Low-parametric deep learning

Francesco Tudisco
GSSI – Gran Sasso Science Institute

Data-driven paradigm

- Input
  – set of observations $(x, y) \in R^d \times R^k$
  – underlying model M

- Goal
  infer the map $f: x \mapsto y$ in a **fast** and **reliable** way

Supervised classification

$x =$

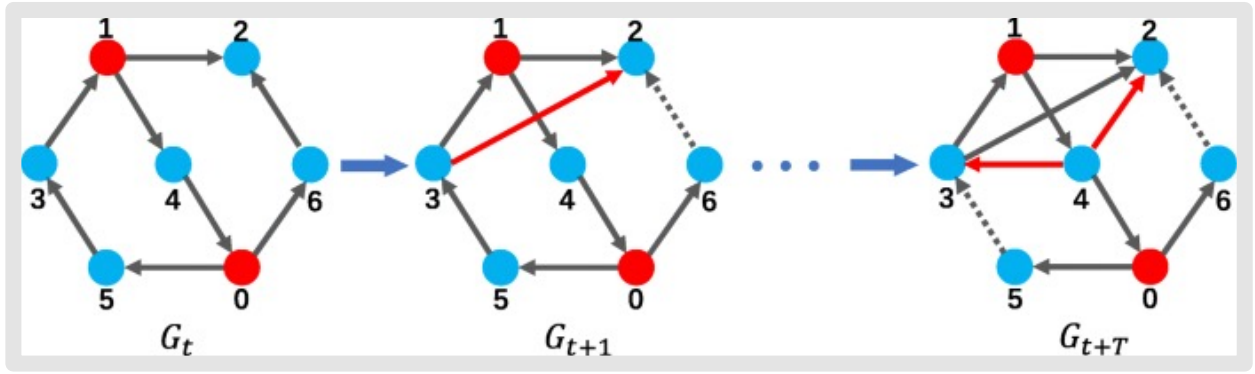
Dog 0.43

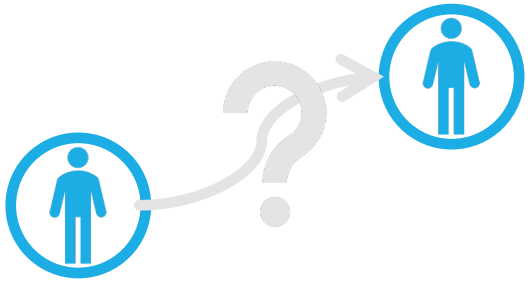
Mop 0.56

Pole 0.01

Prediction & recommendation

$x =$

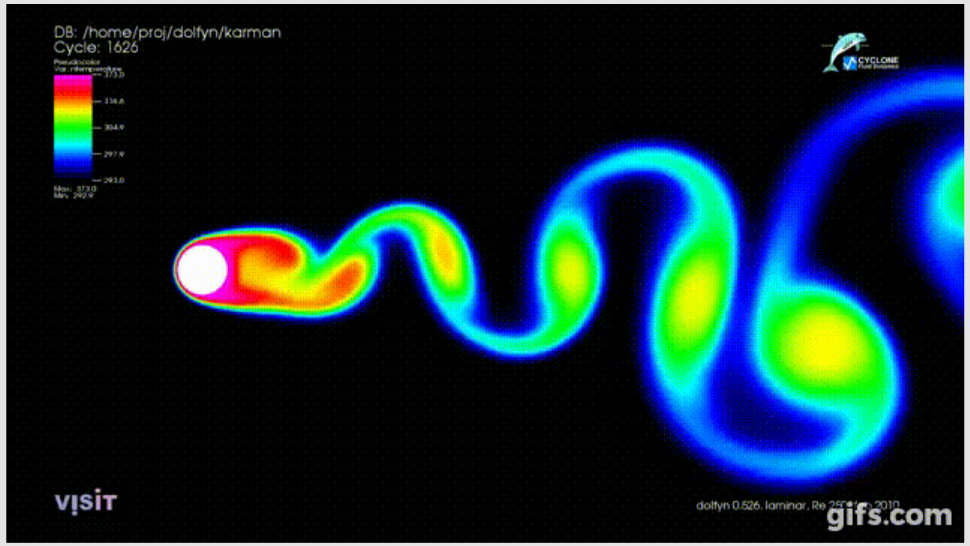




Scientific computing

$$\begin{cases} \mathcal{F}(f(x);\theta) = a(x) \\ \mathcal{B}(f(x)) = b(x) \end{cases}$$

