EE698R : Advanced Topics in Machine Learning **Semester 2021-22-II**

Meta Transfer Learning

Rohan Baijal
190714
`rbaijal@iitk.ac.in`

Sarvesh Chandra
190773
`csarvesh@iitk.ac.in`

1 Motivation

A good machine learning model often requires training with a large number of samples. Humans, in contrast, learn new concepts and skills much faster and more efficiently. Kids who have seen cats and birds only a few times can quickly tell them apart. People who know how to ride a bike are likely to discover the way to ride a motorcycle fast with little or even no demonstration. Is it possible to design a machine learning model with similar properties — learning new concepts and skills fast with a few training examples? That's essentially what meta-learning aims to solve.

2 Problem Definition

We define an episode in mathematical terms :

$$\mathcal{T}_{episode} \sim p(\mathcal{T})$$

Eg. Let's take the example of image classification where boy, girl, house, park, airplane, cat, rat, etc. are the different classes all of which have samples in the dataset. If we want to train a classifier which classifies into just 4 of these classes, then we first sample the 4 classes randomly and then sample the data points randomly from each class. This is the data $\mathcal{D}_1$ for an episode $\mathcal{T}_1$ (or task). The task here is training the classifier of 4 classes. In the next iteration we choose some different set of 4 classes (to get a dataset for the next episode or task).

Suppose in overall dataset we have many classes (eg. 20k) and there are less samples for each class. We define two sets, Training set and the Test set having 20k and 4 classes respectively which are disjoint. It is tough to use Transfer Learning in this case. Instead of learning a classifier, we are learning how to learn a classifier quickly (with small data). "Learning how to learn" is "Meta-Learning".

This seems like a very challenging task. But here are some ways in which we can learn "how to learn" :

- Learn the hyperparameters: i.e., model architecture (no. of layers, neurons per layer), learning rate, regularization weights, drop-out rate (all parameters which we are not fine tuning in the model training)
- Learn some parameters and leave some for adapting later: also used in Transfer Learning
- Learn initial weights
- Learn a model to predict some hyper-parameter (eg. η) given the current state (Weights, δw)

