We conduct experiments on ImageNet with EfficientNet-B0 and B1 as the backbone networks. With a small number of parameters, EfficientNet achieved a performance comparable to those of ResNext [17] and NASNet [11]. We decide to use a small sized EfficientNet (B0 and B1) for efficient computation.
NoisyStudent. NoisyStudent used JFT-300M as extra training data for self-training. We implement NoisyStudent (EfficientNet-B1 and B0) to evaluate the performance on ImageNet, using the pre-trained models in PyTorch. The input image size is 240×240 (B1) and 224×224 (B0), and random crops are applied during the pre-training phase of NoisyStudent. We set the percentage of the center crop to be 88.2% for B1 and 87.5% for B0.
FixNet. FixNet also used JFT-300M as extra training data. However, their pre-trained weights in EfficientNet-B0 and B1 are not released. Therefore, we refer to the performance results presented by Touvron et al. [8] for comparison.
AdvProp. AdvProp achieved high performance without using extra training data by using adversarial examples to improve the classification performance through self-supervised training. We set the input size and the percentage of the center crop to be the same as those for NoisyStudent. Similarly, we use the $L2$ norm as in the original AdvProp [27] for our evaluation.

4.3 Pre-trained Models

We use NoisyStudent (EfficientNet-B0 and B1) as the teacher model, and EffcientNet-B0 and B1 pre-trained only on ImageNet as the student model. NoisyStudent is trained on samples of the extra training data through self-training, using additional data augmentation. On the contrary, FixNet's training process is more complicated: the model is pre-trained on a smaller size of inputs and fine-tuned on a larger size of inputs. Since our goal is to produce the effect of using extra training data without actually using it, we chose NoisyStudent as our teacher model.

4.4 Performance Evaluation

Top-1 accuracy and top-5 accuracy. We used top-1 and top-5 accuracy for the evaluation of the classification performance. Top-1 accuracy measures how the model correctly predicts the target label. Top-5 accuracy checks if the target label is included in one of the top 5 predictions. We chose these two metrics to examine how our model accurately selects the correct label, as well as how often the model includes the correct label within the top 5 predictions.

Table 1. Performance results. EfficientNet PyTorch represents the original EfficientNet model pre-trained on ImageNet without extra training data. All methods use EfficientNet-B0 and EfficientNet-B1 as the backbone networks. Note that both NoisyStudent and FixNet use extra training data (JFT-300M) during training on ImageNet, whereas AdvProp, EfficientNet PyTorch, and ExRep only use the ImageNet training data. The best results are highlighted in bold.

Method	Extra Training Data	Size	Top-1 ACC	Top-5 ACC
EfficientNet-B1 NoisyStudent	JFT-300M	240	81.3%	95.7%
EfficientNet-B0 NoisyStudent	JFT-300M	224	78.6%	94.3%
EfficientNet-B1 FixNet	JFT-300M	240	82.6%	96.5%
EfficientNet-B0 FixNet	JFT-300M	224	80.2%	95.4%
EfficientNet-B1 AdvProp	-	240	79.2%	94.3%
EfficientNet-B0 AdvProp	-	224	77.0%	93.2%
EfficientNet-B1 PyTorch	-	240	78.8%	94.1%
EfficientNet-B0 PyTorch	-	224	77.6%	93.5%
EfficientNet-B1 ExRep	-	240	**79.8**%	**94.5**%
EfficientNet-B0 ExRep	-	224	**77.9**%	**93.6**%

ExRep Performance. We present our performance results in Table 1, where the same test dataset (ImageNet 2012 benchmark) is used. Note that we use NoisyStudnet as the teacher model and EfficientNet PyTorch as the student model. We also use EfficientNet-B0 and EfficientNet-B1 as the backbone networks with input sizes 224×224 and 240×240, respectively. ExRep achieves 79.8% top-1 accuracy and 94.5% top-5 accuracy for EfficientNet-B1, and 77.9% top-1 accuracy and 93.6% top-5 accuracy for EfficientNet-B0 without extra training data, outperforming AdvProp.

EfficientNet PyTorch vs. ExRep. EfficientNet PyTorch is trained on ImageNet with the model in the work of Tan and Le [6], which is pre-trained on ImageNet without extra training data. We used EfficientNet PyTorch as the student model of ExRep. As shown in Table 1, ExRep achieves 1.0% higher top-1 accuracy and 0.4% higher top-5 accuracy for EfficientNet-B1 and 0.3% higher top-1 accuracy and 0.1% higher top-5 accuracy for EfficientNet-B0 compared to those of EfficientNet PyTorch for both EfficientNet-B1 and B0. EfficientNet-B1 shows a higher performance increase compared to that of EfficientNet-B0, because the teacher model (NoisyStudent) in EfficientNet-B0 NoisyStudent has lower incremental performance than that of EfficientNet-B1 NoisyStudent.

