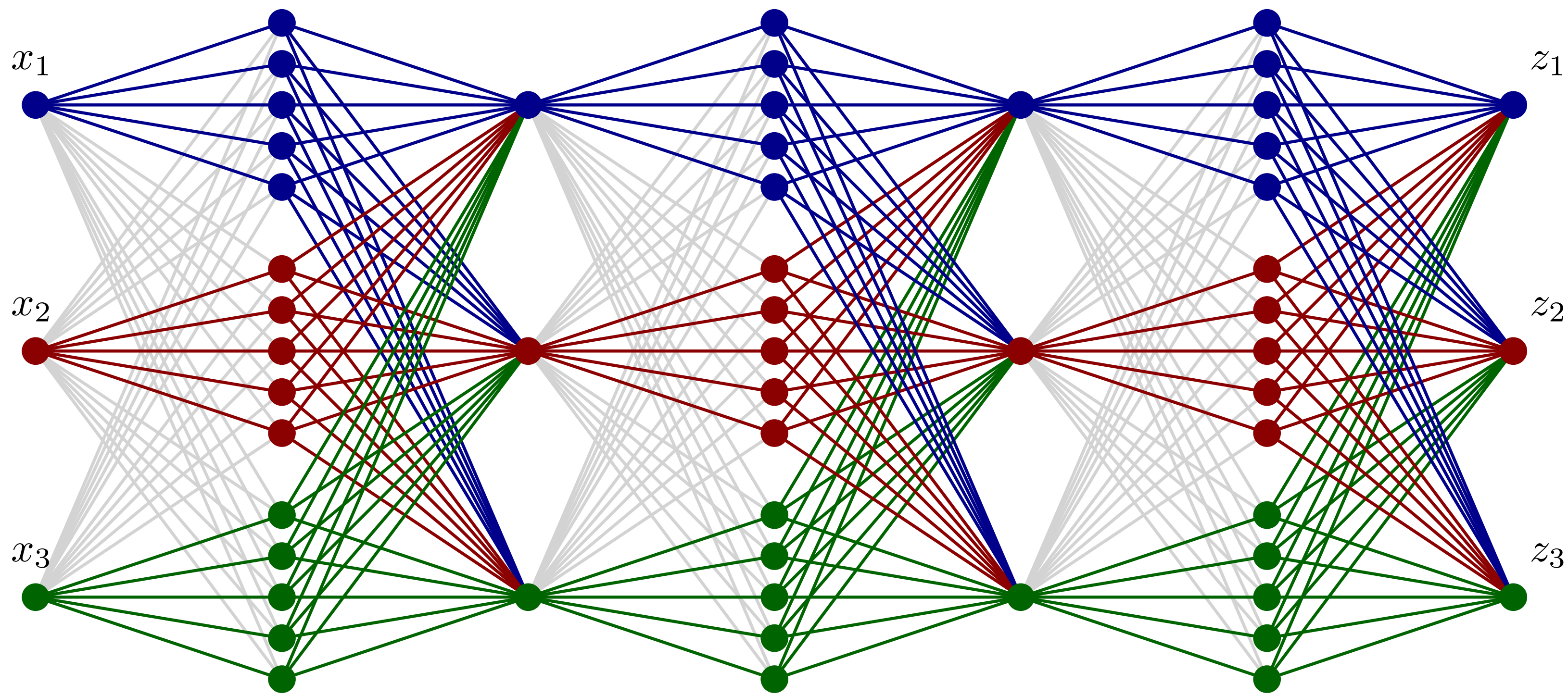
The Strong Lottery Ticket Hypothesis and the Random Subset Sum Problem



Frederik Mallmann-Trenn

King's College London

30 June 2025

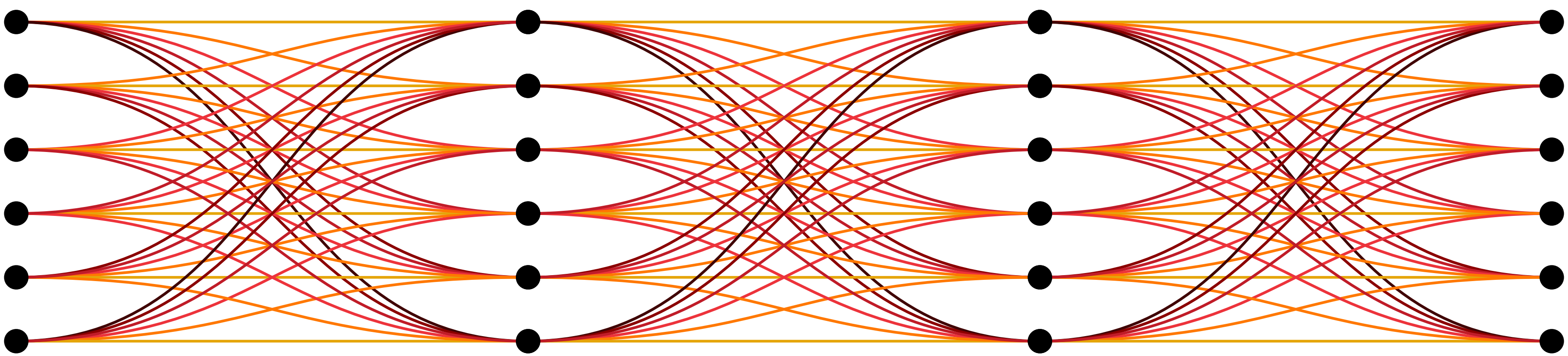
The slides are based on Francesco d'Amore's (Gran Sasso Science Institute in Italy) slides

Thank you so much!

Artificial neural networks are large

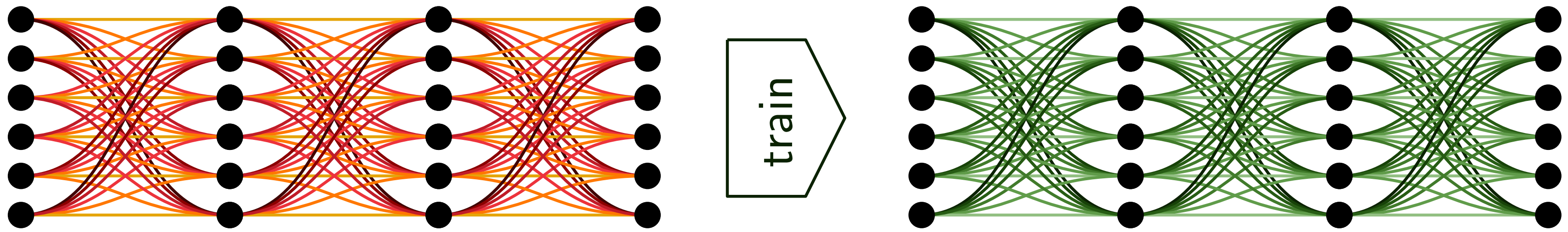
Usually ranging from millions to hundreds of billions parameters

- RESNET-50: > 20 millions parameters [He et al. 2015]
- BERT: > 100 millions parameters [Devlin et al. 2018]
- GPT-3: > 100 billions parameters [Brown et al. 2020]



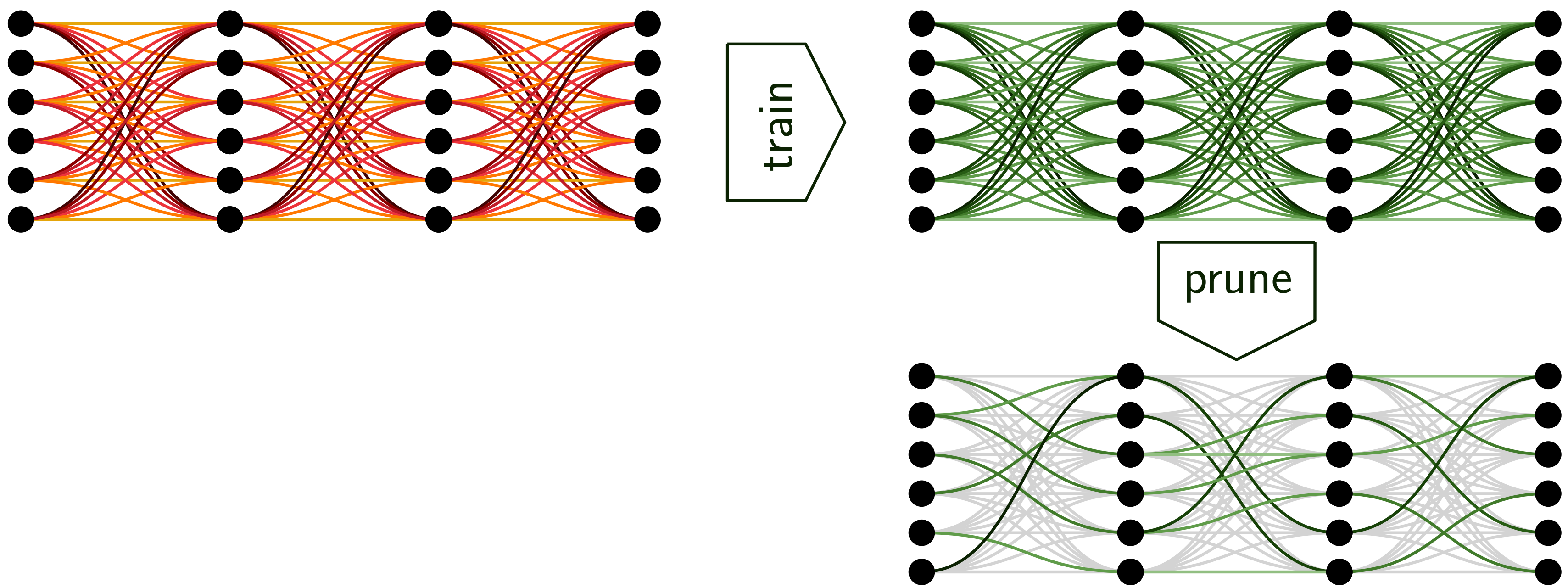
Training and Inference are expensive

- Training large and dense networks yields good results
- However, it is very resource intensive



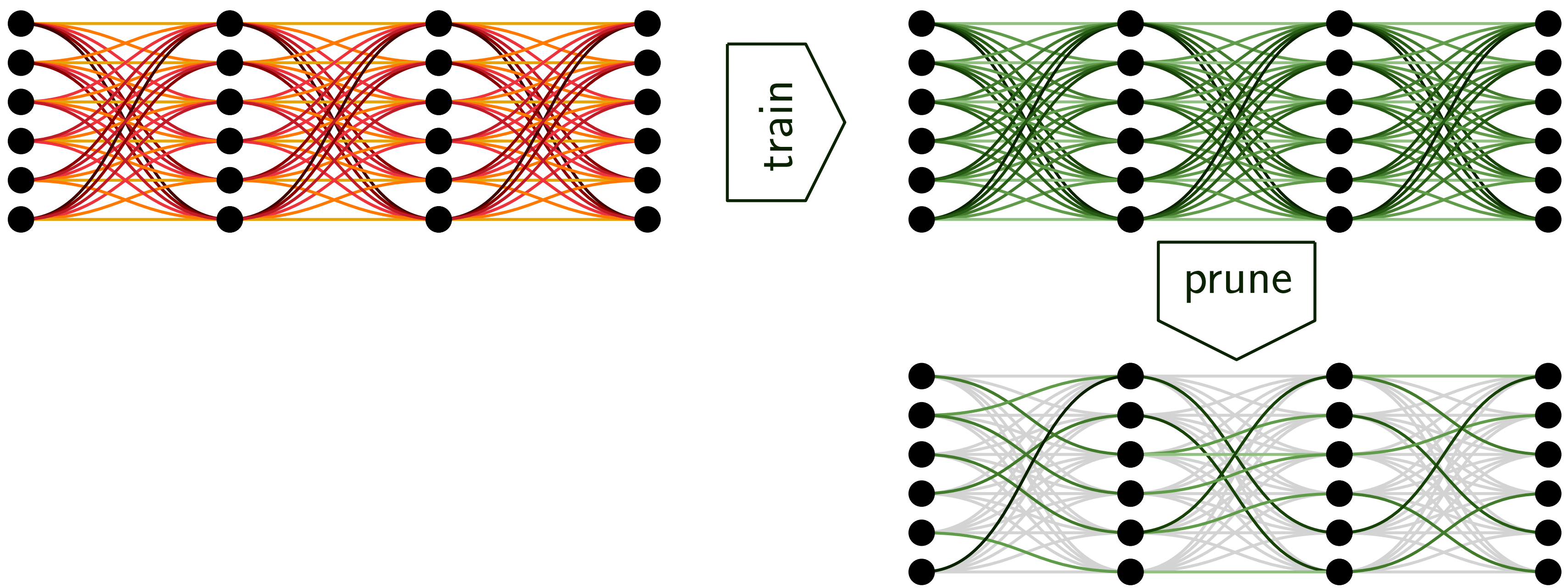
Training and Inference are expensive

- Training large and dense networks yields good results
- However, it is very resource intensive
- To make them smaller we can remove edges (pruning), which works well



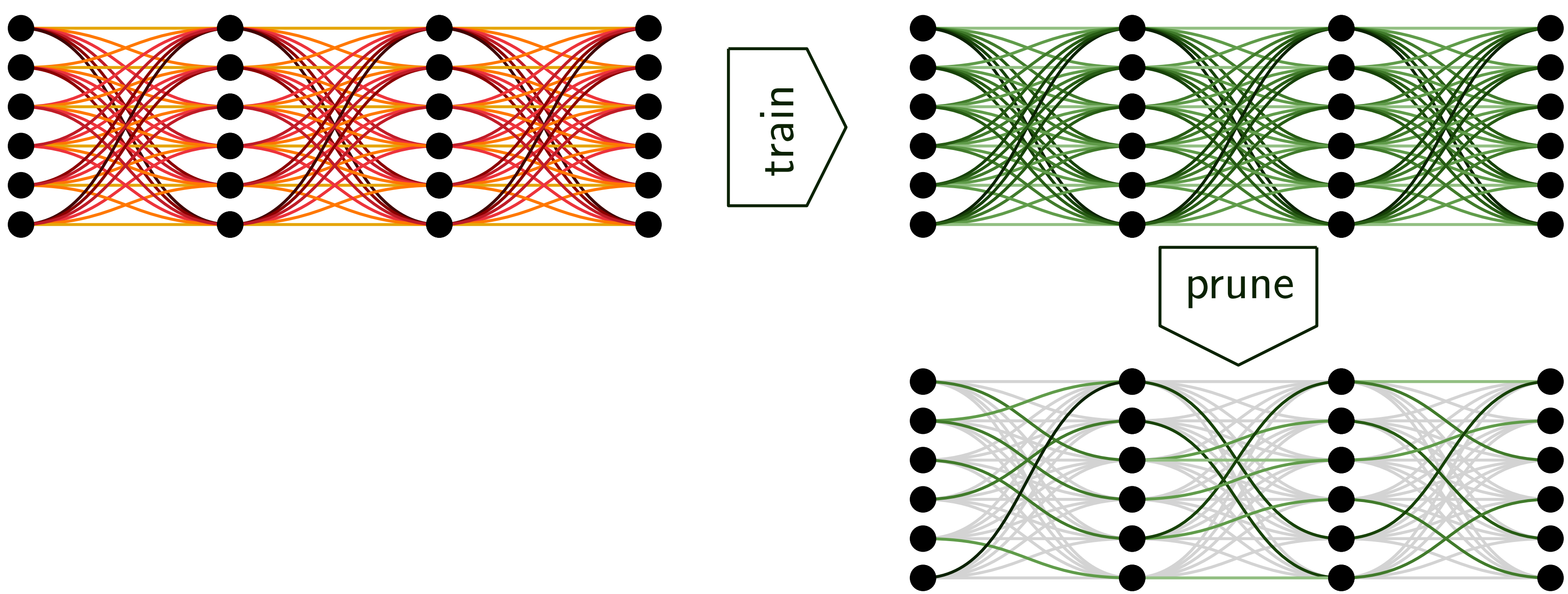
Training and Inference are expensive

- Training large and dense networks yields good results
- However, it is very resource intensive
- To make them smaller we can remove edges (pruning), which works well
- Pruning $\sim 60 - 80\%$ of the edges can lead to better accuracies [Diffenderfer and Kailkhura 2021]



