This CVPR workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

# Similar Class Style Augmentation for Efficient Cross-Domain Few-Shot Learning

Manogna Sreenivas and Soma Biswas

Indian Institute of Science, Bangalore, India
{manognas, somabiswas}@iisc.ac.in

## Abstract

*Cross-Domain Few-Shot Learning (CD-FSL) aims to recognize new classes from unseen domains, given limited training samples. Majority of the state-of-the-art approaches for this task introduce new task-specific additional parameters for adapting to the novel task, which involves changing the trained model architecture, in addition to increasing the number of model parameters. The first contribution of this work is to revisit the existing approaches like modifying the Batch Normalization affine parameters and the scale hyperparameter in cosine similarity based softmax loss for adapting the trained model to the new tasks, without changing the model architecture. Secondly, to aid the model learning with few examples per class, we propose to augment the data of each class with the styles of the semantically similar classes. Extensive evaluation on the challenging Meta-Dataset shows that this simple framework is very effective for the CD-FSL task. We also show that the Similar-class Style Augmentation module can be seamlessly integrated with existing approaches to further improve their performance, thus establishing the state-of-the-art in this challenging area.*

## 1. Introduction

Over the past decade, significant progress has been made in computer vision tasks such as image classification [6], object detection [17, 21], image segmentation [3, 8, 10] etc. However, while humans are capable of recognizing new objects by looking at just a handful of samples, the performance of state-of-the-art deep neural networks usually degrades significantly in such low-data regimes. In real-world scenarios, collecting and annotating large amounts of data can be difficult and expensive, so it is necessary for machines to learn from limited labeled data. Additionally, test data may come from a different distribution than training data. The objective of cross-domain few shot learning (CD-FSL) is to adapt a given feature extractor trained on large scale annotated data to a new task with few labeled samples per class from unseen domains.

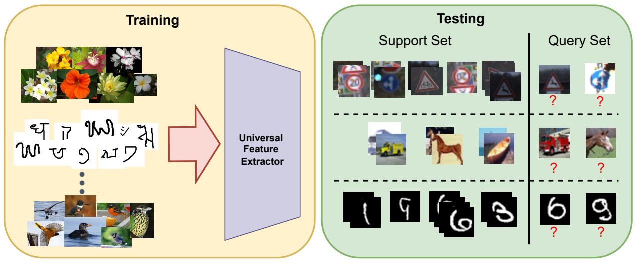


Figure 1. CD-FSL task: Training (left): Use labeled multi-domain data to learn universal feature extractor; Testing (right): N-way K-shot test tasks with unseen classes from unseen domains. Support set comprises of limited labeled samples, which is used to adapt the network and is then evaluated on the corresponding query set.

Most state-of-the-art CD-FSL frameworks address employ a two-step process, (i) Learning a universal feature extractor using training samples from different datasets; (ii) adapting the model to novel tasks from new domains using few labeled samples per class. Fig. 1 illustrates the training and testing stages of the CD-FSL task. After learning the universal feature extractor in the first step, the recent approaches [15, 16, 24] usually incorporate task-specific learnable modules to adapt to the new task. Though they give impressive performance for the CD-FSL task, these frameworks result in changing the model architecture, in addition to introducing additional model parameters. As this is not be desirable in many practical applications, here we propose a simple, yet effective framework for the CD-FSL task, without changing the model architecture or increasing the model parameters.

In this work, we propose a simple and effective framework for the CD-FSL task that does not require changing the model architecture or increasing the model parameters. Our approach builds on existing research in the field and focuses on modifying only a few parameters of the trained model to adapt it to unseen domains. Specifically, we propose to update the Batch Normalization (BN) affine parameters and analyze the importance of the scale parameter in the cosine similarity based softmax loss for the CD-FSL task. Additionally, we address the challenge of limited training data in novel tasks by proposing to augment the labelled data using styles from other semantically similar classes. The proposed framework is termed **S**imilar-class **S**tyle **A**ugmentation with appropriate **B**atch **N**orm and **S**cale parameters (**SSA-BNS**). We evaluate our framework on the Meta-Dataset benchmark and demonstrate that it achieves close to state-of-the-art performance while requiring fewer parameters than existing approaches that alter the model architecture. We also show that our proposed Similar-class Style Augmentation (SSA) module can be integrated with existing CD-FSL frameworks to further improve their performance.

## 2. Related work

Here, we discuss some of the related works in the areas of Few-Shot Learning (FSL), Cross-Domain Few-Shot learning (CD-FSL) and data augmentation.

**Few-Shot Learning (FSL):** FSL approaches [22, 26] often employ a Nearest Centroid Classifier (NCC), where the class centroids are obtained using the labelled support set samples. These centroids are then used to classify the samples from query set based on a distance metric like cosine similarity or Euclidean distance. Several prior works [9, 22, 26] formulate FSL as a learning to learn or meta-learning problem. In this perspective, the model is trained over a distribution of few-shot tasks sampled from the training data. Having the experience of performing well on the few-shot tasks, the model is now equipped to effectively learn from limited data of a novel test task. [22] uses Euclidean distance based NCC to meta-learn from training data. Matching Networks [26] classify query samples based on a distance-weighted linear combination of the support labels. In MAML [9], the authors propose to learn the model such that these parameters act as a good initialization for future tasks, so that it can adapt using just a few gradient updates with only limited data. Such learning strategies, having proven effective for in-domain FSL are also extended to the CD-FSL setting. But the additional challenge here is to actually learn image representations that are effective across domains.

