Improving Augmentation Efficiency for Few-Shot Learning

WONHEE CHO1 AND EUNWOO KIM1 (Member, IEEE)

1 School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, Korea.

Corresponding author: Eunwoo Kim (eunwoo@cau.ac.kr)

This research was supported in part by the Chung-Ang University Research Scholarship Grants in 2020, in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government Ministry of Science and ICT (MSIP) [No. NRF-2020R1A4A078916].

ABSTRACT While human intelligence can easily recognize some characteristics of classes with one or few examples, learning from few examples is a challenging task in machine learning. Recently emerging deep learning generally requires hundreds of thousands of samples to achieve generalization ability. Despite recent advances in deep learning, it is not easy to generalize new classes with little supervision. Few-shot learning (FSL) aims to learn how to recognize new classes with few examples per class. However, learning with few examples makes the model difficult to generalize and is susceptible to overfitting. To overcome the difficulty, data augmentation techniques have been applied to FSL. It is well-known that existing data augmentation approaches rely heavily on human experts with prior knowledge to find effective augmentation strategies manually. In this work, we propose an efficient data augmentation network, called EDANet, to automatically select the most effective augmentation approaches to achieve optimal performance of few-shot learning without human intervention. Our method overcomes the disadvantages of relying on domain knowledge and requiring expensive labor to design data augmentation rules manually. We demonstrate the proposed approach on widely used FSL benchmarks (Omniglot and mini-ImageNet). Experimental results using three popular FSL networks show that ours improves performance over existing baselines through an optimal combination of candidate augmentation strategies.

INDEX TERMS Few-shot learning, automatic search, efficient augmentation.

I. INTRODUCTION

Over the past decade, we have witnessed remarkable performance gain with deep learning in many tasks, such as classification [2], [3], detection [4], [5], and segmentation [6], [7]. The extensive calculation of deep learning through artificial neural networks, advances in computing power, and a large number of labeled examples have allowed deep learning models to achieve performance improvement. However, many real-world scenarios do not allow us to access sufficient labeled data due to some reasons, including privacy, security, high labeling costs, and difficulty in managing data. Therefore, many researchers have attempted to learn deep learning algorithms with few examples, and the field of few-shot learning (FSL) [17], [18] has been recently emerged.

Few-shot learning recognizes patterns in data with few examples. In general, FSL approaches can be divided into two major streams; 1) how to make up for insufficient data by adding supporting data [22], [23] and 2) how to represent large parameter space covered by few training samples [25], [26]. FSL approaches can be further categorized into four families: metric-based [19]–[21], data augmentation-based [22]–[24] optimization-based [25], [26], and semantic-based approaches [27], [28].

In this work, we address both metric-based and data augmentation-based FSL approaches. Among them, metric-based FSL has been actively studied, and Matching Networks (MatchingNet) [19], Prototypical Networks (Prototypical-Net) [20], and Relation Networks (RelationNet) [21] have been popularly used. Note that it is important to learn an appropriate metric space for the approaches. One can learn the similarity between two samples, extract features from the samples, and calculate the distance between the features. MatchingNet compares the cosine distance between samples mapped in embedding space, and PrototypicalNet computes the Euclidean distances between prototype samples representing classes. RelationNet introduces a relation metric to compare relationships between samples or between classes.
Even if the metric-based FSL algorithms made an important contribution to the early FSL, they are still unsatisfying compared to traditional many-shot learning approaches in terms of task performance. Commonly, data augmentation is a de facto practical technique for improving task performance. Therefore, this study aims to find an optimal data augmentation strategy from a pool of candidate augmentation techniques for metric-based FSL.

In the data augmentation-based approach, few available samples are augmented to generate diverse samples to enrich the training experience. Data augmentation methods provide additional training data by transforming existing data samples into supplementary ones. The data augmentation-based FSL approaches [22]–[24] augment data by hallucinating the feature vectors from the training dataset. IDeMeNet [24] generates synthesized examples from a data augmentation method. However, data augmentation methods have different strategies to improve task performance, depending on the domain. It is important to note that such augmentation-based algorithms generate training samples via hand-craft data augmentation rules. This plays an important role in improving performance. However, those approaches rely on expert domain knowledge and thus require high-cost labor. Moreover, optimal augmentation rules can be problem-specific, making them hard to be applied to other tasks. Therefore, manually designed data augmentation strategies may produce suboptimal solutions.

We overcome the difficulty of manual design choice in augmentation-based FSL. Focusing on this point, we first apply various augmentation methods to the metric-based FSL algorithms. Since most of the existing FSL algorithms have manually applied the augmentation method, we propose an efficient data augmentation method, termed EDANet, that automatically searches for an optimal augmentation rule. Thus, it provides an optimal combination of candidate augmentation methods that help enrich the knowledge of the network. It consists of two phases: augmentation and classification phases. In the augmentation phase, we explore data augmentation policies that improve performance. We use a density matching algorithm [30] to find the best combination of data augmentation strategies automatically. After that, we evaluate the performance of metric-based FSL networks with the augmented dataset in the classification phase. Figure 1 gives an overview of the proposed EDANet. Our approach reduces the manual design efforts, improves the quality of augmented data, and enhances the generalization ability of FSL models. It also automatically searches for augmentation policies such as augmentation type and magnitude. To our knowledge, this is the first work to automate the search mechanism for augmentation-based FSL.

