Learning to learn by gradient descent by gradient descent

Marcin Andrychowicz[1], **Misha Denil**[1], **Sergio Gómez Colmenarejo**[1], **Matthew W. Hoffman**[1], **David Pfau**[1], **Tom Schaul**[1], **Brendan Shillingford**[1,2], **Nando de Freitas**[1,2,3]

[1]Google DeepMind [2]University of Oxford [3]Canadian Institute for Advanced Research

Outline

- Introduction
  - Gradient descent
- Method
  - Method
  - Architecture
- Experiments

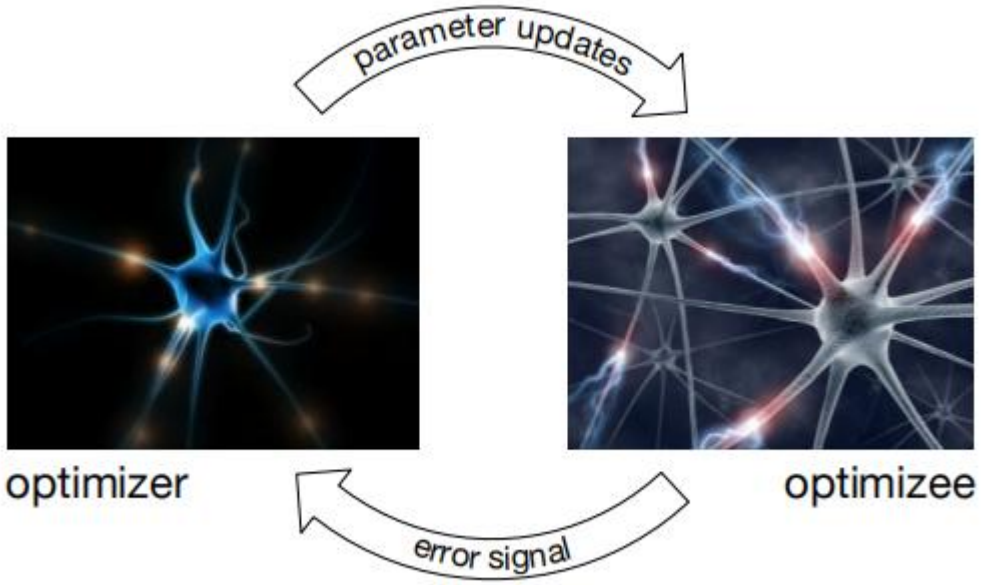
Gradient descent

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

- Much of the modern work in optimization is based around designing update rules tailored to specific classes of problems
- The types of problems differing between different research communities
- The weak Generalization ability.

Methods

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



- Casting algorithm design as a learning problem.
- The update rule g using a recurrent neural network.

Develop a procedure for constructing a learning algorithm which performs well on a particular class of optimization problems.

Methods

Objective:

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

Objective function f(θ) defined over some domain $\theta \in \Theta$.

find the minimizer $\theta^* = \arg\min_{\theta \in \Theta} f(\theta)$.

- the objective function depends only on the final parameter value

Methods

Objective:

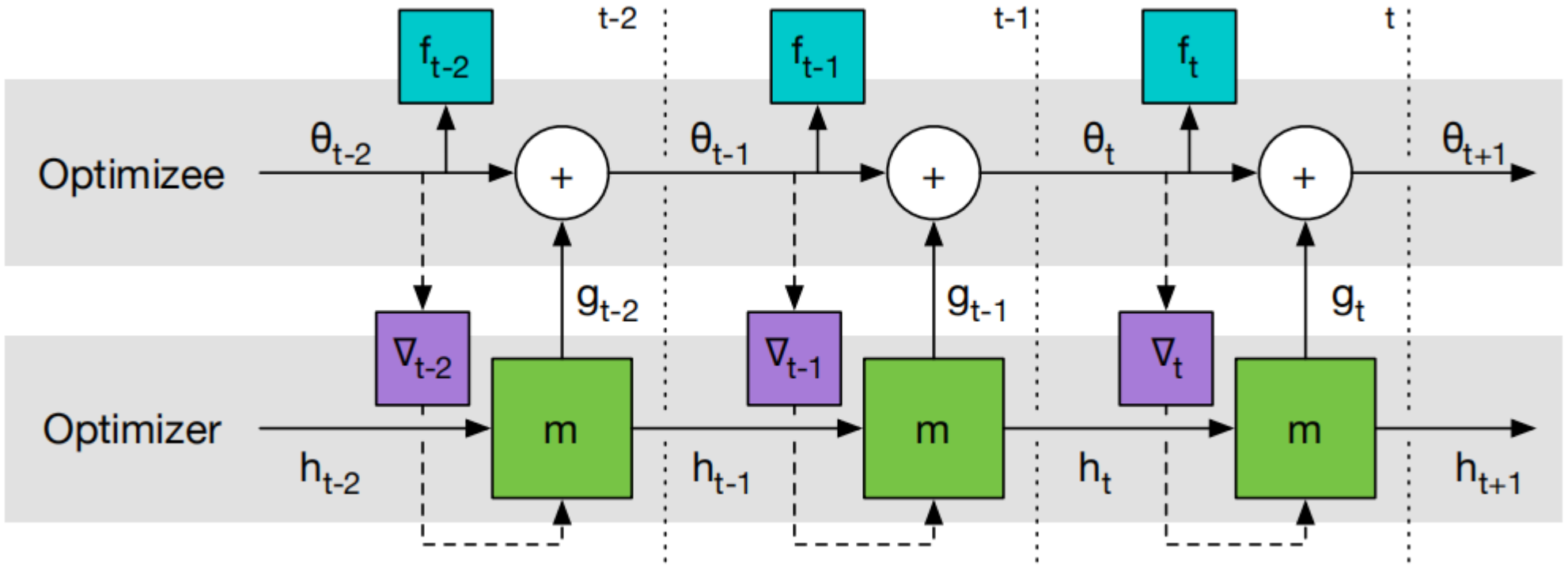
$$\mathcal{L}(\phi) = \mathbb{E}_f \left[\sum_{t=1}^{T} w_t f(\theta_t) \right] \qquad \text{where} \qquad \theta_{t+1} = \theta_t + g_t \,,$$

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

- by sampling a random function f
- applying backpropagation to the computational graph

Architecture

BPTT



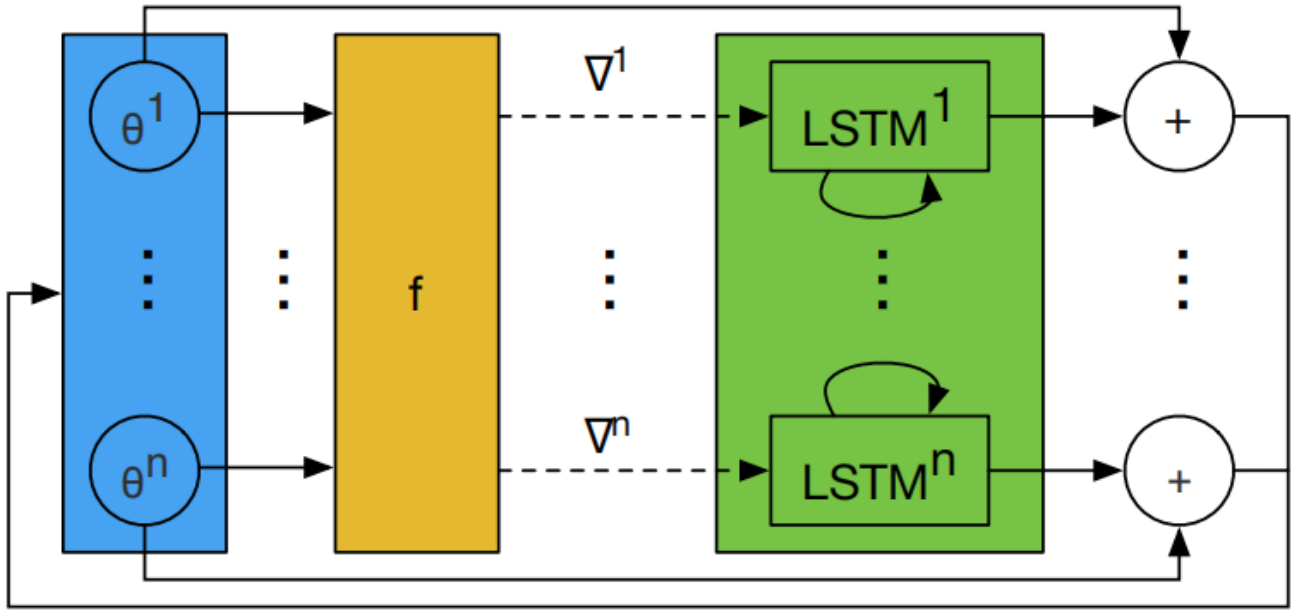
- The gradients in dashed lines are not propagated during gradient descent.
- The gradients of the optimizee do not depend on the optimizer parameters avoid computing second derivatives of f

