Few-shot Classification with SoftTriple Loss

Yuzhe You, Zhen Gong[1]

Abstract

Few-shot classification is defined as the problem of learning a classifier to adapt to novel classes during training with only limited labeled examples. In this study, we investigate the feasibility of improving baseline performance by incorporating the newly proposed SoftTriple loss into the model. We denote our model as *BaselineST*. We observe that selecting certain values for the number of centroids and specific combinations of weights for the hybrid loss function achieves minor improvement on the state-of-the-art performance of the Baseline++ model in certain datasets.

1. Introduction

With wider and more robust models being proposed for machine learning, deep learning models have achieved state-of-the-art performance on visual recognition tasks such as image classification (Chen et al., 2019; Shi et al., 2021). Nonetheless, the performance of these models relies heavily on the size of the annotated training instances and the diversity of visual variations among the images, and producing a large-scaled labeled dataset is often time-consuming and requires considerable human cost (Chen et al., 2019; Shi et al., 2021). In addition, the scarcity of certain classes (e.g. rare species and uncommon diseases) further exacerbates the in-feasibility of creating a sufficiently large dataset to effectively improve models' classification performance (Chen et al., 2019).

Few-shot classification, known as the problem of learning to adapt to unseen tasks during training with only limited labeled examples, has gained considerable attention in recent years as an attempt to address the unavailability of large-scaled training data (Shi et al., 2021). While a variety of methods have been proposed to increase model accuracy in few-shot classification, a baseline model with its linear classifier substituted with cosine similarity (denoted as *Baseline++*) is known for achieving strong performance with a prediction accuracy similar to the state-of-art meta-learning methods (Chen et al., 2019). The proposed Baseline++ model utilizes cross-entropy loss with the SoftMax function as its activation function, however a novel loss function known as the SoftTriple loss is recently proposed for distance metric learning with the goal to improve SoftMax loss via introduction of multiple centroids for each class (Qian et al., 2019). In this study, we aim to investigate the feasibility of enhancing the Baseline++ model with SoftTriple loss as an attempt to further improve few-shot classification accuracy. We denote our new model as *BaselineST*.

2. Related Work

Few-shot Classification

Few-shot Classification is a field of machine learning that has been extensively studied in recent years. A variety of methods have been proposed to improve model performance on novel classes with few labeled training samples, with two of the dominant methods being optimization-based and metric-based meta-learning (Shi et al., 2021). In (Finn et al., 2017), Finn et al proposed a meta-learning algorithm that is model-agnostic and therefore compatible with any model trained with gradient descent and thus applicable to a variety of learning problems. In (Sun et al., 2019), Sun et al proposed a novel few-shot learning method denoted as meta-transfer learning (MTL), which learns to adapt a deep NN for few-shot learning tasks. In (Snell et al., 2017), Snell et al provided an analysis over recent approaches involving meta-learning and further extended Prototypical Networks to zero-shot learning and achieved state-of-the-art results on the CU-Birds dataset. In (Sung et al., 2018), Sung et al presented a conceptually simple, flexible, and general framework for few-shot learning known as the Relation Network (RN). In (Ravi & Larochelle, 2016), Ravi et Larochelle proposed an LSTM-based meta-learner model that could learn the exact optimization algorithm used to train another learner neural network classifier in the few-shot regime. In this work, we will be focusing on the performance of simple baseline methods and distance metric learning methods.

Distance metric learning

Distance metric learning (DML) is defined as the problem

[1]University of Waterloo, Waterloo, Ontario, CA. Correspondence to: Yuzhe You, Zhen Gong <y28you@uwaterloo.ca, z33gong@uwaterloo.ca>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

of learning the embeddings where examples from the same class are closer than examples from different classes (Chen et al., 2019). The task can be cast as an optimization problem with triple constraints, and an optimal sampling strategy is essential for DML due to a vast number of triple constraints (Chen et al., 2019). In (Weinberger & Saul, 2009), Weinberger & Saul directly applied PCA to reduce the dimensionality of input feature and demonstrated how to learn a Mahalanobis distance metric for kNN classification from labeled examples. In (Xing et al., 2002), Xing et al proposed an algorithm that, given examples of similar (or dissimilar) pairs of points in $\mathbb{R}^n$, learns a distance metric over $\mathbb{R}^n$ that respects these relationships. In (Lim et al., 2013), Lim et al presented an efficient and robust structural metric learning algorithm that enforces group sparsity on the learnt transformation while optimizing for structured ranking output prediction. In (Qian et al., 2015), Qian et al addressed the highly expensive computational cost of DML by developing two strategies within SGD, mini-batch and adaptive sampling, to effectively reduce the number of updates in SGD. Different forms of constraints for metric learning are also developed, with early work focusing on optimizing pairwise constraints, while later works developed the triplet constraints (Weinberger & Saul, 2009). For SoftTriple loss, Qian et al focuses only on triplet constraints (Qian et al., 2019).

3. Overview

In this section, we first provide an overview of the few-shot classification problem and the Baseline++ model with a distance-based classifier proposed in (Chen et al., 2019), which is known to be competitive to the current state-of-art meta-learning algorithms. We then provide an explanation of the novel SoftTriple Loss and its improvements from the conventional SoftMax Loss.

3.1. Few-Shot Classification

Few-shot classification is defined as the problem of learning to generalize to unseen classes during the training stage of the model (Shi et al., 2021). The goal is to learn a classifier that is capable of recognizing unseen classes with only limited labeled examples in order to mitigate the need for large-scale annotated data in the real world (Chen et al., 2019; Shi et al., 2021). A N-way K-shot image classification problem is defined as the following: given a support set composed of N labels and K labeled images for each label, and a query set with Q query images, the task is to classify the query images among the N classes given the N×K images in the support set (Bennquin, 2020). When the value of K is small (e.g. $K < 10$), the problem is denoted as few-shot image classification (Bennquin, 2020). The focus of few-shot classification is about adapting to unseen tasks quickly through outstanding feature extraction ability (Shi et al., 2021).