We apply the proposed method to a range of metric-based FSL methods and evaluate them using image classification benchmark datasets. We use the Omniglot [18] and mini-ImageNet [19] datasets as benchmarks and analyze the performance of the approaches in terms of different distance metrics. Experimental results show that EDANet improves the performance by providing an optimal combination of candidate data augmentation techniques. Specifically, EDANet significantly improves the classification accuracy over baseline methods (i.e., MatchingNet [19], PrototypicalNet [20], and RelationNet [21]) on mini-ImageNet, achieving 63.51% one-shot accuracy and 79.74% five-shot accuracy compared to the competing approaches. Our approach achieves 98.61% one-shot accuracy and 99.13% five-shot accuracy on Omniglot, outperforming existing augmentation-based FSL baselines. By modifying the backbone architecture to ResNet-18 and ResNet-50, EDANet improves the classification task by 5.67% on the Omniglot dataset and 1.97% on mini-ImageNet for one-shot accuracy, respectively.

II. RELATED WORK

A. FEW-SHOT LEARNING

Few-shot learning (FSL) aims to learn a model with a limited amount of labeled examples for predicting novel classes. FSL is also known as $n$-way $k$-shot learning, where $n$ and $k$ denote the number of classes and the number of data points per class, respectively. Each sub-task consisting of $n$-way $k$-shot is called an episode (mini-batch), which is why FSL is called episodic learning. A query and a support set are used for episodic training, where the support set is used to learn to solve a classification task, and the query set is used to evaluate the performance of the task. The major challenge of FSL is the insufficient supply of training examples, assuming that massive amounts of datasets are expensive to label correctly. Thus, FSL tries to improve the prediction capability and generalization performance
of a model with limited datasets. FSL approaches can be classified into four categories: metric-based [19]–[21], data augmentation-based [22]–[24], optimizer-based [25], [26], and semantic-based approaches [27], [28]. Among them, we address metric-based and data augmentation-based approaches that are closely related to this work.

Metric-based approaches [19]–[21] learn a representation of data with a metric in the feature space. These approaches transform data into a low-dimensional subspace, cluster the transformed samples, and compare the clusters using a metric function. MatchingNet [19] is one of the popular metric-based methods that compute the cosine distance to classify query samples. It proposes the full context encoding module (FCE), which conditions the weights on the whole support set across classes. PrototypicalNet [20] compute an average feature, called a prototype, for each class in the support set, then classify query samples by calculating the Euclidean distance between each prototype and the samples in the query set. RelationNet [21] classify samples of novel classes by calculating the relation score between the query and the sample of each novel class.

### B. DATA AUGMENTATION

Data augmentation is a practical strategy used for a variety of learning-based tasks. Random crop, rotation, clipping, flip, scaling, and color transformation are used as baseline augmentation methods [1], [2], [16] when applying them to image benchmark datasets, such as CIFAR [10] and ImageNet [9]. Mixup [12] generates an augmented image by mixing two images by linear interpolation. Cutout [13] is a kind of regional dropout strategy where random patches are replaced with zeros (black pixel). Recently, CutMix [14] has been proposed to cut and paste a part of a randomly selected image into another image. Note that such augmentation techniques have been naturally used in supervised learning. The chronic problem of FSL is that there are not enough training samples. Since FSL is a data-hungry method, it also deploys data augmentation approaches [22]–[24]. SGM and PMN [22], [23] generate additional samples from trained hallucinators. Based on this, they propose strategies to improve the performance of FSL by using the generated samples. iDeMeNet [24] has adaptively fused samples from the support set to generate synthesized samples. It trains an embedding sub-module, which maps samples to feature representations and then performs FSL.

However, the aforementioned approaches rely on hand-crafted rules and require task-specific domain knowledge and cumbersome exploration to find optimal augmentation strategies. We break away from the augmentation techniques that have been routinely applied in the existing metric-based FSL. In our work, we do not generate synthetic samples but explore promising augmentation techniques to find an optimal augmentation strategy. As such, we do not rely on hand-crafted rules and require task-specific knowledge to find the optimal augmentation strategy. We propose an efficient automatic augmentation method for FSL.

### C. AUTOMATED LEARNING

One promising way is to find data augmentation methods automatically [29]–[32]. Recently, automating the design process of a neural network, called neural architecture search (NAS) [8], has been proposed to search for an optimal network architecture and to reduce the manual design efforts. Automating augmentation using NAS has been recently proposed in the field of data augmentation [29], [30], [32]. AutoAugment [29] uses an RNN controller to augment training data with a randomly selected augmentation method. Initially, all augmentation methods are explored uniformly, and we find optimal augmentation techniques that yield the best performance based on a reward function. While AutoAugment has achieved promising results, it is less efficient and expensive. PBA [32] generates augmentation policies based on population-based training [33]. Fast AutoAugment [30] uses hyperparameter optimization to explore optimal augmentation policies. Unlike the above-mentioned methods, in this work, we focus on developing an automatic augmentation method for the field of FSL. We adopt the augmentation strategy in [30] to accelerate our augmentation phase, which will be described in the following section.

