Dense for the Price of Sparse: Training Extremely Sparse Networks from Scratch with Random Sparse Support

Ilan Price[1,2] & Jared Tanner[1,2]

MLC DLCT Presentation
1 Oct 2021

[1]Mathematical Institute, University of Oxford
[2]The Alan Turing Institute

UNIVERSITY OF OXFORD

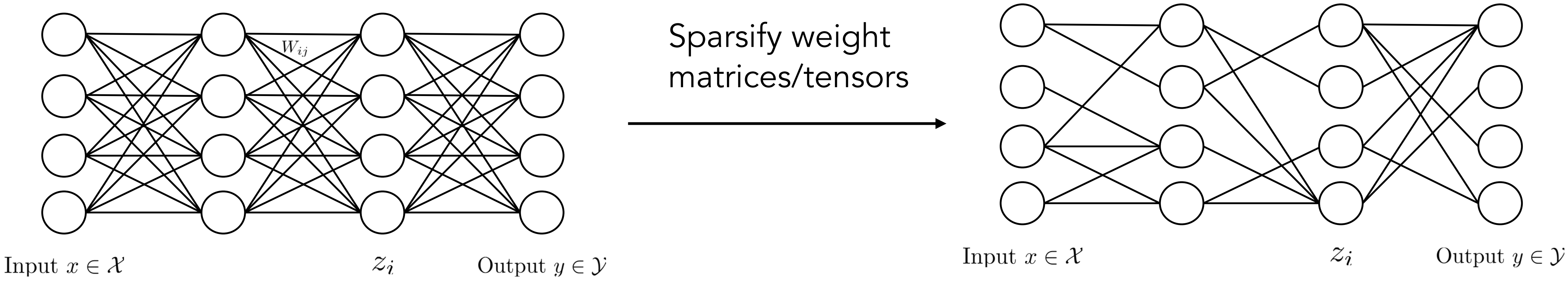
The Alan Turing Institute

Outline

- Context: sparse deep networks and the goal of sparse training
- Sparse training vs (?) subspace training
- DCTpS: computationally efficient random subspace training
- Some results
- Limitations and open questions

Sparse Deep Networks

- 'Sparse' deep learning can refer to multiple different things (See Hoefler et al[1] for a full review).
- Here we consider persistent (fixed for all inputs) sparsity of the weights



UNIVERSITY OF OXFORD

The Alan Turing Institute

[1]Hoefler et al. "Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks" 2021

Sparse Deep Networks

- Deep networks are most-often vastly over parameterised. Parameter counts now range from $\mathcal{O}(10^6)$ to $\mathcal{O}(10^{12})$!

- We have known for a long time that we can "prune" most of these while maintaining good accuracy

Huge storage and computational savings (in theory at least)

UNIVERSITY OF OXFORD

The Alan Turing Institute

Sparse Deep Networks

- Deep networks are most-often vastly over parameterised. Parameter counts now range from $\mathcal{O}(10^6)$ to $\mathcal{O}(10^{12})$!

- We have known for a long time that we can "prune" most of these while maintaining good accuracy

Huge storage and computational savings (in theory at least)

- The most consistently successful methods: pruning during and/or after training (followed by some fine tuning) – e.g. Iterative Magnitude Pruning

Sparse Deep Networks

- Deep networks are most-often vastly over parameterised. Parameter counts now range from $\mathcal{O}(10^6)$ to $\mathcal{O}(10^{12})$!
- We have known for a long time that we can "prune" most of these while maintaining good accuracy

Huge storage and computational savings (in theory at least)

- The most consistently successful methods: pruning during and/or after training (followed by some fine tuning) – e.g. Iterative Magnitude Pruning
- Can we prune before training? So storage and compute is cheaper during training too?

Static sparse training[1]

- Naïve (uniform) random sparsification performs poorly at extreme sparsities

1. We will touch on Dynamic Sparse Training at the end

Static sparse training[1]

- Naïve (uniform) random sparsification performs poorly at extreme sparsities

- The lottery ticket hypothesis (Frankle and Carbin, 2019):

A randomly-initialized, dense neural network contains a subnetwork that is initialized such that—when trained in isolation—it can match the test accuracy of the original network after training for at most the same number of iterations.

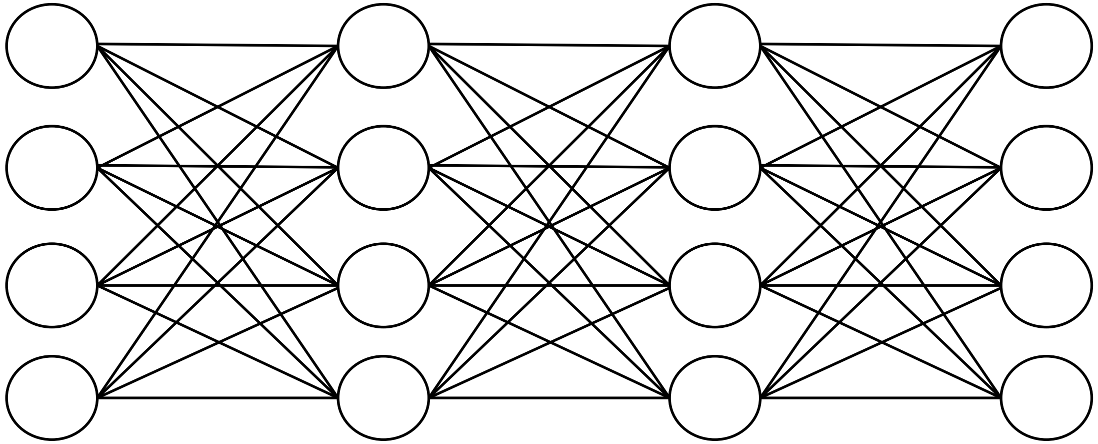
UNIVERSITY OF OXFORD

The Alan Turing Institute

1. We will touch on Dynamic Sparse Training at the end

Static sparse training[1]

- Naïve (uniform) random sparsification performs poorly at extreme sparsities
- The lottery ticket hypothesis [(Frankle and Carbin, 2019)]:



UNIVERSITY OF OXFORD

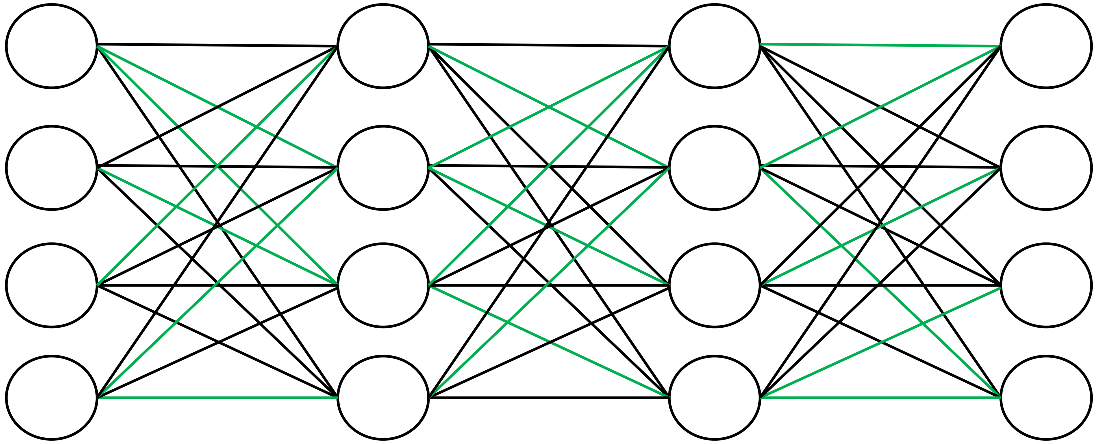
The Alan Turing Institute

[1] We will touch on Dynamic Sparse Training at the end

Static sparse training[1]

- Naïve (uniform) random sparsification performs poorly at extreme sparsities

- The lottery ticket hypothesis (Frankle and Carbin, 2019):

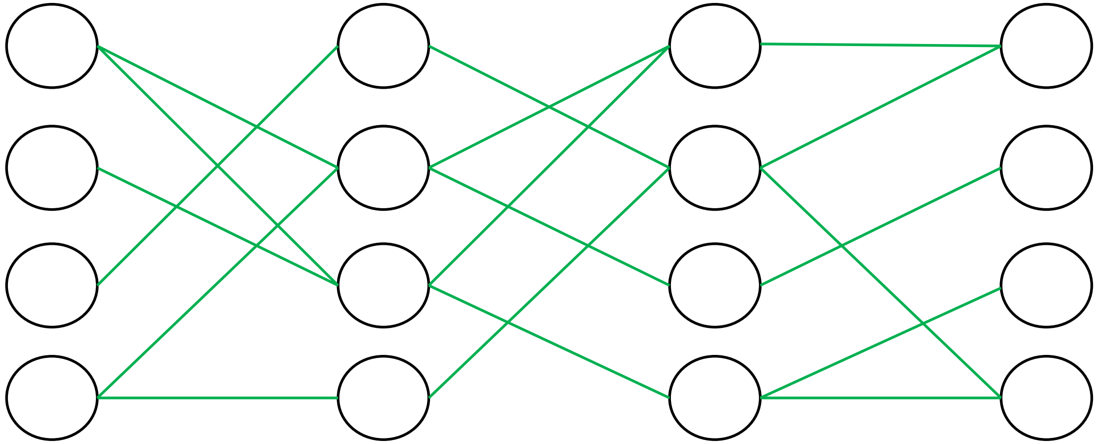


1. We will touch on Dynamic Sparse Training at the end

Static sparse training[1]

- Naïve (uniform) random sparsification performs poorly at extreme sparsities

- The lottery ticket hypothesis [(Frankle and Carbin, 2019)](https://arxiv.org/abs/1803.03635):



UNIVERSITY OF OXFORD

The Alan Turing Institute

1. We will touch on Dynamic Sparse Training at the end

Static sparse training[1]

- Naïve (uniform) random sparsification performs poorly at extreme sparsities

- The lottery ticket hypothesis [(Frankle and Carbin, 2019)]:

A randomly-initialized, dense neural network contains a subnetwork that is initialized such that—when trained in isolation—it can match the test accuracy of the original network after training for at most the same number of iterations.