In recent years, significant progress has been made to improve model adaption to novel classes with limited samples in few-shot classification problems (Shi et al., 2021). Some of the dominant methods utilized include optimization-based and metric-based meta-learning, initialization-based methods, domain adaption, and more (Chen et al., 2019; Shi et al., 2021). Additionally, a baseline method with a distance-based classifier was proposed in (Chen et al., 2019), which is known to achieve competitive performance with state-of-art meta-learning methods on certain datasets (Chen et al., 2019). The proposed method, denoted as *Baseline++*, is a variant of the original baseline model which makes predictions based on the cosine distance between the input feature and the learned weight vectors representing each class, thus achieving reduced intra-class variations in comparison to a model with a linear layer (Chen et al., 2019). A more detailed explanation of the underlying infrastructure of the Baseline and Baseline++ models will be included in the *Methodology* section.

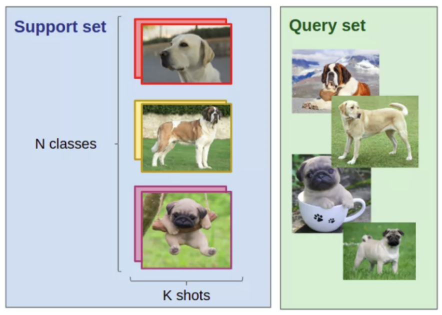

Figure 1. Example of a few-shot classification problem. In this example, there are N classes with K examples in each class, thus constituting an N-way K-shot classification problem. Taken from (Bennquin, 2020).

3.2. SoftTriple Loss

First proposed by Qian et al in 2019, the SoftTriple loss is designed to improve SoftMax loss by introducing multiple centers for each class, as having multiple of them could result in improved model performance in terms of capturing the hidden distribution of the dataset by reducing the intra-class variance (Qian et al., 2019). In the conventional SoftMax loss, each class is only assigned one representative center in the last fully connected layer, that examples within the same class are all collapsed into the same center (Qian et al., 2019). However, this may be impractical when applying to cases of real-world data, as illustrated in Fig. 2, pictures of birds from the same species may have different poses and colors, and real-world data usually contains multiple local clusters and a single center may be insufficient to capture its inherent structure (Qian et al., 2019).

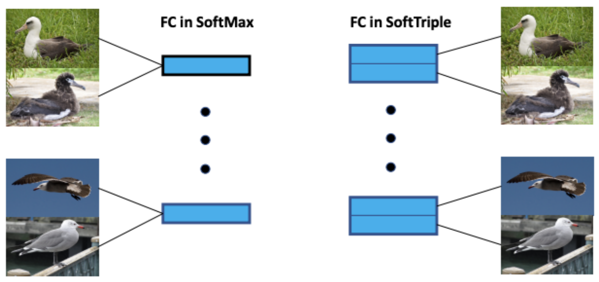

Figure 2. Illustration of SoftTriple loss in comparison to the conventional SoftMax loss. The new proposed SoftTriple Loss placed multiple centers for each class as opposed to only one in the conventional SoftMax loss. Taken from (Qian et al., 2019).

In contrast, the novel SoftTriple loss deems a lot more flexible when it comes to modeling intra-class variance in real-world datasets, as it keeps track of multiple centers for each class and assigns each image to one of them (Qian et al., 2019). The SoftTriple loss combines Triple loss with SoftMax loss and uses a relaxed version of similarity between example $\mathbf{x}_i$ and the class c defined as the following (Qian et al., 2019):

$$S'_{i,c} = \sum_k \frac{\exp(\frac{1}{\gamma}\mathbf{x}_i^\top \mathbf{w}_c^k)}{\sum_k \exp(\frac{1}{\gamma}\mathbf{x}_i^\top \mathbf{w}_c^k)} \mathbf{x}_i^\top \mathbf{w}_c^k$$

By applying the smoothed similarity, we can then derive the SoftTriple loss (Qian et al., 2019):

$$l_{\text{SoftTriple}(\mathbf{x}_i)} = -\log \frac{\exp(\lambda(S'_{i,y_i} - \delta))}{\exp(\lambda(S'_{i,y_i} - \delta)) + \sum_{j \neq y_i} \exp(\lambda S'_{i,j})}$$

Figure 6 illustrates the differences between SoftMax loss and SoftTriple loss (Qian et al., 2019). The dimension of the FC layer is first increased from the SoftMax loss to include multiple centroids for each class, then the distribution over different classes is computed with the similarity obtained from each class (Qian et al., 2019).

4. Methodology

4.1. Dataset Selection

For our main experiment, we use the CUB-200-2011 (*Caltech-UCSD Birds-200-2011*) dataset for fine-grained classification. The CUB-200-2011 dataset is an extended version of the CUB-200 dataset, with roughly double the number of images per class and new part location annotations (Wah et al., 2011). The CUB dataset contains 200 classes and 11,788 images in total (Chen et al., 2019). For evaluation purpose, we reproduce the experimental setup in (Chen et al., 2019) by randomly splitting up the dataset into 100 base, 50 validation, and 50 novel classes. Standard data augmentation including random crop, left-right flip and color jitter is applied during the training stage (Chen et al., 2019). The input size of the images used from the CUB dataset is 84 by 84.

