Highly parallel auto-differentiate system for deep learning

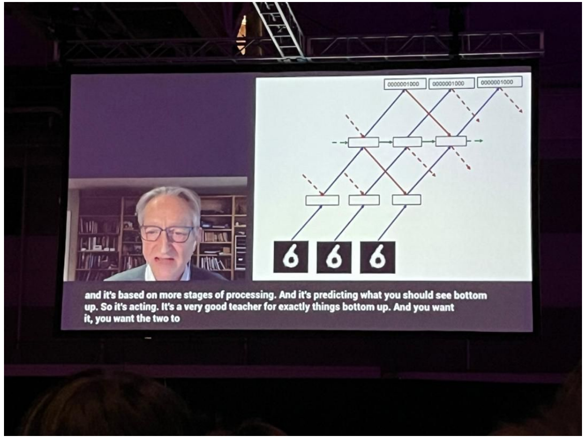
Zeliang Zhang

Overview

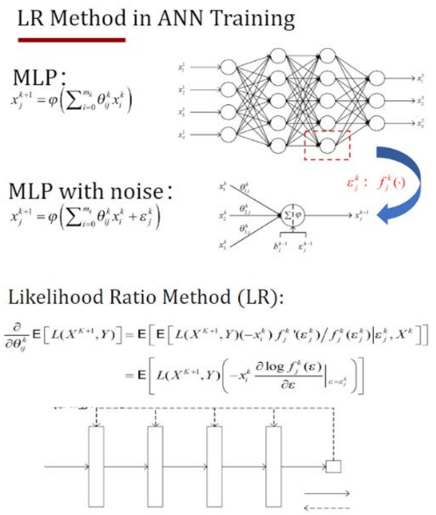
- Background & Related work
- Optimized primitives using GPU Tensor Cores
- High performance neural network training w/o BP
- Evaluations
- Future work

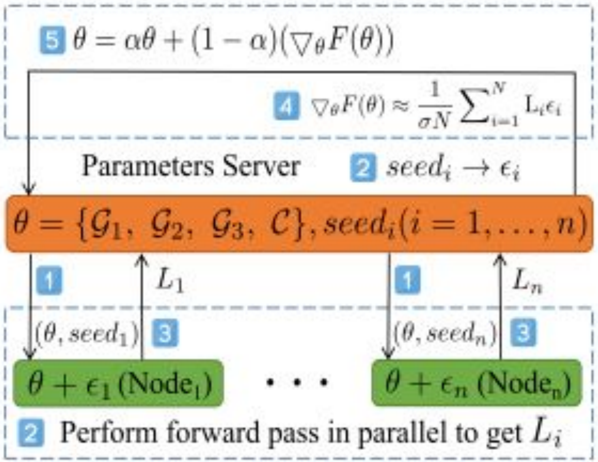
Background & Related work

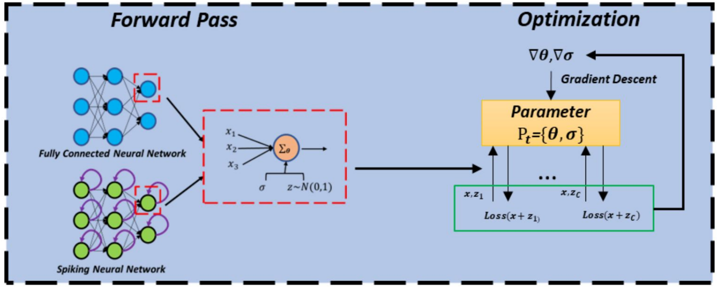
- Pattern: Big model, Big data, Big time
- Acceleration: CPU/GPU/TPU/NPU
- Optimization: Chain-rule based backpropagation



But in fact, chain rule is not the only way to get the gradient......