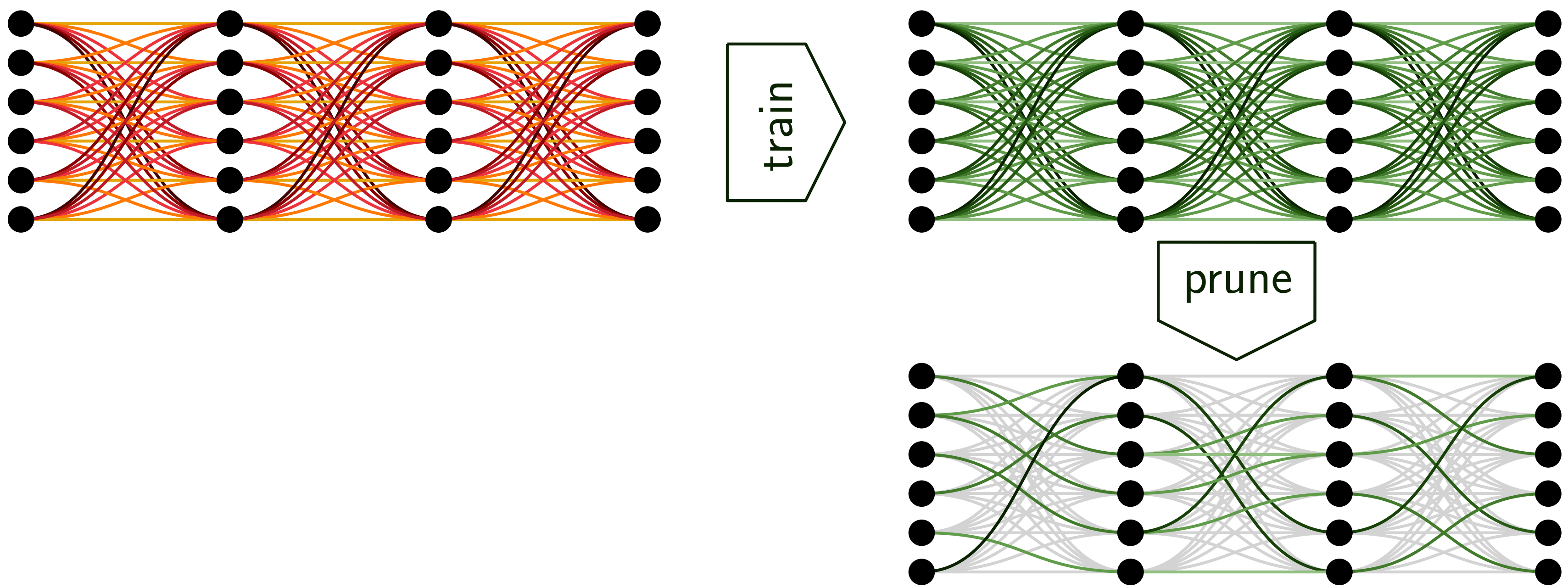
Training and Inference are expensive

- Training large and dense networks yields good results
- However, it is very resource intensive
- To make them smaller we can remove edges (pruning), which works well
- Pruning $\sim 60 - 80\%$ of the edges can lead to better accuracies [Diffenderfer and Kailkhura 2021]
- Pruning $\sim 99\%$ of the edges can perform well [Hoefler et al. 2021]



What if we train the tiny one?

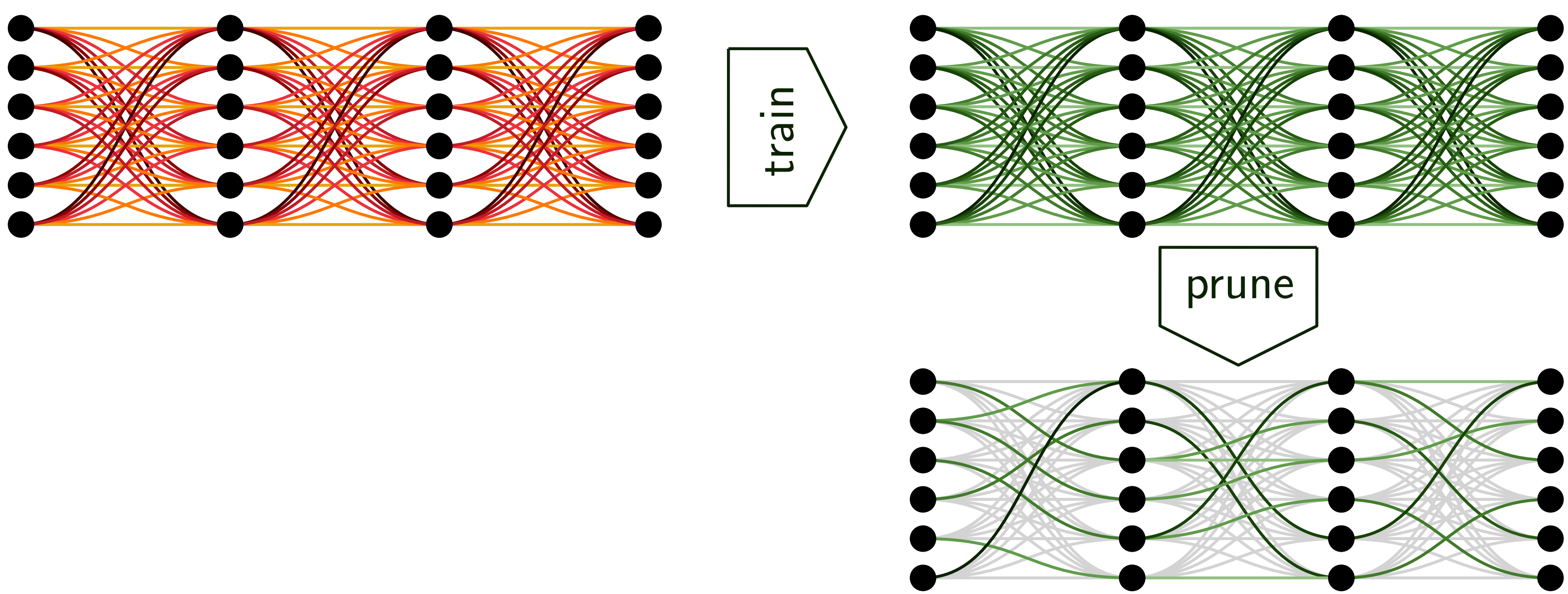
- Maybe, we can avoid the effort of dense training



What if we train the tiny one?

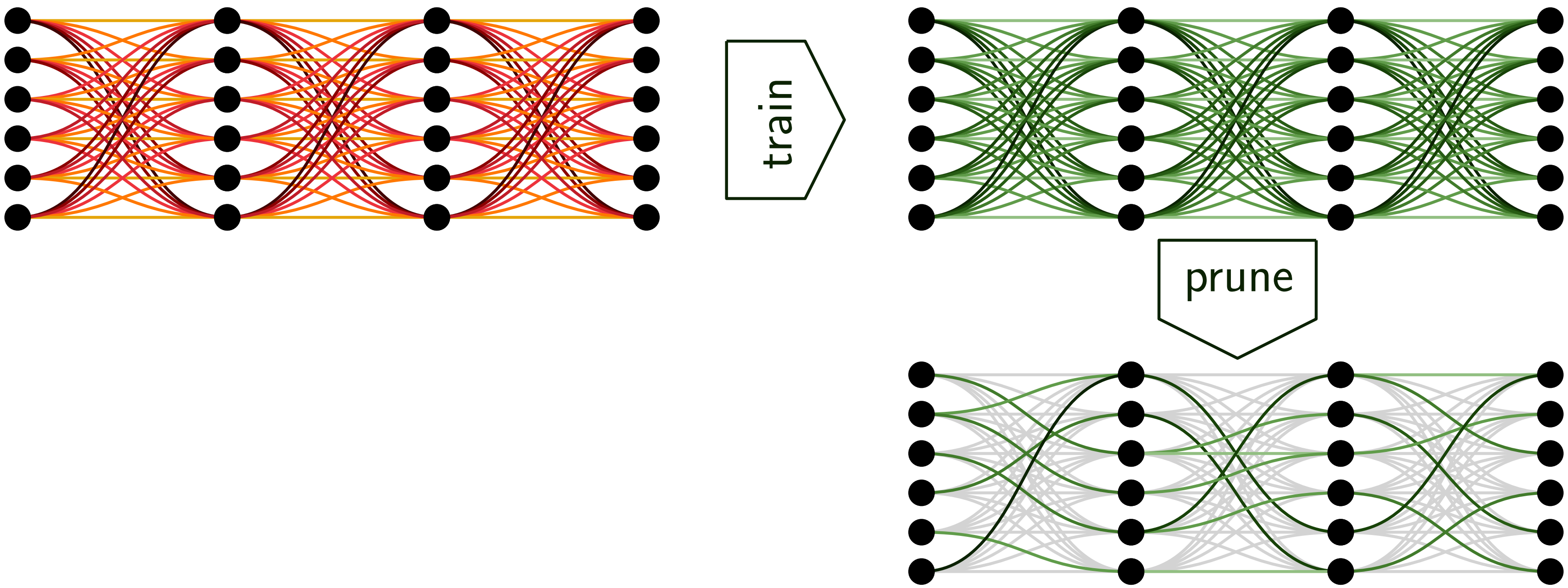
- Maybe, we can avoid the effort of dense training
- **The Lottery Ticket Hypothesis:**
  "A randomly-initialized, dense neural network contains a subnetwork that is initialized such that—when trained in isolation—it can match the test accuracy of the original network after training for at most the same number of iterations."

[Frankle and Carbin 2019, ICLR]



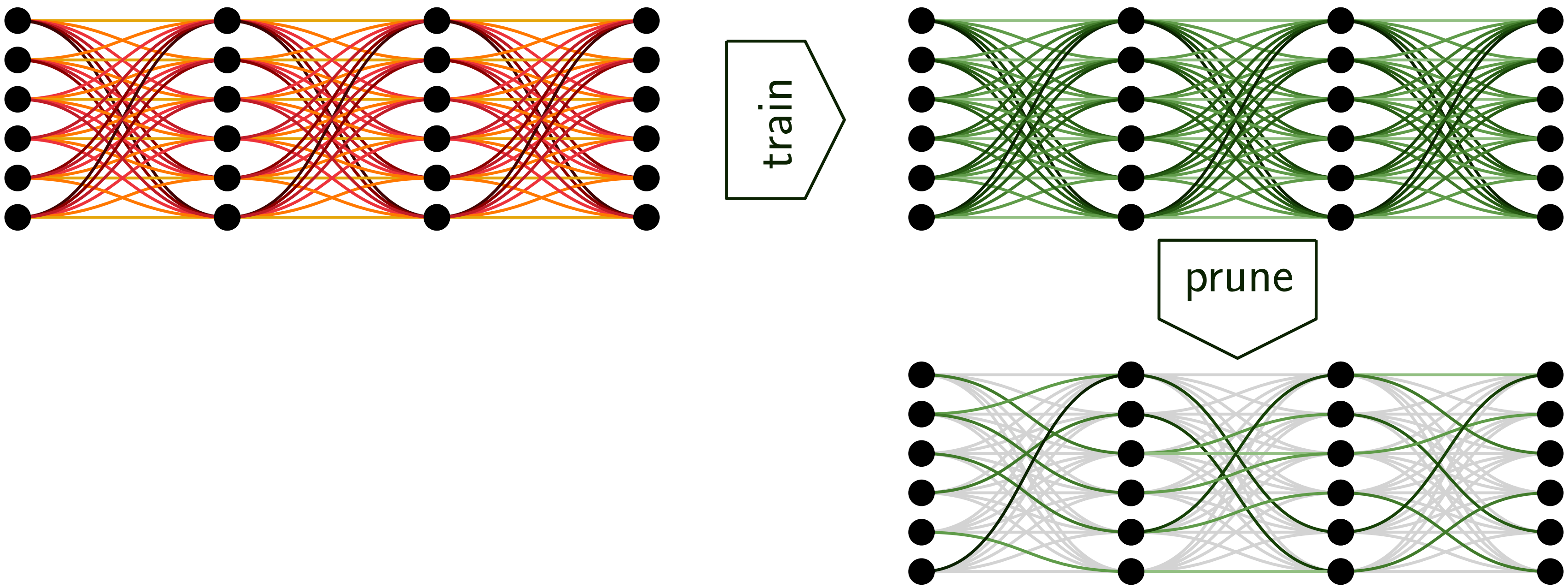
What if we train the tiny one?

- To validate this hypothesis, let's train a dense network, then prune it



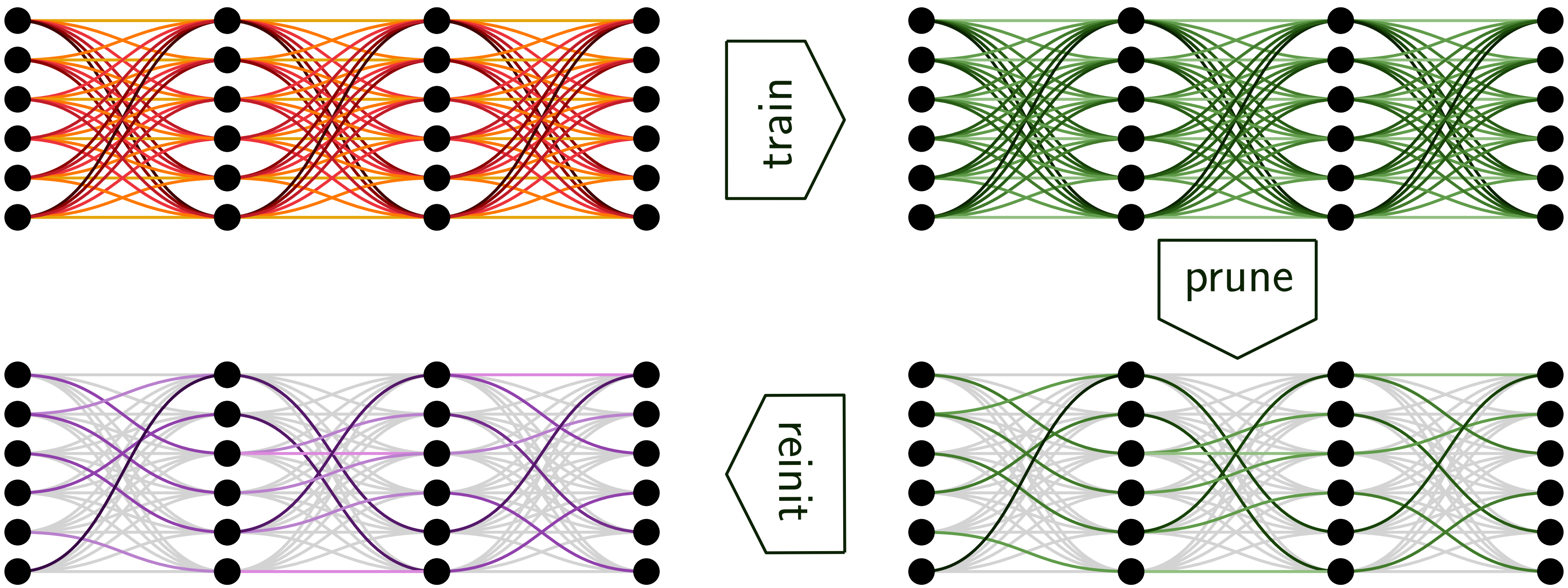
What if we train the tiny one?