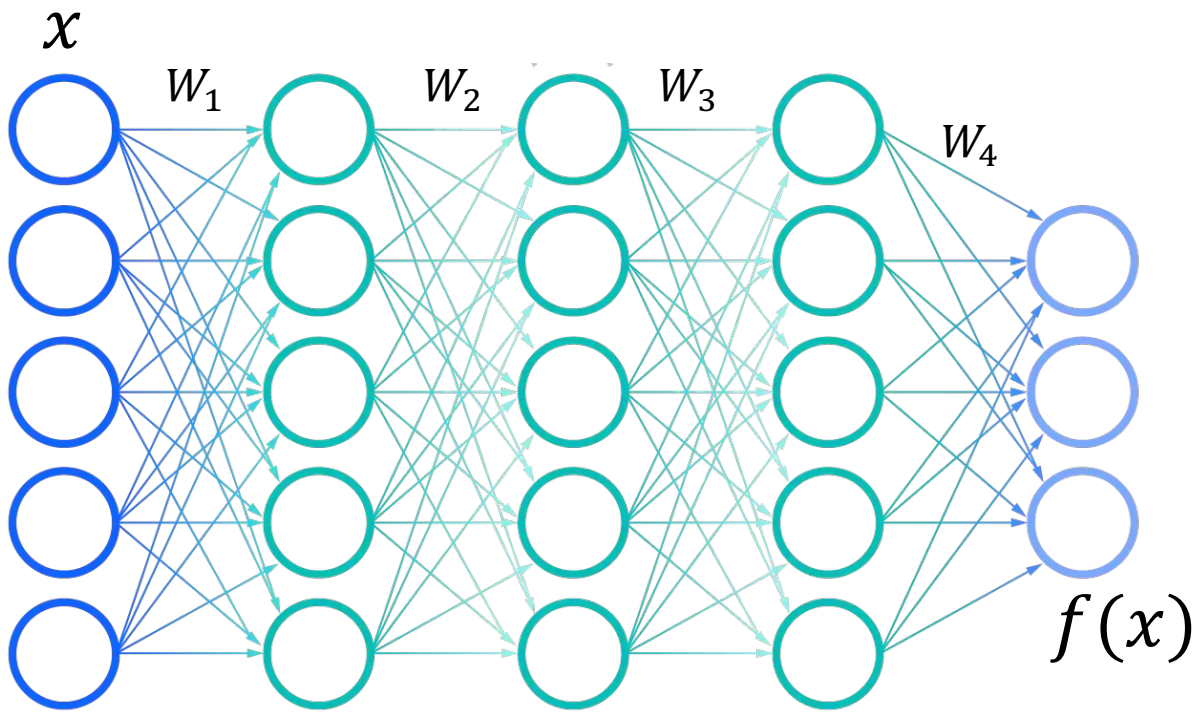
Field measurements



Artificial Neural Network ansatz

To compute $f : x \mapsto y$ we need an *ansatz* function space

Artificial Neural Networks (NNs) are the space of choice in most applications



Feed-forward Neural Nets

Parametric family of functions that we can write as

$$f(x; W) = \{\sigma_\ell \circ A_\ell \circ \cdots \circ \sigma_1 \circ A_1\}(x)$$

where:

- $W = (W_1, \ldots, W_\ell)$ are the parameters
- $A_i : R^{N_i} \to R^{N_{i+1}}$ are simple mappings, for instance
  - MatMult $A_i(x) = W_i x$
  - Convolution $A_i(x) = W_i * x$
  - Dot product $A_i(x) = W_i x \; (W_i x)^T$
- σ_i are entrywise nonlinear activation functions

Memory and computation footprints

Each time we do inference on x

We need to compute $f(x)$

This requires

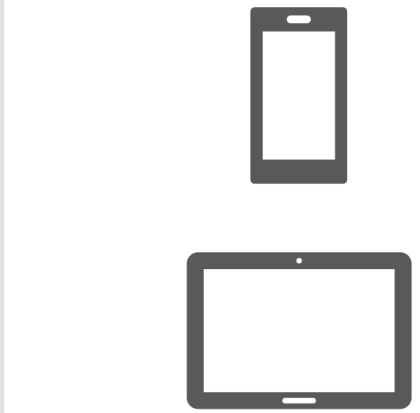
- To evaluate ℓ mappings $A_i \approx \sum_i N_i N_{i+1}$
- To evaluate ℓ activations $\sigma_i \approx \sum_i N_i$
- To store the weights $W_i \sim N_{i+1} \times N_i \approx \sum_i N_i N_{i+1}$

$N_i = N$		
	$O(\ell N^2)$	$O(\ell N^2)$
	Operations to make	Variables to store

Examples

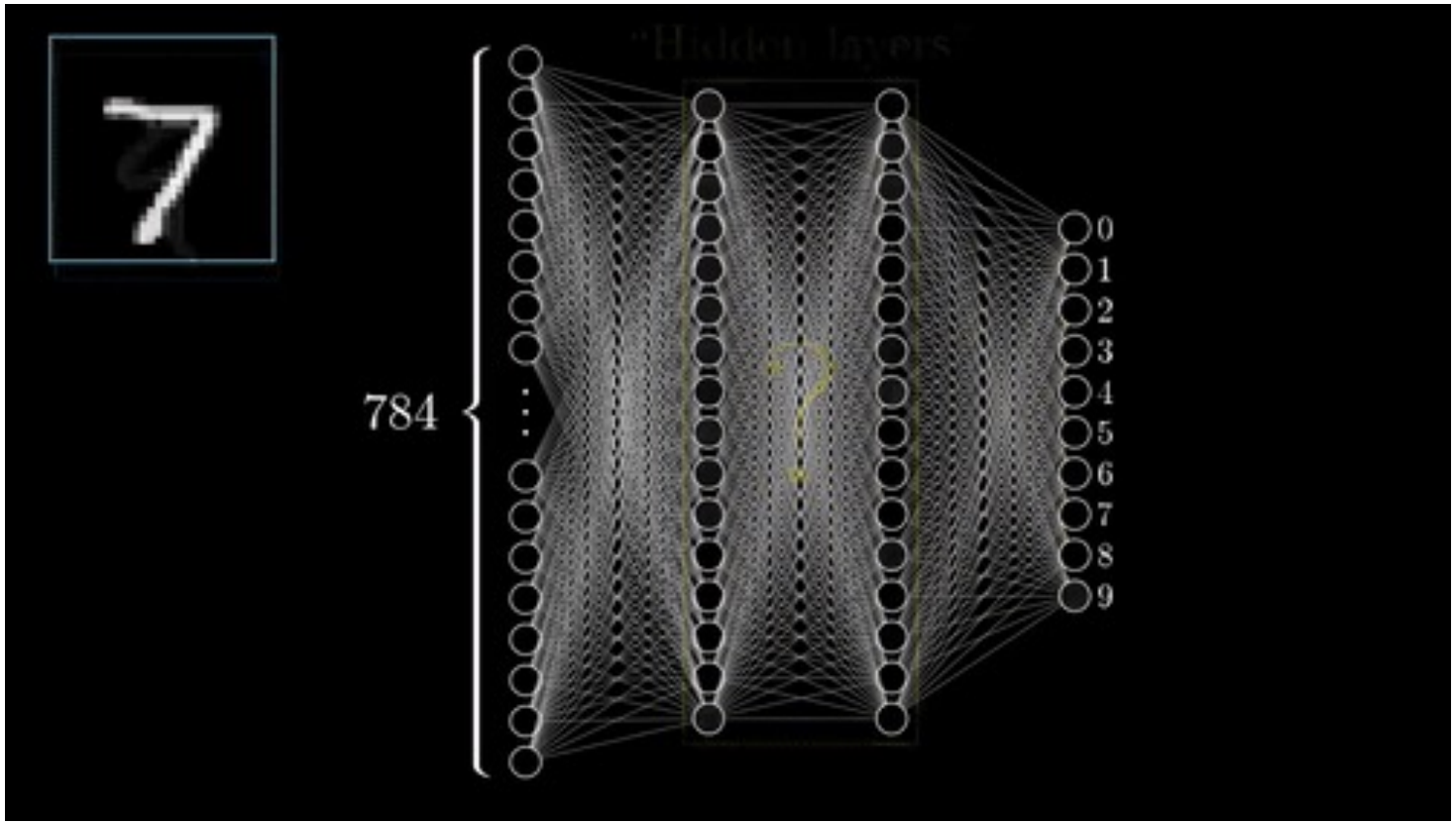
AlexNet ≈ 62 million parameters
VGG16 ≈ 135 million parameters
ResNet50 ≈ 23 million parameters
M-BERT ≈ 4 billion parameters
DALL-E ≈ 3.5 billion parameters

