On the Episodic Difficulty of Few-shot Learning

Bai Yunwei BAIYUNWEI@U.NUS.EDU
He Zhenfeng HE.ZHENFENG@U.NUS.EDU
Hu Junfeng JUNFENGH@U.NUS.EDU
National University of Singapore

Editors: Emtiyaz Khan and Mehmet Gönen

Abstract

Dog vs. hot dog and *dog vs. wolf*, which one tends to be a harder comparison task? While simple, this question can be meaningful for few-shot classification. Few-shot learning enables trained models to recognize unseen classes through just a few labelled samples. As such, trained few-shot models usually have to possess the ability to assess the similarity degree between the unlabelled and labelled samples. In each few-shot learning episode, a combination of the labelled support set and unlabelled query set are sampled from the training dataset for model-training. In the episodic settings of few-shot learning, most algorithms draw the data samples uniformly at random for training. However, this approach disregards concepts of difficulty of each training episode, which may make a difference. After all, it is usually easier to differentiate between a dog and a hot dog, versus the dog and a wolf. Therefore, in this paper, we delve into the concept of episodic difficulty, or difficulty of each training episode, discovering several insights and proposing strategies to utilize the difficulty. Firstly, defining episodic difficulty as a training loss, we find and study the correlation between episodic difficulty and visual similarity among data samples in each episode. Secondly, we assess the respective usefulness of easy and difficult episodes for the training process. Lastly, based on the assessment, we design a curriculum for few-shot learning to support training with incremental difficulty. We observe that such an approach can achieve faster convergence for few-shot algorithms, reducing the average training time by around 50%. It can also make meta-learning algorithms achieve an increase in final testing accuracy scores. Our major implementation is available at: https://github.com/WendyBaiYunwei/EpisodicDifficulty.

Keywords: Few-Shot Learning; Curriculum Learning; Deep Learning; Machine Learning; Data Selection

1. Introduction

While deep learning has empowered the salient progress of machine learning performance, the large models are data-hungry (Ford, 2018). Usually, data can be hard or expensive to collect (Zhou et al., 2018), thus keeping deep learning away from numerous real-life applications. Few-shot learning (FSL) algorithms enable trained models to classify unseen tasks based on just a few reference samples, which mitigates the data collection problems.

Most of the FSL algorithms sample query and support data points from a dataset uniformly at random (Wang et al., 2020). However, different query and support data point combinations can lead to vastly different training loss even for a converged model. This can be intuitively explained by the fact that, it is easier to differentiate between a dog

and a hot dog, versus a dog and a wolf. Through utilizing the concept of difficulty, we may enhance existing sampling approaches, data selection or other related techniques for few-shot learning. For example, curriculum learning (CL) rearranges the order of training data for faster convergence and better generalization.

Essentially, instead of chasing another state-of-the-art performance of few-shot learning models, we study the concept of difficulty behind episodic training. In this paper, we demonstrate that the similar-looking support set samples are more difficult than the contrary. Meanwhile, the similar-looking queries and their targets are the easier queries. Easy episodes can help achieve fast convergence. Hard episodes may hinder a model from fast convergence. Moreover, curriculum learning can cut down training time for FSL algorithms by a large margin and improve meta-learning performance by a small margin.

On the whole, the contribution of this paper can be summarized as follows:

- Propose and study a new definition for FSL episodic difficulty;
- Study the usefulness of easy and hard episodes respectively;
- Design a procedure to support FSL training with incremental difficulty. Demonstrate its effectiveness for model performance, in terms of both convergence and generalization.

2. Related Works

Few-shot learning algorithms aim to classify query samples based on just a few labelled reference samples. There are different approaches to FSL, including algorithm-based, model-based methods and data-augmentation approaches (Wang et al., 2020). Model-based FSL algorithms learn a similarity function which can output a similarity score for a combination of support set data points and a query data point (Sung et al., 2018). Algorithm-based algorithms typically include meta-learning algorithms (Ye et al., 2020; Vinyals et al., 2016), which "learn how to learn". For example, Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017) does not assume a fixed model, and can prepare a fast-adapting initialization for few-shot classification tasks. If we view few-shot learning in a hypothesis space, model-based methods constraints hypothesis space through prior knowledge whereas algorithm-based methods alter search strategies in the hypothesis space through prior knowledge (Wang et al., 2020). Data-augmentation approaches like (Miller et al., 2000) and (Hariharan and Girshick, 2017) serve to generate more data samples to improve generalization in few-shot learning.