- To validate this hypothesis, let's train a dense network, then prune it
- Let's test the subnetwork by retraining it



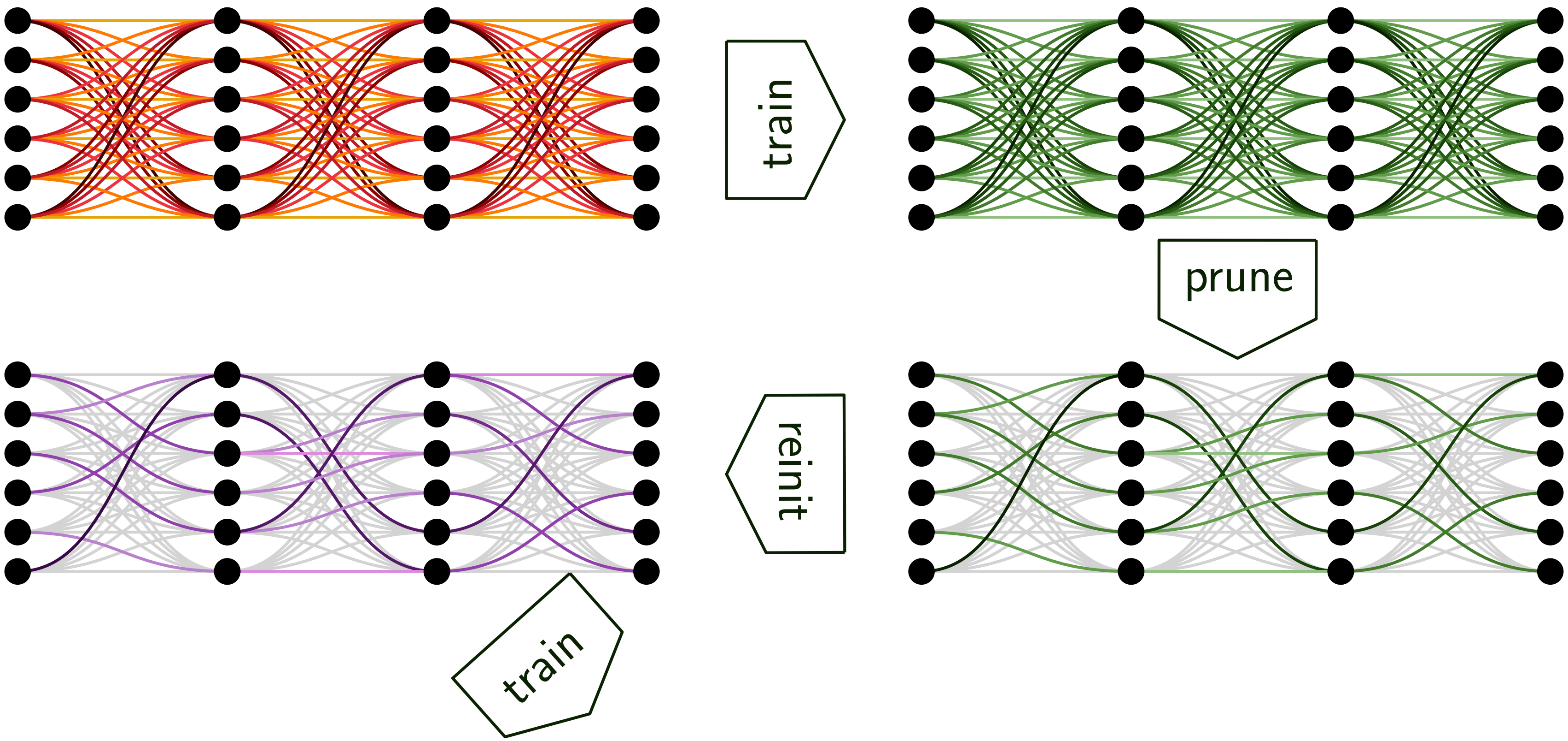
What if we train the tiny one?

- To validate this hypothesis, let's train a dense network, then prune it
- Let's test the subnetwork by retraining it
    - Reinitialize



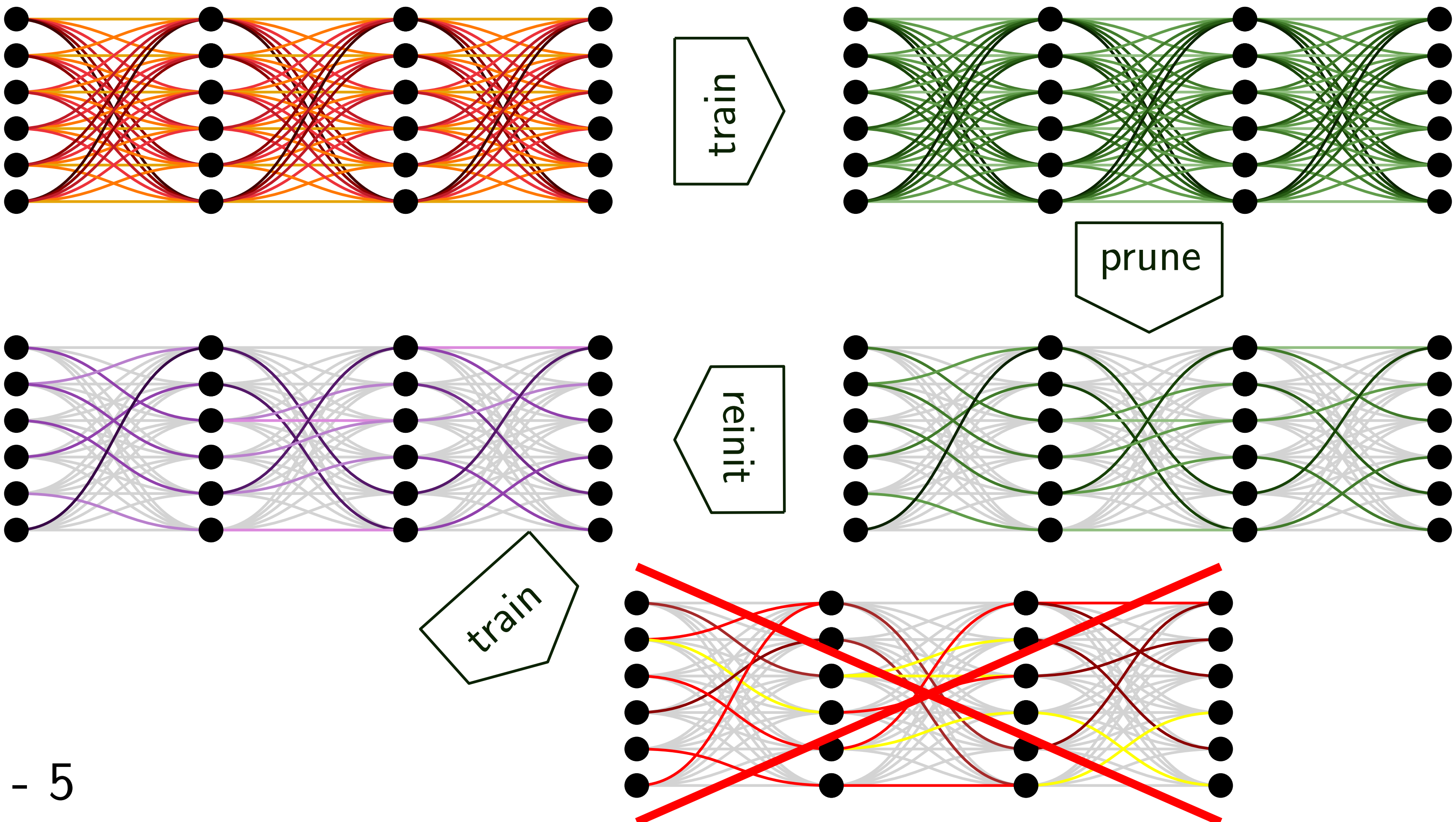
What if we train the tiny one?

- To validate this hypothesis, let's train a dense network, then prune it
- Let's test the subnetwork by retraining it
    - Reinitialize
    - Train



What if we train the tiny one?

- To validate this hypothesis, let's train a dense network, then prune it
- Let's test the subnetwork by retraining it
    - Reinitialize
    - Train
    - Bad accuracies

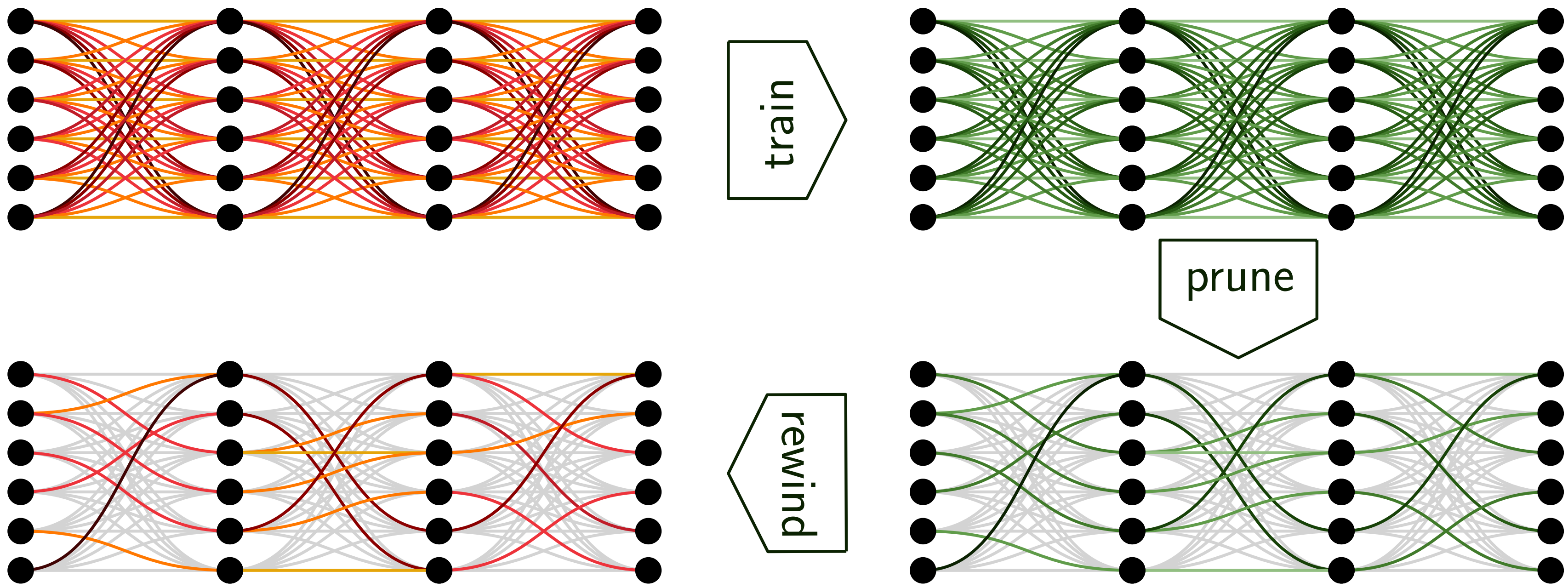


Not reinitialization, rewind instead

- Starting from a random point might be too much

[Frankle and Carbin 2019, ICLR]

- Rewind instead (back to initial random weights)

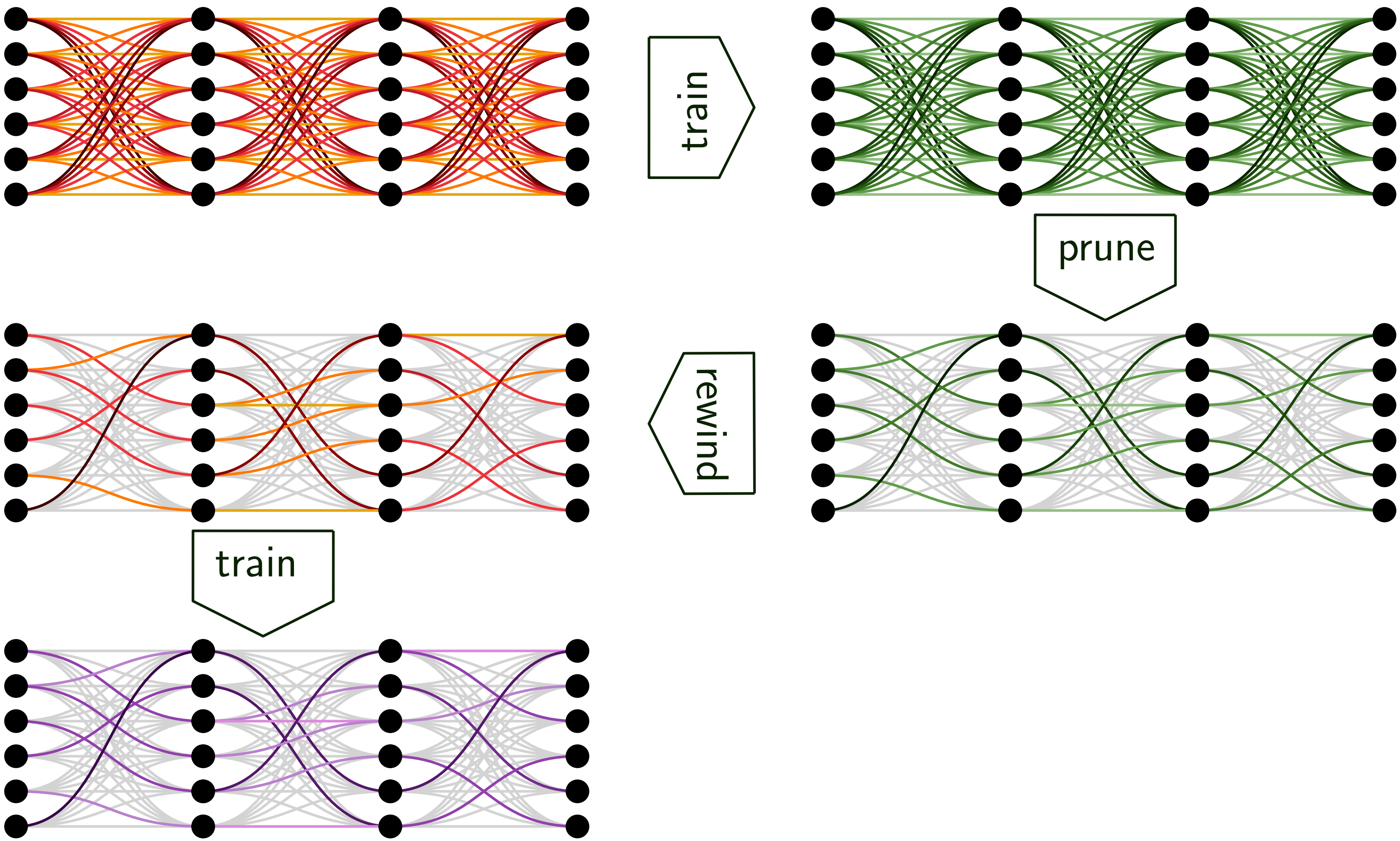


Not reinitialization, rewind instead

- Starting from a random point might be too much

[Frankle and Carbin 2019, ICLR]

