Review Series of Recent Deep Learning Papers:

Parameter Prediction Paper: Decoupled Neural Interfaces Using Synthetic Gradients

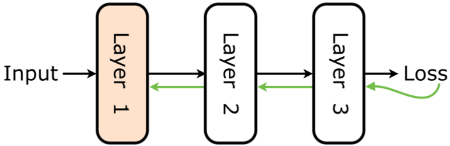
Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver Koray Kavukcuoglu

Reviewed by : Arshdeep Sekhon

[1]Department of Computer Science, University of Virginia
https://qdata.github.io/deep2Read/

August 25, 2018

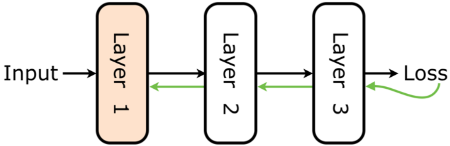
Locking in Neural Networks



To update Layer 1:

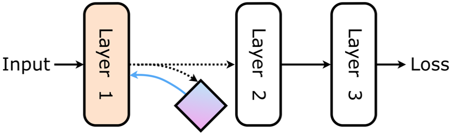
1. Forward Propagation through Layer 2 and Layer 3
2. Backward Propagation through Layer 2 and Layer 3

Why is locking a problem?



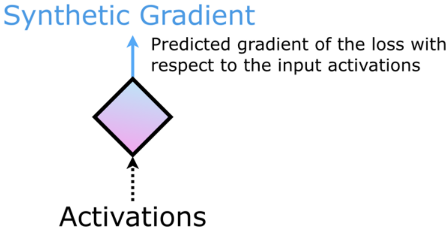
1. Updates in sequential and synchronous manner
2. A distributed system: Updates depend on the slowest part
3. parallelizing training of neural network modules can speed up training.

Decoupled Neural interface



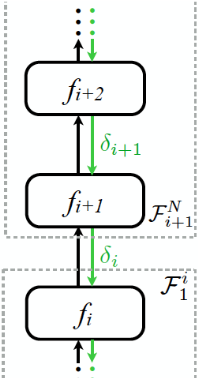
1. Layer 1 will be updated before Layer 2 and Layer 3 have even been executed.
2. No longer locked to the rest of the network.

A Decoupled Neural Interface



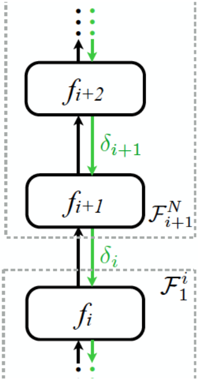
1. Decoupled Neural Interfaces predict gradients : synthetic gradients from previous layer outputs or activations
2. do not rely on backpropagation to get error gradients

Gradients for FeedForward Networks



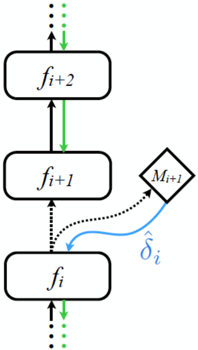
1. A network with N layers f_i,i $\epsilon\{1, \cdots, N\}$
2. For the i_{th} layer, input h_{i-1}, output $h_i = f_i(h_{i-1})$
3. The complete graph is represented by $\mathbb{F}_1^N$

Gradients for FeedForward Networks



1. $$\theta_i \leftarrow \theta_i - \alpha \delta_i \frac{\delta h_i}{\delta \theta_i}; \; \delta_i = \frac{\delta L}{\delta h_i} \tag{1}$$

Synthetic Gradients for FeedForward Networks

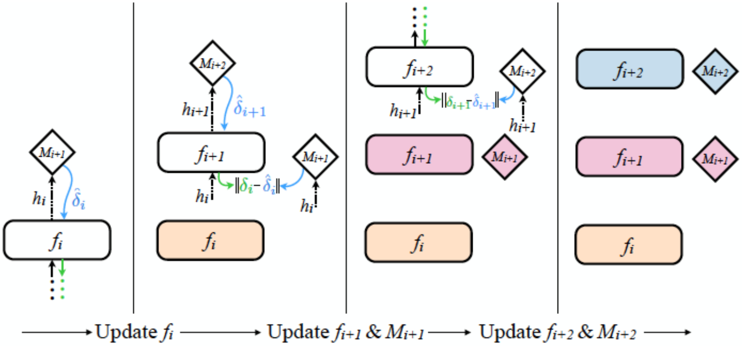


$$\theta_n \leftarrow \theta_n - \alpha\hat{\delta}_i \frac{\delta h_i}{\delta \theta_n}; \;\; \delta_i = M_{i+1}(h_i) \tag{2}$$

Synthetic Gradients for FeedForward Networks

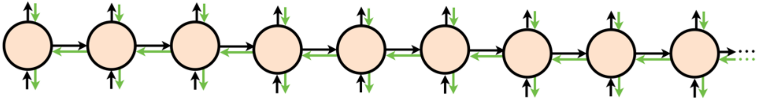
1. $$\theta_n \leftarrow \theta_n - \alpha\hat{\delta}_i \frac{\delta h_i}{\delta\theta_n};\ \delta_i = M_{i+1}(h_i) \tag{3}$$
2. n $\epsilon\{1, \cdots, n\}$
3. To train $M_{i+1}(h_i)$,
4. wait for true error gradient to be computed
5. after a full forwards and backwards pass of $\mathbb{F}_{i+1}^{N}$
6. Minimize $||\hat{\delta}_i - \delta_i||_2^2$

use backpropagated $\hat{\delta_{i+1}}$ instead of the true gradients



Synthetic Gradients for RNNs

The task: stream prediction; possibly infinite



Unrolling the recurrent network:

Forward Graph: $\mathbb{F}_1^{\text{inf}}$ made up of f_i where i varies from 1 to inf

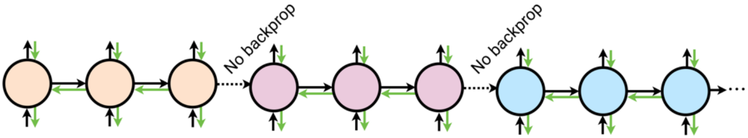
At a particular point in time t, minimise Loss over the next steps

$$\sum_{\tau=t}^{\text{inf}} L_\tau \tag{4}$$

$$\theta \leftarrow \theta - \alpha \sum_{\tau=t}^{\text{inf}} \frac{\delta L_\tau}{\delta \theta} \tag{5}$$

Synthetic Gradients for RNNs

Truncated Backpropagation



At a particular point in time t, minimise Loss over the next steps

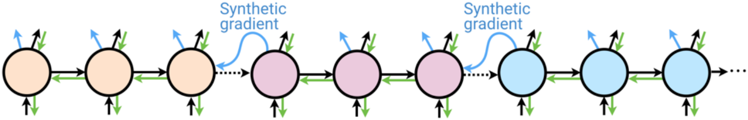
$$\theta \leftarrow \theta - \alpha\Big(\sum_{\tau=t}^{T} \frac{\delta L_\tau}{\delta\theta} + \Big(\sum_{\tau=T+1}^{\inf} \frac{\delta L_\tau}{\delta h_T}\Big)\frac{\delta h_T}{\delta\theta}\Big) \tag{6}$$

$$\theta \leftarrow \theta - \alpha\Big(\sum_{\tau=t}^{T} \frac{\delta L_\tau}{\delta\theta} + \Big(\delta_T\Big)\frac{\delta h_T}{\delta\theta}\Big) \tag{7}$$

truncated BPTT: $\delta_T = 0$; limits temporal dependency learnt by rnn

Synthetic Gradients for RNNs

Truncated Backpropagation



$$\theta \leftarrow \theta - \alpha\Big(\sum_{\tau=t}^{T} \frac{\delta L_\tau}{\delta\theta} + \Big(\hat{\delta_T}\Big)\frac{\delta h_T}{\delta\theta}\Big) \tag{8}$$

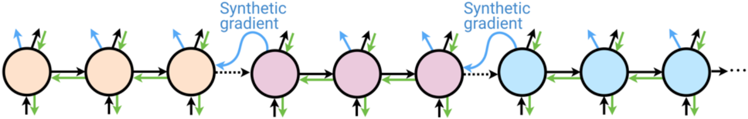
$\hat{\delta_T} = M_T(h_T)$; learned approximation of the future loss gradients

divide unrolled rnn into subnetworks of length T

insert a DNI between $\mathbb{F}_t^{t+T-1}$ and $\mathbb{F}_{t+T}^{t+2T-1}$

Synthetic Gradients for RNNs

Truncated Backpropagation



$$\theta \leftarrow \theta - \alpha\Big(\sum_{\tau=t}^{T}\frac{\delta L_\tau}{\delta\theta} + \Big(\hat{\delta_T}\Big)\frac{\delta h_T}{\delta\theta}\Big) \tag{9}$$

train M_T by minimizing $d(\delta_T, \hat{\delta_T})$

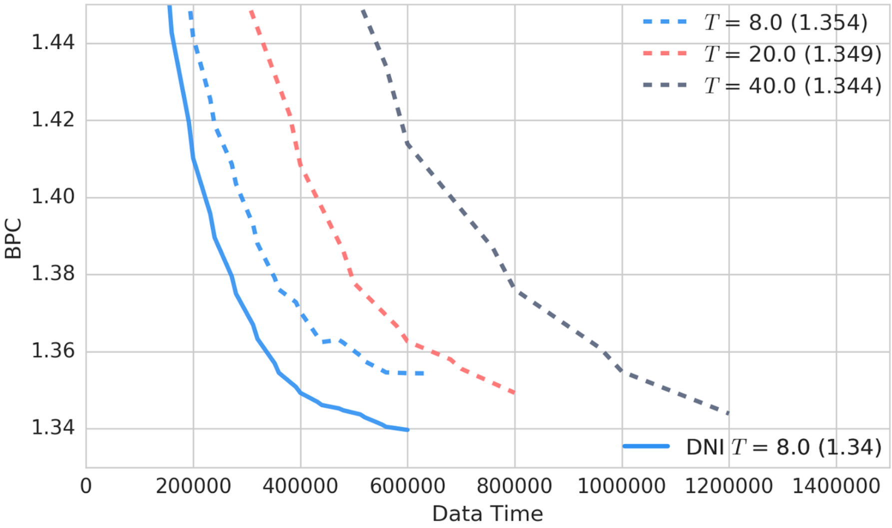
true δ_T not available: Bootstrapping

$\delta_T = \sum_{\tau=T+1}^{2T}\frac{\delta L_\tau}{\delta h_T} + \hat{\delta}_{2T+1}\frac{h_{2T}}{h_T}$

Other applications

1. Add an auxiliary task
2. Combine with true backpropagation gradients
3. Arbitrary Network Graphs

Results: Penn Tree Bank Language Modeling



Results: Copy Repeat Copy Task

1. **Copy**: Copy a sentence of length N
2. **Repeat Copy**: copy a sentence of length N R times

	BPTT							DNI				
$T =$	2	3	4	5	8	20	40	2	3	4	5	8
Copy	7	8	10	8	-	-	-	16	14	18	18	-
Repeat Copy	7	5	19	23	-	-	-	39	33	39	59	-

3. Max sequence length successfully modeled increases with DNI for the same T in BPTT