Synthetic Gradients

Закирова Ксения

Paper by Jaderberg

Decoupled Neural Interfaces using Synthetic Gradients

Max Jaderberg [1] **Wojciech Marian Czarnecki** [1] **Simon Osindero** [1] **Oriol Vinyals** [1] **Alex Graves** [1] **David Silver** [1]
Koray Kavukcuoglu [1]

Abstract

Training directed neural networks typically requires forward-propagating data through a computation graph, followed by backpropagating error signal, to produce weight updates. All layers, or more generally, modules, of the network are therefore locked, in the sense that they must wait for the remainder of the network to execute forwards and propagate error backwards before they can be updated. In this work we break this constraint by decoupling modules by introducing a model of the future computation of the network graph. These models predict what the result of the modelled subgraph will produce using only local information. In particular we focus on modelling error gradients: by using the modelled *synthetic gradient* in place of true backpropagated error gradients we decouple subgraphs, and can update them independently and asynchronously *i.e.* we realise *decoupled neural interfaces*. We show results for feed-forward models, where every layer is trained asynchronously, recurrent neural networks (RNNs) where predicting one's future gradient extends the time over which the RNN can effectively model, and also a hierarchical RNN system with ticking at differ-

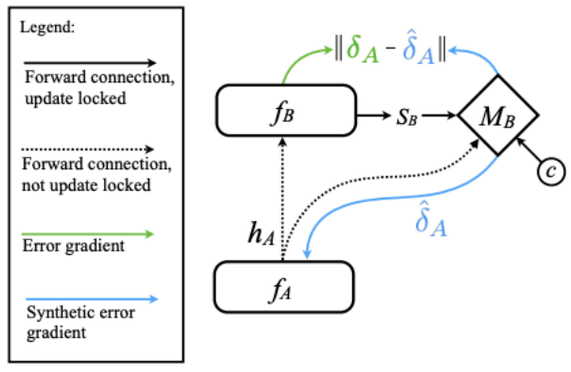
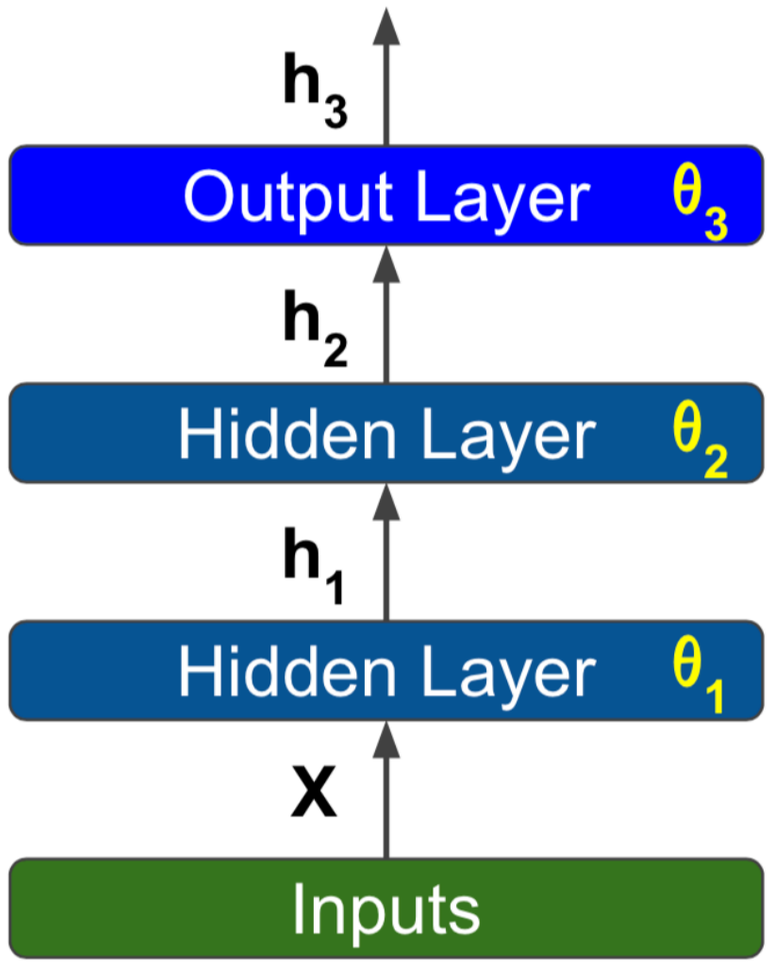


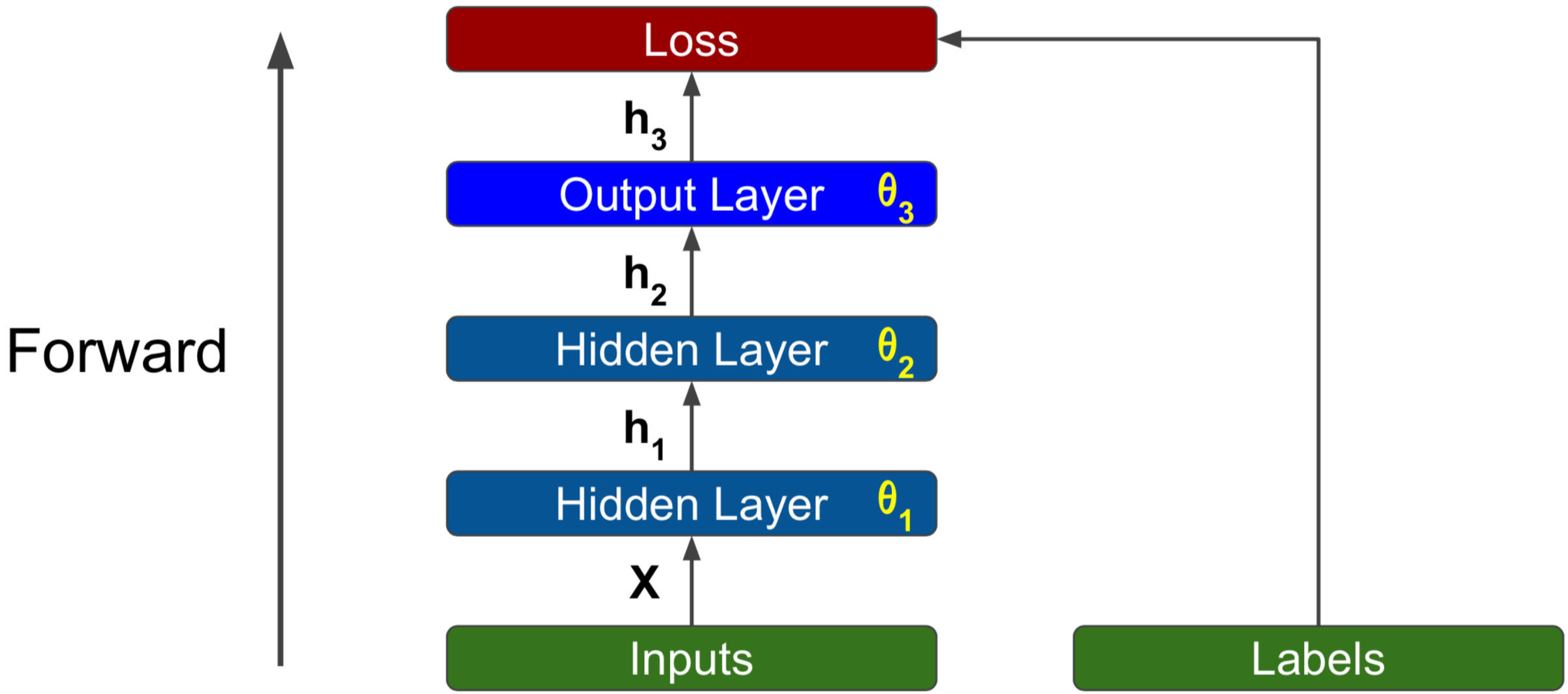
Figure 1. General communication protocol between A and B. After receiving the message h_A from A, B can use its model of A, M_B, to send back *synthetic gradients* $\hat{\delta}_A$ which are trained to approximate real error gradients δ_A. Note that A does not need to wait for any extra computation after itself to get the correct error gradients, hence decoupling the backward computation. The feedback model M_B can also be conditioned on any privileged information or context, c, available during training such as a label.

creating a forward processing graph which defines the flow of data from the network inputs, through each module, producing network outputs. Defining a loss on outputs allows errors to be generated, and propagated back through the network graph to provide a signal to update each module.

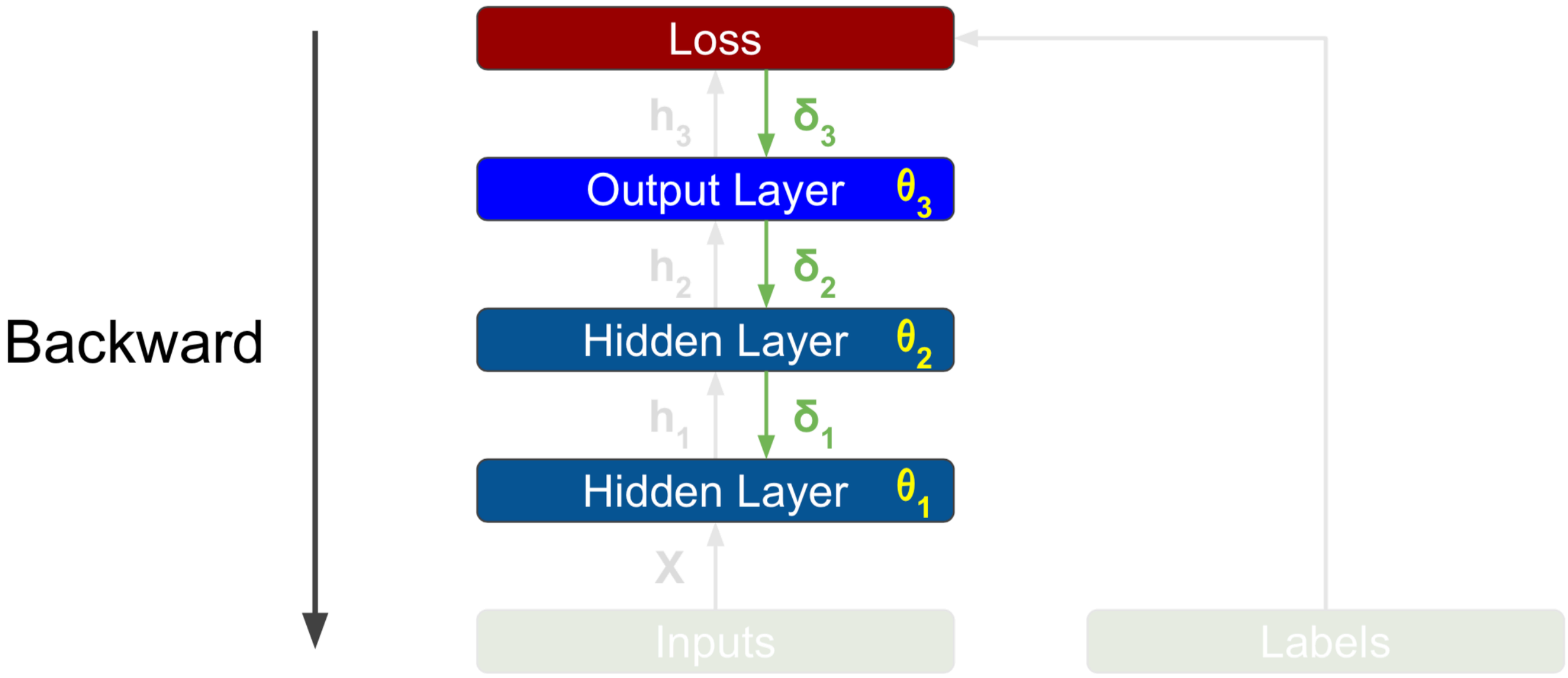
Backpropagation



Backpropagation

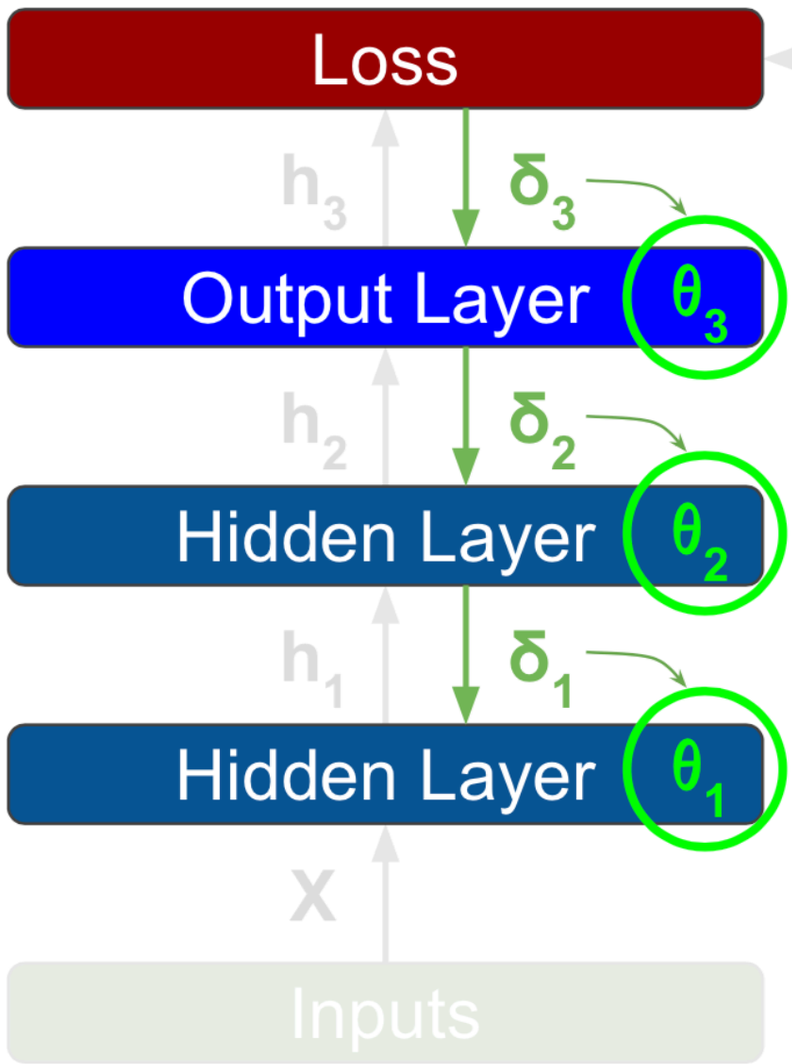


Backpropagation



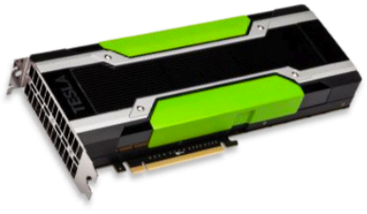
Backpropagation

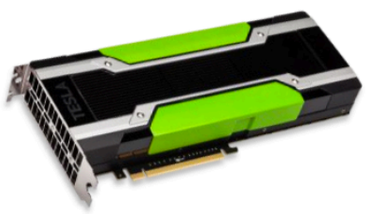
Gradient Descent Step



$$\theta_i^{t+1} = \theta_i^t + \alpha \left(\frac{\partial L}{\partial \theta_i^t} \right)$$

Model Parallelism





Loss

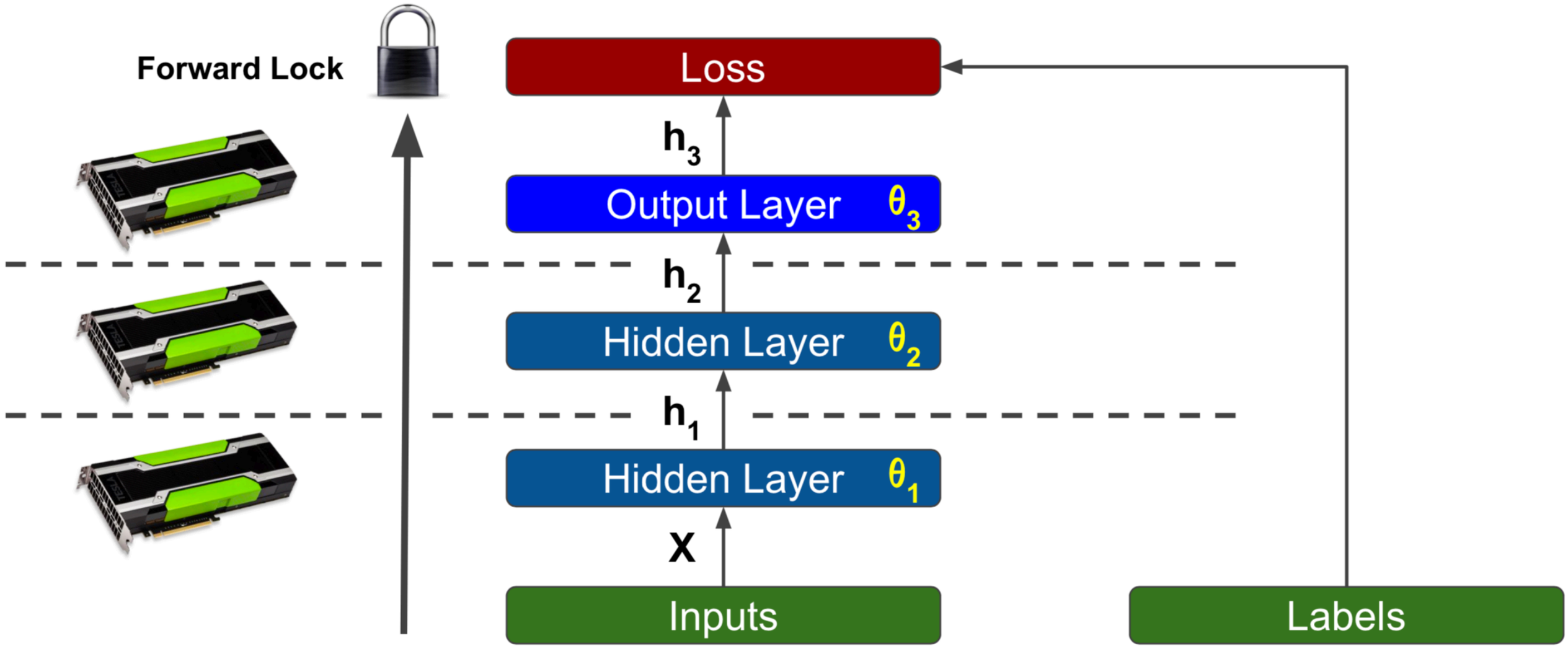
Output Layer

Hidden Layer

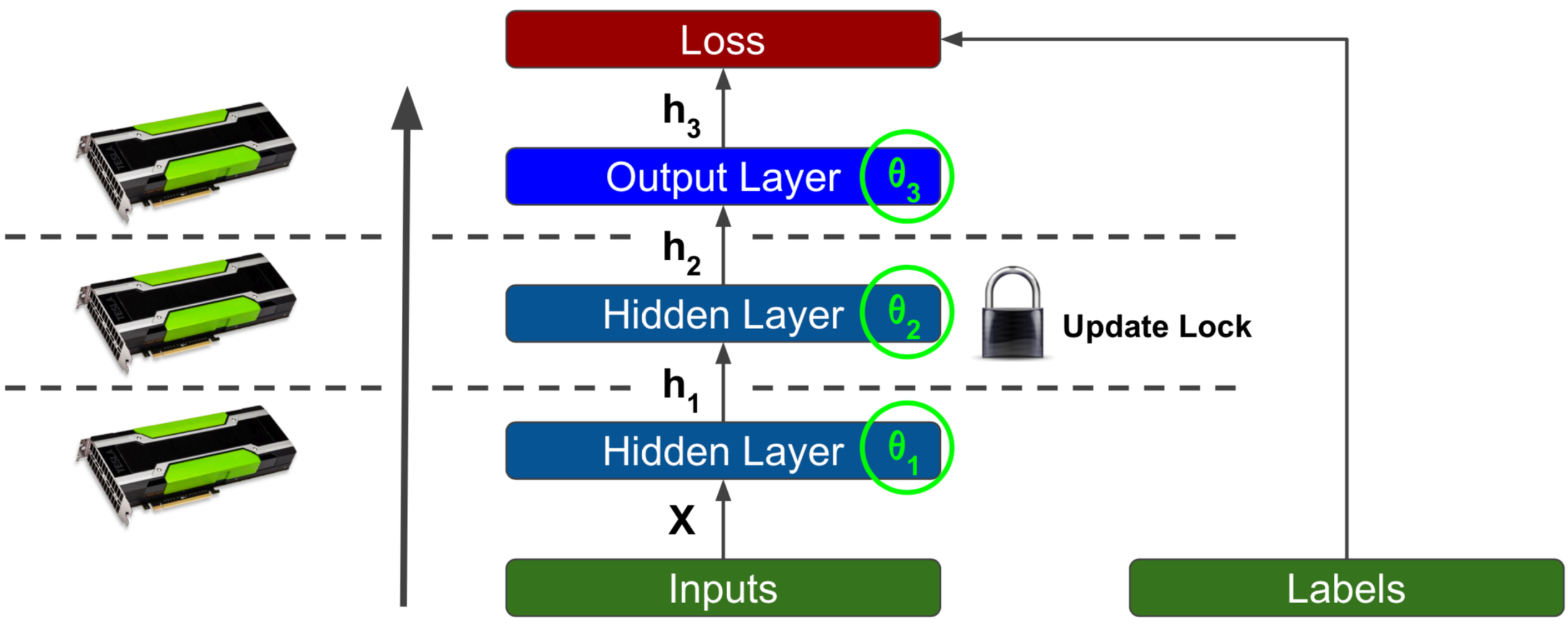
Hidden Layer

Inputs

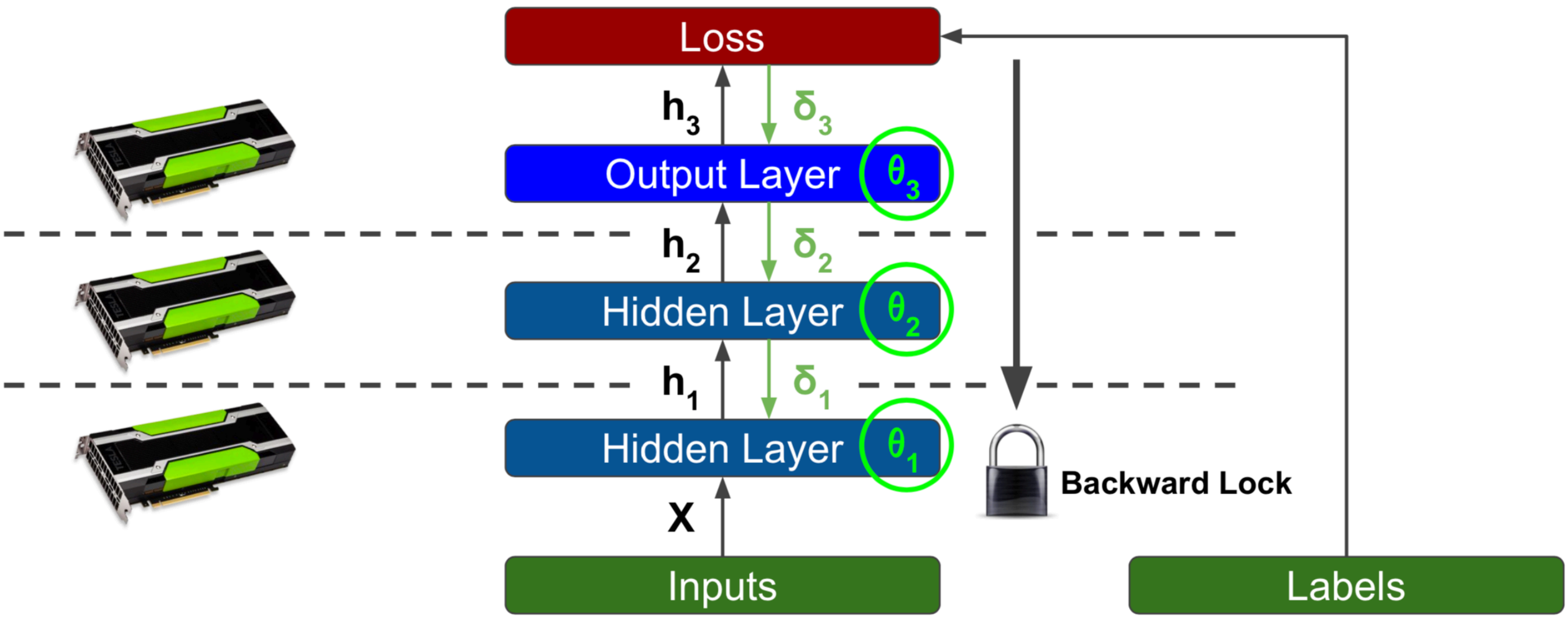
Model Parallelism



Model Parallelism



Model Parallelism



How can we break the lock?

Approximation!

