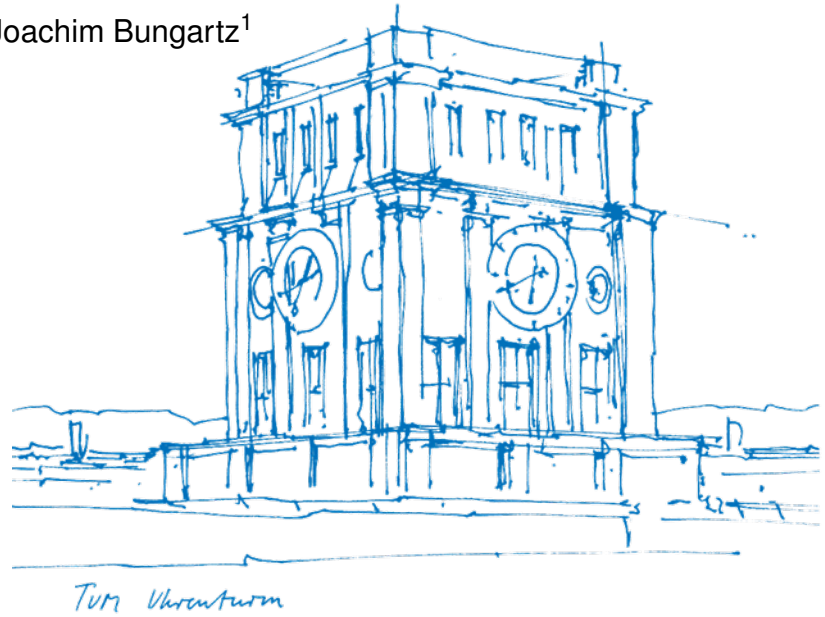
A Fast Multipole Method for Training Neural Networks

Severin Reiz[1] , Chen Chao[2] , Tobias Neckel[1], George Biros[2], Hans-Joachim Bungartz[1]

ISC PhD Forum, June 17th 2019

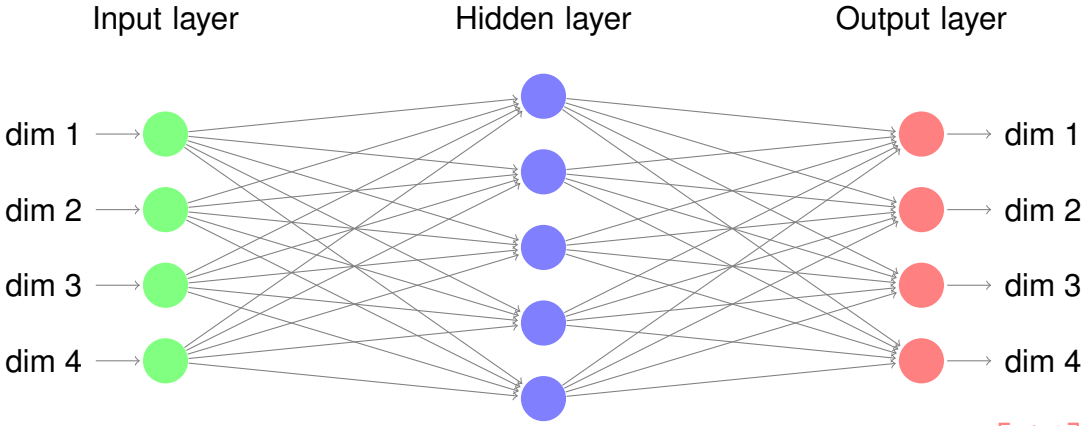


exact $\rightarrow$ approximate

- Hessian Matrix Vector product
  - Barnes-Hut Scheme: $\mathbf{N^2 \rightarrow N \log N}$
  - Fast Multipole: $\mathbf{N^2 \rightarrow N}$
- Hessian Pseudo Inverse
  - Sherman-Morrison-Woodbury $\mathbf{N^3 \rightarrow N \log^2 N}$

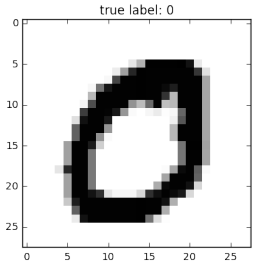
[1]Technical University of Munich, Germany
[2]University of Texas at Austin, USA

Multilayer Perceptron Model



$$\begin{bmatrix} x_0^{l=0} \\ x_1^{\ell=0} \\ x_2^{\ell=0} \\ x_3^{\ell=0} \end{bmatrix} \quad \begin{bmatrix} x_0^{l=1} \\ x_1^{\ell=1} \\ x_2^{\ell=1} \\ x_3^{\ell=1} \\ x_4^{\ell=1} \end{bmatrix} = s(W_{\ell=1} \cdot \begin{bmatrix} x_0^{\ell=0} \\ x_1^{\ell=0} \\ x_2^{\ell=0} \\ x_3^{\ell=0} \end{bmatrix}) \quad [x]^{\ell=2} = s(W_{\ell=2} \cdot \begin{bmatrix} x_0^{\ell=1} \\ x_1^{\ell=1} \\ x_2^{\ell=1} \\ x_3^{\ell=1} \\ x_4^{\ell=1} \end{bmatrix})$$

$$W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$$



$$\begin{bmatrix} x_0^{\ell=0} \\ x_1^{\ell=0} \\ \vdots \\ x_{784}^{\ell=0} \end{bmatrix} \rightarrow \ldots \rightarrow \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{10} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Loss function f: mean squared error

$$f\left(x^L, y\right) = \left\| x^{L_{max}} - y \right\|^2$$

Optimize weight tensor W = $[W_{\ell=0}, W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}, \ldots]$

