Learning to learn by gradient descent by gradient descent

Akhmad Sumekenov

Higher School of Economics

20 марта 2019 г.

Overview

MUSK

Learning to Learn without Gradient Descent by Gradient Descent

Yutian Chen [1] Matthew W. Hoffman [1] Sergio Gómez Colmenarejo [1] Misha Denil [1] Timothy P. Lillicrap [1] Matt Botvinick [1] Nando de Freitas [1]

Abstract

We learn recurrent neural network optimizers trained on simple synthetic functions by gradient descent. We show that these learned optimizers exhibit a remarkable degree of transfer in that they can be used to efficiently optimize a broad

dient descent, evolutionary strategies, simulated annealing, and reinforcement learning.

For instance, one can learn to learn by gradient descent by gradient descent, or learn local Hebbian updates by gradient descent (Andrychowicz et al., 2016; Bengio et al., 1992). In the former, one uses supervised learning at the

Мотивация

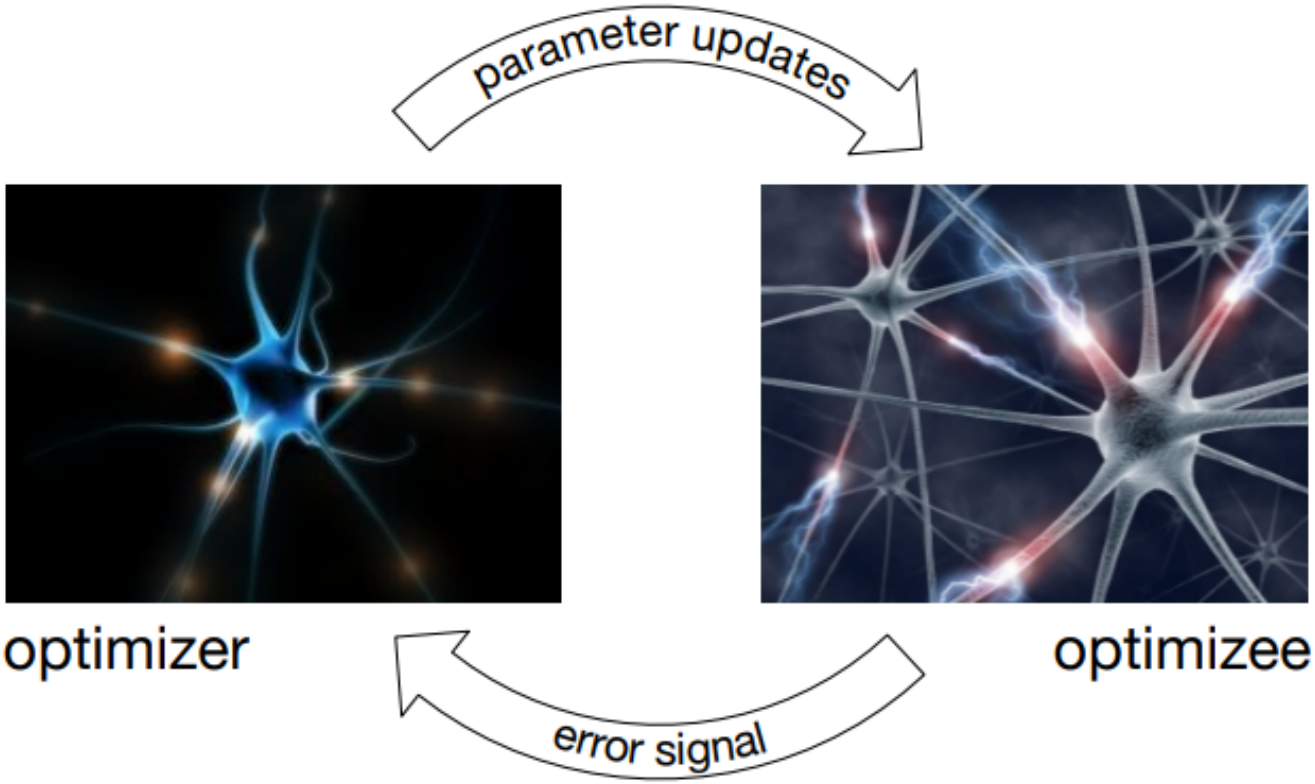
- Самый банальный способ оптимизации:

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t)$$

- Позже люди придумали более умные способы: Nesterov, RMSProp, Adam, etc.
- Но эти способы созданы руками!