Prohibitive for **online learning** and for **limited-resource devices**, e.g. smartphones, satellites, drones, etc.



!

Not all weights are equally important

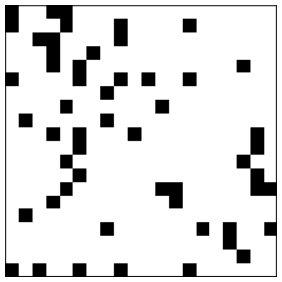


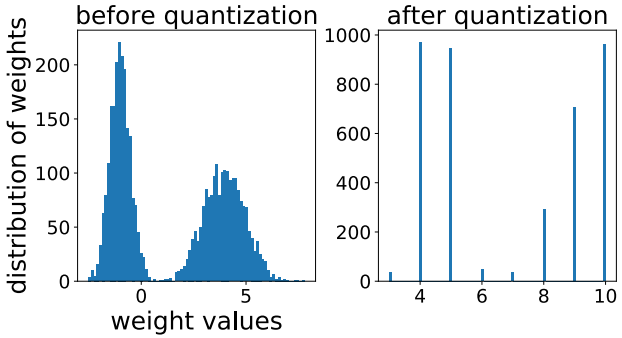
Model compression, aka **pruning**

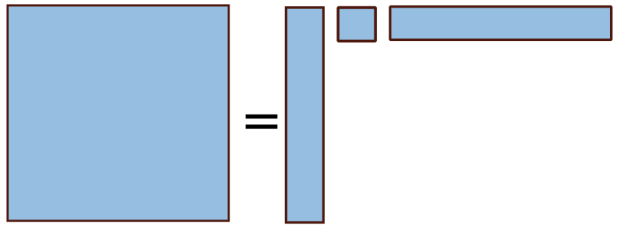
Constrain the parameter space

Most common examples:

- **sparsity**: $nnz(W_i) = O(N)$
- **quantization**: $(W_i)_{ij} \in \{-1, 0, +1\}$
- **low-rank**: $rank(W_i) = r_i \ll N$