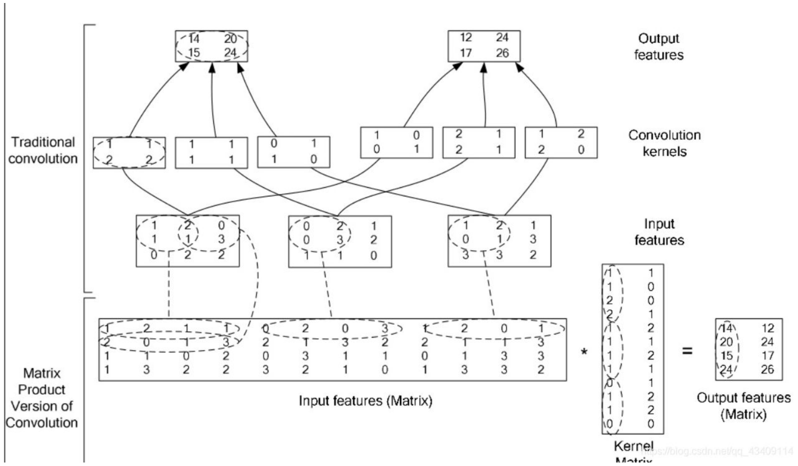


Optimized primitives using GPU tensor cores

“Primitives” in NN:

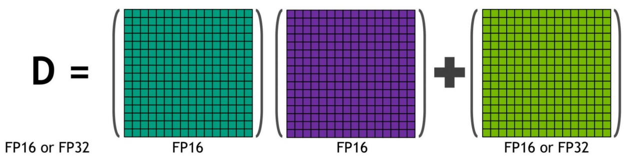
- Convolutional layer
- Fully connected layer
- Element-wise activation
- Hadamard product

Matrix multiplications

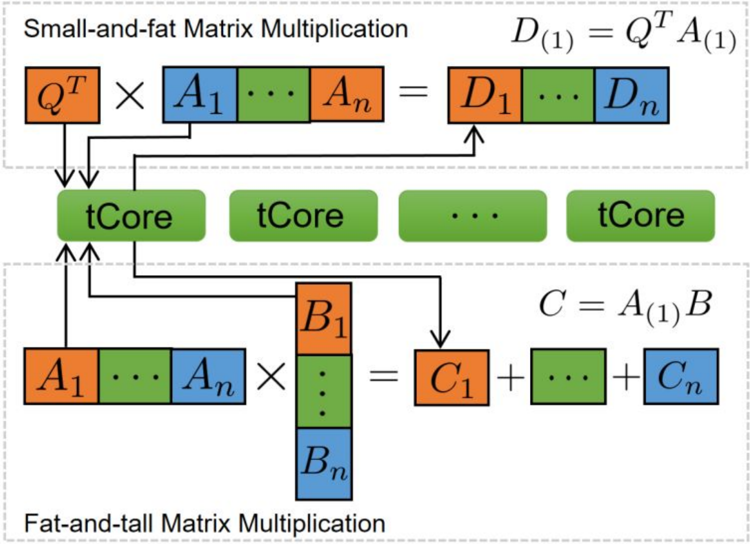


Optimized primitives using GPU tensor cores

```
wmma::mma_sync(Dmat, Amat, Bmat, Cmat);
```



D = AB + C



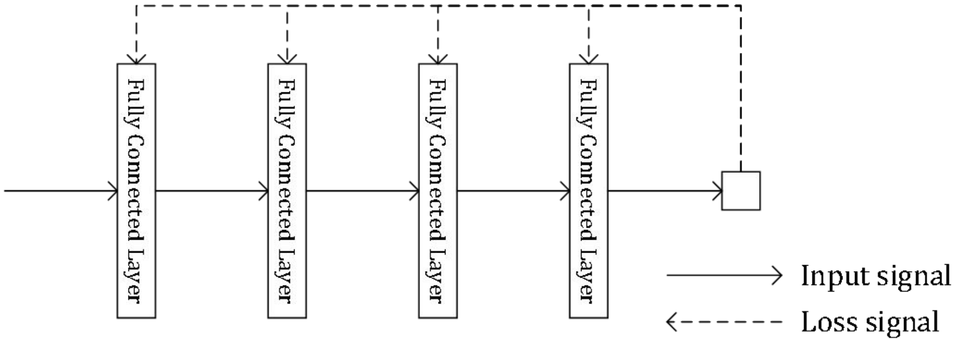
*What we need to do is just to accelerate the **matrix multiplications***

