



Few Shot Learning

Jiapeng Guo
Mingkang Hui

Problem Statement

Few-shot classification aims to learn a classifier to recognize unseen classes during training with limited labeled examples.
It involves pretrain and finetune.

Goal:

We want to see how much effort we made on pretrain could lead to a good result.

In other words, what is the relationship between the amount of pretrain data and downstream performances.

See how the depth (complexity) of model could affect the results.

Support set



N=3

Query set



Solution & Value



We test on several kinds of baseline and models to see the relationship between the 'width' and 'depth' of the process of pertaining and compare the final results, based on loss and accuracy.

Value:

This could give us more insights about how the width and depth could affect the results.

And the result may help us to save time on pretraining.

Problem motivation



Current initial model based algorithms of few-shots learning require massive pretraining and data, these are all very expensive.

Hard to get data(even for pretraining)
Training takes time and resources

So we want to see the impact of reducing effort on training.

To see relationship between model complexity, amount of training and final results.

Background Work



The framework is based on a survey, [A Closer Look at Few-shot Classification](#).
And it's related repo.

For Initialization based methods, which tackle the few-shot learning problem by “learning to fine-tune”. Finn et al. (2017; 2018); Nichol & Schulman (2018); Rusu et al. (2019).

For Distance metric learning based methods, it address the few-shot classification problem by “learning to compare”. The intuition is that if a model can determine the similarity of two images, it can classify an unseen input image with the labeled instances Koch et al. (2015).

Background Work



Domain adaptation techniques aim to reduce the domain shifts between source and target domain Pan et al. (2010); Ganin & Lempitsky (2015), as well as novel tasks in a different domain Hsu et al. (2018).

Hallucination based methods directly deal with data deficiency by “learning to augment”. This class of methods learns a generator from data in the base classes and use the learned generator to hallucinate new novel class data for data augmentation. These generators either transfer variance in base class data to novel classes Hariharan & Girshick (2017), or use GAN models Antoniou et al. (2018) to transfer the style.

Technical challenges



Three aspects:

System

Data

Algorithm

System: Training is expensive but the resource we have is poor.

We have 3 network: ResNet10, ResNet18, ResNet34

And two baseline: Baseline and Baseline ++

So we need to train $3 * 2 = 6$ models.

Technical challenges



Data:

Data is limited. All we got is minilmagenet and CUB dataset. The number of classes and pictures are limited and the domains are also limited.


Algorithm:

We only tested on Baseline and Baseline++, and has not test on other algorithms to see the relationship.



Approach

- We test three models on two different datasets.
- Models include ResNet 10, ResNet 18, ResNet 34.
- We trained models on two approaches: baseline and baseline++
- Two datasets are the mini-ImageNet dataset and CUB-200-2011 dataset.



For this project, we compare model performance under different settings.

First is generic object recognition. For object recognition, we use the mini-ImageNet dataset commonly used in evaluating few-shot classification algorithms. Second is cross-domain adaptation. For the cross-domain scenario (mini-ImageNet \rightarrow CUB), we use mini-ImageNet as our base class and the 50 validation and 50 novel class from CUB. Evaluating the cross-domain scenario allows us to understand the effects of domain shifts to existing few-shot classification approaches.

Baseline and Baseline++ few-shot classification methods.

Both the baseline and baseline++ method train a feature extractor f_θ and classifier $C(\cdot|W_b)$ with base class data in the training stage. In the fine-tuning stage, we fix the network parameters θ in the feature extractor f_θ and train a new classifier $C(\cdot|W_n)$ with the given labeled examples in novel classes. The baseline++ method differs from the baseline model in the use of cosine distances between the input feature and the weight vector for each class that aims to reduce intra-class variations.

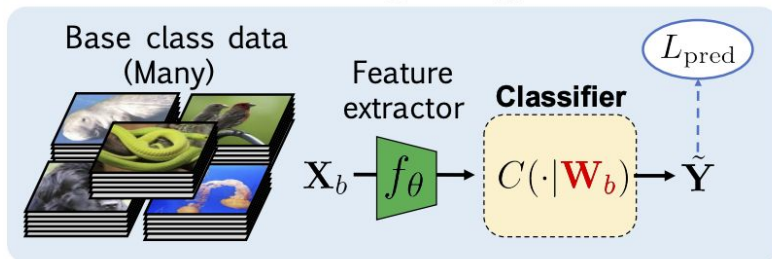


In the training stage for the Baseline and the Baseline++ methods, we train 70 epochs with a batch size of 16. We saved models at 30, 50, and 70 epochs for comparison.