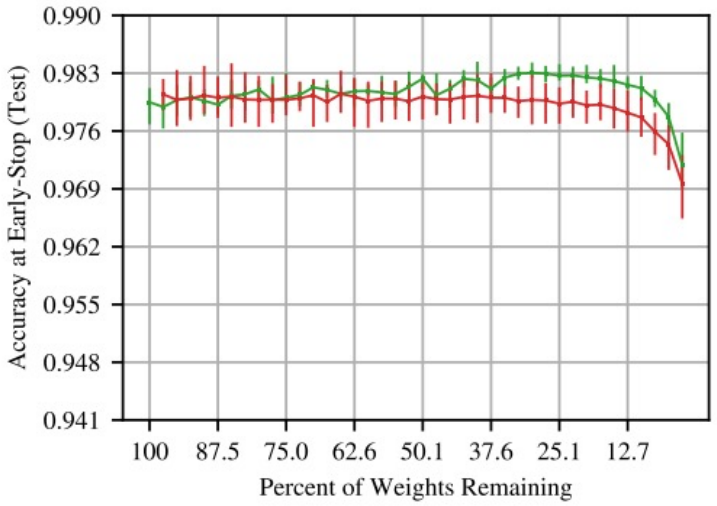




Lottery ticket hypothesis

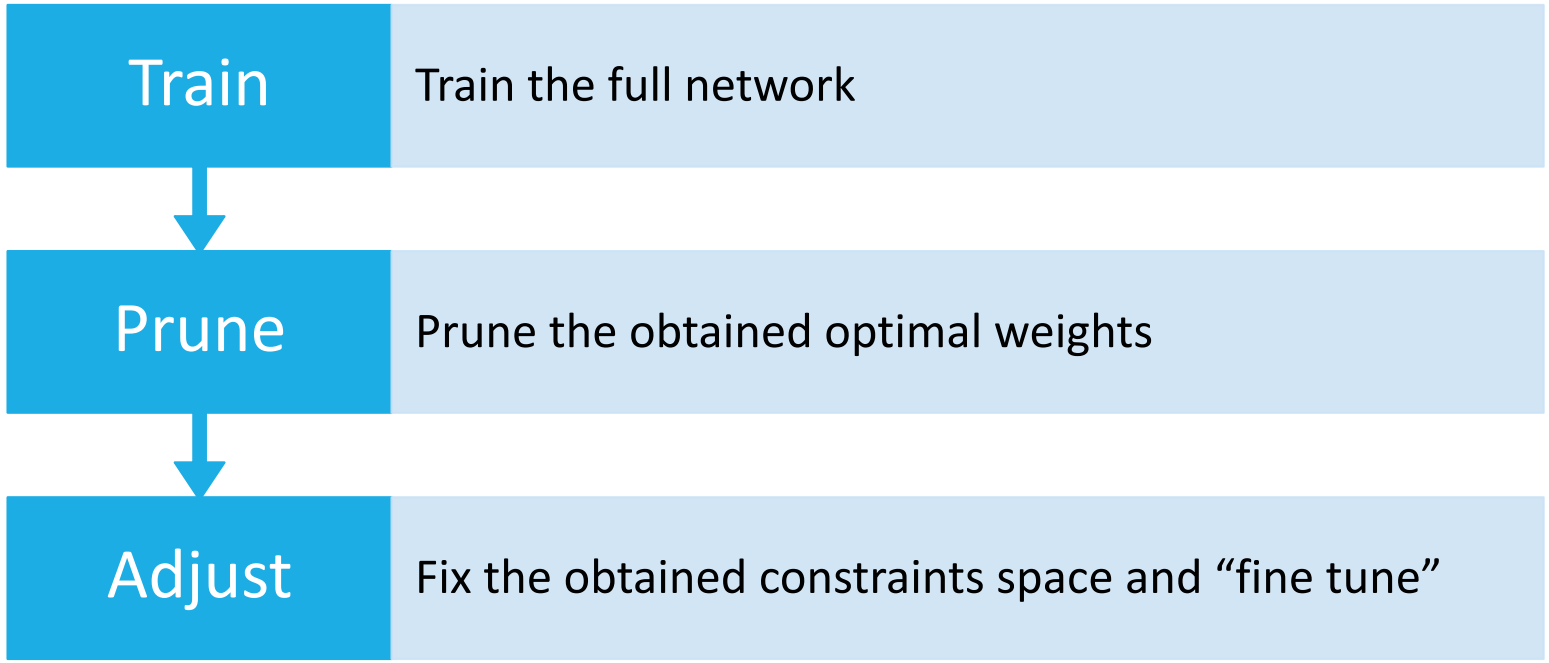
The lottery ticket hypothesis: finding sparse, trainable neural networks, J Frankle, M Carbin, ICLR 2019

Dense NNs contain subnetworks that, when trained in isolation and for the same number of epochs, can match the accuracy of the original full net.



Performance of different pruning strategies as the compression rate increases on LeNet5/MNIST
$98\% \rightarrow 97\%$

In practice


Train
Train the full network
Prune
Prune the obtained optimal weights
Adjust
Fix the obtained constraints space and “fine tune”

Is this happening by chance?

A network of depth ℓ can be approximated arbitrarily well by pruning a network of depth 2ℓ

📒 *Proving the lottery ticket hypothesis, E Malach et al, ICML20*

For any $\varepsilon > 0$, and any neural network $f(x; W)$ of depth ℓ and width N, there exist:

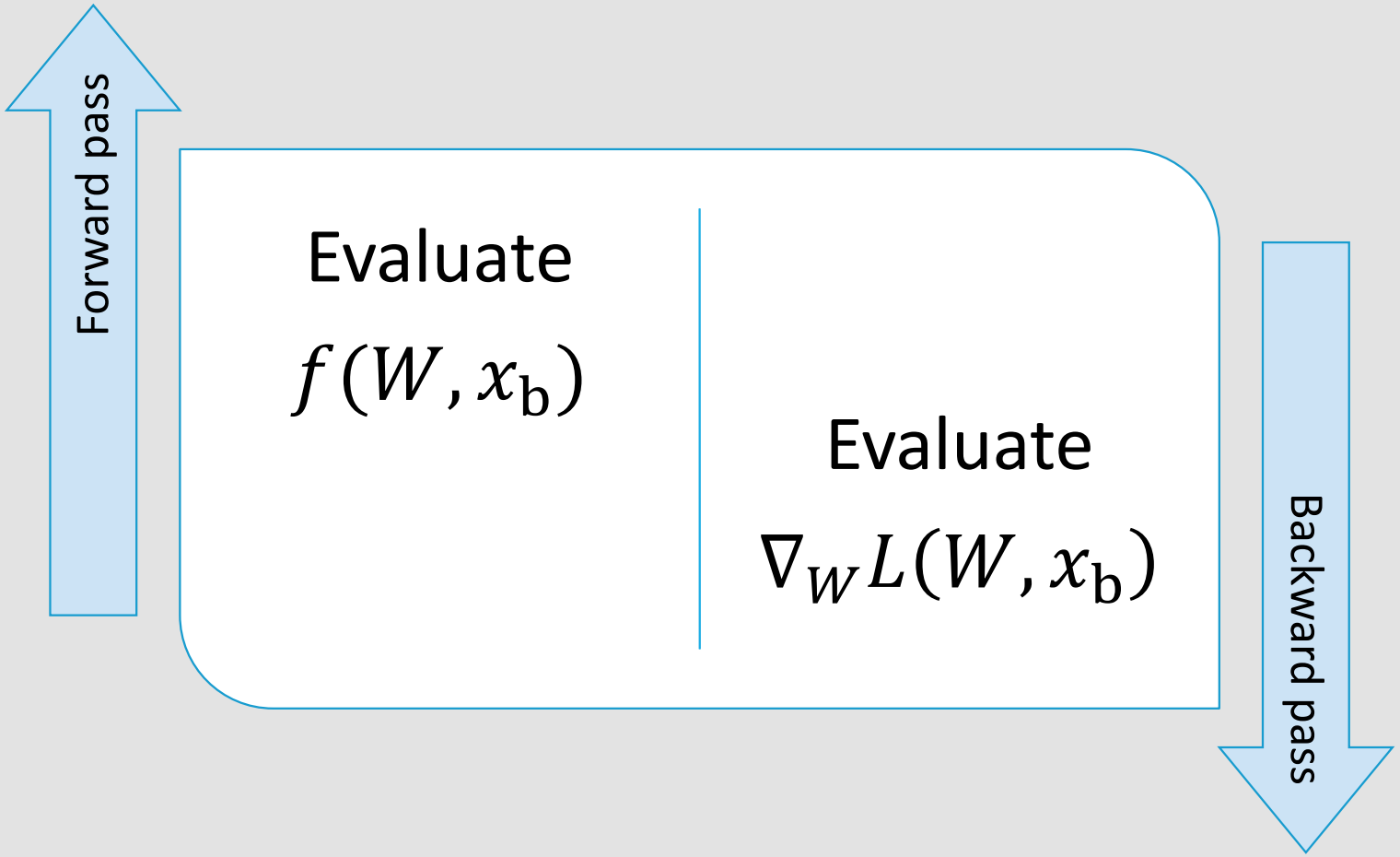
- a larger network g of depth 2ℓ and width $\text{poly}(\ell, N, \varepsilon^{-1})$,
- and a subnetwork $\tilde{f}$ with only $O(\ell N^2)$ parameters