**Cross-Domain Few-Shot Learning (CD-FSL):** We now discuss prior methods for CD-FSL task that involves learning multi-domain feature extractor(s). SUR [7] proposes to train multiple feature extractors, one for each domain. During test time, these feature extractors provide universal representations, over which a task-specific feature selection mechanism is employed along with NCC. URT [18] alternatively propose a meta-learning based selection mechanism to adapt the universal representations. As these are computationally complex, requiring a test sample to forward pass through multiple feature extractors, later works [15,24] propose effective ways to learn a single network using multi domain data. In FLUTE [24], the authors propose to learn domain specific FiLM [20] layers while keeping the rest of the backbone shared across all training domains. In URL [15], given the features from the universal feature extractor, a linear transformation is used which is task specific and learnable, to obtain the adapted features. In TSA [16], task-specific adapters are included in the backbone, which are then fine-tuned using support set samples. These adapters can be serial/residual blocks parameterized by a matrix or channel-wise transformation. These approaches are evaluated on the large-scale Meta-Dataset [25] benchmark, which establishes several baselines, extending the FSL methods for CD-FSL.

**Data Augmentation:** Data augmentation, in general, improves the generalization ability of a model, as it introduces more variations in the training data. It has been successfully used in several tasks including semi-supervised learning [23], unsupervised representation learning, contrastive learning [1,2,4,28]. Popular image based augmentations [5] include random crop, scale, flip, rotation, contrast enhancement, color distortions etc. Recently, several works propose to use Generative Adversarial Networks for data augmentation [29, 31]. Style transfer is another interesting data augmentation technique as it preserves the semantic content of the image. [12] propose to train a style transfer network (STN) based on Adaptive Instance Normalization (AdaIN) layer. AdaIN layer transfers the feature statistics (channel-wise mean and variance) of one image to another. A decoder is then trained to obtain the style transferred image. MixUp [11] is another popular data augmentation method where virtual samples and labels are created as convex combinations of two random training samples. This encourages linear behaviour for intermediate samples and is observed to improve model robustness. In CD-FSL, since the model has access to few support set samples to update the base network, it is challenging to employ GAN or STN based methods. Therefore, we emphasize the need for carefully designing augmentation methods for CD-FSL. In our work, we investigate the potential benefits of data augmentation in improving the adaptation performance for the CD-FSL task.

## 3. Problem definition

Firstly, we formulate the CD-FSL task and describe the notations used in this work. In general, a few shot learning task is described as N-way K-shot classification problem, where N represents the number of classes and K represents the number of samples per class available for training. We follow the Meta-Dataset benchmark where the tasks are sampled such that the number of classes N and samples per class K are varied. Each task $\mathcal{T} = (\mathcal{S}, \mathcal{Q})$ comprises of a support set $\mathcal{S}$ and a query set $\mathcal{Q}$. The model has to learn from a small labeled support set $\mathcal{S} =$

$\{(x_i^s, y_i^s), i = 1 \dots n_{\mathcal{S}}\}$ and is then evaluated on a query set $\mathcal{Q} = \{x_i^q, i = 1 \dots n_{\mathcal{Q}}\}$ consisting of the same classes as in the corresponding support set. Here, $x_i^s$ and $y_i^s$ denotes the support sample and its corresponding label, while $x_i^q$ refers to a query sample. $n_{\mathcal{S}}$ and $n_{\mathcal{Q}}$ denote the number of samples in the support and query set respectively. We denote the universal feature extractor as $F$ and the feature representation of a sample $x_i^s$ as $z_i^s = F(x_i^s)$.

In the cross-domain scenario, the training and test domains comprises of $M_{train}$ and $M_{test}$ datasets respectively.

$$\mathcal{D}^{train} = \mathcal{D}_1 \cup \mathcal{D}_2 \dots \mathcal{D}_{M_{train}}$$

$$\mathcal{D}^{test} = \mathcal{D}_1^{'} \cup \mathcal{D}_2^{'} \dots \mathcal{D}_{M_{test}}^{'}$$

The objective is to use labeled data from $\mathcal{D}^{train}$ to learn a feature extractor $F$. This feature extractor is then adapted for novel few-shot tasks $\mathcal{T}$ sampled from $\mathcal{D}^{test}$, comprising of previously unseen classes and also from unseen domains. Now, we explain how recent state-of-the-art approaches address this task, which also serves as the motivation for this work.

**Few shot task adaptation:** Most of the SOTA approaches for the CD-FSL task involves two stages, which are explained below.

**1) Learning a Universal Feature Extractor:** Firstly, it is important to learn an effective feature extractor from the diverse set of training datasets ($\mathcal{D}_{train}$) with different classes and possibly different domains. A simple way to learn the feature extractor is to jointly use the labelled data from all training domains. However, the domain shifts among the various datasets pose challenges in such joint training and hence requires careful learning mechanisms and different approaches have been proposed for this task. Since the training of the feature extractor plays an important role in the final performance, for fair comparison, we use the same universal feature extractor as used in [15, 16] and primarily focus on the next stage, i.e task-specific model adaptation.

**2) Task-specific model adaptation:** Given the universal feature extractor, the goal is to adapt it to a task from an unseen domain with unseen classes using just a few labelled samples per class. To successfully handle this challenging task, most of the recent methods like FLUTE [24], URL [15], TSA [16] employ task specific additional parameters, and propose to fine-tune these parameters using the support set samples and a nearest centroid classifier. Though these methods obtain impressive performance on the challenging Meta-Dataset [25], they involve either increasing the number of model parameters [15,16,24] and/or changing the originally trained architecture [15, 16]. Since this may not be desirable for many practical applications, in this work, we revisit adapting the BatchNorm (BN) affine parameters, scale parameter of the cosine similarity metric and also propose a new augmentation technique termed SSA to address CD-FSL task, while not adding any parameters or changing the model architecture.

## 4. Proposed SSA-BNS Framework

With this background, we now describe the proposed framework (Fig. 2) for handling the CD-FSL task. As mentioned earlier, we use the trained model as in earlier works [15, 16] for fair comparison. We also make use of non-parametric classifiers which have been extensively used in few-shot learning methods as they avoid overfitting, which often happens while training a parameterized classifier using very less number of samples. First, we describe in brief the training of the feature extractor and the NCC for completion.

