# Meta Transfer Learning (2/2)

Astitva Chaudhary ( 180157), Nakul Singh (18807455)

April 3, 2022

## 1 Introduction

We need to formulate an algorithm, which when given a task, gets good at learning similar tasks. A human analogy would be, as a child we are taught the alphabet. We then gradually get good at stringing together words from alphabets, and subsequently, sentences from words. Now we need to mirror this behaviour in a machine. Can a machine learn similar tasks quickly? This process of learning the strategy lo learn certain kind of tasks is called **Meta Learning**. Since the learning procedure implicitly uses transfer learning it is often referred to as **meta transfer learning**.

During $\mathcal{T}^{(\mathbf{test})}$ (learning strategy updates for the meta model), if the base model gives poor performance, then the meta model updates the learning strategy. It then again gets feedback of the updated strategy during the next iteration of $\mathcal{T}^{(\mathbf{test})}$. If the strategy is good, the meta model reinforces it, else it tries to go away from it.

## 2 Meta Transfer Learning Algorithm Steps

### 2.1 Pretraining over a large similar dataset

For example: Training over a dataset in English versus Punjabi. There is lots of data available for English and very little for Punjabi. We can learn the parameters by training over an English dataset and transfer them to the model training over a Punjabi dataset.

$$[\mathbf{\Theta}, \theta] \leftarrow [\mathbf{\Theta}, \theta] - \alpha \nabla \mathcal{L}_D([\mathbf{\Theta}, \theta]) \tag{1}$$

Here, $\mathbf{\Theta}$ is the parameters obtained from the pretrained model, $\theta$ is the final parameters for the new task and $\mathcal{L}_D([\Theta, \theta])$ is the categorical cross entropy loss function. $\mathcal{L}_D([\Theta, \theta])$ can be approximated as,

$$\frac{1}{|D|} \sum_{(x,y^d)\sim D} \mathcal{L}(\mathcal{F}_{[\Theta,\theta]}(x), y^d) \tag{2}$$

In meta transfer learning, we transfer $\Theta$ to a new task and learn the final weight $\theta$ according to the new task. In transfer learning, we discard the parameter $\theta$.

### 2.2 Base Training

Given $\mathcal{T}^{train}$, for example, randomly learn some phonemes from Punjabi.
For the base training, we get the update step as,

$$\theta' \leftarrow \theta - \beta \nabla_\theta \mathcal{L}_{\mathcal{T}^{(train)}}([\Theta, \theta], \Phi) \tag{3}$$

Here, initialize $\theta$ from previous task (random initialization can also be done), $\Theta$ is not modified and $\Phi$ is obtained from the meta model (parameters which are adaptable from the task to task, for example: learning rate, initial weight size, etc.) Also, both $\Theta$ and $\Phi$ have small number of parameters thus making the entire training process computationally efficient.

## 2.3 Meta Learning

Given $\mathcal{T}^{(test)}$. For the meta learning step, we get the update step as,

$$\Phi \leftarrow \Phi - \gamma \nabla_{\Phi} \mathcal{L}_{\mathcal{T}^{(test)}}([\Theta, \theta], \Phi) \tag{4}$$

Here, $\Theta$ and $\theta$ are kept fixed. And, the update step for $\theta$ is given as,

$$\theta \leftarrow \theta - \nabla_{\theta} \mathcal{L}_{\mathcal{T}^{(test)}}([\Theta, \theta], \Phi) \tag{5}$$

$$\Phi = \{\Phi_{S_1} \Phi_{S_2}\} \tag{6}$$

In the paper $\Phi$ is a parameter responsible for the scaling and the shifting of weights.

$$w \odot \Phi_{S_1} + b + \Phi_{S_2} \tag{7}$$

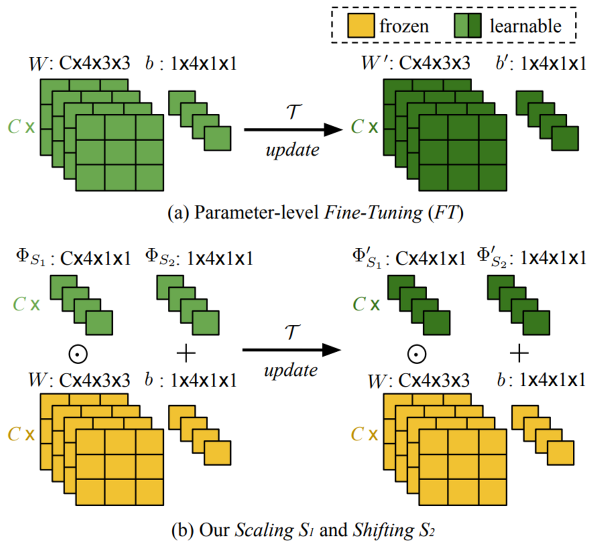

Figure 1: Representation of $\Phi$ as in the paper

# 3 Testing the Meta Model

Given a learned strategy (not updated anymore); which is used by the base model to learn and then tested. If this performance is good, then the meta model has learnt well from the meta training data.

# 4 Advantages

1. Using big data from a similar problem (information held by $\Theta$), it starts from a strong initialization yielding fast convergence for meta transfer learning.
2. Small data will lead to 'Catastrophic forgetting'. It can be avoided by not changing $\Theta$. ($\Theta$ is not learnt on small data, only $\Phi$ is learnt, which has less number of parameters).
3. Avoids overfitting, since we have smnaller number of trainable parameters.
   #trainable parameters $\ll$ #training datapoints

## 5 Curriculum Learning

Curriculum learning (CL) is a training strategy that trains a machine learning model from easier data to harder data, which imitates the meaningful learning order in human curricula. It involves learning the easy things/concepts first. The difficulty of a task is decided by the machine itself based on a test metric.
For example: for a 4 class classification task with $F_1$ score as the test metric, we decide the classification difficulty level of each class. Say the classes $C_1, C_2, C_3$ and $C_4$ have the $F_1$ score as, 0.98, 0.80, 0.90 and 0.86 respectively. Then the class $C_2$ must be most difficult to learn class.
Across all task $\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_m$, identify the $m$ most difficult classes (from each task get the most difficult classes to learn) based on the relative accuracy, and now you have obtained the $m$ hard classes. In the next epoch we only sample from these classes,

$$\mathcal{T}^{(hard)} \sim \mathcal{P}(\mathcal{T} \mid m) \tag{8}$$

The gist of curriculum learning is 'GROW UP THROUGH HARDNESS'.

## 6 Ablation Analysis

We check how much gain in accuracy each innovative step has brought.
For example,

- Start with a baseline model.
- Add first variation (Meta Transfer Learning) and check the improvement.
- Add both the variations (Meta Transfer Learning and Curriculum Learning) and check the improvement.

## 7 References

- Meta-Transfer Learning for Few-Shot Learning : Qianru Sun, Yaoyao Liu, Tat-Seng Chua, Bernt Schiele