- Rewind instead

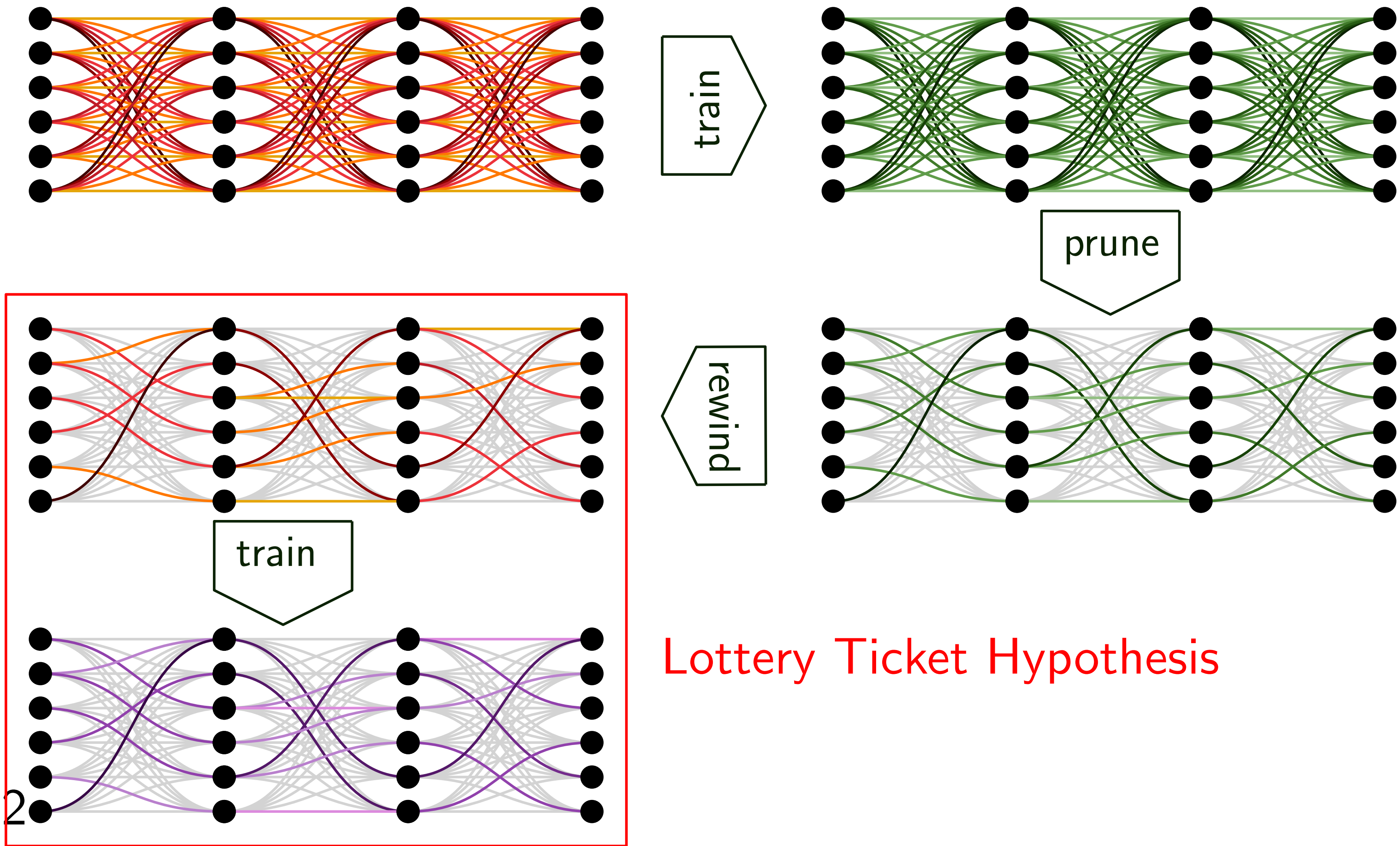


Not reinitialization, rewind instead

- Starting from a random point might be too much

[Frankle and Carbin 2019, ICLR]

- Rewind instead



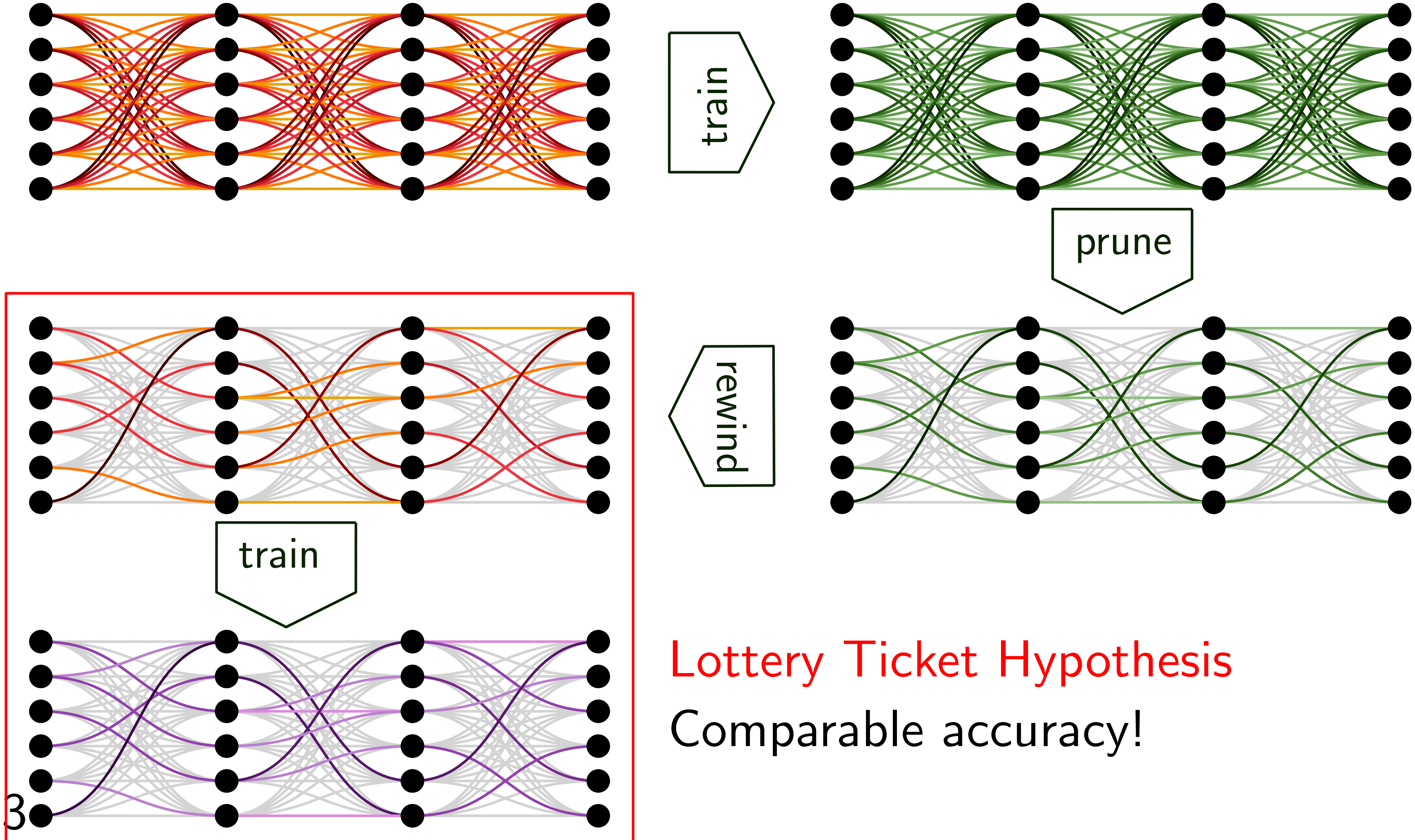
Lottery Ticket Hypothesis

Not reinitialization, rewind instead

- Starting from a random point might be too much

[Frankle and Carbin 2019, ICLR]

- Rewind instead

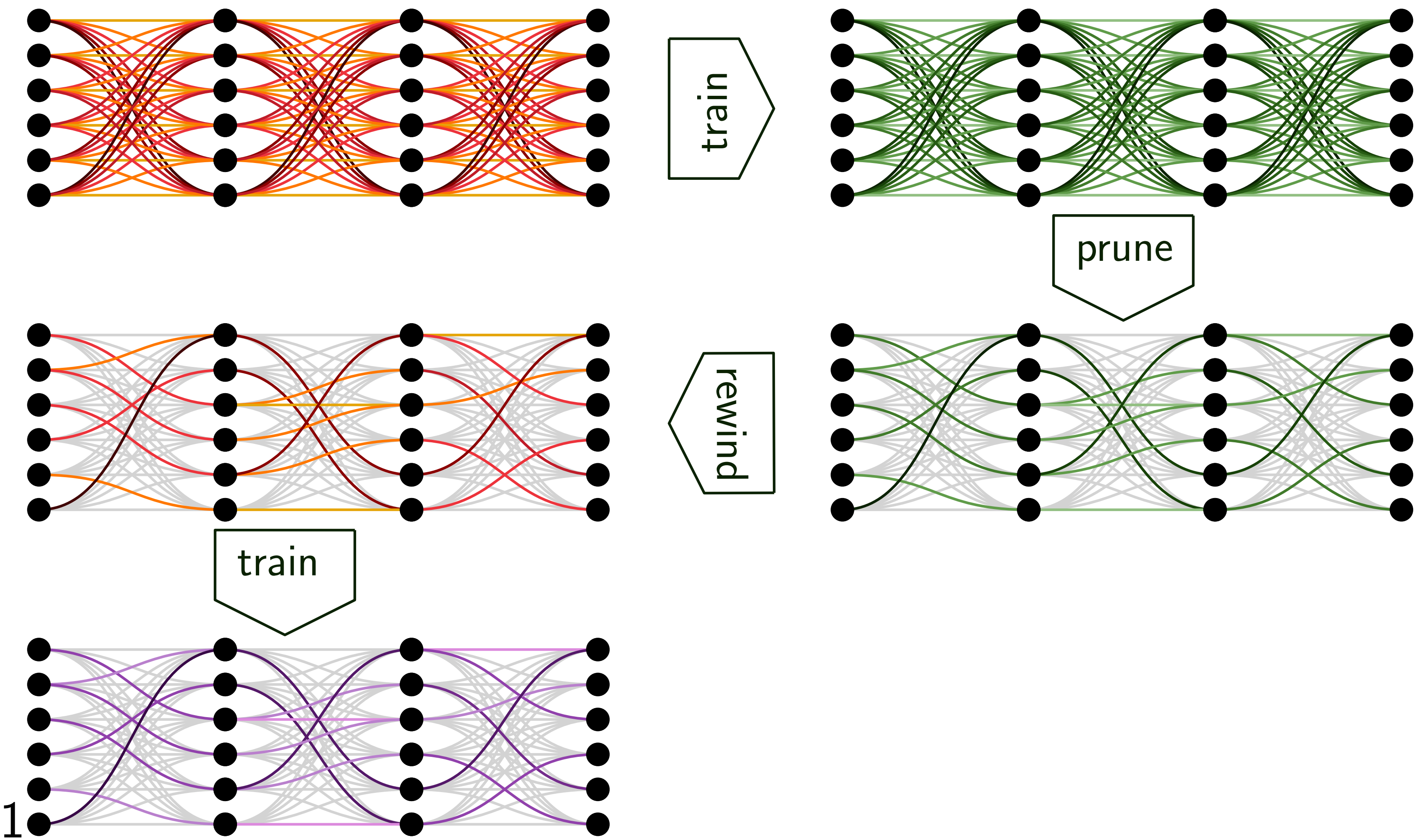


Lottery Ticket Hypothesis

Comparable accuracy!

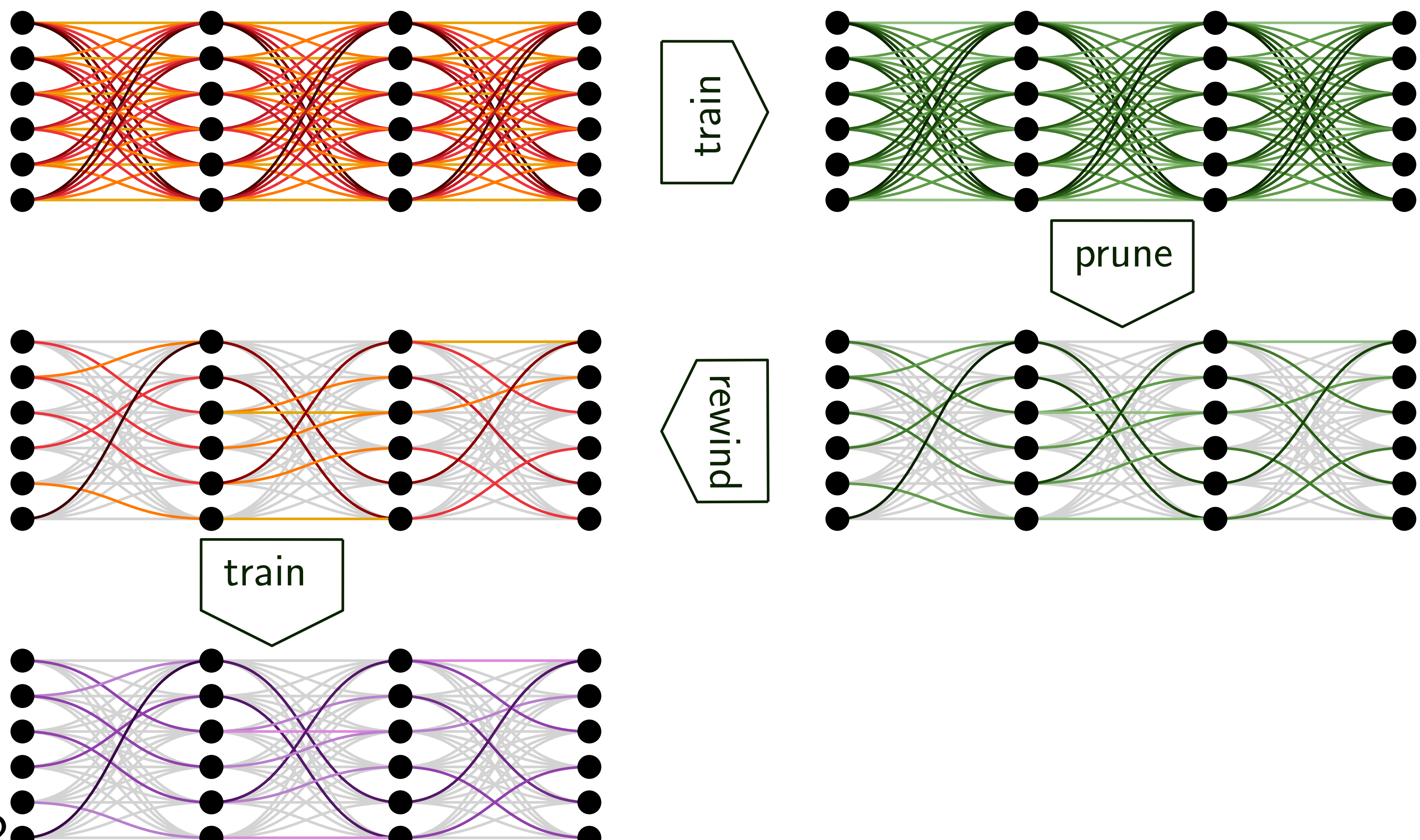
Lottery tickets

- What does it mean?