Approximation

How we do it earlier:

$$\theta_i^{t+1} = \theta_i^t + \alpha \left(\frac{\partial L}{\partial \theta_i^t} \right)$$

Approximation

How we do it earlier: $$\theta_i^{t+1} = \theta_i^t + \alpha \left(\frac{\partial L}{\partial \theta_i^t} \right)$$

Let's do some approximation:

$$\frac{\partial L}{\partial \theta_i} = f_{Bprob}((h_i, x_i, y_i, \theta_i), (h_{i+1}, x_{i+1}, y_{i+1}, \theta_{i+1}), \dots) \frac{\partial h_i}{\partial \theta_i} \approx$$

$$\approx \hat{f}_{Bprop}(h_i) \frac{\partial h_i}{\partial \theta_i}$$

Approximation

How we do it earlier: $$\theta_i^{t+1} = \theta_i^t + \alpha \left(\frac{\partial L}{\partial \theta_i^t} \right)$$

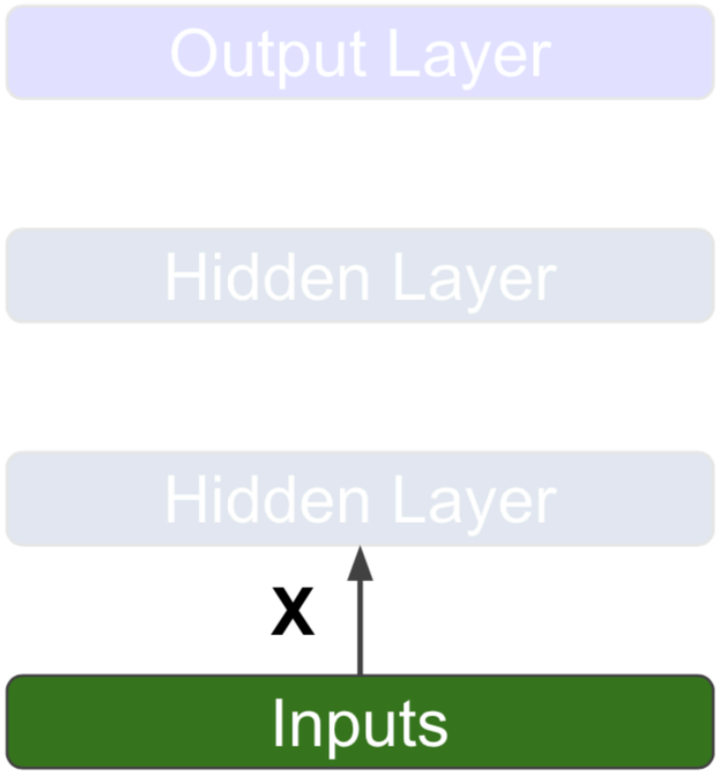
Let's do some approximation:

$$\frac{\partial L}{\partial \theta_i} = f_{Bprob}((h_i, x_i, y_i, \theta_i), (h_{i+1}, x_{i+1}, y_{i+1}, \theta_{i+1}), \dots) \frac{\partial h_i}{\partial \theta_i} \approx$$

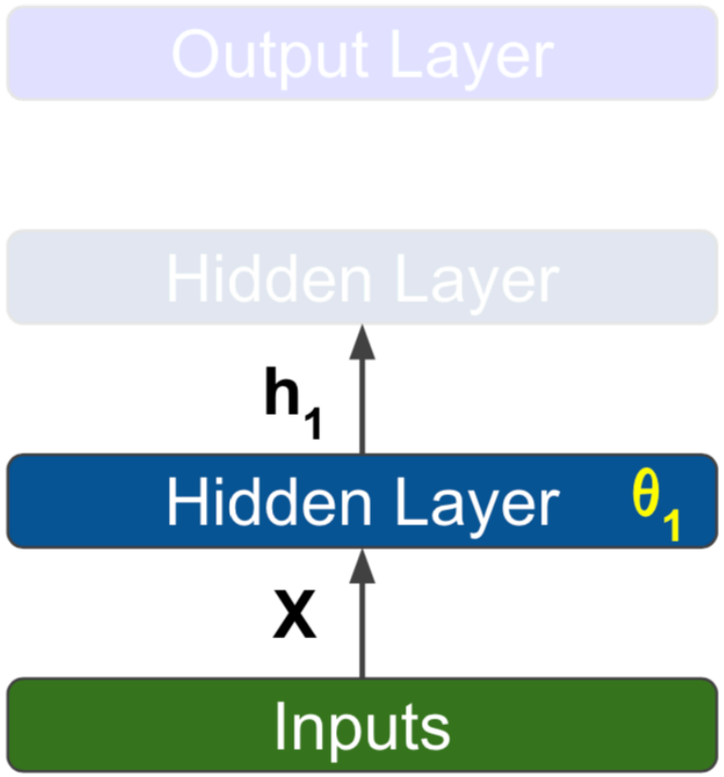
$$\approx \hat{f}_{Bprop}(h_i) \frac{\partial h_i}{\partial \theta_i}$$

(and generally it's the main idea)