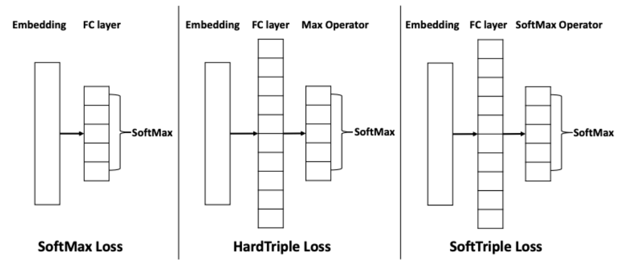

Figure 3. Illustration of differences between SoftMax loss and SoftTriple loss. In SoftTriple loss, the dimension of the FC layer is increased to include multiple centroids for each class. Taken from (Qian et al., 2019).

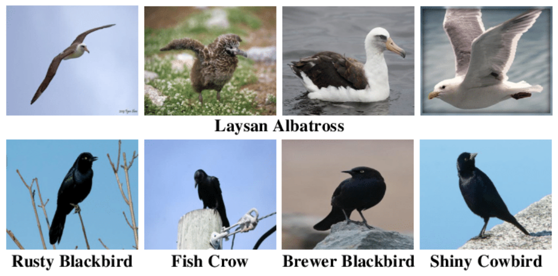

Figure 4. Examples of CUB-200-2011 dataset (Wah et al., 2011). Large variance in the same subcategory is shown in the first row, while small variance among different subcategories is shown in the second row. Picture and description taken from (He et al., 2018).

In addition, we run the models on the Omniglot dataset and cross-domain character recognition (Omniglot → EMNIST) as an additional experiment and compare the results with the original Baseline and Baseline++ models. The Omniglot dataset is a commonly used dataset for character recognition and for evaluating few-shot classification algorithms (Chen et al., 2019). It contains 1,623 characters from 50 languages, and we follow the procedure in (Chen et al., 2019) to augment the classes by rotations in 90, 180, 270 degrees, which results in 6492 classes (Chen et al., 2019). These classes are split into 4112 base, 688 validation, and 1692 novel classes (Chen et al., 2019).

For the cross-domain character recognition experiment, Latin characters are removed from the Omniglot dataset and no data augmentation is applied to prevent over-fitting (Chen et al., 2019). While Omniglot has 1597 base classes, EMNIST dataset has 62 classes in total, and these will be splitted into 31 validations and 31 novel classes (Chen et al., 2019). The Omniglot characters will also be inverted from white-on-black to black-on-white (Chen et al., 2019). For both the Omniglot experiment and the cross-domain character recognition experiment, we use an input size of 28 by 28.

Please note that in the original study of Baseline and Baseline++ models, the *mini*-ImageNet dataset is also utilized for further model evaluation on object recognition and cross-domain scenario (*mini*-ImageNet $\rightarrow$ CUB) (Chen et al., 2019). However, conducting these experiments will require downloading 155G of ImageNet data. Due to time constraints and the enormous size of the data, we decide to omit *mini*-ImageNet for this study.

4.2. Implementation Details

In this study, we investigate the feasibility of incorporating the SoftTriple loss into the general structure of Baseline++ model as an attempt to further improve few-shot classification accuracies. Similar to the original Baseline++ model, our model (denoted as *BaselineST*) is divided into a training stage and a fine-tuning stage.

Training stage. In the original Baseline and Baseline++ models, a feature extractor f_θ and a classifier $C(.|\mathbf{W}_b)$ are trained from scratch by minimizing a standard cross-entropy classification loss L_{pred}. The base class data is used as training examples that are fed into the feature extractor, and the resulted feature matrix (embedding) is fed into the classifier to produce corresponding predictions. The outputted scores, along with the target label of the example, are then used as the inputs for the cross-entropy loss function to minimize loss. In order to incorporate the SoftTriple loss into the training stage for BaslineST, we make the following adjustments:

After the training examples are fed into the feature extractor f_θ, the resulting output is used directly as the embedding for the SoftTriple loss. For all experiments, we use a four-layer convolution backbone. In the case of the CUB dataset, the output dimensionality of the backbone is 1600, while in the case of the Omniglot dataset and the cross-domain character recognition the output dimension of the extracted feature is 64. This number is equivalent to the dimensionality of embedding we use to define our SoftTriple loss function.

In addition, we select a λ value of 20, a γ value of 0.1, a τ value of 0.2, a margin size 0.01, and a K with value 10 as recommended in (Qian et al., 2019) for the SoftTriple loss. The λ parameter controls the amount of regularization applied to the model, and is otherwise known as the shrinkage parameter when the value is non-negative. The γ parameter stands for the scale factor that is used to calculate distances between data and similarities between examples. The τ parameter acts as another form of regularization added to the loss, and the margin is defined as the distance from the data point to the decision surface and is used to to break the tie explicitly (Qian et al., 2019). Selecting the number of centroids for each class could be quite challenging, and for this study the number of center is set to $K = 10$, the same value used in (Qian et al., 2019) on the CUB-2011 dataset.