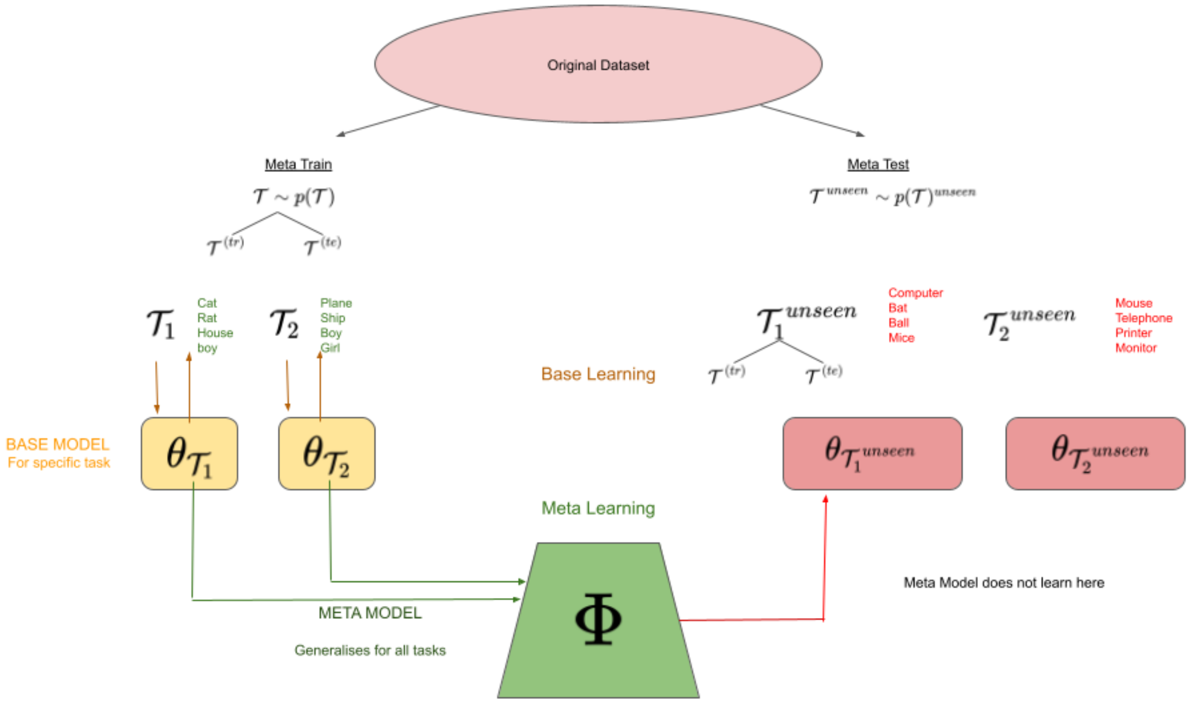


Figure 1: Overview

If the initial weights are a good estimate of the actual ones along with an optimal choice of hyper-parameters then the test set will have some data available for adaptation which will help in training (thus not completely zero shot learning). Eg. a face recognition system trained before can quickly learn to recognize some faces when its used in a new device.

3 Meta Learning Methodology

3.1 Pretraining over a large dataset

We first randomly initialize a feature extractor Θ (e.g.CONV layers in ResNets) and a classifier θ (e.g. the last FC layer in ResNets), and then optimize them by gradient descent as follows :

$$[\Theta;\theta] =: [\Theta;\theta] - \alpha\nabla\mathcal{L}_{\mathcal{D}}([\Theta;\theta])$$

where $\mathcal{L}$ denotes the loss function (eg. cross entropy loss) for the task.

$$\mathcal{L}_{\mathcal{D}}([\Theta;\theta]) = \frac{1}{|\mathcal{D}|}\sum_{(x,y^d)\in\mathcal{D}} l(f_{[\Theta;\theta]}(x), y^d)$$

The learned θ is then discarded.

3.2 Base Training Phase

Now we have $\mathcal{T} \sim p(\mathcal{T})$. Construct a $\mathcal{T}^{(tr)}$. We will now consider a new θ which is specifically for this task.

$$\theta' =: \theta - \beta\nabla_{\theta}\mathcal{L}_{\mathcal{T}^{(tr)}}([\Theta;\theta], \Phi)$$

θ is initialized from the previous task.
Φ are the parameters of the Meta Learner.

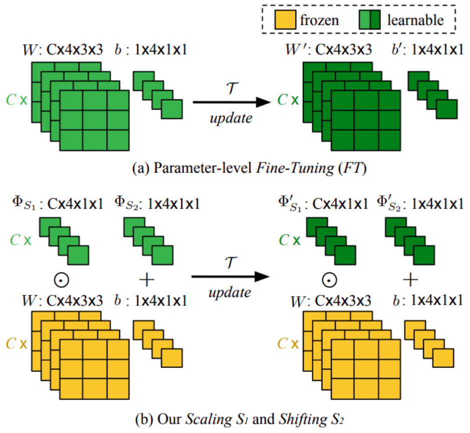
3.3 Meta Training Phase

We construct a $\mathcal{T}^{(te)}$ from $\mathcal{T}$ and use it to update the meta learner as well as the base learner.

$$\Phi =: \Phi - \gamma\nabla_{\Phi}\mathcal{L}_{\mathcal{T}^{(te)}}([\Theta;\theta'], \Phi)$$

$$\theta =: \theta - \gamma\nabla_{\theta}\mathcal{L}_{\mathcal{T}^{(te)}}([\Theta;\theta'], \Phi)$$

Note that the θ' comes from the last training episode in the Base Training phase.



(a) Parameter-level *Fine-Tuning* (*FT*)

(b) Our *Scaling* *S₁* and *Shifting* *S₂*

Figure 2: Meta Model : $w \circ \Phi_{S1} + b + \Phi_{S2}$