such that $\left|\tilde{f} - f\right| \leq \varepsilon$ with probability at least $(1 - \varepsilon)$.

Problem(s): the result is not constructive and no proof of the existence of winning tickets

How about the training phase?

This approach completely **ignores training costs**

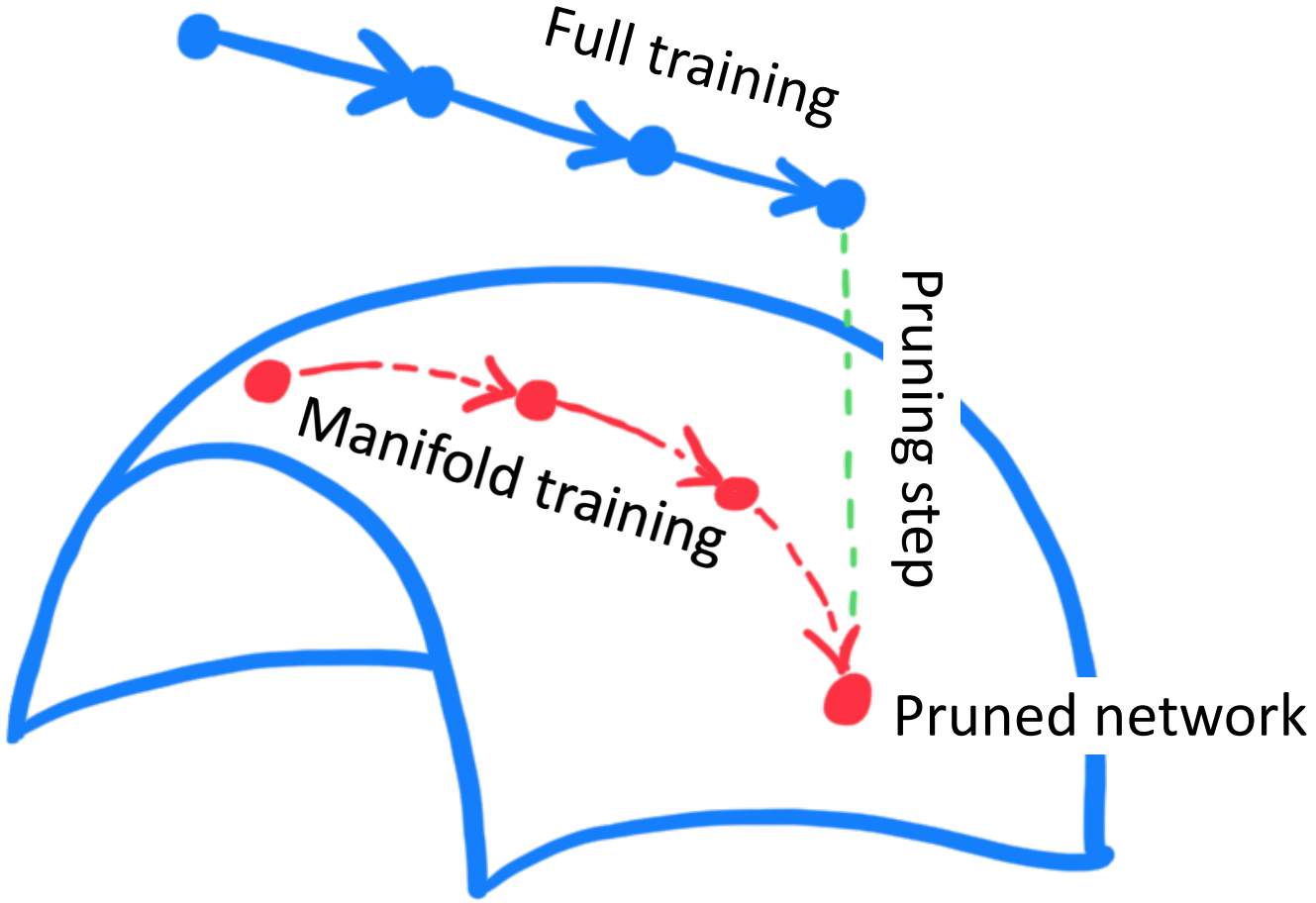


Training costs

Each iteration requires $O(\mathbf{2}\ell N^2)$

Proposed approach

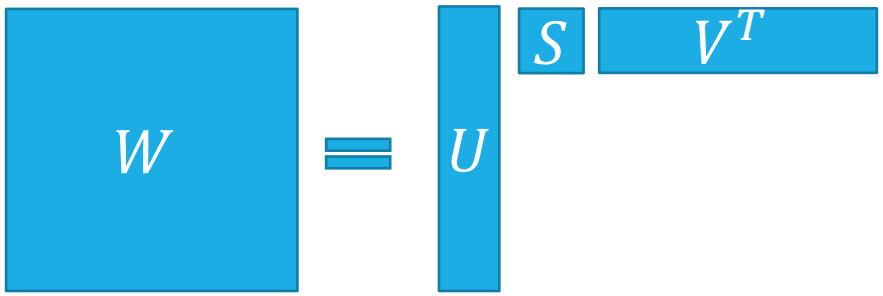
Manifold training



Constraints space = smooth manifold $\mathcal{M}$

Low–rank matrix manifold

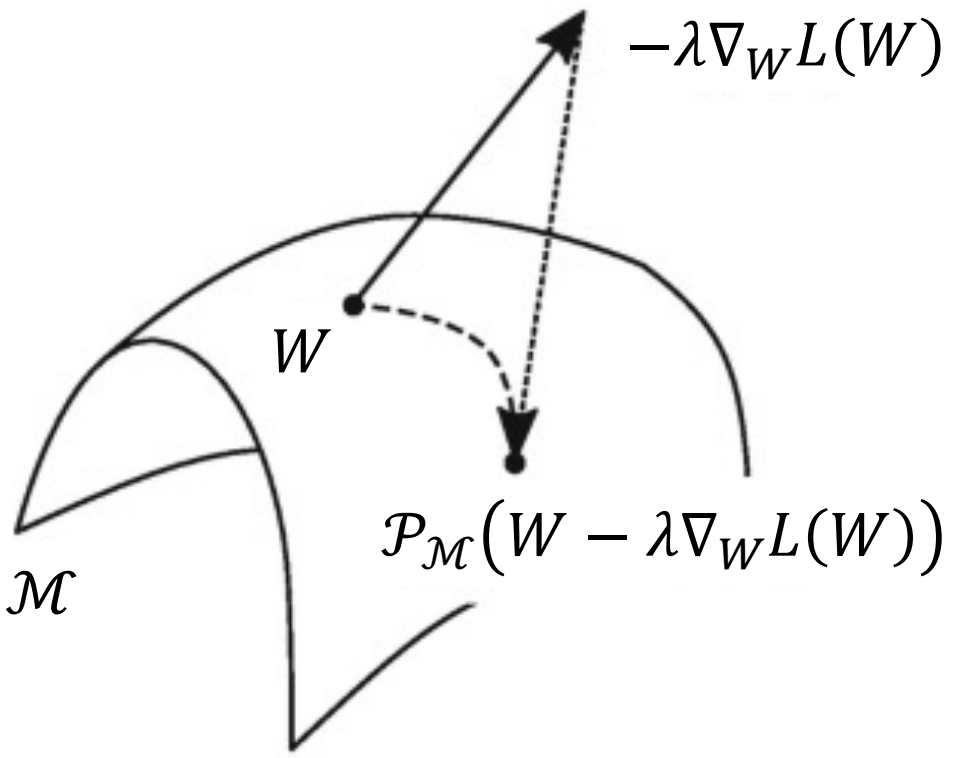
- $\mathcal{M} = \{W : \mathrm{rank}(W) = r \ll N\}$
- $W \in \mathcal{M}$ can be written as:



Memory requirement $O(2Nr)$

Operation cost $O(2Nr)$

Manifold SGD



1. Computing $\mathcal{P}_{\mathcal{M}}\left(W-\lambda\nabla_W L(W)\right)$ can be expensive
2. How can we find "the right $\mathcal{M}$" ?

DLRT algorithm's key ingredients:
– efficient
– adaptive

1. Compute $\mathcal{P}_{\mathcal{M}}\big(W - \lambda \nabla_W L(W)\big)$ using only the three factors U, S, V
   - $U \leftarrow U - \lambda \nabla_U L(U; S, V, x)$
   - $V \leftarrow V - \lambda \nabla_V L(V; U, S, x)$
   - $S \leftarrow S - \lambda \nabla_S L(S; U, V, x)$

2. Adjust the rank of S to match a desired compression rate

$$\vartheta = \frac{\#\ \text{final}\ \ \text{parameters}}{\#\ \text{initial}\ \ \text{parameters}}$$