### III. METHODOLOGY

We present an automatic augmentation strategy for the metric-based FSL without requiring domain knowledge and painful design efforts. The proposed Efficient Data Augmentation network, termed EDANet, consists of augmentation and classification phases. In the augmentation phase, augmentation data is obtained by automatically exploring the most efficient augmentation policy for the FSL model from the original data. In the classification phase, the FSL network is trained with the augmented training data, and then the performance is evaluated with the original test dataset. We describe the proposed augmentation framework and few-shot augmentation with the proposed framework in Section III-A.

In Section III-B, we apply EDANet to popular FSL problems.

#### A. EDANET: EFFICIENT DATA AUGMENTATION NETWORK

Manual data augmentation techniques generally require expert knowledge and painstaking design efforts despite the advantage of improving performance by enriching training data. As a remedy, we automate the augmentation procedure to find an optimal strategy and improve task performance. The search space of the automatic augmentation process contains diverse augmentation techniques, listed in Table 1. An augmentation operation $\mathcal{O}$ receives a sample $x$ given the magnitude $\lambda$. The result can be either $\mathcal{O}(x; \lambda)$ with probability $p$ or the original $x$ itself with probability $1 - p$. For example, when the rotation is selected as an augmentation operation, the magnitude becomes the degree. In the augmentation phase, we first introduce the search space of the augmentation techniques (operations) and perform density matching to find optimal augmentation policies. Figure 2 (b) shows the augmentation phase of EDANet. We follow the
search strategy presented in [30] to find a good augmentation policy for FSL.

**Auto Augmentation Phase.** Let \( O \) denotes the set of augmentation operations, which transforms an input sample by applying the operation. Specifically, each augmentation operation \( O_i \) contains a parameter, the magnitude \( \lambda_i \). The input samples are augmented by a policy consisting of \( t \) different sub-policies \( \tau \)'s, and each \( O_i(x; \lambda) \) is applied to transform \( x \) with the probability \( p \). Figure 2 (a) shows an example of augmenting samples by each sub-policy \( \tau \). One can generate \( T(D) \) indicating a set of augmented samples of dataset \( D \) transformed by all sub-policies \( \tau \in T \):

\[
T(D) = \bigcup_{\tau \in T} \{(\tau(x), y) : (x, y) \in D\}. \tag{1}
\]

Note that the dataset \( D \) is divided into \( k \)-folds. Each fold consists of \( D_{\text{model}} \) and \( D_{\text{aug}} \). We find a promising policy giving the highest performance for \( D_{\text{aug}} \) based on model \( M(\theta) \) that is trained by \( D_{\text{model}} \). Using the Bayesian optimization approach [30], top-\( n \) augmentation methods are selected based on the minimum error rates of classifier \( \theta \) when predicting dataset \( D_{\text{aug}} \). All the best augmentation methods are merged from each fold.

**Density Matching.** Our goal is to find an efficient augmentation policy that matches the density of \( D_{\text{model}} \) and \( D_{\text{aug}} \). We follow the practice in [30] that divides the original \( D_{\text{train}} \) into \( D_{\text{model}} \) and \( D_{\text{aug}} \) that are used for learning the

### TABLE 1: Data augmentation include geometric transformations, color transformations, etc. Affine transformation is a type of geometric transformation, such as rotation and translation. Color enhancement performs augmentation in the color channel space, such as invert and equalize. Recently, other image manipulation methods have been proposed, such as cutout [13] and sample pairing [15].

| Operation          | Magnitude \( \lambda \) |
|--------------------|--------------------------|
| **Affine transformation** |                          |
| ShearX             | Continuous               |
| ShearY             | Continuous               |
| TranslateX         | Continuous               |
| TranslateY         | Continuous               |
| Rotate             | Continuous               |
| **Color enhancement** |                          |
| AutoContrast       | None                     |
| Invert             | None                     |
| Equalize           | None                     |
| Solarize           | Discrete                 |
| Posterize          | Discrete                 |
| Contrast           | Continuous               |
| Color              | Continuous               |
| Brightness         | Continuous               |
| Sharpness          | None                     |
| **Others**         |                          |
| Cutout             | Discrete                 |
| Sample Pairing     | Continuous               |

FIGURE 2: An overall procedure of automatic augmentation by EDANet. (a) An example of augmentation from the searched policies. Each operation \( O_i \) given the magnitude \( \lambda_i \) is called with the probability \( p_i \). (b) The proposed method splits the train data set \( D_{\text{train}} \) into \( k \)-folds, each of which consists of two data sets \( D_{\text{model}} \) and \( D_{\text{aug}} \). Then model parameter \( \theta \) is trained on each \( D_{\text{model}} \). After training model \( M(\theta) \), the algorithm evaluates the augmentation policies on \( D_{\text{aug}} \). The top-\( n \) policies obtained from each of the \( k \)-folds are appended to an augmentation list \( T_*(D_{\text{train}}) \). Finally, we measure the performance of the network model with \( T_*(D_{\text{train}}) \).
model parameter $\theta$ and exploring the augmentation policy $T$, respectively. To find a set of learned augmentation policies, we have

$$T_* = \operatorname*{argmax}_{\tau} \mathcal{R}(\theta \mid D_{\text{aug}}), \quad (2)$$

where $\mathcal{R}(\theta \mid D_{\text{aug}})$ gives the accuracy using the model parameter $\theta$ for $D_{\text{aug}}$. We can find a policy based on the model trained with $D_{\text{model}}$. In other words, it minimizes the distance between the density of $D_{\text{model}}$ and the density of $T(D_{\text{aug}})$. We optimize the search process of augmentation strategies in EDANet through Bayesian optimization [30].