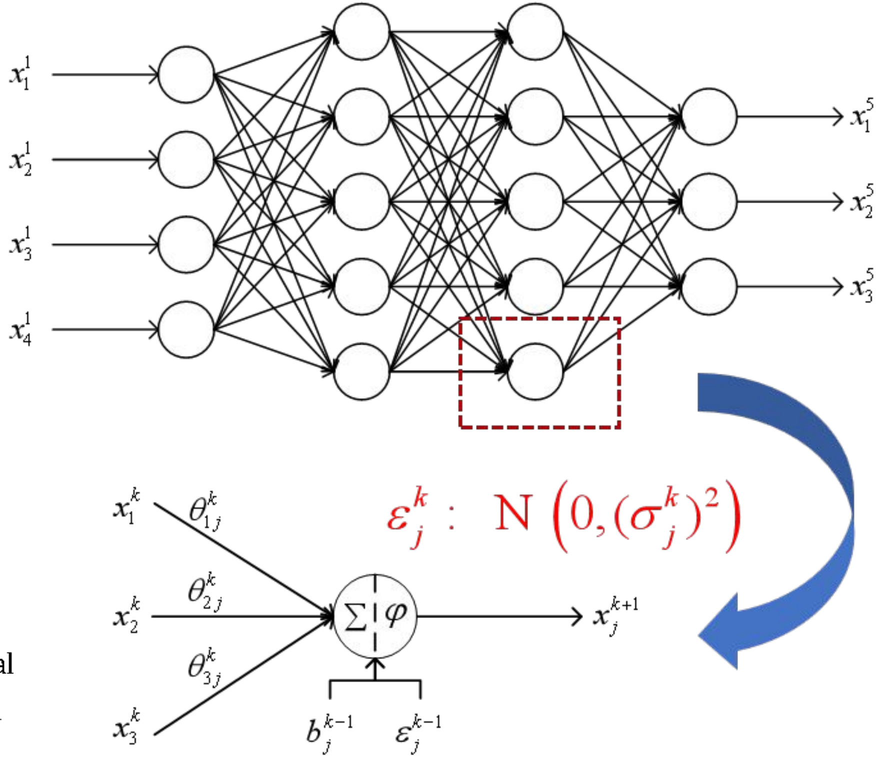
Neural network training w/o BP

Forward: $\tilde{v}^l = \theta^l x^l + b^l, \quad v^l \sim \mathrm{N}\left(\tilde{v}^l, \Sigma^l\right)$

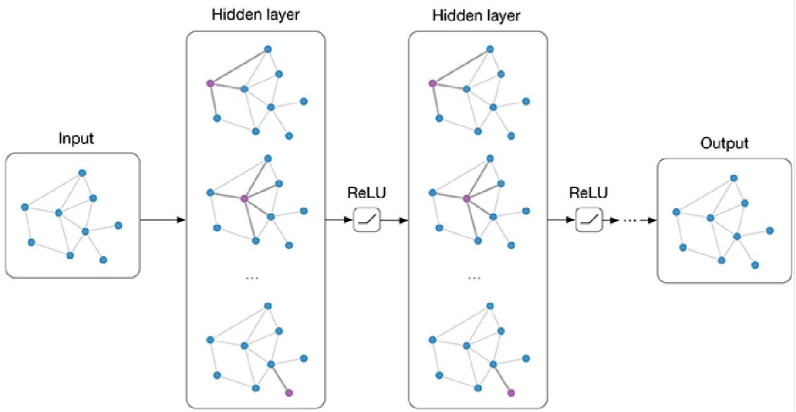
$\boxed{+\ \varepsilon^l}$

"Backward": $\nabla_{\theta^l}\mathbb{E}\left[L(x^\tau)\right] = \mathbb{E}\left[L(x^\tau)(\Sigma^l)^{-\frac{1}{2}}\varepsilon^l(x^l)^\top\right]$





