Few Shot Learning Via Learning to Learn

Ankish Bansal
ankishb@iitk.ac.in
Roll No:17204004

Sidharth Singla
ssid@iitk.ac.in
Roll No:17211404

Sirshendu Mandal
sirshend@iitk.ac.in
Roll No:160694

1 Problem Description

In this project, we are dealing with classification task using Meta learning approach. As in tradition setting, Neural Network need millions of data samples to learn any distribution or task. It is very expensive to collect million of examples. Whereas, in few shot learning, we deal with small amount of data to train the network. In Few-Shot problem setting, the goal is to develop human like ability, as we are pretty good at learning or recognising faces even after seeing once. As in medical field, collecting the million of image of disease or bacteria is very challenging and expensive. In such field, develping human like ability can be very helpful.

The traditional machine learning models usually deal with complete problem as such. But in few shot learning, main problem is broken into sub problems or tasks. This, now becomes similar to multi-task problem setting, where group of tasks are somewhat related. So we can update the parameters of the other tasks by using the training data point being used for a different task. In a way, we have a bunch of tasks, each with their own parameters , and the data used for a particular task , acts as metadata for subsequent test instances. In this way ,we try to achieve our purpose using as less data as possible. Another approac is transfer learning, where the experience of trained model on large dataset is transferred to another network, which are dealing with similar problem with fewer example. In this setting, fine tuning is used to train the pretrained network.

1.1 Relation between Few Shot learning approaches with Meta Learning

- Meta Learning is closely related to **Multi-task learning** as the learning agent receives information about several tasks.
- Meta Learning is different from the **Transfer learning**, as it does not involve designated source and target tasks.
- In meta Learning, Meta data helps in adding past learning periods experience for future learning, even the learning happens in different domains.

1.2 Mathematical Difference between Traditional and Few Shot Learning approaches

1.2.1 Notation used

As shown in Figure 1, there is meta-training and meta-testing dataset, where meta-testing dataset is completely hidden while training. In each dataset, we have many task or episode, samples from the dataset. Each episode/task has its own training data D_{train} and D_{test}, where our objective is to classify D_{test} by showing D_{train}.

Intution behind this task dataset is that using few shots of few classes, we classify the testing samples from the same classes which is also shown to network at testing time. This will be more clarrified with the following example and mathematical expression.

1.3 Example to explain the data-set

Suppose for one task, we choose n classes and for each class, we choose k samples, so we have nk samples for one episode. From these nk examples, we create D_{train} and D_{test} in one episode. So testing of an example is limited only to those n classes at a time.

1.4 Mathematical Formulation of Traditional and Few-Shot Supervised

- Traditional Supervised Problem Setting

$$Input : X, Output : y \tag{1}$$
$$y = f(X; \theta) \tag{2}$$

- Few-Shot Supervised Problem Setting

$$Input : D^{train}, x^{test}, Output : y^{test} \tag{3}$$
$$y^{test} = f(D^{train}, x^{test}; \theta) \tag{4}$$

where y^{test} represent the prediction at x^{test} given D^{train} for any one task and f is the model. Thest notation will be cleared from the following picture[1].

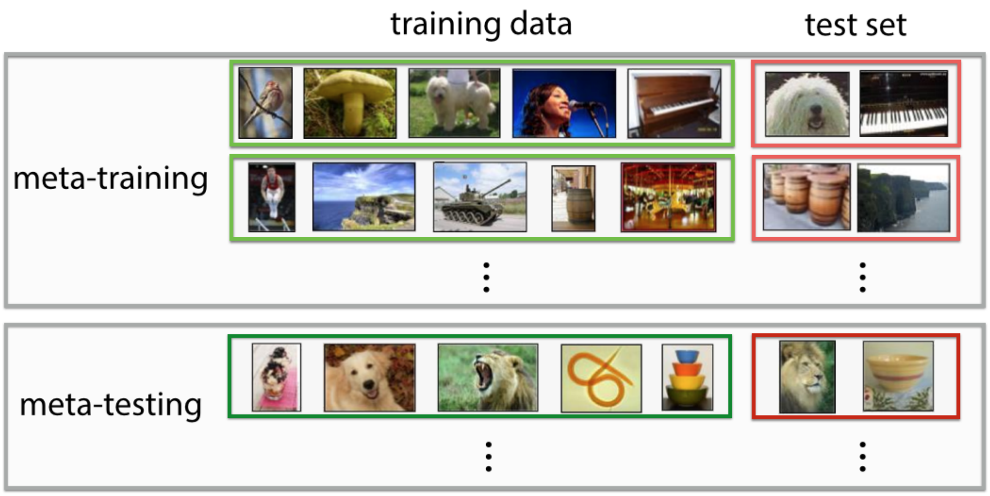

Figure 1: Illustration of data-set in few shot learning (*source: Lecture 27)