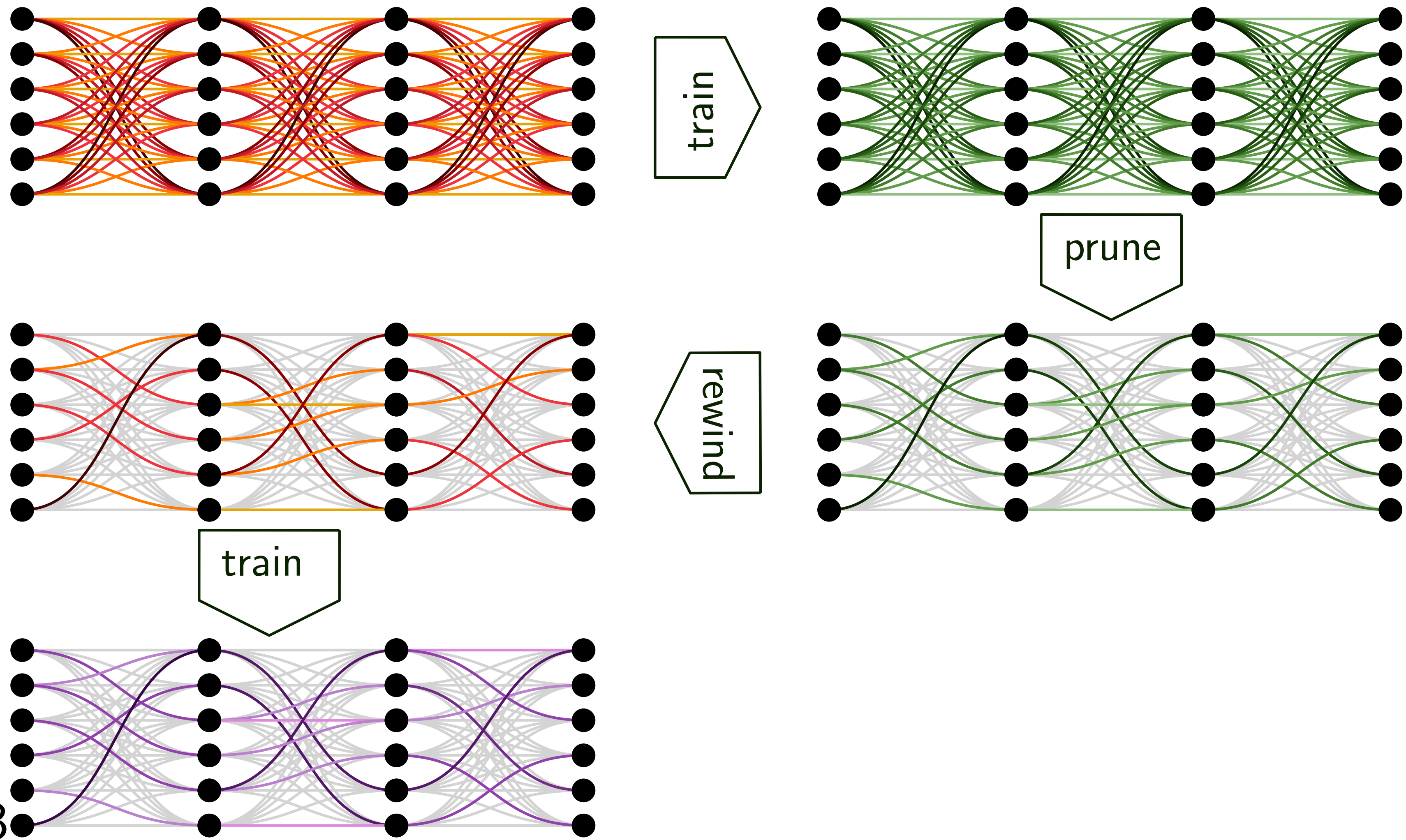
Lottery tickets

- What does it mean?
- This is not a good algorithm (we are still training a dense network)



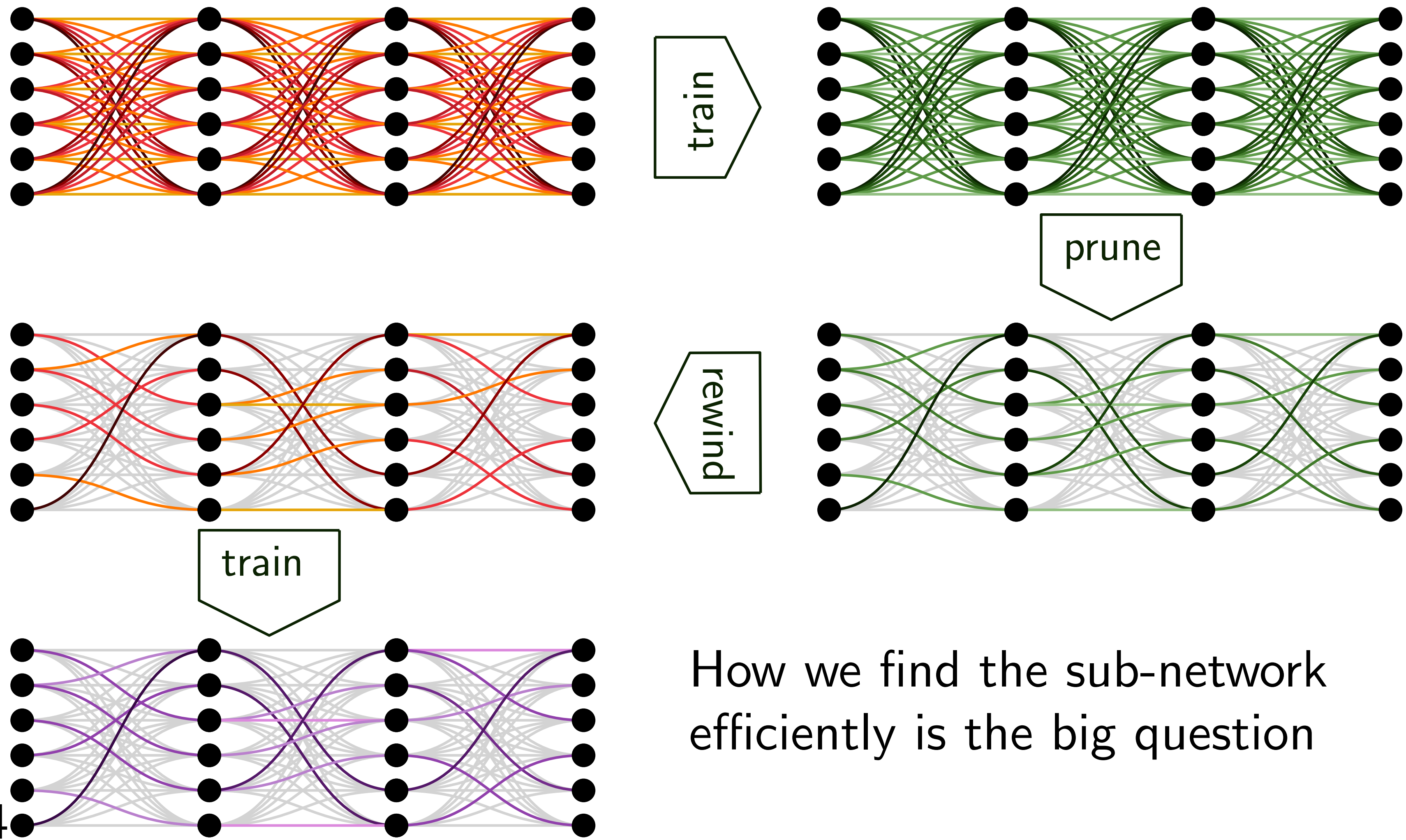
Lottery tickets

- What does it mean?
- This is not a good algorithm (we are still training a dense network)
- Existential result
    - Training is about topology + initialization



Lottery tickets

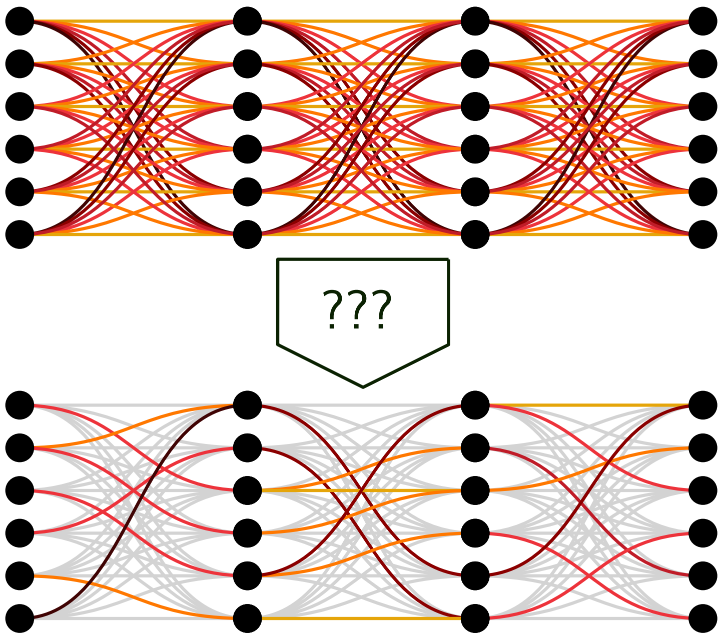
- What does it mean?
- This is not a good algorithm (we are still training a dense network)
- Existential result
    - Training is about topology + initialization



How we find the sub-network efficiently is the big question

The Lottery Ticket Hypothesis (LTH)

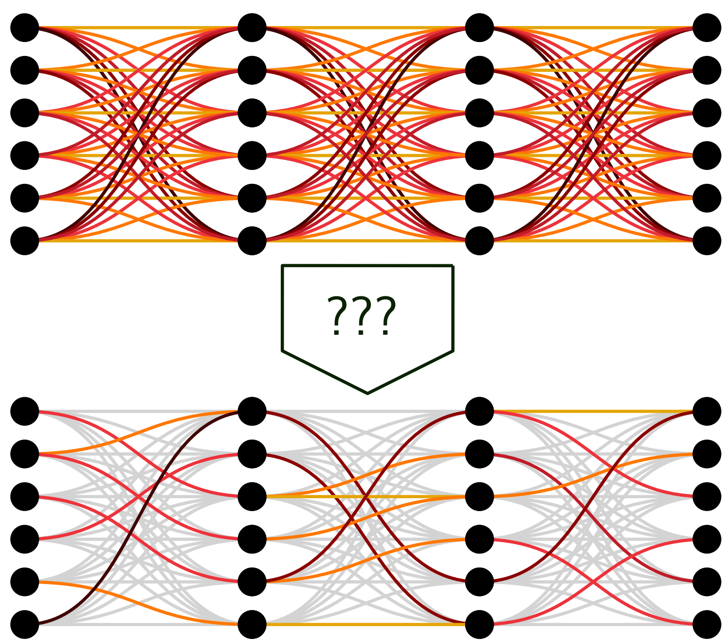
[Frankle and Carbin 2019, ICLR]: winning lottery tickets always exists



The Lottery Ticket Hypothesis (LTH)

[Frankle and Carbin 2019, ICLR]: winning lottery tickets always exists

How do we find lottery ticket without training a dense network?

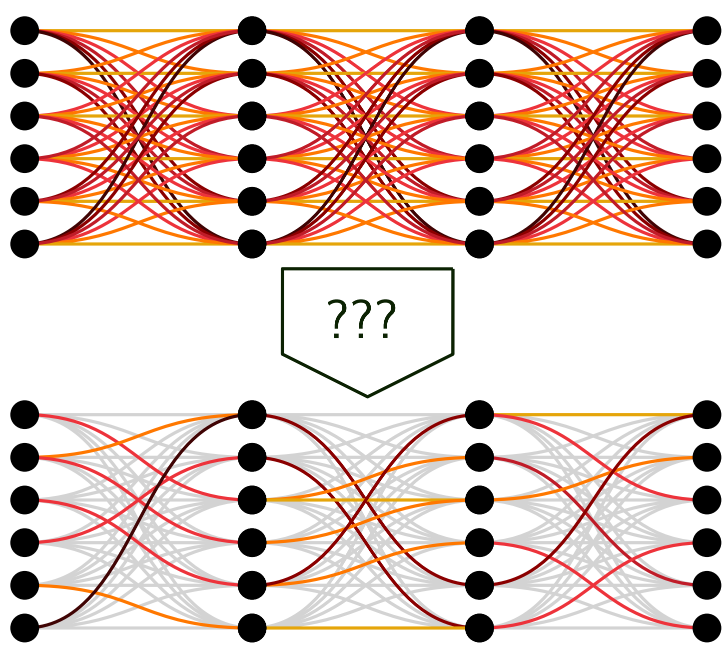


The Lottery Ticket Hypothesis (LTH)

[Frankle and Carbin 2019, ICLR]: winning lottery tickets always exists

How do we find lottery ticket without training a dense network?

Lot of subsequent work . . . (but no definitive answer)



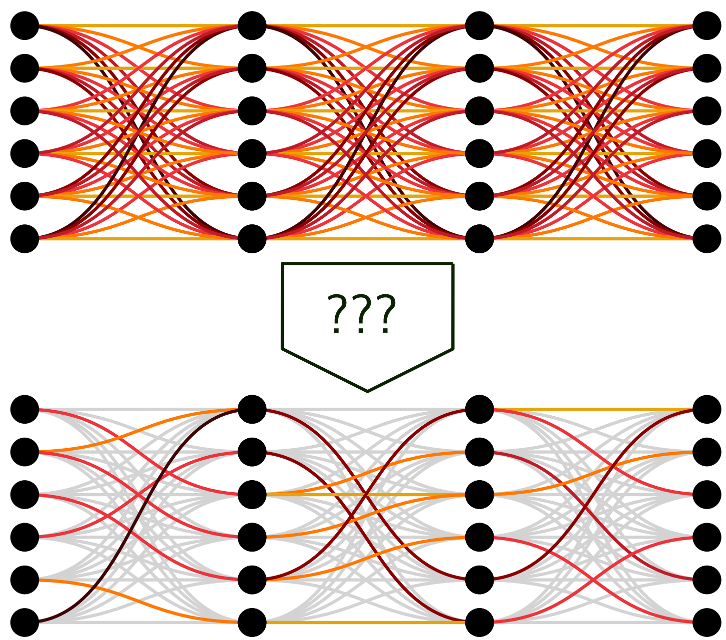
The Lottery Ticket Hypothesis (LTH)

[Frankle and Carbin 2019, ICLR]: winning lottery tickets always exists

How do we find lottery ticket without training a dense network?

Lot of subsequent work . . . (but no definitive answer)

If we want to understand deep learning, we should probably understand this first.



The *Strong* Lottery Ticket Hypothesis (SLTH)

Intuition

- Do we really need to train any parameters? Image we start with a incredibly large, and random network

The *Strong* Lottery Ticket Hypothesis (SLTH)

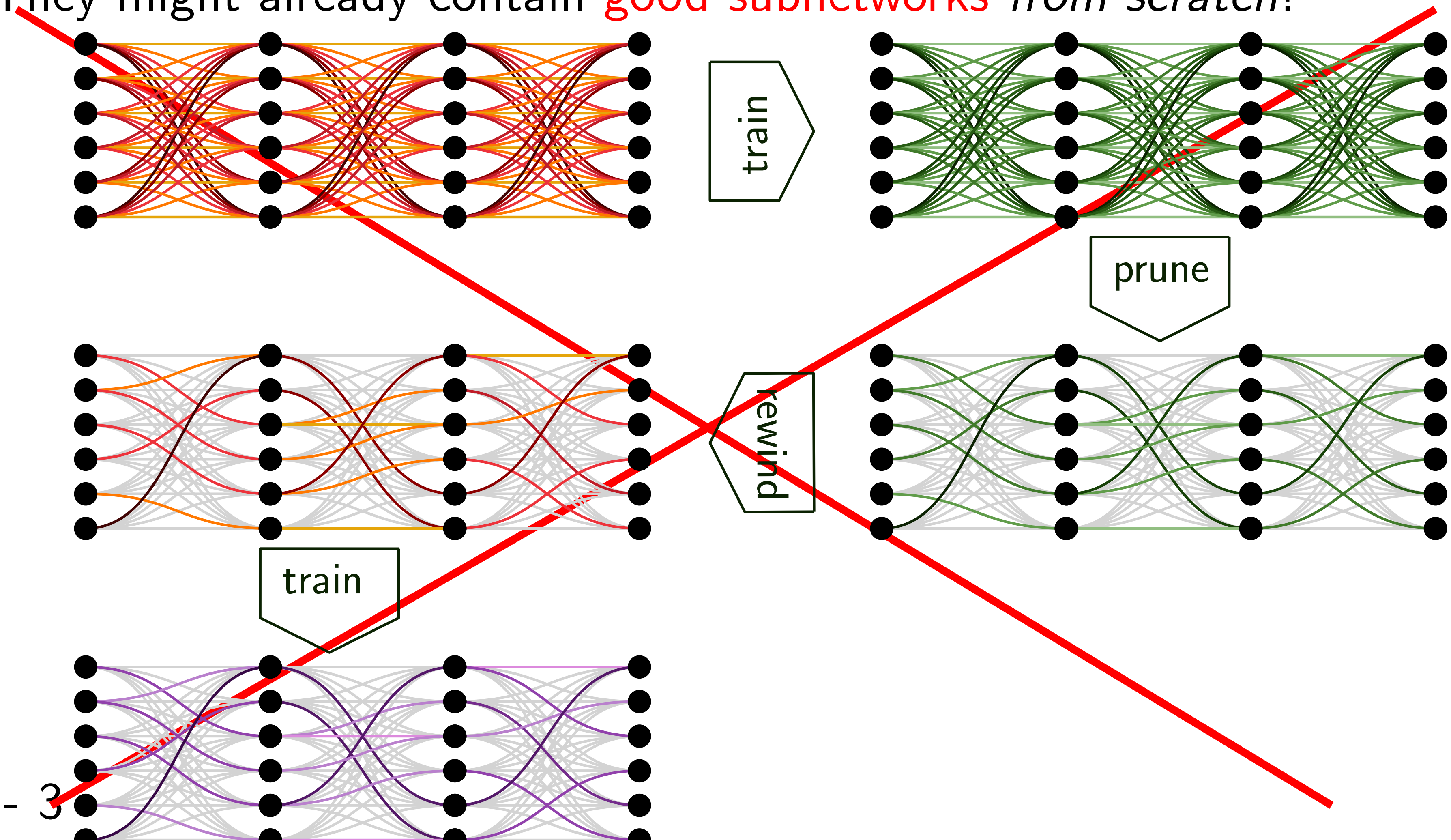
Intuition

- Do we really need to train any parameters? Image we start with a incredibly large, and random network
- They might already contain good subnetworks *from scratch*!