Curriculum learning experimentally demonstrates that the same model can achieve better performance in terms of testing accuracy scores and convergence speed, when the easy data points are fed first, followed by the harder ones (Bengio et al., 2009). However, most of the curriculum learning approaches rely on the training loss or accuracy scores of a trained model to differentiate between the easy and hard training samples (Soviany et al., 2021; Wang et al., 2021). There are alternative measurements of difficulty. For example, (Stergiadis et al., 2021) defines difficulty as the size of support set images, and (Spitkovsky et al., 2010) measures the difficulty of language tasks through sentence lengths.

There are a few works indirectly associating few-shot learning with episodic difficulty. (Sun et al., 2019) proposes a new approach for meta few-shot learning while introducing a curriculum for the proposed approach. The curriculum, known as hard-task training, consists of difficult classes wrongly classified by the model. (Zhou et al., 2020) points out that some data points are better than others in training. Therefore, they introduce a class-wise similarity ratio to select the effective classes for training. A recent work, (Zhang et al., 2021), proposes a curriculum-learning approach for meta-learning to be coupled with a new model called the BrotherNet. Another recent work (Arnold et al., 2021) observes a normal distribution in uniform sampling, and believes that the uniform sampling can lead to the best accuracy scores of models.

3. Preliminary

Episodic sampling sub-samples few-shot tasks from a large base dataset. In every episodic sampling, one task represented by $\mathcal{T}$ is obtained as the output. We define each task $\mathcal{T}$ as a concatenation of support set $\mathcal{T}_{\mathcal{S}}$ and a query $\mathcal{T}_{\mathcal{Q}}$. In this paper, we refer to data points involved in each of the episode as a "combination".

To obtain $\mathcal{T}$, we first sample from the class distribution $p(C)$, and then sample from data distribution $p(x, y|C)$ conditioned on C. Here, x represents one datapoint and y represents one label. In few-shot learning, a n-way-k-shot episode is sampled in n classes to obtain k support data samples for each class. The query data point class belongs to one of the n support set classes. For one support set, there are n x k data points (Arnold et al., 2021).

4. Definitions and Existence of Episodic Difficulty

During every training episode, we have a combination of support set data points and query data points. It is intuitive that the similar-looking support set data points are more difficult because the model may get confused by other similar-looking but actually-wrong support set data points. Meanwhile, the different-looking support set target data points and query data points are more difficult. For example, in figure 1, the support set on the left side is arguably easier than the right side.

To define difficulty level $Difficulty$ formally, we define the input data points as $\{x_i \in X | 1 \leq i \leq |X|\}$, where X is the entire training dataset. Next, we use a metric $sim(x_i, x_j)$ to measure the degree of similarity of a pair of input data points x_i and x_j; the higher the metric value, the higher the pairwise similarity. In this work, we adopt cosine similarity as the similarity metric for images. In addition, we define the target data point corresponding to the query data point as x_{target}. Furthermore, we denote the support set as $S \in \mathcal{S}$, where $\mathcal{S}$ denote the entire support set, where each element corresponds to the prototype of each class. We further define the sum of similarity scores among one set of support set as a similarity score sum $\Omega(S)$, i.e: $\Omega(S) = \sum_{i,j} sim(x_i, x_j),\ \forall x \in S$. We have another support set S' for comparison, defined in the same way. Finally, we can define the difficulty level $Difficulty$ which satisfies following constraints:

$$Difficulty(S, x) \in \mathbb{R} \tag{1}$$

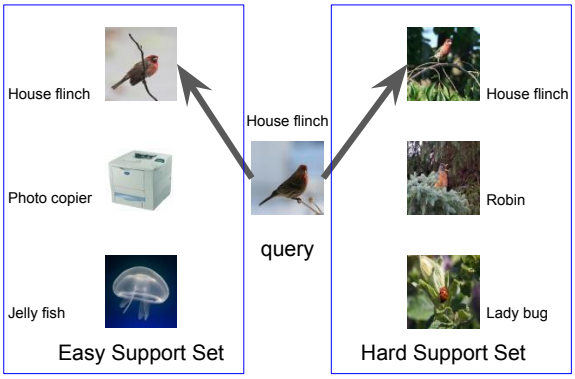

Figure 1: Given the same query image, the support set on the left is easier compared to the right, according to our definition of difficulty. As all the images are taken from the mini-imagenet dataset, both of the two sample combinations may appear during real training.

$$\begin{aligned} Difficulty(S, x_i) > Difficulty(S', x_j)\ iff \\ (sim(x_j, x_{target}) > sim(x_i, x_{target})) \wedge (\Omega(S) > \Omega(S')) \end{aligned} \tag{2}$$

After defining the difficulty measure, we verify experimentally whether it corresponds to the traditional definition of difficulty. Specifically, we measure the training accuracy scores of a converged model when the model is fed with data point combinations of different difficulty levels. The finding is positive as we discover that there is a correlation between our definition of difficulty and the traditional definition of difficulty in curriculum learning, as illustrated in figure 2.