Nevertheless, minimizing only the SoftTriple loss during the training stage would imply training only the feature extractor but not the linear classifier. We find this design undesirable as training only the feature extractor without taking into account the classifier used for fine-tuning and testing stages would likely result into poor performance. Therefore, the rest of the training stage is designed as the following: after the output of the feature extractor is used as the input to calculate the SoftTriple loss, the embedding is then passed into the classifier to produce final scores, which will then be utilized to calculate the standard cross-entropy loss. The final loss calculated is a combination of SoftTriple loss and cross-entropy loss as the following:

$$L_{pred} = w_1 * L_{Cross-Entropy} + w_2 * L_{SoftTriple}$$

where w_1 and w_2 are weights applied to each loss respectively. $L_{Cross-Entropy}$ takes the scores converted from the embedding by the classifier and y as its inputs, while $L_{SoftTriple}$ takes the output embedding from the feature extractor and y as its inputs. Please note that applying $w_1 = 1$ and $w_2 = 0$ would be equivalent to the original design of minimizing only the cross-entropy loss, while setting $w_1 = 0$ and $w_2 = 1$ is equivalent to training only the feature extractor by minimizing the SoftTriple loss. The goal of this design is to achieve a balance between the SoftTriple loss and the standard cross-entropy loss, such that the weights can be generalized enough to achieve satisfiable accuracies in few-shot classification while also ensuring the performance of the classifier used is maximized during the testing stage.

For the training stage, the method is trained from scratch with the Adam optimizer. For w_1 and w_2 that are not 1 and 0, we initialize the learning rate as 10^{-4} to avoid the exploding gradient problem. For more detailed reasons regarding this design choice, please see *Issues & Limitations* in the results section. We select the Baseline++ design as the classifier for our model, where an input feature $f_\theta(\mathbf{x}_i)$'s cosine similarity to each weight vector $[\mathbf{w}_1, \mathbf{w}_2, ...\mathbf{w}_3]$ is computed to obtain the similarity scores $[s_(i,1), s_(i,2), ..., s_(i,c)]$ for all classes (Chen et al., 2019). The prediction probability for each class is then obtained by normalizing the similarity scores with a SoftMax function, which is achieved by feeding the output into the cross-entropy loss (Chen et al., 2019). The SoftMax function is also responsible for preventing the learned weight vectors from collapsing into zeros (Chen et al., 2019).

Fine-tuning stage. For the fine-turning stage, we retain the general structure from the Baseline and Baseline++ models. During this stage, the pre-trained network parameter θ is fixed in the feature extractor f_θ. To adapt the model to recognize novel classes in the fine-tuning stage, a new Baseline++ classifier $C(.|\mathbf{W}_n)$ is trained by minimizing the cross-entropy loss function with the few labeled examples from the support set. In each experiment, 5 classes are randomly sampled from novel classes, and k instances are sampled from each class for the support set and 16 for the query set (Chen et al., 2019). The results are then averaged over 600 experiments and the entire support set is used to train a new Baseline++ classifier for 100 iterations with a batch size of 4 (Chen et al., 2019). We select the stochastic gradient descent optimizer with learning rate of 0.01, momentum of 0.9, dampening of 0.9, and a weight decay of 0.001. After the model is fine-tuned with the support set, we run the model on the test set to evaluate its performance based on its test accuracy.

5. Results

5.1. Re-evaluation of Baseline and Baseline++

We first conduct experiments using the original implementation of Baseline and Baseline++ models to verify the results provided by (Chen et al., 2019). We run the source code provided by the original authors and use the same common settings of 1-shot and 5-shot classification, in which 1 or 5 labeled instances are available from each novel class (Bennquin, 2020).

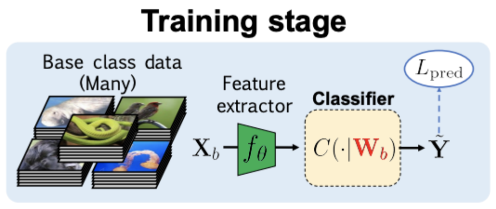

Figure 5. Training stage of Baseline and Baseline++ models. During this stage, a feature extractor and a classifier is trained from scratch using base examples with a standard cross-entropy classification loss. Taken from (Chen et al., 2019).

For the CUB datatset, we use a four-layer convolution backbone (Conv-4) with embedding dimensionality of 1600 and an input size of 84 by 84. For Omniglot and cross-domain character recognition experiment, we use a four-layer convolution backbone (Conv-4S) with only 1 input channel and an output dimension of 64 with an input size of 28 by 28.

During the training stage, we follow the original procedure of training a feature extractor f_θ and a classifier $C(.|\mathbf{W}_b)$ from scratch by minimizing standard cross-entropy classification loss (denoted as L_{pred}) using training examples in the base classes (Chen et al., 2019). During the fine-tuning stage, the pre-trained network parameter θ is fixed in the feature extractor f_θ and a new classifier $C(.|\mathbf{W}_n)$ is trained by minimizing the cross-entropy loss with the support set's labeled examples in the novel classes $\mathbf{X}_n$ (Chen et al., 2019).

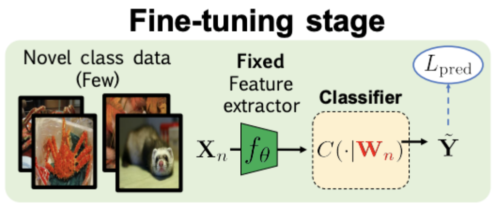

Figure 6. Fine-tuning stage of Baseline and Baseline++ models. In this stage, the pre-trained parameters θ are fixed while a new classifier is trianed based on the support set. Taken from (Chen et al., 2019).

Table 1. **5-way 5-shot classification results for the original Baseline model**. Data augmentation is only applied on CUB dataset. All experiments are conducted with a Conv-4 backbone.