### Classification Phase

After the augmentation phase, we learn few-shot classification networks with the augmented dataset $T_*(D_{\text{train}})$, which contains $\{x_1, y_1, \ldots, x_n, y_n\}$, where $x$ is a sample and $y$ is the class label. After that, we test the proposed method with the original test set.

Figure 3 shows the classification phase of the proposed method. In the training stage, each task takes the form of $L_N$-way $k$-shot, and it consists of a support set $X^s$ and a query set $X^q$. Then, we train the classification model $M$ that minimizes $L_N$-way prediction loss for the query set, also known as episodic training. In the testing stage, the episode consists of a novel support set and a novel query set. The classification model $M$ can predict novel classes. The model $M$ can be MatchingNet [19], PrototypicalNet [20], or RelationNet [21] as the backbone of the proposed method, which will be described in the following section.

### B. FEW-SHOT LEARNING OBJECTIVE FUNCTIONS

The support set and query set for training FSL are extracted from $D_{\text{aug}}$ and $D_{\text{model}}$, respectively. The support set $X^s$ contains $m$ examples as $(x^s_1, y^s_1), \ldots, (x^s_m, y^s_m)$. The query set $X^q$ contains $t$ examples as $(x^q_1, y^q_1), \ldots, (x^q_t, y^q_t)$.

1) Matching Networks

MatchingNet [19], given a query sample $x^q$ (a novel unseen sample), predicts the class of $x^q$ by comparing it with the support set. The prediction of class for the query sample is defined as

$$\hat{y} = \sum_{i=1}^{k} \operatorname{Attention}(x^q, x^s_i) \cdot y^s_i, \quad (3)$$

where $\hat{y}$ is the predicted class of the query sample. $\operatorname{Attention}(\cdot, \cdot)$ is a simple attention mechanism between $\hat{x}$ and $x^s_i$, which is the softmax function over the cosine distance between $\hat{x}$ and $x^s_i$ as follows:

$$\operatorname{Attention}(\hat{x}, x^s_i) = \frac{e^{\cos(f(\hat{x}), g(x^s_i))}}{\sum_{j=1}^{k} e^{\cos(f(\hat{x}), g(x^q_j))}} \quad (4)$$

To be specific, two embedding functions $f$ and $g$ are used to learn embeddings of $\hat{x}$ (input in the query set) and $x^s_i$ (input in the support set), respectively. We convert the support set label into one-hot encoding and multiply them with attention to get the probability of $\hat{y}$ belonging to each class in the support set. Then, we select $\hat{y}$ with the maximum probability value as the class label.
2) Prototypical Networks

PrototypicalNet [20] learn the embeddings of the data points using the embedding function, \( f_\phi \), where \( \phi \) is the set of parameters of the embedding function. The prototype represents the mean embedding of data points in each class. The prototype \( c_j \) of the \( j \)-th class is calculated using the embedding of data points for the class as

\[
e_{j} = \frac{1}{|X|} \sum_{i} f_\phi(x_i). \tag{5}
\]

Then, the softmax function is applied after calculating the Euclidean distance \( d(\cdot, \cdot) \) between the query set embedding and the class prototype. Through this, the probability of the query sample of the class \( j \) is predicted as follows:

\[
p(y = j|x^q) = \frac{\exp(-d(f_\phi(x^q), c_j))}{\sum_j \exp(-d(f_\phi(x^q), c_j))}. \tag{6}
\]

Finally, we define the loss function as the negative log probability of the class \( j \) as

\[
\mathcal{L}(\phi) = -\log p(y = j|x^q), \tag{7}
\]

and we minimize the loss using SGD.

3) Relation Networks

RelationNet [21] consist of two important functions: the embedding function, denoted by \( f_\eta \), and the relation function, denoted by \( g_\varphi \), where \( \eta \) and \( \varphi \) are the parameters of the embedding and relation functions, respectively. It first takes a sample, \( x_i \), from the support set and feeds it into the embedding function to extract the feature. Similarly, it learns the embedding of a query sample \( x^q \) by passing through the embedding function, \( f_\eta(\cdot) \). Then, it combines \( f_\eta(x_i) \) and \( f_\eta(x^q) \) using the concatenation operation, i.e., \( \text{concat}(f_\eta(x_i), f_\eta(x^q)) \), which is fed into the relation function, \( g_\varphi \). The function generates the relation score ranging from 0 to 1:

\[
r_{ij} = g_\varphi(\text{concat}(f_\eta(x_i), f_\eta(x^q))). \tag{8}
\]

This represents the similarity between the samples in the support and query sets. For RelationNet, we use the mean squared error as the loss function, which is computed as

\[
\arg \min_{\varphi, \eta} \sum_{i=1}^{m} \sum_{j=1}^{n} (r_{ij} - \alpha)^2, \tag{9}
\]

where \( \alpha \) is 1 if \( y_i = y_j \) and 0 otherwise.