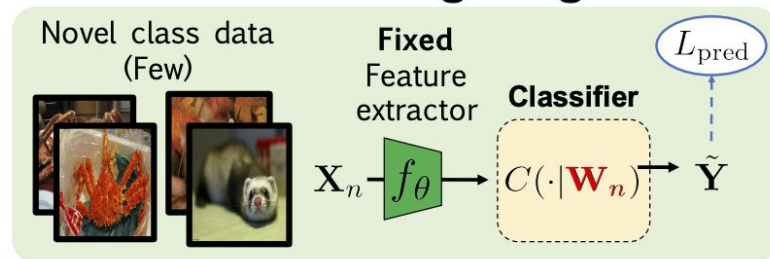
In the fine-tuning stage, we average the results over 600 experiments. In each experiment, we randomly sample 5 classes from novel classes, and in each class, we also pick 5 instances for the support set and 16 for the query set. For Baseline and Baseline++, we use the entire support set to train a new classifier for 100 iterations with a batch size of 4.

All methods are trained from scratch and use the Adam optimizer with initial learning rate 10^{-3} . We apply standard data augmentation including random crop, left-right flip, and color jitter in both the training or meta-training stage. For Baseline++, we multiply the cosine similarity by a class-wise learnable scalar to adjust original value range $[-1, 1]$ to be more appropriate for subsequent softmax layer.