**Universal Feature Extractor:** We use the universal feature extractor proposed in URL [15], which is also used in TSA [16]. They learn the network in two stages: 1) Firstly, $M_{train}$ single domain feature extractors are trained, one for each of the training domains, using cross entropy loss. 2) For efficient task adaptation in the future, knowledge distillation is performed from the single domain feature extractors to a universal feature extractor, all of them sharing the same architecture. The domain specific features are mapped to a common space using domain specific adaptors, by minimizing Kullback-Liebler (KL) divergence between the model predictions of domain-specific features and the universal features. Alongside, the distance between these two sets of features are also minimized through Centered Kernel Alignment [13].

**Nearest Centroid Classifier (NCC):** Given the universal feature extractor $F$ and a few-shot task $\mathcal{T} = (\mathcal{S}, \mathcal{Q})$ sampled from $\mathcal{D}^{test}$, class centroids are obtained by averaging the feature representations $z_i^s = F(x_i^s)$ of the support set samples $x_i^s \in \mathcal{S}$, for each class $k = 1 \dots C$ present in this task as:

$$\mathbf{c}_k = \frac{1}{|\mathcal{S}_k|} \sum_{x_i^s \in \mathcal{S}_k} z_i^s; \quad \mathcal{S}_k = \{x_i^s | y_i^s = k\} \tag{1}$$

The NCC loss based on cosine similarity, with scaling factor $\eta$ is obtained as

$$\mathcal{L}_{NCC}(z_i^s, y_i^s; \eta) = -\log p(y = y_i^s \mid z_i^s; \eta) \tag{2}$$

$$\text{where} \quad p(y = y_i^s \mid z_i^s; \eta) = \frac{e^{\eta \cos \theta_{i, y_i^s}}}{\sum_{j=1}^{C} e^{\eta \cos \theta_{i,j}}}$$

$$\cos \theta_{i, y_i^s} = \frac{\mathbf{c}_{y_i^s}^T z_i^s}{\|z_i^s\| \|\mathbf{c}_{y_i^s}\|}.$$

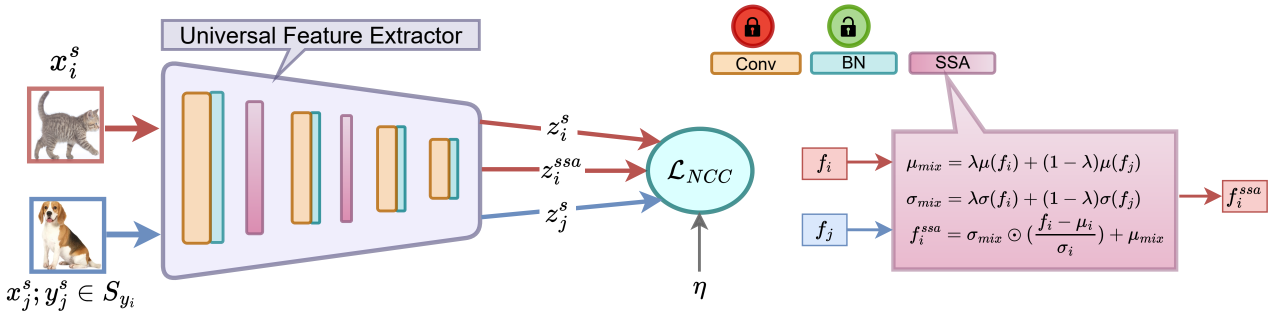

Universal Feature Extractor
$x_i^s$
$x_j^s; y_j^s \in S_{y_i}$
$z_i^s$
$z_i^{ssa}$
$z_j^s$
$\mathcal{L}_{NCC}$
$\eta$
Conv
BN
SSA
$f_i$
$f_j$
$\mu_{mix} = \lambda\mu(f_i) + (1-\lambda)\mu(f_j)$
$\sigma_{mix} = \lambda\sigma(f_i) + (1-\lambda)\sigma(f_j)$
$f_i^{ssa} = \sigma_{mix} \odot (\frac{f_i - \mu_i}{\sigma_i}) + \mu_{mix}$
$f_i^{ssa}$

Figure 2. Proposed SSA-BNS framework: A support set instance $x_i^s$ is augmented using a similar class sample $x_j^s, y_j^s \in S_{y_i^s}$ through SSA modules inserted in the universal feature extractor. Few shot task adaptation is done minimizing the $L_{NCC}$ loss over the actual features along with SSA augmented features.

Now, we describe each of the components of the proposed framework, namely updating the BN affine and scale parameters and the similar class style augmentation module.

## 4.1. Revisiting BatchNorm Adaptation for CD-FSL

In this work, we investigate how well we can adapt the trained model for a test task without modifying its architecture, or adding additional parameters. Specifically, we only adapt the BN affine parameters present in the network.
**Batch Normalization:** Let $f^l$ denote the feature activations at layer $l$. The batch normalized feature activations $f^l_{BN}$ are obtained as

$$f^l_{BN} = \gamma^l \hat{f}^l + \beta^l; \quad \text{where} \quad \hat{f}^l = \frac{f^l - \mu^l}{\sqrt{(\sigma^l)^2 + \epsilon}} \tag{3}$$

Here, $\mu^l$ and $\sigma^l$ are the running mean and standard deviation estimated from the training data.
Recently, adaptation of BN parameters $(\gamma^l, \beta^l)$ have been successfully used for many applications. In TENT [27], the BN scale and shift parameters are adapted to minimize the test entropy and has been very effective in mitigating performance degradation due to distribution shifts for Test-Time Adaptation (TTA) task. CD-FSL task is very different from TTA, as the classes in test tasks are unseen and there are only limited samples to learn from.