IV. EXPERIMENTS

In this section, we show the results of our approach on mini-ImageNet and Omniglot datasets and compare the results with other few-shot approaches. We summarize the proposed method as follows. The proposed method first finds an efficient augmentation policy in the augmentation phase and applies it to the original training set \( \mathcal{D}_{\text{train}} \) to provide \( \mathcal{T}_\text{c}(\mathcal{D}_{\text{train}}) \). Then, we add the augmented data to the \( \mathcal{D}_{\text{train}} \) and use it in the classification phase. In the classification phase, the few-shot models described in Section III-B are trained with the augmented dataset \( \mathcal{T}_\text{c}(\mathcal{D}_{\text{train}}) \) and are evaluated with the test set \( \mathcal{D}_{\text{test}} \). We run every experimental scenario independently five times and report the average results. All the experiments were implemented based on the PyTorch library [36].

A. SETTINGS

1) Datasets

We used the mini-ImageNet and Omniglot benchmarks to evaluate the few-shot classification algorithms. The mini-ImageNet dataset proposed in [19] is derived from the ILSVRC-12 dataset [11]. It consists of 60,000 color images of size 84x84 and has 100 classes with 600 examples each. We split the dataset into 64 training, 16 validation, and 20 test classes. Omniglot [18] contains 1,623 handwritten characters from 50 different alphabets. There are 20 examples associated with each character, and each example is drawn by different people. We applied automatic augmentation to 1,200 characters for training and used the remaining 423 characters for testing. We follow a similar procedure to the metric-based approaches by resizing the image into 28x28. Metric-based approaches [19]–[21] usually augment the training set by rotating the images. However, for the mini-ImageNet dataset, we used the augmentation techniques shown in Table 1 to find and apply the optimal augmentation policy. Note here that since the Omniglot dataset is a grayscale image, of the 16 operations shown in Table 1, only five affine transformation methods were chosen as the augmentation candidates. The dataset to which the efficient augmentation methods are applied is added to the original training set through the augmentation phase. After that, a few-shot classification task is performed through the classification phase. We compared with the augmentation-based approaches, SGM [22], PMN [23] and IDEMeNet [24], for few-shot recognition.

2) Implementation Details

Following the practice in [30], we applied the steps to make implementation easier. First, we split the training data, \( \mathcal{D}_{\text{train}} \), into five folds. Second, we perform exploration-and-exploitation using HyperOpt [35] to search for optimal augmentation policies. We selected augmentation operations from the PIL python library\(^1\). We apply Test-Time-Augmentation (TTA) [34] to estimate the appropriate augmentation policy without repeated training. After training the FSL model, an effective augmentation policy is estimated from a pool of candidate augmentation policies. Finally, the augmentation policy that achieves the highest performance is selected. The original implementations of MatchingNet [19], PrototypicalNet [20], and RelationNet [21] use the embedding architecture containing four convolution layers (Conv-4). Other than the architecture, we used ResNet-18 and ResNet-50 as additional embedding functions for diverse experiments. We have the same training procedure for the

\(^1\)https://python-pillow.org/
embedding functions as those of existing approaches. We used the 5-way classification setting in the experiments.

### B. RESULTS

1) mini-ImageNet

We performed the experiments on 5-way 1-shot and 5-shot according to the common setup in metric-based FSL. Since the mini-ImageNet dataset consists of RGB images, EDANet considered all 16 augmentation operations listed in Table 1. We obtained the classification accuracy by averaging over 600 randomly generated episodes from the test set. We compare EDANet with existing augmentation-based algorithms SGM [22], PMN [23], and IDeMeNet [24]. EDANet was trained under three different embedding functions (models) described in Section III-B.

We report the accuracy of the compared methods on mini-ImageNet in Table 2. EDANet achieves the best accuracy when using PrototypicalNet as the embedding function. We employed ResNet-50 as another backbone for a fair comparison with data augmentation-based approaches [22]–[24], which shows better performance than the Conv-4 architecture. The deeper backbone network performs better than shallow networks on every shot. Notably, our approach based on Conv-4 gives similar or better performance than the compared methods employing the ResNet-50 backbone. This shows that the automatic augmentation method is effective in FSL, compared to other approaches with hand-crafted augmentation rules. Note that 5-shot learning achieves higher accuracy than 1-shot learning, and the augmented dataset helps the network generalize better. Compared to SGM [22], PMN [23], and IDeMeNet [24], we observe that EDANet performs better than those approaches in both 1-shot and 5-shot. The 1-shot performance of the proposed method improved over the compared approaches. For EDANet based on MatchingNet, 5-shot learning performance is lower than that of SGM. However, EDANet performs better than other methods on average for the 5-shot learning.