DATA SET	REPORTED	OURS	MATCHING?
CUB	64.16± 0.71	69.19± 0.66	×(BETTER)
OMNIGLOT	99.12± 0.13	99.20± 0.09	√
CROSS-CHAR	86.00± 0.59	86.23± 0.56	√

In the training stage for the Baseline and Baseline++ methods on the CUB dataset, we train the feature extractor and the classifier for 200 epochs with a batch size of 16. Please note that the number of epochs here differs from the 400 epochs stated in the open-review paper from (Chen et al., 2019). We make such change under the advice of the source files distributed by the authors as one of them states that using 400 epochs for the baseline models will lead to over-fitting. For the additional experiments on Omniglot and cross-domain character recognition on EMNIST, we take the advice from the original paper to use an epoch size of 5 and uses the validation set to select the epoch or episode with the best accuracy for both methods to avoid over-fitting (Chen et al., 2019).

The reproduced results with comparisons to the reported results in the original paper are recorded in tables 1 $\sim$ 4. For this experiment, training augmentation is omitted for Omniglot and cross-domain character recognition to reduce the risk of over-fitting. We re-evaluate the original study with the following criteria: if there exists a difference larger than 1% between the reported result and the reproduced

Table 2. **5-way 5-shot classification results for Baseline++ model proposed in (Chen et al., 2019)**. Data augmentation is only applied on CUB dataset. All experiments are conducted with a Conv-4 backbone.

DATA SET	REPORTED	OURS	MATCHING?
CUB	79.34± 0.61	79.29± 0.63	√
OMNIGLOT	99.38± 0.10	99.04± 0.6	√
CROSS-CHAR	87.31± 0.58	80.92± 0.62	×(WORSE)

Table 3. **5-way 1-shot classification results for the original Baseline model**. Data augmentation is only applied on CUB dataset. All experiments are conducted with a Conv-4 backbone.

DATA SET	REPORTED	OURS	MATCHING?
CUB	47.12± 0.74	47.72± 0.71	√
OMNIGLOT	94.89± 0.45	94.87± 0.44	√
CROSS-CHAR	63.94± 0.87	63.15± 0.90	√

result, then it is considered a mismatch between the models' reported performance and their actual performance.

While the majority of our experiments' accuracies match the values indicated in the report, we do make some interesting observations. In the case of the Omniglot dataset, though both Baseline and Baseline++ models achieve very high accuracies (99% in 5-shot and 93% $\sim$ 95% in 1-shot), the performance of the Baseline++ model is not strictly superior than the Baseline model as implied in the original study. In (Chen et al., 2019), the reported test accuracy of Baseline++ on the Omniglot dataset for 5-way 1-shot classification is 94.89± 0.45, higher than Baseline's accuracy of 95.41± 0.39 by about 0.52%. Nevertheless, in our case, the Baseline model's performance (94.87± 0.44) exceeds Baseline++'s (93.50± 0.54) by 1.37 %. We interpret this as a minor discrepancy resulted from the randomization of examples chosen for the feature evaluation. Therefore, the Baseline model is just as competitive as the Baseline++ model in the case of the Omniglot dataset.

Furthermore, we notice a significant discrepancy between the Baseline++ model's reported accuracies and actual performance in terms of cross-domain character recognition (Omniglot $\rightarrow$ EMNIST). For 5-way 5-shot classification, our reproduced result for Baseline++ is 80.92± 0.62, much lower than the reported accuracy of 87.31± 0.58. And for 5-way 1-shot classification, our reproduced result is 59.31± 0.86, lower than the reported result with a difference larger than 5%. More importantly, the performance of the Baseline++ model is noticeably poorer than the original Baseline model, as Baseline achieves an accuracy of 86.23± 0.56 in 5-shot classification and 63.15± 0.90 in 1-shot classification. We follow the experimental setup provided by the original paper to train the models with a Conv-4 backbone and without data augmentation. We have reached out to the original authors regarding the discrepancy, but have yet earned a response during the time of drafting this report. We remain unsure what the cause of the performance mismatch is.

Table 4. **5-way 1-shot classification results for Baseline++ model proposed in (Chen et al., 2019)**. Data augmentation is only applied on CUB dataset. All experiments are conducted with a Conv-4 backbone.

DATA SET	REPORTED	OURS	MATCHING?
CUB	60.53± 0.83	62.42± 0.89	×(BETTER)
OMNIGLOT	95.41± 0.39	93.50± 0.54	×(WORSE)
CROSS-CHAR	64.74± 0.82	59.31± 0.86	×(WORSE)

For the purpose of fair comparison, for this study, we will be evaluating the performance of *BaselineST* based on our reproduced results of Baseline and Baseline++ models.

5.2. Evaluation of BaselineST

After reproducing the reported experiments from the original paper and re-evaluating Baseline and Baseline++'s test accuracies on the selected dataset, we move on to evaluating the performance of our implementation of *BaselineST*.

We select a series of weights (w_1, w_2) and apply each to the standard cross-entropy loss and the SoftTriple loss respectively, and sum them together as the new minimized hybrid loss during the training stage. For w_1 and w_2 that are not 1 and 0, we apply a learning rate of 10^{-4} and train the model for 1000 epochs with batch size of 16 on the CUB dataset. For the cross-domain character recognition experiment, we apply a learning rate of 10^{-3} and train the model for 5 epochs. We select the Adam optimizer for the training stage. For each variation of *BaselineST*, we apply the following weights:

$$BaselineST_1 : w_1 = 0.8,\ w_2 = 0.2$$
$$BaselineST_2 : w_1 = 0.6,\ w_2 = 0.4$$
$$BaselineST_3 : w_1 = 0.5,\ w_2 = 0.5$$
$$BaselineST_4 : w_1 = 0.0,\ w_2 = 1.0$$

The test results of each method have been recorded in tables 5 & 6. In the case of CUB dataset (table 5), we observe that applying a combination of weights $w_1 = 0.8$ and $w_2 = 0.2$ (*BaselineST*$_1$) minorly improves the original Baseline++ model as it increases the 1-shot classification accuracy by 1.05% (62.42±0.89 $\rightarrow$ 63.47±0.87) and the 5-shot classification accuracy by 0.63% (79.29±0.63 $\rightarrow$ 79.92±0.65). In *BaselineST*$_4$, we experiment with the weights $w_1 = 0$ and $w_2 = 1$, which is equivalent to minimizing only the Soft-

Triple loss during the training stage. As we anticipated in the *Methodology* section, training only the feature extractor with the SoftTriple loss and neglecting the classifier used in later stages indeed result in relatively poor performance of the model. Nevertheless, to our surprise, the *BaselineST*$_4$ model is still able to achieve a performance that is not so far off from the original Baseline model (47.74± 0.71 $\rightarrow$ 43.85± 0.68m, 69.19± 0.66 $\rightarrow$ 65.09± 0.64).