2 Literature Review

There has been extensive research on few shot learning and meta learning in past few years. We have studied some of the related approaches to MAML(Model Agnostic Meta Learning) [Finn et al.[2017]], which evolves the learning to learn method. There are three main approaches in this area, that has been used. First is metric based method [Vinyals et al.[2016]] and [Snell et al.[2017]], where the author has used distance based method to compare the embedding of the testing image with the embedding of the given few training images. Second approach is recurrent based, where sequentially data of each task is used to train the network. And last, there is optimizer based approaches [Finn et al.[2017]],[Ravi et al.(Meta-LSTM)], where the idea is to learn optimizer as well, along with model. In [Ravi et al.(Meta-LSTM)], author has modeled LSTM cell[Hochreiter et al.[1997]] as optimizer. And in [Finn et al.[2017]], author is presenting gradient based approach to learn the optimizer and model both. Let us raise a question, why this is good idea to learn optimizer for each task separately? The answer to that is, treat each task as a one problem which network is dedicated to solve. In that sense, learning different optimizer for each task give the extra advantage. And we have extensively studied on this approach and modified the previous work, which is described in the following section.

3 Description of prior work on the problem

As discussed, MAML is a Gradient based Approach. The advantage of this approach is gradient descent algorithm, which always find an optima(local or global). And also cheap over Meta-LSTM based approach.

The key points of MAML are:

- MAML is finding a good initialization of model's parameters for several tasks.
- Good initialization of parameters means that it can achieve good performance on several tasks with small number of gradient steps.

3.1 Higher level idea behind MAML

There are two learner, **Meta-Learner** and **Main learner**

- Meta Learner quickly adapts the parameter for current task by taking few gradients.
- Main learner predicts the model output using the parameter initialization given by meta-learner and globally update the parameter w.r.t. all task.

$$y^{test} = \mathbf{f}(x^{test}; \mathbf{g}(D^{train}, \theta)) \tag{5}$$

where **g represent the meta-learner** and **f represent the main learner**.

The last equation also represent the difference between few shot and meta learning. There is no **g** in few shot learning objective, which is meta learner or optimizer.

3.2 Objective function of MAML

First step is to adapt the parameter of current task, that is re-initialization of model for the current task. This is also called Parameter Adaptation or **Fine-Tuning** step, and the parameter is computed as

$$\phi_i = \theta - \alpha \nabla_\theta L^i_{train}(\theta) \tag{6}$$

where ϕ_i represent the parameter initialization for task i and ∇ represent the gradient step in the direction of the current task distribution. This adaptation can be done in iterative manner, by updating the ϕ_i k times in the same way. This is all about meta learner.

Now we look at the Objective function of main learner, which globally update the model by computing the loss function on the test dataset for each task, using the initialization for each task done by the meta learner.

$$min_\theta \sum_{task_i} L^i_{test}(\theta - \alpha \nabla_\theta L^i_{train}(\theta)) \tag{7}$$

After this step, the entire model is updated by parameter θ.

3.2.1 Difference between Few shot Learning and Meta Learning

In few shot Learning, network prediction or output is as follows

$$Input : D^{train}, x^{test}, Output : y^{test} \tag{8}$$

$$y^{test} = f(D^{train}, x^{test}; \theta) \tag{9}$$

Whereas in meta Learning, network prediction or output is as follows

$$Input : D^{train}, x^{test}, Output : y^{test} \tag{10}$$

$$y^{test} = f(x^{test}; \mathbf{g}(D^{train}, \theta)) \tag{11}$$

where **g represent the meta-learner** and **f represent the main learner**.

NOTE: There is no g in few shot, that is meta learner.

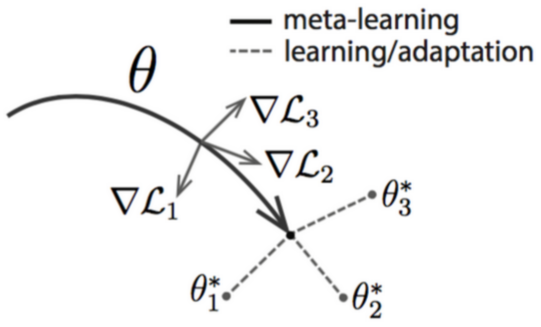

Figure 2: Illustration of MAML parameters updation direction (*source: [3])

3.3 Pros/Cons of MAML

Advantage of MAML are

- MAML is gradient based approach, which always find optima(local/global)
- For a sufficient deep f, MAML can universally approximate any data-set **Finn & Levine** 18
- There is no overfitting (it happened in few shot setting)
- MAML problem setting can be formulate in Bayesian setting.