The *Strong* Lottery Ticket Hypothesis (SLTH)

Intuition

- Do we really need to train any parameters? Image we start with a incredibly large, and random network
- They might already contain good subnetworks *from scratch*!

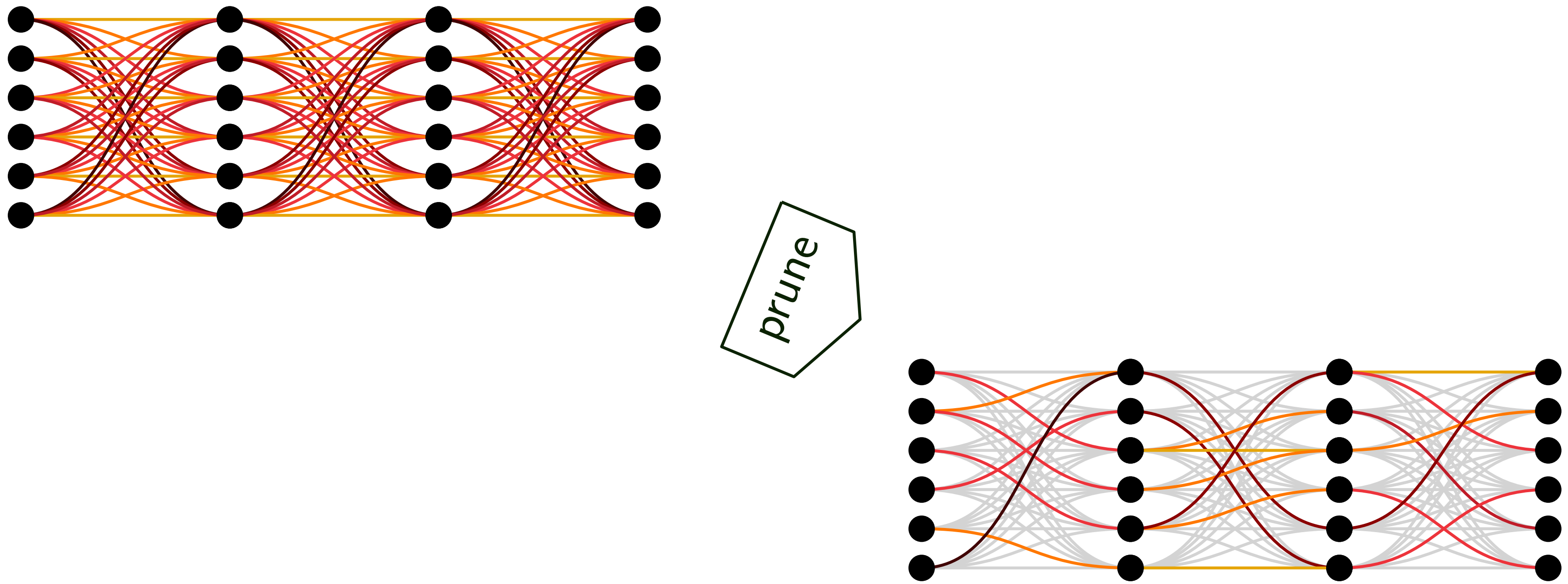


The *Strong* Lottery Ticket Hypothesis (SLTH)

Intuition

- Do we really need to train any parameters? Image we start with a incredibly large, and random network
- They might already contain good subnetworks *from scratch*!

Learn by pruning

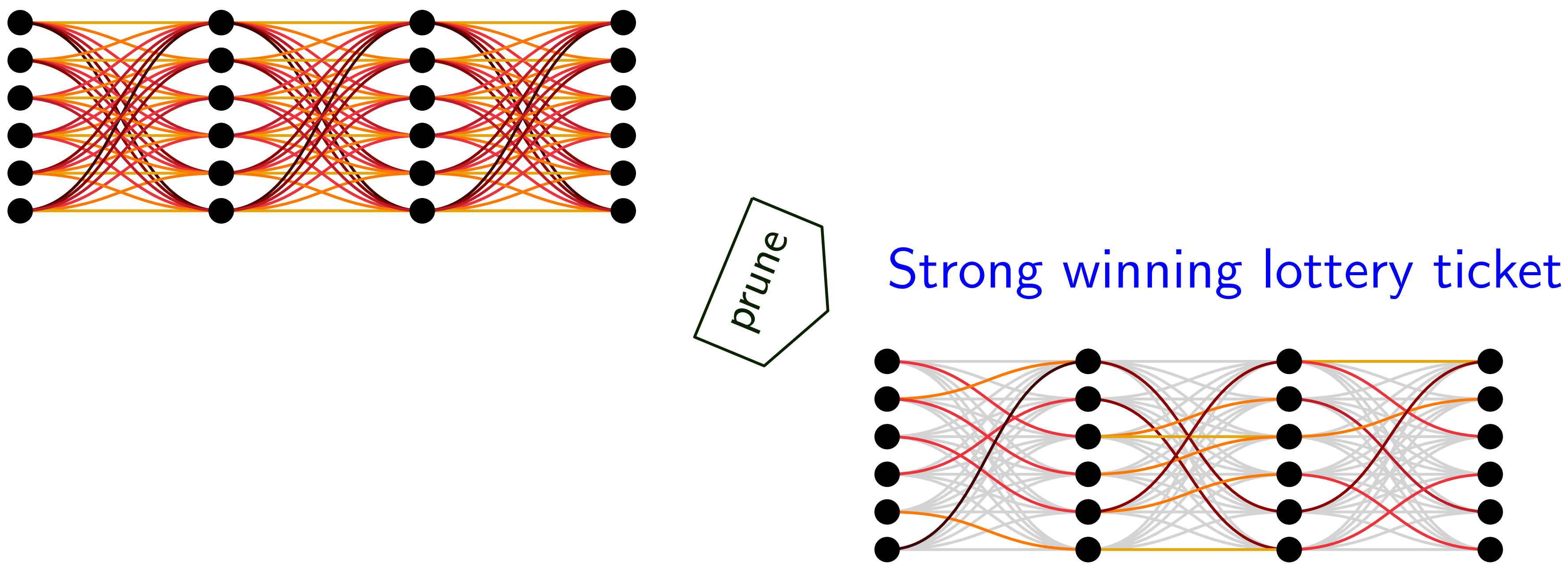


The *Strong* Lottery Ticket Hypothesis (SLTH)

Intuition

- Do we really need to train any parameters? Image we start with a incredibly large, and random network
- They might already contain good subnetworks *from scratch*!

Learn by pruning



The *Strong* Lottery Ticket Hypothesis (SLTH)

SLTH: A network with random weights contains, with high probability, sub-networks that can approximate any given sufficiently-smaller neural network. [Ramanujan et al. 2020, CVPR]

The *Strong* Lottery Ticket Hypothesis (SLTH)

SLTH: A network with random weights contains, with high probability, sub-networks that can approximate any given sufficiently-smaller neural network. [Ramanujan et al. 2020, CVPR]

[Zhou et al. 2019, NeurIPS]: proposes a way to find f: prune weights according to some probability learned through stochastic gradient descent

The *Strong* Lottery Ticket Hypothesis (SLTH)

SLTH: A network with random weights contains, with high probability, sub-networks that can approximate any given sufficiently-smaller neural network. [Ramanujan et al. 2020, CVPR]

[Zhou et al. 2019, NeurIPS]: proposes a way to find f: prune weights according to some probability learned through stochastic gradient descent

- Decent accuracy

The *Strong* Lottery Ticket Hypothesis (SLTH)

SLTH: A network with random weights contains, with high probability, sub-networks that can approximate any given sufficiently-smaller neural network. [Ramanujan et al. 2020, CVPR]

[Zhou et al. 2019, NeurIPS]: proposes a way to find f: prune weights according to some probability learned through stochastic gradient descent

- Decent accuracy

[Ramanujan et al. 2020, CVPR] improves on it: random ResNet-50 pruned to match ResNet-34 on ImageNet

The *Strong* Lottery Ticket Hypothesis (SLTH)

SLTH: A network with random weights contains, with high probability, sub-networks that can approximate any given sufficiently-smaller neural network. [Ramanujan et al. 2020, CVPR]

[Zhou et al. 2019, NeurIPS]: proposes a way to find f: prune weights according to some probability learned through stochastic gradient descent

- Decent accuracy

[Ramanujan et al. 2020, CVPR] improves on it: random ResNet-50 pruned to match ResNet-34 on ImageNet

[Diffenderfer and Kailkhura 2021, ICLR]: quantized strong winning lottery tickets in ResNet-50 (binary weights) outperform the original on ImageNet

Do we have a theorem?

Target result: *Let $\mathcal{F}$ be the class of neural networks with a given size. If a network g with random weights is sufficiently large, then, with high probability, it is possible to prune g to approximate any network in $\mathcal{F}$*

Do we have a theorem?

Target result: *Let $\mathcal{F}$ be the class of neural networks with a given size. If a network g with random weights is sufficiently large, then, with high probability, it is possible to prune g to approximate any network in $\mathcal{F}$*

- Size: parameter count and depth

Do we have a theorem?

Target result: *Let $\mathcal{F}$ be the class of neural networks with a given size. If a network g with random weights is sufficiently large, then, with high probability, it is possible to prune g to approximate any network in $\mathcal{F}$*

- Size: parameter count and depth
- With high probability: $1 - \delta$ for any given $\delta > 0$

Do we have a theorem?

Target result: *Let $\mathcal{F}$ be the class of neural networks with a given size. If a network g with random weights is sufficiently large, then, with high probability, it is possible to prune g to approximate any network in $\mathcal{F}$*

- Size: parameter count and depth
- With high probability: $1 - \delta$ for any given $\delta > 0$
- Approximation: distance w.r.t. some metric is ε for any given $\varepsilon > 0$

Overview of the (theoretical) results

SLTH holds for:

- [Malach et al. 2020, ICML]: polynomially overparameterized dense networks with ReLU activation functions

Overview of the (theoretical) results

SLTH holds for:

- [Malach et al. 2020, ICML]: polynomially overparameterized dense networks with ReLU activation functions
- [Pensia et al. 2020, NeurIPS]: logarithmically overparameterized dense networks with ReLU activation functions

Overview of the (theoretical) results

SLTH holds for:

- [Malach et al. 2020, ICML]: polynomially overparameterized dense networks with ReLU activation functions
- [Pensia et al. 2020, NeurIPS]: logarithmically overparameterized dense networks with ReLU activation functions
- [Diffenderfer and Kailkhura 2021, ICLR]: polynomially overparameterized binary dense networks

Overview of the (theoretical) results

SLTH holds for:

- [Malach et al. 2020, ICML]: polynomially overparameterized dense networks with ReLU activation functions
- [Pensia et al. 2020, NeurIPS]: logarithmically overparameterized dense networks with ReLU activation functions
- [Diffenderfer and Kailkhura 2021, ICLR]: polynomially overparameterized binary dense networks
- [Sreenivasan et al. 2022, AIStat]: polylogarithmically overparameterized binary dense networks

Overview of the (theoretical) results

SLTH holds for:

- [Malach et al. 2020, ICML]: polynomially overparameterized dense networks with ReLU activation functions
- [Pensia et al. 2020, NeurIPS]: logarithmically overparameterized dense networks with ReLU activation functions
- [Diffenderfer and Kailkhura 2021, ICLR]: polynomially overparameterized binary dense networks
- [Sreenivasan et al. 2022, AIStat]: polylogarithmically overparameterized binary dense networks
- [da Cunha et al. 2022, ICLR]: logarithmically overparameterized convolutional neural networks (CNNs) with ReLU activation functions and non-negative inputs

Overview of the (theoretical) results

SLTH holds for:

- [Malach et al. 2020, ICML]: polynomially overparameterized dense networks with ReLU activation functions
- [Pensia et al. 2020, NeurIPS]: logarithmically overparameterized dense networks with ReLU activation functions
- [Diffenderfer and Kailkhura 2021, ICLR]: polynomially overparameterized binary dense networks
- [Sreenivasan et al. 2022, AIStat]: polylogarithmically overparameterized binary dense networks
- [da Cunha et al. 2022, ICLR]: logarithmically overparameterized convolutional neural networks (CNNs) with ReLU activation functions and non-negative inputs
- [Burkholz 2022a,b, NeurIPS, ICML]: logarithmically overparameterized dense networks, CNNs, and residual architectures with a wider class of activation functions and less depth overhead

Overview of the (theoretical) results

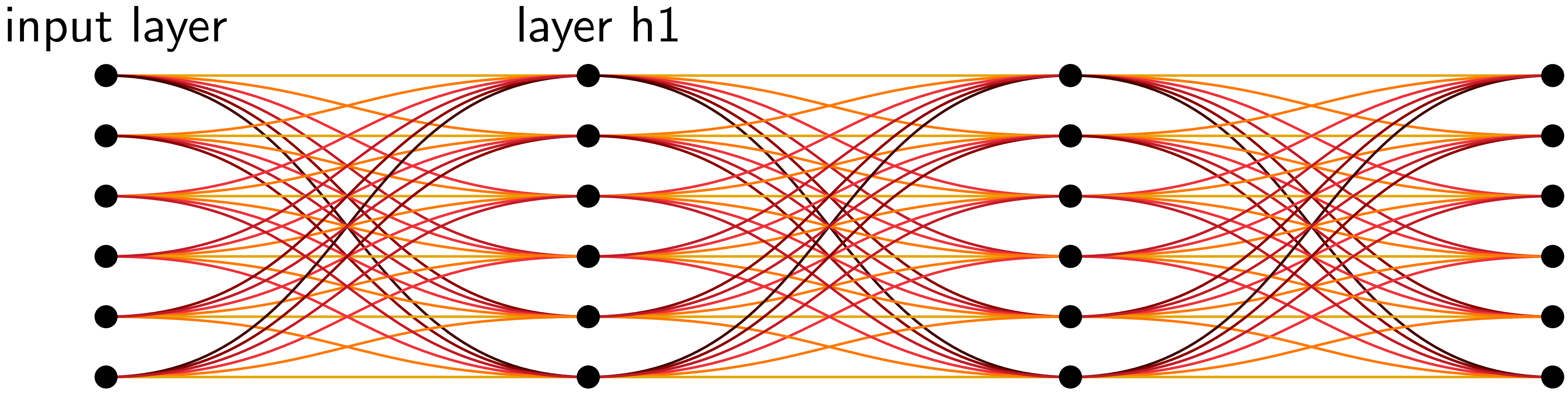
SLTH holds for:

- [Malach et al. 2020, ICML]: polynomially overparameterized dense networks with ReLU activation functions
- [Pensia et al. 2020, NeurIPS]: logarithmically overparameterized dense networks with ReLU activation functions
- [Diffenderfer and Kailkhura 2021, ICLR]: polynomially overparameterized binary dense networks
- [Sreenivasan et al. 2022, AIStat]: polylogarithmically overparameterized binary dense networks
- [da Cunha et al. 2022, ICLR]: logarithmically overparameterized convolutional neural networks (CNNs) with ReLU activation functions and non-negative inputs
- [Burkholz 2022a,b, NeurIPS, ICML]: logarithmically overparameterized dense networks, CNNs, and residual architectures with a wider class of activation functions and less depth overhead
- [Ferbach et al. 2022, ICLR]: logarithmically overparameterized equivariant networks with ReLU activation functions

Review: Feedforward

Dense network: $f(\mathbf{x}) = \mathbf{W}_\ell \sigma(\mathbf{W}_{\ell-1} \ldots \sigma(\mathbf{W}_1 \mathbf{x}))$