Proving the low-rank lottery ticket hypothesis

ϑ = compression rate

$\widetilde{W}^{(n)} = U^{(n)} S^{(n)} V^{(n)}$
comp low-rank weights

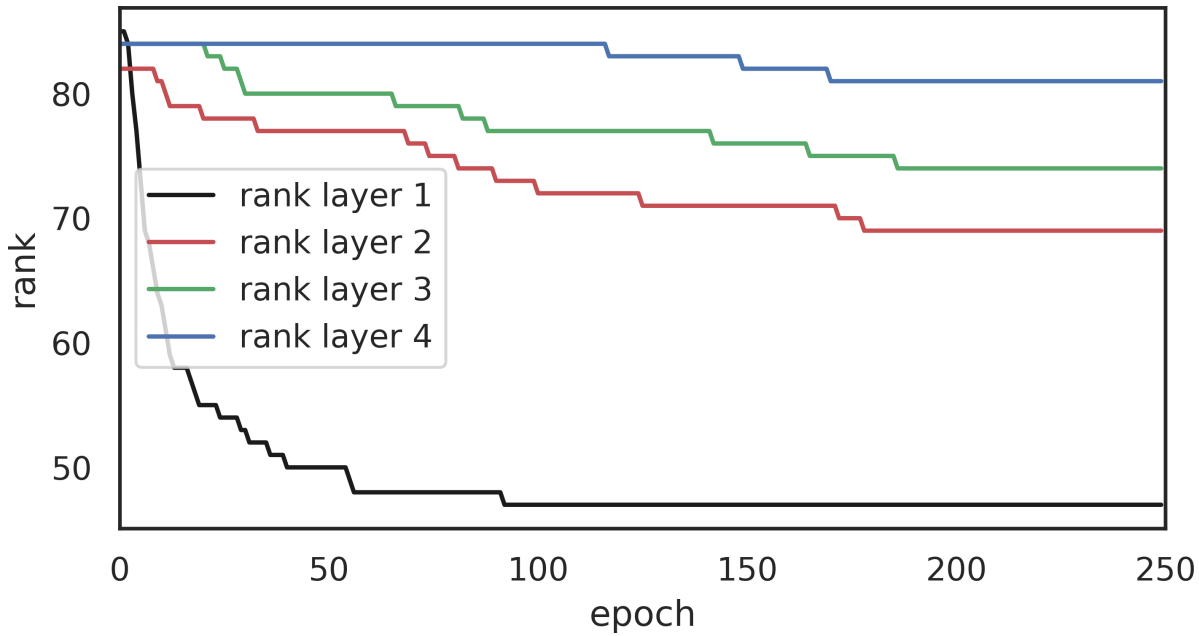
$W^{(n)}$ = ideal full weights

If the loss is smooth enough and its gradient is "ε-far" from the low-rank manifold, then

$$\left| f\left(x; \widetilde{W}^{(n)}\right) - f\left(x; W^{(n)}\right) \right| \leq O(\varepsilon + \vartheta)$$