Training stage

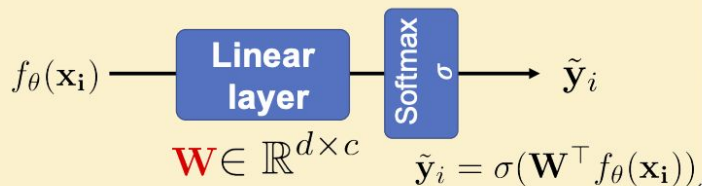


Fine-tuning stage

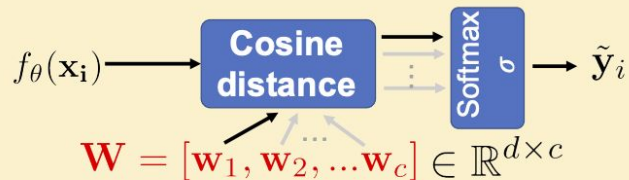


Classifier $C(\cdot | \mathbf{W})$

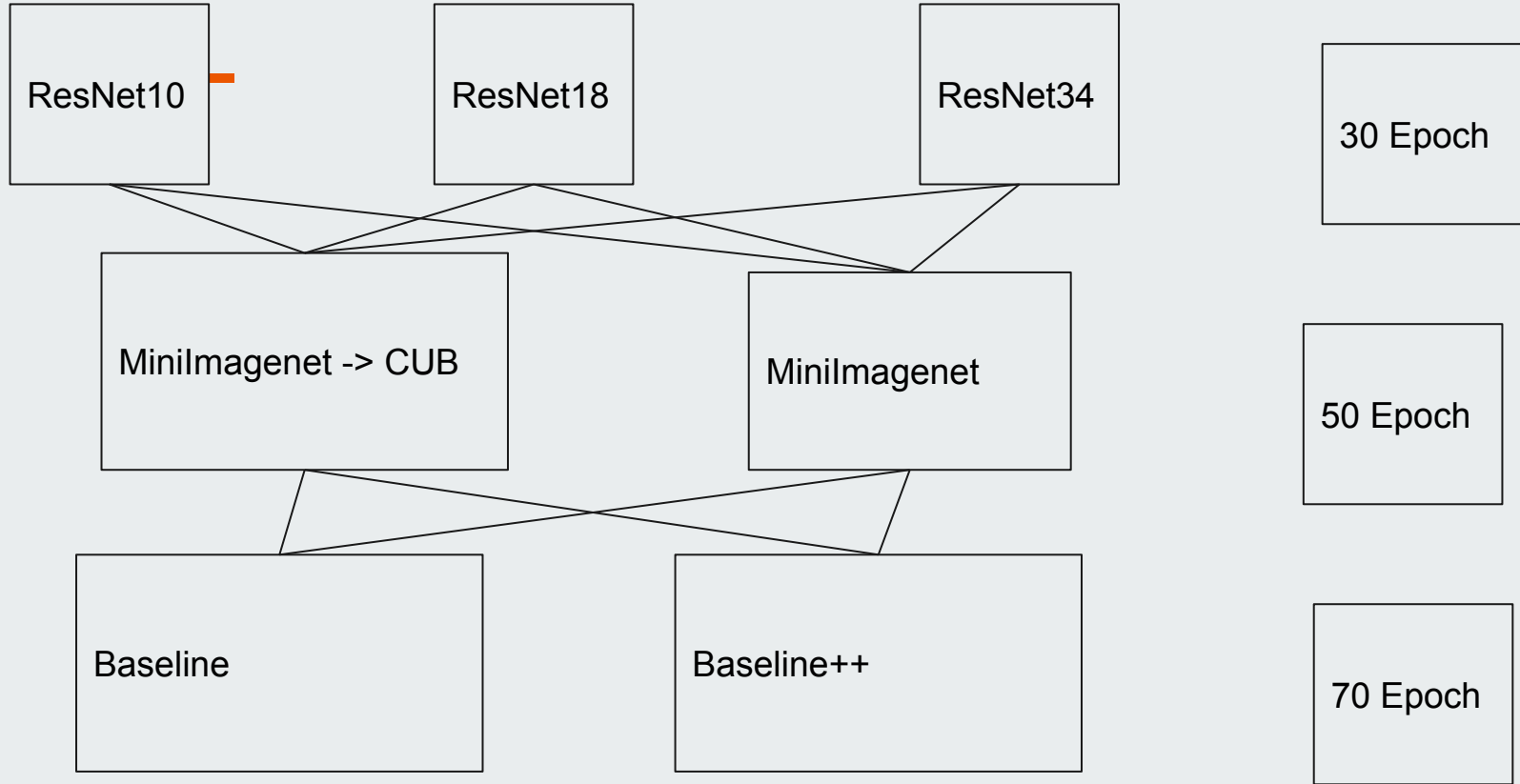
Baseline



Baseline++



Solution diagram



Implementation detail

We initiated a virtual machine on GCP. Below is details of this VM:

Machine Type: n1-standard-2

CPU platform: Intel Haswell

GPUs: 1 x NVIDIA Tesla K80

Storage: 500 GB

Dataset: MinilImagenet, CUB

Framework: PyTorch

Cloud Storage: GCP storage

Limitations: Not enough amount of GPUs for faster training

The mini-ImageNet dataset consists of a subset of 100 classes from the ImageNet dataset Deng et al. (2009) and contains 600 images for each class. The dataset was first proposed by Vinyals et al. (2016), but recent works use the follow-up setting provided by Ravi & Larochelle (2017), which is composed of randomly selected 64 base, 16 validation, and 20 novel classes.

The CUB dataset contains 200 classes and 11,788 images in total. Following the evaluation protocol of Hilliard et al. (2018), we randomly split the dataset into 100 base, 50 validation, and 50 novel classes.



Demo

Experiment design flow:

We will show testing accuracy for all scenarios. Also, we will show training plots, i.e. epoch versus loss.



Experimental evaluation

Testing Accuracy on Minilmagenet



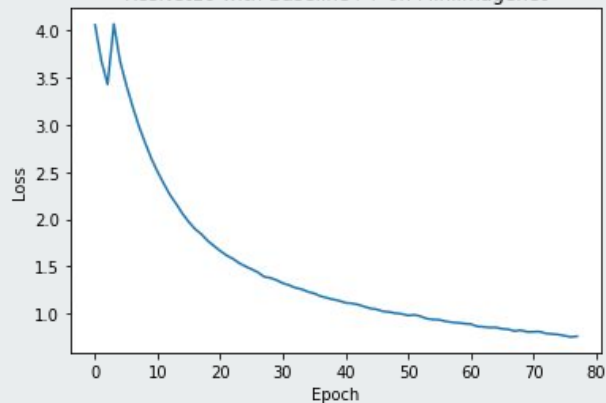
	Baseline	Baseline++
ResNet10-30	73.82% +- 0.63%	72.92% +- 0.66%
ResNet10-50	75.48% +- 0.63%	73.02% +- 0.64%
ResNet10-70	75.22% +- 0.61%	73.57% +- 0.64%
ResNet18-30	74.48% +- 0.64%	73.21% +- 0.67%
ResNet18-50	73.84% +- 0.65%	74.20% +- 0.67%
ResNet18-70	74.86% +- 0.66%	74.44% +- 0.70%
ResNet34-30	72.92% +- 0.67%	71.66% +- 0.65%
ResNet34-50	73.21% +- 0.62%	73.35% +- 0.65%
ResNet34-70	74.35% +- 0.63%	73.22% +- 0.65%