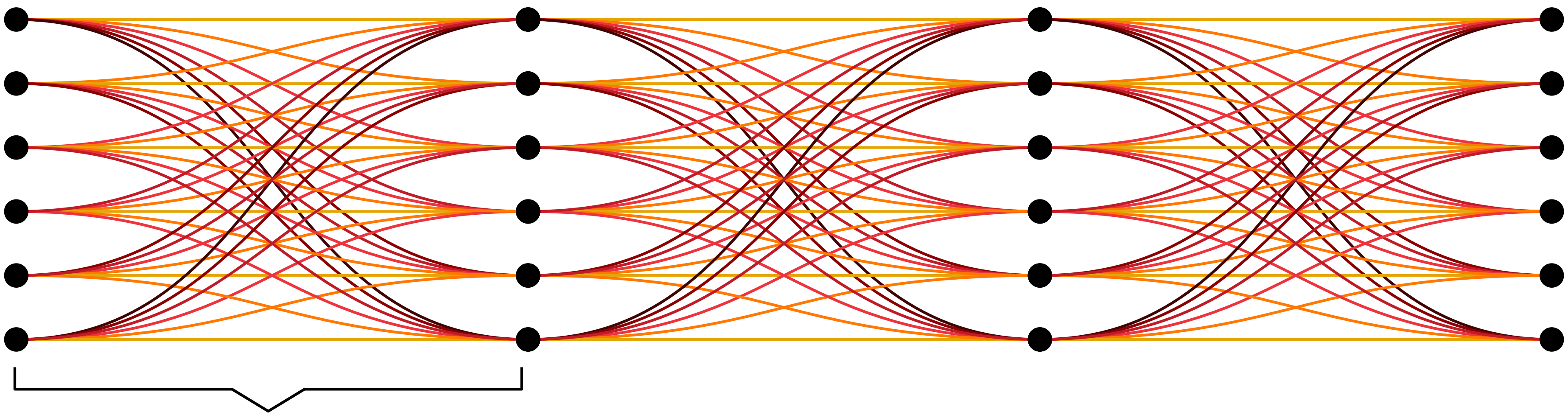
- $\mathbf{x} \in \mathbb{R}^{d_0}$, $\mathbf{W}_i \in \mathbb{R}^{d_{i-1} \times d_i}$
- $\sigma(x) = \max(0, x)$ (ReLU)

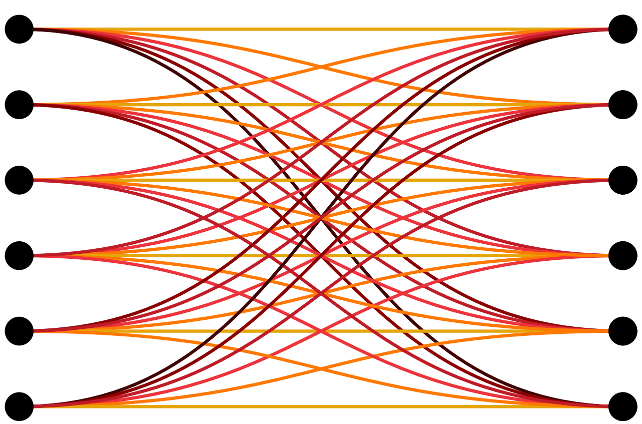


Review: Feedforward

Dense network: $f(\mathbf{x}) = \mathbf{W}_\ell \sigma(\mathbf{W}_{\ell-1} \ldots \sigma(\mathbf{W}_1 \mathbf{x}))$

- $\mathbf{x} \in \mathbb{R}^{d_0}$, $\mathbf{W}_i \in \mathbb{R}^{d_{i-1} \times d_i}$
- $\sigma(x) = \max(0, x)$ (ReLU)

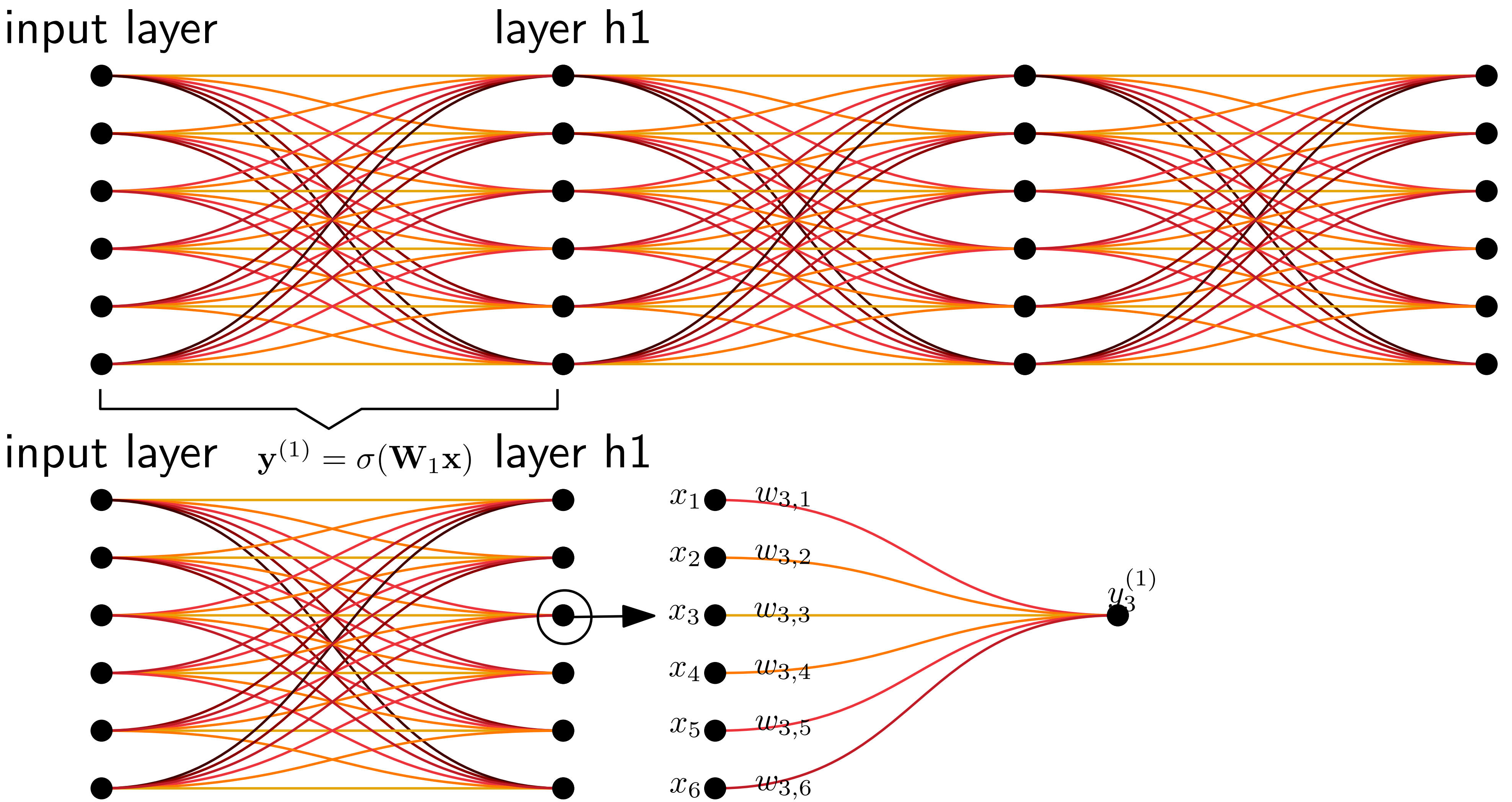




Review: Feedforward

Dense network: $f(\mathbf{x}) = \mathbf{W}_\ell \sigma(\mathbf{W}_{\ell-1} \dots \sigma(\mathbf{W}_1 \mathbf{x}))$

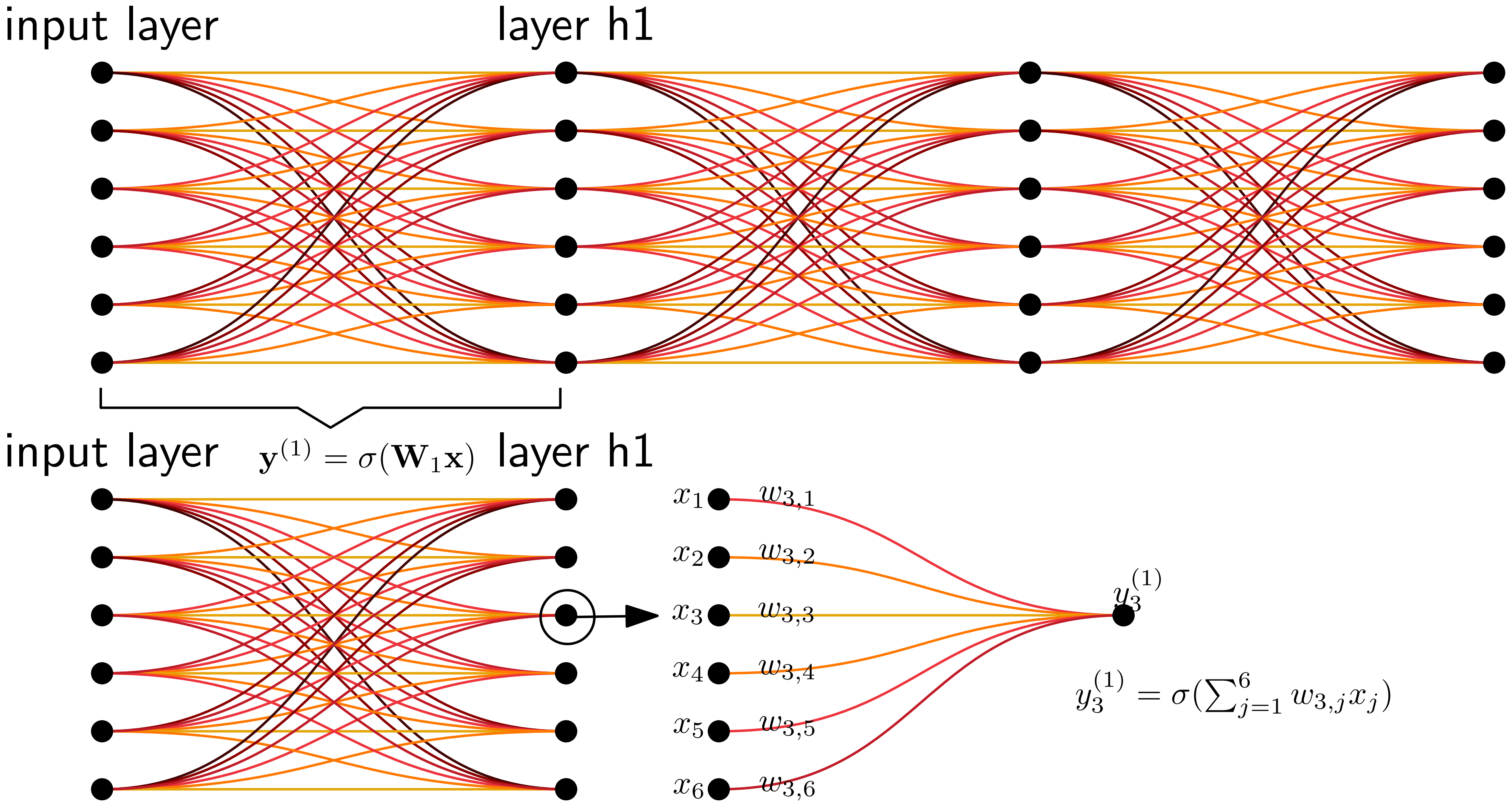
- $\mathbf{x} \in \mathbb{R}^{d_0}$, $\mathbf{W}_i \in \mathbb{R}^{d_{i-1} \times d_i}$
- $\sigma(x) = \max(0, x)$ (ReLU)



Review: Feedforward

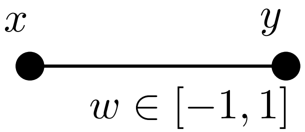
Dense network: $f(\mathbf{x}) = \mathbf{W}_\ell \sigma(\mathbf{W}_{\ell-1} \ldots \sigma(\mathbf{W}_1 \mathbf{x}))$

- $\mathbf{x} \in \mathbb{R}^{d_0}$, $\mathbf{W}_i \in \mathbb{R}^{d_{i-1} \times d_i}$
- $\sigma(x) = \max(0, x)$ (ReLU)



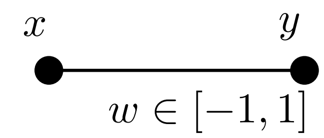
Review: SLTH in dense networks

- Approx one edge: approx $y = wx$ for all x within error ε (no ReLU)



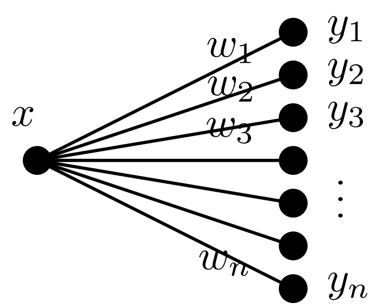
Review: SLTH in dense networks

- Approx one edge: approx $y = wx$ for all x within error ε (no ReLU)



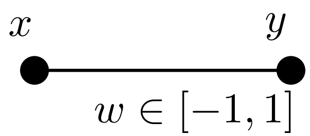
  - Naïve approach

    sample many weights $w_i \sim$ Unif$[-1, 1]$
    until getting w, and prune the others



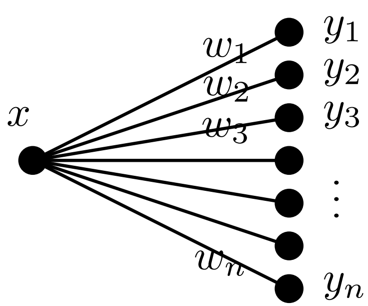
Review: SLTH in dense networks

- Approx one edge: approx $y = wx$ for all x within error ε (no ReLU)



  - Naïve approach

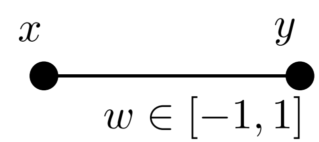
    sample many weights $w_i \sim$ Unif$[-1, 1]$
    until getting w, and prune the others



    roughly $1/\varepsilon$ samples

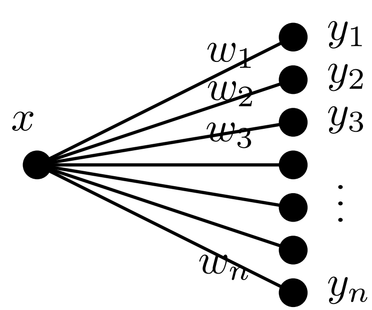
Review: SLTH in dense networks

- Approx one edge: approx $y = wx$ for all x within error ε (no ReLU)



- Naïve approach

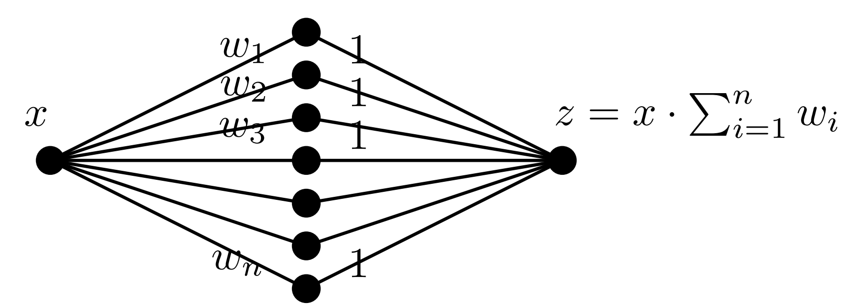
  sample many weights $w_i \sim$ Unif$[-1, 1]$
  until getting w, and prune the others



  roughly $1/\varepsilon$ samples

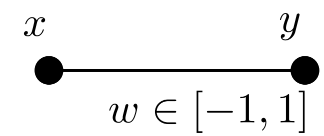
- Random subset sum (RSS) approach:

  add intermediate layer, sample
  $w_i \sim$ Unif$[-1, 1]$ and find a good subset



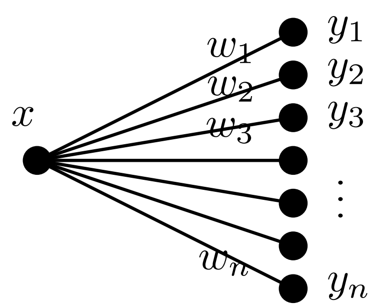
Review: SLTH in dense networks

- Approx one edge: approx $y = wx$ for all x within error ε (no ReLU)



- Naïve approach

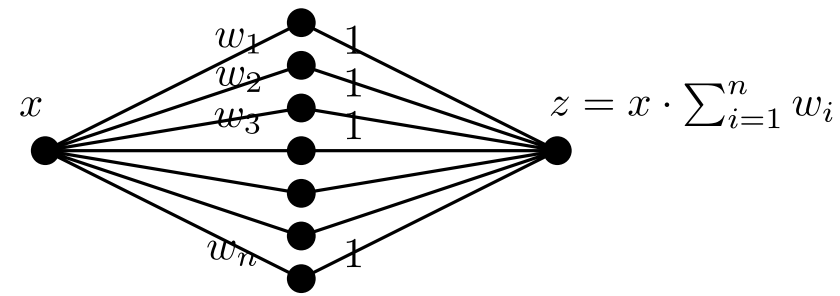
  sample many weights $w_i \sim$ Unif$[-1, 1]$
  until getting w, and prune the others



  roughly $1/\varepsilon$ samples

- Random subset sum (RSS) approach:

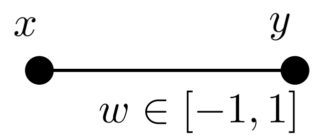
  add intermediate layer, sample
  $w_i \sim$ Unif$[-1, 1]$ and find a good subset



Say $w = 0.5$, $w_1 = 0.6$, $w_2 = -0.1$, ...