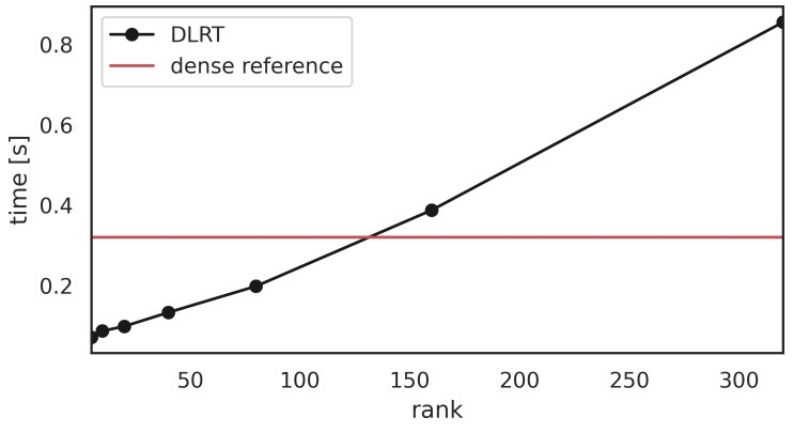
Fully-connected on MNIST

Rank Evolution

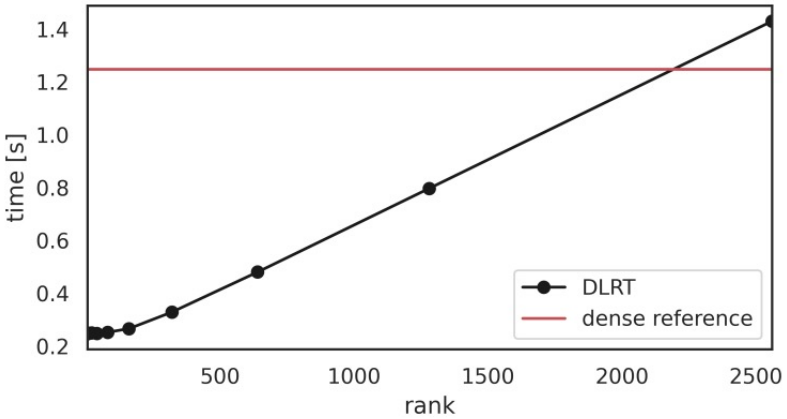


Fully-connected on MNIST

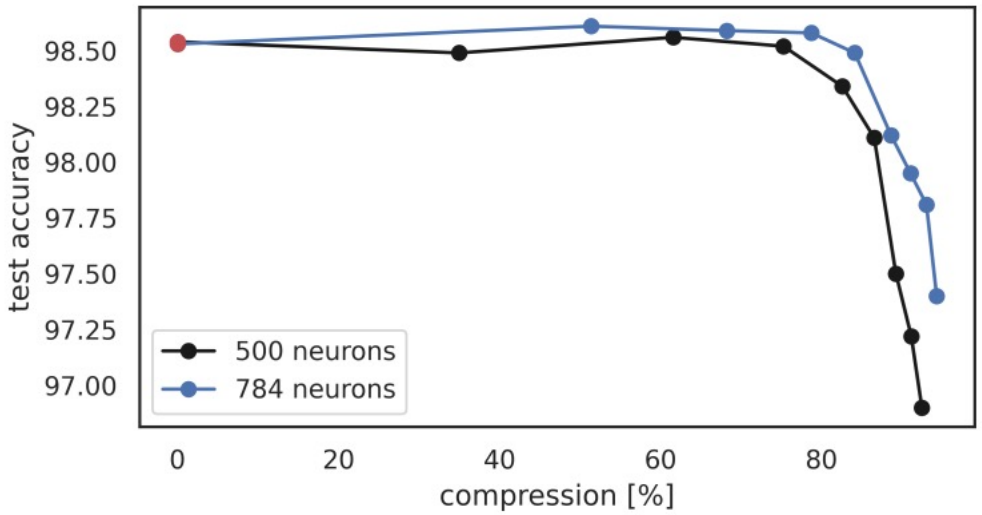
Timings & Accuracy



(a) Training timings



(b) Prediction timings



LeNet5 on MNIST

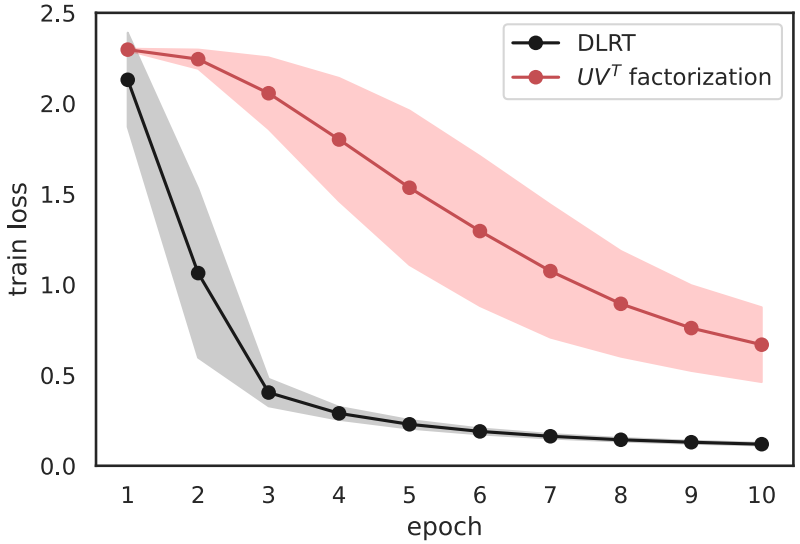
	method	mean test acc.	ranks	params	c.r.	params	c.r.
	LeNet5	$\mathbf{98.8\% \pm 0.06}$	$[20, 50, 500, 10]$	430500	0%	861000	0%
DLRT	$\vartheta = 0.09$	$98.2\% \pm 0.26$	$[10, 23, 62, 10]$	37445	$90.9\% \pm 0.3$	532176	$35.5\% \pm 1.8$
DLRT	$\vartheta = 0.11$	$98.2\% \pm 0.44$	$[10, 20, 48, 10]$	30278	$93.1\% \pm 0.45$	412898	$53.3\% \pm 3.5$
DLRT	$\vartheta = 0.13$	$97.9\% \pm 0.49$	$[9, 16, 37, 10]$	24542	$94.3\% \pm 0.17$	316997	$63.2\% \pm 1.1$
DLRT	$\vartheta = 0.15$	$98.1\% \pm 0.33$	$[9, 16, 28, 10]$	20033	$95.4\% \pm 0.23$	251477	$71.4\% \pm 1.83$
DLRT	$\vartheta = 0.2$	$98.1\% \pm 0.34$	$[8, 8, 15, 10]$	13091	$96.9\% \pm 0.16$	135536	$83.4\% \pm 1.21$