$$\partial \nabla_t / \partial \phi = 0$$

Architecture

Coordinatewise LSTM optimizer:

- Too many parameters
  - All LSTMs have shared parameters
  - But separate hidden states



Architecture

Information sharing between coordinates:

- Ignore correlations between coordinates into effect
  - Global averaging cells
  - NTM-BFGS optimizer

$$
\begin{aligned}
g_t &= \mathrm{read}(M_t, \theta_t) \quad = -M_t \nabla f(\theta_t) \\
\theta_{t+1} &= \theta_t + g_t \\
M_{t+1} &= \mathrm{write}(M_t, \theta_t, g_t)\,.
\end{aligned}
$$

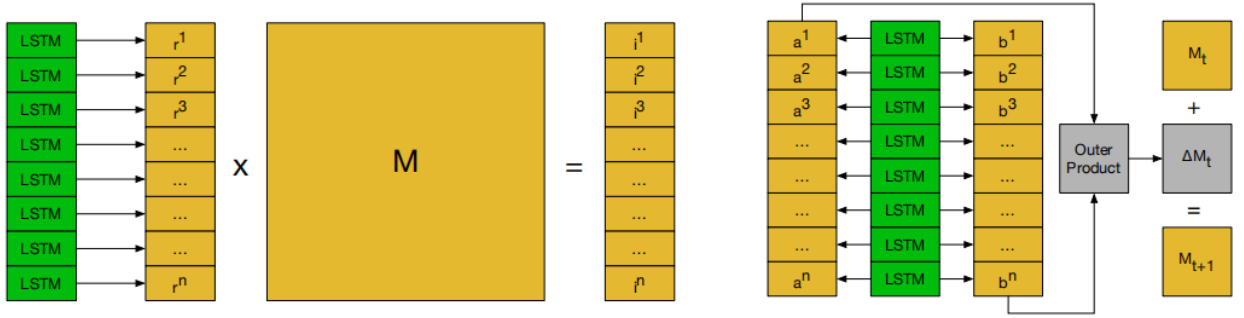


Figure 14: **Left:** NTM-BFGS read operation. **Right:** NTM-BFGS write operation.

Architecture

Preprocessing and postprocessing

- neural networks naturally disregard small variations in input signals and concentrate on bigger input values

$$(\log(|\nabla|),\ \mathrm{sgn}(\nabla)) \text{ as an input,}$$

$$\nabla^k \to \begin{cases} \left(\frac{\log(|\nabla|)}{p}, \mathrm{sgn}(\nabla)\right) & \text{if } |\nabla| \geq e^{-p} \\ (-1, e^p \nabla) & \text{otherwise} \end{cases}$$

where p > 0 is a parameter controlling how small gradients are disregarded

Experiments

Experiments:

- Quadratic functions
- Training a small neural network on MNIST
- Training a convolutional network on CIFAR-10
- Neural Art

