Sparsity in Deep Learning

Ilan Price

C6.5 Theories of Deep Learning
Mathematical Institute, University of Oxford
2 December, 2021

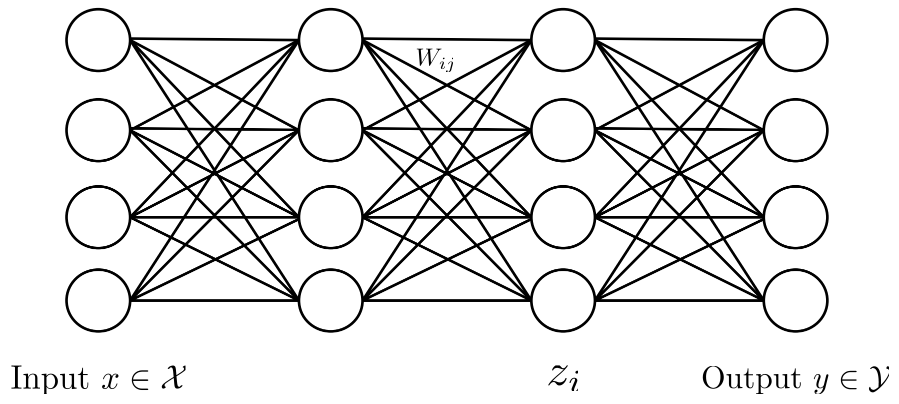
Outline

1. Where can sparsity appear in deep learning?
2. The why, what, and how of neural network pruning
3. Static sparse training and the lottery ticket hypothesis
4. Dynamic sparse training

Outline

1. **Where can sparsity appear in deep learning?**
2. The why, what, and how of neural network pruning
3. Static sparse training and the lottery ticket hypothesis
4. Dynamic sparse training

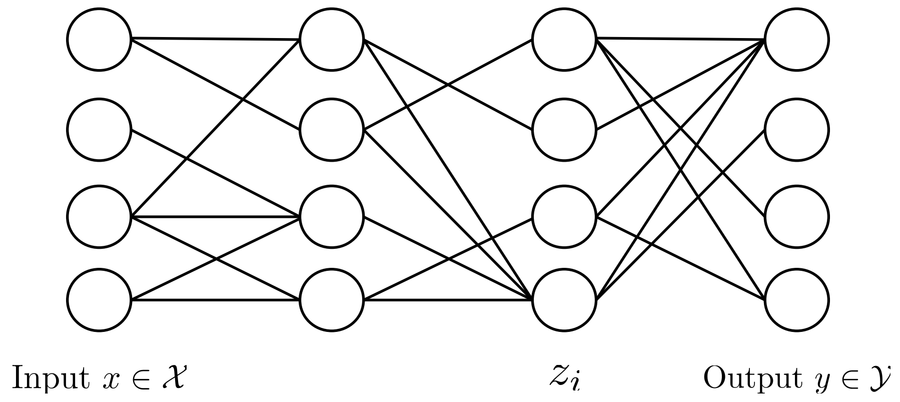
Where can sparsity appear in deep learning?



Weights

This talk

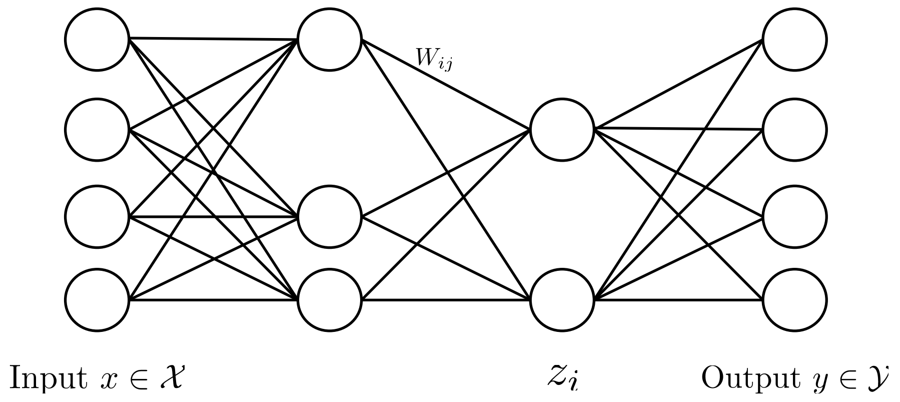
Neurons



Persistent

or

‘per input’



Where can sparsity appear in deep learning?