Optimizer and Optimizee

Также как и в компьютерном зрении нейронная сеть научилась выявлять свои image features вместо hand-crafted features, вроде разных вертикальных и горизонтальных фильтров, мы хотим создать нейронную сеть, которая оптимизирует функцию сама.



Формулировка

- f - optimizee, нейронная сеть, loss которой мы хотим минимизировать
- g - optimizer, вторая сеть, выход которой это величина, на которую меняются веса первой сети (которую мы хотим оптимизировать):

$$\theta_{t+1} = \theta_t + g_t(\nabla f(\theta_t), \phi)$$

- ϕ - параметры optimizer сети.

Loss

- Обозначим loss of optimizer:

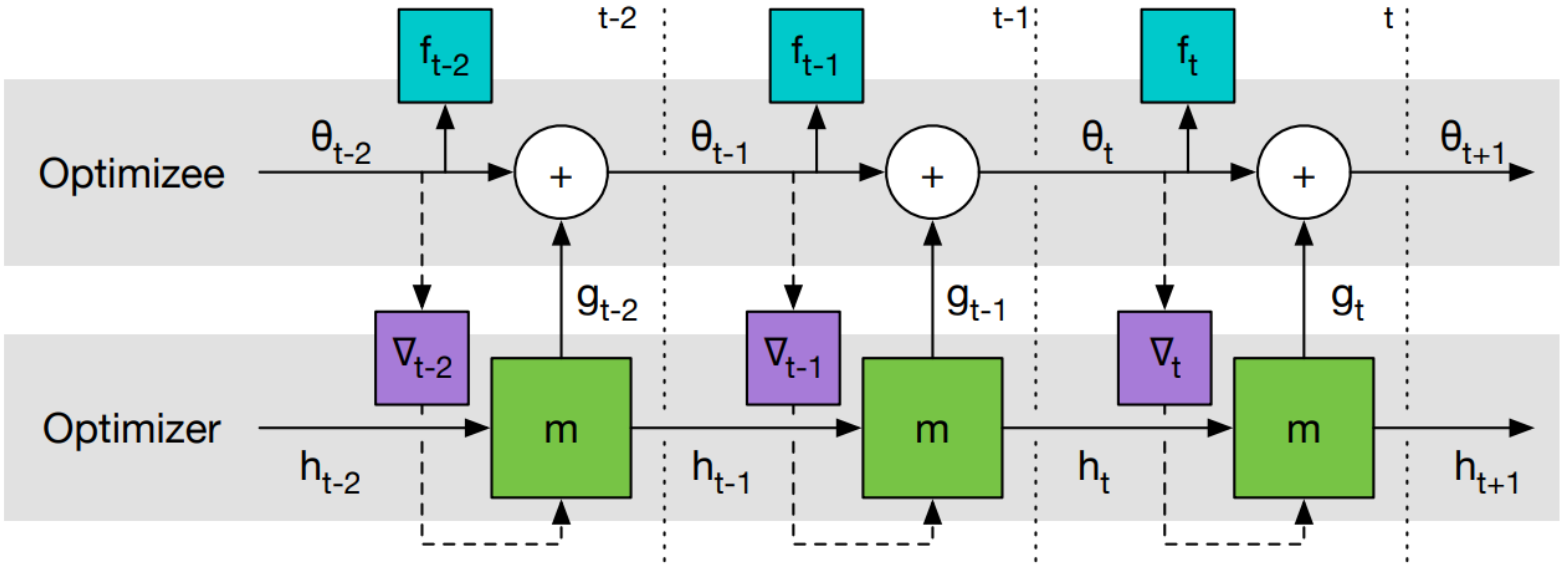
$$\mathcal{L}(\phi) = \mathbb{E}_f\Big(\Big[f(\theta^*(f,\phi))\Big]\Big)$$

где $f \in \mathbb{D}$, т.е. принадлежит какому-то распределению функций.

- Для удобства обучения, берется сумма до установленного горизонта T:

$$\mathcal{L}(\phi) = \mathbb{E}_f\Big[\sum_{t=1}^{T} \omega_t f(\theta_t)\Big]$$

Архитектура



Архитектура

- Update rule:

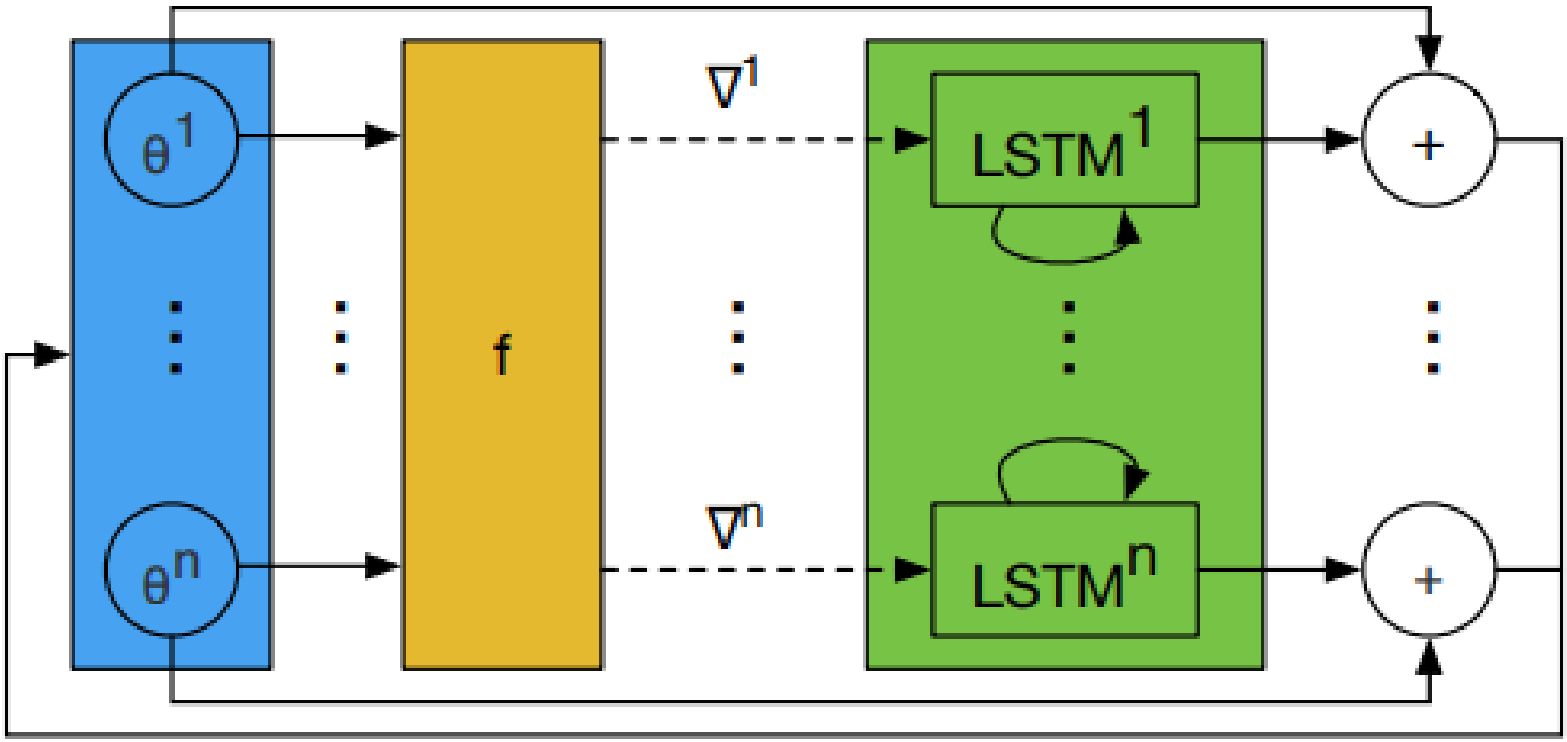
$$\theta_{t+1} = \theta_t + g_t$$

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

- m - это RNN сеть с LTSM модулем

Coordinate-wise LSTM

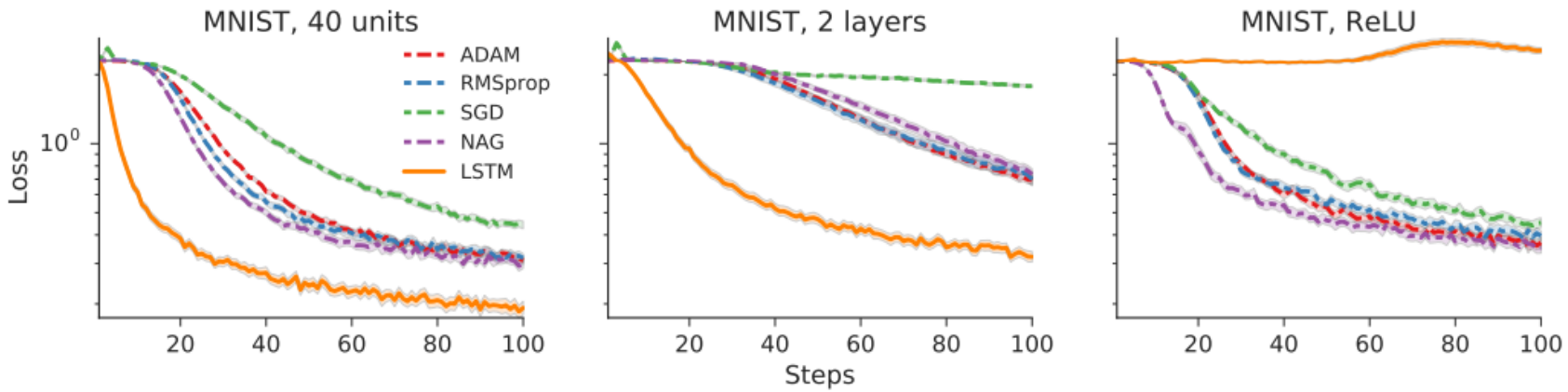
- Использовать обычный fully-connected RNN очень затратно
- Поэтому каждая координата весов оптимизируется отдельно от других с помощью LSTM модуля:



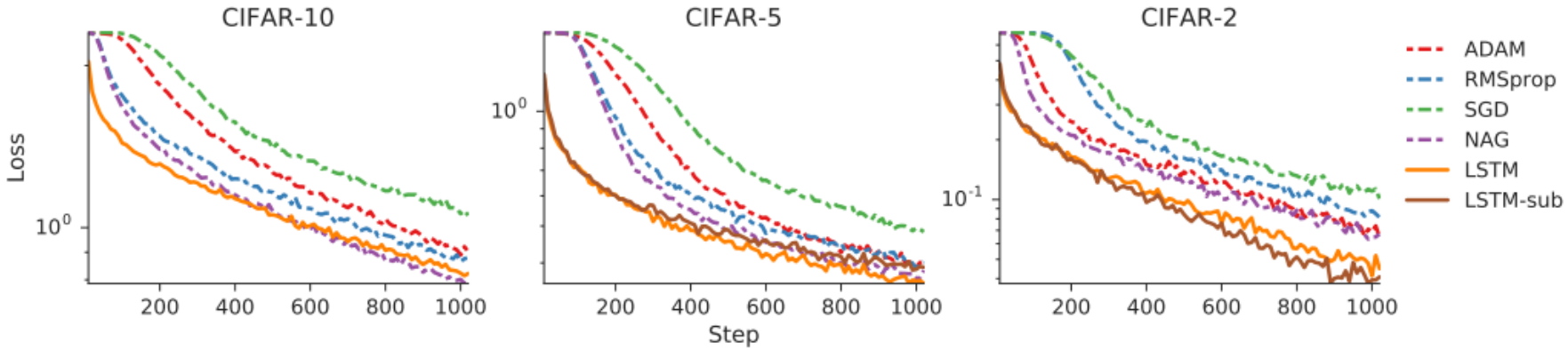
Coordinate-wise LSTM

- Из-за такой архитектуры, на обучение требуется намного меньше времени
- LSTM в оптимизаторе является аналогом запоминания старых значений градиентов в RMSProp, ADAM
- Позволяет не беспокоиться о магнитуде значений в разных координатах, потому что они оптимизируются независимо