| Method                  | Model     | Backbone   | Distance Metric | 5-Way 1-shot | 5-Way 5-shot |
|------------------------|-----------|------------|-----------------|--------------|--------------|
| No Augmentation (our impl.) | MatchingNet | Conv-4     | Cosine Dist.    | 91.81%       | 92.33%       |
|                        | PrototypicalNet | Conv-4     | Euclidean Dist. | 92.29%       | 93.22%       |
|                        | RelationNet   | Conv-4     | Relation Score  | 92.41%       | 93.50%       |
| Manual Augmentation (our impl.) | MatchingNet | Conv-4     | Cosine Dist.    | 93.25%       | 94.51%       |
|                        | PrototypicalNet | Conv-4     | Euclidean Dist. | 95.01%       | 95.88%       |
|                        | RelationNet   | Conv-4     | Relation Score  | 95.36%       | 96.02%       |
| Auto Augmentation (EDANet) | MatchingNet | Conv-4     | Cosine Dist.    | 94.67%       | 95.35%       |
|                        | PrototypicalNet | Conv-4     | Euclidean Dist. | 96.12%       | 96.78%       |
|                        | RelationNet   | Conv-4     | Relation Score  | 96.64%       | 97.13%       |
| Auto Augmentation (EDANet) | MatchingNet | ResNet-18  | Cosine Dist.    | 97.75%       | 98.81%       |
|                        | PrototypicalNet | ResNet-18  | Euclidean Dist. | 98.61%       | 98.97%       |
|                        | RelationNet   | ResNet-18  | Relation Score  | 98.57%       | 99.13%       |

TABLE 3: Few-shot classification results on Omniglot.
manual augmentation approaches. Table 4 shows the results with respect to different augmentation strategies. No augmentation follows the method suggested in existing papers without adding augmentation techniques. Manual augmentation applies three commonly used augmentation techniques: flip, crop, and rotation. Auto augmentation corresponds to EDANet, which explores the best augmentation rule among 16 augmentation techniques listed in Table 1. In the study, we used the small backbone network (Conv-4). It can be seen that the proposed automatic augmentation yields the best accuracy compared to other methods by large margins. The proposed method automatically explores optimal augmentation methods without requiring expert efforts, resulting in more than 7% performance improvement over no augmentation for both 1-shot and 5-shot results. Similarly, it outperforms the manual augmentation method by a margin of 4% to 6%. The experiments show that EDANet is a promising candidate to achieve competitive performance without laborious manual design costs.

2) Results on various augmentation methods

We conducted additional experiments to see the performance of some augmentation operations in Table 1. We selected eight augmentation operations (including no augmentation) and applied each of them as an augmentation method under the MatchingNet framework. The few-shot learning results from the augmentation strategy are summarized in Table 5. The performance of MatchingNet without augmentation gives the lowest performance (37.1% for 1-shot learning and 50.4% for 5-shot learning). All of the augmentation operations give a performance improvement of 2% or more over the method without augmentation. We also observe that augmentation methods (e.g., translate, contrast, brightness, etc.) other than rotation can improve task performance better in metric-based FSL. Note also that the color enhancement operations (contrast, brightness, saturation, and hue) do not improve performance in metric-based FSL. In addition, we applied some combinations of the

### TABLE 4: Few-shot classification results for ablation study on mini-ImageNet.

| Method                  | Model              | Distance Metric | Backbone | 1-shot  | 5-shot  |
|------------------------|--------------------|-----------------|----------|---------|---------|
| No Augmentation        | MatchingNet [19]   | Cosine Dist.    | Conv-4   | 43.56%  | 55.31%  |
|                        | PrototypicalNet [20]| Euclidean Dist. | Conv-4   | 49.42%  | 68.20%  |
|                        | RelationNet [21]   | Relation Score  | Conv-4   | 50.44%  | 65.32%  |
| Manual Augmentation    | MatchingNet        | Cosine Dist.    | Conv-4   | 45.60%  | 58.95%  |
|                        | PrototypicalNet    | Euclidean Dist. | Conv-4   | 52.63%  | 70.33%  |
|                        | RelationNet        | Relation Score  | Conv-4   | 53.22%  | 68.12%  |
| Auto Augmentation (EDANet) | MatchingNet    | Cosine Dist.    | Conv-4   | 49.37%  | 63.87%  |
|                        | PrototypicalNet    | Euclidean Dist. | Conv-4   | 57.01%  | 75.11%  |
|                        | RelationNet        | Relation Score  | Conv-4   | 57.84%  | 74.61%  |

2) Omniglot

We conducted another experiment on Omniglot and computed few-shot classification accuracy by averaging over 1,000 randomly generated episodes from the test set. We experimented with the proposal based on two different backbone architectures in metric-based few-shot classification. For comparison, we compared with MatchingNet [19], PrototypicalNet [20], and RelationNet [21] without data augmentation and with manually augmentations (rotation, crop, and flip). Since the Omniglot dataset contains black-and-white binary images, we applied a total of five affine transformation methods except for the color enhancement techniques shown in Table 1.

The results of the compared approaches are shown in Table 3. The proposed method achieved better performance than the baseline methods for the dataset when PrototypicalNet was used under 5-way 1-shot learning and RelationNet was used under 5-way 5-shot learning. Compared with the manually augmented using the Conv-4 backbone, the proposed method achieves better performance by about 1%. As shown in the table, there is a performance gap depending on the backbone architecture. The RelationNet using the relation score produces the best performance among the distance metrics for both 1-shot and 5-shot. PrototypicalNet and RelationNet outperform MatchingNet using the cosine distance, and their performance differences are marginal. EDANet using the automatic augmentation strategy gives better performance than the manual augmentation-based approaches under the same models (embedding function). Even with the larger backbone network, ResNet-18, it further improves the performance by 2% to 3%. The experiments show that EDANet finds an optimal data augmentation strategy from the pool of candidate augmentation techniques for metric-based FSL.