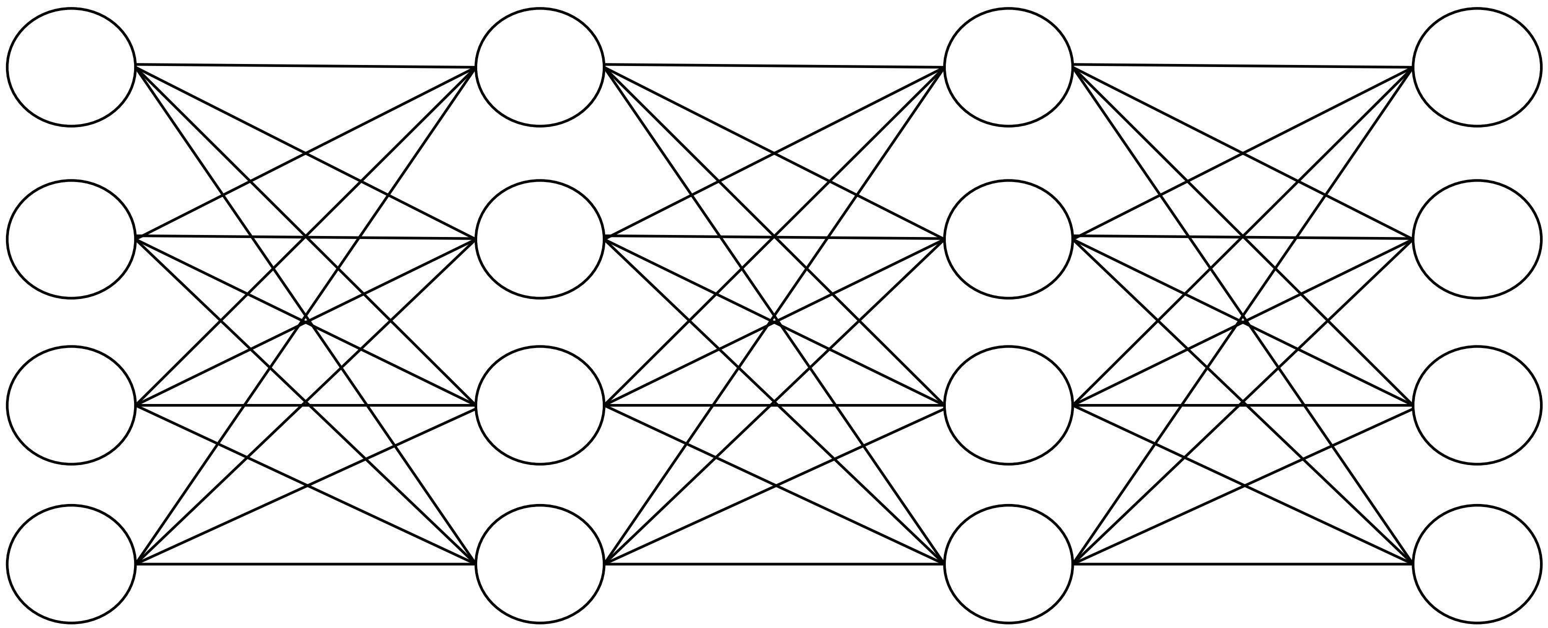
- How to find trainable, extremely sparse sub-networks in practice?

UNIVERSITY OF OXFORD

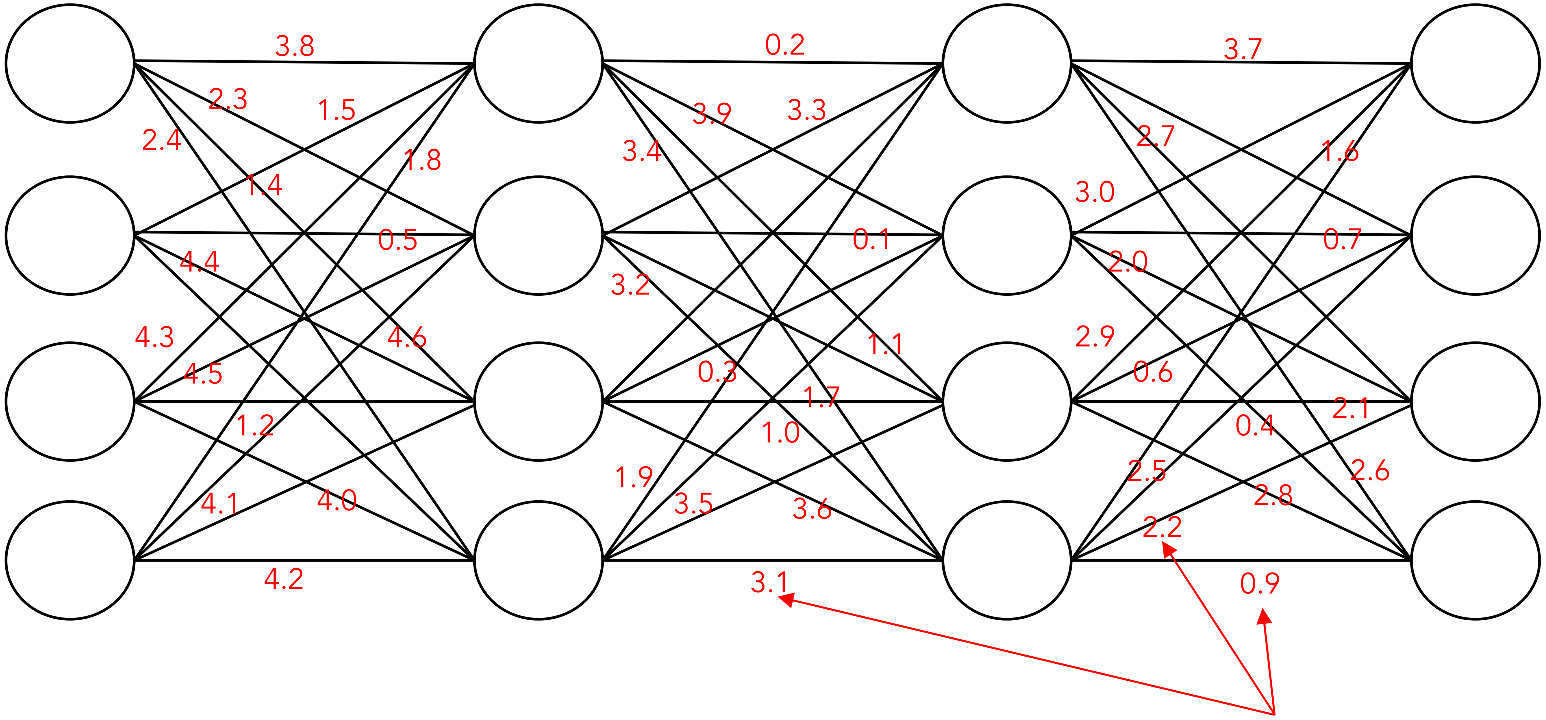
The Alan Turing Institute

1. We will touch on Dynamic Sparse Training at the end

Pruning at initialization (Pal)



Pruning at initialization (PaI)

