Associated learning

Hung-Hsuan Chen

1

Review: backpropagation

- A method to efficiently compute the gradient of the loss with respect to the parameters
  - Chain rule
  - Layer-wise update

2

Backward locking

- Since the parameters in a DNN are updated in a layer-wise fashion:
  - Updating the weights of one layer *locks* the weight updates in the other layers
- Challenging to apply parallel computing or a pipeline structure to update the weights in different layers simultaneously

3

Time complexity of training a DNN

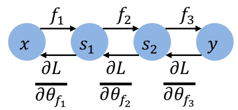
- Training accuracy:
  - n: number of training instances
  - ℓ: number of hidden layers.
  - Time complexity: $O(n\ell)$
- As a network becomes deeper, more training time are needed

4

Can we "parallelize" the training?

- Two types of parallelizations
- Type 1
  - Compute SGD at different cores using different subsamples
  - Average the gradients of different cores, and apply parameter updates
- Type 2
  - Update parameters of different layers simultaneously
  - Seems more challenging
  - We focus on this type

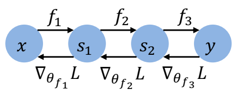
Naively using SGD and backpropagation



	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$	$t=11$	$t=12$
$(x^{(1)}, y^{(1)})$	$f_1^{(0)}$	$f_2^{(0)}$	$f_3^{(0)}$	$\frac{\partial L}{\partial \theta_{f_3}}$	$\frac{\partial L}{\partial \theta_{f_2}}$	$\frac{\partial L}{\partial \theta_{f_1}}$						
$(x^{(2)}, y^{(2)})$							$f_1^{(1)}$	$f_2^{(1)}$	$f_3^{(1)}$	$\frac{\partial L}{\partial \theta_{f_3}}$	$\frac{\partial L}{\partial \theta_{f_2}}$	$\frac{\partial L}{\partial \theta_{f_1}}$

- Notation
  - f_i: function at layer i
  - $f_i^{(j)}$: parameter values for layer i after j updates
  - θ_{f_i}: parameters for layer i
- Two hidden layers
  - Updating all parameters once requires 6 time units
- If ℓ hidden layers
  - Updating all parameters once requires $2(\ell + 1)$ time units
- Training time: $O(\ell)$
  - Training time is long for "deep" network

Add pipeline directly



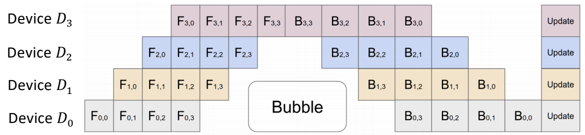
	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$	$t=11$	$t=12$
$(x^{(1)}, y^{(1)})$	$f_1^{(0)}$	$f_2^{(0)}$	$f_3^{(0)}$	$\frac{\partial L}{\partial \theta_{f_3}}$	$\frac{\partial L}{\partial \theta_{f_2}}$	$\frac{\partial L}{\partial \theta_{f_1}}$						
$(x^{(2)}, y^{(2)})$		$f_1^{(0)}$	$f_2^{(0)}$	$f_3^{(0)}$	$\frac{\partial L}{\partial \theta_{f_3}}$	$\frac{\partial L}{\partial \theta_{f_2}}$	$\frac{\partial L}{\partial \theta_{f_1}}$					
$(x^{(3)}, y^{(3)})$			$f_1^{(0)}$	$f_2^{(0)}$	$f_3^{(1)}$	$\frac{\partial L}{\partial \theta_{f_3}}$	$\frac{\partial L}{\partial \theta_{f_2}}$	$\frac{\partial L}{\partial \theta_{f_1}}$				
$(x^{(4)}, y^{(4)})$				$f_1^{(0)}$	$f_2^{(0)}$	$f_3^{(2)}$	$\frac{\partial L}{\partial \theta_{f_3}}$	$\frac{\partial L}{\partial \theta_{f_2}}$	$\frac{\partial L}{\partial \theta_{f_1}}$			
$(x^{(5)}, y^{(5)})$					$f_1^{(0)}$	$f_2^{(1)}$	$f_3^{(3)}$	$\frac{\partial L}{\partial \theta_{f_3}}$	$\frac{\partial L}{\partial \theta_{f_2}}$	$\frac{\partial L}{\partial \theta_{f_1}}$		

- If ℓ hidden layers and fully pipelined
  - Starting from $2(\ell + 1)$ time unit, all paramters are updated once for every time unit
- However, the updating is wrong
  - E.g., at $t = 6$, $\nabla_{\theta_{f_1}} L = \frac{\partial L}{\partial \theta_{f_3}}\Big|_{f_3 = f_3^{(2)}} \cdot \frac{\partial f_3}{\partial \theta_{f_2}}\Big|_{f_2 = f_2^{(1)}} \cdot \frac{\partial f_2}{\partial \theta_{f_1}}\Big|_{f_1 = f_1^{(0)}}$

How to pipeline?