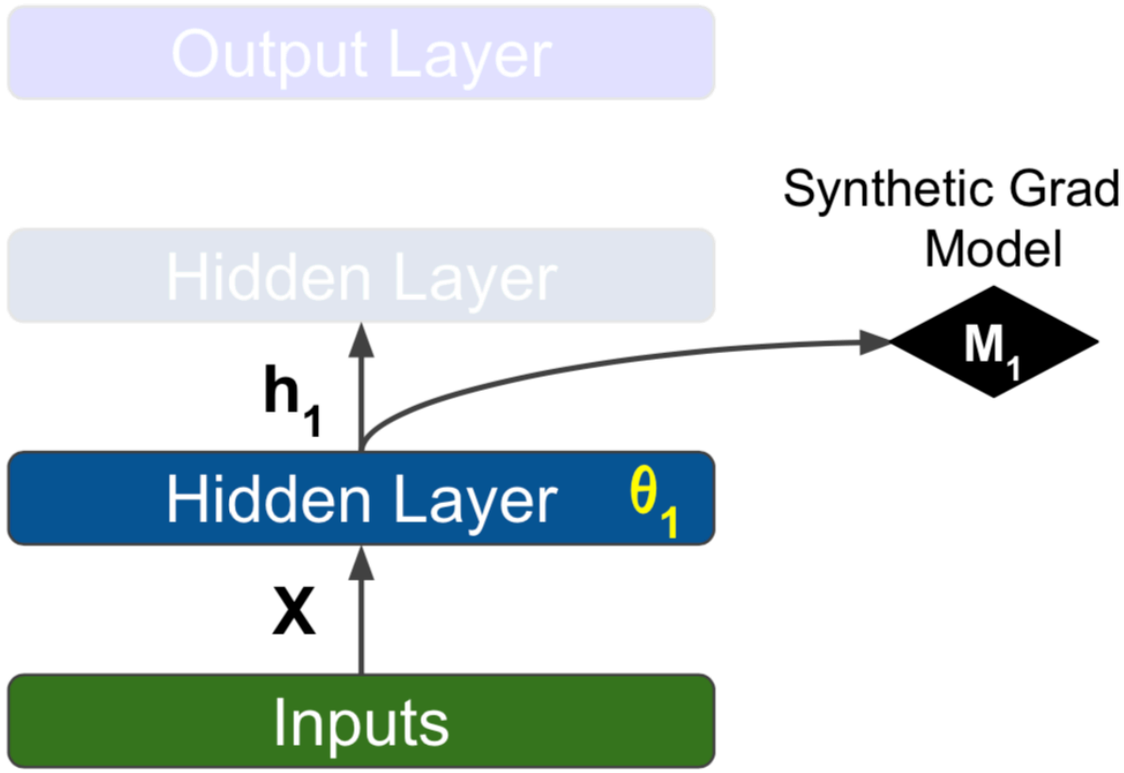
Decoupled Neural Interface



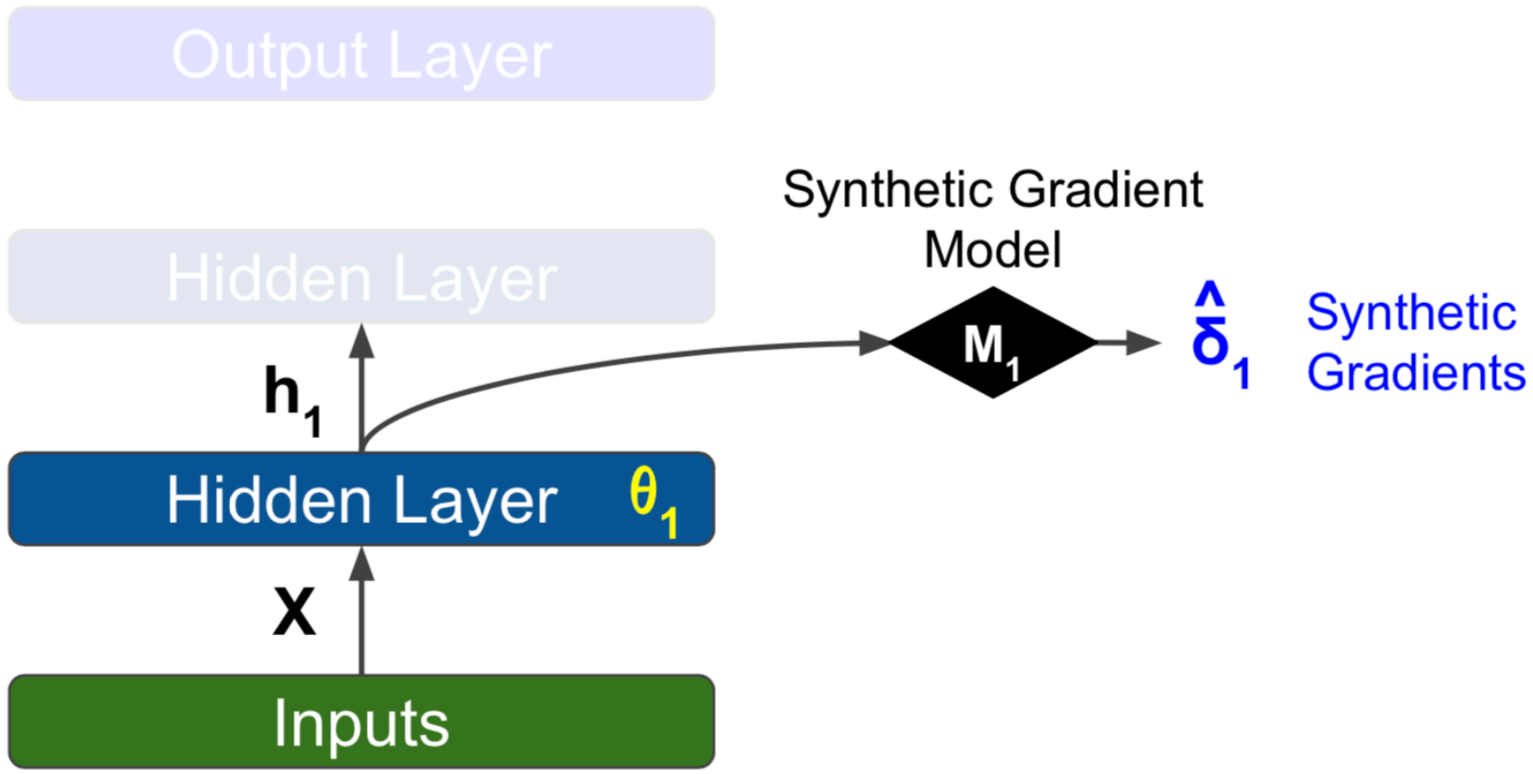
Decoupled Neural Interface



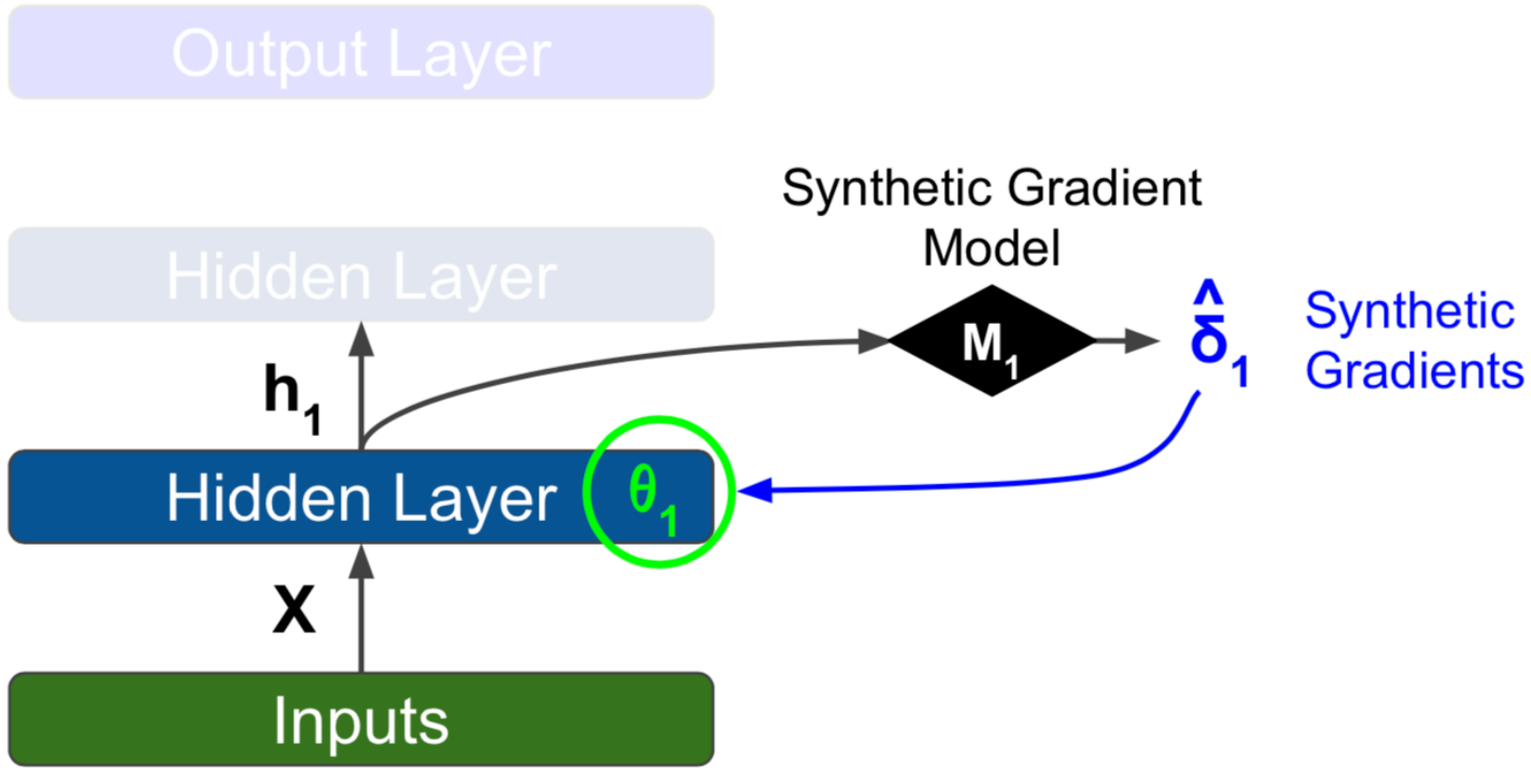
Decoupled Neural Interface



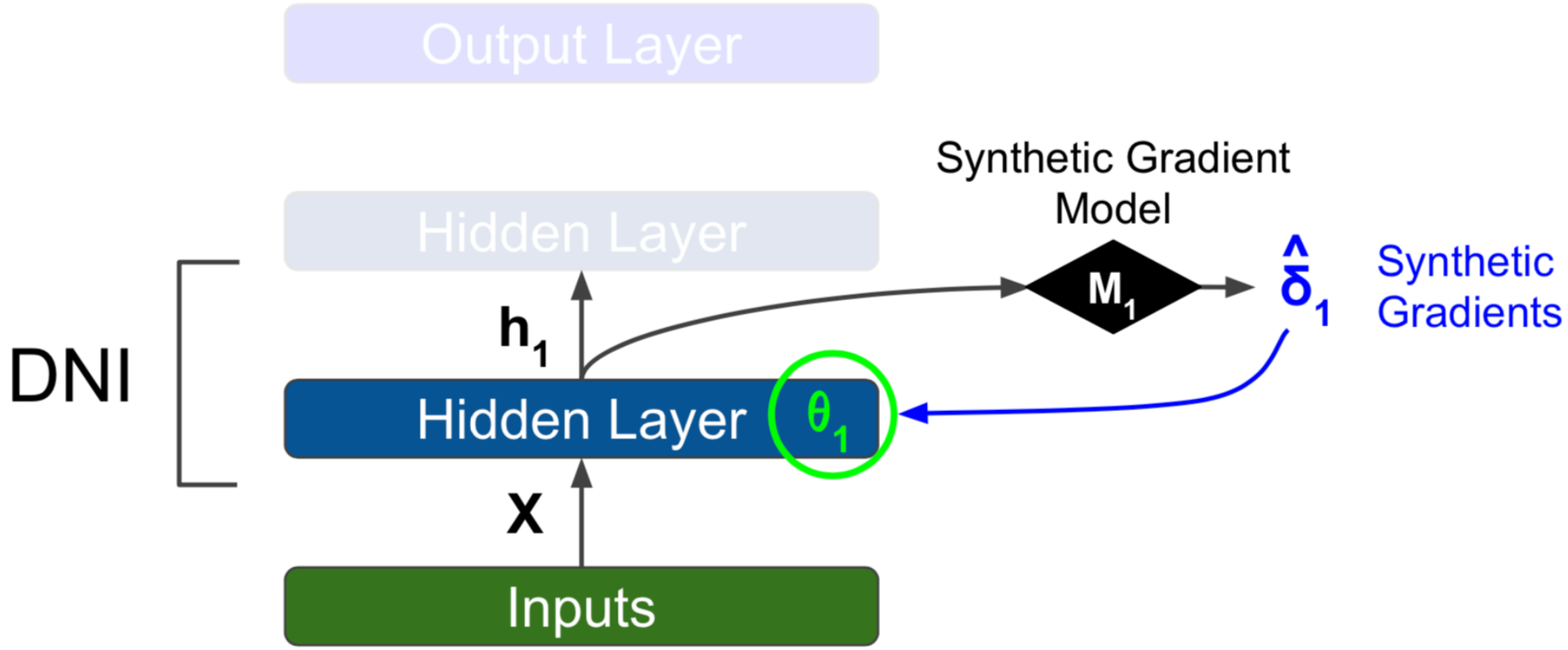
Decoupled Neural Interface



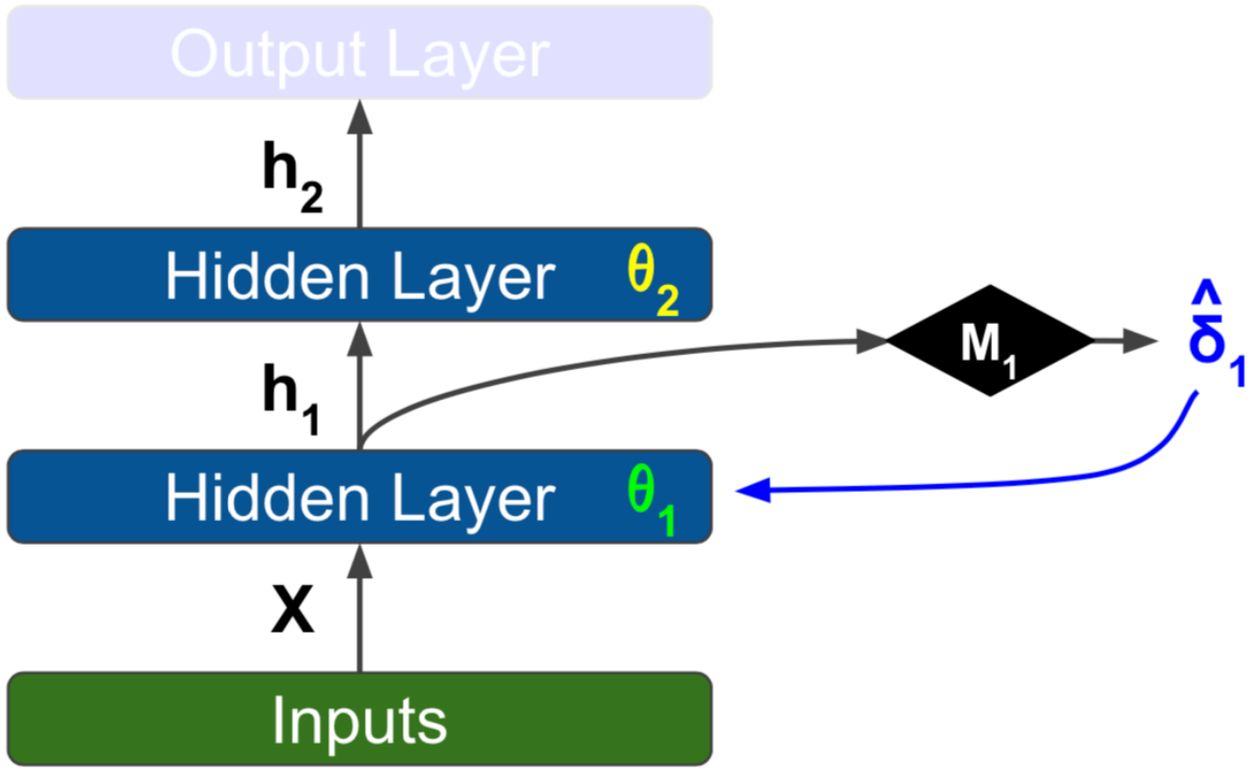
Decoupled Neural Interface



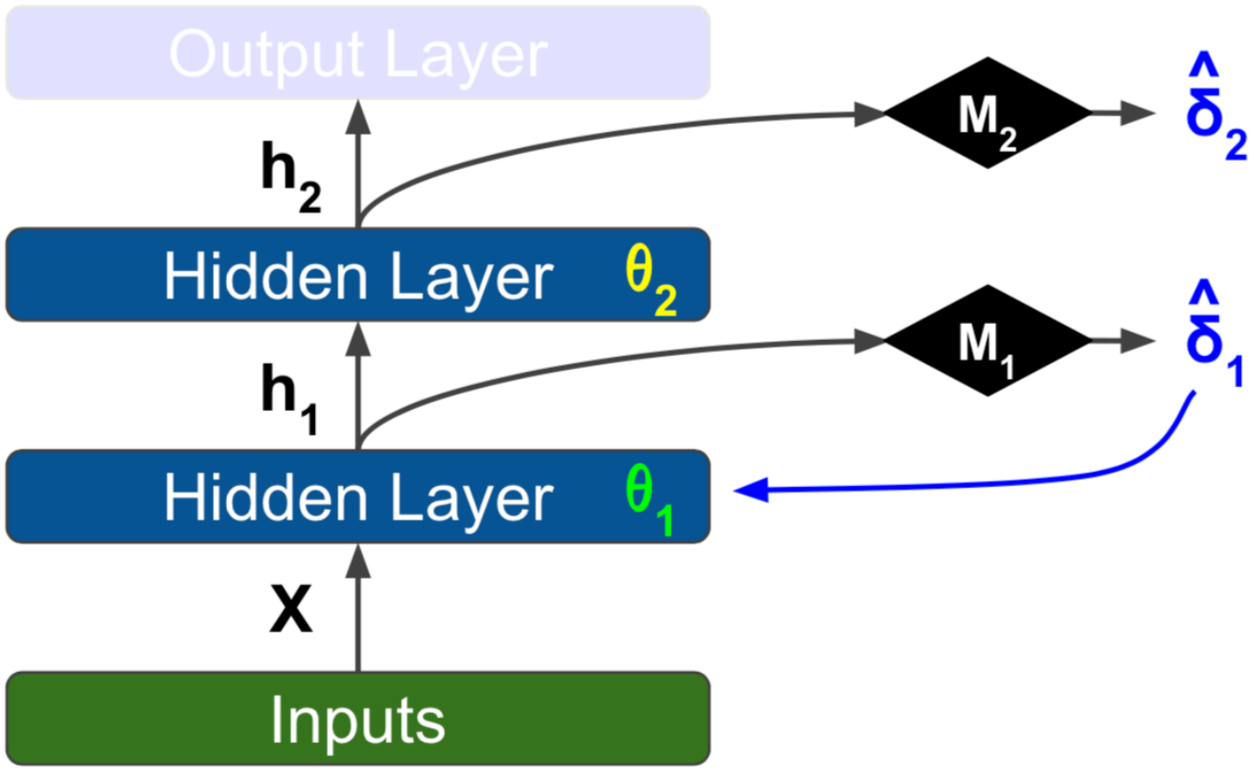
Decoupled Neural Interface



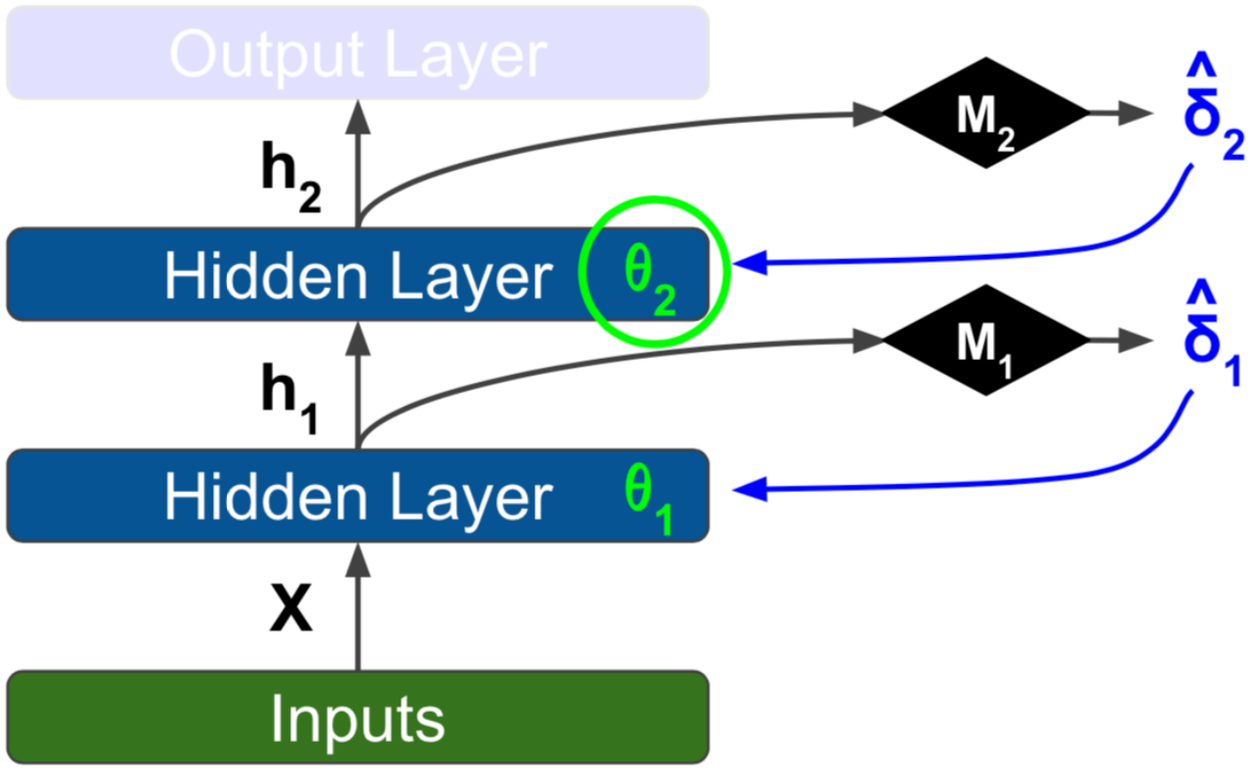
Decoupled Neural Interface



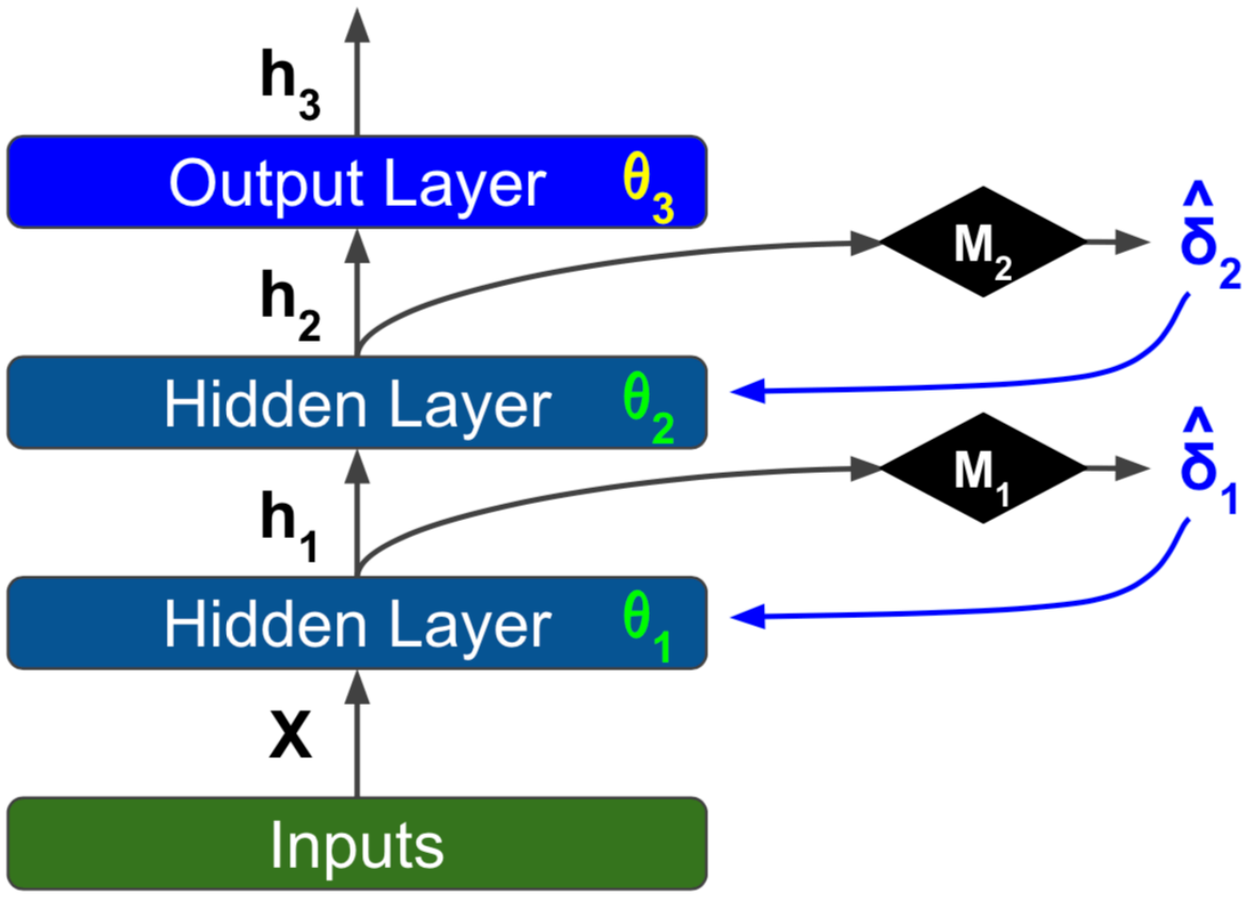
Decoupled Neural Interface



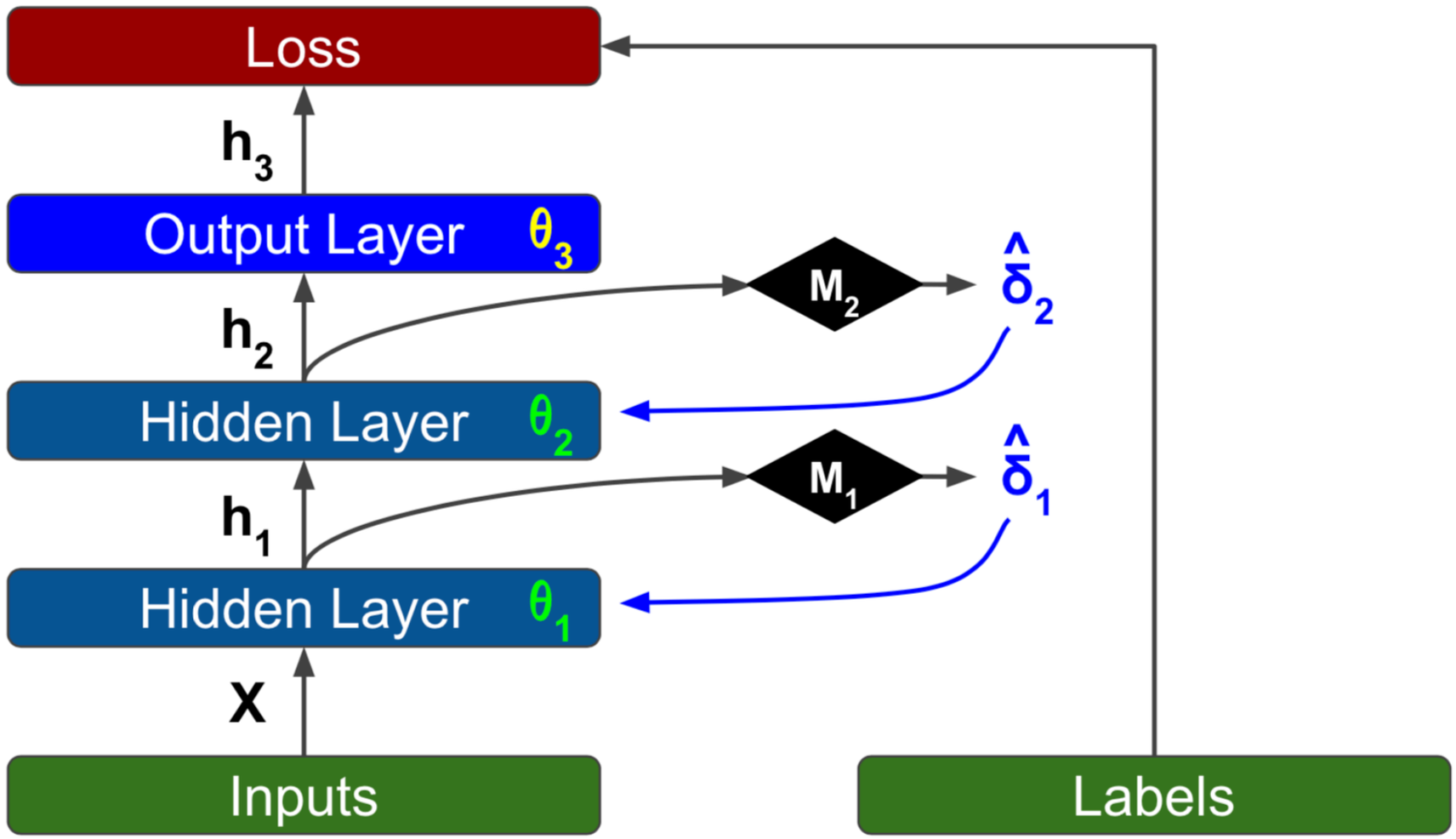
Decoupled Neural Interface



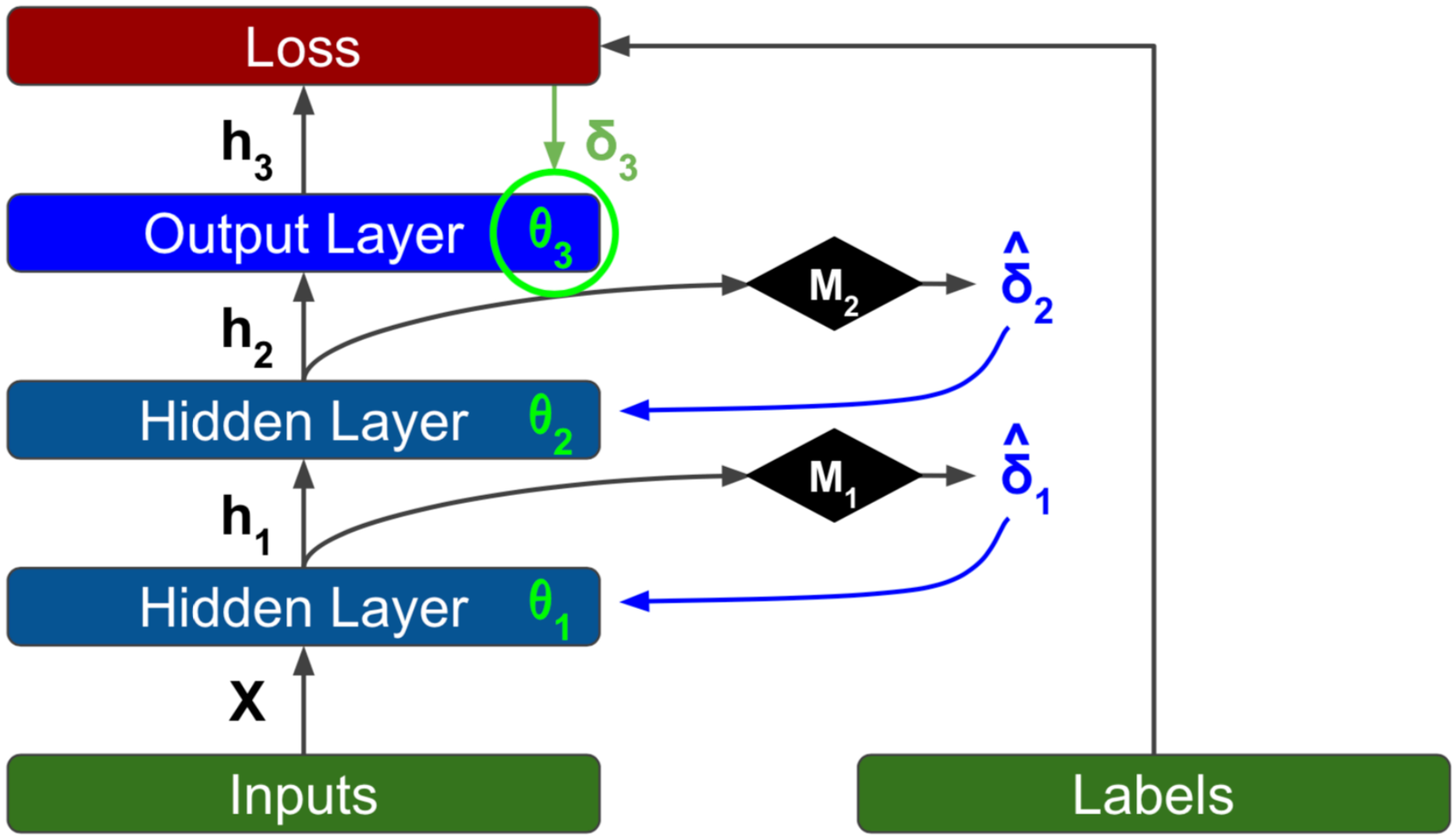
Decoupled Neural Interface



Decoupled Neural Interface



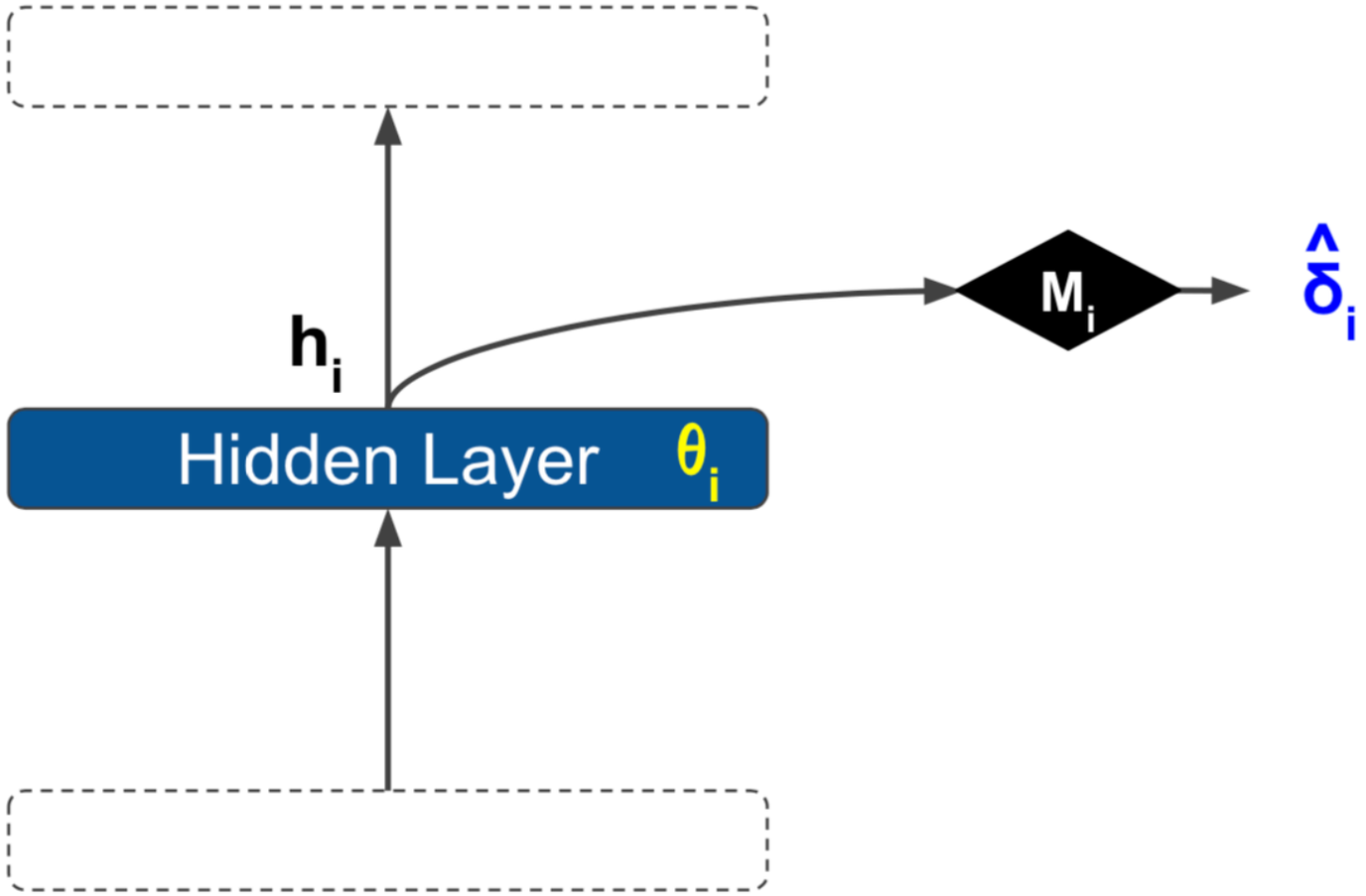
Decoupled Neural Interface



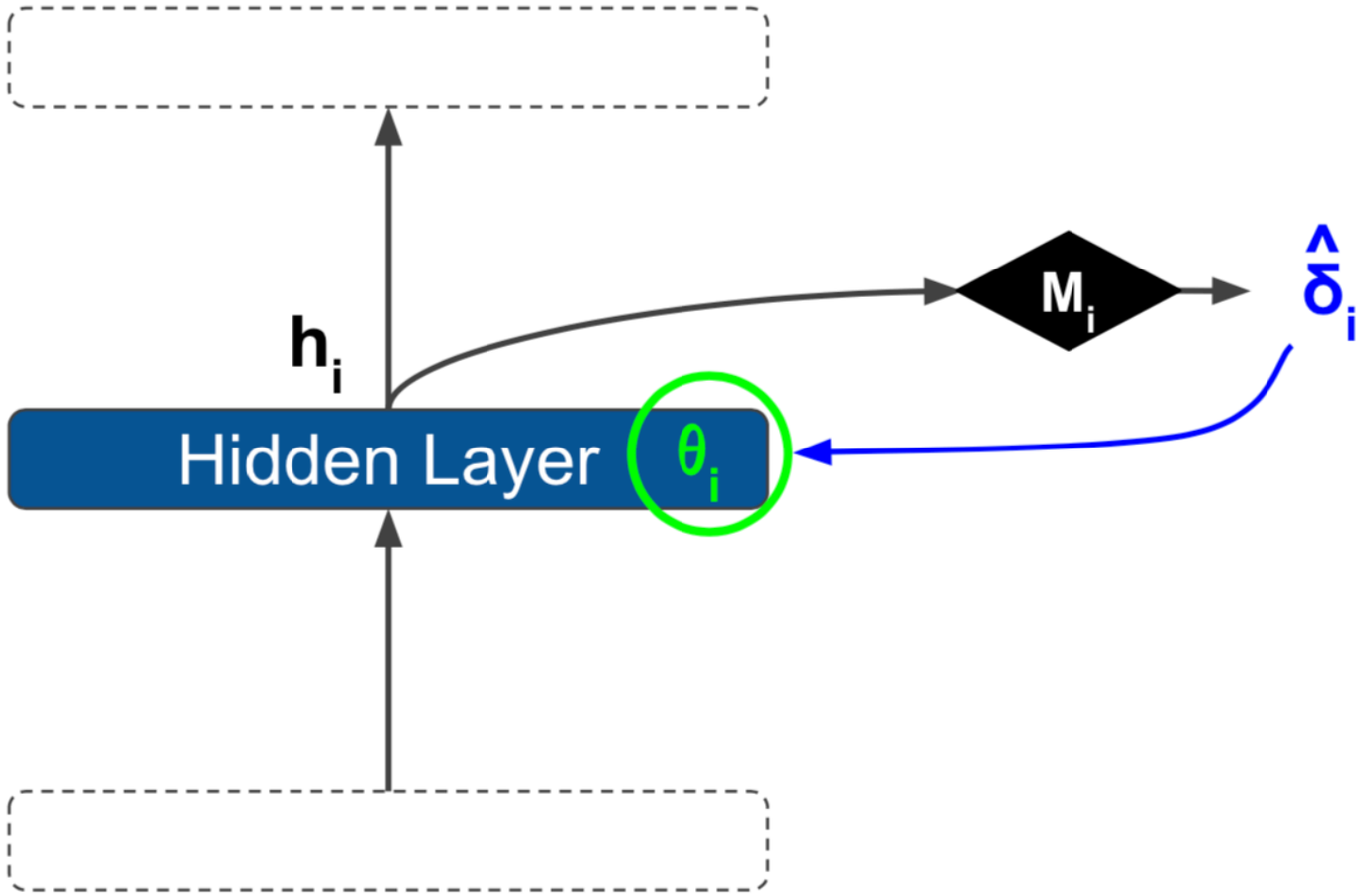
Decoupled Neural Interface

Only for training!