In this work, we investigate the effectiveness of adapting BN parameters for CD-FSL task. The combined BN scale and shift parameters, which we denote as $\{\gamma, \beta\}$, are optimized to minimize the NCC loss over the support set as

$$\min_{\gamma,\beta} \frac{1}{n_S} \sum_{(x_i^s, y_i^s) \in \mathcal{S}} \mathcal{L}_{NCC}(z_i^s, y_i^s; \eta) \quad \text{where} \quad z_i^s = F(x_i^s) \tag{4}$$

We now study the impact of cosine similarity scale factor $\eta$ on the CD-FSL performance.

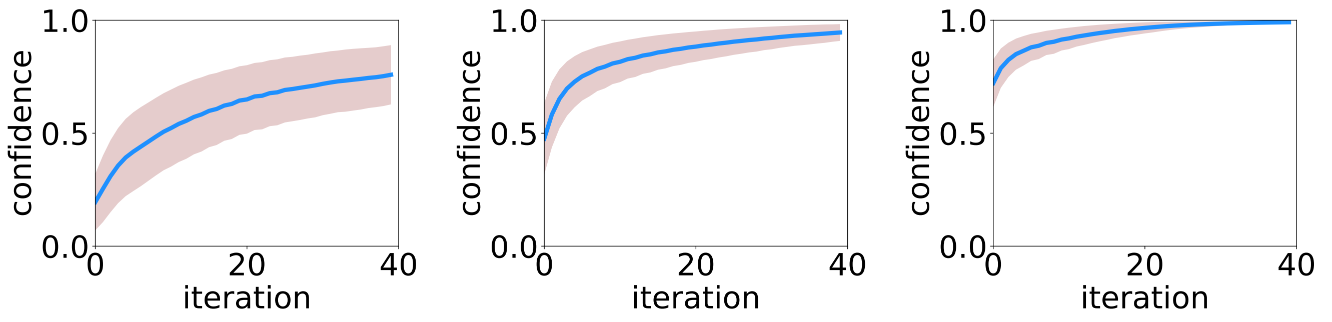
## 4.2. Effect of Scale Factor on CD-FSL

Following several prior works in the few-shot learning regime [7, 22, 24], in this work, we use a Nearest Centroid Classifier based on cosine similarity metric. Using this classifier, the posterior probabilities over $C$ classes are computed using softmax as

$$p(y = k \mid z_i^s; \eta) = \frac{e^{\eta \cos\theta_{i,k}}}{\sum_{j=1}^{C} e^{\eta \cos\theta_{i,j}}}; \quad k = \{1, \ldots, C\} \tag{5}$$

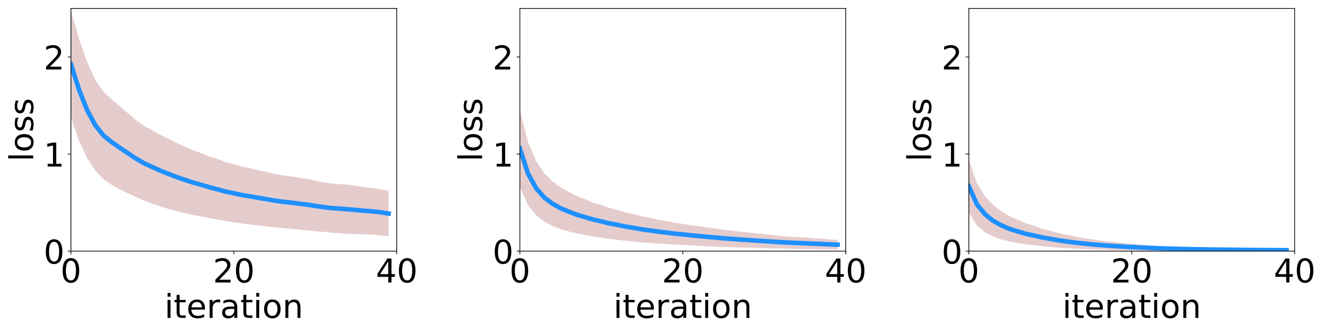
where $\eta$ is the scale hyper-parameter. The softmax scores being a function of the logits, is dependent on the range of values taken by the logits. The cosine similarity values, being the logits here, are in the range [-1,1]. Thus, it is a standard practice to scale these logits with a factor $\eta$ in order to expand the input range for the softmax function, thereby assigning reasonable probability scores. This scale factor although plays a significant role in the training process of cosine similarity based metric learning, has only been scarcely studied, especially in the few shot regime. AdaCos [30] proposed an approach of identifying an optimal scale parameter based on the number of classes, for a large scale face recognition task. However, the CD-FSL task significantly differs from that addressed in [30], as the number of samples available per class is very low here. In addition, each test task can have varying number of classes and can come from an unseen domain, which can be very different from the training domains. Thus, the analysis in [30], although insightful, may not be directly applicable to the CD-FSL task.

In the previous CD-FSL works URL [15] and TSA [16], the scale parameter $\eta$ was fixed to 10. In this work, we investigate the impact of this scale factor on the CD-FSL task, where we only update the BN parameters using the labelled support set. Here, since the target domain can be very different from the training domains with few labeled examples per class, we expect the model to be less confident in classifying the training data, as compared to other



(a) $scale = 10$ (b) $scale = 25$ (c) $scale = 50$

Figure 3. The mean and standard deviation of confidence scores of the predicted class over 600 tasks from Aircraft dataset for 40 training iterations.



(a) $scale = 10$ (b) $scale = 25$ (c) $scale = 50$

Figure 4. The mean and standard deviation of loss curves over 600 tasks for 40 training iterations from Aircraft dataset.

applications where the test domain is similar to the training domains or there are more training examples available. This implies that for majority of the tasks and training examples, the cosine similarities will be lower, resulting in low probability values. Since the scale factor controls the range of probability values that the cosine similarity gets mapped to, we expect that a higher scale factor will be beneficial for the CD-FSL task, so that the model does not incur any loss even when a training sample is correctly classified.