A visualization of the difficulty contrast can also be illustrated by UMAP (McInnes et al., 2018b) projection in figure 3. We project the entire mini-imagenet training data points to a two-dimensional space. In the projection space, similar data points tend to stay closer. As such, it is expected that for harder combinations, the support set points tend to be near while the queries tend to be further away from their targets in the projection space. The projection results support our definition of difficulty.

Difficulty measurement in curriculum learning is usually reliant on a trained model; the higher the training accuracy scores of data samples, the lower their difficulty levels are. However, with the intuitive definition, we can omit the process of training an extra model to predict episodic difficulty.

We believe that the concept of difficulty is useful because of the intuition as follows. Given two models f_a, f_b and their corresponding parameters θ_a, θ_b, where a is for a randomly initialized model and b is for a nearly converged model. Suppose that we have two episodes of training tasks $\mathcal{T}_1 = (S_1, x_1), \mathcal{T}_2 = (S_2, x_2)$, where $Difficulty(\mathcal{T}_1) < Difficulty(\mathcal{T}_2)$, and there is a loss function $\mathcal{L}$ s.t $\mathcal{L}(f_a, \mathcal{T}_1) < \mathcal{L}(f_a, \mathcal{T}_2)$ and $\mathcal{L}(f_b, \mathcal{T}_1) < \mathcal{L}(f_b, \mathcal{T}_2)$. The gradient descent update with learning rate α will be $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}$. Given that f_a is a naive model while f_b is a mature model, the assumption is that loss of all task combinations will be high towards f_a and low towards f_b. Hence, task $\mathcal{T}_1$ will be more suitable to f_a as $\mathcal{T}_2$ has

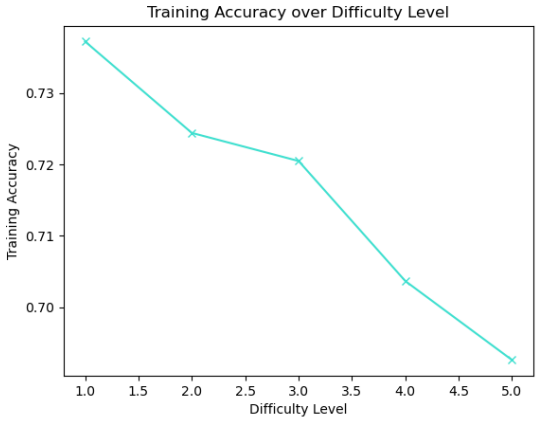

Figure 2: Illustrates the correlation between our definition of difficulty level and training accuracy of a converged model. In general, the higher the difficulty level, the lower the training accuracy scores. The training accuracy scores is obtained from a converged relation network (Sung et al., 2018) model on the mini-imagenet dataset. Here, 5 is the maximum difficulty level.

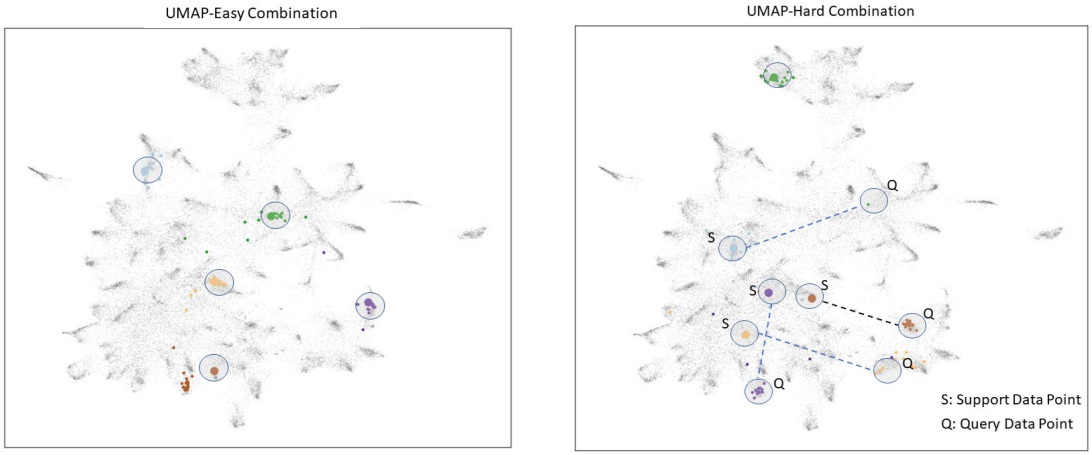

Figure 3: UMAP (McInnes et al., 2018a) projection of the entire training data points, with an easy combination of data samples highlighted on the left side, and a difficult combination on the right side. On the left side, the support set, represented by the larger dots, tend to be far away from each other, and the queries represented by the small colored dots generally stay near to their support set targets. On the right side, the support set, represented by the larger dots, tend to be nearer to each other, and the queries generally stay further away from their support set targets. Each color represents one class.

a chance of divergence. Meanwhile, task $\mathcal{T}_2$ will be more suitable to f_b as the loss of $\mathcal{T}_1$ approaches zero when the model nears convergence. As such, we study whether focusing on easy episodes at the beginning and adding to the difficulty later will make a difference.