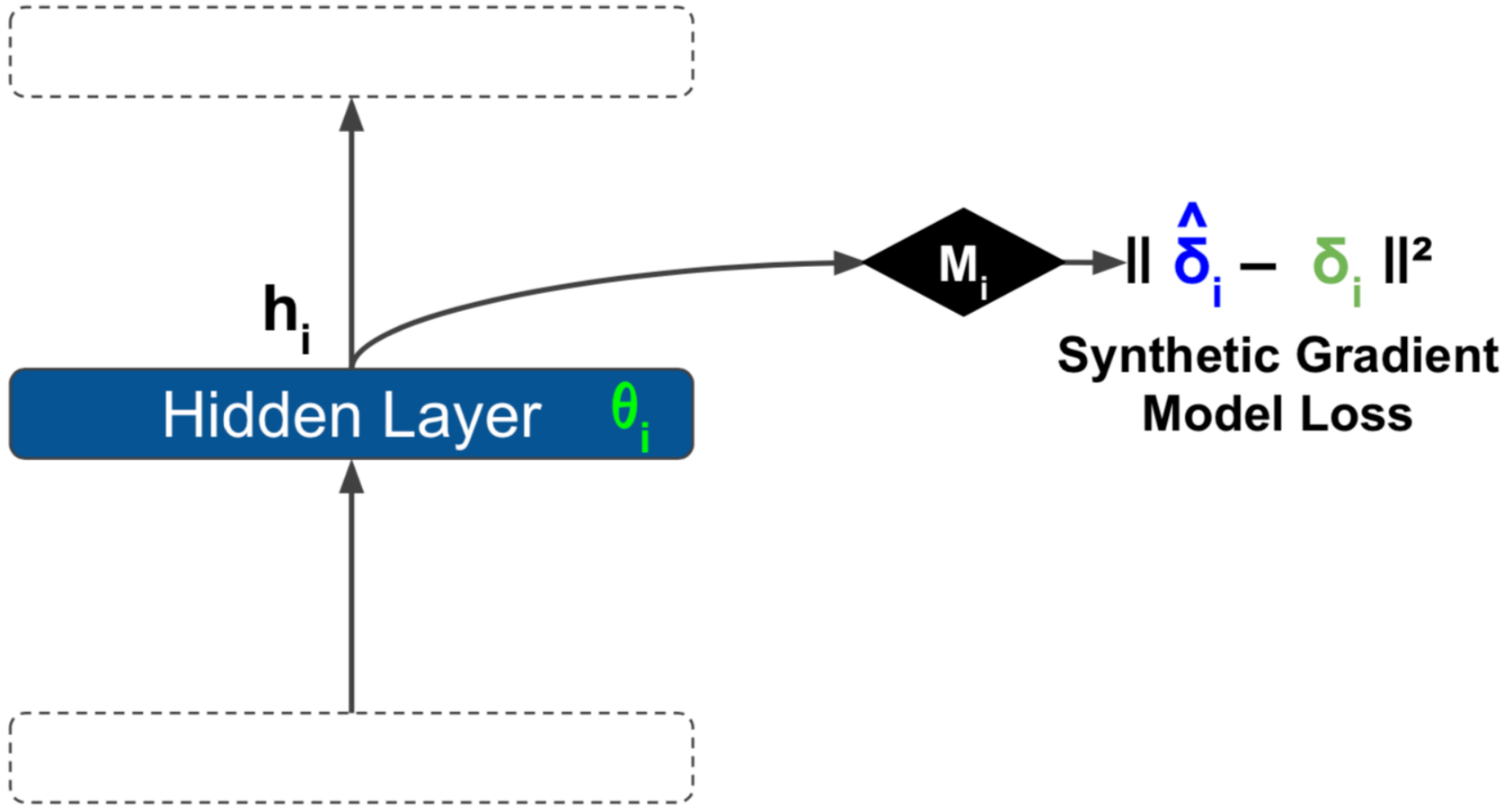
Training a SG model



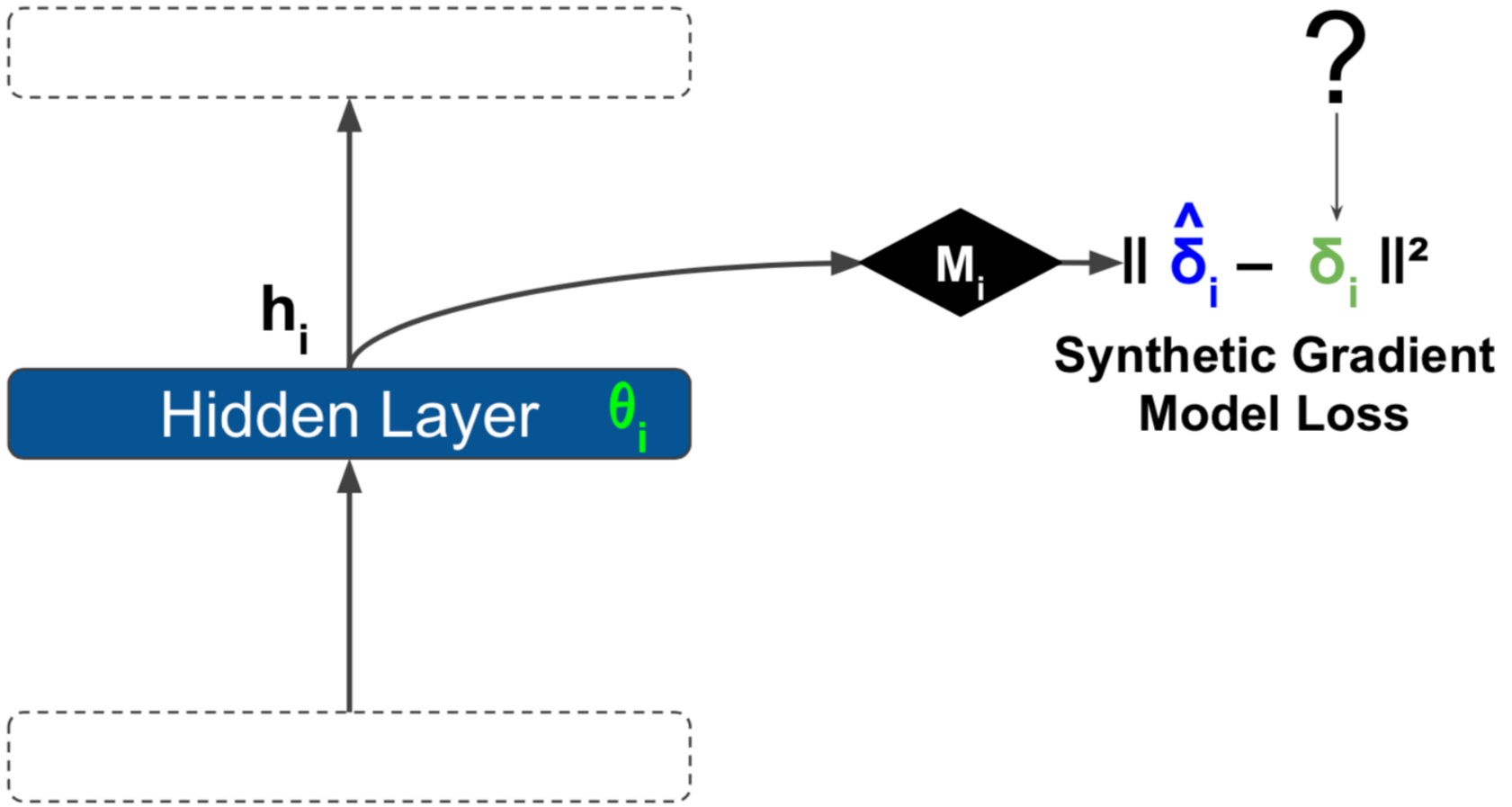
Training a SG model



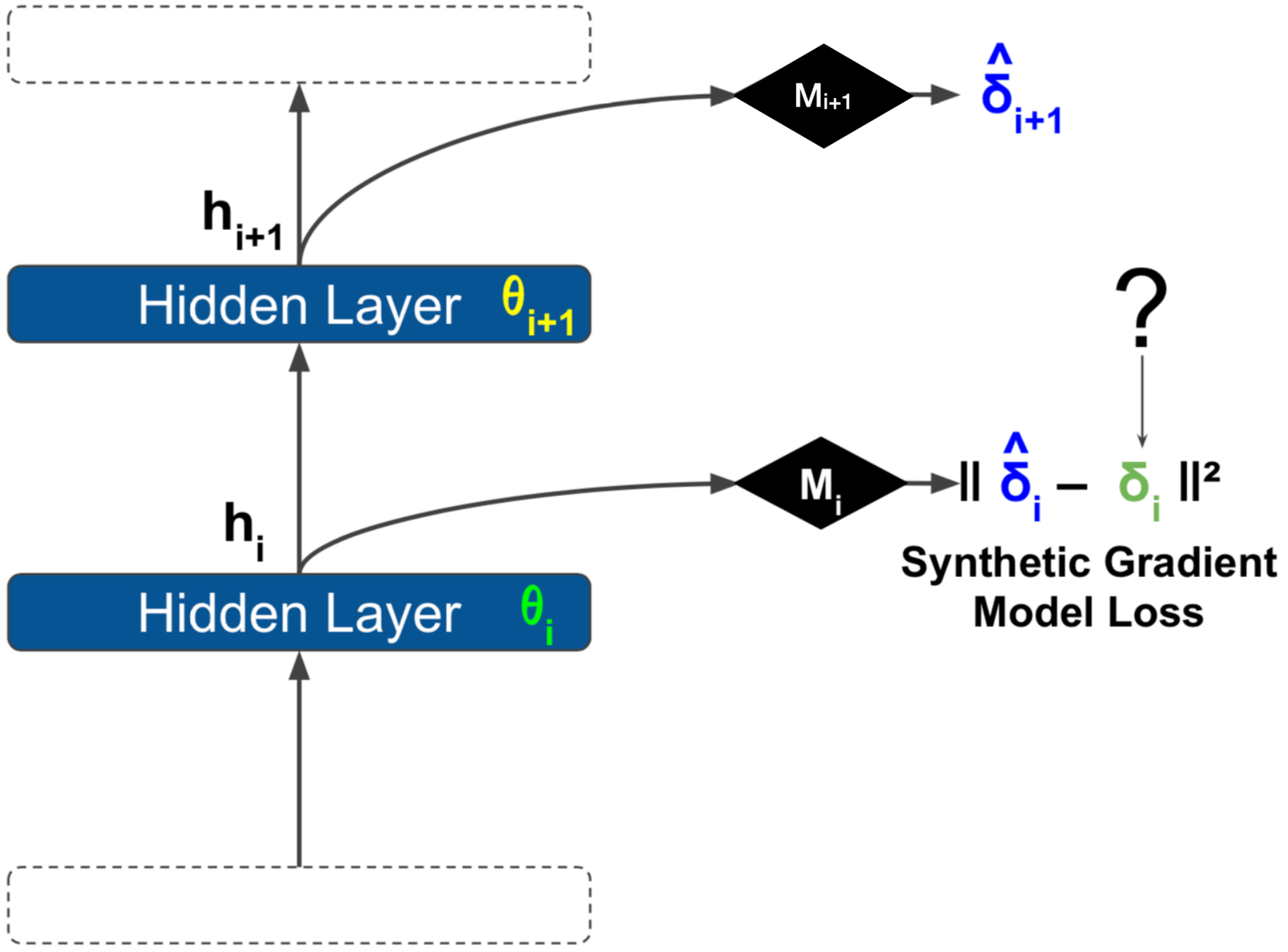
Training a SG model



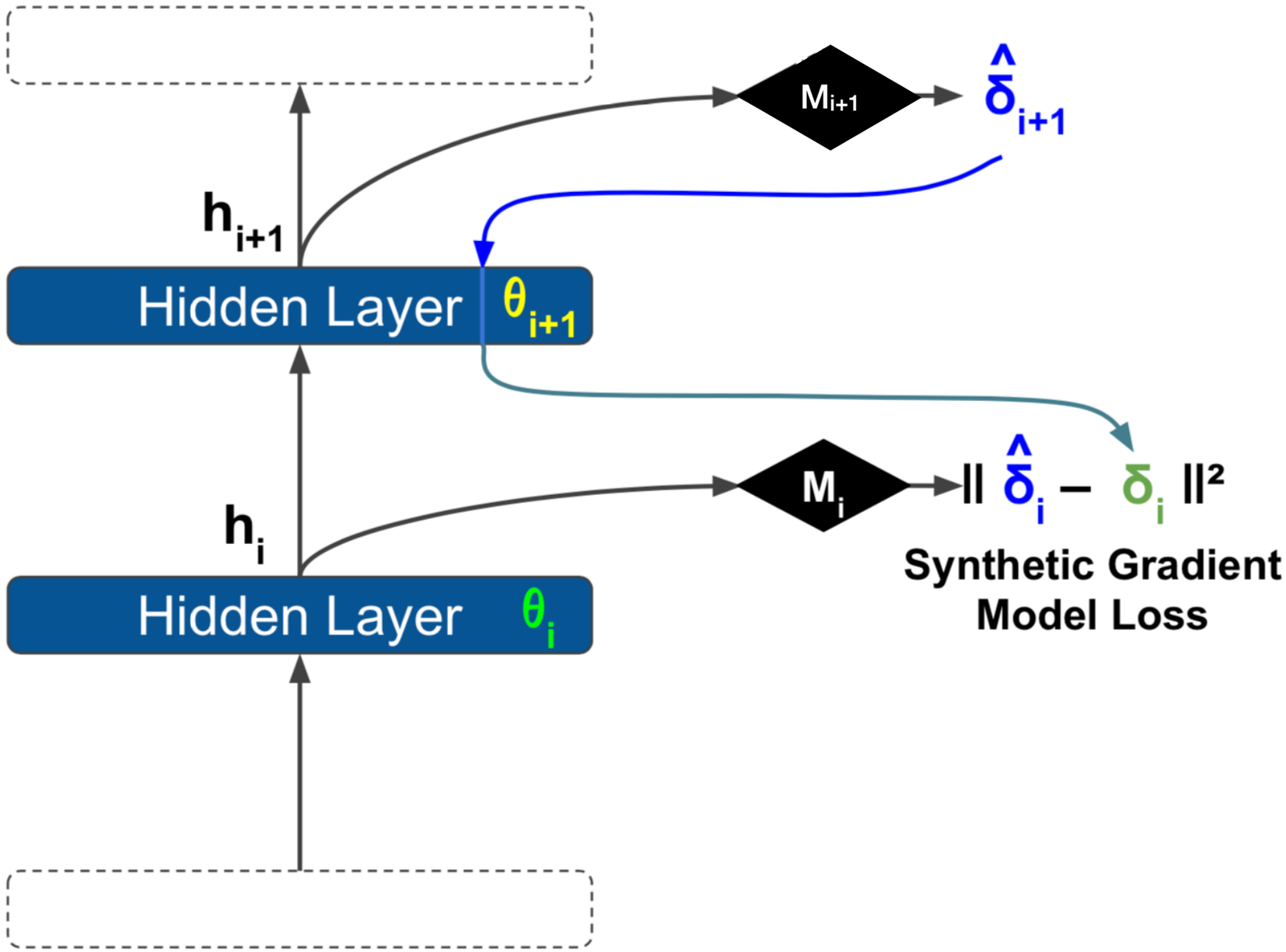
Training a SG model



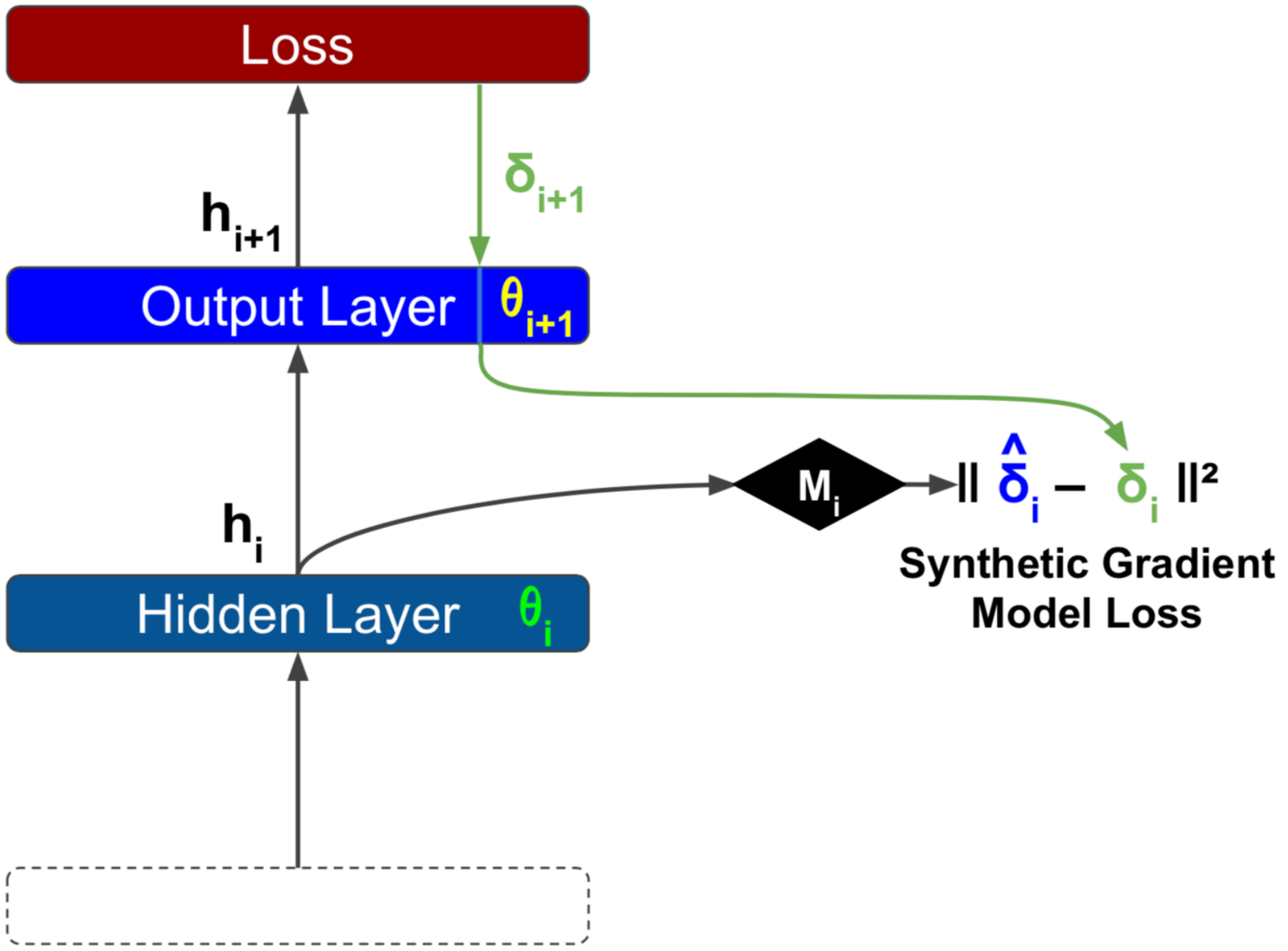
Training a SG model



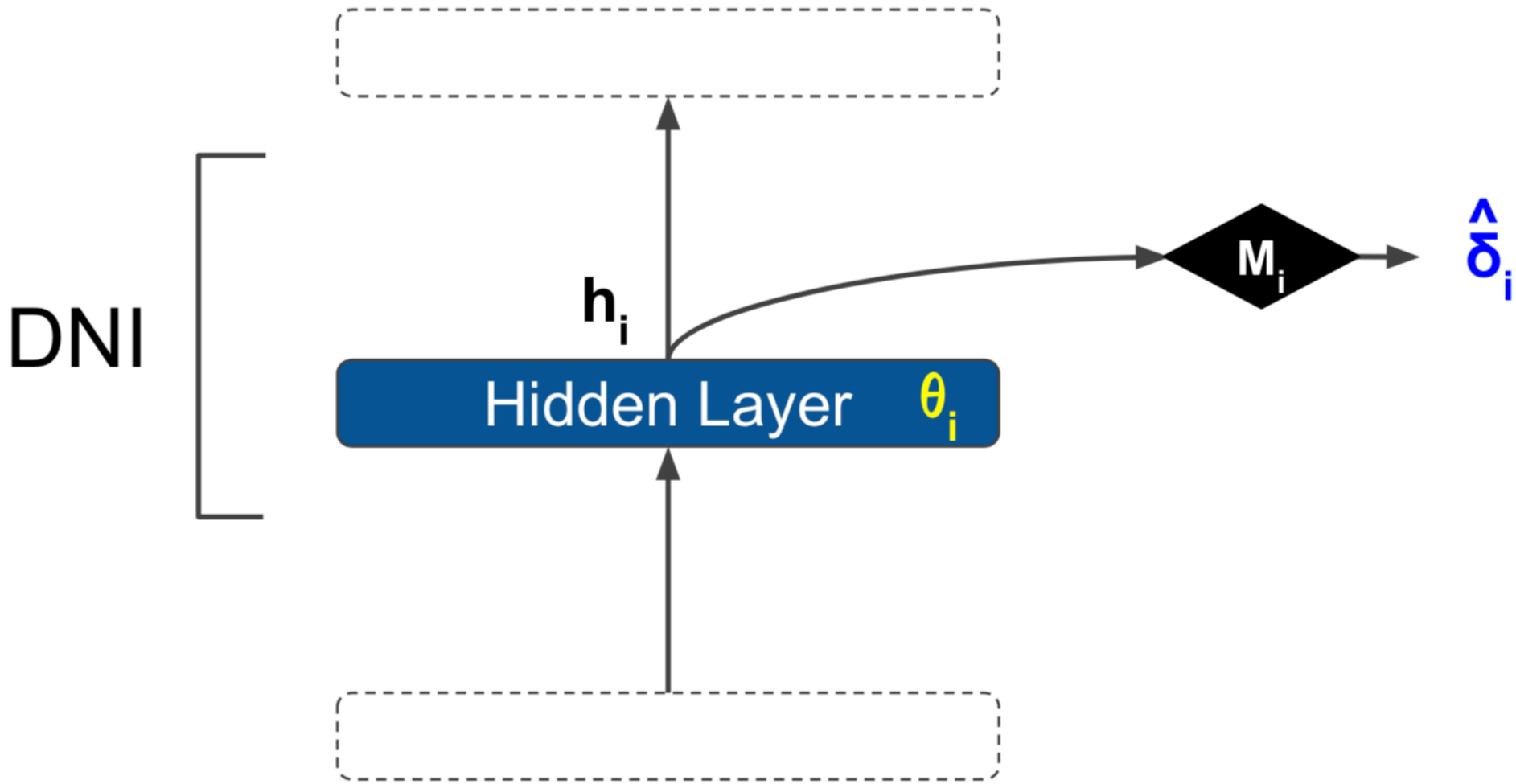
Training a SG model



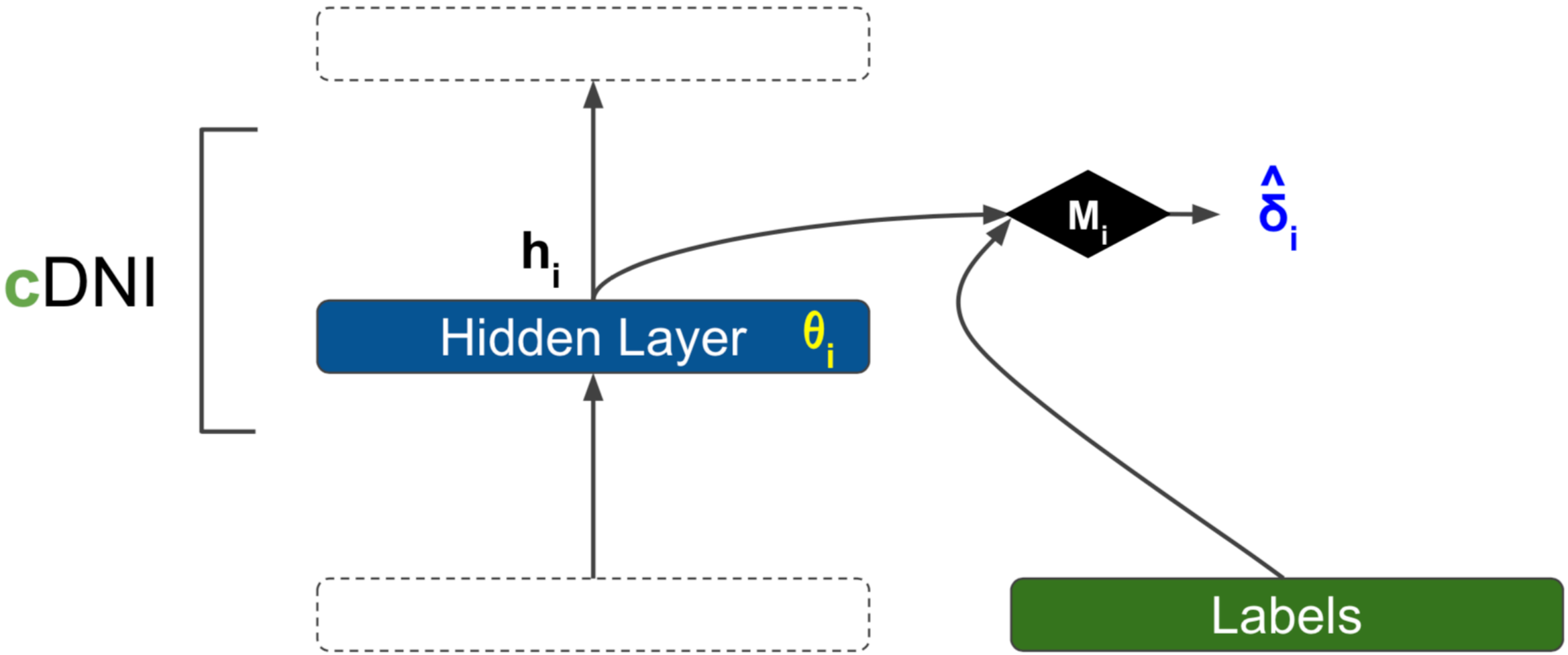
Training a SG model



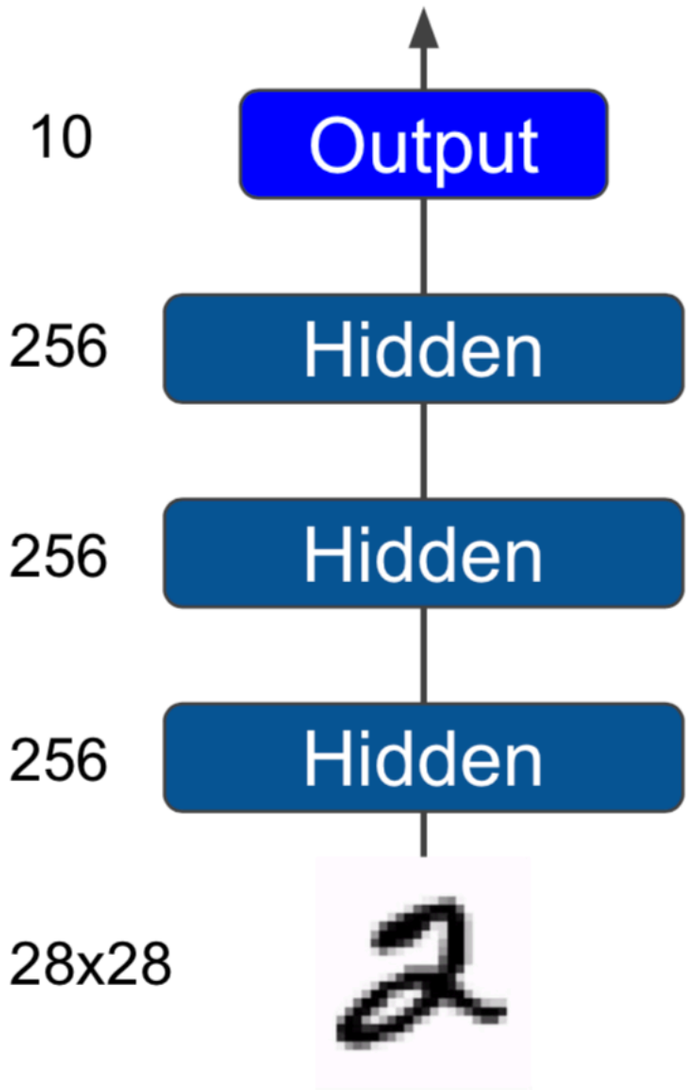
Providing context



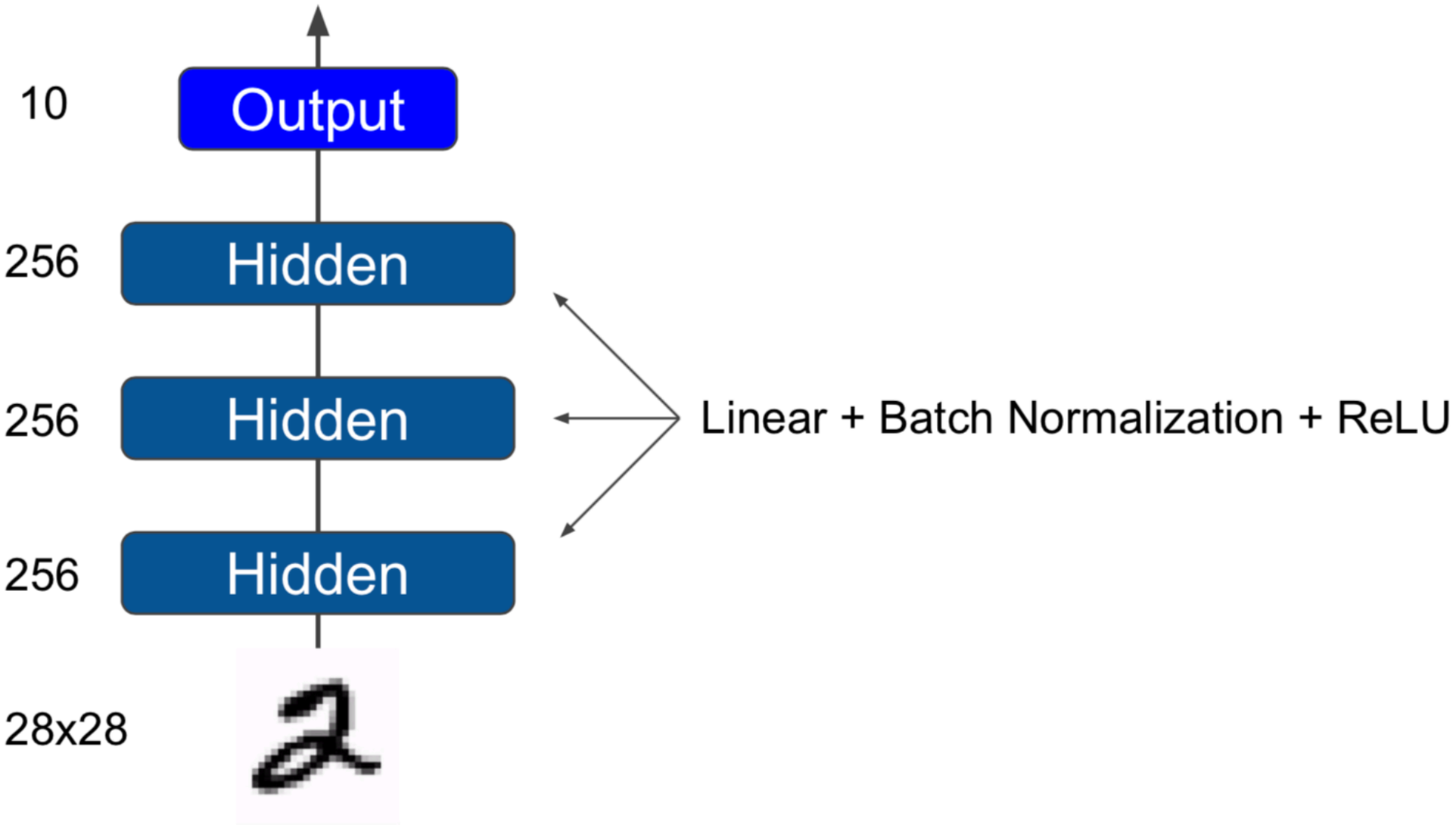
Providing context