For the CD-FSL task, using the universal feature extractor, we perform experiments by varying scale parameter $\eta$ as 10, 25 and 50. We use one of the training domains *Aircraft*, to analyse the impact of scale factor on the training process. To understand the confidence scores of the predicted class of support set samples, we plot the confidence scores averaged over all the support samples and the tasks, as training progresses in Fig. 3. We observe from Fig. 3a that for a scale of 10, the model is only able to assign confidence scores in the range of 0.6-0.8 even towards the end of training, although the support set samples get correctly classified. Using a scale of 25, the model learns to correctly classify samples while also gradually assigning high confidence. On the other hand, using a scale of 50, the model predictions attain confidence>0.8 for most of the samples in only about 10 iterations. Correspondingly, the loss converges in <25 iterations resulting in overfitting on the support set as shown in Fig. 4c, resulting in degraded performance when compared to using a scale of 25.

As we later show in Table 2, empirically also, we observe that using a higher scale ($\eta = 25$) improves the CD-FSL performance significantly as compared to setting it to 10, which was the default in URL and TSA. This verifies our hypothesis, that because of the challenging nature of the CD-FSL task, a higher scale factor is desirable. Needless to mention, that if the scale factor is increased to a very high value, the model assigns a high probability score irrespective of the cosine similarity value, which in turn reduces the discriminability between samples and can hurt the training process.

### 4.3. Similar Class Style Augmentations

Our objective is to learn to recognize new classes in unseen domains using limited training data. Here, we propose to augment the support set in the feature space using the styles of similar class samples. For example, suppose we have a *cat* training class which has examples of *sitting cat, sleeping cat*, we can augment it with the styles of a neighbouring class, say *dog*, which might have examples of *standing dog, running dog*. It may not be meaningful to augment it with the styles of semantically dissimilar classes, say *bird*, which may have examples of *flying birds*. The early layer feature statistics are representative of the domains/styles of the input sample [12, 32]. In this section, we simplify the notation, by dropping the superscript $s$ referring to support set samples for clarity.

Given a sample $(x_i, y_i) \in \mathcal{S}$, we propose to mix its feature statistics (representative of style) with that of another sample $x_j$ to synthesize a style augmented feature of class $y_i$. Although, this a label preserving transform, as explained earlier, arbitrarily picking samples could adversely affect the training process. Given the small support set, in order to effectively style augment while also avoiding irrelevant feature perturbations, we propose to condition the mixing of styles to only relevant classes based on the similarity of the class representations. We now formulate the SSA (**S**imilar-class **S**tyle **A**ugmentation) module.

Firstly, we obtain the pairwise class similarities based on their centroids. The cosine similarity of class $i$ with that of class $j$ can be obtained from their centroids $\mathbf{c}_i, \mathbf{c}_j$ as

$$sim(\mathbf{c}_i, \mathbf{c}_j) = \frac{\mathbf{c}_i^T \mathbf{c}_j}{\|\mathbf{c}_i\| \|\mathbf{c}_j\|} \tag{6}$$

For a class $k$, the set of similar classes is determined as:

$$S_k = \{t | sim(\mathbf{c}_t, \mathbf{c}_k) > \tau; t = 1, .., C\} \tag{7}$$

where $\tau$ is a pre-set threshold.

Denoting $f_i$ as the intermediate feature at a layer $l$, of support sample $x_i$ belonging to class $y_i$, we randomly select another sample $x_j$ belonging to one of its similar classes $S_{y_i}$, i.e., $y_j \in S_{y_i}$. The style of $x_j$ is used to perturb the style of $x_i$ to get the style augmented sample as follows:

$$\begin{aligned}
\mu_{\text{ssa}}(f_i; f_j, y_j \in S_{y_i}) &= \lambda \mu(f_i) + (1 - \lambda) \mu(f_j) \\
\sigma_{\text{ssa}}(f_i; f_j, y_j \in S_{y_i}) &= \lambda \sigma(f_i) + (1 - \lambda) \sigma(f_j) \\
f_i^{ssa} = \sigma_{\text{ssa}}(f_i; f_j, y_j \in S_{y_i}) &\odot \frac{f_i - \mu(f_i)}{\sigma(f_i)} + \mu_{\text{ssa}}(f_i; f_j)
\end{aligned} \tag{8}$$

Here, $\mu(f_i), \mu(f_j)$ and $\sigma(f_i), \sigma(f_j)$ are the channel-wise mean and standard deviation of the features $f_i$ and $f_j$ respectively. Let $z_i, z_j$ and $z_i^{ssa}$ denote the output feature vectors (from last layer) obtained on passing $f_i, f_j$ and $f_i^{ssa}$ respectively through the rest of the network. The SSA augmented feature $z_i^{ssa}$ is obtained by mixing the styles of $x_i$ and $x_j$. However the content of $x_i$ is preserved in the feature $z_i^{ssa}$ and hence belongs to the class $y_i$.

**SSA-BNS:** To summarize, we minimize the NCC loss over the support set features and their augmentations obtained through SSA as

$$\min_{\gamma, \beta} \frac{1}{2n_{\mathcal{S}}} \sum_{(x_i^s, y_i^s)} L_{NCC}(z_i^s, y_i^s; \eta) + L_{NCC}(z_i^{ssa}, y_i^s; \eta) \tag{9}$$

For evaluation on query set $\mathcal{Q}$, cosine similarity metric is used to assign the query sample $x_i^q$ to the nearest centroid as:

$$\hat{y}_i^q = {}_{k} \cos \theta_{i,k}; \quad \cos \theta_{i,k} = \frac{\mathbf{c}_k^T z_i^q}{\|z_i^q\| \|\mathbf{c}_k\|} \tag{10}$$

where $z_i^q$ is the feature of the query sample $x_i^q$.