5. Usefulness of Easy Episodes

Based on experiments, we find out that starting training on easy samples will allow FSL models to achieve a higher testing accuracy scores given a limited training time. For example, when given just 10,000 episodes of training, focusing on easy tasks allows an FSL model to achieve higher testing accuracy scores when compared to standard episodic training. We report the difference in table 1, based on which we can deduce that the easy episodes are more effective especially at the beginning. This observation can be explained by the fact that the easy episodes illustrate the concept of similarity in the most typical way. Therefore, when given limited computation resources, we can select the relatively easy samples to achieve relatively better model performance.

Table 1: Accuracy Achievable with 10,000 Episodes of Training

Task	**Dataset**	**Method**	**Backbone**	**With CL (%)**	**Without CL (%)**
5-way-1-shot	mini-imagenet	Relation Net	conv4	46.00	40.62
5-way-5-shot	mini-imagenet	Relation Net	conv4	60.65	58.64
5-way-1-shot	cifar-fs	Relation Net	conv4	51.31	47.09
5-way-5-shot	cifar-fs	Relation Net	conv4	65.72	63.14
2-way-1-shot	ARSC	Induction Net	BRNN (Lin et al., 2017)	78.55	69.75

6. Usefulness of Hard Episodes

In our work, we feed harder examples to the model during the training and study the effect of these hard episodes. In our experiment, we define 5 different difficulty levels, with 1 being the easiest and 5 being the hardest. More details on the implementation are given in the later section. We train the chosen FSL model from scratch on hard tasks (difficulty level 4, 5) alone. We report the result in figure 6. As shown in the figure, while the model can still achieve convergence, the convergence is slower. We further record the training loss for the hard training against standard episodic training. We find that the training loss for hard training tends to be high in general as shown in figure 4. This can be accounted for by the simplicity of our chosen model backbone and the lack of common patterns among the hard combinations. We visualize these combinations and find that, interestingly, some of them tend to be meaningless for training. Figure 5 is a real example of hard combination that appears during training. The query image on the right matches one target class on the left, but the combination is too difficult for even humans to discern. Therefore, we suspect that these hard-tasks should appear less frequently or even be omitted during training.

7. Leverage Curriculum Learning to Speed Up Convergence

Based on our understanding of easy and hard tasks, we design a curriculum for FSL and study its effects. Essentially, we feed the combinations of support set and query data points

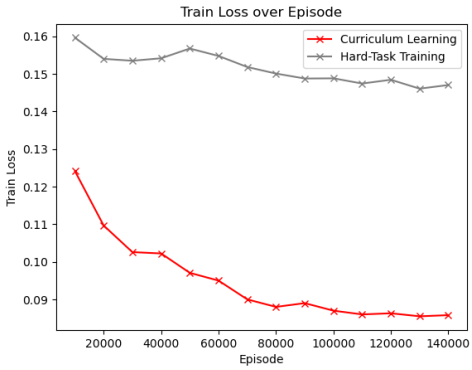

Figure 4: Average train loss for hard tasks consistently remains high when compared to the train loss of curriculum learning. The results were obtained from 5-way-1-shot training of mini-imagenet on relation network.

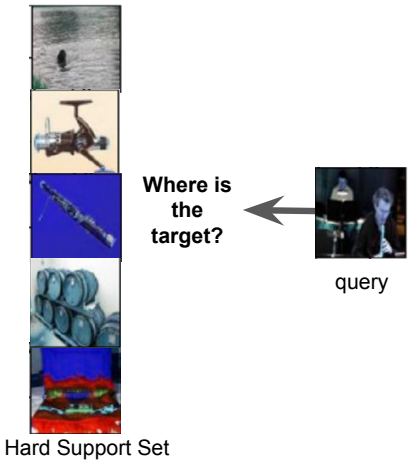

Figure 5: A real example of hard combination during 5-way-1-shot training. The training is done on mini-imagenet dataset.

in an order where the easiest ones are fed first, followed by the harder ones. To implement the difficulty differentiation, we rely on preprocessing the training dataset, which is elaborated on in the experiment section. After differentiating the difficulty level of different sample combinations, we build a data sampler, *Sampler* which takes in difficulty level and produces a combination of samples corresponding to the input difficulty level. We define one support set and query combination as (S, x), and the *Sampler* fulfills the following relationships:

$$Sampler(diff) := (S, x) \tag{3}$$

$$\begin{aligned} diff_i > diff_j \Rightarrow Difficulty(Sampler(diff_i)) \\ > Difficulty(Sampler(diff_j)) \end{aligned} \tag{4}$$

When designing the scheduling algorithm for curriculum learning, we first expose the learner to the $\frac{1}{MaxDifficulty}$ easiest subset of data points, or data points of difficulty 1, while hiding the rest. We increment the difficulty level one by one. In the end, the learner will be exposed to the full training set, which is of the highest difficulty level.