- Gpipe
  - Yanping Huang et al.
  - NeurIPS 2019
- Associated learning
  - Yu-Wei Kao and Hung-Hsuan Chen
  - Neural Computation 33(1) 2021.

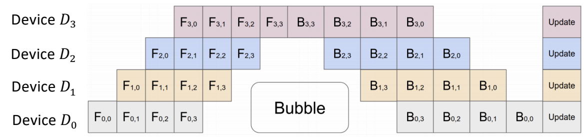
Gpipe



- Each mini-batch i contains 4 micro-batches $(i, 0) \dots (i, 3)$
- $F_{i,j}$: computing forward pass micro-batch (i, j)
- $B_{i,j}$: computing backward pass micro-batch (i, j)
- Time 0: D_0 processes $F_{0,0}$
- Time 1: D_0 processes $F_{0,1}$; D_1 processes $F_{0,0}$
- Time 2: D_0 processes $F_{0,2}$; D_1 processes $F_{0,1}$, D_2 processes $F_{0,0}$
- …
- Time 7: D_3 processes $B_{3,3}$
- Time 8: D_3 processes $B_{3,2}$, D_2 processes $B_{2,3}$
- Time 9: D_3 processes $B_{3,1}$, D_2 processes $B_{2,2}$, D_1 processes $B_{1,3}$
- Time 10: D_3 processes $B_{3,0}$, D_2 processes $B_{2,1}$, D_1 processes $B_{1,2}$,D_1 processes $B_{0,3}$
  - Now D_3 can update the weights, but it waits till Time 14 to update
- Time 11: D_2 processes $B_{2,0}$, D_1 processes $B_{1,1}$,D_1 processes $B_{0,2}$
  - Now D_2 can update the weights, but it waits till Time 14 to update
- …
- Time 14: $D_0 - D_3$ update the weights for each layer (simultaneously)

9

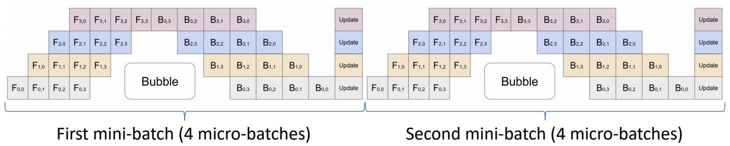
Gpipe



- If no Gpipe, computing forward and backward for one mini-batch (i.e., $F_{0,0}, F_{0,1}, \dots, B_{0,0}$ and all updates) requires 36 time units
- With Gpipe, computation time reduces to 15 time units

10

Gpipe



- Chain rule works as normal
- Reduce training time by pipelining
- However, still need to inser bubble for each mini-batch

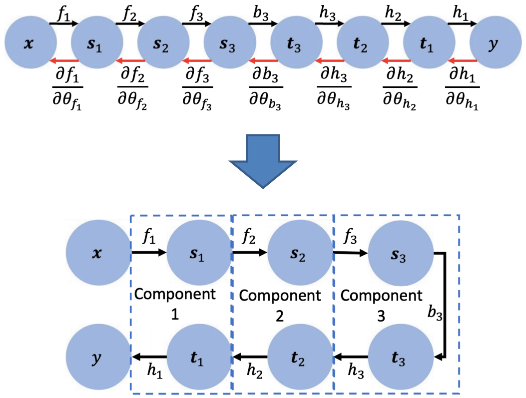
11

Associated learning (AL)

- Modularize a DNN into smaller components
  - Each small component has a local objective
  - Local objectives are mutually indenpdent
- AL fully pipelines the forward and backward process, because the objectives are mutually independent

12

Modularize DNN (e.g., 6 hidden layers)