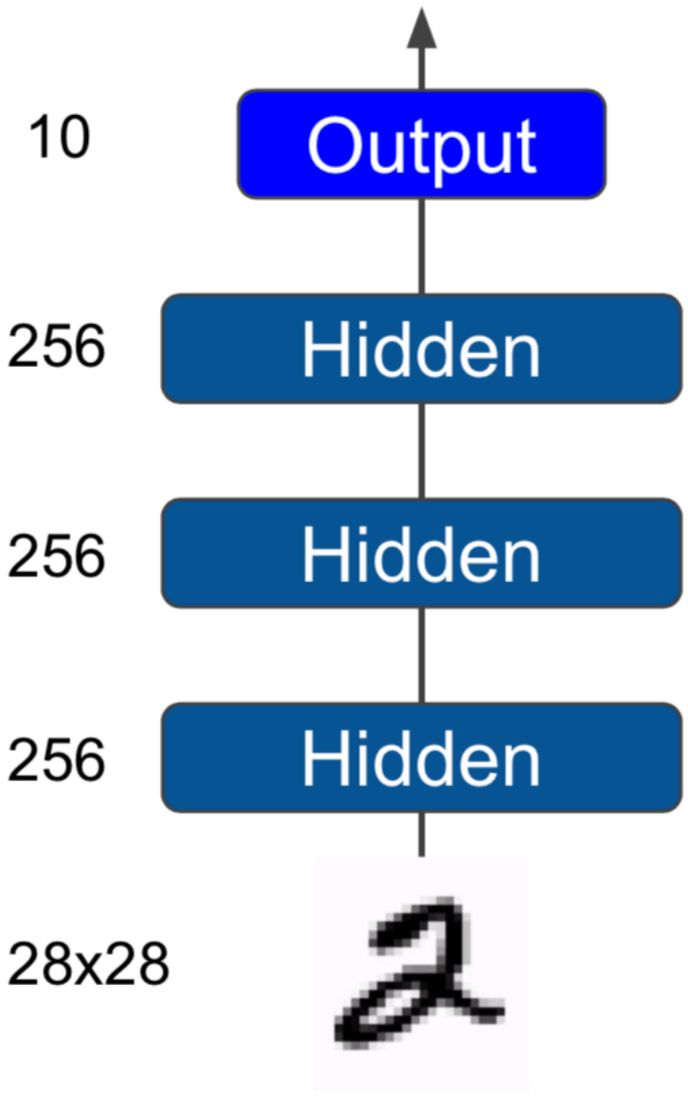
Result on MNIST

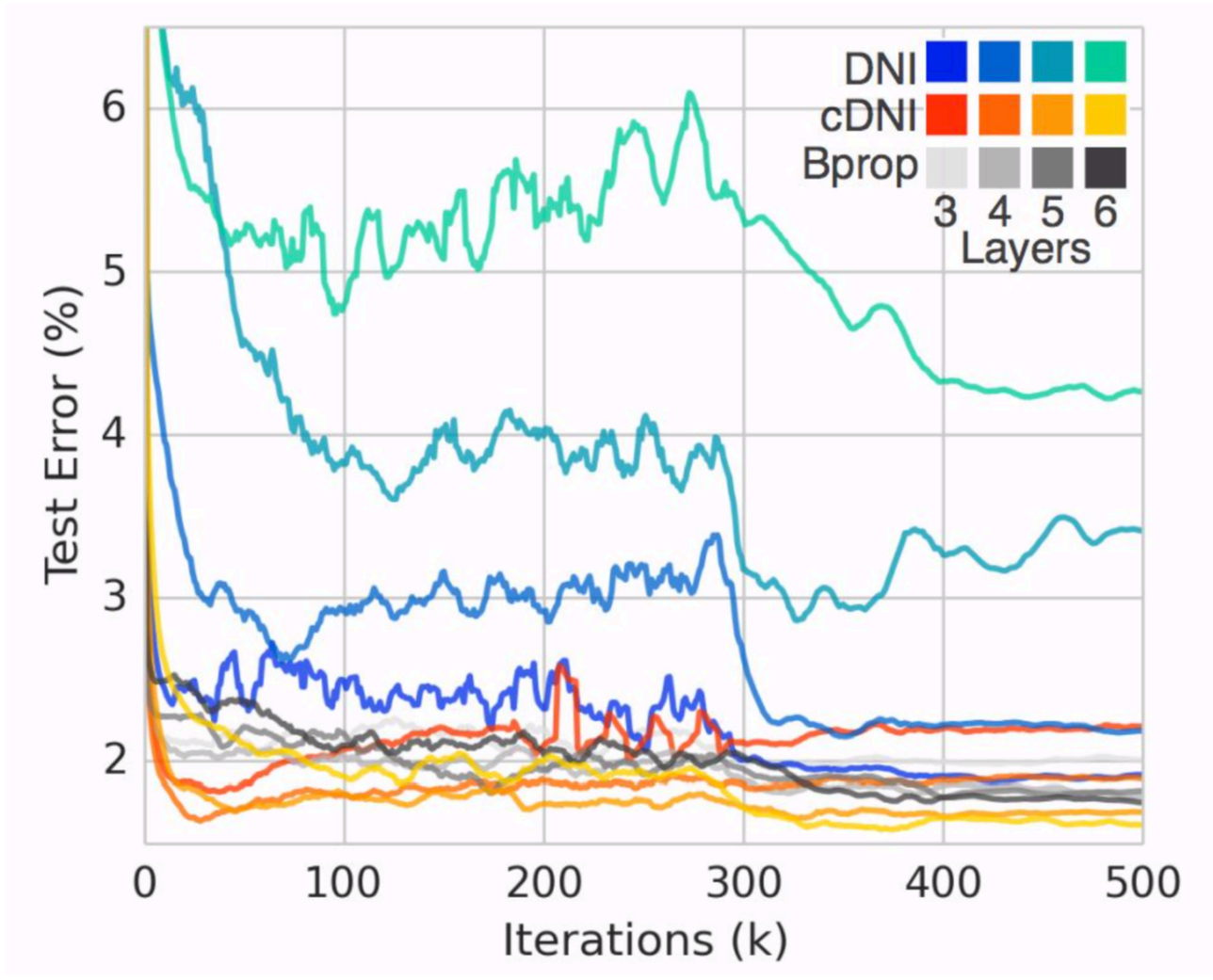


Result on MNIST

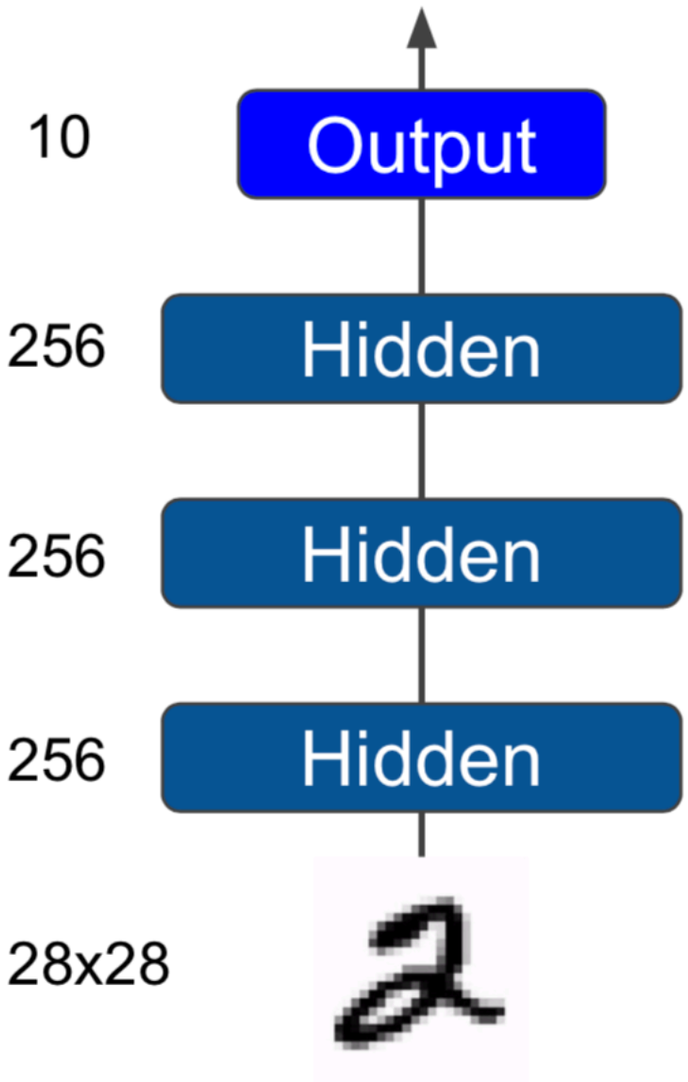


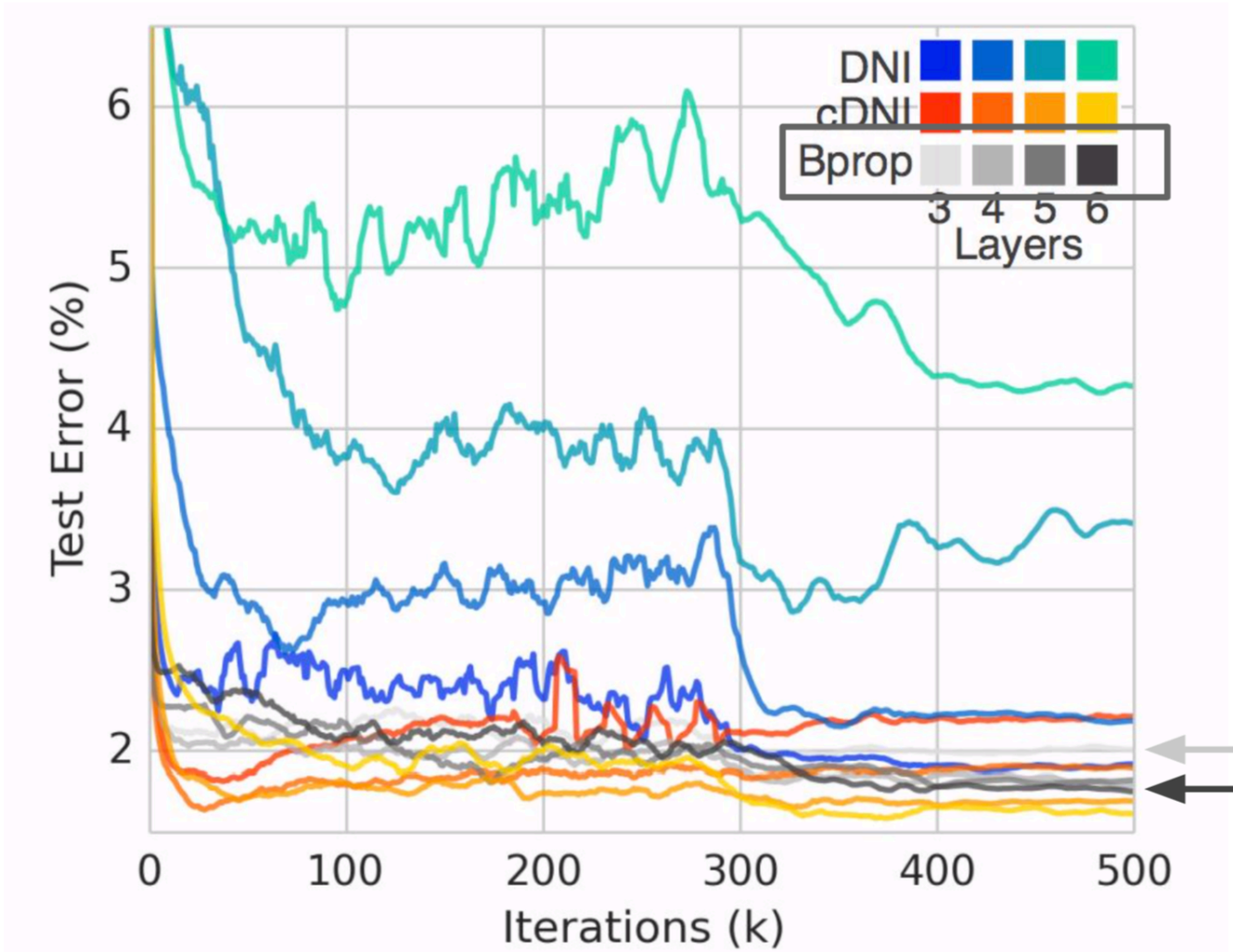
Result on MNIST



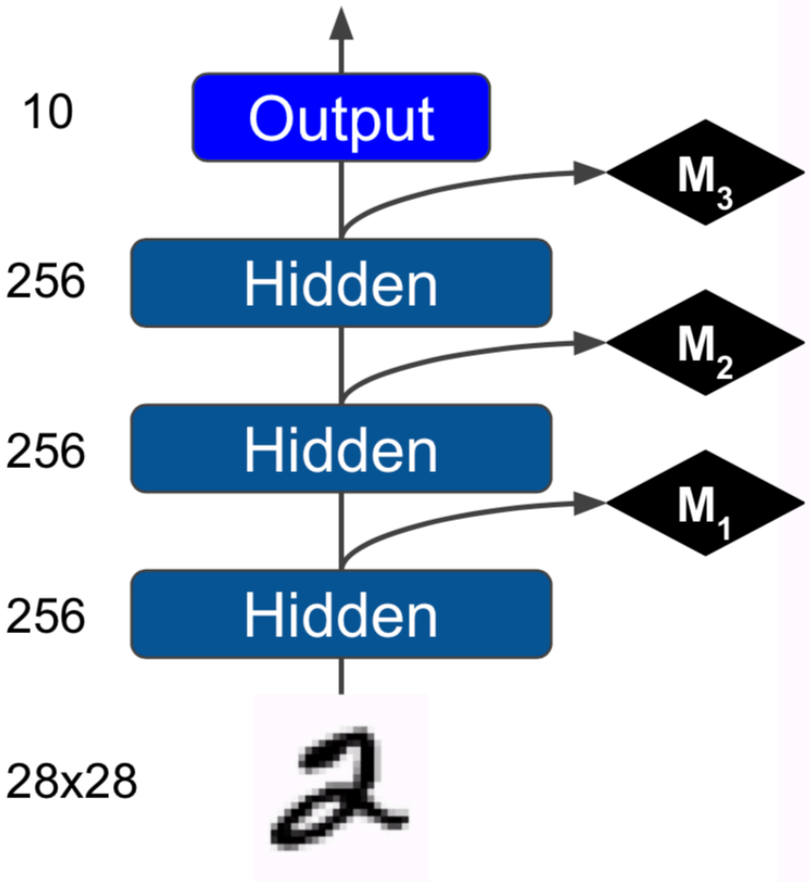


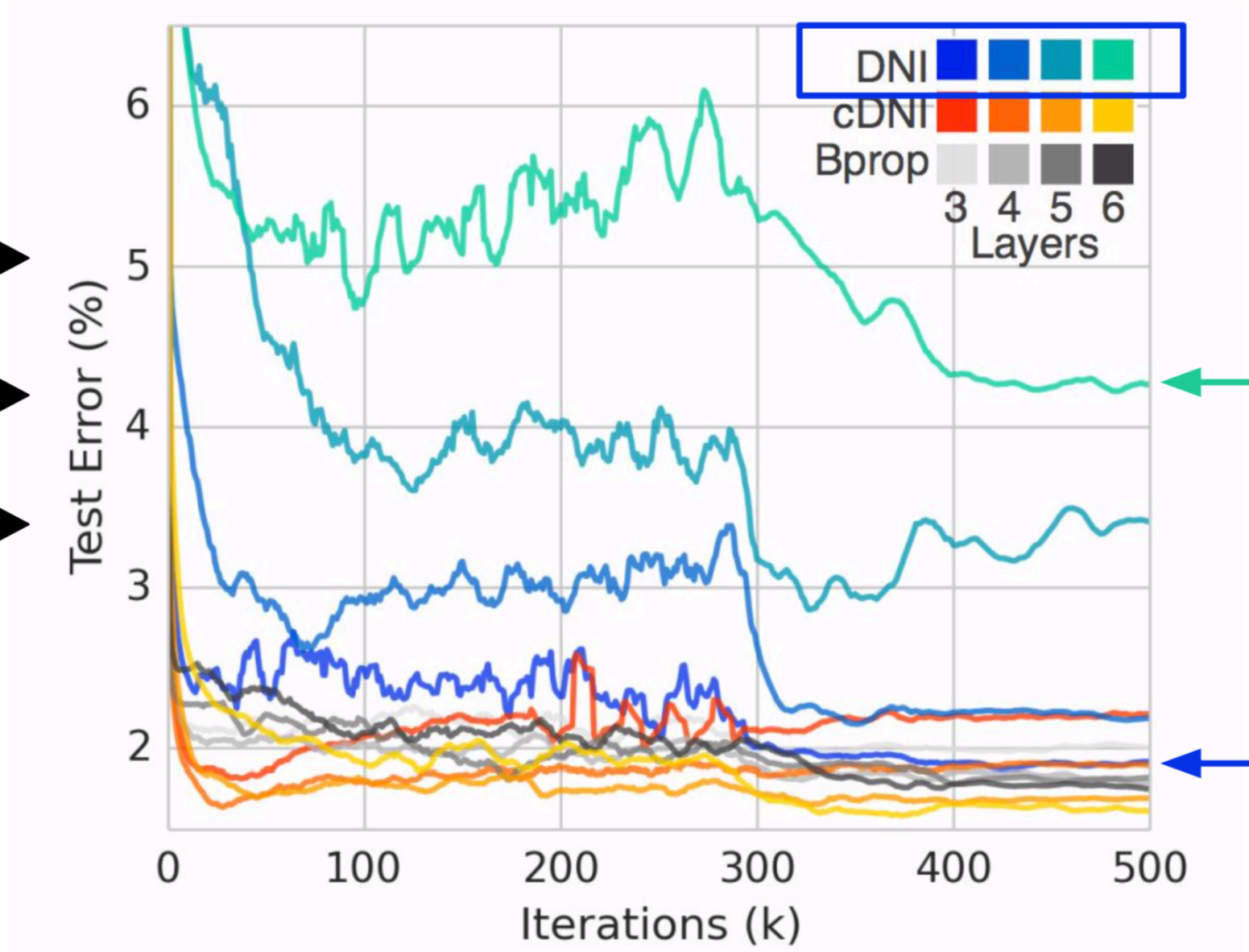
Result on MNIST



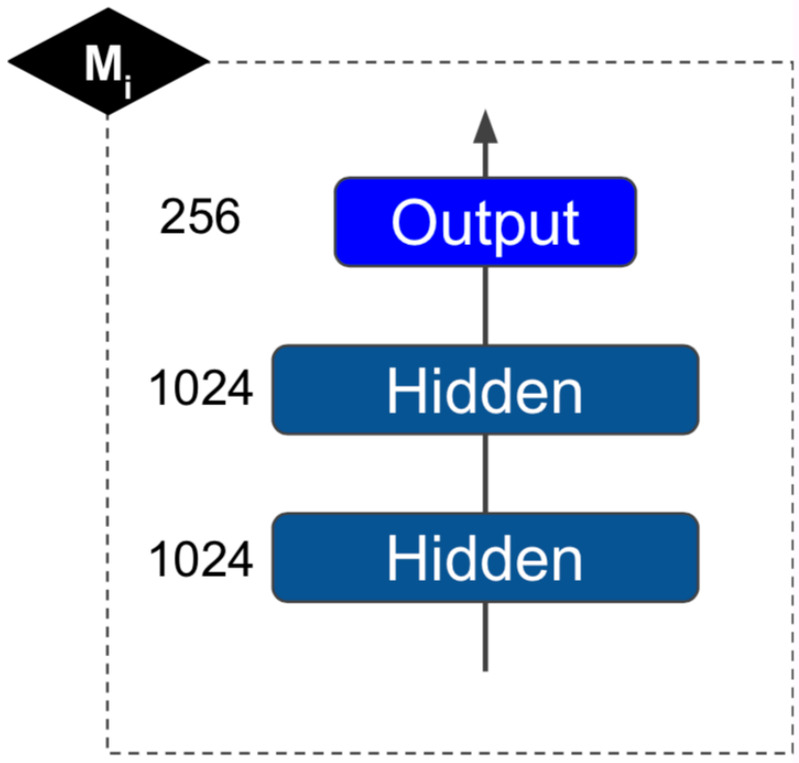


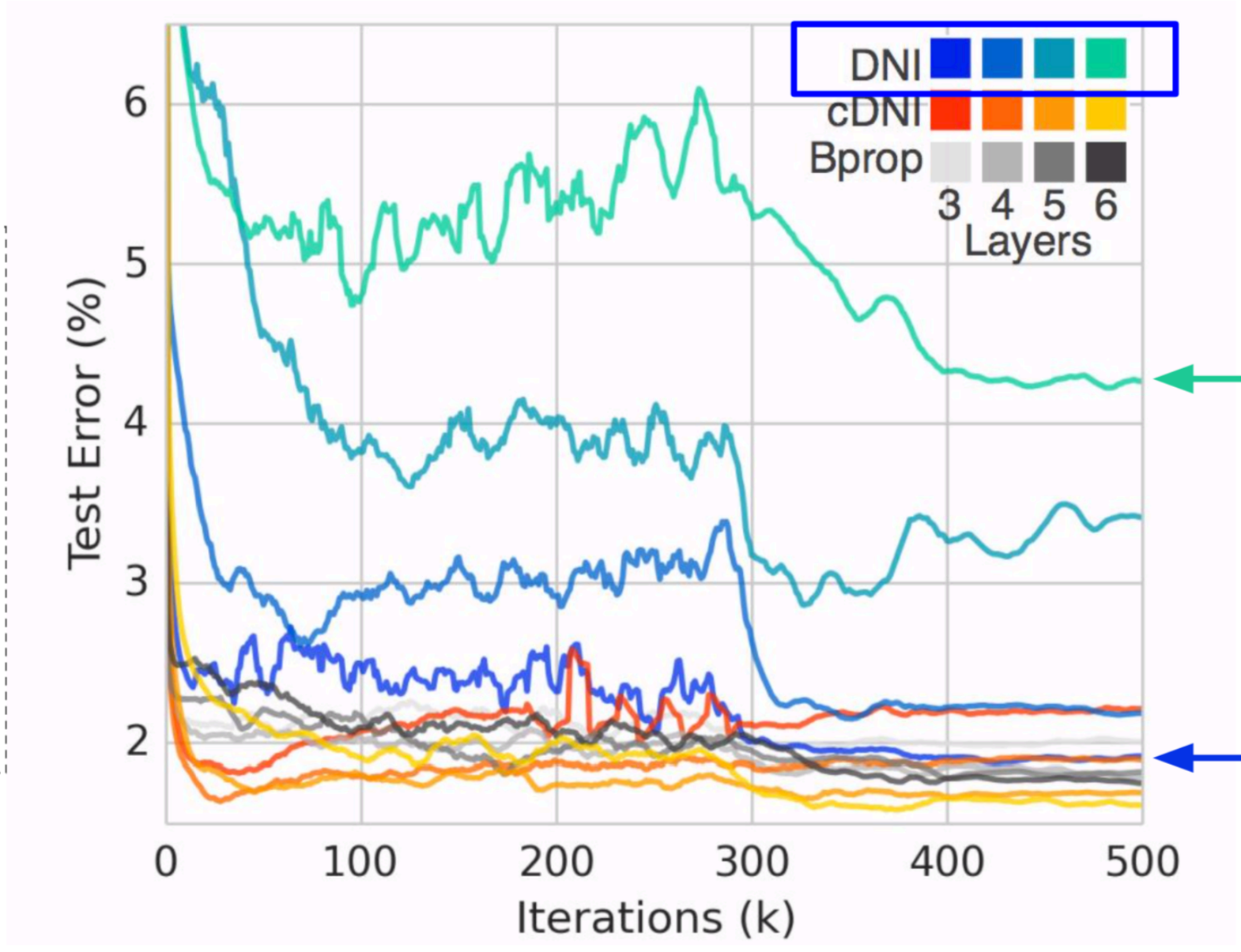
Result on MNIST



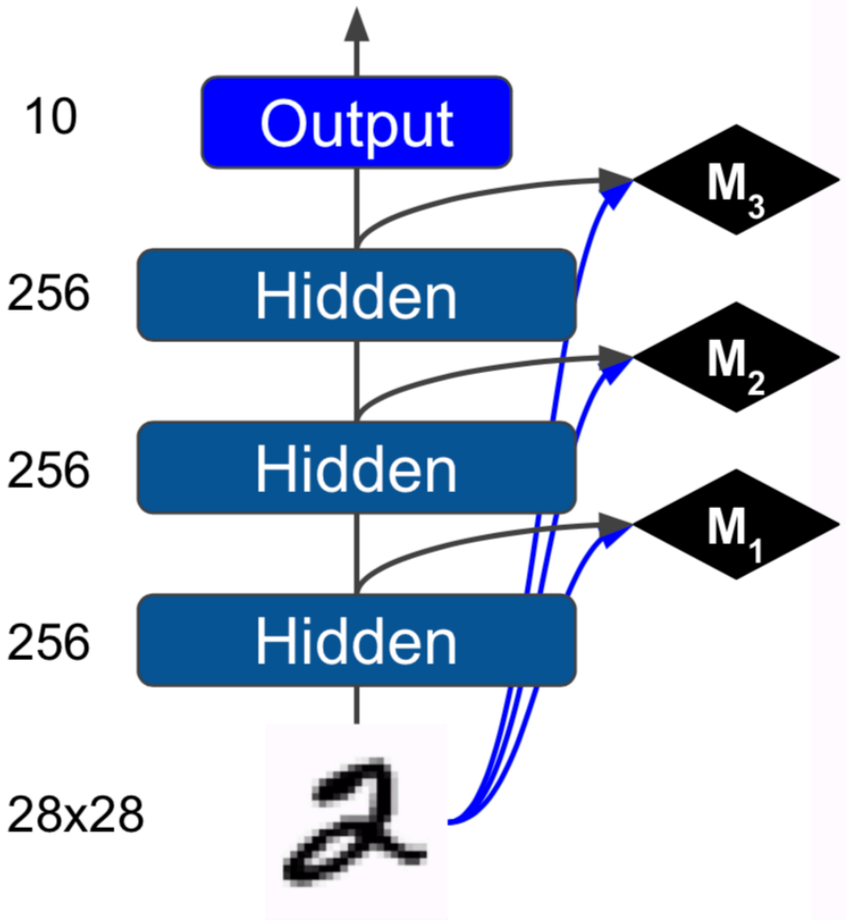


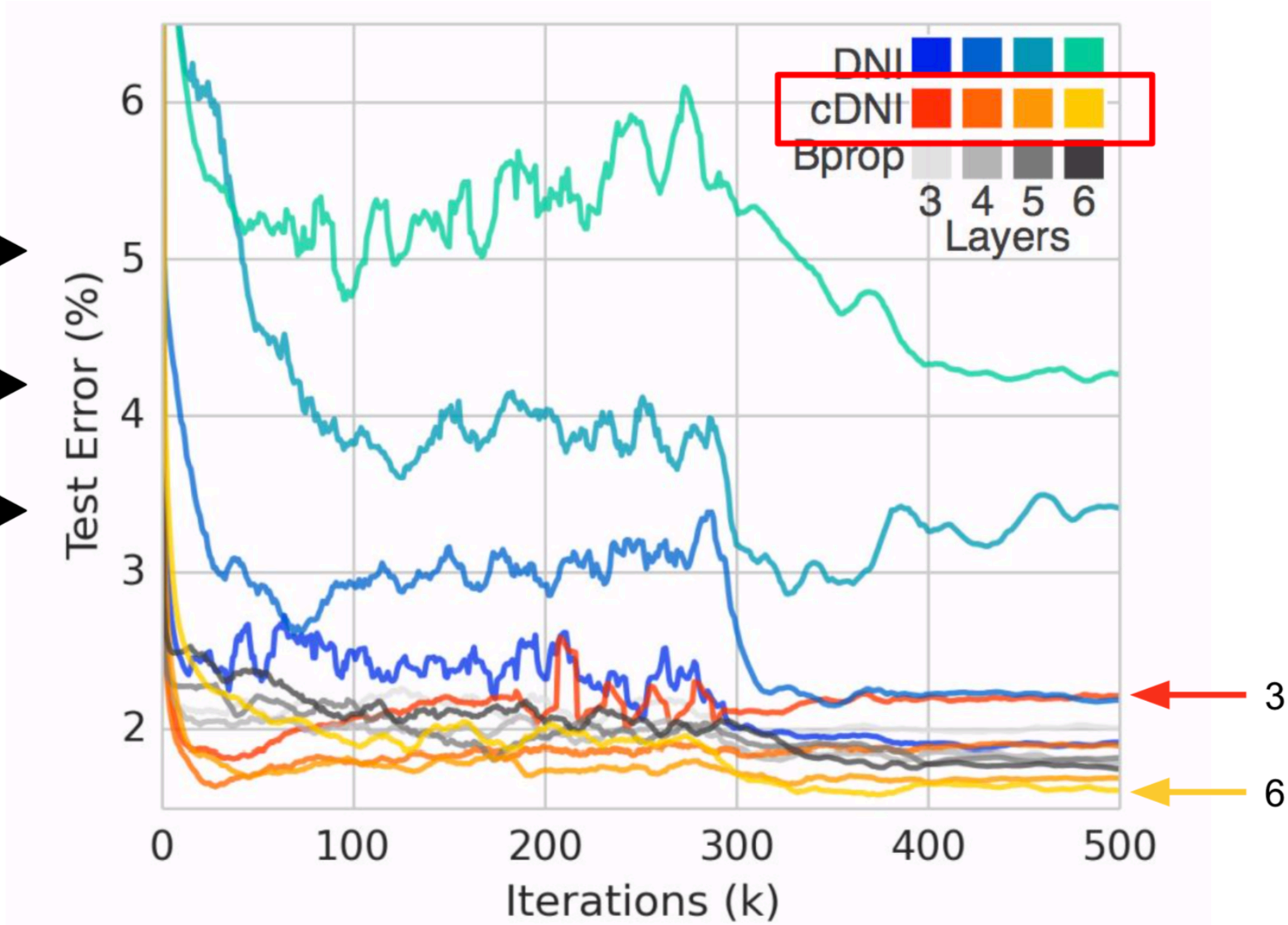
Result on MNIST



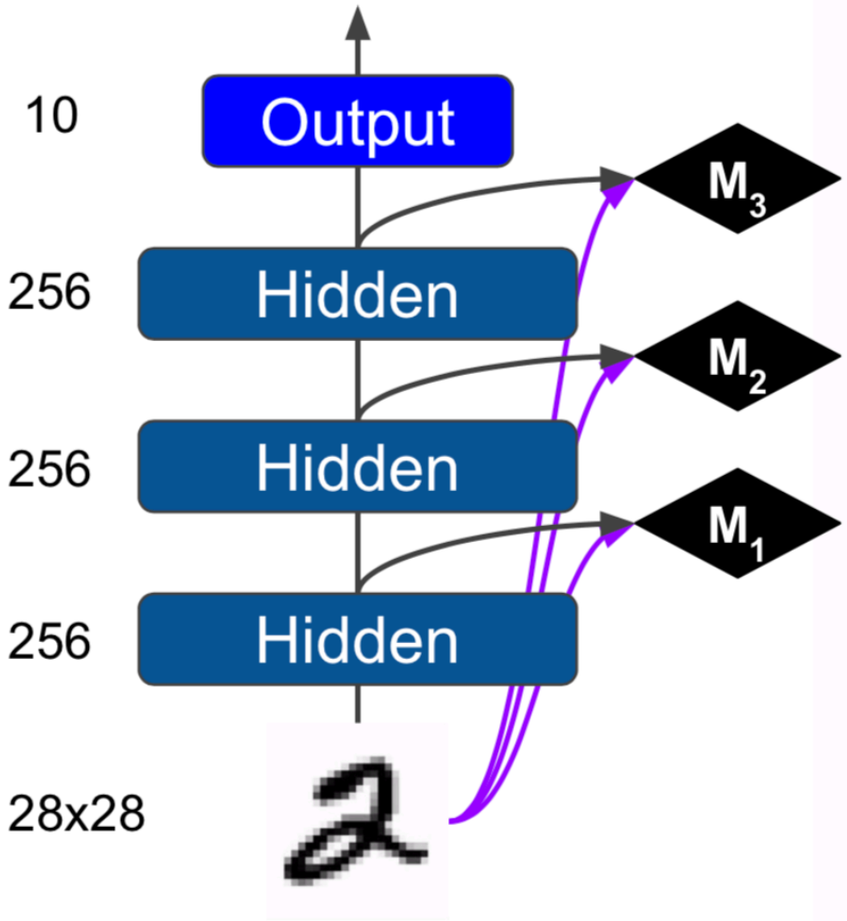