The overall training algorithm for curriculum learning can be simplified as algorithm 1.

Algorithm 1: Curriculum Training Algorithm with a Baby Step Scheduler
Input: *Model*, *Sampler*, *interval*
Output: *Model*
difficulty ← *minimum-difficulty*
for *episode in* {1...max-episodes} **do**
  Update *Model* **with** $Sampler(difficulty)$
  if $(episode \equiv interval) = 1$ *and* max-difficulty-not-reached **then**
    difficulty ← *difficulty* + 1
  end
end
return *Model*

Based on our experiments, easy-to-hard curriculum learning can significantly reduce training time for FSL algorithms. Figure 6 demonstrates the faster model convergence achieved through our approach. We take convergence speed as the number of episodes taken for the model to reach its highest testing accuracy score. As the speed of convergence is affected by initialization, we use 3 different manual seeds for initialization and calculate the average convergence episodes. No matter what seed we use, the convergence speed improvement is consistent without compromising any final testing accuracy scores. Table 2 illustrates the faster convergence on different tasks and algorithms. For example, for mini-imagenet dataset, the average episodes needed for 5-way-1-shot tasks convergence are cut down from 150,000 episodes to 80,000 episodes, reducing by near 50% on average. Table 2 also demonstrates the faster convergence in the NLP model. On average, the number of training episodes is reduced from 5,000 episodes to 3,000 episodes.

8. Leverage Curriculum Learning to Achieve Better Accuracy

Aside from the faster convergence for FSL algorithms, we notice that curriculum learning can slightly boost algorithm-based meta-learning accuracy scores, and we report the

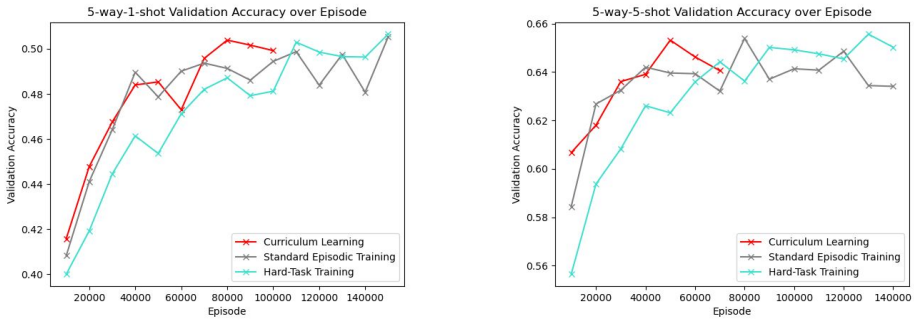

Figure 6: Illustrates that the proposed approach, represented by the red line, can achieve faster convergence as compared to the grey-line traditional approach. The illustration on the left is obtained from 5-way-1-shot tasks. The other illustration is obtained from 5-way-5-shot tasks. The algorithm is the relation network and the dataset is mini-imagenet. We train all models until the validation accuracy consistently drops. The extra episodes with dropping validation accuracy are truncated from the plot.

Table 2: Number of Episodes Needed for Convergence

Task	**Dataset**	**Method**	**Backbone**	**Accuracy**	**with CL (Episodes)**	**Without CL (Episodes)**
5-way-1-shot	mini-imagenet	Relation Net	conv4	50.23	80,000	150,000
5-way-5-shot	mini-imagenet	Relation Net	conv4	65.52	37,500	72,500
5-way-1-shot	cifar-fs	Relation Net	conv4	55.21	30,000	70,000
5-way-5-shot	cifar-fs	Relation Net	conv4	69.09	20,000	50,000
2-way-1-shot	ARSC	Induction Net	BRNN	81.25	3,000	5,000

improvements in table 3. We note that the improvement is not very significant, which is reflected among other works related to curriculum learning (Bengio et al., 2009; Soviany et al., 2021; Zhang et al., 2021).