DLRT	$\vartheta = 0.3$	$97.5\% \pm 0.48$	$[4, 6, 8, 10]$	9398	$97.9\% \pm 0.08$	80792	$91.2\% \pm 0.59$
DLRT	$\vartheta = 0.4$	$96.0\% \pm 0.94$	$[2, 4, 4, 10]$	7250	$98.3\% \pm 0.06$	47882	$94.4\% \pm 0.3$
DLRT	$\vartheta = 0.45$	$94.1\% \pm 0.49$	$[2, 2, 3, 10]$	6647	$\mathbf{98.4\% \pm 0.07}$	35654	$\mathbf{95.4\% \pm 0.4}$
	(SSL)(ft)	99.18%		110000	74.4%		$< 0\%$
	(NISP) (ft)	99.0%		100000	76.5%		$< 0\%$
	(GAL)	98.97%		30000	93.0%		$< 0\%$
	(LRNN)	98.67%	$[3, 3, 9, 9]$	18075	95.8%		$< 0\%$
	(SVD prune)	94.0%	$[2, 5, 89, 10]$	123646	71.2%		$< 0\%$

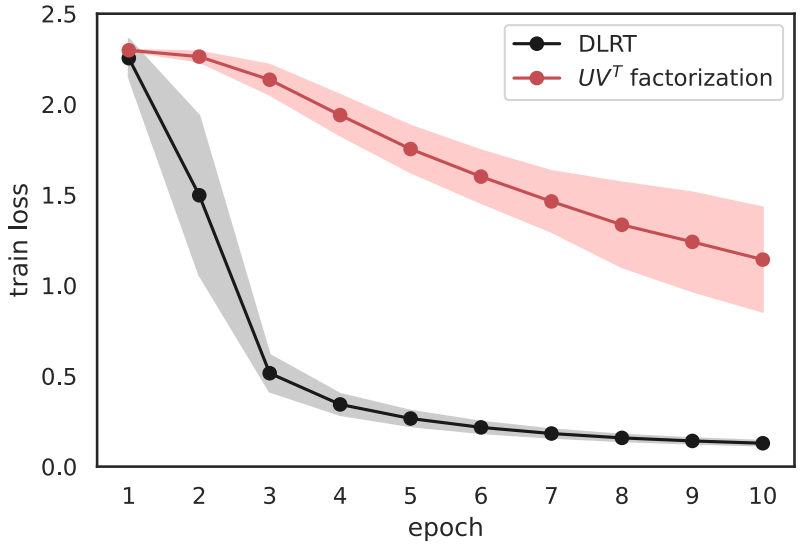
Imagenet1K – 1million images & 1000 classes

Model	test acc. [%]	c.r. eval [%]	c.r. train [%]
ResNet-50	-0.56	54.1	14.2
VGG16	-2.19	86	78.4

Test accuracy w.r.t. full-rank baseline for $\vartheta = 0.1$

The bound does not depend on the singular values





Open Review

Low-rank lottery tickets @ NeurIPS 2022

Thank you!