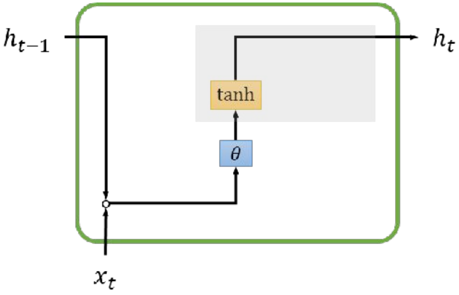
High performance neural network training w/o BP



$$f(H^{(l)},A)=\sigma\left(\hat{D}^{-0.5}\hat{A}\hat{D}^{-0.5}H^{(l)}W^{(l)}\right)$$

$$\hat{D}=\mathrm{diag}(A),\ \hat{A}=A+I$$

$$f(H^{(l)},A)=\sigma\left(\hat{D}^{-0.5}\hat{A}\hat{D}^{-0.5}H^{(l)}W^{(l)}+Z^{(l)}\right)$$



RNN

Forward:

$$(h_t,c_t)=\varphi(u_t,v_t),\quad u_t=\theta^{hh}h_{t-1}+b^{hh}+z_t^{hh},\quad v_t=\theta^{xh}x_t+b^{xh}+z_t^{xh}$$

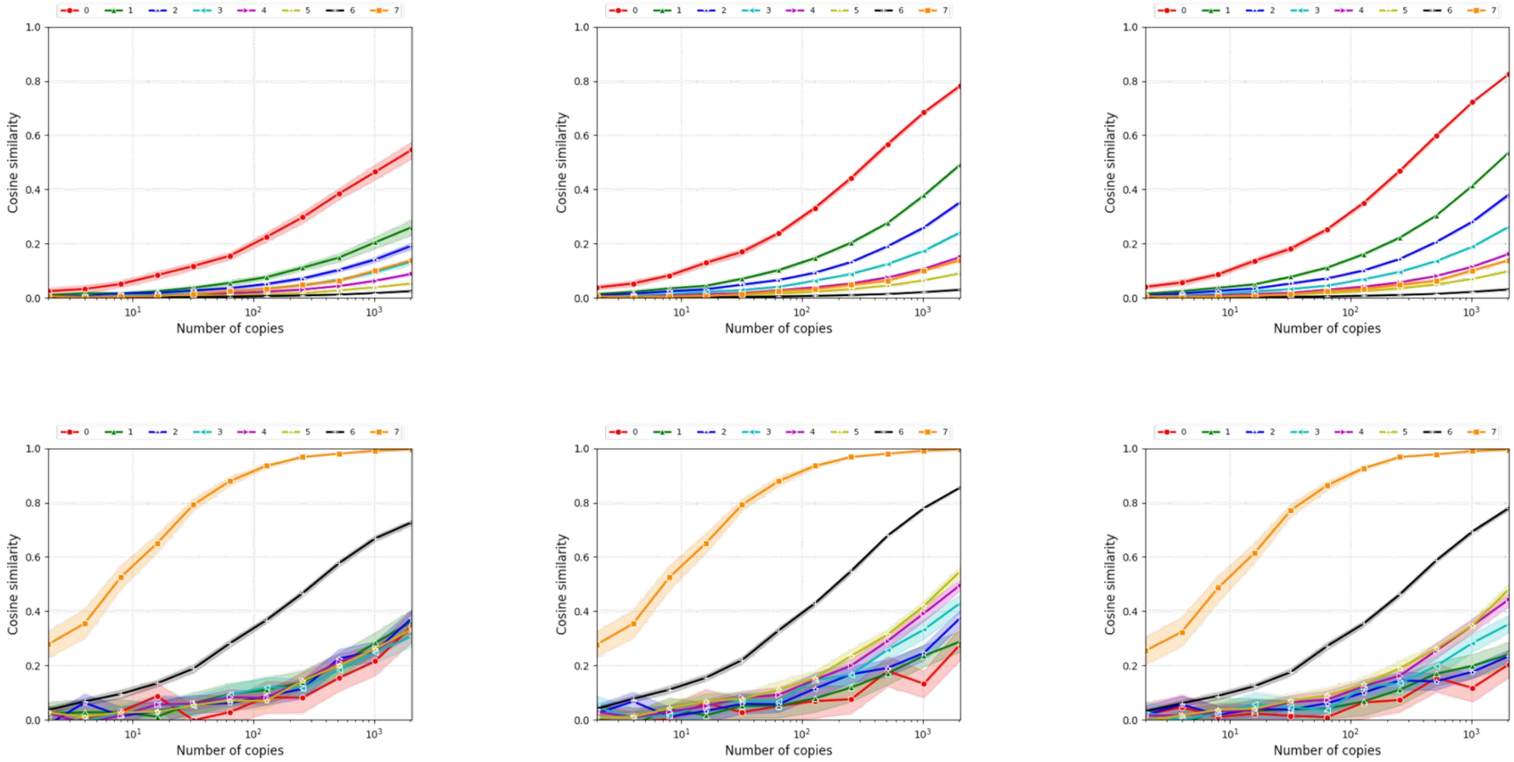
$$z_t^{hh}=(\Sigma^{hh})^{-\frac{1}{2}}\varepsilon_t^{hh},\ \varepsilon_t^{hh}\sim \mathrm{N}(0,I),\quad z_t^{xh}=(\Sigma^{xh})^{-\frac{1}{2}}\varepsilon_t^{xh},\ \varepsilon_t^{xh}\sim \mathrm{N}(0,I)$$