Table 3: Accuracy Achievable with and without Curriculum Learning of Training

Task	Dataset	Method	Backbone	with CL (%)	Without CL (%)
5-way-1-shot	mini-imagenet	MAML	cnn4	$43.85_{\pm 1.30}$	$43.66_{\pm 1.75}$
5-way-5-shot	mini-imagenet	MAML	cnn4	$59.01_{\pm 0.95}$	$58.27_{\pm 1.05}$
5-way-1-shot	mini-imagenet	Prototypical Net	res12	$60.75_{\pm 0.18}$	$60.12_{\pm 0.21}$
5-way-5-shot	mini-imagenet	Prototypical Net	res12	$74.14_{\pm 0.16}$	$73.94_{\pm 0.16}$
5-way-1-shot	mini-imagenet	FEAT	res12	$62.50_{\pm 0.19}$	$62.17_{\pm 0.25}$
5-way-5-shot	mini-imagenet	FEAT	res12	$78.83_{\pm 0.29}$	$78.76_{\pm 0.34}$
5-way-1-shot	mini-imagenet	DeepSet	res12	$61.20_{\pm 0.19}$	$60.34_{\pm 0.17}$
5-way-5-shot	mini-imagenet	DeepSet	res12	$74.85_{\pm 0.30}$	$74.82_{\pm 0.33}$
5-way-1-shot	cifar-fs	MAML	cnn4	$49.10_{\pm 1.35}$	$48.79_{\pm 1.45}$
5-way-5-shot	cifar-fs	MAML	cnn4	$62.70_{\pm 0.95}$	$62.17_{\pm 1.25}$

8.1. Experiment Set-ups in CV

Datasets: We use the mini-imagenet dataset sampled from imagenet dataset (Deng et al., 2009). This dataset consists of 84x84 colored images. There are 100 classes, and 64 are for training, 16 for validation and 20 for testing. There are 600 data points for each class. Besides, we also used the cifar-fs (Bertinetto et al., 2019) sampled from cifar-100 dataset (Krizhevsky et al., 2009), which consists of size 32x32 colored images. We follow the splits for this dataset according to (Bertinetto et al., 2019).

Baseline Methods: We use the standard Relation Network (Sung et al., 2018), Prototypical Network (Snell et al., 2017), FEAT and its variant DeepSet (Ye et al., 2020) and MAML (Long, 2018) with random episodic sampling our baseline models. All models have a randomly-initialized resnet-12 (Ye et al., 2020) backbone, except the relation network, which has a light-weighted 4-layer CNN backbone. We compare results obtained from curriculum learning against results obtained from random sampling for each of these models, which we refer to as "standard episodic training".

Implementation Details: When implementing the *Sampler*, we first preprocess the dataset by converting all the images to 512-vectors via image to vector techniques, with a resnet-18 (He et al., 2016) backbone pretrained on the Imagenet dataset (Russakovsky et al., 2015; pyp). After that, we build a sorted adjacency list for each image vector. Each image vector (represented by an identifier) is mapped to a list of image identifiers within the same class. The list of neighbour images are sorted based on cosine similarity between the key-neighbour pair. The most similar neighbours are sorted to the front of the list. For approximating support set difficulty, we randomly sample a list of support sets and sort them according to their similarity levels. We determine the "difficulty" according to two parts, one is the similarity degree between the query and the target, and another one is the intra-support-set similarity degree. The difficulty assignment is rank-based. We first create a set of support set combinations and sort them based on the intra-support-set difference. After that, for sampling an episode, we first sample a support set. Suppose that the difficulty

score is 2 out of 5, then we will randomly sample a support set among the first $\frac{2}{5}$ of the sorted list. Given the support set, we proceed to sample the queried data point based on a similar method; we will also sample the query set from first $\frac{2}{5}$ of the aforementioned sorted adjacency list, given each support set image. As such, we can approximately fulfill the concept of difficulty designed in our paper. When handling multiple shots, we first calculate the average embeddings of the multiple images of the same class in the support set, and then selected the query according to the average embedding.

Training Details: For both 5-way-1-shot training and 5-way-5-shot training of the relation network, we use a learning rate of 0.001 throughout the training. For 5-way-1-shot training, we set the number of queries per episode to 15. For 5-way-5-shot training, the number of queries per episode is 10. For training of meta-learning models, we follow training hyper parameters specified in (Long, 2018) and (Ye et al., 2020). In our experiments, we define the total number of difficulty levels as 5, with the last 2 levels representing the harder combinations, or the hard tasks. The initialization seeds we use for measuring convergence are 0, 42 and 222. For measuring meta-learning performance, we use seed 0. Lastly, we increase the difficulty level after every "interval" number of episodes, where "interval" is the only hyper parameter to be tuned. The "interval" value we choose is between 1,000 and 20,000 episodes.

8.2. Experiment Set-ups for NLP

Datasets: For NLP experiment, we use the Amazon Review Sentiment Classification (ARSC) (Yu et al., 2018), which consists of either positive or negative English reviews for 23 types of products on Amazon. We follow the train-test split specified in (Geng et al., 2019). For each product domain, there are three different binary classification tasks. These buckets form 23 x 3 = 69 tasks in total. Four categories, namely Books, DVD, Electronics and Kitchen, are selected as test domains, 12 other categories are selected as validation domains and 57 are selected as the train domains (Geng et al., 2019).