Testing Accuracy on cross domain, Minilmagent -> CUB

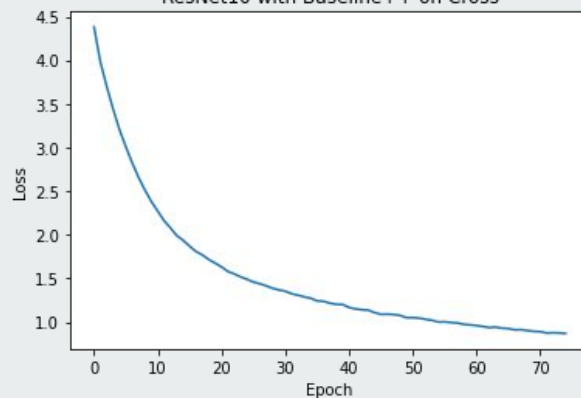
	Baseline	Baseline++
ResNet10-30	62.35% +- 0.74%	60.62% +- 0.75%
ResNet10-50	65.07% +- 0.74%	60.80% +- 0.71%
ResNet10-70	64.80% +- 0.73%	61.29% +- 0.74%
ResNet18-30	64.67% +- 0.75%	62.46% +- 0.78%
ResNet18-50	65.90% +- 0.70%	63.51% +- 0.77%
ResNet18-70	66.38% +- 0.71%	63.42% +- 0.75%

Training Plots

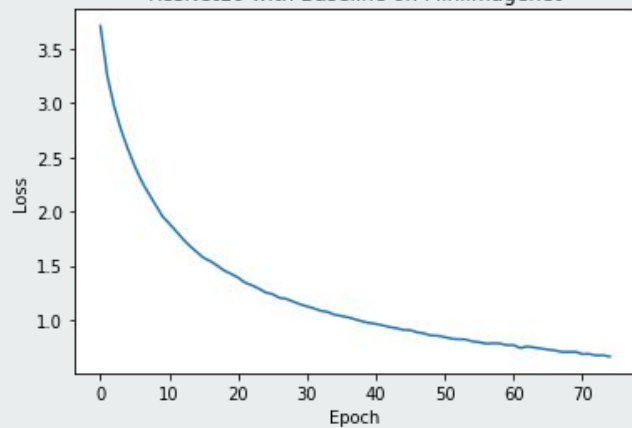
ResNet10 with Baseline++ on Minilmagenet