Backward:

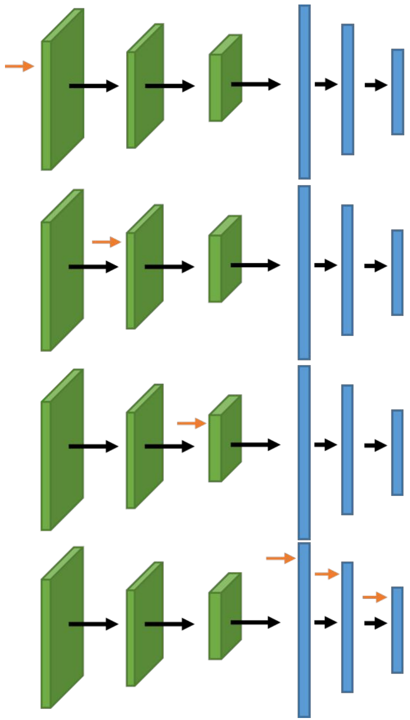
$$\nabla_{\theta^{hh}}\mathbb{E}\left[L(h,c;\omega)\right]=\mathbb{E}\left[L(h,c;\omega)\sum_{t=1}^{T}(\Sigma^{hh})^{-\frac{1}{2}}\varepsilon_t^{hh}h_t^{\top}\right]$$

$$\nabla_{\theta^{xh}}\mathbb{E}\left[L(h,c;\omega)\right]=\mathbb{E}\left[L(h,c;\omega)\sum_{t=1}^{T}(\Sigma^{xh})^{-\frac{1}{2}}\varepsilon_t^{xh}x_t^{\top}\right]$$

Problems & Challenges



Solutions for variance reduction



Layer-wise perturbation

$$I = \frac{1}{N}\sum_{i=1}^{N} g(\mathbf{X}_i) = \frac{1}{N}\sum_{i=1}^{N} G_i$$

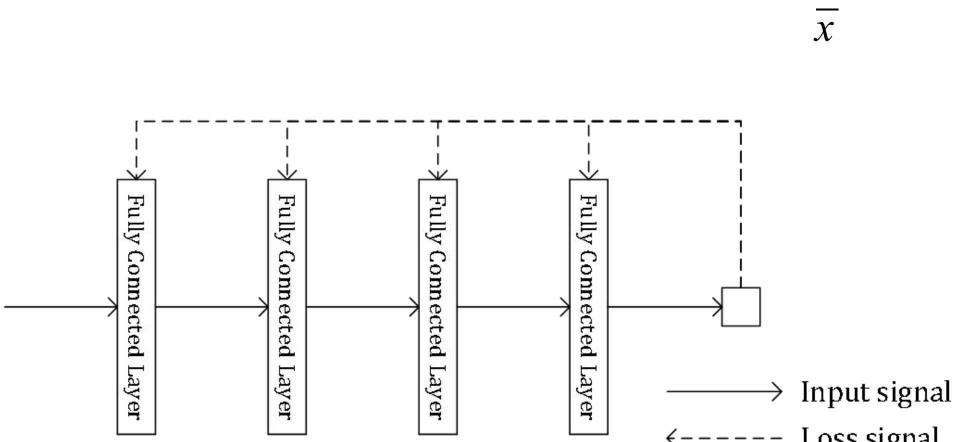
$$= \frac{1}{n}\sum_{i=1}^{n} \frac{G_{2i-1}+G_{2i}}{2} = \frac{1}{n}\sum_{i=1}^{n} H_i$$