Implementation Details: The experiment setup for NLP is similar as that for CV tasks, except in a few aspects. In this NLP experiment, we use the Induction Network as the training algorithm and baseline (Geng et al., 2019). When preprocessing the data, we consider each sentence as one data point. Due to the large amount of data points, we only randomly sample 30 sentences within the same category. When measuring the similarity between data points, we transform the original sentence texts to vectors using the MiniLM-L3-v2 (Reimers and Gurevych, 2019) model, before measuring the cosine similarity between the vectors.

9. Further Discussions

This section discusses the ablation variants of the aforementioned implementations and some other observations.

9.1. Choice of Scheduler

Apart from the baby-step scheduler we employ for SimCL, we also implement a one-pass scheduler and a combination of one-pass and baby-step scheduler called "half normal scheduler". We perform experiments on these schedulers and compare their performance.

One-Pass Scheduler: Just like baby-step scheduler, one-pass scheduler divides samples to bins of different difficulty levels. One-pass scheduler starts with the lowest difficulty bin, choosing increasingly difficult bin over the training. Unlike baby-step scheduler that expands the training range, one-pass scheduler only focuses on one bin at a time (Soviany et al., 2021). Based on our experiments, this scheduler does not work well even when compared to standard episodic sampling. This can be explained by catastrophic forgetting (Lopez-Paz and Ranzato, 2017; Laenen and Bertinetto, 2021; Tian et al., 2020).

Half-Normal Scheduler: In this work, we invent a half normal scheduler as illustrated in figure 7, which increases the overall difficulty level for training while reviewing the easier samples. In this scheduling algorithm, the sampler samples from the dataset with a "half normal distribution". The mean of the normal distribution is the current difficulty limit, and the standard deviation is a hyperparameter. We only take the difficulty level with values below the mean, or the difficulty level limit. We gradually increase the difficulty limit along the training procedure, placing a greater weight on sampling the current-highest-possible difficulty level. This procedure is illustrated in figure 7. With this design, the model can learn from harder and harder sample combinations while not forgetting the easier ones. When the difficulty level reaches the maximum, the sampler will focus on the harder combinations and occasionally revise the easier ones to prevent catastrophic forgetting. This scheduler has similar performance as the baby-step scheduler performance. The difference between a one-pass scheduler and a half-normal scheduler is that the latter revises easy examples while the former does not. The better performance of the latter also corroborated with catastrophic forgetting (Lopez-Paz and Ranzato, 2017; Laenen and Bertinetto, 2021; Tian et al., 2020).

9.2. Effect of "Oversampling" Easy Data points

Our work could result in oversampling of certain easy data points. Due to more training time allocated to easier tasks, data points which are nearer to class centroids tend to be oversampled while the outliers have less chance of being sampled.

We record the number of times each data point is used for training, and find out that the inter-class variance at the end of curriculum training is much larger when compared to standard episodic training. Let N denote the number of times each class is sampled from, K be all classes in a dataset, c be a particular class, the inter-class variance is calculated according to formula 5.

$$Var(c_i) = \frac{\sum(N(c_i) - \frac{\sum_{c_j \in K} N(c_j)}{|K|})}{|K| - 1} \tag{5}$$

When the number of given labelled samples is large, the variance also gets larger, resulting in data imbalance and harming the model training in terms of both convergence and generalization (Ochal et al., 2021). As such, we try adding in a "variance control unit"

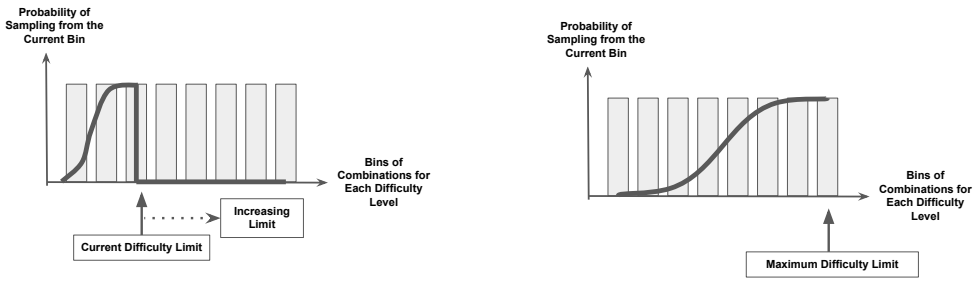

Figure 7: Illustrates the idea of how the sampler samples data with increasing difficulty. Combinations of support set and query data points of different difficulty levels are divided into buckets. A sampler starts from an easy difficulty level and increases the difficulty value until the maximum value is hit. The thick black line represents the approximate probability distribution of sampling. The higher the height of the black line, the higher the probability that the sampler will sample from that particular bin.