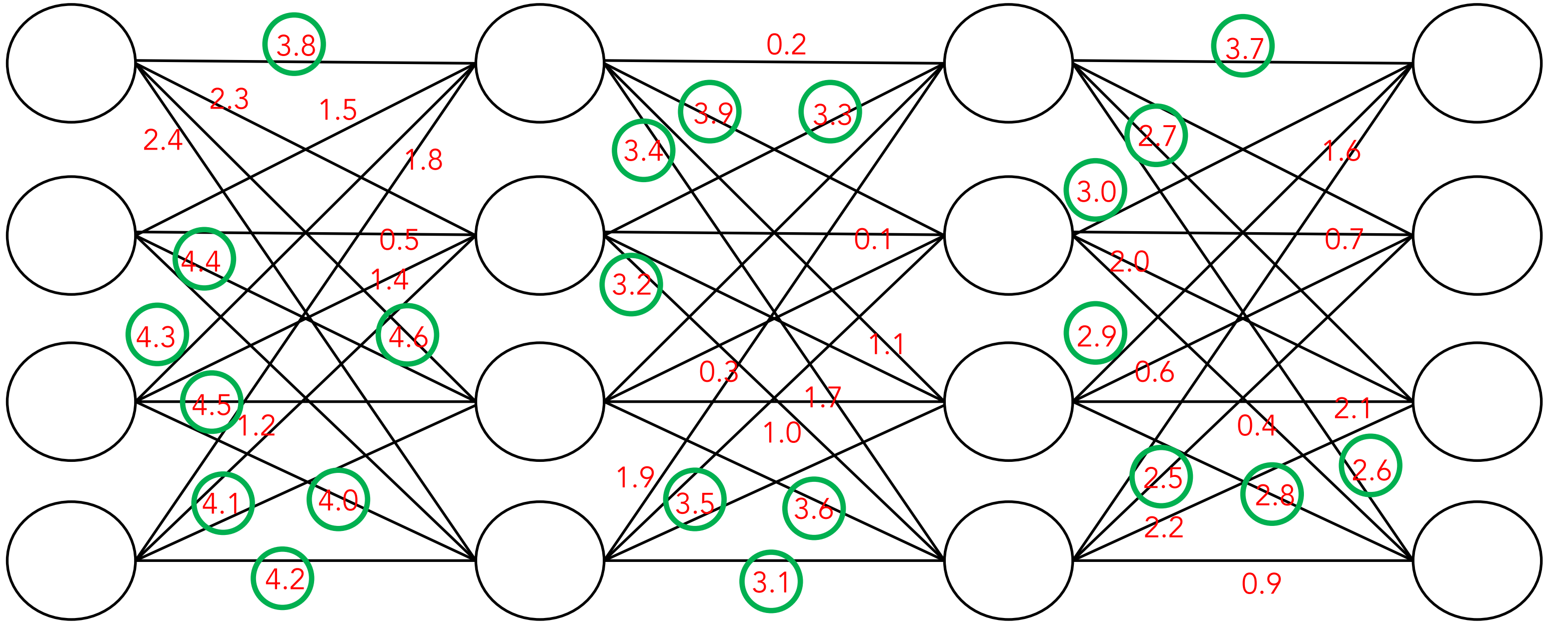


Saliency scores
for each weight

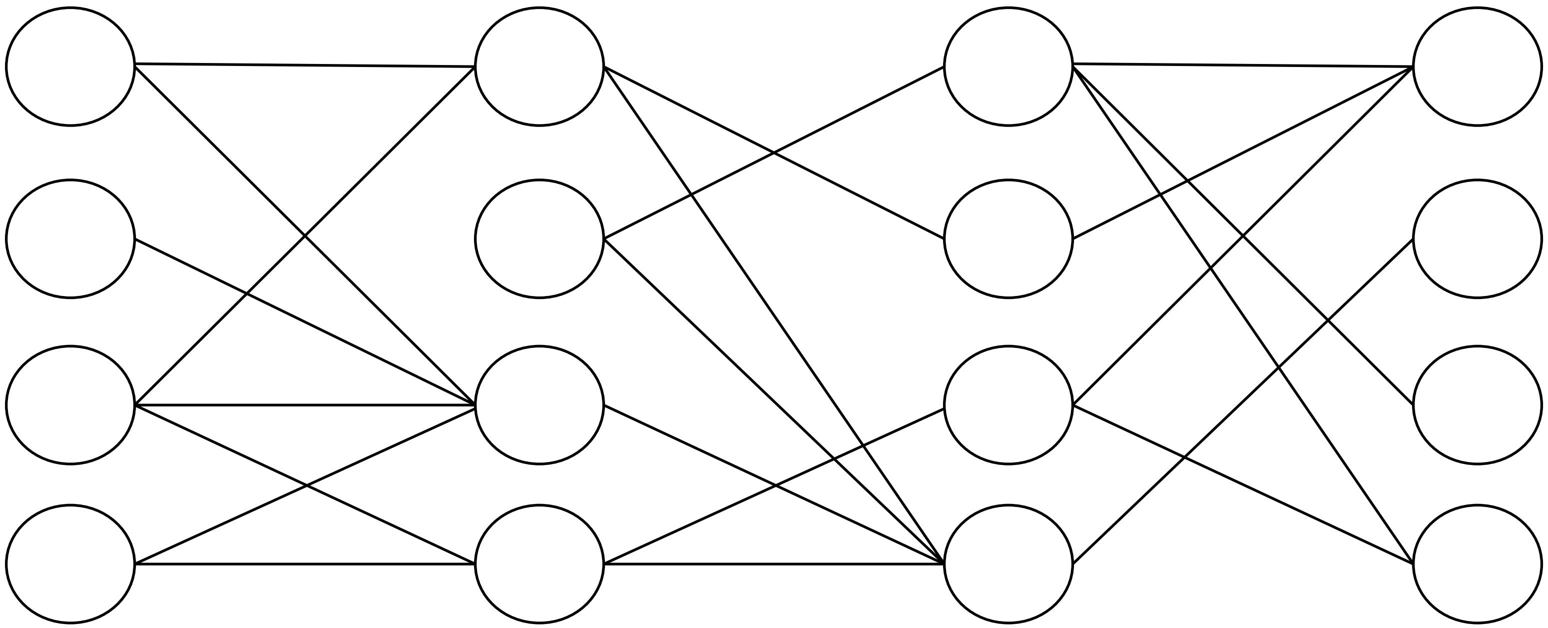
UNIVERSITY OF OXFORD

The Alan Turing Institute

Pruning at initialization (Pal)



Pruning at initialization (Pal)



Standard Pruning At Initialisation (PaI)

Generic steps:

1. Initialize a dense network
2. Define scalar objective $\mathcal{R}$
3. Calculate vector of saliency scores $G(\mathbf{w}) = \frac{\partial \mathcal{R}}{\partial \mathbf{w}} \odot \mathbf{w}$
4. Prune parameters with lowest scores

SOTA at Extreme Sparsities →

FORCE[1]: $G(\mathbf{w}) = \left| \frac{\partial \mathcal{L}(\bar{\mathbf{w}})}{\partial \mathbf{w}} \odot \mathbf{w} \right|$

$\bar{\mathbf{w}}$ is the param vector post-pruning

SynFlow[2]: $\mathcal{R} = \mathbf{1}^\top \left(\prod_{l=1}^{L} |\mathbf{w}^{[l]}| \right) \mathbf{1}$

$|\mathbf{w}^{[l]}|$ is the element-wise absolute value of the parameters in the l^{th} layer

UNIVERSITY OF OXFORD

The Alan Turing Institute

[1] de Jorge, Pau, et al. "Progressive skeletonization: Trimming more fat from a network at initialization." 2020
[2] Tanaka, Hidenori, et al. "Pruning neural networks without any data by iteratively conserving synaptic flow.", 2020.

Pruning at initialization (PaI)

How important are these saliency scores at initialization?

1. After PaI we can often reshuffle the locations of the weights within layers and still train to the same accuracy[2]

[2] Frankle et al. *Pruning Neural Networks at Initialization: Why Are We Missing the Mark?* ICLR2021

Pruning at initialization (PaI)

How important are these saliency scores at initialization?

1. After PaI we can often reshuffle the locations of the weights within layers and still train to the same accuracy[2]

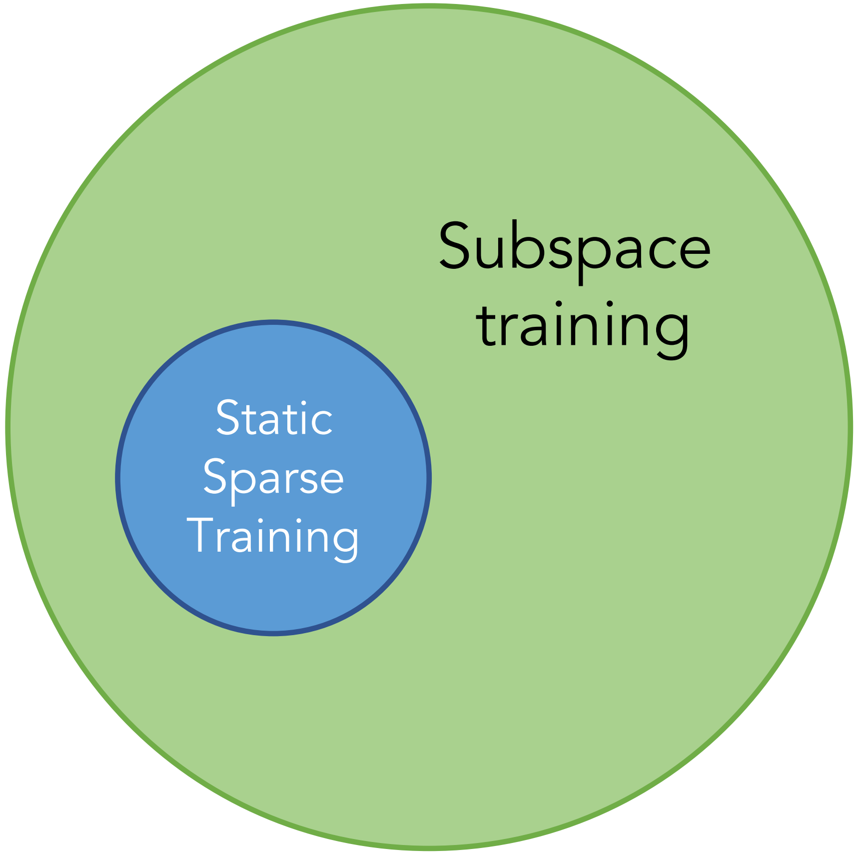
2. Results from other literature on low effective dimensionality in network training

UNIVERSITY OF OXFORD

The Alan Turing Institute

[2] Frankle et al. *Pruning Neural Networks at Initialization: Why Are We Missing the Mark?* ICLR2021

Zooming out again…



Subspace Training

Fixed (untrainable) offset

Network weights

$$\mathbf{w} = \mathbf{d} + U\theta$$

Trainable parameters

Fixed Subspace embedding

$$\mathbf{w}, \mathbf{d} \in \mathbb{R}^N$$

$$\theta \in \mathbb{R}^k, \;\; k << N$$

$$U \in \mathbb{R}^{N \times k}$$

UNIVERSITY OF OXFORD

The Alan Turing Institute

Subspace Training

Sparse Networks:

zero vector

↓

$$\mathbf{w} = \mathbf{d} + U\theta$$

↑

"k-sparse disjoint":

- 1 non-zero per column
- $\leq$ 1 non-zero per row

Pal → Finding the right such U

Subspace Training

(Dense) low-dimensional networks[2] :

Randomly sampled

$$\mathbf{w} = \mathbf{d} + U\theta$$

$U \sim$ e.g. Gaussian

$\mathbf{d} \sim$ standard NN init

Excellent performance with extremely few trainable parameters… but still dense

[2] Li, Chunyuan, et al. "Measuring the Intrinsic Dimension of Objective Landscapes." *International Conference on Learning Representations*. 2018

Interlude: a "Trend"?

- *A Generalized Lottery Ticket Hypothesis*, Alabdulmohsin, Tolstikhin et al. 2021

"We introduce a generalization to the lottery ticket hypothesis in which the notion of "sparsity" is relaxed by choosing an arbitrary basis in the space of parameters."