$$G_i = g(\mathbf{X}_i) \text{ and } H_i = \frac{G_{2i-1}+G_{2i}}{2}$$

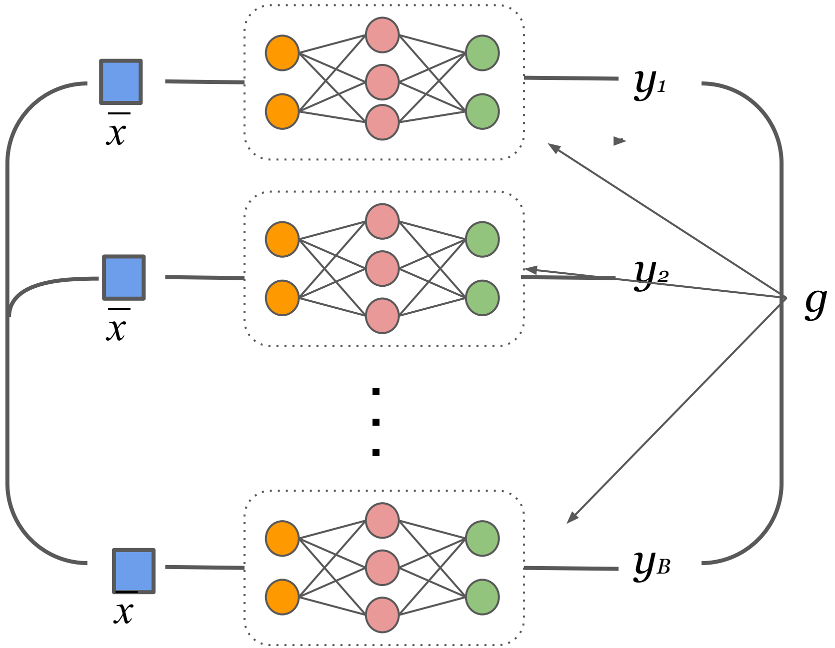
$$\begin{aligned} \sigma_h^2 &= \operatorname{Var}(H_i) \\ &= \frac{1}{4}\left(\sigma_g^2 + \sigma_g^2 + 2\operatorname{Cov}(H_{2i-1}, H_{2i})\right) \\ &= \frac{1}{2}\left(\sigma_g^2 + \operatorname{Cov}(G_{2i-1}, G_{2i})\right). \end{aligned}$$

Antithetic Variable

Implementations



Layer-wise parallel



Model-level parallel

Evaluations

Model	Method	Acc.	Time/epoch
MLP	BP	99.5	1 min 17 s
	LR-F	92.2	3 min 09 s
	LR-L	92.2	3 min 45 s
	LR-M	99.5	58 s

Results on MNIST dataset

Model	Method	Acc.	Time/epoch
RNN	BP	88.3	3 min 15 s
	LR-F	84.2	4 min 35 s
	LR-L	84.3	5 min 27 s
	LR-M	88.4	2 min 20 s

Results on Ag-News dataset

Model	Method	Acc.	Time/epoch
GCN	BP	80.4	1 min 51 s
	LR-F	75.6	3 min 23 s
	LR-L	75.2	4 min 19 s
	LR-M	78.8	1 min 43 s

Results on Cora dataset

Limitations & Future work

Improvement for the pure layer-wise parallel strategy

Fine-grained parallelism design

Better suitable data structure

…

Thanks