- Sparsity in DL is an active and growing area of research

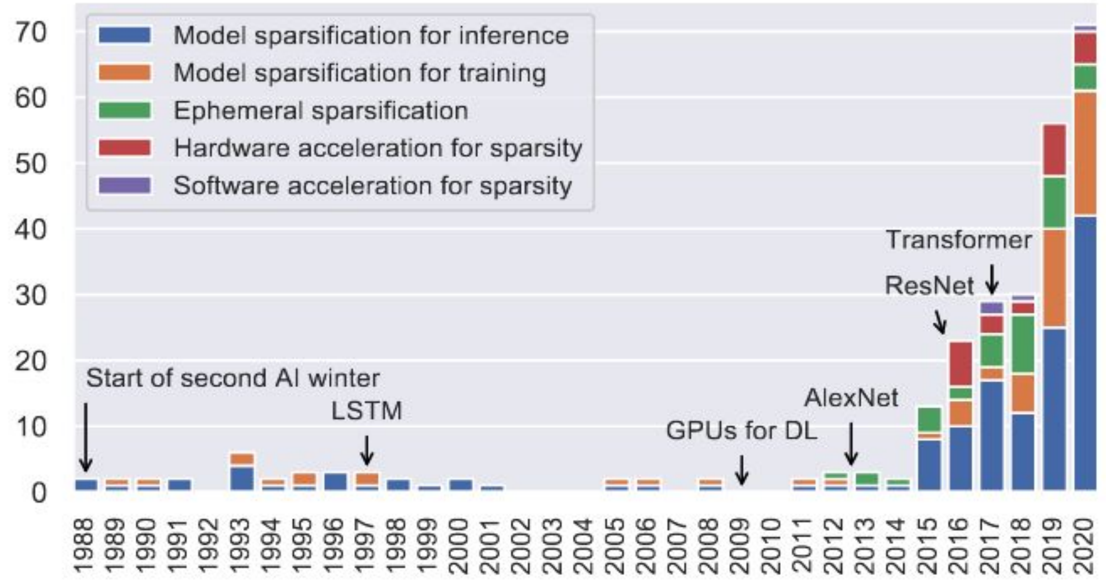


Figure 1 from [1]

- See thorough review paper by Torsten Hoefler et al. [1]

Why sparsify the weights?

- Deep networks are most-often vastly over parameterised
  - Parameter counts now range from $\mathcal{O}(10^6)$ to $\mathcal{O}(10^{12})$!
    - Resnet101 - 45 million
    - GPT-3 - 175 billion
    - T5-XXL - 1.6 trillion
  - Can even succeed with random labels (i.e. memorise the dataset) (Zhang et al, 2017)

Why sparsify the weights?

- Do we want or need this over-parameterisation?
  - Needed at inference? **No**

  Theory:
  - *"all function classes that are optimally approximated by a general class of representation systems—so-called affine systems—can be approximated by deep neural networks with minimal connectivity and memory requirements" (Bolcskei et al, 2019)*
  - *"trajectory growth can remain exponential in depth in sparse neural networks, with the sparsity parameter appearing in the base of the exponent." (Price & Tanner, 2019)*

Why sparsify the weights?

- Do we want or need this over-parameterisation?
    - Needed at inference? **No**

  Practice:
    - Pruning research often shows 20x to even 100x compression possible without sacrificing much performance.
    - Sparsifying can even improve performance.

Why sparsify the weights?

- Do we want or need this over-parameterisation?
  - Need in training? Something of an open question.
    - Benefits of overparameterisation for training (Nguyen, 2019, Arora et al, 2018, and many more)

vs.

    - Improvements in sparse training (more on this later)

Why sparsify the weights?

- Do we want or need this over-parameterisation?
    - Desirable? **No**
        - Less over-parameterised (i.e. sparser) should generalise better

