Meta-Learning & Few-Shot Learning

Chen Shangyu

January 5, 2019

Learning to learn by gradient descent by gradient descent

Weights of original model is updated by gradients generated from a meta-learner (LSTM):

$$\theta_{t+1} = \theta_t + g_t \tag{1}$$

$$\begin{bmatrix} g_t \\ h_{t+1} \end{bmatrix} = m(\nabla_t, h_t, \phi) \tag{2}$$

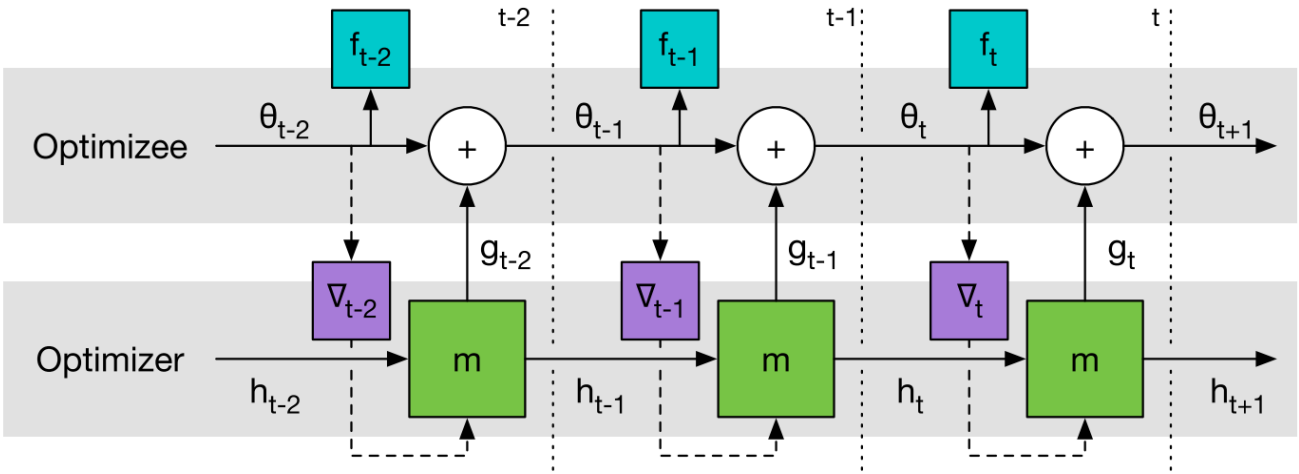


Figure: Computational graph used for computing the gradient of the optimizer.

Weights are Updated Independently

Coordinatewise is equal to element-wise.