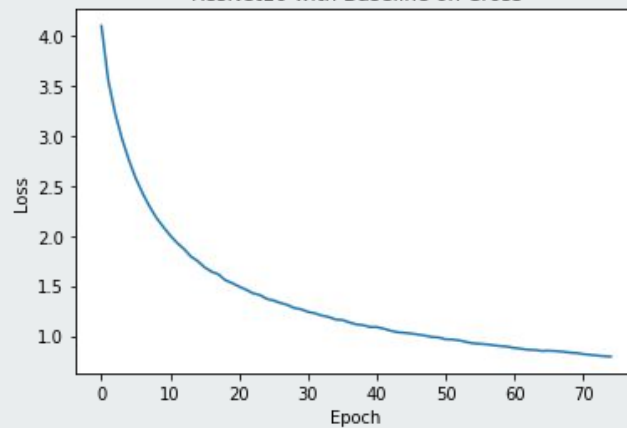
ResNet10 with Baseline++ on Cross



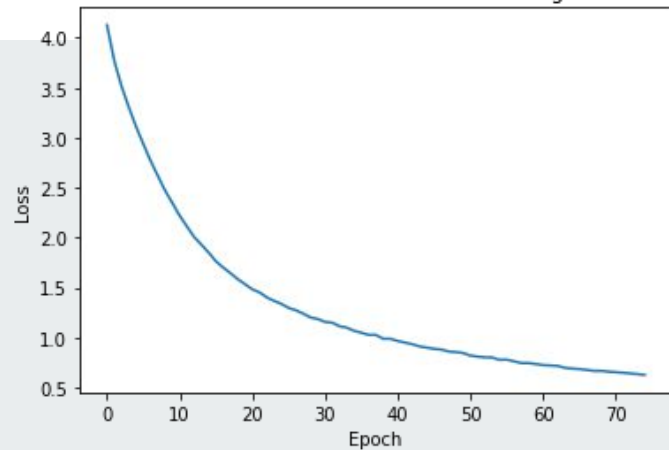
ResNet10 with Baseline on Minilmagenet



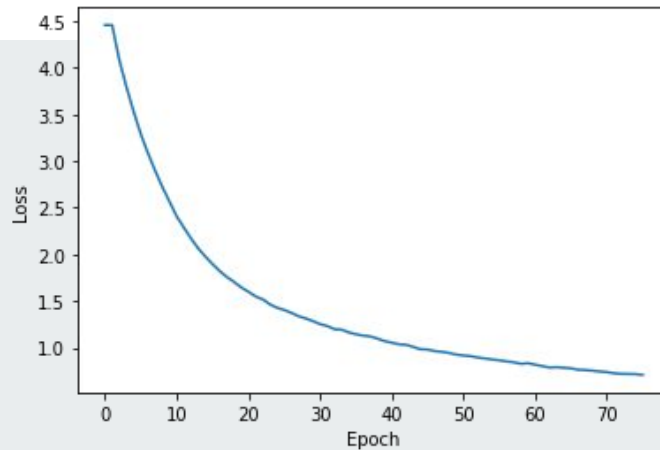
ResNet10 with Baseline on Cross



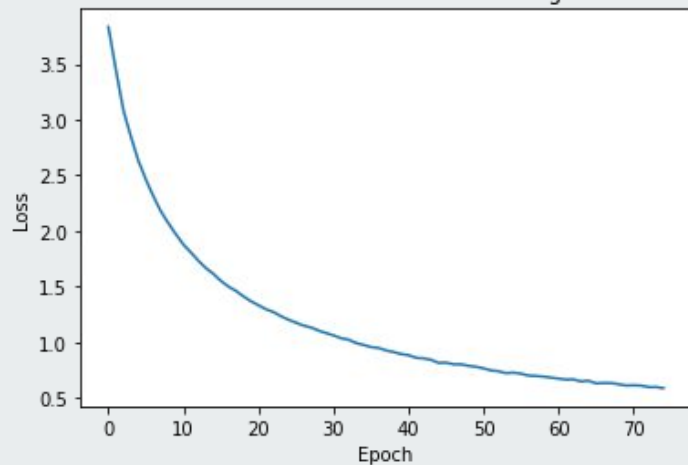
ResNet18 with Baseline++ on Minilmagenet