for curriculum learning. The reduction in variance is implemented through sampling with weighted probability, which placed a greater weight on less-sampled classes. The weight is calculated according to formula 6. In the formula, $w(x_i)$ is the probability of sampling a particular data point x_i. The term ϵ is a small positive constant to prevent division by zero. Through the weighted-sampling, the final inter-class variance is reduced by almost half.

$$w(x_i) \propto 1/(N(x_i) + \epsilon) \tag{6}$$

9.3. Extra Computation Cost of Curriculum Learning

While curriculum learning in itself just rearranges data without introducing extra computation cost, one needs extra computation for difficulty measurement. However, we believe that the extra computation cost can be negligible using our previously proposed definition of difficulty. In our implementation, we build a buffer to store support sets and query sets as a list of image name combinations. To sample and rank 100,000 episodes of training combinations according to their difficulty, one only needs less than one minute on a standard CPU. Our CPU model is Intel(R) Xeon(R) Silver 4310 CPU @ 2.10GHz, and it takes 47.57 seconds to finish all the 5-way-5-shot sampling, and 35.36 seconds for 5-way-1-shot, which is almost all the extra computation cost involved with curriculum learning.

Due to the approximation nature of our proposed difficulty measurement, one can use low-precision hardwares with, for example, 16-bit or even 8-bit adders or multipliers to cut down the computation cost. In our calculation of similarity scores, we use 16-bit representations for floating numbers.

10. Conclusion

To conclude, this work looks into the concept of difficulty behind FSL. It firstly demonstrates the existence of varying difficulty levels during FSL training, proposes an intuitive difficulty measurement and investigates how one can make use of the difficulty. For example, we may cut down unnecessarily difficult episodes to save computation resources, utilize easy episodes to achieve higher testing accuracy scores within limited training time and design a curriculum to speed up the full training procedure or achieve a better performance among meta-learning models. There may be other potentially useful applications leveraging the concept of difficulty. For example, one can consider incremental difficulty for data augmentation during FSL training. While simple and overlooked, the concept of difficulty can enhance FSL training in different ways.

References

Img2vec-pytorch. URL https://pypi.org/project/img2vec-pytorch/.

Sébastien Arnold, Guneet Dhillon, Avinash Ravichandran, and Stefano Soatto. Uniform sampling over episode difficulty. *Advances in Neural Information Processing Systems*, 34, 2021.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

Luca Bertinetto, Joao F. Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyxnZh0ct7.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

Martin Ford. *Architects of Intelligence: The truth about AI from the people building it*. Packt Publishing Ltd, 2018.

Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. Induction networks for few-shot text classification. *arXiv preprint arXiv:1902.10482*, 2019.

Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3018–3027, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Steinar Laenen and Luca Bertinetto. On episodes, prototypical networks, and few-shot learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

Liangqu Long. Maml-pytorch implementation. https://github.com/dragen1860/MAML-Pytorch, 2018.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2018a. URL https://arxiv.org/abs/1802.03426.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018b.

Erik G Miller, Nicholas E Matsakis, and Paul A Viola. Learning from one example through shared densities on transforms. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 1, pages 464–471. IEEE, 2000.

Mateusz Ochal, Massimiliano Patacchiola, Amos Storkey, Jose Vazquez, and Sen Wang. Few-shot learning with class imbalance, 2021. URL https://arxiv.org/abs/2101.02523.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL http://arxiv.org/abs/1908.10084.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey. *arXiv preprint arXiv:2101.10382*, 2021.

Valentin I Spitkovsky, Hiyan Alshawi, and Dan Jurafsky. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759, 2010.

Emmanouil Stergiadis, Priyanka Agrawal, and Oliver Squire. Curriculum meta-learning for few-shot classification. *arXiv preprint arXiv:2112.02913*, 2021.

Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 403–412, 2019.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018.

Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *European Conference on Computer Vision*, pages 266–282. Springer, 2020.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8808–8817, 2020.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. Diverse few-shot text classification with multiple metrics. *arXiv preprint arXiv:1805.07513*, 2018.

Ji Zhang, Jingkuan Song, Yazhou Yao, and Lianli Gao. Curriculum-based meta-learning. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 1838–1846, 2021.

Donghao Zhou, Zheng Yan, Yulong Fu, and Zhen Yao. A survey on network data collection. *Journal of Network and Computer Applications*, 116:9–23, 2018.

Linjun Zhou, Peng Cui, Xu Jia, Shiqiang Yang, and Qi Tian. Learning to select base classes for few-shot classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4624–4633, 2020.