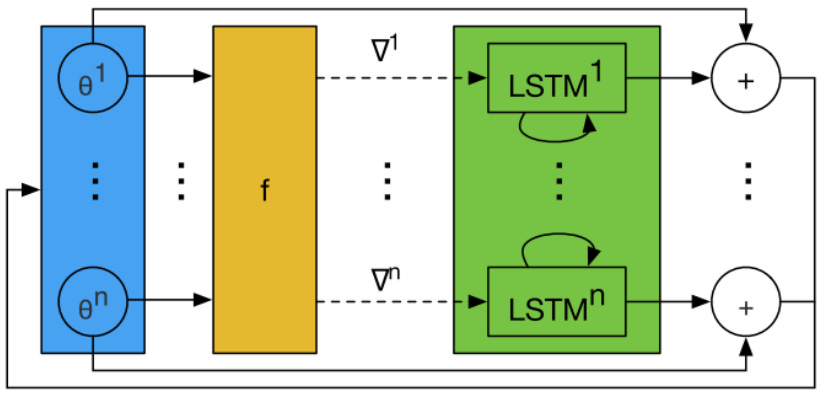


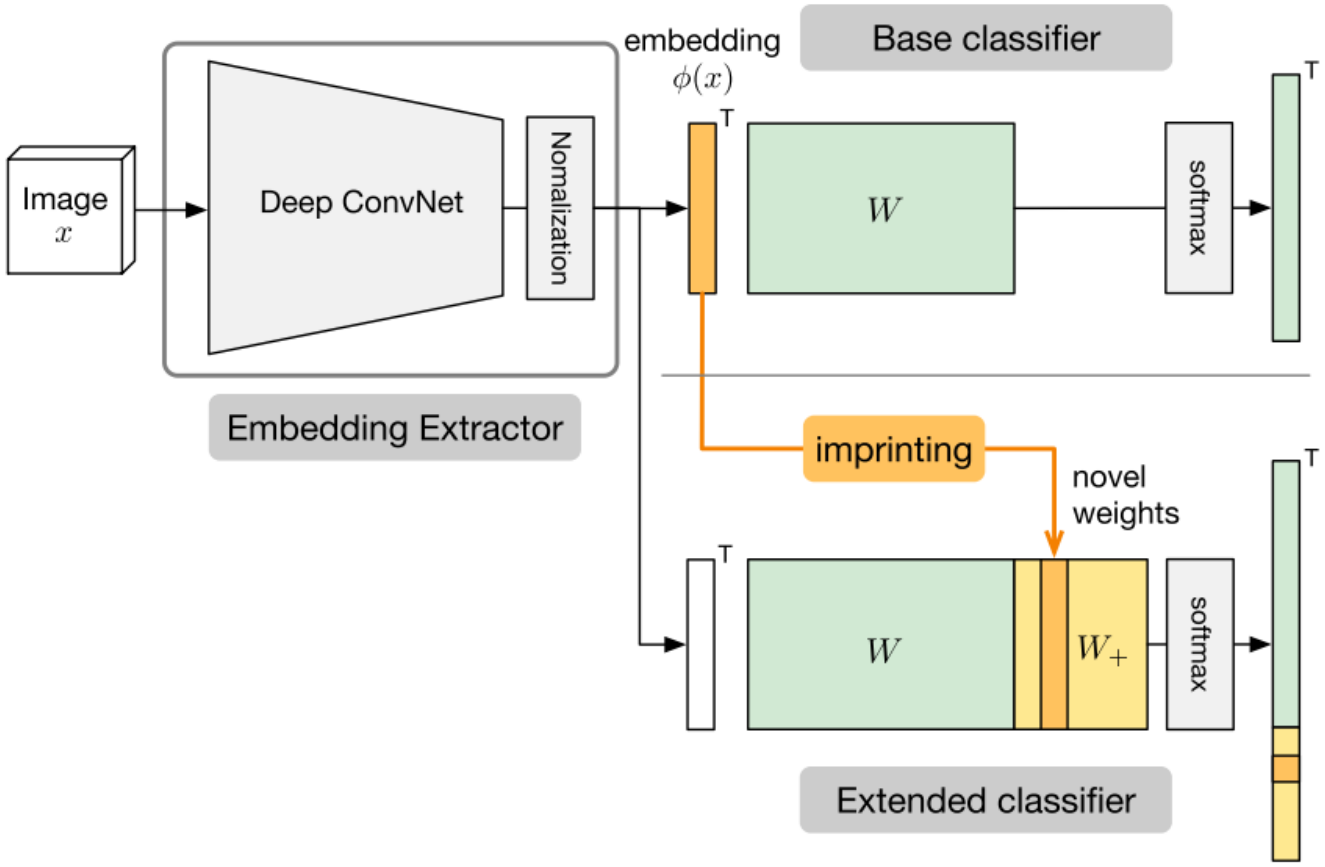
Figure: All LSTMs have shared parameters, but separate hidden states.

Few-Shot Learning

Problem Setting: Given plenty of labeled data as base classes, k instances per classes (total N classes: *N-way K shot*) as **novel classes**, model is required to classify other more instances from novel classes.
Think about kNN.

Low-Shot Learning with Imprinted Weights

Imprinted weights / embedding of low-shot samples are regarded as centroids, similar to KNN.



Matching Networks for One Shot Learning

Most important / strangest: Training procedure is based on a simple machine learning principle: test and train conditions must match.