Эксперименты: MNIST



Эксперименты: CIFAR

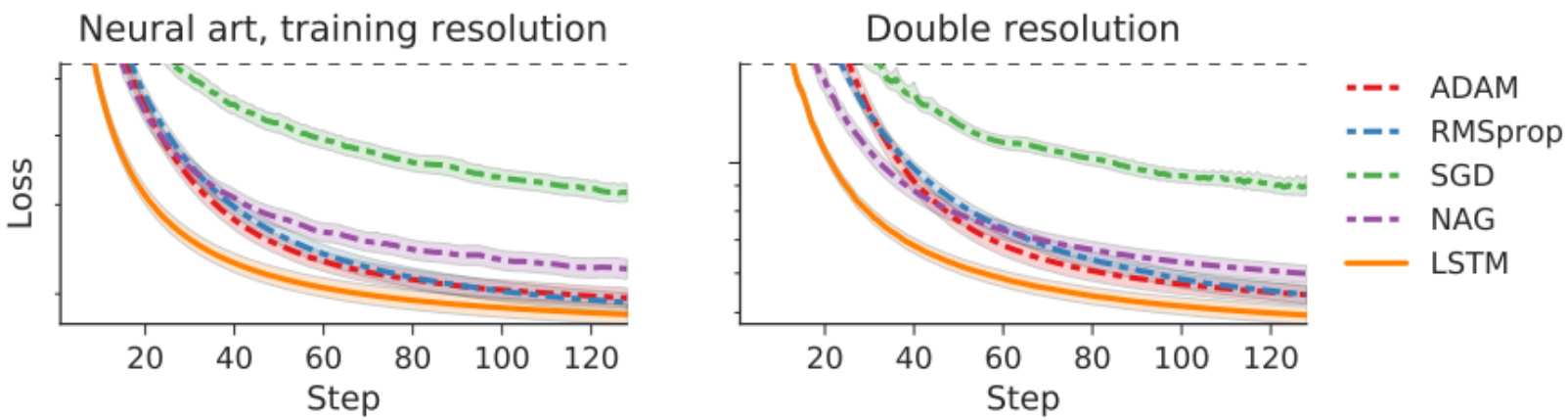


Эксперименты: Neural Art

- В создании картин с похожей стилистикой учавствует Content Image(c) и Style Image(s)
- Функция потерь формулируется таким образом:

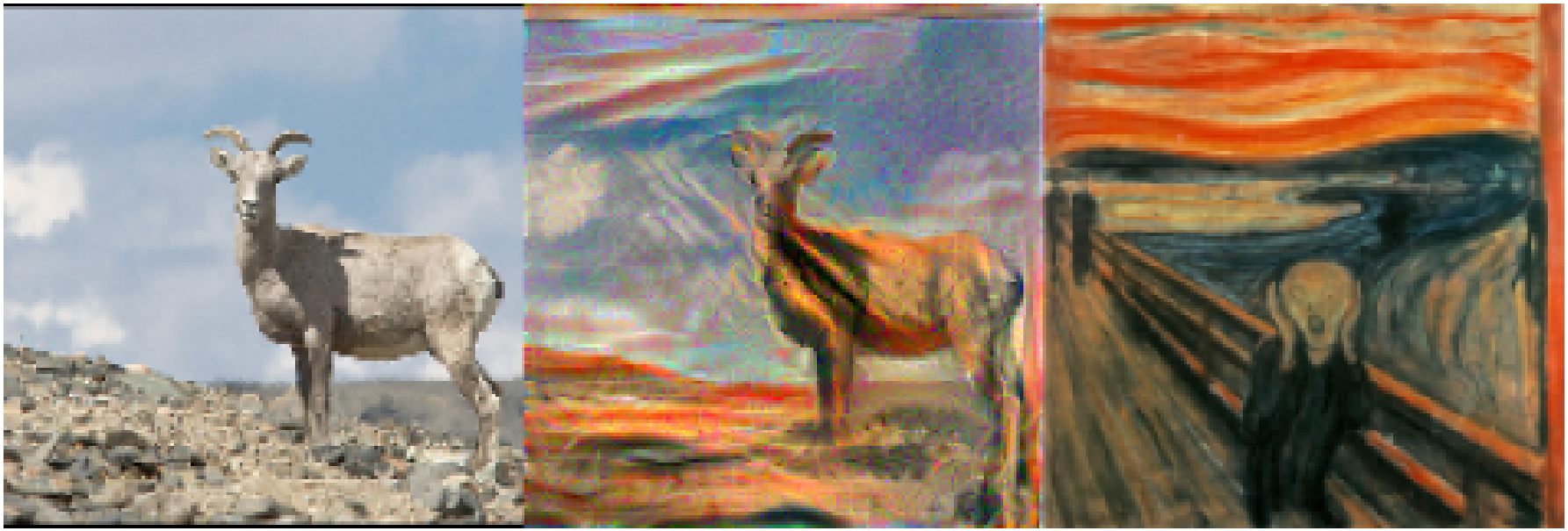
$$\mathcal{L}(\theta) = \alpha \mathcal{L}_{content}(c, \theta) + \beta \mathcal{L}_{style}(s, \theta) + \mathcal{L}_{reg}(\theta)$$

Эксперименты: Neural Art



Эксперименты: Neural Art

Эксперименты: Neural Art



Выводы

- Сформулирована задача "обучения с помощью обучения"
- Архитектура хорошо генерализируется на похожие модели с большим количеством параметров
- Лучше чем hand-crafted способы оптимизации(ADAM, RMSprop, SGD)

References

Marcin Andrychowicz (2016)

Learning to learn by gradient descent by gradient descent

Google DeepMind

The End