Though this work is inspired from MixStyle [32], there are significant differences between the two approaches as discussed below:
1) Mixstyle mixes styles of samples from different source domains during training, primarily to interpolate the domain information. In our work, during training for the new tasks, the source domain data is not available. All the examples that are used for augmentation belong to the same unseen domain.
(2) In MixStyle, the two samples can come from any class. We observe that class-agnostic mixing is not meaningful for this application as we are primarily mixing the styles, and it also hurts the performance as seen empirically. Thus, the style mixing takes place only between semantically similar classes to generate realistic augmentations, which aid the training process.

## 5. Experimental Evaluation

Here, we describe the datasets used, implementation details and the experimental results.

**Meta-Dataset Benchmark:** Meta-Dataset [25] is a recent benchmark proposed for the CD-FSL task. This benchmark comprises of 10 datasets of which eight act as training domains and the other two as test domains. The training domains comprises of ImageNet, Omniglot, Aircraft, Birds, Textures, Quick draw, Fungi and VGG Flower datasets. Traffic Signs and MSCOCO act as test domains. Additionally, following prior works [15, 16, 24], we also include MNIST, CIFAR-10 and CIFAR-100 as test domains. The few shot tasks are sampled with varying number of classes $N$, with $N$ varying from 5 to the maximum number of classes available in the dataset. The number of samples per class $K$, although varying, is capped at 100 samples and also the total support set size is limited to 500 samples in all.

**Implementation details:** We use a ResNet-18 backbone trained on the eight training domains following the Meta-Dataset protocol [25]. This universal feature extractor is the same as that used in [15, 16], which is trained by distilling knowledge from eight individually trained feature extractors.

For few shot task adaptation, we only finetune the BN affine parameters $\{\gamma, \beta\}$ on the support set of a given task

| Dataset | SUR | URT | FLUTE | tri-M | URL* | TSA* | SSA-BNS | TSA*+SSA |
|---|---|---|---|---|---|---|---|---|
| Imagenet | $56.2 \pm 1.0$ | $56.8 \pm 1.1$ | $58.6 \pm 1.0$ | $51.8 \pm 1.1$ | $58.8 \pm 1.1$ | $59.5 \pm 1.0$ | $56.6 \pm 1.0$ | $58.9 \pm 1.1$ |
| Omniglot | $94.1 \pm 0.4$ | $94.2 \pm 0.4$ | $92.0 \pm 0.6$ | $93.2 \pm 0.5$ | $94.5 \pm 0.4$ | $94.9 \pm 0.4$ | $95.2 \pm 0.5$ | $\mathbf{95.6 \pm 0.4}$ |
| Aircraft | $85.5 \pm 0.5$ | $85.8 \pm 0.5$ | $82.8 \pm 0.7$ | $87.2 \pm 0.5$ | $89.4 \pm 0.4$ | $89.9 \pm 0.4$ | $89.6 \pm 0.4$ | $\mathbf{90.0 \pm 0.5}$ |
| Birds | $71.0 \pm 1.0$ | $76.2 \pm 0.8$ | $75.3 \pm 0.8$ | $79.2 \pm 0.8$ | $80.7 \pm 0.8$ | $81.1 \pm 0.8$ | $81.8 \pm 0.8$ | $\mathbf{82.2 \pm 0.7}$ |
| Textures | $71.0 \pm 0.8$ | $71.6 \pm 0.7$ | $71.2 \pm 0.8$ | $68.8 \pm 0.8$ | $77.2 \pm 0.7$ | $77.5 \pm 0.7$ | $76.4 \pm 0.7$ | $\mathbf{77.6 \pm 0.7}$ |
| Quick draw | $81.8 \pm 0.6$ | $82.4 \pm 0.6$ | $77.3 \pm 0.7$ | $79.5 \pm 0.7$ | $82.5 \pm 0.6$ | $81.7 \pm 0.6$ | $\mathbf{82.8 \pm 0.6}$ | $82.7 \pm 0.7$ |
| Fungi | $64.3 \pm 0.9$ | $64.0 \pm 1.0$ | $48.5 \pm 1.0$ | $58.1 \pm 1.1$ | $\mathbf{68.1 \pm 0.9}$ | $66.3 \pm 0.8$ | $66.7 \pm 0.8$ | $66.6 \pm 0.8$ |
| VGG Flower | $82.9 \pm 0.8$ | $87.9 \pm 0.6$ | $90.5 \pm 0.5$ | $91.6 \pm 0.6$ | $92.0 \pm 0.5$ | $92.2 \pm 0.5$ | $92.8 \pm 0.6$ | $\mathbf{93.0 \pm 0.5}$ |
| Traffic Sign | $51.0 \pm 1.1$ | $48.2 \pm 1.1$ | $63.0 \pm 1.0$ | $58.4 \pm 1.1$ | $63.3 \pm 1.1$ | $82.8 \pm 1.0$ | $77.9 \pm 1.1$ | $\mathbf{84.9 \pm 1.1}$ |
| MSCOCO | $52.0 \pm 1.1$ | $51.5 \pm 1.1$ | $52.8 \pm 1.1$ | $50.0 \pm 1.0$ | $57.3 \pm 1.0$ | $57.6 \pm 1.0$ | $56.1 \pm 0.9$ | $\mathbf{58.1 \pm 1.0}$ |
| MNIST | $94.3 \pm 0.4$ | $90.6 \pm 0.5$ | $96.2 \pm 0.3$ | $95.6 \pm 0.5$ | $94.7 \pm 0.4$ | $96.7 \pm 0.4$ | $98.3 \pm 0.5$ | $\mathbf{98.5 \pm 0.4}$ |
| CIFAR-10 | $66.5 \pm 0.9$ | $67.0 \pm 0.8$ | $75.4 \pm 0.8$ | $78.6 \pm 0.7$ | $74.2 \pm 0.8$ | $82.9 \pm 0.7$ | $79.4 \pm 0.7$ | $\mathbf{82.9 \pm 0.7}$ |
| CIFAR-100 | $56.9 \pm 1.1$ | $57.3 \pm 1.0$ | $62.0 \pm 1.0$ | $67.1 \pm 1.0$ | $63.5 \pm 1.0$ | $70.4 \pm 0.9$ | $69.0 \pm 0.9$ | $\mathbf{70.8 \pm 0.9}$ |
| Average seen | 75.9 | 77.4 | 74.5 | 76.2 | 80.4 | 80.4 | 80.2 | **80.8** |
| Average unseen | 64.1 | 62.9 | 69.9 | 69.9 | 70.6 | 78.1 | 76.1 | **79.0** |
| Average all | 71.4 | 71.8 | 72.7 | 73.8 | 76.6 | 79.5 | 78.7 | **80.1** |