In (Chen et al., 2019), the test accuracies of Baseline++ are reported after training the model for 200 epochs with a learning rate of 10^{-3}. Since using the new hybrid loss function with the same learning rate would result in the exploding gradient problem, we lower the learning rate of the Adam optimizer to 10^{-4} and increase the epoch number to 1000. For the purpose of fair comparison, we also train the Baseline++ model with the original implementation for 1000 epochs with a learning rate of 10^{-4} and examine the resulted performance. The corresponding model has been recorded as Baseline++* in table 5. We find that the performance of Baseline++* matches the reported results of Baseline++ in 5-shot classification, with a slight increase of 2% in 1-shot classification.

After conducting our experiment on the CUB dataset, we proceed to test each variation of the *BaselineST* model in the cross-domain character recognition scenario. The results of each method have been recorded in table 6. While the test results of *BaselineST* is not far off from the benchmark Baseline++ performance (59.31±0.86 $\rightarrow$ 57.12±0.83, 80.92±0.62 $\rightarrow$ 79.32±0.62), we are unfortunately unable to exceed the state-of-the-art performance of the original Baseline model. As an extra experiment, we then select the *BaselineST* variant with the highest test accuracies (*BaselineST*$_3$) and retrain the model with a Baseline classifier that only consists of one simple linear layer (denoted as *BaselineST*$_3^*$ in table 6). Our results show that replacing the Baseline++ classifier with the Baseline one may improve the performance of the *BaselineST* model by around 2.3% $\sim$ 2.7% (57.12±0.83 $\rightarrow$ 59.42±0.90, 79.32±0.62 $\rightarrow$ 82.04±0.64).

Our experiments show that choosing different combinations of loss weights will results in varying performance on each dataset. For example, selecting weights $w_1 = 0.8$, $w_2 = 0.2$ achieves the best performance among all *BaselineST* variants on the CUB dataset, and the test accuracies are competitive to the state-of-the-art performance of the Baseline++ model. On the other hand, choosing weights $w1 = 0.5$, $w2 = 0.5$ achieves the best test accuracies for the *BaselineST* model in the cross-domain character recognition scenario. Furthermore, while applying different values of (w_1, w_2) drastically affects the performance of the *BaselineST* model on the CUB dataset, adjusting the weights of the two losses does not have as much impact on the model's performance when it comes to cross-domain character recognition, as the test results demonstrate more consistent behaviours.

Table 5. **Performance of BaselineST on the CUB dataset in comparison to Baseline & Baseline++.**

METHOD	1-SHOT	5-SHOT
BASELINE	47.74± 0.71	69.19± 0.66
BASELINE++	62.42± 0.89	79.29± 0.63
BASELINE++*	64.42± 0.90	79.24± 0.64
BASELINEST$_1$	63.47± 0.87	79.92± 0.65
BASELINEST$_2$	61.29± 0.82	78.77± 0.64
BASELINEST$_3$	58.55± 0.89	77.92± 0.63
BASELINEST$_4$	43.85± 0.68	65.09± 0.64

Table 6. **Performance of BaselineST on cross-domain character recognition (Omniglot $\rightarrow$ EMNIST) in comparison to Baseline & Baseline++.**

METHOD	1-SHOT	5-SHOT
BASELINE	63.15± 0.90	86.23± 0.56
BASELINE++	59.31± 0.86	80.92± 0.62
BASELINEST$_1$	55.63± 0.84	78.25± 0.62
BASELINEST$_2$	54.76± 0.88	76.93± 0.67
BASELINEST$_3$	57.12± 0.83	79.32± 0.62
BASELINEST$_4$	55.43± 0.88	78.62± 0.63
BASELINEST$_3^*$	59.42± 0.90	82.04± 0.64

In addition, due to the Omniglot dataset having a huge number of bases classes (4112 classes in total after applying data augmentation), we notice that using a combination of two losses significant increases the training time. Using our GPU, training for 1 epoch takes more than 9 hours. While we are able to re-evaluate Baseline and Baseline++'s benchmark performance by reproducing the original experiments on the Omniglot dataset, due to our time constraints and the large number of experiments we need to conduct with each method, we decide to omit the Omniglot dataset for this experiment.

5.3. Experiment with Centroid Numbers

In section 5.2, we evaluate the performance of the *BaselineST* model on the CUB dataset by selecting different combinations of weights applied to the standard cross-entropy loss and the SoftTriple loss during the training stage, while maintaining a K value of 10. Among all *BaselineST* variants, *BaselineST*$_1$ achieves the highest test accuracy with minor improvements from the Baseline++ model.