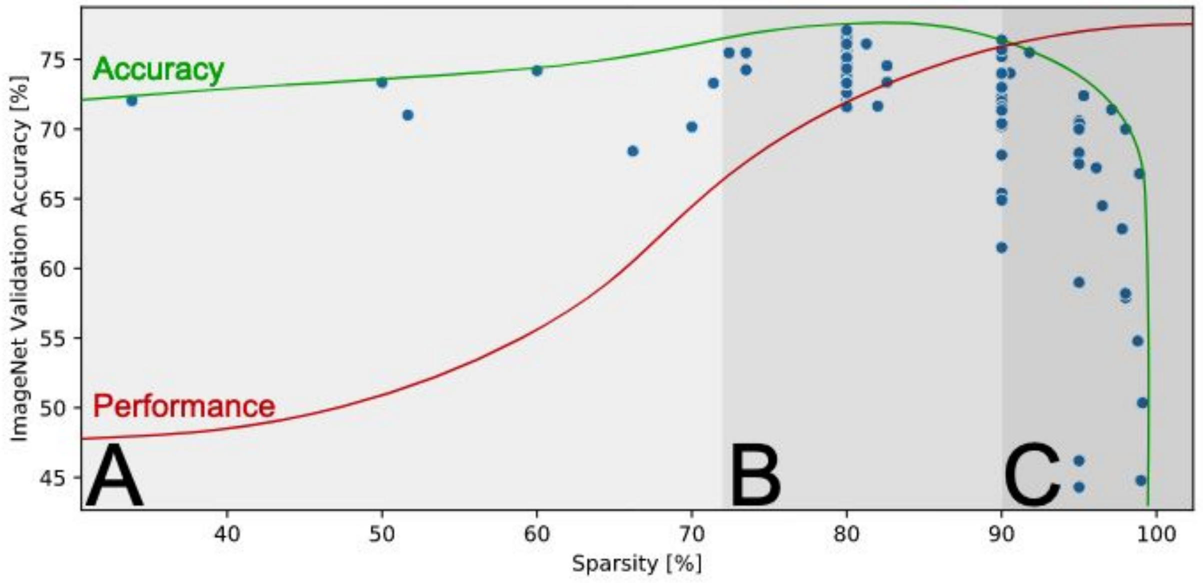


Fig 4. from [1]

Why sparsify the weights?

- Do we want or need this over-parameterisation?
  - Desirable? **No**
    - Less over-parameterised (i.e. sparser) should generalise better
    - Huge storage and computational costs. Sparsity can help:
      - sparse matrix-vector products: $\mathcal{O}(mn) \longrightarrow \mathcal{O}(pmn)$ computational cost
      - storage - different methods depending on sparsity level e.g. CSR takes $\mathcal{O}(m \log_2 n_c + n_r \log_2 m)$
    - Caveat: waiting on appropriate hardware and package implementations!

[Note: sparsity only one approach to compressing storage/compute of deep nets. See also network compression, quanization, factorisation based approaches and more]

Neural network pruning techniques

Which parameters are unimportant?

$$L(\mathrm{w} + \delta \mathrm{w}) - \mathcal{L}(\mathrm{w}) \approx \delta \mathrm{w}^\top \nabla_{\mathrm{w}} \mathcal{L}(\mathrm{w}) + \frac{1}{2} \delta \mathrm{w}^\top H(\mathrm{w}) \delta \mathrm{w}$$

1. Optimal brain damage (Lecun et al, 1989)

Assume convergence ($\nabla_{\mathrm{w}} \mathcal{L} \approx 0$), and approx. H with its diagonal then impact on the loss of removing w_i is given by

$$\frac{\mathrm{w}_i^2 \nabla_{\mathrm{w}_i}^2 \mathcal{L}}{2}$$:= a salience score per parameter

Even more naive: approx. Hessian as the identity:

2. Iterative Magnitude Pruning (IMP): iteratively zero weights with largest magnitude

And other ways to approximate the Hessian, e.g. (Wang et al, 2019)

Neural network pruning techniques

Many other methods:

- Variational Dropout (Molchanov et al, 2017)
- L0-regularisation (Louizos et al., 2018)
- Methods which drop ‘stagnant’’ weights (i.e. ‘not updated’ as a proxy for ‘not important’) (Karnin, 1990)
- LAP (magnitude pruning which accounts for the magnitude of incoming and outgoing connections of the neurons it connects) (Park et al, 2020)

And more! See [1]. As a general rule: hard to consistently beat well done done iterative magnitude pruning (Gale et al, 2019)

Outline

1. Where can sparsity appear in deep learning?
2. The why, what, and how of neural network pruning
3. **Static sparse training and the lottery ticket hypothesis**
4. Dynamic sparse training

Static sparse training

- So far, pruning only during or after training means training still is expensive. Can the nets be sparse from scratch?