Table 1. Average accuracy over 600 tasks are reported for all the compared approaches for CD-FSL task. URL, TSA, SSA+BNS, TSA+BNS uses the same universal feature extractor. * indicates that these approaches uses additional parameters as compared to the feature extractor (Table 3).

using the NCC. As in TSA [16], we also use Adadelta optimizer with learning rate of 0.001 and train for 40 iterations. We set $\lambda$ to 0.5 in eqn.(8) to perform SSA, scale parameter $\eta$ to 25 and the threshold $\tau$ to 0.7 in all the experiments. MixStyle layers are inserted after the first and second ResNet blocks.

**Experimental Results:** Table 1 reports the average accuracy and 95% confidence interval over 600 tasks obtained using the proposed framework, alongside comparisons with the state-of-the-art approaches. The first group of results correspond to the seen domains and the second group to unseen domains. The results of the other works are directly taken from [16]. For the proposed work, we reports two sets of results. First, we report the results of our SSA-BNS framework, which does not include any change in architecture or increase in number of parameters compared to the originally trained model using the source datasets. We observe that for the unseen domains, the proposed SSA-BNS significantly outperforms FLUTE [24], tri-M [19] and URL [15], and is second only to TSA [16]. For the seen domains, it performs comparably to all the other approaches. On an average, SSA-BNS achieves an accuracy of 78.7% as compared to 76.6% obtained by URL. It is only second to [16], which obtains an average accuracy of 79.5%. Note that all the other approaches uses significantly more parameters and also involves change in the trained model architecture as shown in Table 3. In addition, we also report the results of integrating the proposed SSA module into the state-of-the-art [16] approach. We

| SSA | BNS($\eta$) | Seen domains | | Unseen domains | |
|---|---|---|---|---|---|
| | | Aircraft | Fungi | CIFAR-100 | MSCOCO |
| ✗ | ✗ | 87.0 | 65.6 | 59.9 | 53.1 |
| ✗ | 10 | 89.1 | 66.0 | 66.9 | 54.5 |
| ✗ | 25 | 89.5 | 66.4 | 68.4 | 55.7 |
| ✗ | 50 | 89.5 | 66.2 | 67.7 | 55.4 |
| ✓ | 25 | **89.6** | **66.7** | **69.0** | **56.1** |

Table 2. CD-FSL performance on two seen and unseen domains (averaged over 600 tasks). Effect of BN adaptation, scale factor $\eta$ and the SSA module.

observe that this simple, yet effective augmentation (with no additional parameters) can be used to further improve the performance of the SOTA approaches for both seen and unseen domains.

## 6. Additional Analysis

Here, we perform additional analysis to better understand the proposed framework and the usefulness for each of its components.

**Ablation Study:** We study the importance of each component for two seen and unseen domains and report the results in Table 2. Firstly, we do not adapt the network and simply use a NCC to classify the query samples, which corresponds to the first row in Table 2. Then we study the role of BN adaptation for the CD-FSL task. While earlier works like URL [15], TSA [16] use a default scale factor of 10, we experiment with other values (25 and

| Method | #Additional parameters | # Trainable parameters |
|---|---|---|
| URL | 262144 | 262144 |
| TSA | 1482752 | 1482752 |
| TSA+SSA | 1482752 | 1482752 |
| SSA-BNS | None | 9600 |

Table 3. Comparison of computational complexity with state-of-the-art CD-FSL methods

| Type | Seen domains | | Unseen domains | |
|---|---|---|---|---|
| | Aircraft | Fungi | CIFAR-100 | MSCOCO |
| RandAugment [5] | 88.8 | 65.2 | 66.9 | 55.2 |
| MixUp [11] | 88.4 | 66.3 | 67.9 | 54.6 |
| Feature MixUp [14] | 88.9 | 66.3 | 68.3 | 55.3 |
| Random MixStyle [32] | 89.6 | 66.2 | 68.2 | 55.3 |
| **SSA (Proposed)** | **89.6** | **66.7** | **69.0** | **56.1** |

Table 4. Comparison of the proposed BNS module with different augmentation techniques.

50) to study its impact. We observe that using a scale of 25 consistently performs better as compared to using a smaller scale factor like 10. Although using scale of 50 is preferable over 10, the results degrade when compared to using a scale of 25. This happens due to the assignment of high confidence for correctly classified support samples with low cosine similarity values as discussed in Section 4.2. We also observe that the SSA module consistently improves the performance for both the seen and unseen domains. The effect of SSA module can also be seen by comparing the accuracies obtained by TSA and TSA+SSA in Table 1. This analysis shows that each of the proposed components contribute significantly towards improving the performance.

**Trainable Parameters:** We compare the complexity of the proposed SSA-BNS framework with that of the previous SOTA methods, URL [15] and TSA [16]. Specifically, we report (i) the number of additional parameters incorporated in the universal feature extractor for few shot task adaptation and (ii) the number of trainable parameters in Table 3.