- *How many degrees of freedom do we need to train deep networks: a loss landscape perspective*, Larson et al, 2021

"recent works, spanning pruning, lottery tickets, and training within random subspaces, have shown that deep neural networks can be trained using far fewer degrees of freedom than the total number of parameters"

Where were we…

Accuracy: random subspace >> “random pruning” subspace

Compute/Storage: random subspace << “random pruning” subspace

Try and get the best of both worlds: efficiency of sparse nets with random subspace selection.

UNIVERSITY OF OXFORD

The Alan Turing Institute

Where were we…

Accuracy: random subspace >> “random pruning” subspace

Compute/Storage: random subspace << “random pruning” subspace

Try and get the best of both worlds: efficiency of sparse nets with random subspace selection.

Important features:

1. Random subspace training → Offset from the origin
2. Random subspace training & pruning at init → “Layer-wise distribution” of trainable parameters.

UNIVERSITY OF OXFORD

The Alan Turing Institute

Best of both: Dense for the Price of Sparse

$$\mathbf{w} = \mathbf{d} + U\theta$$

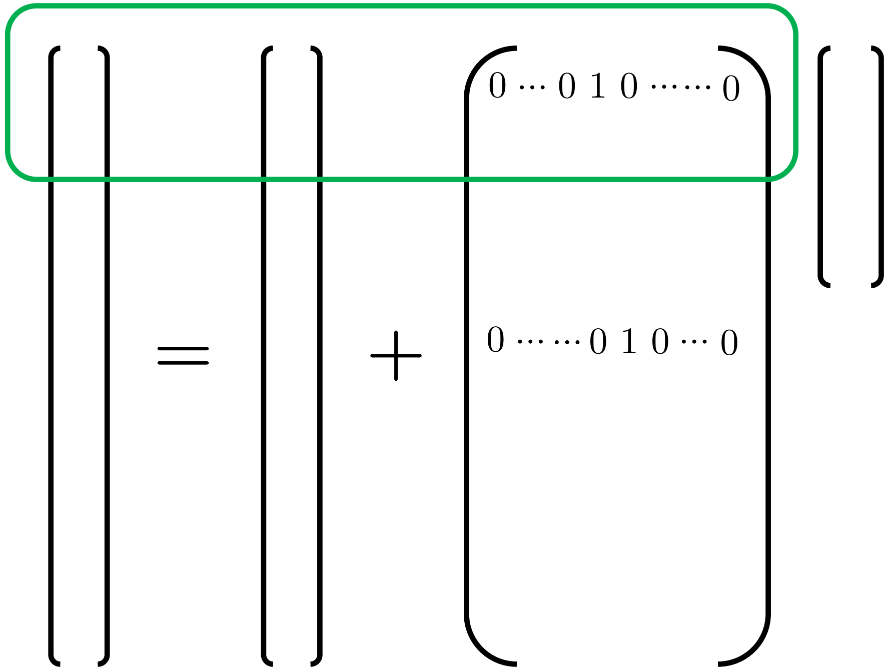
Dense but fixed (untrainable)

"k-sparse disjoint":

- 1 non-zero per column
- $\leq$ 1 non-zero per row

DCT plus Sparse

$$\mathbf{w} = \mathbf{d} + U\theta$$



Each weight matrix W:

$W = D + S$ (D dense, S sparse)

Setting D to be the discrete-cosine-transform matrix, then

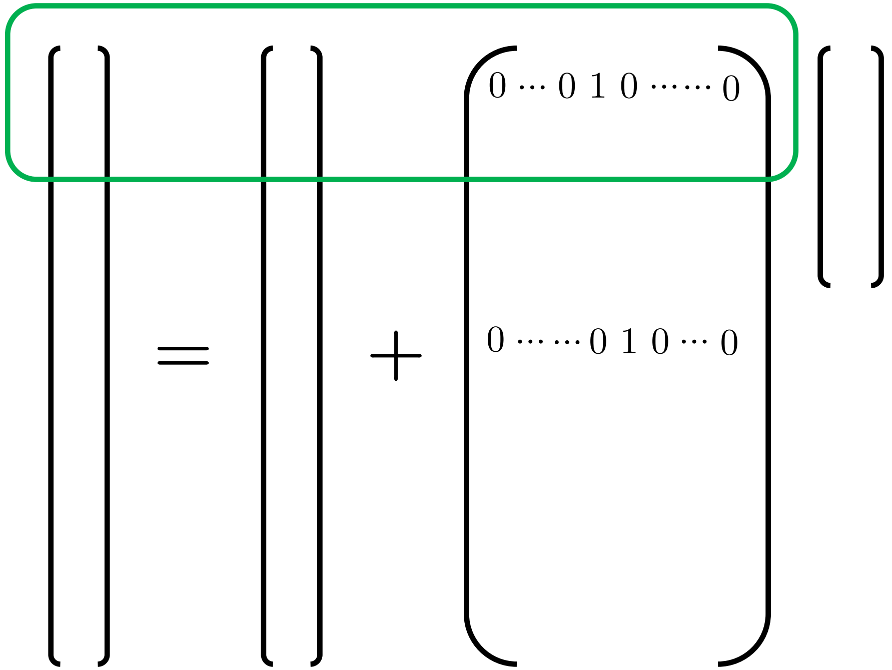
$$\Rightarrow Wx = \mathrm{DCT}(x) + Sx$$

UNIVERSITY OF OXFORD

The Alan Turing Institute

DCT plus Sparse

$$\mathbf{w} = \mathbf{d} + U\theta$$



Each weight matrix W:

$W = D + S$ (D dense, S sparse)

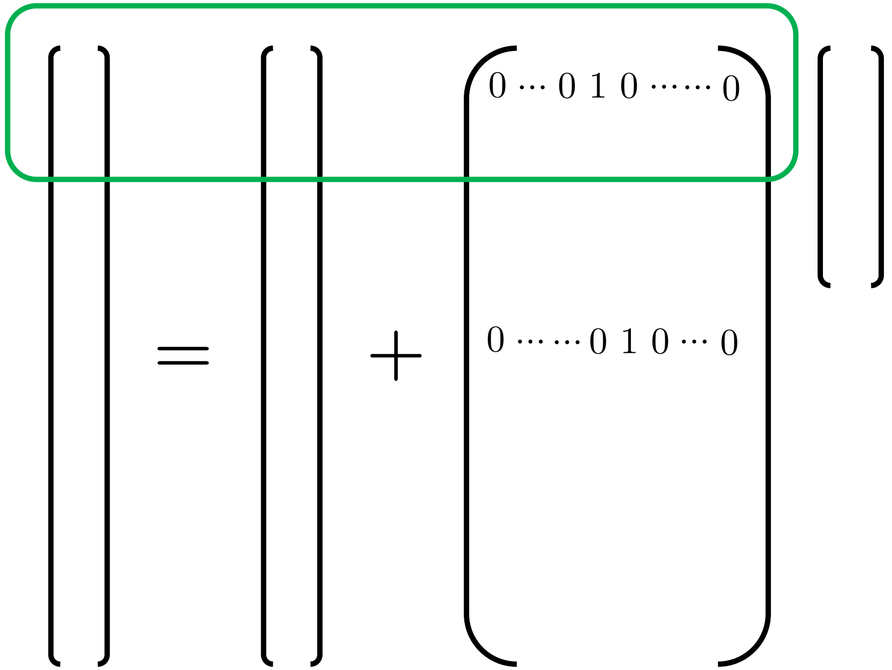
Setting D to be the discrete-cosine-transform matrix, then

$$\Longrightarrow Wx = \mathrm{DCT}(x) + Sx$$

- No storage
- $\mathcal{O}(n \log n)$ compute

DCT plus Sparse

$$\mathbf{w} = \mathbf{d} + U\theta$$



Each weight matrix W:

$W = D + S$ (D dense, S sparse)

Setting D to be the discrete-cosine-transform matrix, then

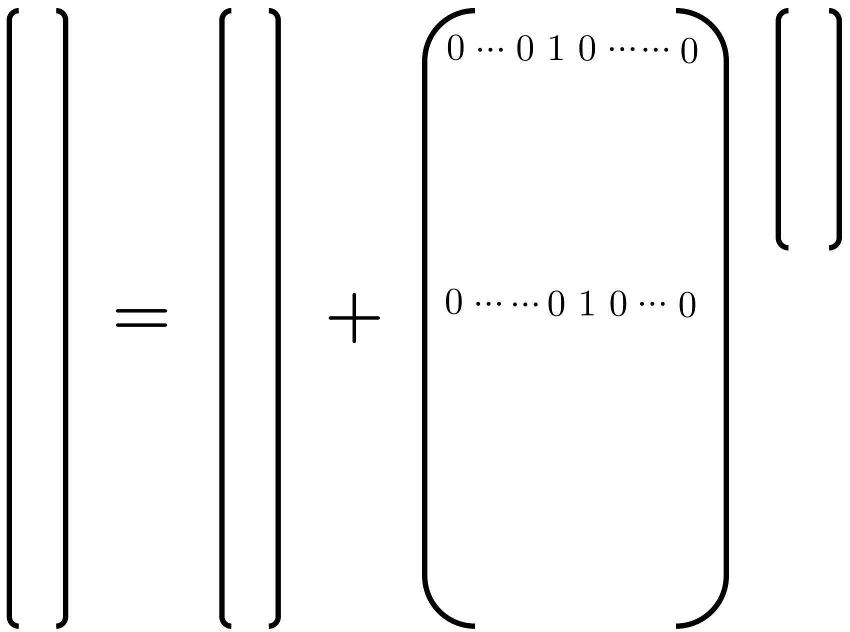
$$\Rightarrow Wx = \mathrm{DCT}(x) + Sx$$

UNIVERSITY OF OXFORD

The Alan Turing Institute

DCT plus Sparse

$$\mathbf{w} = \mathbf{d} + U\theta$$



Each weight matrix W:

$W = D + S$ (D dense, S sparse)