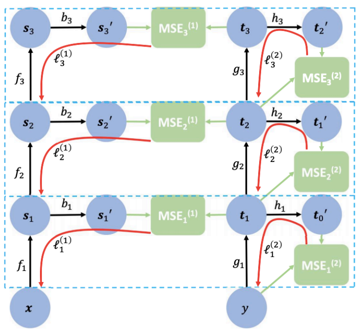
13

How to pipeline

Time unit	1	2	3	4	5	6	7	...
1st mini-batch	Task 1	Task 2	Task 3					
2nd mini-batch		Task 1	Task 2	Task 3				
3rd mini-batch			Task 1	Task 2	Task 3			
4th mini-batch				Task 1	Task 2	Task 3		
5th mini-batch					Task 1	Task 2	Task 3	
...								

14

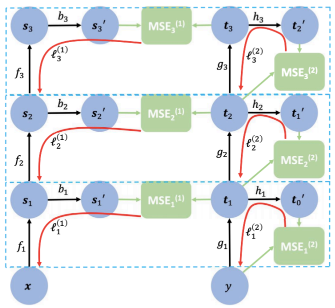
Associated learning – procedure



- Black arrow: forward
- Red arrow: backward gradient flow
- Green arrow: connect the variables that should be compared to minimize their associated distance

15

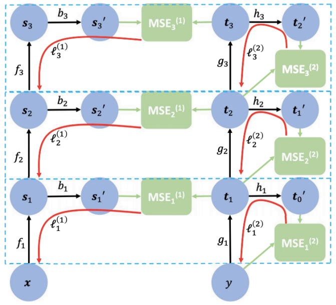
Associated learning – procedure



- Each component contains 2 local forward functions f_i and g_i
  - $s_i = f_i(s_{i-1})$
    - s_0 is x
  - $t_i = g_i(t_{i-1})$
    - t_0 is y
- Initially, we want to find f_i and g_i such that $s_i \approx t_i$ (local objective)
  – Turn out challenging
- Add “bridge function” b_i: find f_i, g_i, b_i such that
  - $s_i' = b_i(s_i) \approx t_i$

16

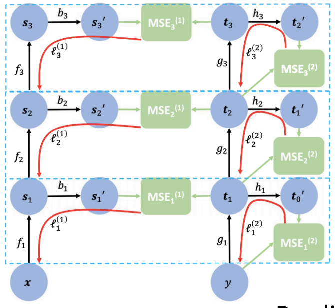
Associated learning – procedure



- The above training allows finding f_1, f_2, f_3, b_3, so we can perform
  $$b_3\Big(f_3\big(f_2(f_1(x))\big)\Big)$$
- We assume $s_3' \approx t_3$, so
  $$t_3 \approx b_3\Big(f_3\big(f_2(f_1(x))\big)\Big)$$
- How to do the following tranform?
  $$t_3 \rightarrow t_2 \rightarrow t_1 \rightarrow t_0 (= y)$$

17

Associated learning – procedure



- Look for h_i such that
  $$h_i(g_i(t_{i-1})) \approx t_{i-1}'$$
- Regarded as an autoencoder
  $$t_{i-1} \xrightarrow{g_i} t_i \xrightarrow{h_i} t_{i-1}'$$
- If we can find such h_is, then
  $$t_{i-1} \approx h_i(t_i)$$
- So we know how to
  $$t_3 \rightarrow t_2 \rightarrow t_1 \rightarrow t_0$$
- Prediction function
  $$y = h_1\left(h_2\left(h_3\left(b_3\left(f_3\left(f_2(f_1(\boldsymbol{x}))\right)\right)\right)\right)\right)$$

18

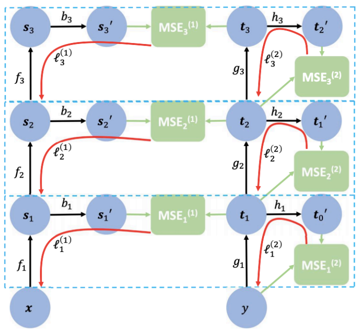
Associated learning

- Prediction function
  $$y = \underbrace{h_1\left(h_2\left(h_3\right.\right.}_{\text{backward function}}\left(b_3\left(\underbrace{f_3\left(f_2(f_1(\boldsymbol{x}))\right)}_{\text{Forward function}}\right)\right)\left.\left.\right)\right)$$