Weakness of MAML are

- Expensive (in term of computing Power), because of second order computations. Many methods have been proposed to make it approximate it to first order method, (**REPTILE**[Nichol et al.[2018]]).
- Defining right **task distribution** is not easy. **We have done some modification to tackle this problem.**

4 Novelty of our work as compared to prior work

As described, the right task distribution is challenging and weakness of MAML. The intution behind this weakness is that if we have only two tasks and both are completely different from each other. And after optimizing, let's suppose, the first task gives the direction of the parameters at 0 degree and second task gives the direction at 180 degree. Then, this model will converge in the direction of the average of both task, which will be 90 degree, which is not good direction for the both tasks.

So to tackle this problem, we comes up with a solution to find the weighted average of parameter, by using LSTM cell [Hochreiter et al.[1997]]. So, LSTM have memory of previous tasks stored in cell state c_{t-1} and it also compute he features from the current task. Now, output gate will decide how much it has to include from the previous task and how much from the current task. **In other words, it will helps to find the similarity/relation between the current task and previous task, which helps in finding the good direction for the parameters update.** The implementation details are elaborated in section 6.

5 Tools/softwares used

We used Pytorch library in Python for the implementation.

6 Experiments

We performed classification task for **MiniImagenet**, **Caltech-UCSD Birds 200**, and **Caltech 256 Objects** datasets with 5-way 1-shot learning.

Here 5-way means data from 5 different class of images is given to the model, and 1-shot signifies 1 image from each class is used during training phases in each episodes.The model makes the prediction on 15 images from each class during testing in each episodes in our experiment.

MiniImagenet:The MiniImagenet consisting of 60,000 colour images with 100 classes, each having 600 examples. The data is split into 64 training classes, 12 validation classes, and 24 test classes.

Caltech-UCSD Birds 200: CUB-200 is an image dataset with photos of 200 bird species. It contains 200 categories, one corresponding to each species of bird. Total number of images in dataset is 6,033. We split it into 150 training and 50 testing classes.

Caltech 256 Objects: It is a collection of 30607 images of various object, split into 256 object categories. Each category contains atleast 80 images. It was collected by choosing a set of object categories, downloading examples from Google Images and then manually screening out all images that did not fit the category. We split it into 180 training and 76 testing classes.

For this task, we used the PyTorch implementation of the Basic model using [[8]], resolved some issues in the code and modified it to perform our experiments.

6.1 Removal of Custom weights and initializations

Since we wanted to experiment with the existing network architecture, we modified the code to remove custom initializations and definitions for different layers of the network. This gave us more freedom to easily play with the architecture. All the custom layers were replaced with default components provided with Pytorch(like conv2d, batchnorm etc.) That led to some more modifications in the main MAML algorithm since we had to extract weight differently now and the update/training procedure for the network also changed.
The Basic network architecture of our model is:

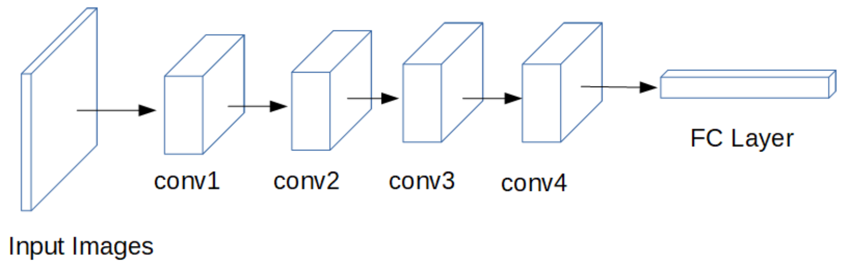

Figure 3: Basic Network Architecture

where each convolution block is defined by:

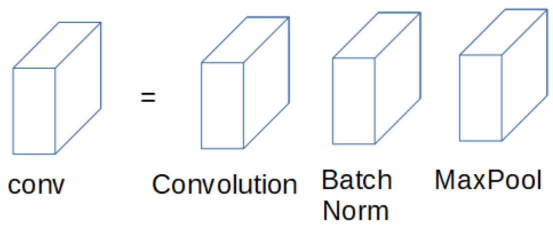

Figure 4: Convolution Block

6.2 Modification in model (By adding LSTM layer)

As explained in the section[4], we want to find the similarity of features from the previous task and current task, for this we did two experiment, one by adding LSTM in Series and other in Parallel.