Setting D to be the discrete-cosine-transform matrix, then

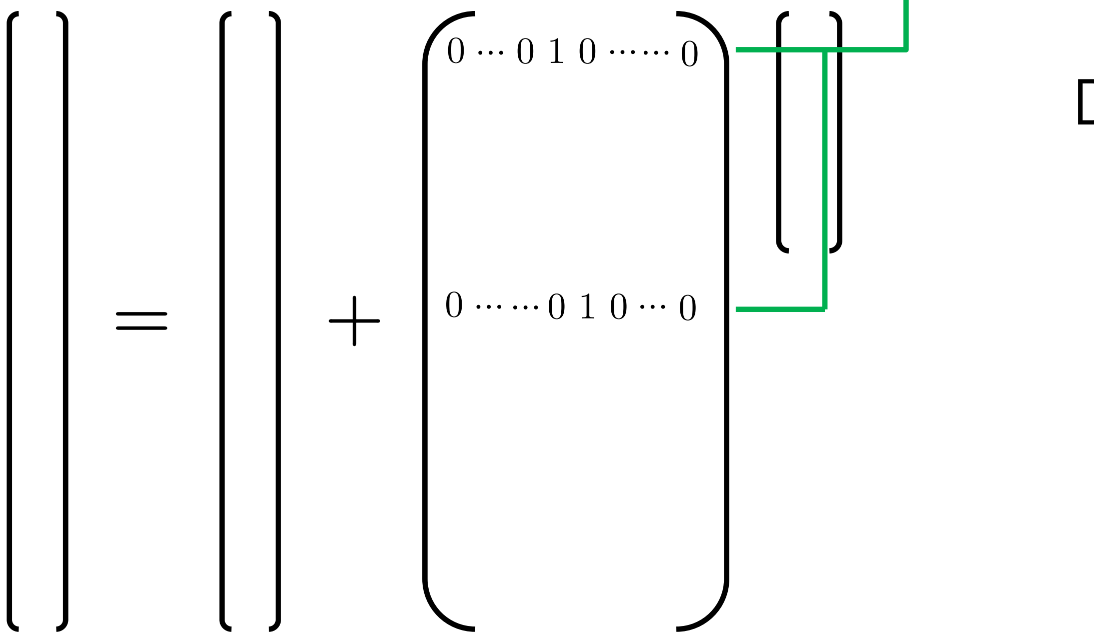
$$\Rightarrow Wx = \mathrm{DCT}(x) + Sx$$

(We also add single trainable scaling param for DCT:)

$$\Rightarrow Wx = \alpha \mathrm{DCT}(x) + Sx$$

DCT plus Sparse

$$\mathbf{w} = \mathbf{d} + U\theta$$



Distribution of 1-sparse rows

$\longleftrightarrow$

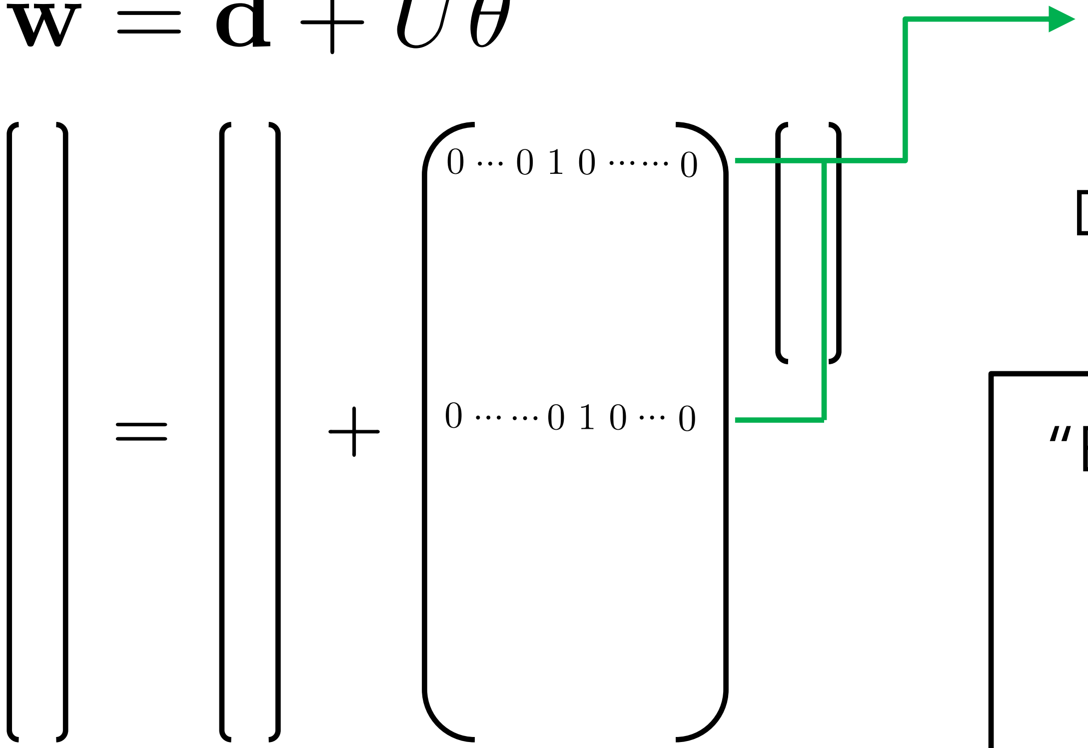
Distribution of trainable parameters across and within S matrices

UNIVERSITY OF OXFORD

The Alan Turing Institute

DCT plus Sparse

$$\mathbf{w} = \mathbf{d} + U\theta$$



Distribution of 1-sparse rows

$\longleftrightarrow$

Distribution of trainable parameters across and within S matrices

"Equal per layer" (EPL):

- $\frac{k}{L}$ trainable params in each S
- Locations in S uniformly random
- No initialization of the dense net

UNIVERSITY OF OXFORD

The Alan Turing Institute

DCT plus Sparse

	Δ Accuracy			Computational Cost			Network Size on Device		
$P =$	0.01	0.001	0.0001	At init.	Training	Final	At init.	Training	Final
Random	-11.9%	-66%	-66%	0	$\mathcal{O}(pmn)$	$\mathcal{O}(pmn)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$
IMP	+0.8%	-7.1%	-64.8%	0	$\mathcal{O}(mn) \xrightarrow{t} \mathcal{O}(pmn)$	$\mathcal{O}(pmn)$	$\mathcal{O}(N)$	$\mathcal{O}(N) \xrightarrow{t} \mathcal{O}(PN)$	$\mathcal{O}(PN)$
FORCE	-6.6%	-26.9%	-62.4%	$\mathcal{O}(mnk)$	$\mathcal{O}(pmn)$	$\mathcal{O}(pmn)$	$\mathcal{O}(N)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$
SynFlow	-6.2%	-31.6%	-60.4%	$\mathcal{O}(mnk)$	$\mathcal{O}(pmn)$	$\mathcal{O}(pmn)$	$\mathcal{O}(N)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$
RigL (ERK)	+0.4%	-16.8%	-65.7%	0	$\mathcal{O}(pmn + \frac{1}{\Delta T} mn)$	$\mathcal{O}(pmn)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$
DCTpS	-5.8%	-15%	-22.8%	0	$\mathcal{O}(q \log q + pmn)$	$\mathcal{O}(q \log q + pmn)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$

UNIVERSITY OF OXFORD

The Alan Turing Institute

DCT plus Sparse

	Δ Accuracy			Computational Cost			Network Size on Device		
$P =$	0.01	0.001	0.0001	At init.	Training	Final	At init.	Training	Final
Random	-11.9%	-66%	-66%	0	$\mathcal{O}(pmn)$	$\mathcal{O}(pmn)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$
IMP	+0.8%	-7.1%	-64.8%	0	$\mathcal{O}(mn) \xrightarrow{t} \mathcal{O}(pmn)$	$\mathcal{O}(pmn)$	$\mathcal{O}(N)$	$\mathcal{O}(N) \xrightarrow{t} \mathcal{O}(PN)$	$\mathcal{O}(PN)$
FORCE	-6.6%	-26.9%	-62.4%	$\mathcal{O}(mnk)$	$\mathcal{O}(pmn)$	$\mathcal{O}(pmn)$	$\mathcal{O}(N)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$
SynFlow	-6.2%	-31.6%	-60.4%	$\mathcal{O}(mnk)$	$\mathcal{O}(pmn)$	$\mathcal{O}(pmn)$	$\mathcal{O}(N)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$
RigL (ERK)	+0.4%	-16.8%	-65.7%	0	$\mathcal{O}(pmn + \frac{1}{\Delta T} mn)$	$\mathcal{O}(pmn)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$
DCTpS	-5.8%	-15%	-22.8%	0	$\mathcal{O}(q \log q + pmn)$	$\mathcal{O}(q \log q + pmn)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$

$$W \in \mathbb{R}^{m \times n}$$

$$q = \max(m, n)$$

$$p = \frac{\|W\|_0}{mn}$$

DCT plus Sparse

	Δ Accuracy			Computational Cost			Network Size on Device		
$P =$	0.01	0.001	0.0001	At init.	Training	Final	At init.	Training	Final
Random	-11.9%	-66%	-66%	0	$\mathcal{O}(pmn)$	$\mathcal{O}(pmn)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$
IMP	+0.8%	-7.1%	-64.8%	0	$\mathcal{O}(mn) \xrightarrow{t} \mathcal{O}(pmn)$	$\mathcal{O}(pmn)$	$\mathcal{O}(N)$	$\mathcal{O}(N) \xrightarrow{t} \mathcal{O}(PN)$	$\mathcal{O}(PN)$
FORCE	-6.6%	-26.9%	-62.4%	$\mathcal{O}(mnk)$	$\mathcal{O}(pmn)$	$\mathcal{O}(pmn)$	$\mathcal{O}(N)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$
SynFlow	-6.2%	-31.6%	-60.4%	$\mathcal{O}(mnk)$	$\mathcal{O}(pmn)$	$\mathcal{O}(pmn)$	$\mathcal{O}(N)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$
RigL (ERK)	+0.4%	-16.8%	-65.7%	0	$\mathcal{O}(pmn + \frac{1}{\Delta T} mn)$	$\mathcal{O}(pmn)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$
DCTpS	-5.8%	-15%	-22.8%	0	$\mathcal{O}(q \log q + pmn)$	$\mathcal{O}(q \log q + pmn)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$	$\mathcal{O}(PN)$