The last bridge function

19

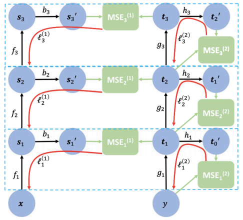
Effective and affiliated parameters



- During training, the model learns
  - f_is, b_is, g_is, h_is
- At prediction, only
  - f_is, h_is and last b_i are used
- We call f_is, h_is and last b_i effective parameters
- The others are affiliated parameters

20

Associated learning and pipeline



	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$
$(x^{(1)}, y^{(1)})$	$f_1^{(0)}, b_1^{(0)}, g_1^{(0)}, h_1^{(0)}$	$f_2^{(0)}, b_2^{(0)}, g_2^{(0)}, h_2^{(0)}$	$f_3^{(0)}, b_3^{(0)}, g_3^{(0)}, h_3^{(0)}$			
$(x^{(2)}, y^{(2)})$		$f_1^{(1)}, b_1^{(1)}, g_1^{(1)}, h_1^{(1)}$	$f_2^{(1)}, b_2^{(1)}, g_2^{(1)}, h_2^{(1)}$	$f_3^{(1)}, b_3^{(1)}, g_3^{(1)}, h_3^{(1)}$		
$(x^{(3)}, y^{(3)})$			$f_1^{(2)}, b_1^{(2)}, g_1^{(2)}, h_1^{(2)}$	$f_2^{(2)}, b_2^{(2)}, g_2^{(2)}, h_2^{(2)}$	$f_3^{(2)}, b_3^{(2)}, g_3^{(2)}, h_3^{(2)}$	
$(x^{(4)}, y^{(4)})$				$f_1^{(3)}, b_1^{(3)}, g_1^{(3)}, h_1^{(3)}$	$f_2^{(3)}, b_2^{(3)}, g_2^{(3)}, h_2^{(3)}$	$f_3^{(3)}, b_3^{(3)}, g_3^{(3)}, h_3^{(3)}$
$(x^{(5)}, y^{(5)})$					$f_1^{(4)}, b_1^{(4)}, g_1^{(4)}, h_1^{(4)}$	$f_2^{(4)}, b_2^{(4)}, g_2^{(4)}, h_2^{(4)}$
$(x^{(5)}, y^{(5)})$						$f_1^{(5)}, b_1^{(5)}, g_1^{(5)}, h_1^{(5)}$

21

Experimental settings

- Dataset
  - MNIST
  - CIFAR-10
  - CIFAR-100
- NN structure
  - MLP
  - Vanilla CNN
  - VGG Net
  - ResNet-20
  - ResNet-32

22

Experimental results on MNIST

	BP	AL
MLP	$98.5 \pm 0.0\%$	$\mathbf{98.6 \pm 0.0\%}$
Vanilla CNN	$99.4 \pm 0.0\%$	$\mathbf{99.5 \pm 0.0\%}$

- MLP: 5 hidden layers
  - 1024 x 1024 x 5120 x 1024 x 1024
- Vanilla CNN: 13 hidden layers
  - First four layers: (3 x 3 x 32) kernels
  - Next four layers: (3 x 3 x 64) kernels
  - Last five layers: fully connected 1280 x 256 x 256 x 256 x 256
- We didn't try more complex networks, as MLP and Vanilla CNN are extremely good

23

Experimental results on CIFAR-10

	BP	AL
MLP	$60.6 \pm 0.3\%$	$\mathbf{62.8 \pm 0.2\%}$
Vanilla CNN	$85.2 \pm 0.4\%$	$\mathbf{85.8 \pm 0.1\%}$
ResNet-20	$\mathbf{91.2 \pm 0.4\%}$	$89.1 \pm 0.5\%$
ResNet-32	$\mathbf{92.0 \pm 0.2\%}$	$88.7 \pm 0.4\%$
VGG	$92.3 \pm 0.2\%$	$\mathbf{92.6 \pm 0.1\%}$

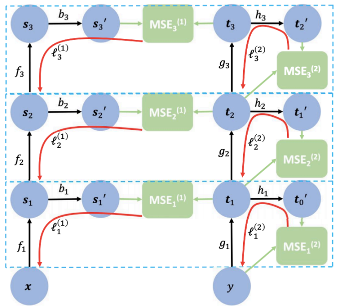
24

Experimental results on CIFAR-100

	BP	AL
MLP	$26.5 \pm 0.4\%$	$\mathbf{29.7 \pm 0.2\%}$
Vanilla CNN	$51.1 \pm 0.2\%$	$\mathbf{52.2 \pm 0.5\%}$
ResNet-20	$\mathbf{63.7 \pm 0.2\%}$	$61.0 \pm 0.6\%$
ResNet-32	$\mathbf{63.7 \pm 0.3\%}$	$59.0 \pm 1.6\%$
VGG	$65.8 \pm 0.3\%$	$\mathbf{67.1 \pm 0.3\%}$