Some important concepts:

- Test: Given N-way K-shot instances (with labels), predict arbitrary instances (without labels).
- Support set: Part of training data: N-way K-shot instances (with labels).
- Query set: Part of training data: arbitrary instances (with labels) to be predicted.

Model Architecture

Defined:

- Few-Shot training set: $S = (x_i, y_i)_{i=1}^{k}$
- Test example: $\hat{x}$

$$\hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i \tag{3}$$

where $a(\hat{x}, x_i) = \frac{e^{c(f(\hat{x}), g(x_i)}}{\sum_{j=1}^{k} e^{c(f(\hat{x}), g(x_j)}}$ .Output for a new class as a linear combination of the labels in the support set.

Meta-Learning as Learn with External Memory

In training, construct a LSTM for storing information:

$$f(\hat{x}, S) = \text{attLSTM}(f^{'}(\hat{x}), g(S), K) \tag{4}$$

where $f^{'}(\hat{x})$ are the CNN extracted features, K is a fixed number of unrolling steps of LSTM.

SkipNet: Learning Dynamic Routing in Convolutional Networks

This work proposed a modified residual network to selectively skip convolutional blocks based on the activations of the previous layer. Whether to skip is learnt by RL.

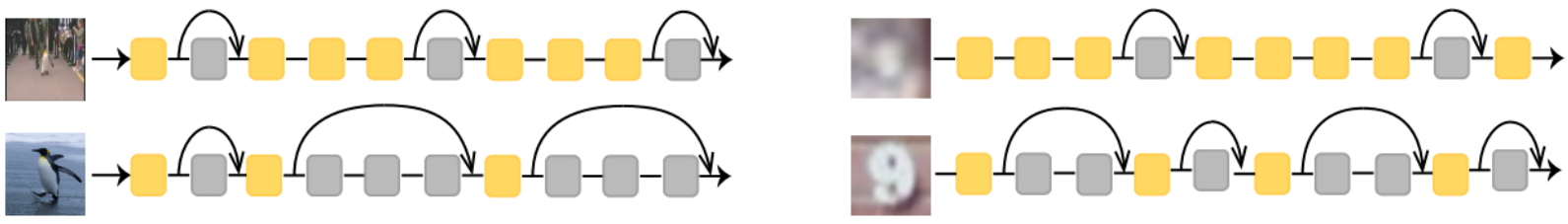
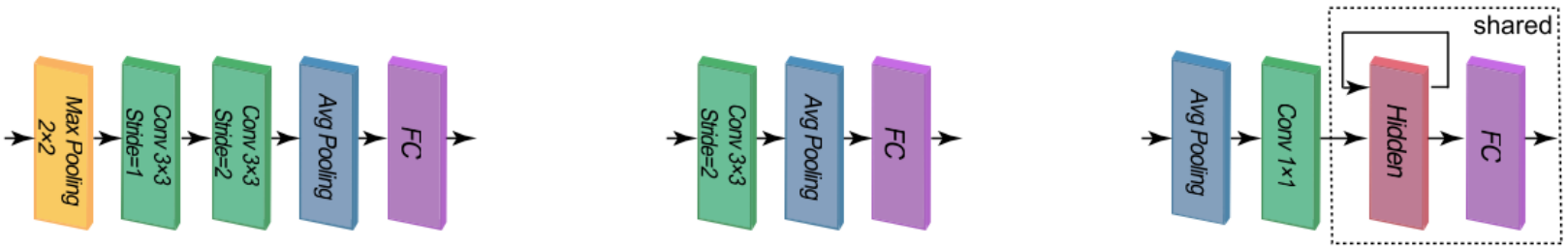


Figure: The SkipNet learns to skip convolutional layers on a per-input basis. More layers are executed for challenging images (top) than easy images (bottom)

Gating Network Design



(a) FFGate-I (b) FFGate-II (c) RNNGate

- Supervised Pre-training Optimizing
- Skipping Policy Learning with Hybrid RL