$P =$ Global density

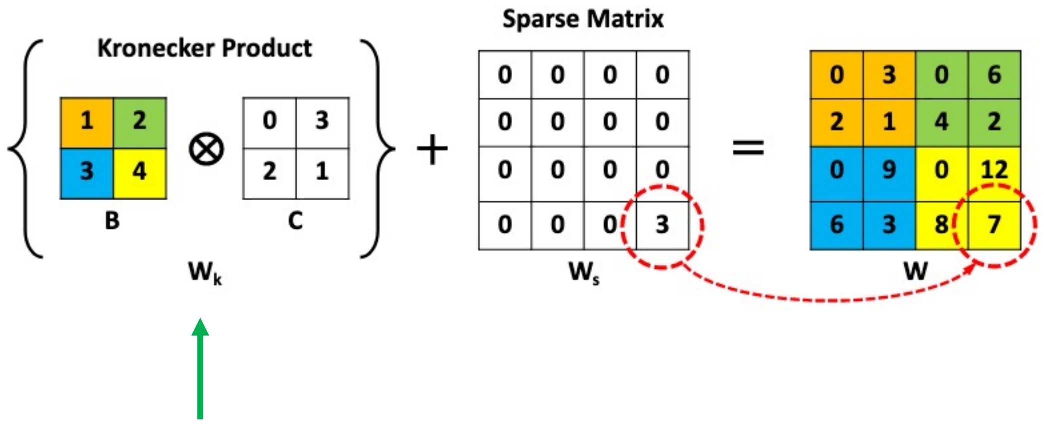
$N =$ Total Network Params

UNIVERSITY OF OXFORD

The Alan Turing Institute

Interlude: a “Trend”?

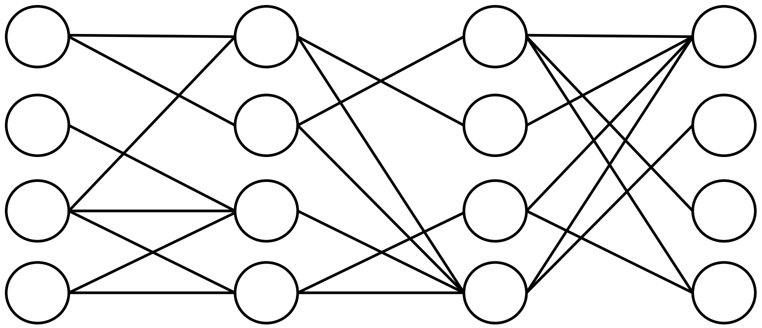
Doping: A technique for efficient compression of LSTM models using sparse structured additive matrices, Thakker et al, 2021



Key difference:
Needs to be
trained and stored

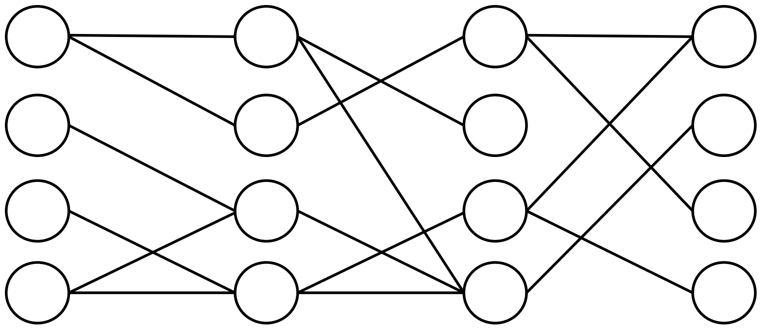
Dynamic Sparse Training

- Static sparse training chooses support set of **w**, then keeps fixed during training.
- DST instead jointly optimises topology *and* weights, subject to fixed sparsity level
- Start sparse, then: Train, Prune, Regrow, Repeat.



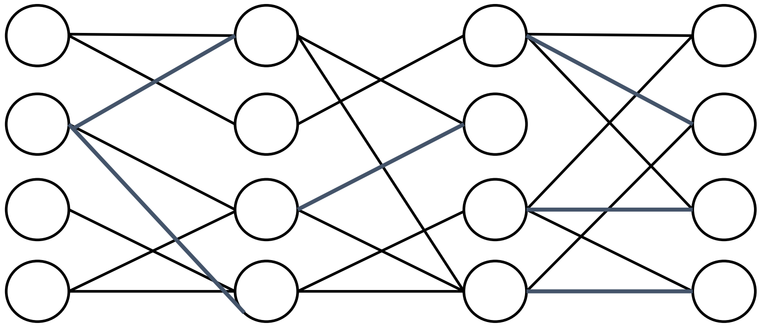
Dynamic Sparse Training

- Static sparse training chooses support set of **w**, then keeps fixed during training.
- DST instead jointly optimises topology *and* weights, subject to fixed sparsity level
- Start sparse, then: Train, Prune, Regrow, Repeat.



Dynamic Sparse Training

- Static sparse training chooses support set of **w**, then keeps fixed during training.
- DST instead jointly optimises topology *and* weights, subject to fixed sparsity level
- Start sparse, then: Train, Prune, Regrow, Repeat.



Combining with Dynamic Sparse training

Straightforwardly combined with DCTpS:

$$Wx = \mathrm{DCT}(x) + Sx$$

Apply DST to the sparse, trainable matrices in each layer

Combining with Dynamic Sparse training

Where are connections initialized?

Which to prune?

Which to regrow?

Where can they regrow?

Combining with Rig-L[1]

Where are connections initialized? → Initialise all W_i as modified Erdos Reyni random bipartite graph, (Details not NB)

Which to prune? → Smallest Magnitude

Which to regrow? → Largest magnitude gradient

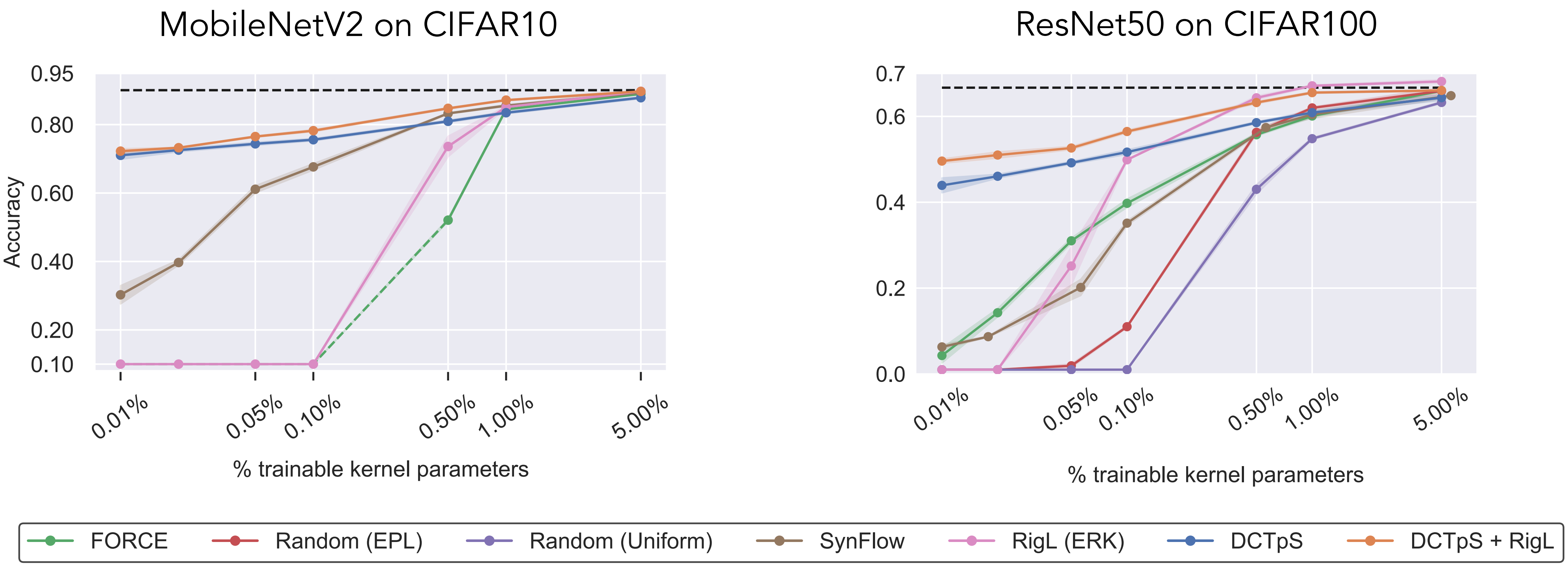
Where can they regrow? → Within the same layer