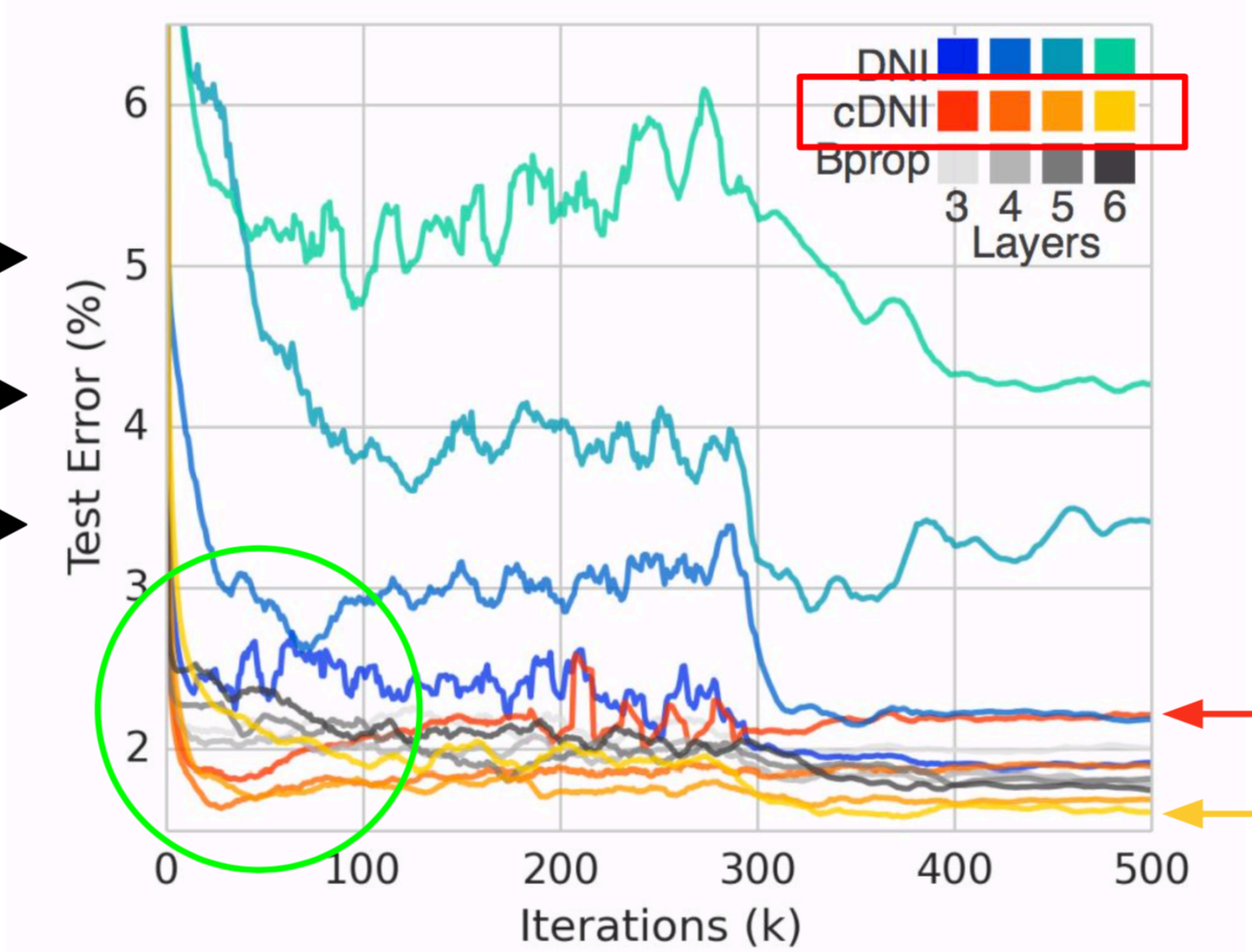
Result on MNIST



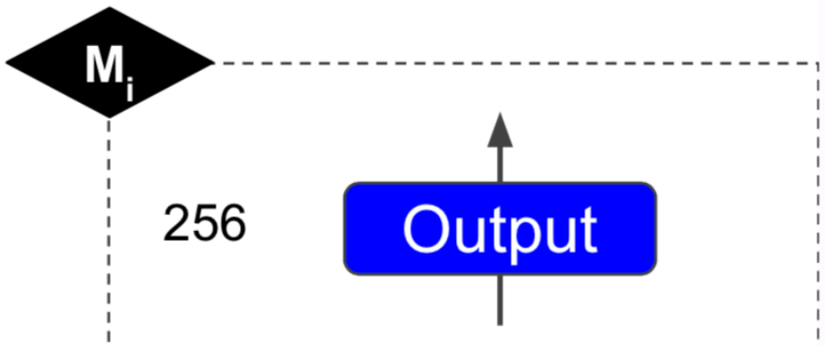


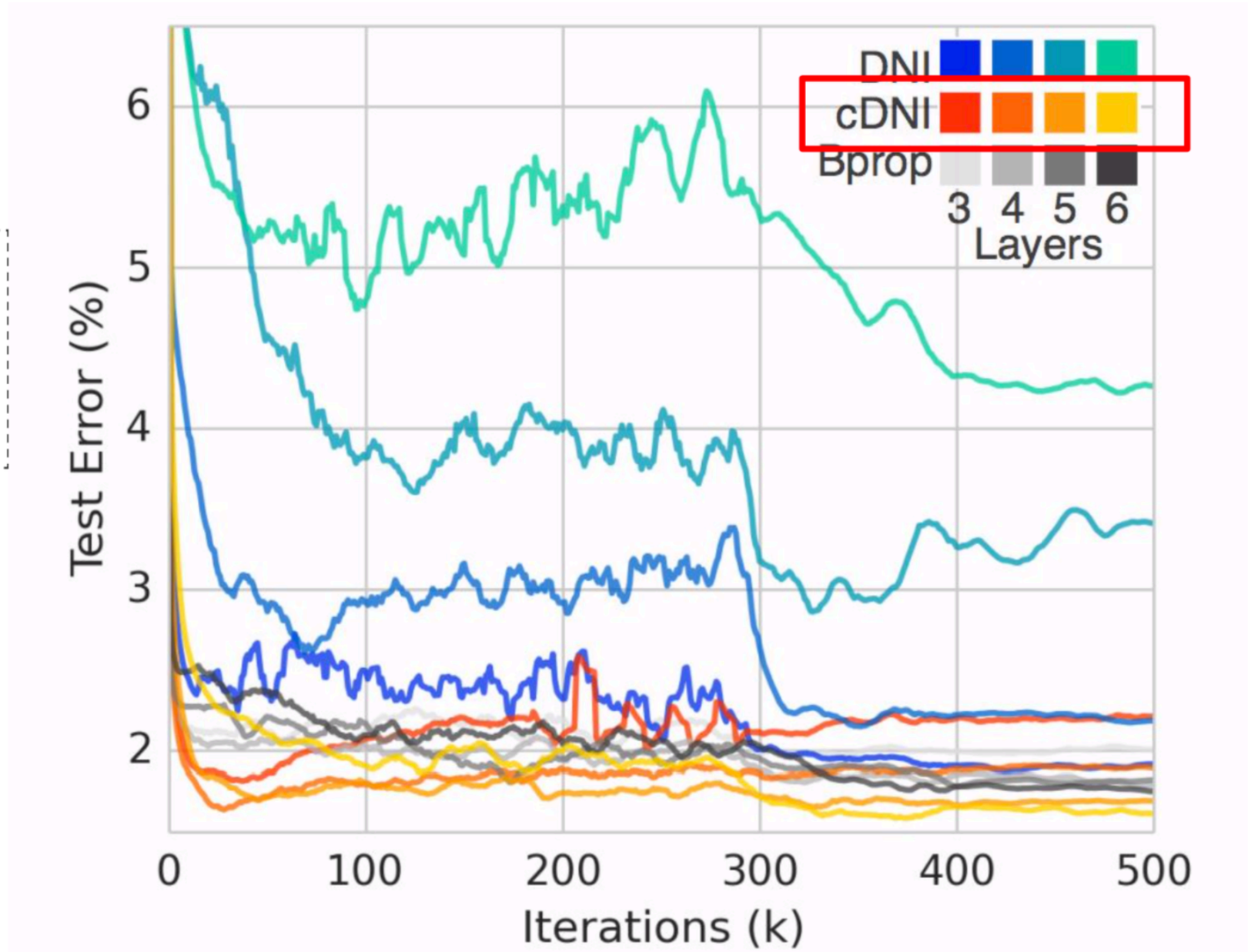
Result on MNIST



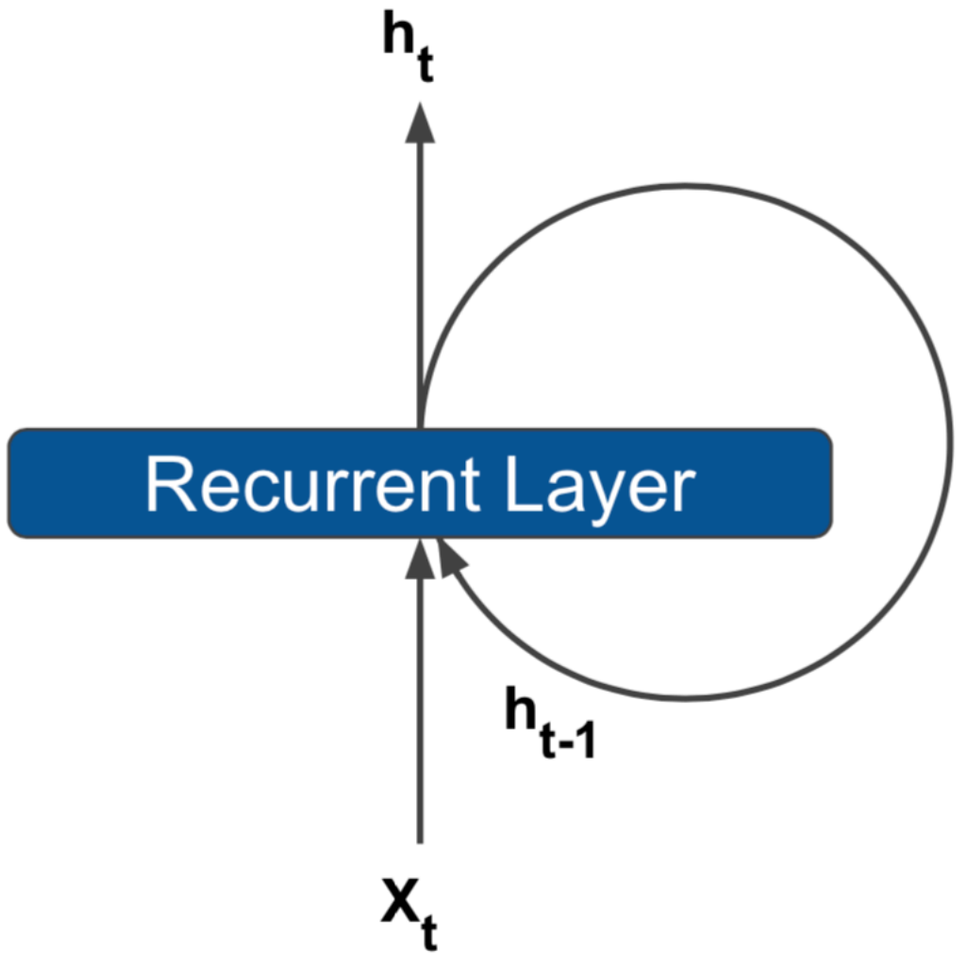


Result on MNIST

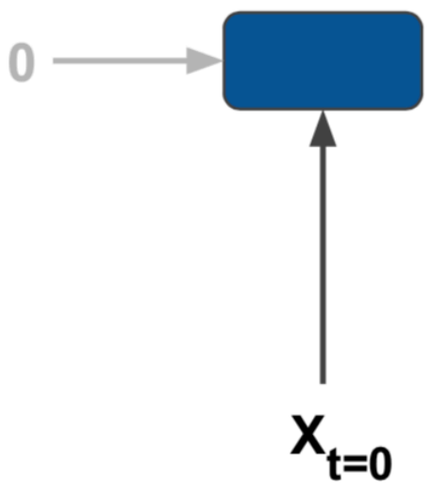




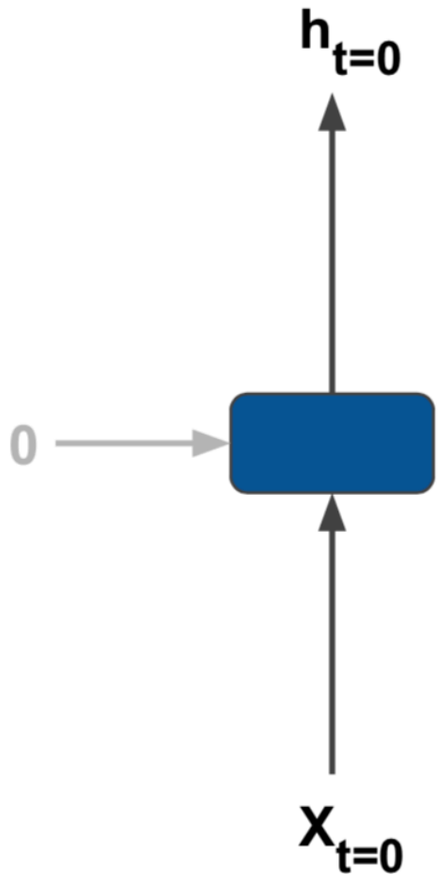
RNNs



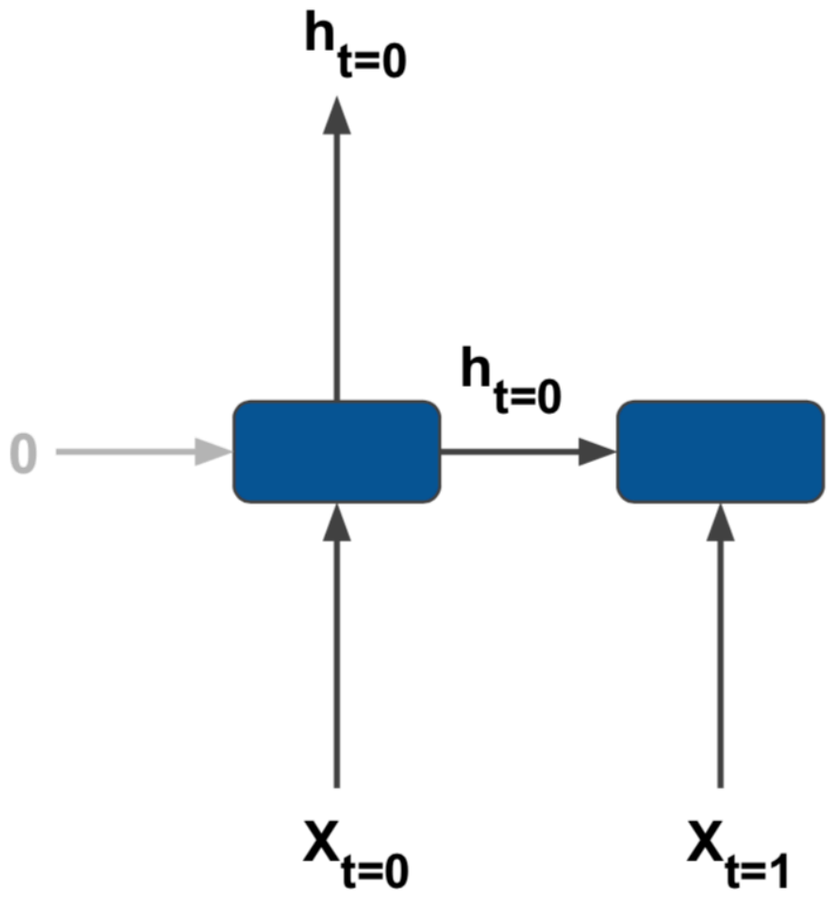
RNNs



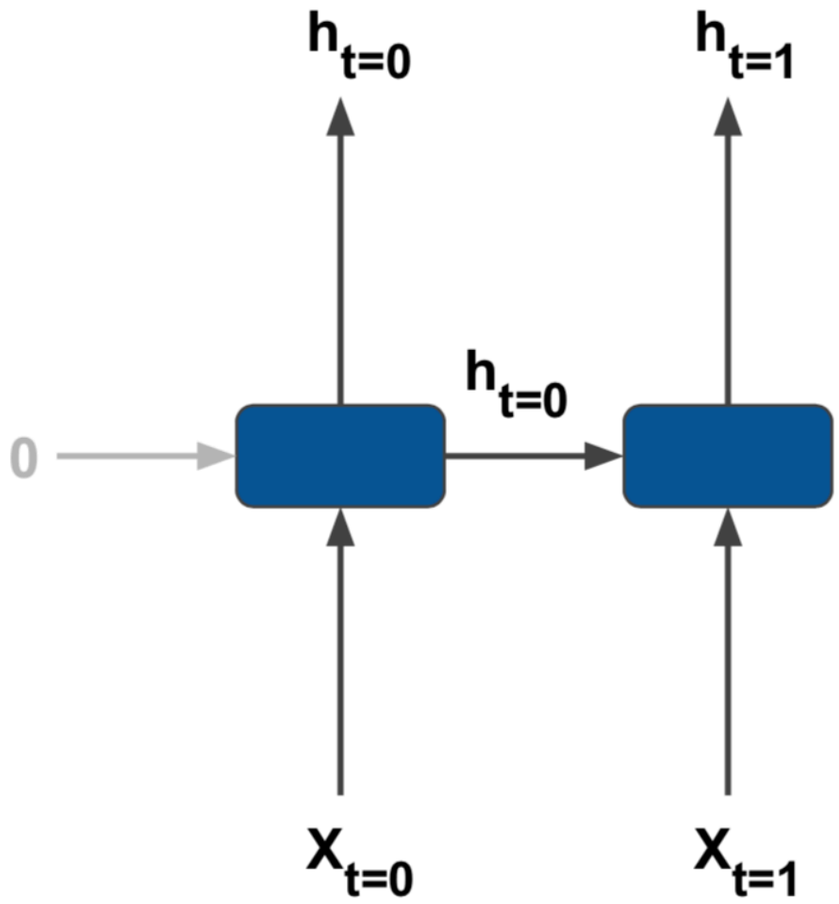
RNNs



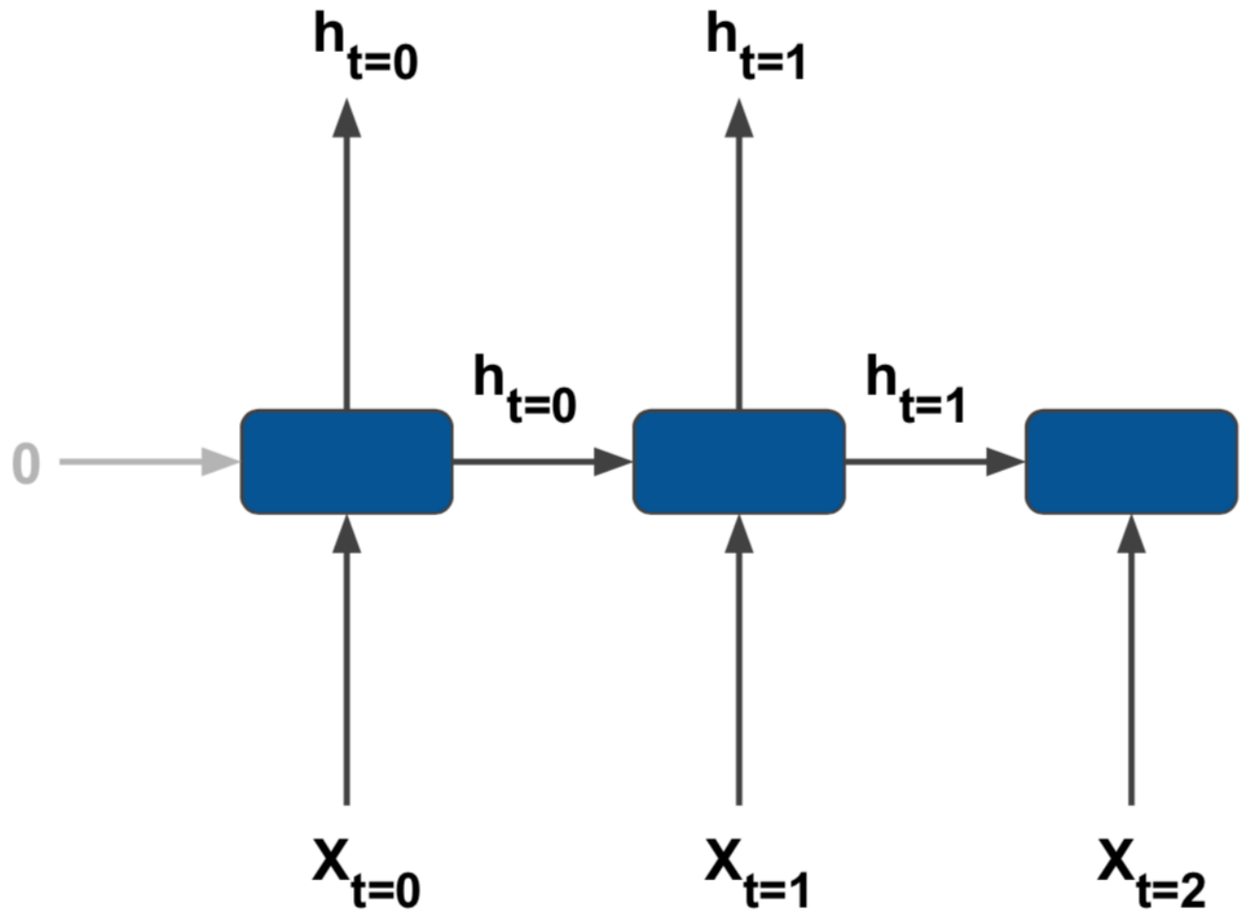
RNNs



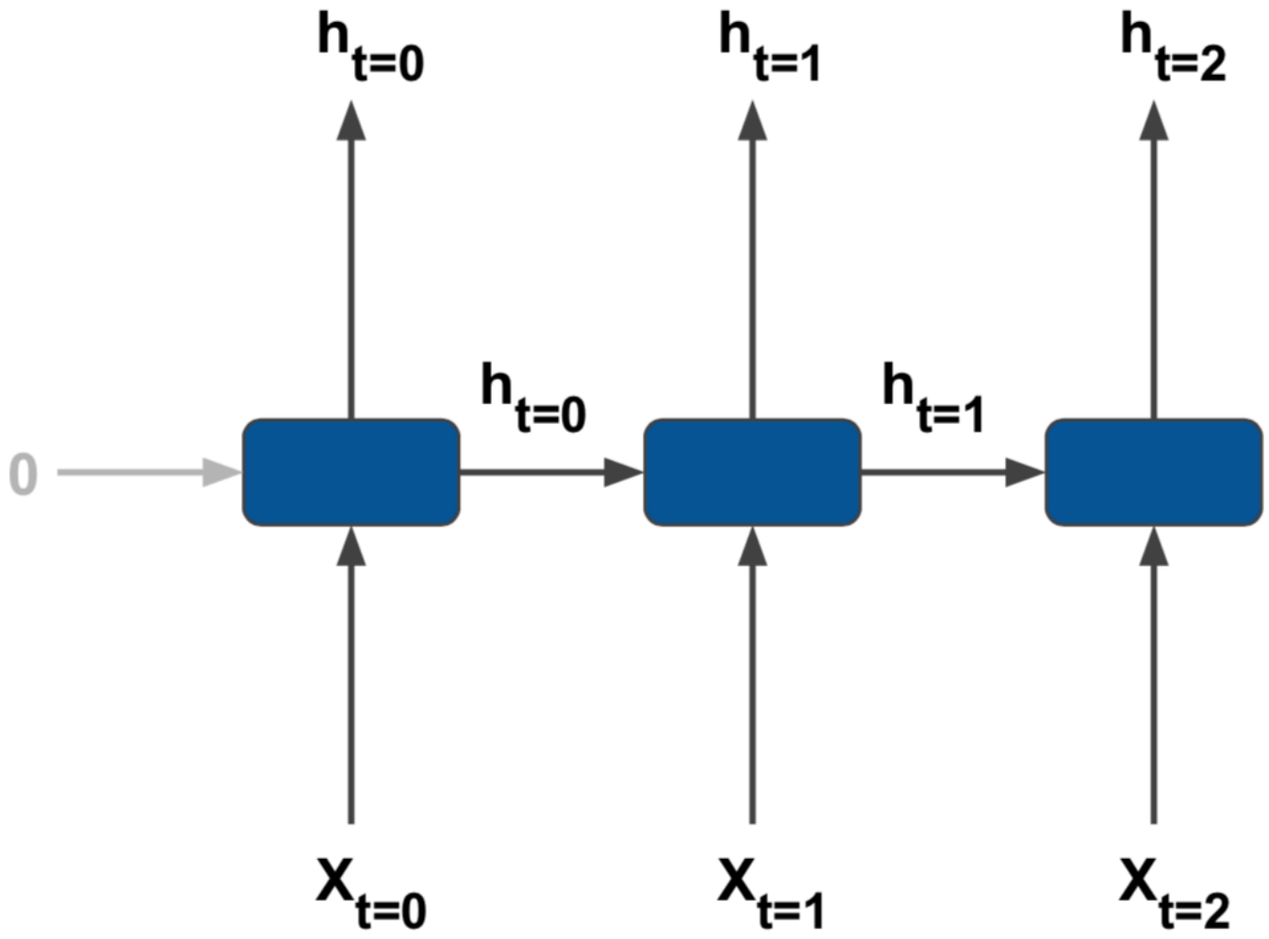
RNNs



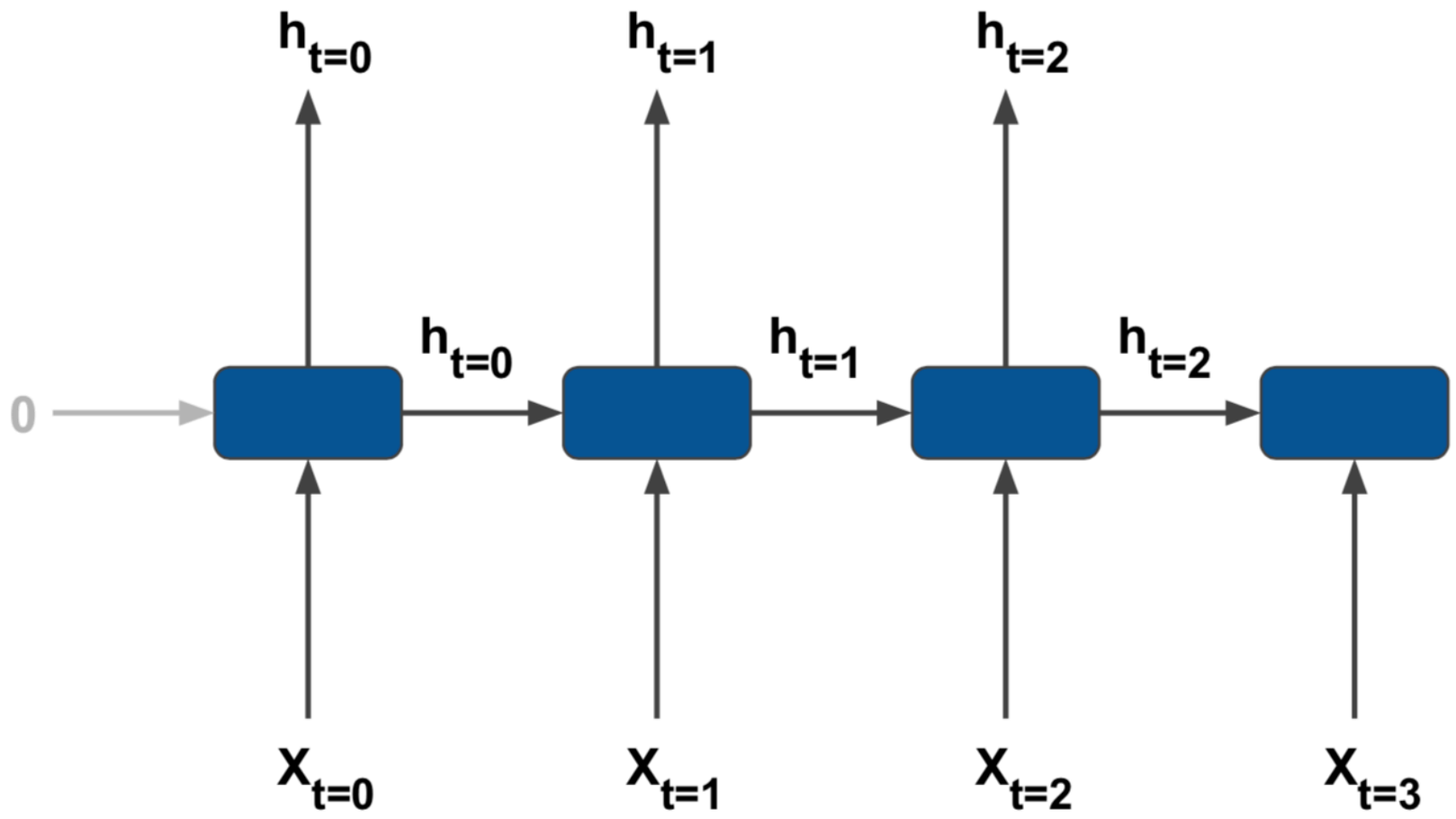
RNNs



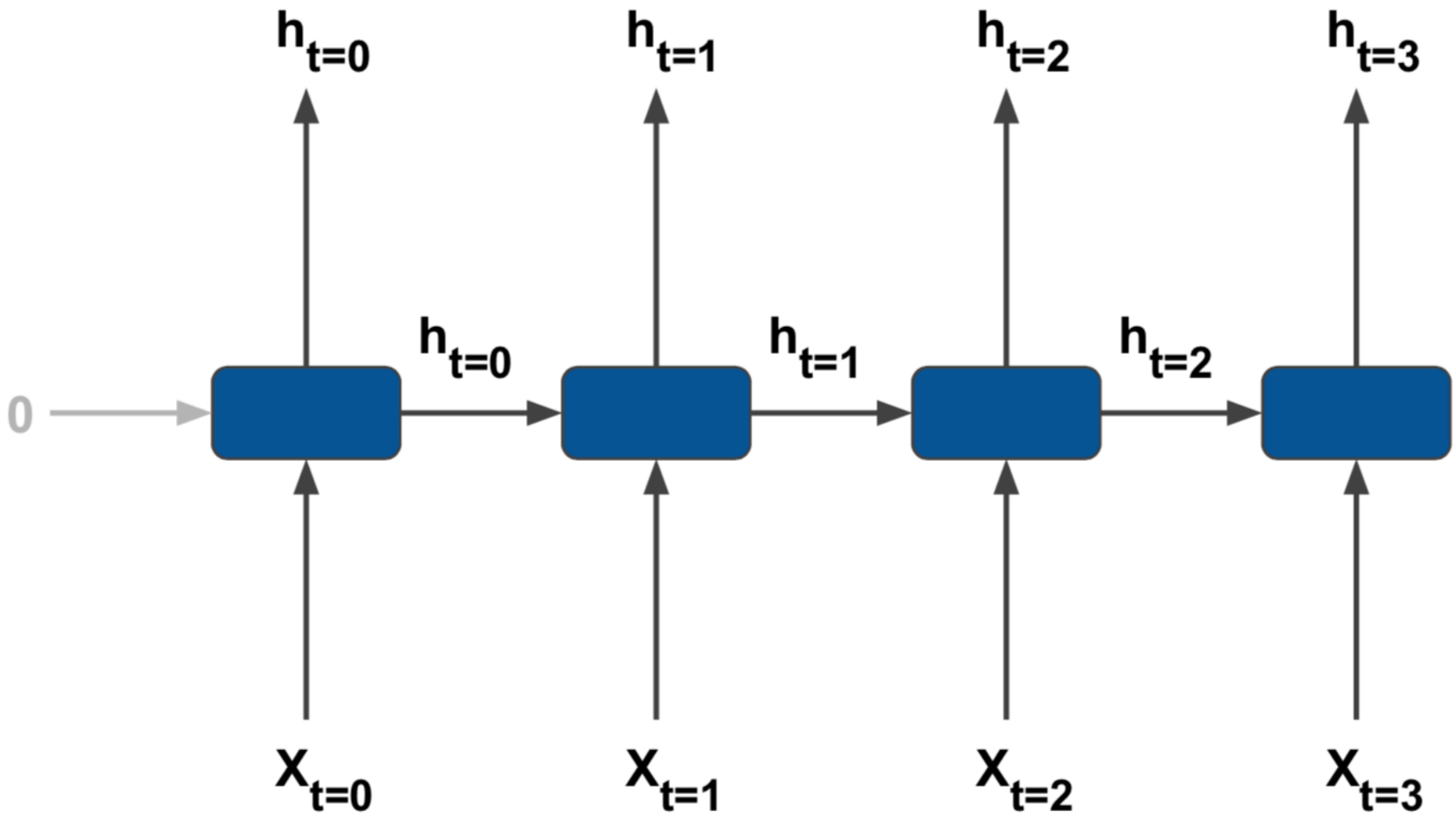