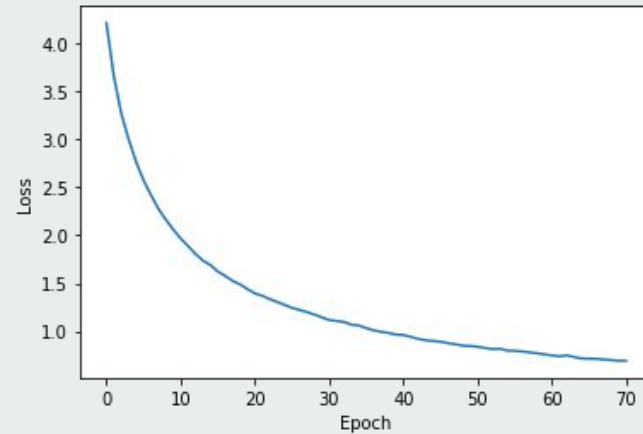
ResNet18 with Baseline++ on Cross



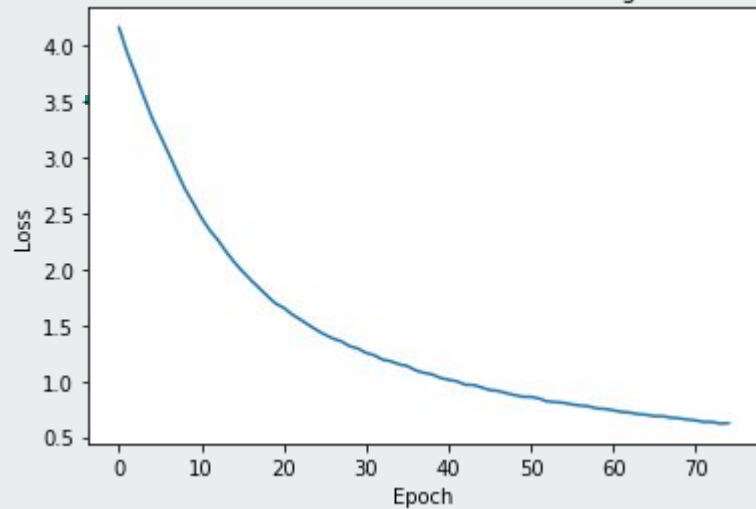
ResNet18 with Baseline on Minilmagenet



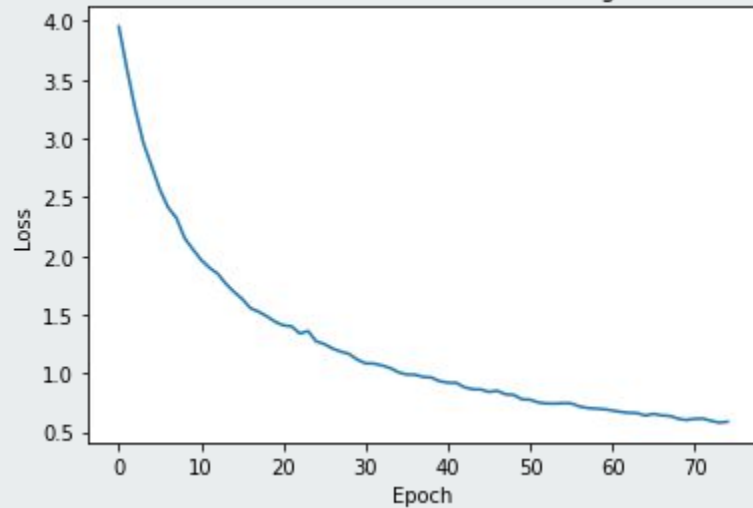
ResNet18 with Baseline on Cross



ResNet34 with Baseline++ on Minilmagenet

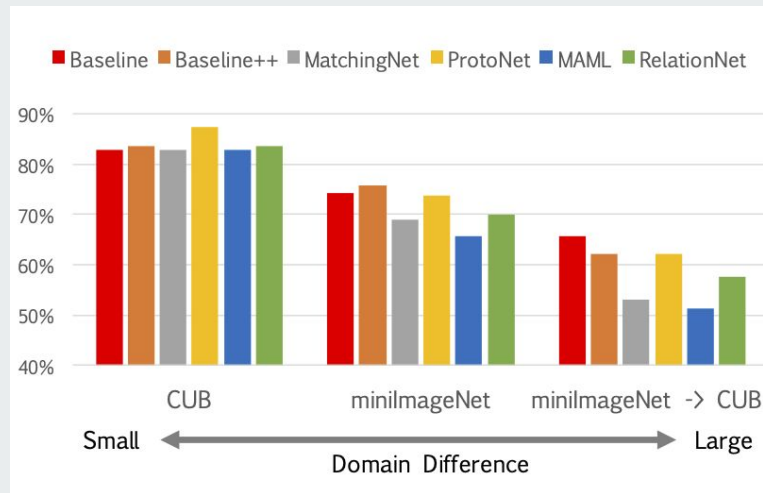



ResNet34 with Baseline on Minilmagenet



Conclusion

- Baseline++ does not improve performance in general. Which is consistent with the result for cross domain setting. But baseline++ shows improvement in the paper. We think it is due to lack of enough training epochs. Paper trained 400 epochs.



- 
- As domain difference increases, model performance drops which is expected.
 - For cross domain testing, model complexity is positively correlated with model performance. However, it is not the case for minilmagenet testing.
 - In our experiments, training 20 or 40 more epochs seems not contribute to a better performance. However, interestingly, our results are very close to the results from paper, which authors trained 400 epochs.
 - We can conclude that even training just 70 epochs, we can still achieve relatively good results. It means that we may save labor on the pre-train stage for few-show learning.