In this following experiment, we maintain the weights used for the *BaselineST*$_1$ model ($w_1 = 0.8$, $w_2 = 0.2$) and train the model with different values of K as an attempt to surpass *BaselineST*$_1$'s performance in table 5. We train the model with $K = 2, 5, 10, 15$ while retaining the rest of the parameter values. All test accuracies have been recorded in table 7. From our results, we observe that choosing a relatively smaller centroid number (e.g. $K = 5$) further

Table 7. **Performance of BaselineST$_1$ on the CUB dataset with different values of K.** Training augmentation is applied.

CENTROID #	1-SHOT	5-SHOT
K=2	63.79± 0.85	80.18± 0.62
K=5	63.41± 0.84	80.66± 0.59
K=10	63.47± 0.87	79.92± 0.65
K=15	63.44± 0.85	79.95± 0.63

improves $BaselineST$'s accuracy in 5-way 5-shot classification by about 0.74% (79.92±0.65 → 80.66±0.59). On the other hand, adjusting the number of centers does not make as much difference in 5-way 1-shot classification, as all test accuracies maintain at a value around 63.5%, with $K = 2$ achieving the highest percentage at 63.79±0.85.

5.4. Issues & Limitations

Exploding gradient. During our implementation of *BaselineST*, we notice that retaining a learning rate of 10^{-3} while combining two losses will result in loss with value NaN after around 100 epochs when used on the CUB dataset. We suspect that it may be an example of the exploding gradient problem, in which the gradients updating the weights grow exponentially due to high learning rate. To mitigate this problem, we therefore reduce the learning rate of the Adam optimizer to 10^{-4} for the training stage. To account for the drastic decrease in learning rate, we increase the number of training epochs from 200 to 1000.

Mismatch of results. As mentioned previously in *Re-evaluation of Baseline and Baseline++*, we are unable to reproduce the reported performance of the Baseline++ model on the cross-domain character recognition experiment using the described experimental setup from the paper. More importantly, our reproduced results show that the original Baseline model in fact achieve a higher test accuracy than the Baseline++ classifier. We have reached out to the authors regarding this matter, but have earned no response at the time of finalizing this report. In this study, we compare the performance of *BaselineST* with our reproduced results of Baseline and Baseline++ for the sake of fair comparison.

Time constraints. In additional to the CUB dataset, the authors of (Chen et al., 2019) experimented the Baseline++ model on the *mini*-ImageNet dataset and claimed that Baseline++ was also able to achieve competitive performance with the state-of-the-art meta-learning methods (Chen et al., 2019). However, conducting these experiments will require downloading 155G of the original ImageNet data. Due to time constraints and the enormous size of the data, we decide to omit *mini*-ImageNet for this study. Furthermore, while we are able to re-evaluate Baseline and Baseline++'s performance on the Omniglot dataset, we decide to omit it for the *BaselineST* experiment due to long training time ($>$ 9 hours for 1 epoch on a GPU). We believe this is due to the Omniglot dataset having a large number of base classes (4112 after data augmentation), and using a combination of two losses significantly increases the training time.

6. Future Work

Deeper backbones. In this study, we use a four-layered convolution network (Conv-4) as the backbone for all methods and experiments. In (Chen et al., 2019), the authors state that using a deep backbone shrinks the performance gap between different methods in the setting of limited domain differences between base and novel classes, such that reducing intra-class variation is not as critical when the backbone is deep enough. Due to time constraints, we were unable to verify this claim with the existing implementations of Baseline and Baseline++ models. Furthermore, while the original study tested out applying various depths of backbones on the CUB dataset, the Omniglot and cross-domain character recognition experiments only support the usage of a Conv-4S backbone. In the future, we may potentially evaluate the effects of using backbones with various depths on the performance of *BaselineST*, and consider the possibility of enhancing the backbone as an attempt to improve Baseline++ classifier's performance on cross-domain tasks.

New classifier design. As stated at the beginning of our report, the goal of our study is to investigate the feasibility of incorporating the SoftTriple loss into the general structure of the Baseline++ model as an attempt to improve the state-of-the-art baseline performance. For our implementation of *BaselineST*, we focus on training the feature extractor with the SoftTriple loss while retaining the original design of the Baseline++ classifier with cosine distance similarity calculation. In the future, it would also be engrossing to investigate the possibility of building a new classifier with the SoftTriple loss by applying a k-nearest neighbor algorithm.

7. Conclusion

In this study, we investigate the feasibility of incorporating the novel SoftTriple loss into the general structure of the Baseline++ model as an attempt to further improve the state-of-the-art baseline performance in few-shot classification. We propose our new model as *BaselineST*, which combines the SoftTriple loss and the standard cross-entropy loss into one hybrid loss function. We find that when selecting a weight pair of $w_1 = 0.8$, $w_2 = 0.2$ for the combined loss and a K value of 5, *BaselineST* achieves a slight improvement in 5-way 5-shot classification on the CUB-200-2011 dataset. More future work needs to done to investigate the possibility of further enhancing the model's performance by deepening the backbones and building a novel classifier with the SoftTriple loss.

References

Bennquin, E. Few-shot image classification with meta-learning, Dec 2020. URL https://www.sicara.ai/blog/2019-07-30-image-classification-few-shot-meta-learning.

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C. F., and Huang, J.-B. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

He, X., Peng, Y., and Zhao, J. Fast fine-grained image classification via weakly supervised discriminative localization. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(5):1394–1407, 2018.

Lim, D., Lanckriet, G., and McFee, B. Robust structural metric learning. In *International conference on machine learning*, pp. 615–623. PMLR, 2013.

Qian, Q., Jin, R., Yi, J., Zhang, L., and Zhu, S. Efficient distance metric learning by adaptive sampling and mini-batch stochastic gradient descent (sgd). *Machine Learning*, 99(3):353–372, 2015.