Experiments

LSTM 优化器的学习过程如下：

1、对 $f(\theta) = \|W\theta - y\|_2^2$ ，随机采样 W 和 y 得到训练数据。

2、初始化函数 f 的参数 θ 和优化器参数 ϕ

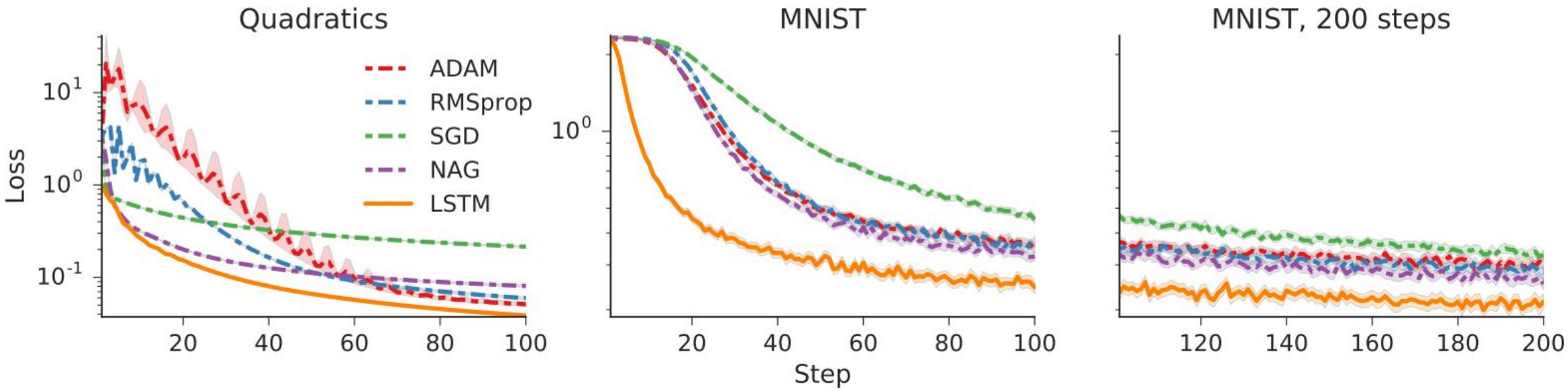
3、根据当前时刻 t 的 θ，求出 f 的值，并求出 θ 的误差 ∇_t

4、将 ∇_t 以及 LSTM 优化器前一时刻的隐藏层作为输入，同时得到 θ 的更新梯度 g_t

5、利用 g_t 更新 θ，并利用上述的损失函数更新优化器参数 ϕ。

6、循环步骤 3-5，直到收敛。

Experiments

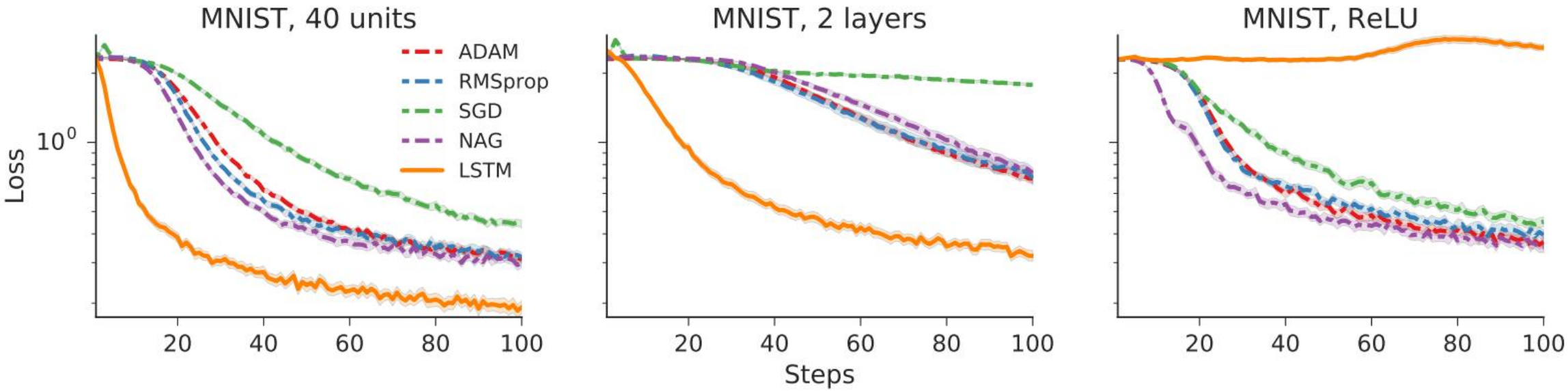


$$f(\theta) = \|W\theta - y\|_2^2$$

Test whether trainable optimizers can learn to optimize a small neural network on MNIST

Experiments

Generalization to different architectures:



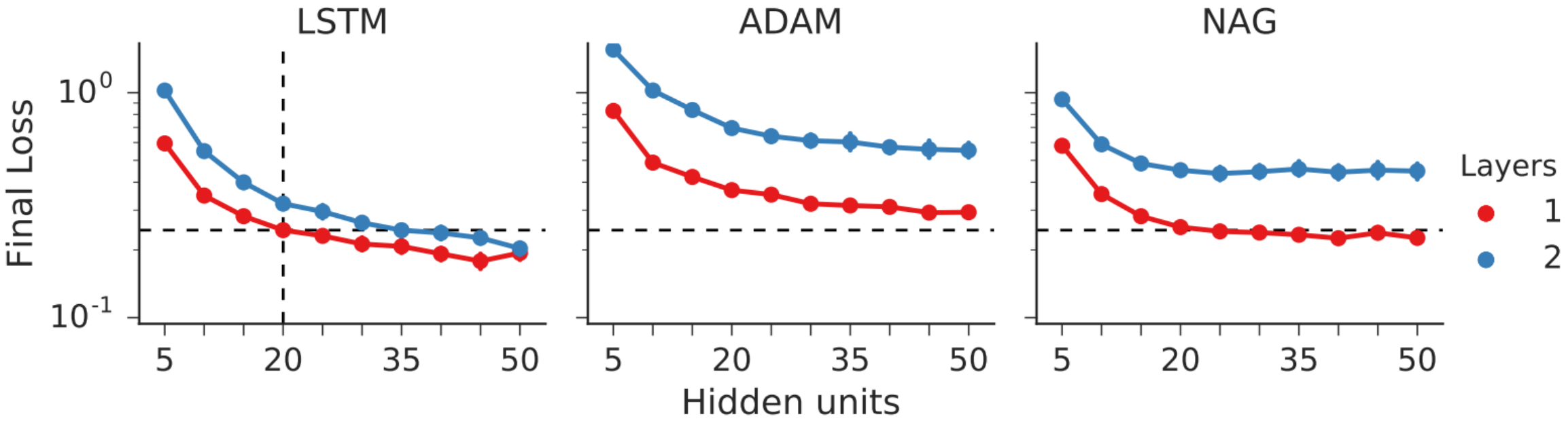
(1) an MLP with 40 hidden units instead of 20.

(2) A network with two hidden layers instead of one.

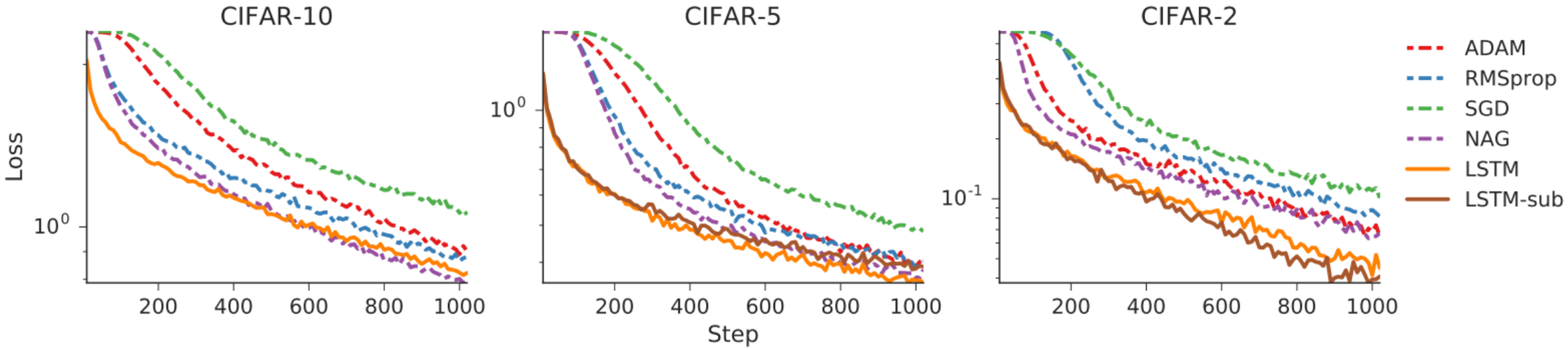
(3) a network using ReLU activations instead of sigmoid.

Experiments

Generalization to different architectures:



Experiments



one proposes parameter updates for the fully connected layers and the other updates the convolutional layer parameters

Experiments

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