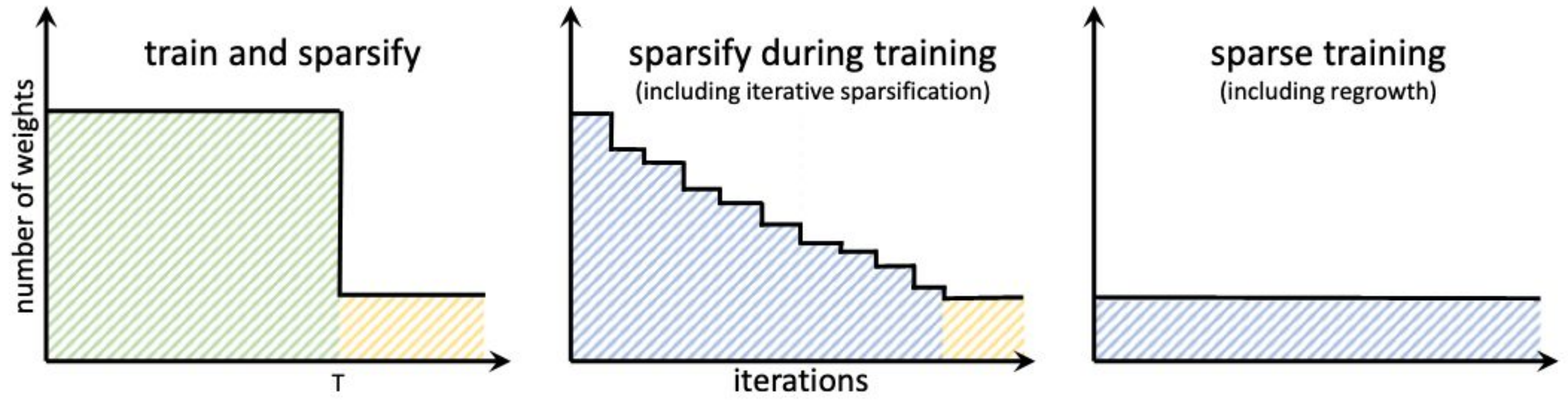


Fig 7. from [1]

Static sparse training

- 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 do they show this?

  Step 1. Train and prune with IMP to get accurate sparse network.

  Step 2. Fix the resulting sparse support.

  Step 3. Rewind those remaining weights to their initial values in that training run.

  Step 4. Train as sparse from that specified initialisation

Pruning-at-initialisation

- But can we find these ‘winning tickets’ before training? Aka: Pruning at Initialisation (PaI)

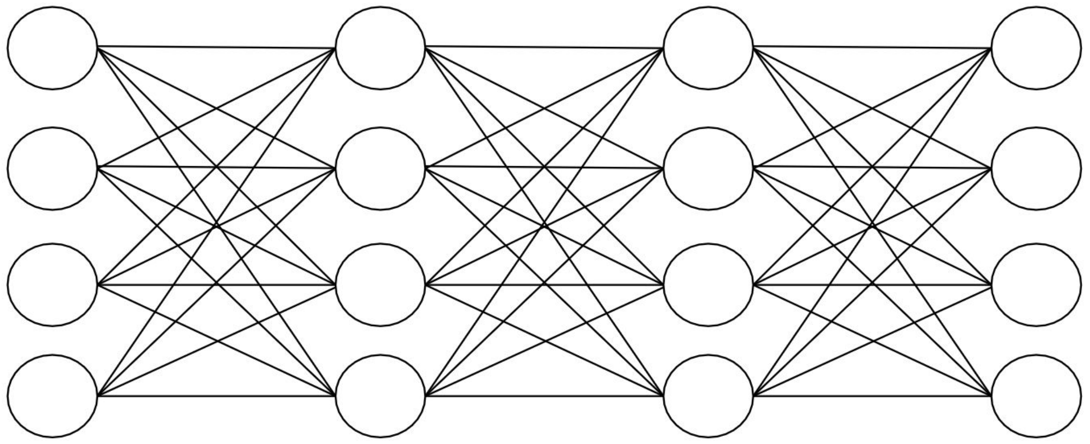
Pruning-at-initialisation

- But can we find these ‘winning tickets’ before training? Aka: Pruning at Initialisation (PaI)

- General framework:

1. Initialize a dense network

2. Define scalar objective $\mathcal{R}$



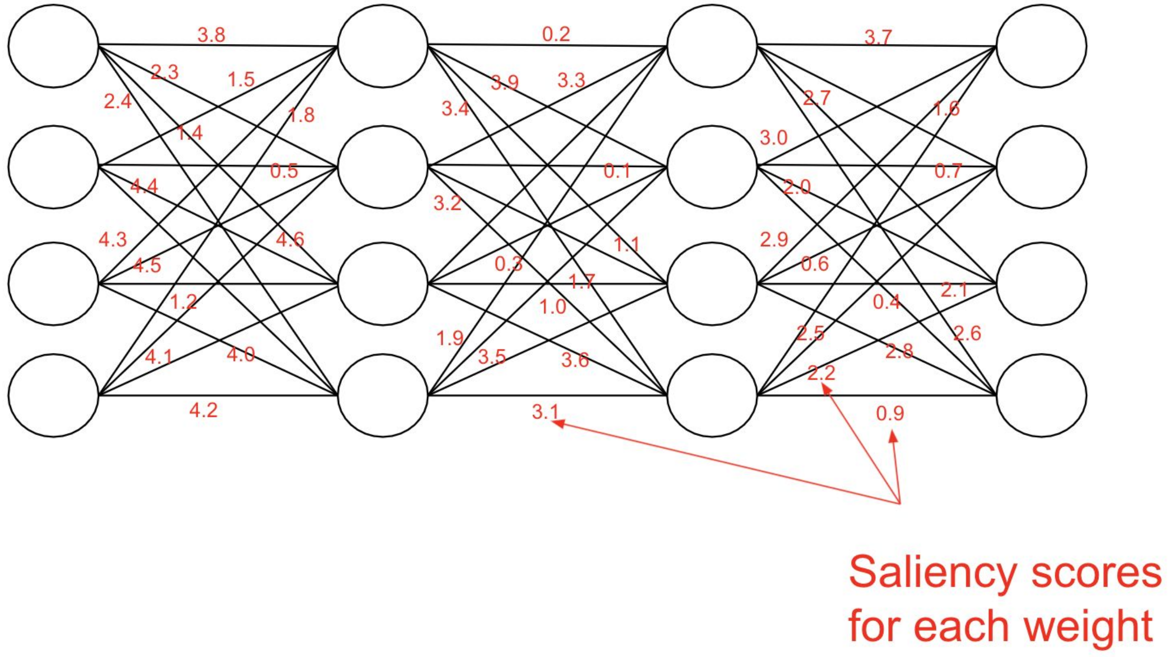
Pruning-at-initialisation

- But can we find these ‘winning tickets’ before training? Aka: Pruning at Initialisation (PaI)

- General framework:

1. Initialize a dense network
2. Define scalar objective $\mathcal{R}$
3. Calculate vector of saliency scores

$$G(\mathbf{w}) = \frac{\partial \mathcal{R}}{\partial \mathbf{w}} \odot \mathbf{w}$$



Pruning-at-initialisation

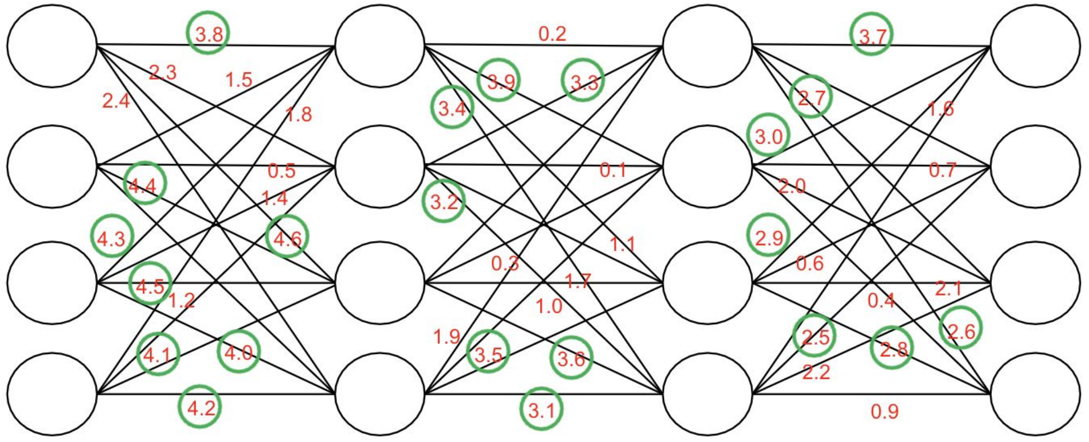
- But can we find these ‘winning tickets’ before training? Aka: Pruning at Initialisation (PaI)