### C. ABLATION STUDY

1) Effectiveness of automatic augmentation

We demonstrate the effectiveness of the proposed EDANet that explores an optimal augmentation strategy from a pool of candidate strategies in metric-based FSL and compare it with manual augmentation approaches. Table 4 shows the results with respect to different augmentation strategies. No augmentation follows the method suggested in existing papers without adding augmentation techniques. Manual augmentation applies three commonly used augmentation techniques: flip, crop, and rotation. Auto augmentation corresponds to EDANet, which explores the best augmentation rule among 16 augmentation techniques listed in Table 1. In the study, we used the small backbone network (Conv-4). It can be seen that the proposed automatic augmentation yields the best accuracy compared to other methods by large margins. The proposed method automatically explores optimal augmentation methods without requiring expert efforts, resulting in more than 7% performance improvement over no augmentation for both 1-shot and 5-shot results. Similarly, it outperforms the manual augmentation method by a margin of 4% to 6%. The experiments show that EDANet is a promising candidate to achieve competitive performance without laborious manual design costs.
augmentation methods to see whether the combination leads to performance improvement. The number of combinations of choosing two out of those above eight selected augmentation operations is 28. We randomly selected three of these methods. The table shows that combining two augmentation methods performs better than the standalone augmentation method (at least 2% improvement). Here, we would like to note that since there are a large number of combinations to select two of the 16 augmentation operations, it is nearly impossible to select appropriate augmentation methods manually. Due to this reason, we can say that our EDANet can be a promising strategy for the success of FSL.

Figure 4 shows some examples of augmented images with their corresponding policies and parameter values selected in EDANet. EDANet provides the probability and magnitude for each policy combination and selects an augmented image from the combination of multiple augmentation operations. This reveals that the proposed method is capable of exploring various augmentation policies for different images.

**TABLE 5:** Performance with respect to different usages of augmentation methods using MatchingNet on mini-ImageNet. There is a slight gap between the results reported in [19] and the results from our implementation due to the difference of the implementation.

| Model                | Augmentation Method                                            | 1-shot | 5-shot |
|----------------------|----------------------------------------------------------------|--------|--------|
|                      | Rotate (90, 180, 270 degrees)                                   | 41.2%  | 56.2%  |
|                      | No augmentation                                                | 37.1%  | 50.4%  |
|                      | Rotate (90, 180, 270 degrees)                                   | 40.2%  | 55.6%  |
|                      | Rotate (1 ~ 270 degrees)                                        | 40.4%  | 55.4%  |
|                      | TranslateX                                                     | 41.5%  | 57.2%  |
|                      | TranslateY                                                     | 41.4%  | 56.7%  |
|                      | Contrast                                                       | 43.5%  | 59.0%  |
|                      | Brightness                                                     | 44.3%  | 59.6%  |
|                      | Saturation                                                     | 43.8%  | 58.8%  |
|                      | Equalize                                                       | 43.9%  | 59.3%  |
|                      | Rotate + Contrast                                              | 46.5%  | 60.7%  |
|                      | TranslateX + Equalize                                          | 46.8%  | 61.2%  |
|                      | Rotate + Brightness                                            | 47.1%  | 61.5%  |
| MatchingNet (our impl.) | Rotate (90, 180, 270 degrees)                                   | 41.2%  | 56.2%  |
|                      | Rotate (90, 180, 270 degrees)                                   | 40.2%  | 55.6%  |
|                      | Rotate (1 ~ 270 degrees)                                        | 40.4%  | 55.4%  |
|                      | TranslateX                                                     | 41.5%  | 57.2%  |
|                      | TranslateY                                                     | 41.4%  | 56.7%  |
|                      | Contrast                                                       | 43.5%  | 59.0%  |
|                      | Brightness                                                     | 44.3%  | 59.6%  |
|                      | Saturation                                                     | 43.8%  | 58.8%  |
|                      | Equalize                                                       | 43.9%  | 59.3%  |
|                      | Rotate + Contrast                                              | 46.5%  | 60.7%  |
|                      | TranslateX + Equalize                                          | 46.8%  | 61.2%  |
|                      | Rotate + Brightness                                            | 47.1%  | 61.5%  |

**FIGURE 4:** Examples of augmented images and their corresponding policies and parameter values in the proposed method on mini-ImageNet.
V. CONCLUSION

Data augmentation is one of the promising methods for the success of few-shot learning that requires a small number of data. In this work, we have proposed an efficient automatic augmentation approach for few-shot learning to explore the best augmentation strategies from a pool of candidate augmentation operations and to reduce the hand-crafted design efforts. The proposed method, EDANet, searches for a combination of augmentation techniques given various candidates augmentation methods using Bayesian optimization and density matching. We have applied the proposed method to three popular few-shot learning baseline models using different distance metrics. Experimental results show that the automatic augmentation rule yields better performance than manual augmentation-based counterparts under the same model, showing its effectiveness in few-shot learning.