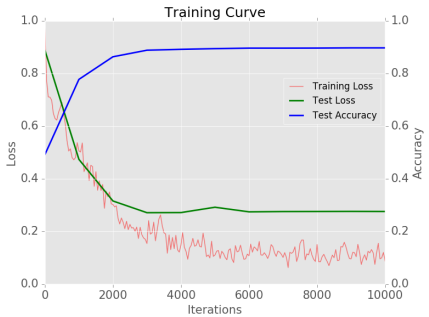
Computational Setup

$$W = [\begin{bmatrix} w_o & w_1 & \cdots & w_{d_0-1} \\ w_{d_0} & w_{d_0+1} & \cdots & \\ \vdots & & & \end{bmatrix}, \ldots], \text{with N := total number of weights}$$

Gradient: $g = \frac{df}{dW} \to \mathbb{R}^{N \times 1}$

Hessian: $H = \frac{d^2 f}{dW^2} \to \mathbb{R}^{N \times N}$

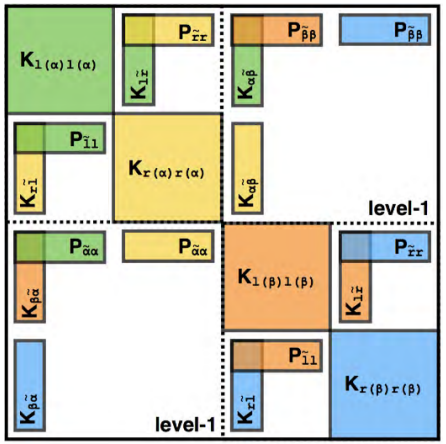
Hessian used for analyzing, shrinking, speed up training [see poster]



Challenges and Remedy

- Network sizes of >1M parameters are common
- Infeasible to store **dense** Hessian

Idea: Hierarchical compression



GOFMM[3]: Hierarchical Off-Diagonal Low-Rank (HODLR)

Hierarchical compression

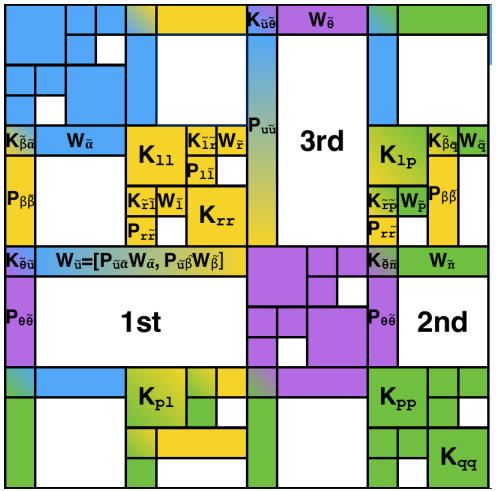
- Matrices are not sparse (few non-zero entries)
- data-sparse (can be described by only few data entries)

$$\tilde{H}_{\alpha\alpha} = \begin{bmatrix} \tilde{H}_{\mathtt{ll}} & 0 \\ 0 & \tilde{H}_{\mathtt{rr}} \end{bmatrix} + \begin{bmatrix} 0 & UV_{\mathtt{lr}} \\ UV_{\mathtt{rl}} & 0 \end{bmatrix} + \begin{bmatrix} 0 & S_{\mathtt{lr}} \\ S_{\mathtt{rl}} & 0 \end{bmatrix}$$

- Hessian Matrix Vector product
  - Barnes-Hut Scheme: $\mathbf{N^2} \rightarrow \mathbf{N} \log \mathbf{N}$
  - Fast Multipole: $\mathbf{N^2} \rightarrow \mathbf{N}$
- Hessian Pseudo Inverse
  - Sherman-Morrison-Woodbury $\mathbf{N^3} \rightarrow \mathbf{N} \log^2 \mathbf{N}$

Distributed parallel compression

- Randomized linear algebra: sampling
- From `LAPACK`: `GEQP3` and `TRSM`



[3] C. Yu, S. Reiz and G. Biros: Distributed-Memory Hierarchical Compression of Dense SPD Matrices, In SC '18: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis

Results: HODLR Compression of Modern Neural Nets

- `acc`uracy schemes, `Comp`ression time in s, `Mult`iplication time, %K fraction uncompressed entries, ε_F error, `Fact`orize time
- **4 nodes** on TACC Stampede2: 2.1GHz (1.4-3.7GHz), Intel Xeon Platinum 8160 “Skylake"(each with 48 cores $\rightarrow$ 192 cores)

Matrix Vector Product

$$u = \tilde{H} \cdot w$$

		Parameters		GOFMM results			
net	N	`scheme`	`acc`	`Comp`	`Mult`	%K	ε_F
mnist	16.5k	`FMM`	`low`	0.5	0.1	19.17%	2E−2
		`FMM`	`high`	2.8	0.1	23.98%	1E−4
enco	121k	`HSS`	`low`	2.8	0.0	0.25%	1E−1
		`HSS`	`high`	37.9	0.6	3.42%	9E−5

Matrix Pseudo Inverse

$$w = \tilde{H}^{-1} \cdot u$$

		Parameters		GOFMM results			
net	N	`scheme`	`acc`	`Fact`		%K	ε_F
mnist	16.5k	`HSS`	`low`	0.1		0.41%	9E−4
		`HSS`	`high`	2.5		5.1%	1E−6
enco	121k	`HSS`	`low`	0.3		0.25%	4E3
		`HSS`	`high`	38.2		3.5%	1E−4

Larger networks and explanations at my poster :-)