UNIVERSITY OF OXFORD

The Alan Turing Institute

[1]Evci et al. "Rigging the Lottery: Making All Tickets Winners", 2020

Some Results

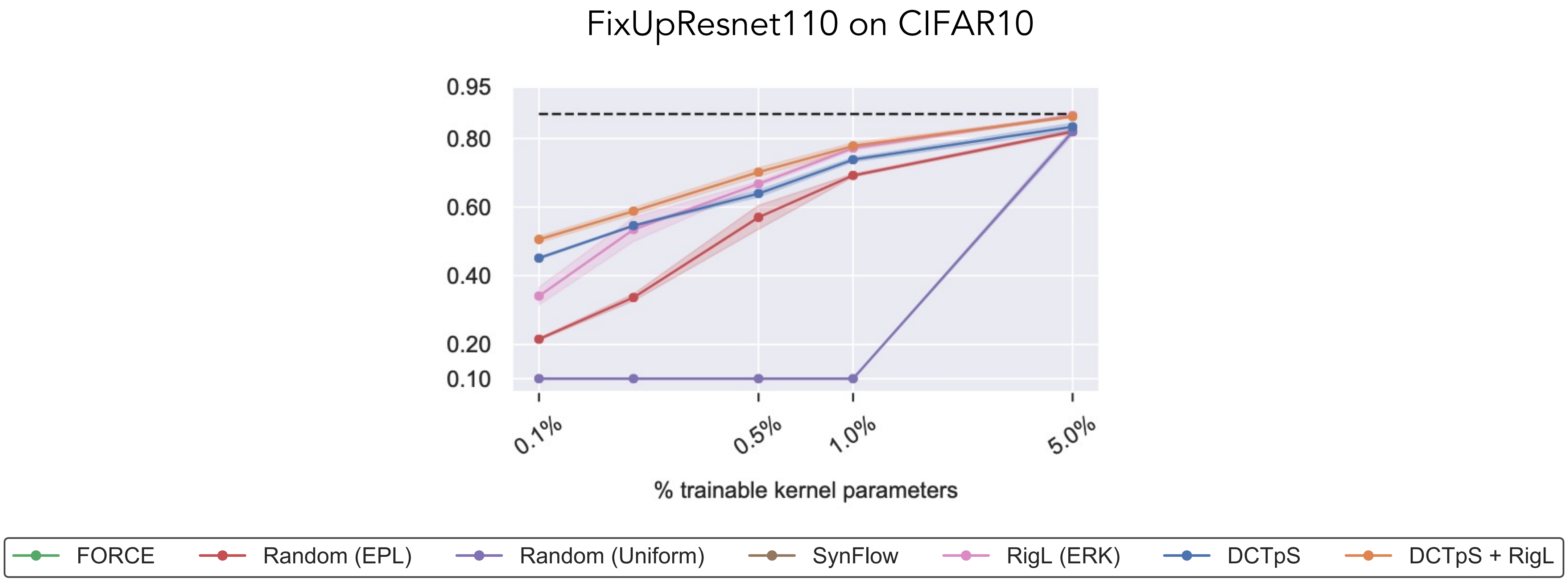


UNIVERSITY OF OXFORD

The Alan Turing Institute

Tanaka et al. "Pruning neural networks without any data by iteratively conserving synaptic flow", 2020
De Jorge et al "Progressive Skeletonization: Trimming more fat from a network at initialization" , 2021
Evci et al. "Rigging the Lottery: Making All Tickets Winners", 2020

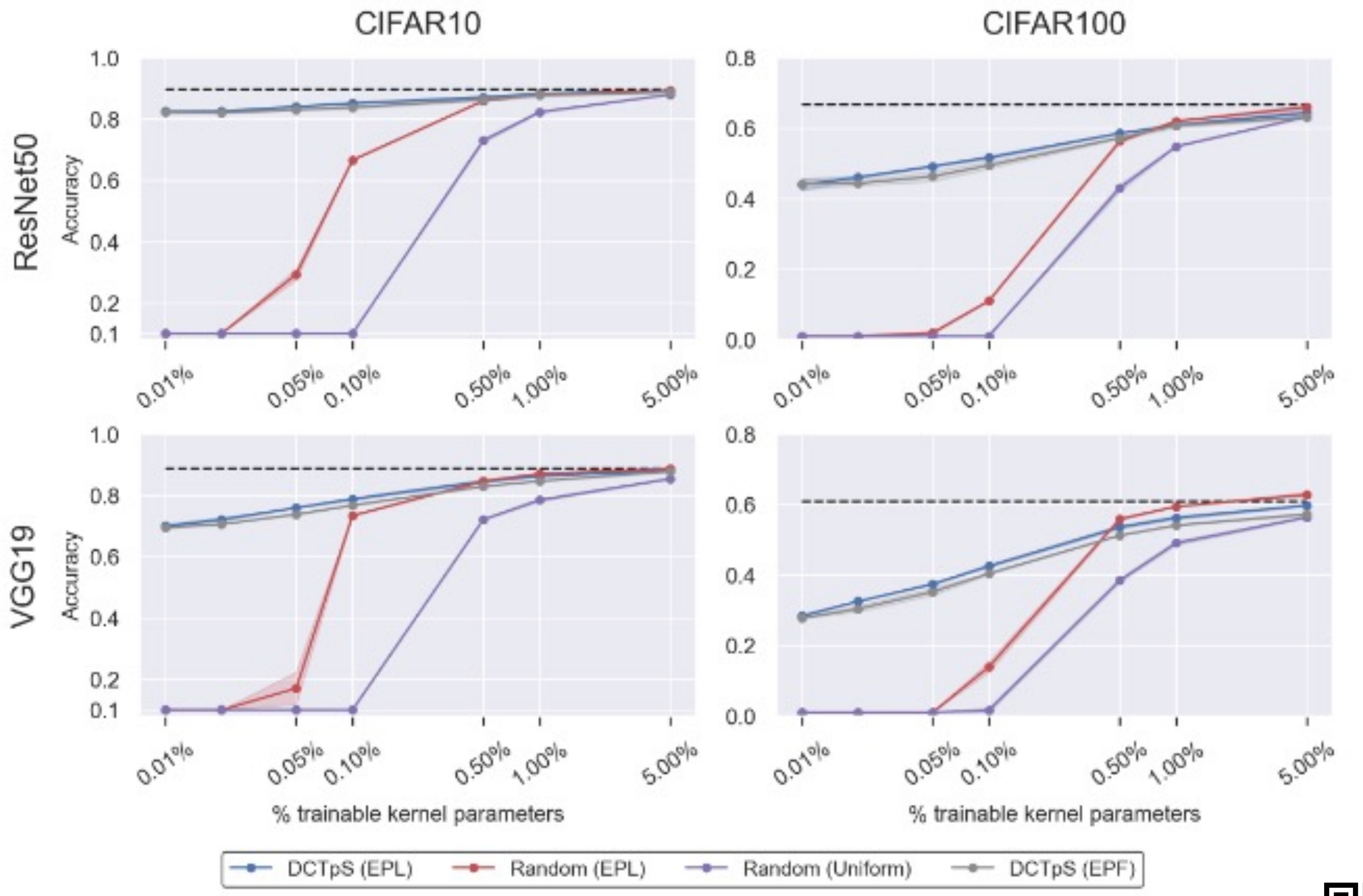
Without Batchnorm



Other support distributions?

"Equal per Filter" (EPF)

(A version of N:M sparsity)



Limitations

- Computational floor imposed by the DCT – can push storage, not compute, down to the extremes. Q: more efficient ways to achieve an appropriate offset?

Limitations

- Computational floor imposed by the DCT – can push storage, not compute, down to the extremes. Q: more efficient ways to achieve an appropriate offset?

- Gains are hardware and implementation dependent – so far these are gains “in theory”

Limitations

- Computational floor imposed by the DCT – can push storage, not compute, down to the extremes. Q: more efficient ways to achieve an appropriate offset?
- Gains are hardware and implementation dependent – so far these are gains "in theory"
- Does not speak to the more general question about how best to use parameters (sparser, larger net vs denser, smaller net, etc)

UNIVERSITY OF OXFORD

The Alan Turing Institute

Limitations

- Computational floor imposed by the DCT – can push storage, not compute, down to the extremes. Q: more efficient ways to achieve an appropriate offset?

- Gains are hardware and implementation dependent – so far these are gains "in theory"

- Does not speak to the more general question about how best to use parameters (sparser, larger net vs denser, smaller net, etc)

- Shrinks the storage footprint of the network – but the hidden representations are not sparse and can be very large (same for all sparse nets)

UNIVERSITY OF OXFORD

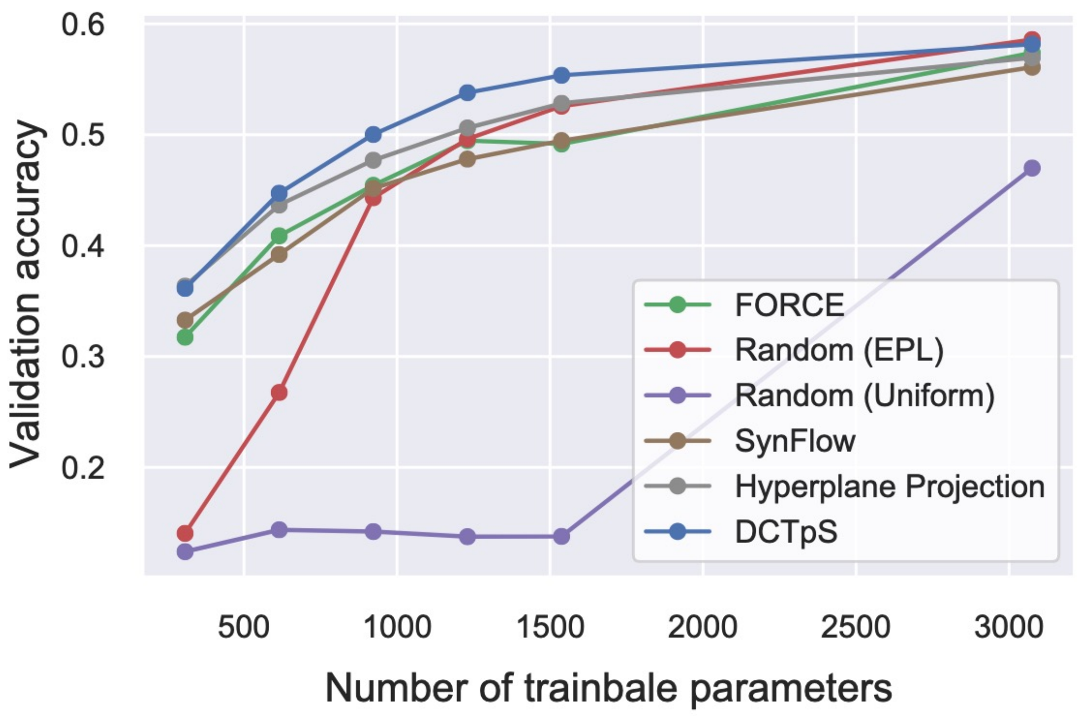
The Alan Turing Institute

Thank you

Reach out on ilan.price@maths.ox.ac.uk or @IlanPrice

Compared with Random Subspace?