REFERENCES

[1] Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.
[2] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
[3] Sun, M., Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning (pp. 6105-6114). PMLR.
[4] Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).
[5] Lin, T. Y., Goyal, P., Girshick, R., He, K., Dollár, P. (2017). Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision (pp. 2980-2988).
[6] Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence, 40(4), 834-848.
[7] He, K., Gkioxari, G., Dollár, P., Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).
[8] Zoph, B., Le, Q. V. (2016). Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578.
[9] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). IEEE.
[10] Krizhevsky, A., Hinton, G. (2009). Learning multiple layers of features from tiny images.
[11] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. International journal of computer vision, 115(3), 211-252.
[12] Zhang, H., Cisse, M., Dauphin, Y. N., Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412.
[13] DeVries, T., Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552.
[14] Sun, S., Han, D., Oh, S. J., Chun, S., Choe, J., Yoo, Y. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 6023-6032).
[15] Inoue, H. (2018). Data augmentation by pairing samples for images classification. arXiv preprint arXiv:1801.02929.
[16] Shorten, C., Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of Big Data, 6(1), 1-48.
[17] Fei-Fei, L., Fergus, R., Perona, P. (2006). One-shot learning of object categories. IEEE transactions on pattern analysis and machine intelligence, 28(4), 594-611.
[18] Lake, B., Salakhutdinov, R., Gross, J., Tenenbaum, J. (2011). One shot learning of simple visual concepts. In Proceedings of the annual meeting of the cognitive science society (Vol. 33, No. 33).
[19] Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D. (2016). Matching networks for one shot learning. Advances in neural information processing systems, 29, 3630-3638.
[20] Snell, J., Swersky, K., Zemel, R. S. (2017). Prototypical networks for few-shot learning. arXiv preprint arXiv:1703.05175.
[21] Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., Hospedales, T. M. (2018). Learning to compare: Relation network for few-shot learning. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1199-1208).
[22] Hariharan, B., Girshick, R. (2017). Low-shot visual recognition by shrinking and hallucinating features. In Proceedings of the IEEE International Conference on Computer Vision (pp. 3018-3027).
[23] Wang, Y. X., Girshick, R., Hebert, M., Hariharan, B. (2018). Low-shot learning from imaginary data. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7278-7286).
[24] Chen, Z., Fu, Y., Wang, Y. X., Ma, L., Liu, W., Hebert, M. (2019). Image deformation meta-networks for one-shot learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 8680-8689).
[25] Finn, C., Abbeel, P., Levine, S. (2017, July). Model-agnostic meta-learning for fast adaptation of deep networks. In International Conference on Machine Learning (pp. 1126-1135). PMLR.
[26] Li, Z., Zhou, F., Chen, F., Li, H. (2017). Meta-sgd: Learning to learn quickly for few-shot learning. arXiv preprint arXiv:1707.09835.
[27] Schonfeld, E., Bahrami, S., Sinha, S., Darrell, T., Akata, Z. (2019). Generalized zero-and few-shot learning via aligned variational autoencoders. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 8247-8255).
[28] Li, A., Luo, T., Lu, Z., Xiang, T., Wang, L. (2019). Large-scale few-shot learning: Knowledge transfer with class hierarchy. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 7212-7220).
[29] Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., Le, Q. V. (2019). Autoaugment: Learning augmentation strategies from data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 113-123).
[30] Lim, S., Kim, I., Kim, T., Kim, C., Kim, S. (2019). Fast autoaugment. Advances in Neural Information Processing Systems, 32, 6665-6675.
[31] Hataya, R., Zdenek, J., Yoshizoe, K., Nakayama, H. (2020, August). Faster autoaugment: Learning augmentation strategies using backpropagation. In European Conference on Computer Vision (pp. 1-16). Springer, Cham.
[32] Ho, D., Liang, E., Chen, X., Stoica, I., Abbeel, P. (2019, May). Population based augmentation: Efficient learning of augmentation policy schedules. In International Conference on Machine Learning (pp. 2731-2741). PMLR.
[33] Jaderberg, M., Dalibard, Y., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., ... Kavukcuoglu, K. (2017). Population based training of neural networks. arXiv preprint arXiv:1711.09846.
[34] Kim, I., Kim, Y., Kim, S. (2020). Learning loss for test-time augmentation. arXiv preprint arXiv:2010.11422.
[35] Bergstra, J., Yamins, D., Cox, D. D. (2013) Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. PTPoc. of the 30th International Conference on Machine Learning (ICML 2013), June 2013, pp. I-115 to I-23.
[36] Passke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems, 32, 8026-8037.

WONHEE CHO received the B.S. degree in Applied Statistics from Chung-Ang University, Seoul, Korea, in 2019. She is currently a M.S. student at School of Computer Science and Engineering, Chung-Ang University, Seoul, Korea. Her research interests include deep learning, machine learning, few-shot learning, and computer vision.
EUNWOO KIM received the B.S. degree in Electrical and Electronics Engineering from Chung-Ang University, Seoul, Korea, in 2011, and the M.S. and Ph.D. degrees in Electrical Engineering and Computer Science from Seoul National University, Seoul, Korea in 2013 and 2017, respectively. He is currently an assistant professor with the School of Computer Science and Engineering, Chung-Ang University, Seoul, Korea. From 2017 to 2018, he was a postdoctoral researcher with the Department of Electrical Engineering and Computer Science, Seoul National University, Seoul, Korea. From 2018 to 2019, he was a postdoctoral researcher with the Department of Engineering Science, University of Oxford, Oxford, UK. His research interests include machine learning, deep learning, model optimization, robotics, and computer vision.