Given the universal feature extractor, the SSA-BNS framework does not require any change in the model architecture, as it only leverages the existing BN layers in the network for adaptation. Here, only the BN affine parameters (9600 in the ResNet-18 architecture) are trainable. We observe that SSA-BNS framework is very efficient, and significantly outperforms URL having 262,144 additional parameters. It is only second to TSA which uses 154 times more parameters. TSA+SSA, which outperforms all the existing approaches has the same number of parameters as TSA, since the proposed SSA module does not introduce any additional parameters.

**Performance with Different Augmentations:** Although it is common to use data augmentations for training deep models, it is not always clear which type of augmentation will help in which application, especially in the low-data regime. Here, we perform extensive experiments using a variety of image and feature space augmentations for the CD-FSL task. Specifically, we investigate the following successful and popularly used data augmentation techniques for our task: 1) RandAugment [5] sequentially applies image transforms like autocontrast, polarize, shear, posterize, equalize etc. to the input image. 2) In MixUp [11], image augmentations and their labels are obtained as convex combinations of image samples and their labels. For e.g., given two samples $x_i$ and $x_j$, mixup samples is created as $x_{aug} = \lambda x_i + (1-\lambda)x_j$ with label $y_{aug} = \lambda y_i + (1-\lambda)y_j$. 3) Inspired from i-Mix [14], we perform Feature MixUp where a feature is perturbed by slightly shifting it towards another sample. The augmented feature of a sample $x_i$ using $x_j$ is obtained as $f_i^{aug} = \lambda f_i + (1-\lambda)f_i$ with $\lambda$ set to 0.9. Such feature mixing layers are inserted after the first two ResNet blocks. 4) Random MixStyle [32]: Here, the styles are augmented as described in Section 4.3, but the samples whose styles are mixed can belong to any class, not necessarily similar classes. 5) SSA (Proposed): In this work, we use similar class style augmentations as described in Section 4.3. We observe that the proposed SSA module outperforms all the commonly used augmentation techniques.

## 7. Conclusion

In this work, we address the challenging task of Cross-Domain Few-Shot Learning (CD-FSL), where a model learnt using diverse source domains has to be adapted to new tasks (with different classes from different domains) with only a few training examples per class. Since changing the trained model architecture and increasing the number of trainable parameters is often infeasible/not desirable, we propose a novel framework, termed SSA-BNS, which modifies the BN affine parameters and the scale parameter of the cosine similarity based softmax loss, without modifying the feature extractor or adding any parameters to the source trained model, along with a similar-class augmentation module for this task. Extensive experiments on several datasets show that the proposed framework outperforms several recent works with much lesser parameters and modifications. We also show that the proposed SSA module can be integrated with the current state-of-the-art approach [16] to further improve its performance, giving the state-of-the-art performance for the challenging CD-FSL task.

## References

[1] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. *NeurIPS*, 32, 2019. 2

[2] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *NeurIPS*, 2020. 2

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 40(4):834–848, 2017. 1

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*. PMLR, 2020. 2

[5] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPRW*, 2020. 2, 8

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 1

[7] Nikita Dvornik, Cordelia Schmid, and Julien Mairal. Selecting relevant features from a multi-domain representation for few-shot classification. In *ECCV*, pages 769–786. Springer, 2020. 2, 4

[8] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 1

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*. PMLR, 2017. 2

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, pages 2961–2969, 2017. 1

[11] Yann N. Dauphin David Lopez-Paz Hongyi Zhang, Moustapha Cisse. mixup: Beyond empirical risk minimization. *ICLR*, 2018. 2, 8

[12] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 2, 6

[13] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*. PMLR. 3

[14] Kibok Lee, Yian Zhu, Kihyuk Sohn, Chun-Liang Li, Jinwoo Shin, and Honglak Lee. i-mix: A domain-agnostic strategy for contrastive representation learning. In *ICLR*, 2021. 8

[15] Wei-Hong Li, Xialei Liu, and Hakan Bilen. Universal representation learning from multiple domains for few-shot classification. In *ICCV*, pages 9526–9535, October 2021. 1, 2, 3, 4, 6, 7, 8

[16] Wei-Hong Li, Xialei Liu, and Hakan Bilen. Cross-domain few-shot learning with task-specific adapters. In *CVPR*, June 2022. 1, 2, 3, 4, 6, 7, 8

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *ECCV*, pages 740–755, 2014. 1

[18] Lu Liu, Will Hamilton, Guodong Long, Jing Jiang, and Hugo Larochelle, editors. *A Universal Representation Transformer Layer for Few-Shot Image Classification*, 2021. 2

[19] Yanbin Liu, Juho Lee, Linchao Zhu, Ling Chen, Humphrey Shi, and Yi Yang. A multi-mode modulator for multi-domain few-shot classification. In *ICCV*, 2021. 7

[20] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018. 2

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 28, 2015. 1

[22] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *NeurIPS*, 30, 2017. 2, 4

[23] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *NeurIPS*, 2020. 2

[24] Eleni Triantafillou, Hugo Larochelle, Richard Zemel, and Vincent Dumoulin. Learning a universal template for few-shot dataset generalization. In *ICML*, pages 10424–10433. PMLR, 2021. 1, 2, 3, 4, 6, 7

[25] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Jordan Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *ICLR*, 2020. 2, 3, 6

[26] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *NeurIPS*, 29, 2016. 2

[27] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021. 4

[28] Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised data augmentation for consistency training. *NeurIPS*, 2020. 2

[29] Shin'ya Yamaguchi, Sekitoshi Kanai, and Takeharu Eda. Effective data augmentation with multi-domain learning gans. In *AAAI*, 2020. 2

[30] Xiao Zhang, Rui Zhao, Yu Qiao, Xiaogang Wang, and Hongsheng Li. Adacos: Adaptively scaling cosine logits for effectively learning deep face representations. In *CVPR*, June 2019. 4

[31] Miaoyun Zhao, Yulai Cong, and Lawrence Carin. On leveraging pretrained gans for generation with limited data. In *ICML*. PMLR, 2020. 2

[32] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. In *ICLR*, 2021. 6, 8