AdvProp vs. ExRep. AdvProp achieves 0.4% higher top-1 accuracy and 0.2% higher top-5 accuracy than those of EfficientNet PyTorch for EfficientNet-B1 but achieves lower performance for EfficientNet-B0. However, ExRep achieves 0.7% higher top-1 accuracy and 0.1% higher top-5 accuracy for EfficientNet-B1, and 0.9% higher top-1 accuracy and 0.4% higher top-5 accuracy for EfficientNet-B0. Therefore, ExRep exhibits greater generalization ability on ImageNet regardless of the model size (B1 and B0).

Table 2. Ablation study for the input distiller and adversarial training. Adv-training denotes adversarial training. In this experiment, we use EfficientNet-B0 and EfficientNet-B1 as the backbone network.

Method	Base Model	Top-1 ACC	Top-5 ACC	Reusable
w/o input distiller (KD+CRD)	EfficientNet-B0	78.0%	93.8%	X
w/o input distiller (KD+CRD)	EfficientNet-B1	80.1%	95.1%	X
w input distiller (KD+CRD+Adv)	EfficientNet-B0	77.9%	93.6%	O
w input distiller (KD+CRD+MSE)	EfficientNet-B0	77.6%	93.5%	O
w input distiller (KD+CRD+Adv)	EfficientNet-B1	79.9%	95.1%	O
w input distiller (KD+CRD+MSE)	EfficientNet-B1	79.7%	95.1%	O

EfficientNet-B0 vs. EfficientNet-B1. ExRep with EfficientNet-B1 achieves the highest top-1 and top-5 accuracy compared to ExRep with EfficientNet-B0. While EfficientNet-B1 achieves 1.2% higher top-1 accuracy and 0.6% higher top-5 accuracy compared to those of EfficientNet-B0, ExRep with EfficientNet-B1 achieves 1.9% higher top-1 accuracy and 0.9% higher top-5 accuracy.
NoisyStudent vs. FixNet. FixNet shows higher top-1 accuracy and top-5 accuracy for both EfficientNet-B1 and EfficientNet-B0 in general. As the teacher model has more informative representation, the student model learns useful representation as well. Therefore, we expect that FixNet will have higher distillation performance using our framework, although not experimented in this work, since the FixNet weights are currently unavailable.

5 Ablation Study

In this section, we present an ablation study to demonstrate the effectiveness of the input distiller and adversarial training (Section 3.1, Section 3.4, and Eq. 11) used in our ExRep: 1) the KD and CRD losses without the input distiller, 2) the KD, CRD, and adversarial losses with the input distiller, and 3) the KD, CRD, and Mean Squared Error (MSE) losses with the input distiller.
Input distiller effect. As shown in Table 2, we experiment with ExRep, which is trained on ImageNet. The input distiller may harm the performance of the original knowledge distillation, but only to a minimal degree, and most importantly, it renders the extra representation reusable for the training of a novel classifier without any available extra training data. In the ablation study presented in Table 2, the performance was compared for the exclusion of the adversarial and MSE losses using the input distiller to demonstrate the effects of those terms.
ResNeXt and Billion-scale dataset. We also carried out an experiment to validate the generalization of our method. Semi-weakly supervised ResNeXt (ResNeXt SWSL) is trained on the Billion-scale dataset [36], which is an internal dataset of Facebook. This ResNeXt SWSL achieves 82.1% top-1 accuracy on 224×224 size of ImageNet. On the other hands, The vanilla ResNeXt achieves 79.7% top-1 accuracy. Our ExRep achieves 81.1% top-1 accuracy. It represents

Table 3. Performance results on ResNeXt. ResNeXt SWSL represents semi-weakly supervised ResNeXT.

Method	Extra Training Data	Size	Top-1 ACC	Top-5 ACC
ResNeXt SWSL [36]	Billion-scale	224	82.1%	96.2%
ResNeXt	-	224	79.7%	94.6%
ResNeXt ExRep	-	224	81.1%	95.1%

that ExRep can adjust to different CNN architecture (EfficientNet and ResNeXt) and other extra training dataset (JFT-300M and Billion-scale).