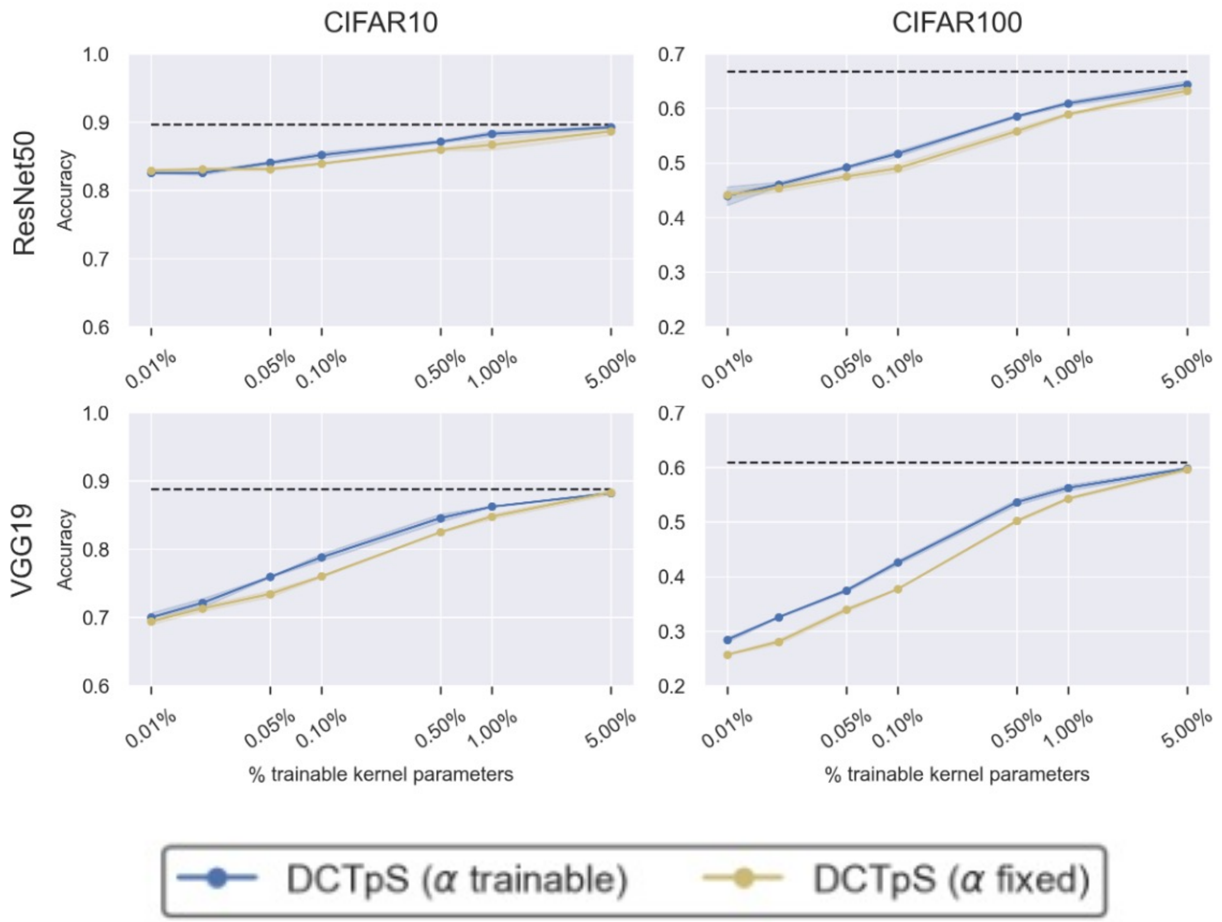
Lenet-5 on CIFAR-10



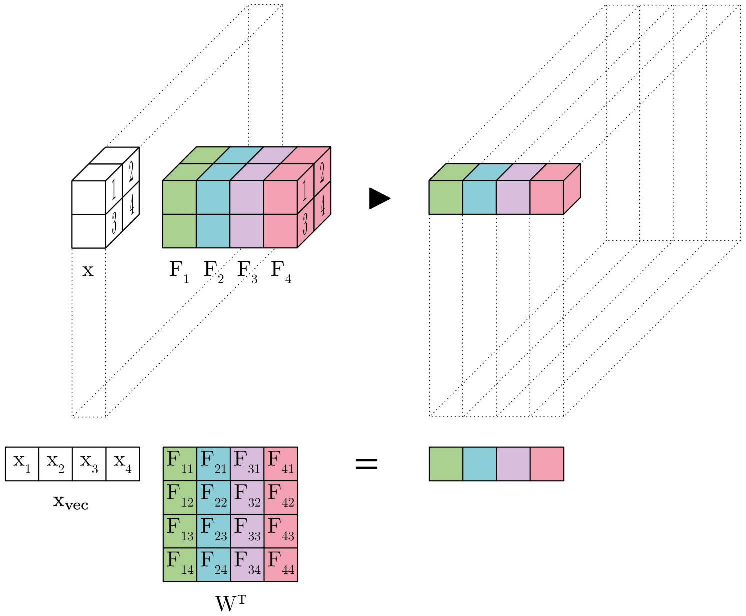
UNIVERSITY OF OXFORD

The Alan Turing Institute

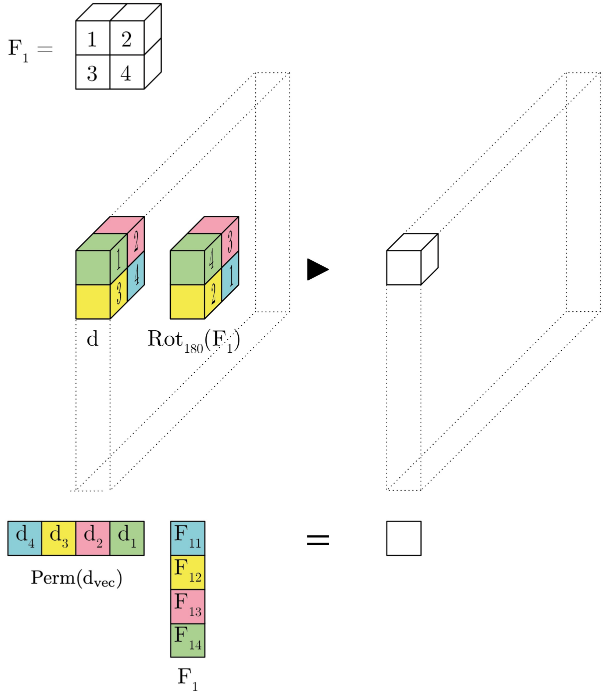
What if we fixed α ?



DCTpS Convolutional Layers



Forward pass

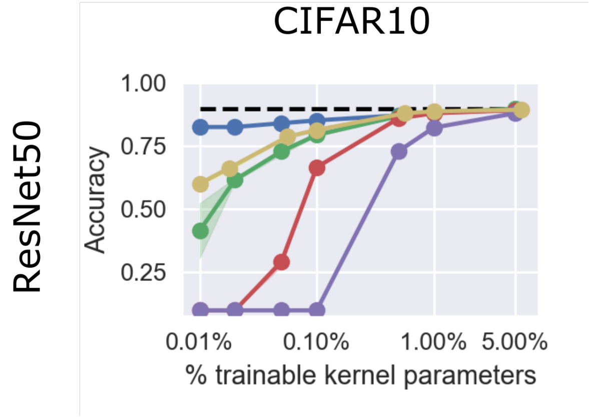


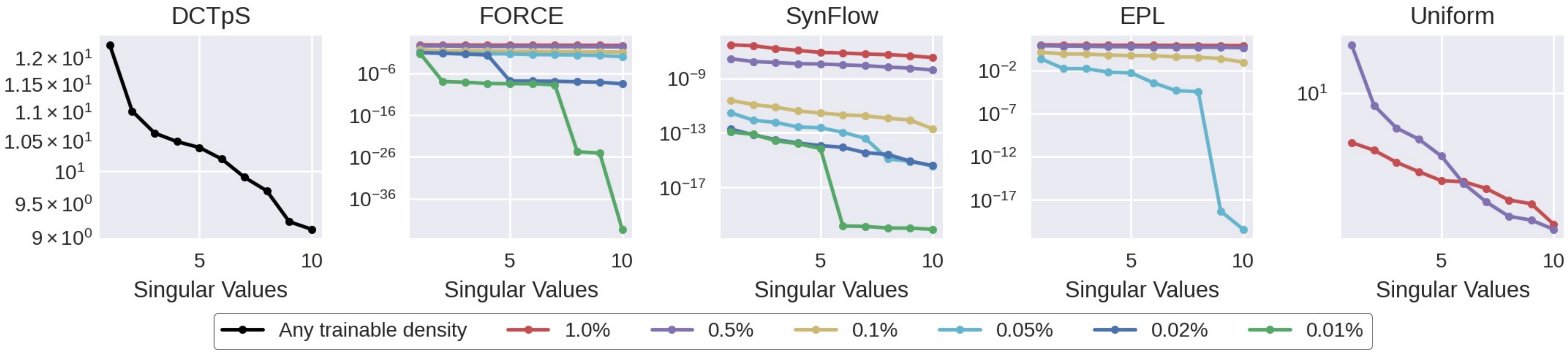
Backprop

UNIVERSITY OF OXFORD

The Alan Turing Institute

When is training possible?





Jacobian Spectra

UNIVERSITY OF OXFORD

The Alan Turing Institute