Qian, Q., Shang, L., Sun, B., Hu, J., Li, H., and Jin, R. Softtriple loss: Deep metric learning without triplet sampling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6450–6458, 2019.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. 2016.

Shi, F., Wang, R., Zhang, S., and Cao, X. Few-shot classification with multi-task self-supervised learning. In *International Conference on Neural Information Processing*, pp. 224–236. Springer, 2021.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Sun, Q., Liu, Y., Chua, T.-S., and Schiele, B. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 403–412, 2019.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., and Hospedales, T. M. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1199–1208, 2018.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. The caltech-ucsd birds-200-2011 dataset, 2011. URL http://www.vision.caltech.edu/visipedia/CUB-200-2011.html.

Weinberger, K. Q. and Saul, L. K. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2), 2009.

Xing, E., Jordan, M., Russell, S. J., and Ng, A. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, 15, 2002.

A. Additional Experiment - *BaselineST+*

In addition to *BaselineST*, we implemented a supplementary model that utilizes the SoftTriple loss denoted as *BaselineST+*.

A.1. Overview

The *BaselineST+* method trains a feature extractor $f(\theta)$ and a classifier $C(.|\mathbf{W}_{CK})$ where a fully connected layer for each center is initialized, such that $\mathbf{W} \in \mathbb{R}^{d\times(C\cdot K)}$. The output $\hat{y}_{temp}$, expressed as $\mathbf{w}_{Ck}^{\top}f(\theta)\mathbf{x}_i$, is then fed through a $SoftMax_{norm}$ layer to derive the probability $P(C = c, K = k|\mathbf{x}_i)$ on each center, and element-wise multiplications are applied between $\hat{y}_{temp}$ and $P(\cdot)$ to determine the weight for each center. The resulting weights of each centroid from the same class is them summed up and fed through a SoftMax function to finalize the probability for each class. The equation for class prediction could be written as the following:

$$\hat{y} = \sigma\left(\sum_{k=1}^{K} \mathbf{w}_{Ck}^{\top}f(\theta)\mathbf{x}_i \odot \sigma_{norm}(\mathbf{w}_{Ck}^{\top}f(\theta)\mathbf{x}_i)\right)$$

A.2. Implementation Details

For this additional experiment, we train the *BaselineST+* model for 200 epochs with a batch size of 16, the same experimental set up in (Chen et al., 2019). We use the Adam optimizer with an initial learning rate of 10^{-4} for the backbones and 10^{-2} for the center. For the SoftTriple parameters, we apply the default settings $\lambda = 20, \gamma = 0.1, \tau = 0.2, margin = 0.01, C = 200, K = 10$. Standard data augmentation including random crop, left-right flip, and color jitter are applied in the training stage.

During the fine-tuning stage, we average all results over 600 experiments. In each experiment, we randomly sample 5 classes from novel classes, and in each class, we pick k instances for the support set and 16 for the query set. We use the entire support set to train a new classifier for 100 iterations with a batch size of 4. Since the fine-tuning stage has a smaller batch size and fewer images in the support and query set, we remove the margin and regularizer, and set $C = N$ as in N-way for the new classifier. We vary K to get a smooth loss.

A.3. Experiment Results

Similar to *BaselineST*, we conduct experiments on the most common setting in few-shot classifications, 1-shot and 5-shot classification, with 10-shot classification as an additional reference (Chen et al., 2019). We use a four-layer convolution backbone (Conv-4) with an input size of 84x84 as in (Snell et al., 2017) and perform 5-way classifications for only novel classes during the fine-tuning or meta-testing stage using the CUB-2011 dataset (Chen et al., 2019). The test results have been recorded in the following table:

METHOD	1-SHOT	5-SHOT	10-SHOT
BASELINE	47.74± 0.71	69.19± 0.66	69.86± 0.71
BASELINE++	62.42± 0.89	79.29± 0.63	81.91± 0.55
BASELINEST+	42.50± 0.76	66.61± 0.61	72.90± 0.63

From the above table, we observe that our implementation of *BaselineST+* is about 2% over margin better than the baseline model in 5-way 5-shot classification. The *BaselineST+* model also utilizes multiple centers for each class, thus enhancing the SoftMax function and producing an overall smoother loss in comparison to the Baseline model. However, the method performs poorly at 1-shot classification, and its discrepancy in terms of performance between 1-shot and 5-shot classifications inspired us to further experiment *BaselineST+* in 10-shot classification, as the model may achieve better test results if more samples are provided in the support set. It is as we expected that the performance of *BaselineST+* is 3% over margin better than the Baseline model. Nevertheless, the *BaselineST+* model does not introduce distance metric like the Baseline++ does, therefore the model still has relatively unsatisfactory results when compared to the Baseline++ model.

As an additional experiment, we also varied the number of centers used for the SoftTriple loss, and recorded the results in the following table:

Centroid #	1-shot	5-shot
K=2	42.40± 0.76	65.32± 0.69
K=4	41.20± 0.75	65.58± 0.72
K=8	39.82± 0.71	66.61± 0.68
K=10	39.82± 0.69	65.75± 0.69
K=100	37.78± 0.73	65.47± 0.71

From the recorded results, we see that the test accuracies of *BaselineST+* decrease further as the we increase the number of centroids in 1-shot classification. This is an expected behaviour as we only test the *BaselineST+* method in 5-way classifications, and introducing too many centers will complexify the classifier without actually contributing to the model's performance, as the number of centroids in each class is greater than the total number of classes. Nonetheless, we observe that when moving on to 5-shot classifications, the drawback of setting a huge number of centers diminished, as a sufficiently large K value would encourage similar centers to merge with each other while keeping the diversity in the generated centers (Qian et al., 2019).