6 Conclusion

In this paper, we introduce ExRep, a novel ImageNet image classification framework. Our proposed method does not require direct access to extra training data to achieve high classification performance, unlike the previous state-of-the-art approaches that exploited the JFT-300M dataset, which is not publicly available. We instead use a teacher model pre-trained on ImageNet as well as on JFT-300M to distill its knowledge acquired during training for the student model pre-trained only on ImageNet. ExRep outperforms AdvProp, the state-of-the-art, which does not use extra training data as in our case, and achieves a performance comparable to those of NoisyStudent and FixNet, which use JFT-300M data for training in addition to ImageNet. We show that ExRep can be applied to other datasets as well, demonstrating the effectiveness of our proposed method. While ExRep achieves a performance comparable to those of previous state-of-the-art approaches, there is still room for improvement. For instance, we plan to integrate BiT, NoisyStudent, and FixNet based on more complex backbone architectures, such as EfficientNet-B7, EfficientNet-L2, and BiT-L, which are more computationally expensive and time-consuming. We expect that the use of more complex teacher models will improve the classification performance; we also aim to further optimize ExRep. Our ExRep implementation is available here[4].

Acknowledgements

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00421, AI Graduate School Support Program (Sungkyunkwan University)), and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2019M3F2A1072217 and 2020R1C1C1006004). Also, this research was results of a study on the "HPC Support" Project, supported by the 'Ministry of Science and ICT' and NIPA.

[4] `https://github.com/cutz-j/ExRep`

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. (2012) 1097–1105
2. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
3. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015) 1–9
4. 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. (2016) 770–778
5. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 1251–1258
6. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946 (2019)
7. Xie, Q., Luong, M.T., Hovy, E., Le, Q.V.: Self-training with noisy student improves imagenet classification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 10687–10698
8. Touvron, H., Vedaldi, A., Douze, M., Jégou, H.: Fixing the train-test resolution discrepancy. In: Advances in Neural Information Processing Systems. (2019) 8252–8262
9. Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., Houlsby, N.: Big transfer (bit): General visual representation learning. arXiv preprint arXiv:1912.11370 (2019)
10. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation policies from data. arXiv preprint arXiv:1805.09501 (2018)
11. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 8697–8710
12. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. (2009)
13. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. (2011)
14. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
15. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. (2014) 2672–2680
16. Tian, Y., Krishnan, D., Isola, P.: Contrastive representation distillation. arXiv preprint arXiv:1910.10699 (2019)
17. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 1492–1500
18. Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., van der Maaten, L.: Exploring the limits of weakly supervised pretraining. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 181–196

19. Joulin, A., Van Der Maaten, L., Jabri, A., Vasilache, N.: Learning visual features from large weakly supervised data. In: European Conference on Computer Vision, Springer (2016) 67–84
20. Thomee, B., Shamma, D.A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., Li, L.J.: Yfcc100m: The new data in multimedia research. Communications of the ACM **59** (2016) 64–73
21. Zhai, X., Puigcerver, J., Kolesnikov, A., Ruyssen, P., Riquelme, C., Lucic, M., Djolonga, J., Pinto, A.S., Neumann, M., Dosovitskiy, A., et al.: A large-scale study of representation learning with the visual task adaptation benchmark. arXiv preprint arXiv:1910.04867 (2019)
22. Wu, Y., He, K.: Group normalization. In: Proceedings of the European conference on computer vision (ECCV). (2018) 3–19
23. Qiao, S., Wang, H., Liu, C., Shen, W., Yuille, A.: Weight standardization. arXiv preprint arXiv:1903.10520 (2019)
24. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. (2020) 702–703
25. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
26. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
27. Xie, C., Tan, M., Gong, B., Wang, J., Yuille, A.L., Le, Q.V.: Adversarial examples improve image recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 819–828
28. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
29. 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. (2017) 4133–4141
30. 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)
31. Huang, Z., Wang, N.: Like what you like: Knowledge distill via neuron selectivity transfer. arXiv preprint arXiv:1707.01219 (2017)
32. Heo, B., Lee, M., Yun, S., Choi, J.Y.: Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In: Proceedings of the AAAI Conference on Artificial Intelligence. Volume 33. (2019) 3779–3787
33. Oord, A.v.d., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 (2018)
34. 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)
35. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115** (2015) 211–252
36. Yalniz, I.Z., Jégou, H., Chen, K., Paluri, M., Mahajan, D.: Billion-scale semi-supervised learning for image classification. arXiv preprint arXiv:1905.00546 (2019)