25

Associated loss at different layers on MNIST after 200 epochs



Number of component layers	1 layer	2 layers	3 layers
$\|s_1' - t_1\|^2$	1.2488×10^{-5}	1.5469×10^{-5}	1.2219×10^{-5}
$\|s_2' - t_2\|^2$	-	3.5818×10^{-7}	3.8033×10^{-7}
$\|s_3' - t_3\|^2$	-		6.7192×10^{-10}

Difference between s_i' and t_i indeed becomes smaller in the "higher" layer

26

Number of layers vs training/test accuracies

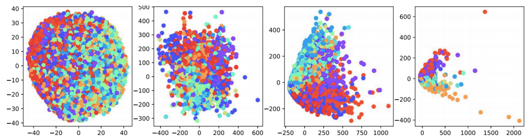
Number of component layers	1 layer	2 layers	3 layers
Training accuracy	1.0	1.0	1.0
Test accuracy	0.9849	0.9860	0.9871

Increasing layers increases test accuracies

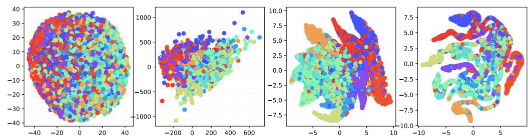
27

t-SNE visualization of MLP on CIFAR-10

- Raw data, 2nd layer, 4th layer, and output layer when training by BP



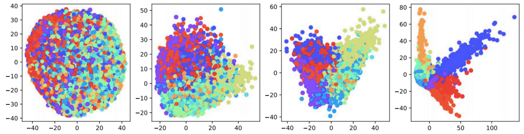
- Raw data, 2nd layer, 4th layer, and output layer when training by AL



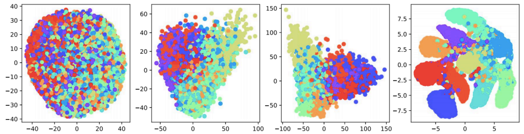
28

t-SNE visualization of CNN on CIFAR-10

- Raw data, 4th layer, 8th layer, and 12th layer when training by BP



- Raw data, 4th layer, 8th layer, and 12th layer when training by AL



29

Inter- and intra-distances and ratios of the two

Dataset	Network	Method	Interclass distance	Intraclass distance	Inter:Intra ratio
CIFAR-10	MLP	BP	39.36	67.97	0.58
		AL	0.73	0.66	**1.11**
	Vanilla CNN	BP	41.82	26.87	1.56
		AL	1.17	0.36	**3.25**
CIFAR-100	MLP	BP	114.42	342.65	0.33
		AL	0.23	0.28	**0.82**
	Vanilla CNN	BP	114.71	163.43	0.70
		AL	0.55	0.51	**1.08**

30

Summary

- Propose a method , AL, to decompose end-to-end backpropagation into many short gradient flows
- Fully pipelined the learning
- Experiments of multiple network structures on multiple datsets show that AL is comparable (and sometimes better) than BP

31

Discussion

- We implemented AL, but we didn't implement a "pipelined" AL
  - How to implement pipeline in Tensorflow or PyTorch?
- Most components in AL do not receive messages directly from the target, but AL performs comparable to BP.
  - This is a nice surprise, but we don't know why
  - If we know why AL works, perhaps we can design better network structure?
- Does AL converge? Does the loss function converge to minima?
- Experiments on other (larger) datasets?
- If you are interested in any of the above topics, please let me know

32

More information

- Associated Learning: Decomposing End-to-End Backpropagation Based on Autoencoders and Target Propagation
  – Yu-Wei Kao (高聿緯), Hung-Hsuan Chen (陳弘軒)
  – MIT Neural Computation 33(1), 2021
  – https://arxiv.org/pdf/1906.05560.pdf

Quiz

- What is backward locking?
- Compare pipeline and parallel computing
- What is the key concept of associated learning?