- General framework:

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



Pruning-at-initialisation

- But can we find these ‘winning tickets’ before training? Aka: Pruning at Initialisation (PaI)

- General framework:

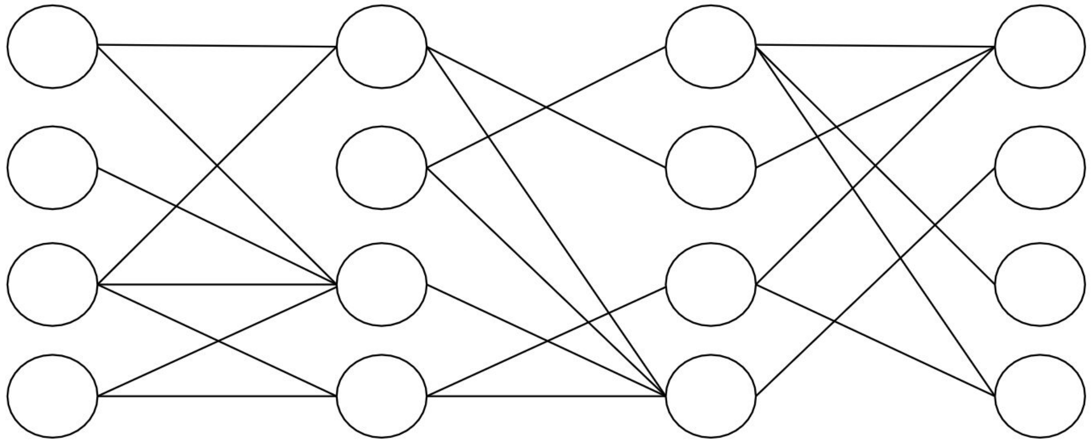
1. Initialize a dense network

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



Pruning-at-initialisation methods

- (Random)

- SNIP
  (Lee et al. 2019)

$$G(\mathrm{w}) = \left|\frac{\partial \mathcal{L}(\mathrm{w})}{\partial \mathrm{w}} \odot \mathrm{w}\right|$$

- GraSP
  (Wang et al. 2019)

$$G(\mathrm{w}) = -\left(H\frac{\partial \mathcal{L}(\mathrm{w})}{\partial \mathrm{w}}\right) \odot \mathrm{w}$$

- FORCE
  (de Jorge et al. 2021)

$$G(\mathrm{w}) = \left|\frac{\partial \mathcal{L}(\bar{\mathrm{w}})}{\partial \mathrm{w}} \odot \mathrm{w}\right|$$

$\bar{\mathrm{w}}$ is the parameter vector after pruning

- SynFlow
  (Tanaka et al. 2020)

$$\mathcal{R} = \mathbb{1}^\top \left(\prod_{l=1}^{L} \left|\mathrm{w}^{[l]}\right|\right) \mathbb{1}$$

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

Pruning-at-initialisation vs. Random

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 (Frankle et al, 2021)

2. Random subspace training suffices (for non-sparse subspaces) (Li et al, 2018)

3. Our recent work on DCT plus Sparse networks: After offsetting from the origin, random sparse support for trainable parameters suffices. (Price and Tanner, 2021)

Outline

1. Where can sparsity appear in deep learning?
2. The why, what, and how of neural network pruning
3. Static sparse training and the lottery ticket hypothesis
4. **Dynamic sparse training**

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.
- Methods defined by:
  - Criterion for pruning?
  - Criterion for regrowing?
  - Where does the pruning and regrowing occur? (ie. only within layers or redistributed between layers?)

Dynamic Sparse Training Algorithms

- SET (Mocanu et al, 2018)
  - Initialise W_i as Erdos Reyni random bipartite graph
  - Prune a fraction of the smallest magnitude weights
  - Regrow randomly

- DSR (Mostafa & Wang, 2019)
  - Similar to SET, but:
    - Proportion pruned is not constant for all iterations
    - Connections can be reallocated across layers

- RigL (ERK) (Evci et al, 2020) (SOTA)
  - Initialise all W_i as Erdos Reyni but modified to account for kernel dimensions
  - Prune based on magnitude
  - Regrow based on gradient magnitude

And others, eg. Deep-R (Bellec et al, 2018), SNFS (Dettmers & Zettlemoyer, 2019)

Questions