6.2.1 LSTM in series

Initially, we added a LSTM layer to the network in series.

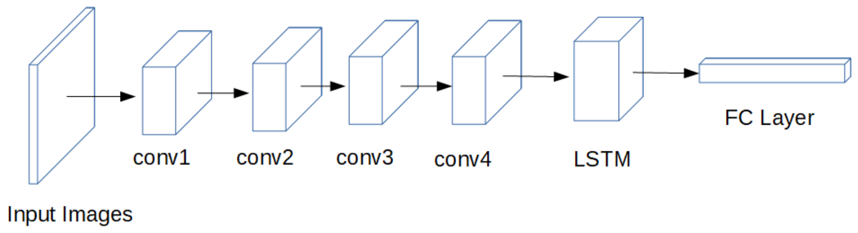

Figure 5: Model Architecture with LSTM in series

But the performance of the model didn't improve from random guess accuracy. After carefully analysis, we found that it is doing the same thing as MAML doing (average of the direction of parameters for each task)

6.2.2 LSTM in parallel

To avoid the problem faced by previous modification (explained in last section), we decide another way, by keeping the learned feature as it is and then pass those learned features to LSTM, which find the weighted features between tasks (or direction to update the parameters in right direction) and finally combine the results from both of them.

So we changed the architecture and put the LSTM layer in parallel so that the final FC layer for prediction gets the output of convolution layers as well as the relationship among images captured by the LSTM.

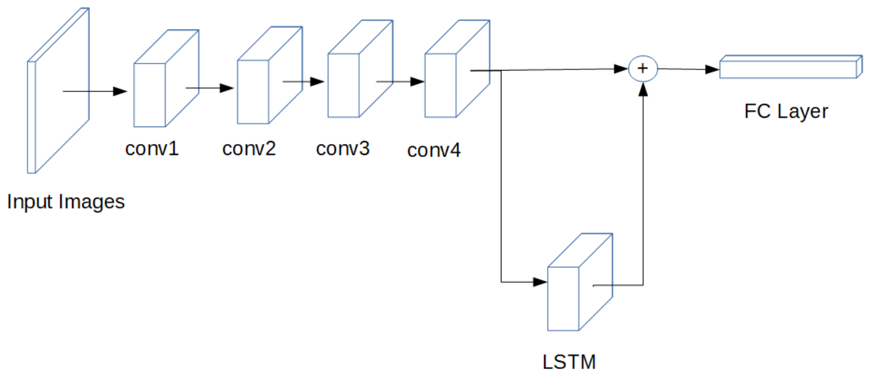

Figure 6: Model Architecture with LSTM in parallel

Our code is available [here].

7 Experimental results

Our experiments results for three dataset is shown in following table, where Basic represent the Base model's prediction and LSTM in parallel represent prediction on our second modification.

Dataset	Model	5-way 1-shot accuracy
MiniImagenet	Basic	42.55% ± 0.86%
	LSTM in Parallel	47.85% ± 0.99%
Caltech-UCSD Birds 200	Basic	43.61% ± 0.91%
	LSTM in Parallel	48.09% ± 0.99%
Caltech 256 Objects	Basic	41.93% ± 0.83%
	LSTM in Parallel	49.84% ± 0.97%

8 Things that we learned while doing the project

We learned few ways to deal with training the network from very small amount of data, using few shot learning and meta learning. We have reviewed few previous approaches on this topic(which are referred in Reference). We have also developed skills on deep learning framework(Pytorch), while implementation of Finn et al.[2017] and our ideas on top of that paper and achieve good results. We have also understand, the value of skills like patience, determination etc. also helps a lot while doing experiment.

9 Our future work

We will continue this work, as our modification helps improving the model. We will also look at the reinforcement side of this paper.

9.1 References

[1] Vinyals et al.[2016] Matching Networks for One Shot Learning

[2] Snell et al.[2017] Prototypical Networks for Few-shot Learning

[3] Finn et al.[2017] Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

[4] MAML-Official tensorflow Implementation

[5] Nichol et al.[2018] On First-Order Meta-Learning Algorithms

[6] Ravi et al.(Meta-LSTM) Optimization as a model for few-shot learning

[7] Hochreiter et al.[1997] Long short-term memory

[8] MAML PyTorch Implementation

[9] Nichol et al.[2018] On First-Order Meta-Learning Algorithms

Disclaimer
We declare that the work done in this project is solely ours. This work has not been re-used from any another course project at IITK or elsewhere, or any other project.