RNNs



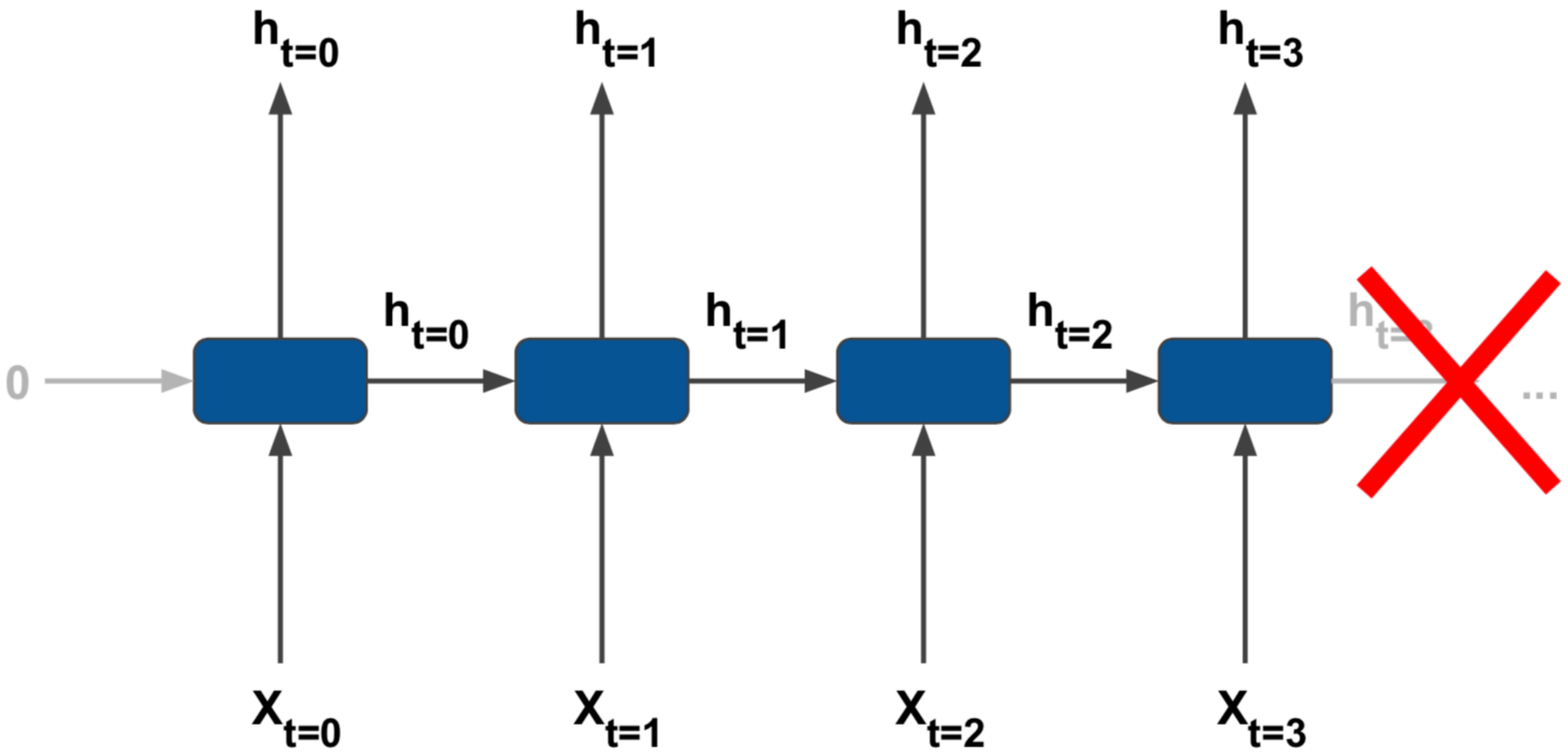
RNNs



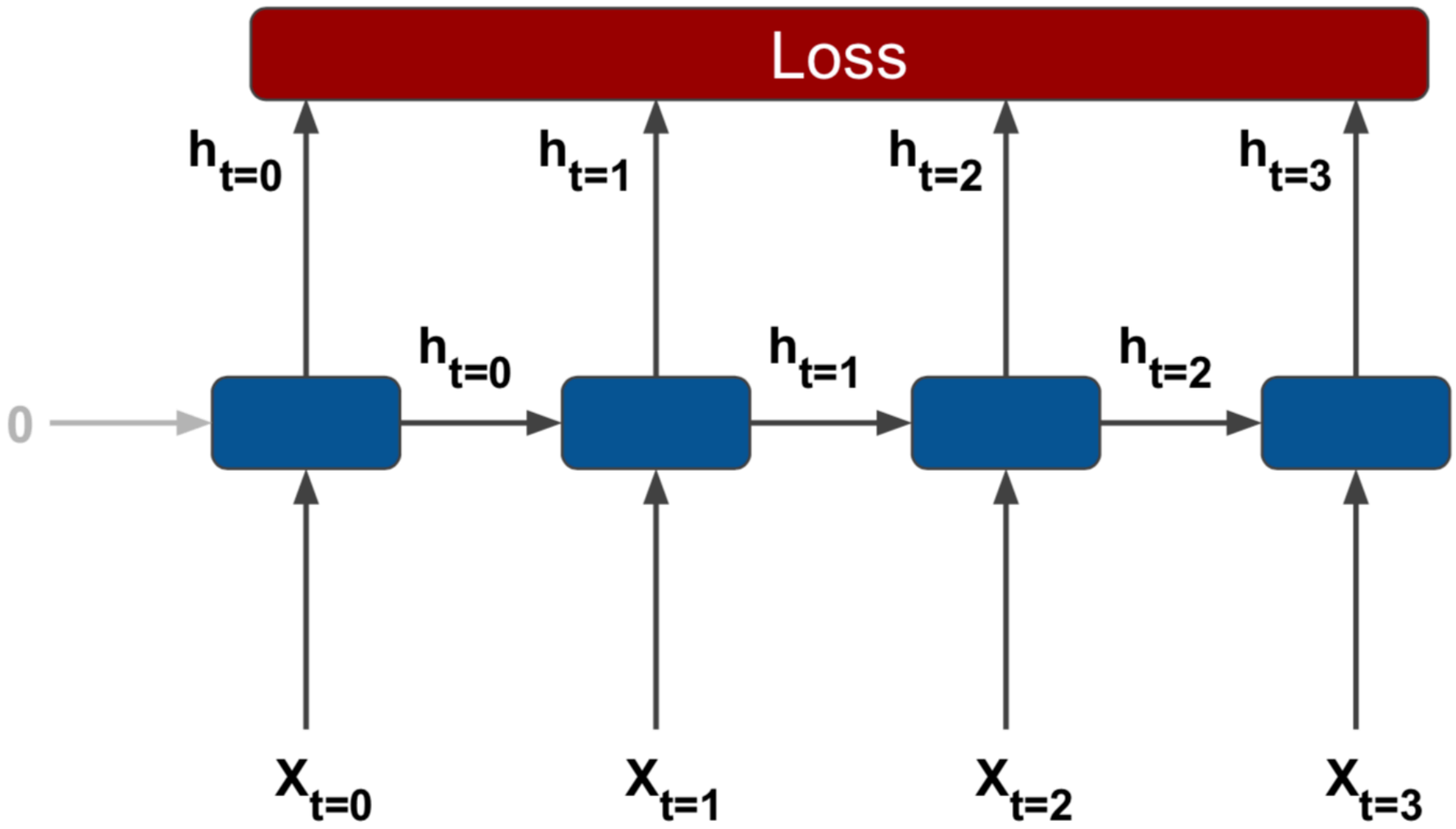
RNNs



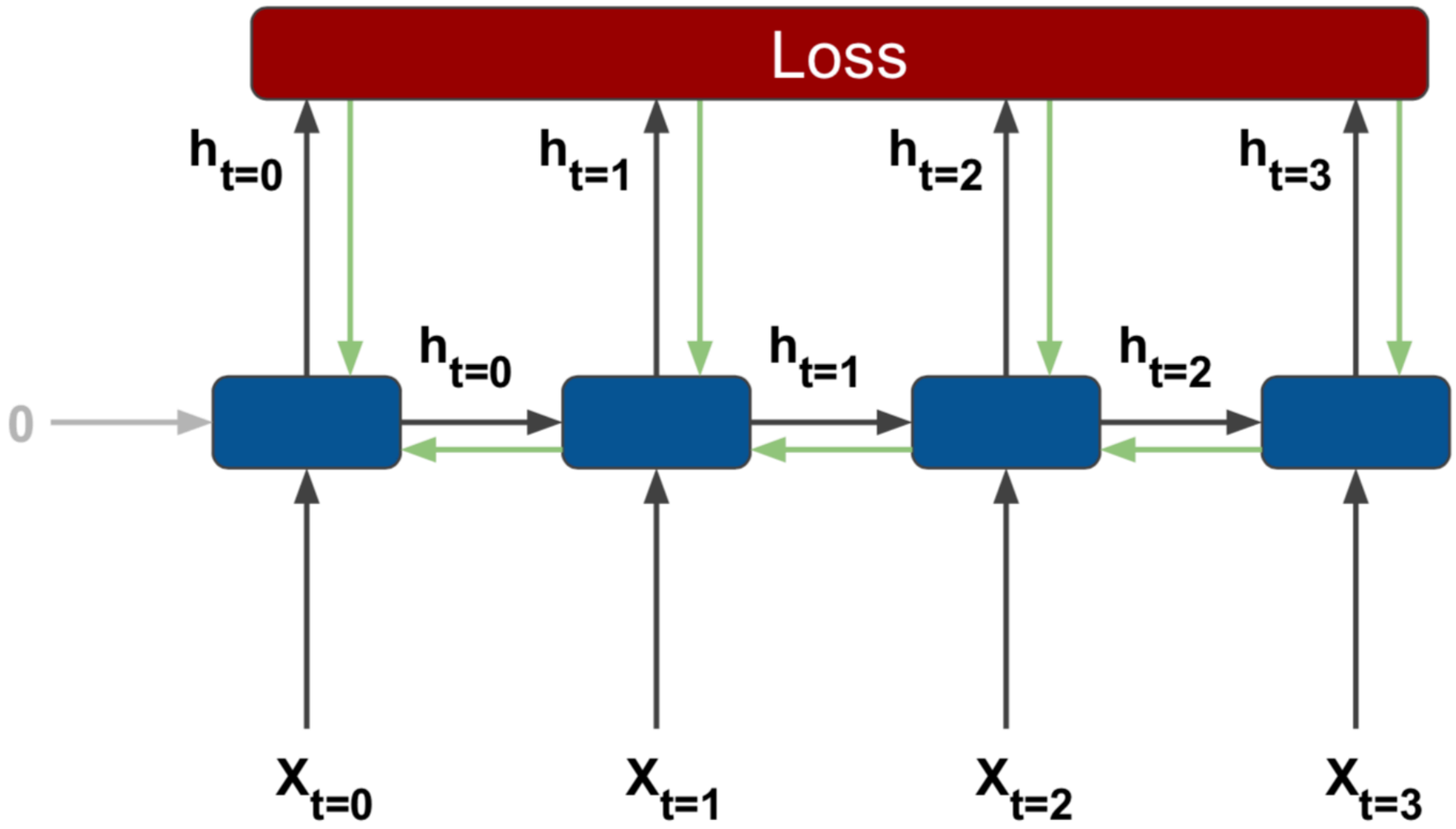
RNNs



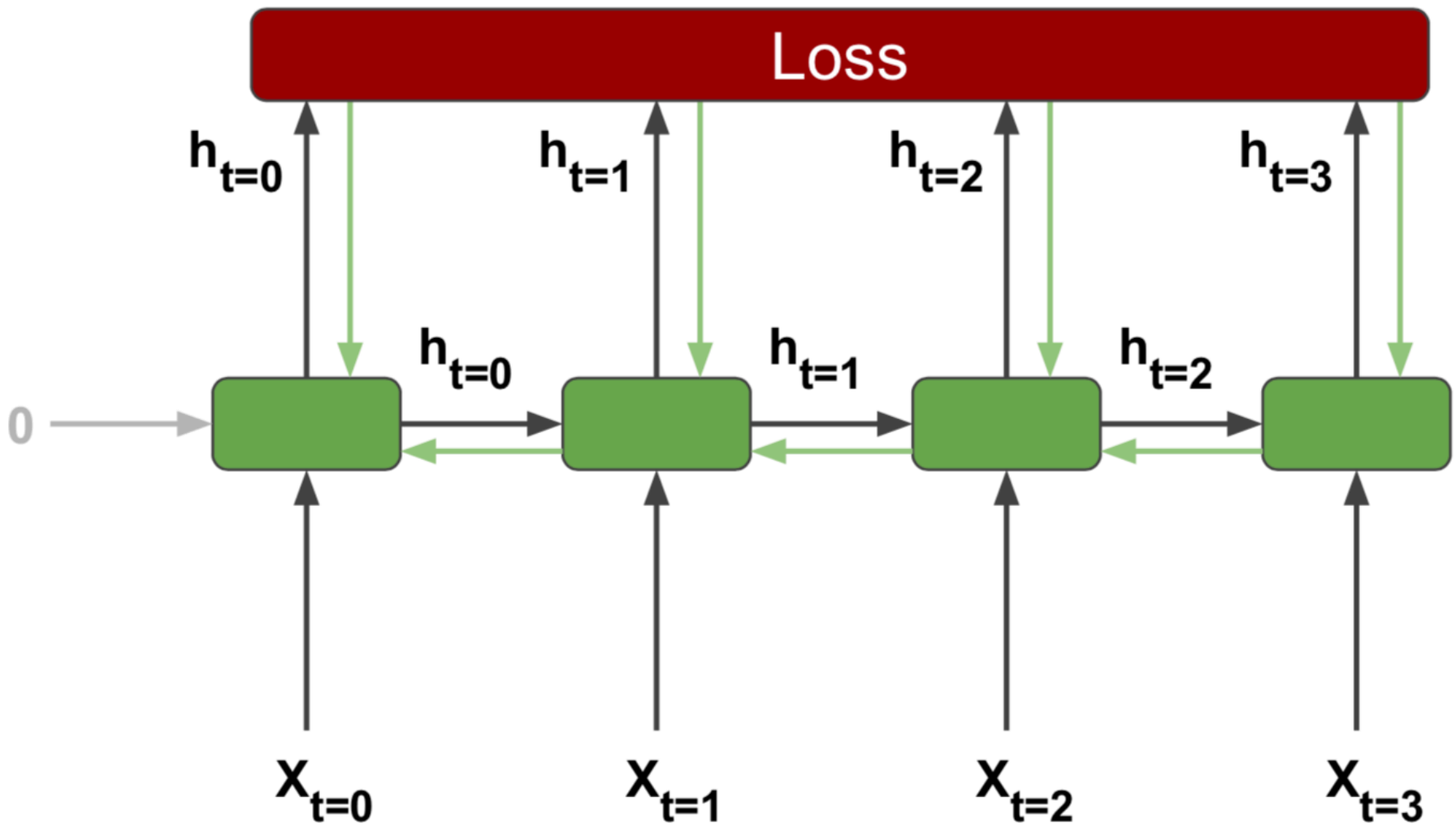
RNNs



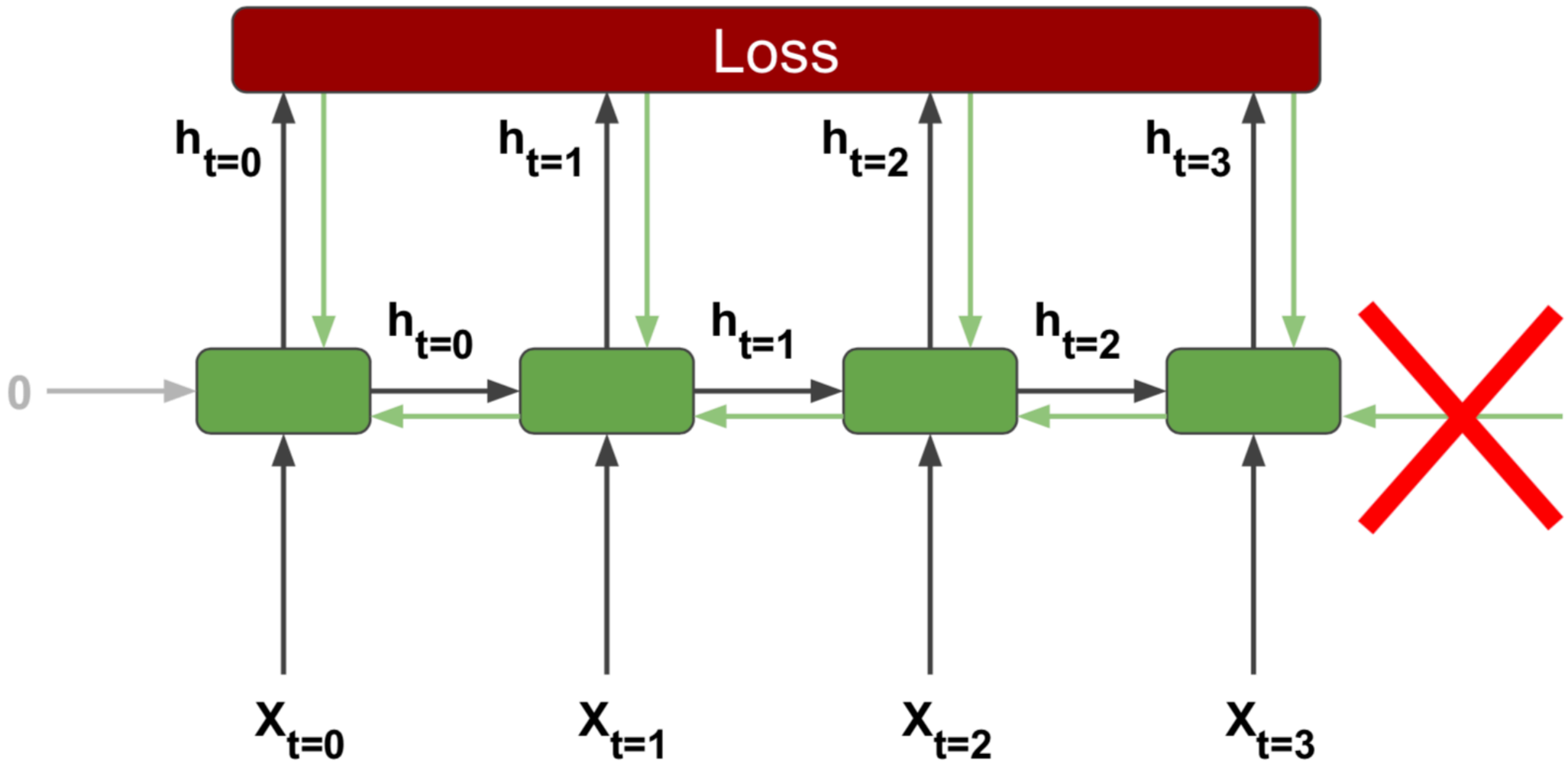
RNNs



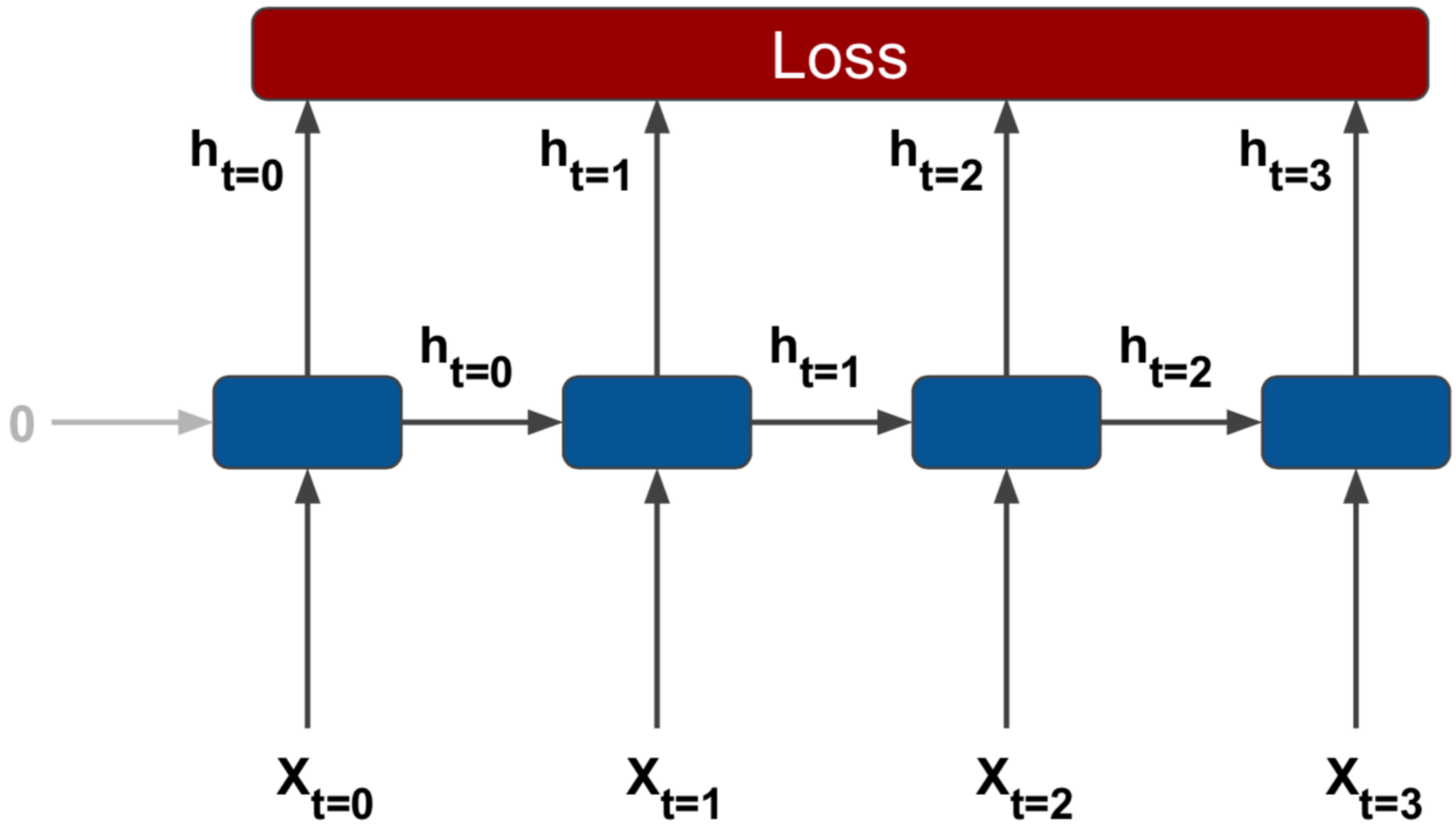
RNNs



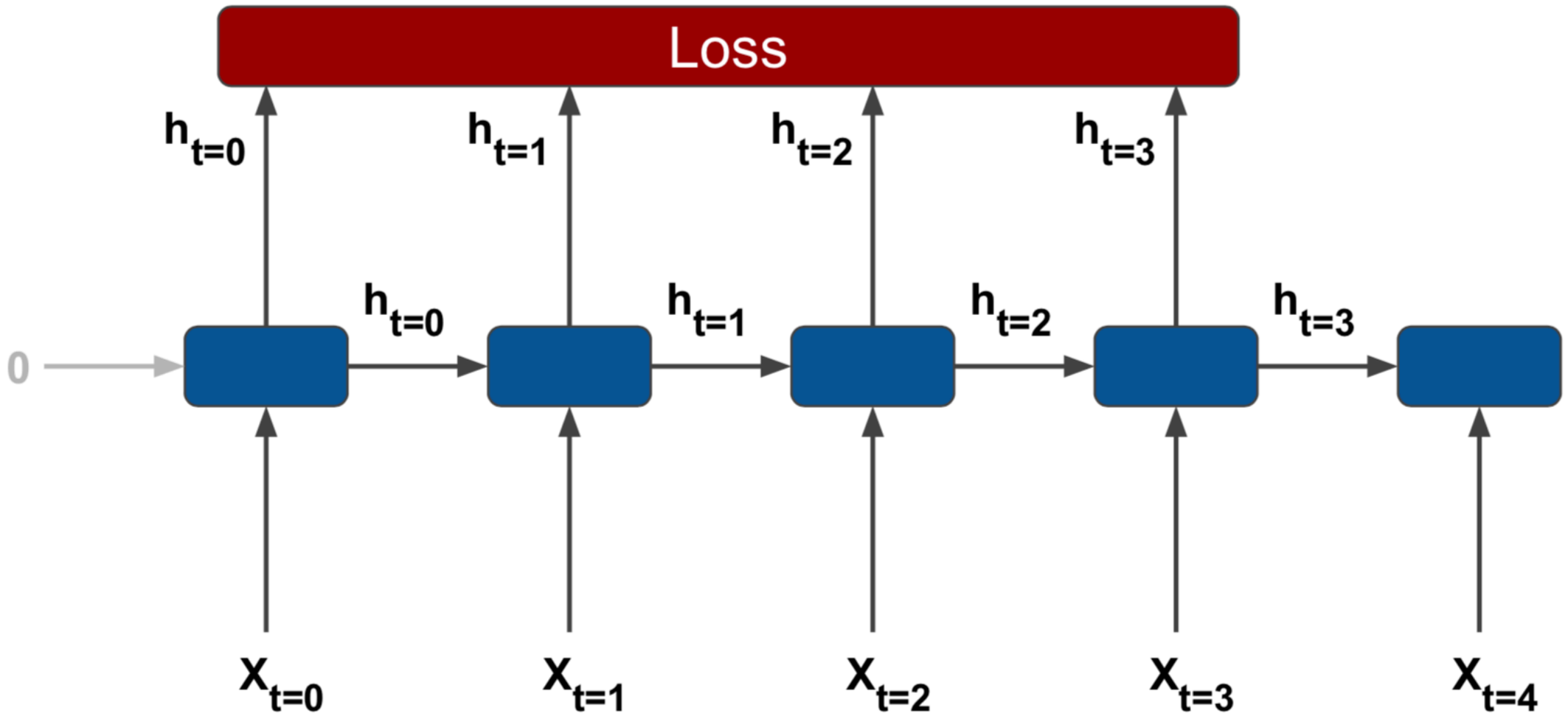
RNNs



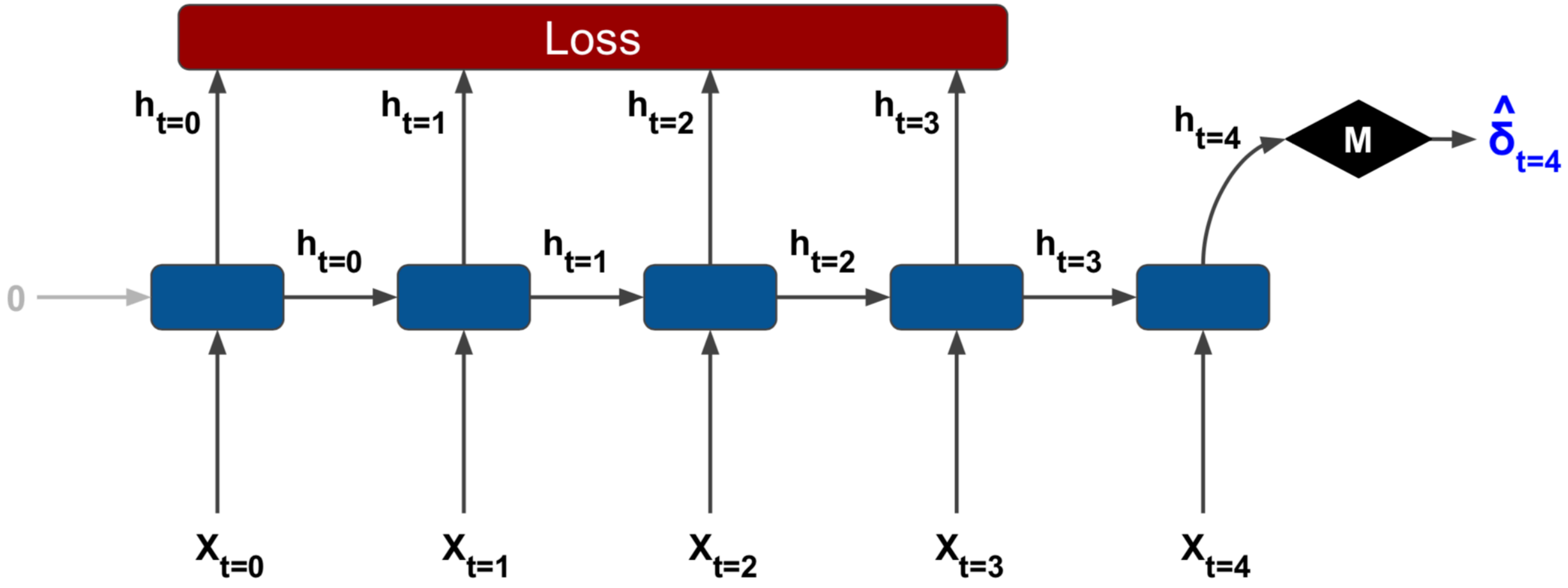
RNNs with SGs



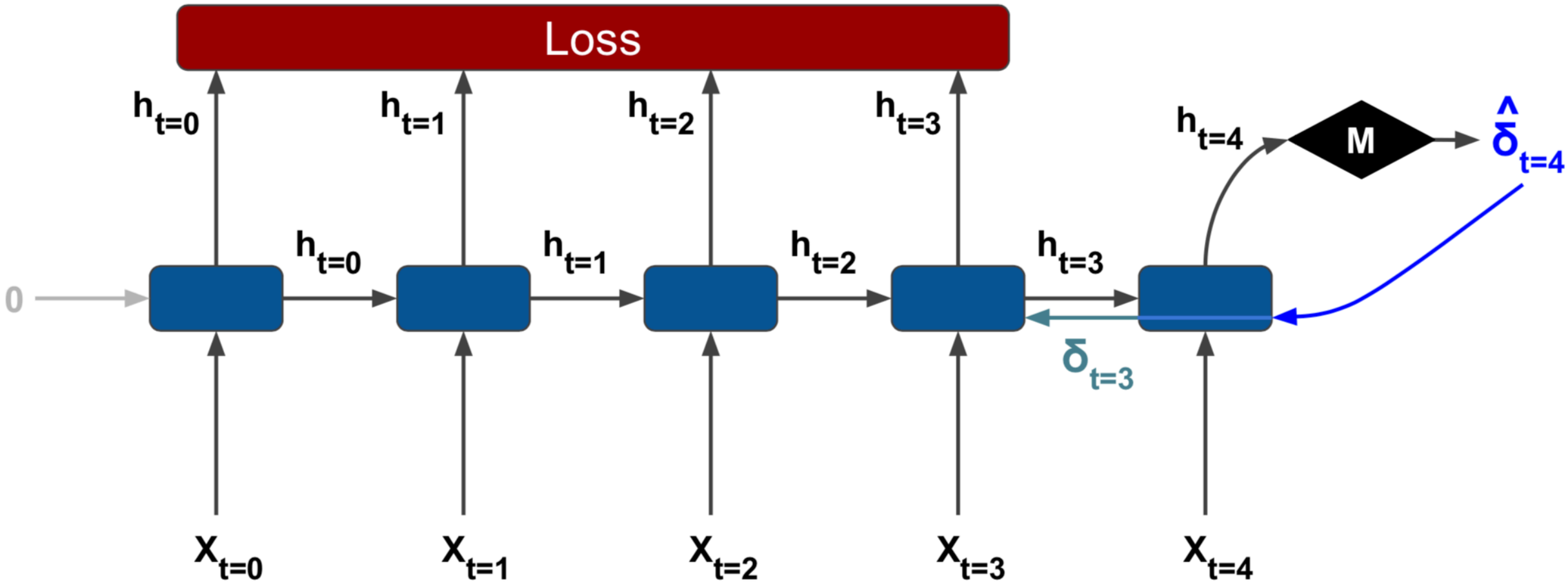
RNNs with SGs



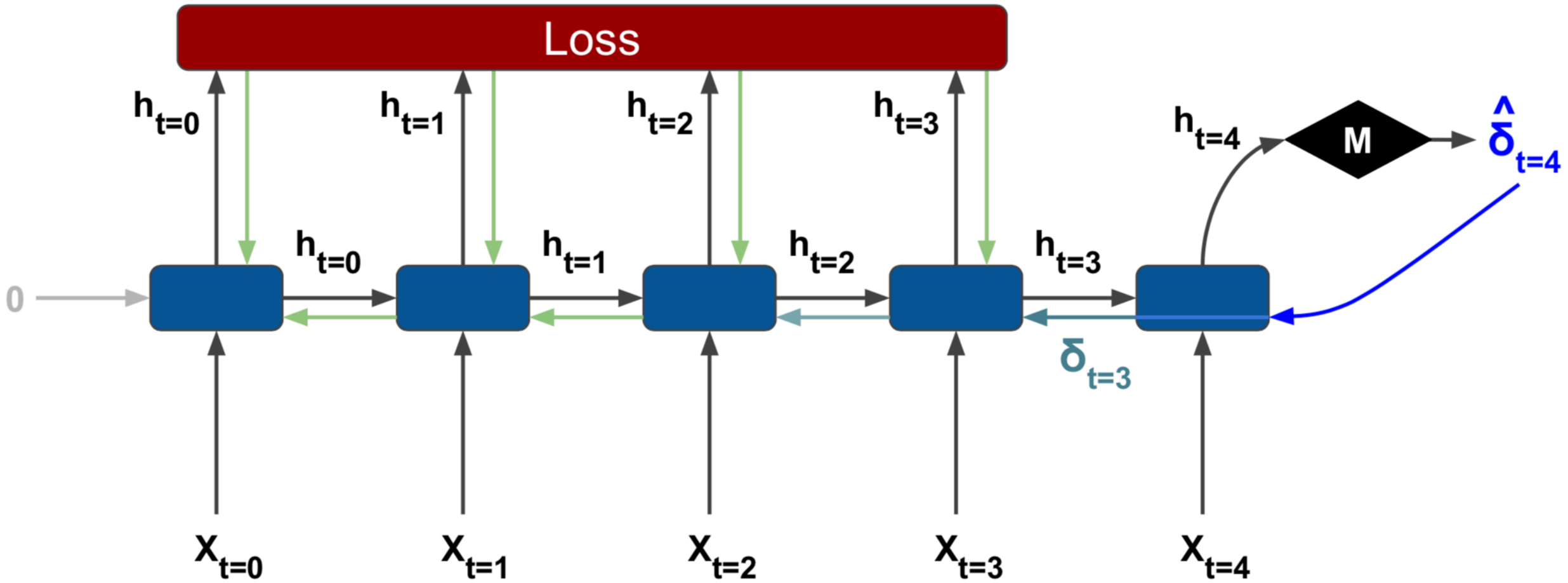
RNNs with SGs



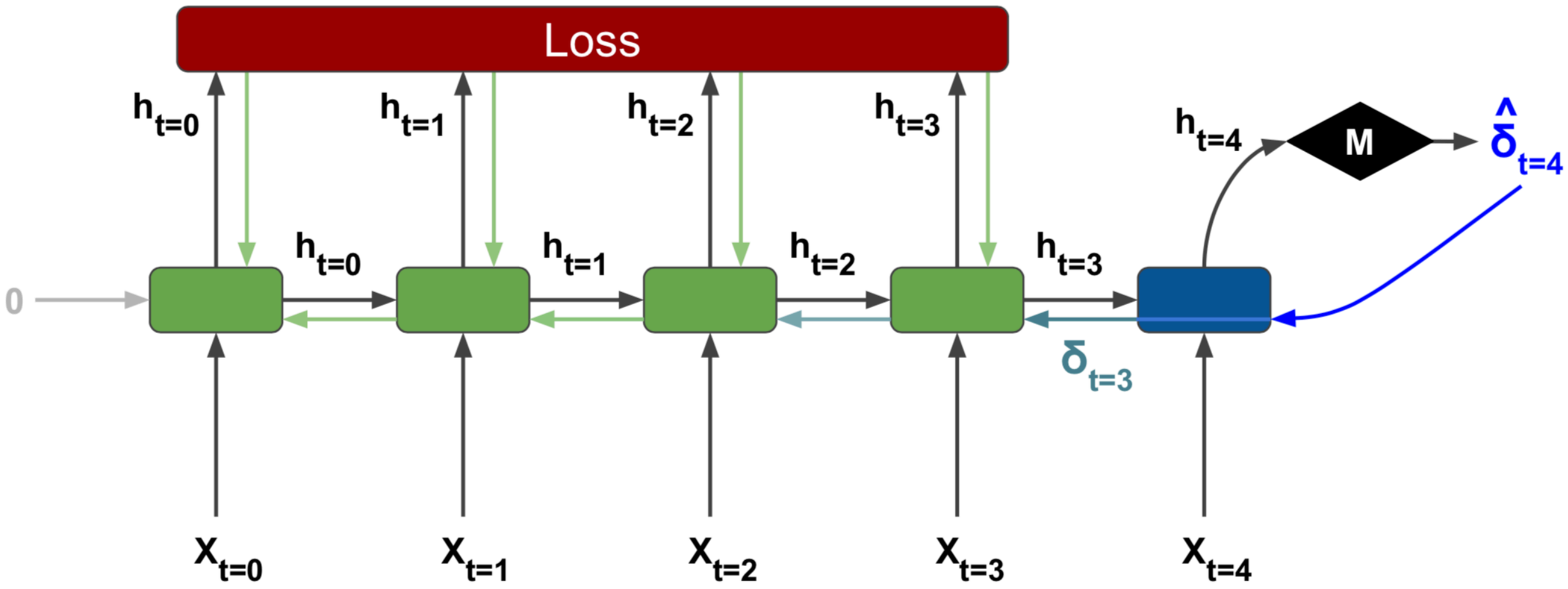
RNNs with SGs



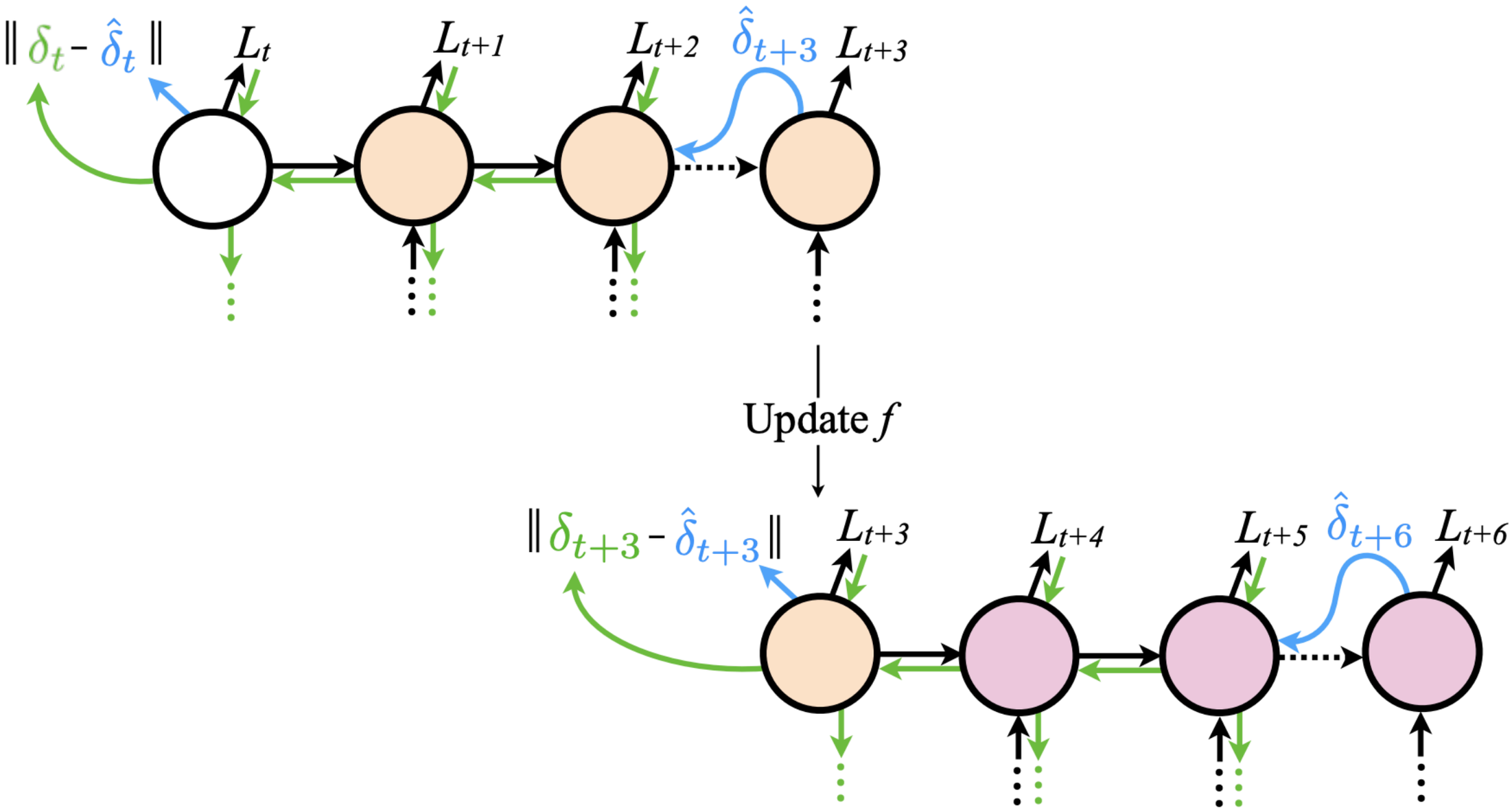
RNNs with SGs



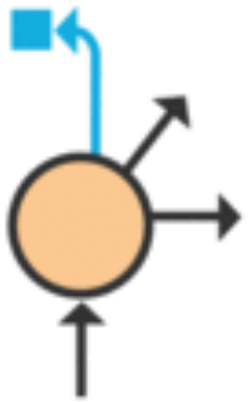
RNNs with SGs



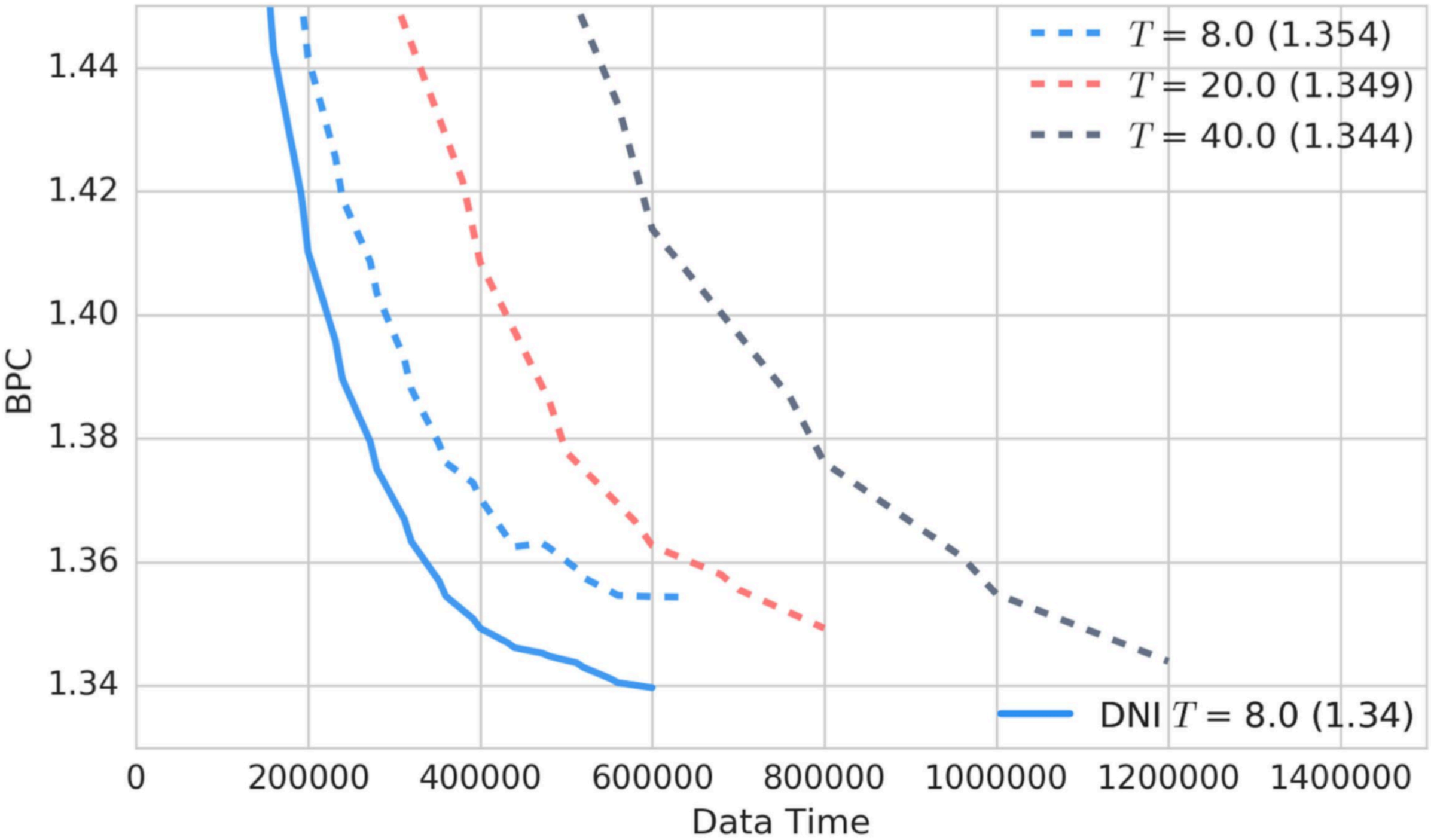
RNNs with SGs



RNNs with SGs



Penn Treebank Results



References

Papers:

- Decoupled Neural Interfaces using Synthetic Gradients. Max Jaderberg et al., 2016:
- Understanding Synthetic Gradients and Decoupled Neural Interfaces, Wojciech Marian Czarnecki et al., 2017:

Blog posts:

- By Max Jaderberg, DeepMind
- By iamtrask

Youtube:

- By Aurélien Géron
- By Siraj Raval

Code examples:

- By Siraj Raval