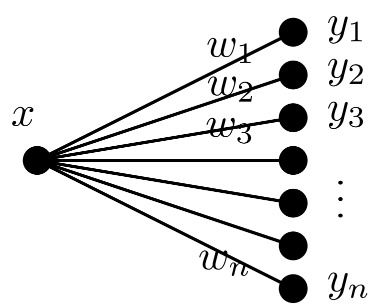
Review: SLTH in dense networks

- Approx one edge: approx $y = wx$ for all x within error ε (no ReLU)



- Naïve approach

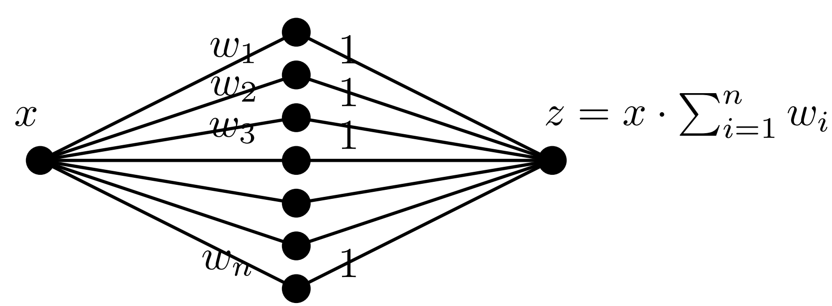
  sample many weights $w_i \sim$ Unif$[-1, 1]$
  until getting w, and prune the others



  roughly $1/\varepsilon$ samples

- Random subset sum (RSS) approach:

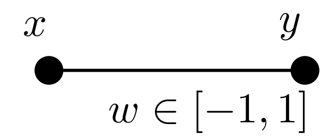
  add intermediate layer, sample
  $w_i \sim$ Unif$[-1, 1]$ and find a good subset



  How many?

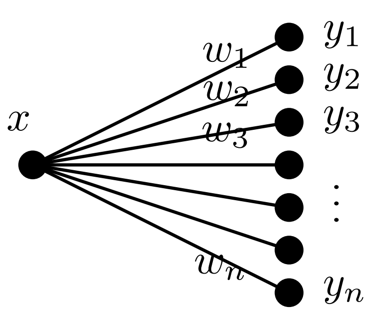
Review: SLTH in dense networks

- Approx one edge: approx $y = wx$ for all x within error ε (no ReLU)



- Naïve approach

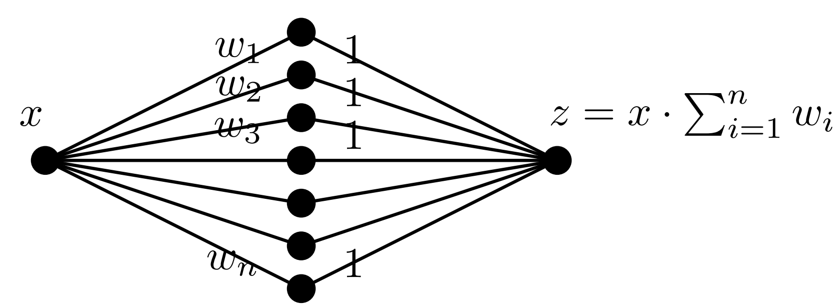
  sample many weights $w_i \sim \text{Unif}[-1, 1]$ until getting w, and prune the others



  roughly $1/\varepsilon$ samples

- Random subset sum (RSS) approach:

  add intermediate layer, sample $w_i \sim \text{Unif}[-1, 1]$ and find a good subset

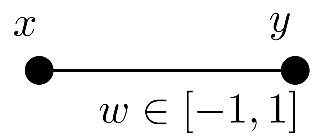


  How many?

Theorem [Lueker 1998; da Cunha et al. 2023, ESA]: *Let $x_1, \ldots, x_n \in [-1, 1]$ be i.i.d. uniform random variables. Given any error parameter $\varepsilon > 0$, there exists a constant $C > 0$ such that if $n \geq C \log 1/\varepsilon$ then, with probability $1 - \exp\left[(n - C \log 1/\varepsilon)^2 / 4n\right]$, for each $z \in [-1, 1]$ there exists a subset $S \subseteq [n]$ such that $\left|z - \sum_{i \in S} x_i\right| < 2\varepsilon$*

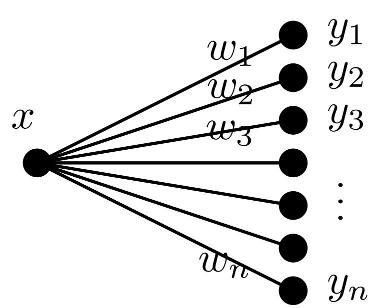
Review: SLTH in dense networks

- Approx one edge: approx $y = wx$ for all x within error ε (no ReLU)



- Naïve approach

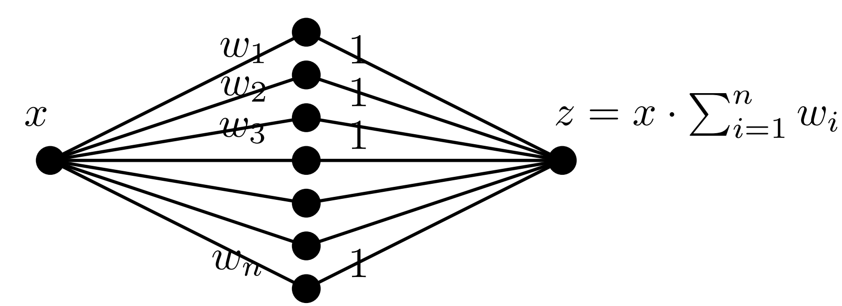
  sample many weights $w_i \sim$ Unif$[-1, 1]$ until getting w, and prune the others



  roughly $1/\varepsilon$ samples

- Random subset sum (RSS) approach:

  add intermediate layer, sample $w_i \sim$ Unif$[-1, 1]$ and find a good subset



  How many?

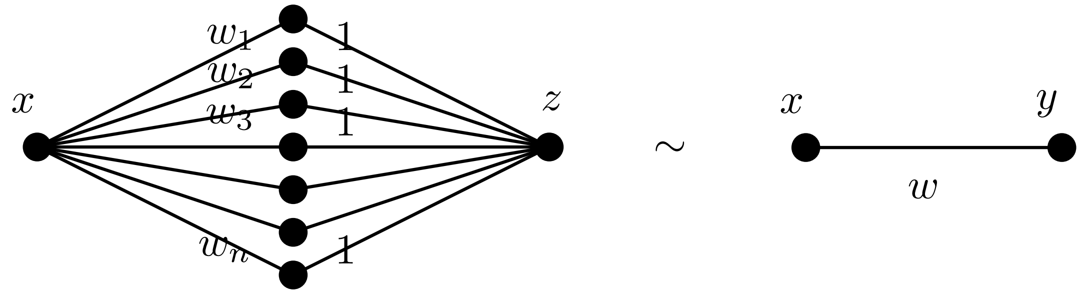
Theorem [Lueker 1998; da Cunha et al. 2023, ESA]: *Let $x_1, \ldots, x_n \in [-1, 1]$ be i.i.d. uniform random variables. Given any error parameter $\varepsilon > 0$, there exists a constant $C > 0$ such that if $n \geq C \log 1/\varepsilon$ then, with probability $1 - \exp\left[(n - C \log 1/\varepsilon)^2 / 4n\right]$, for each $z \in [-1, 1]$ there exists a subset $S \subseteq [n]$ such that $\left|z - \sum_{i \in S} x_i\right| < 2\varepsilon$*

works for all densities $h(x) = pf(x) + (1 - p)g(x)$, where f is "uniform"

Apply RSS for pruning

- Random subset sum (RSS) approach:

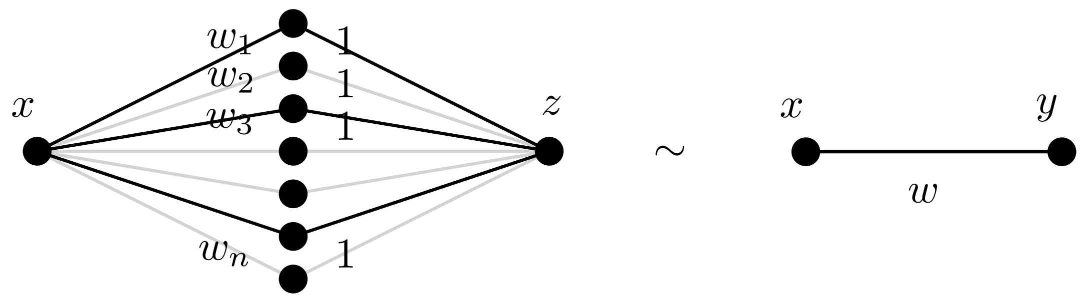
  add intermediate layer, sample $w_i \sim \text{Unif}[-1, 1]$ and find a good subset



Apply RSS for pruning

- Random subset sum (RSS) approach:

  add intermediate layer, sample $w_i \sim$ Unif$[-1, 1]$ and find a good subset

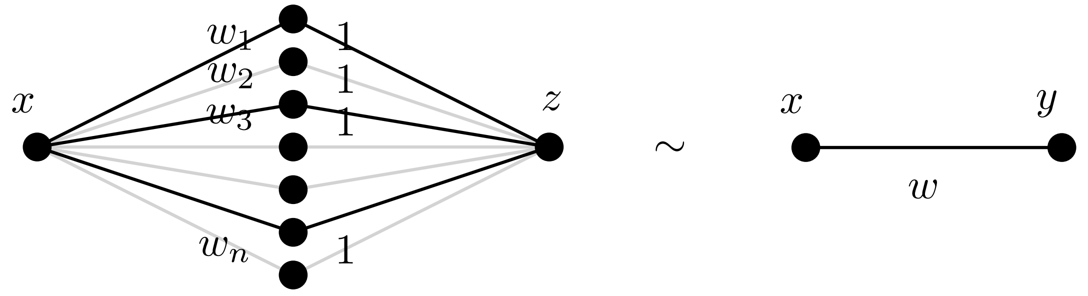


$n \geq C \log 1/\varepsilon \implies \exists S \subseteq [n] : \left|w - \sum_{i \in S} w_i\right| < 2\varepsilon$

Apply RSS for pruning

- Random subset sum (RSS) approach:

  add intermediate layer, sample $w_i \sim$ Unif$[-1, 1]$ and find a good subset



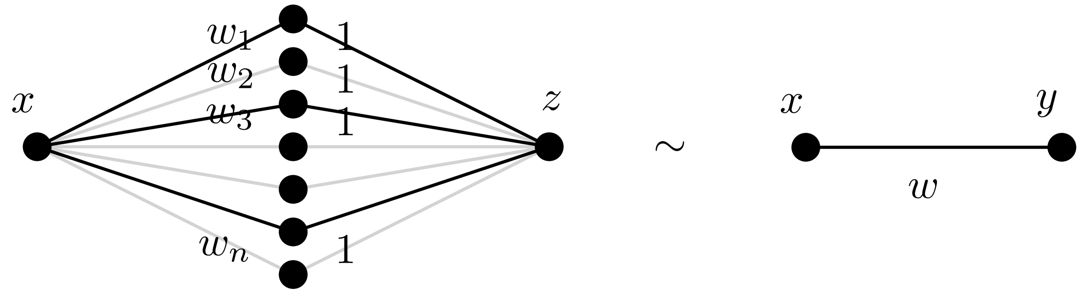
$n \geq C \log 1/\varepsilon \implies \exists S \subseteq [n] : \left| w - \sum_{i \in S} w_i \right| < 2\varepsilon$

$\implies \left| wx - \sum_{i \in S} w_i x \right| \leq |x| \left| w - \sum_{i \in S} w_i \right| < 2\varepsilon |x|$

Apply RSS for pruning

- Random subset sum (RSS) approach:

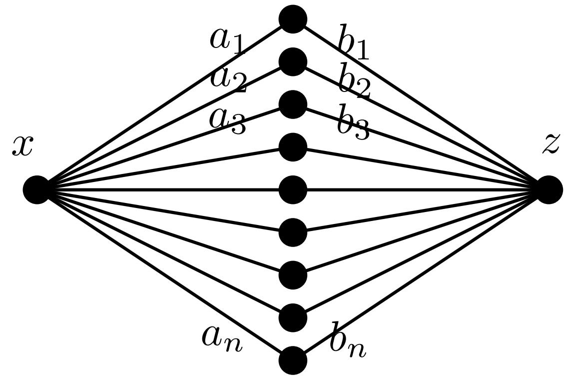
  add intermediate layer, sample $w_i \sim$ Unif$[-1, 1]$ and find a good subset



$$n \geq C \log 1/\varepsilon \implies \exists S \subseteq [n] : \left|w - \sum_{i \in S} w_i\right| < 2\varepsilon$$

$$\implies \left|wx - \sum_{i \in S} w_i x\right| \leq |x|\left|w - \sum_{i \in S} w_i\right| < 2\varepsilon|x|$$

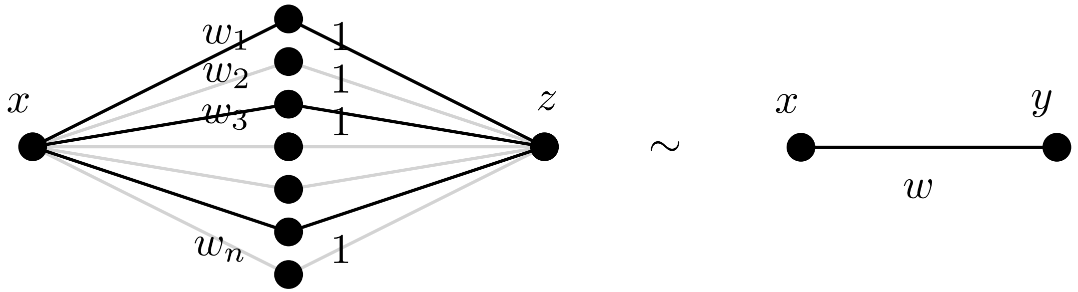
- Completely random initialization + ReLU (non-linearity):



Apply RSS for pruning

- Random subset sum (RSS) approach:

  add intermediate layer, sample $w_i \sim \text{Unif}[-1,1]$ and find a good subset

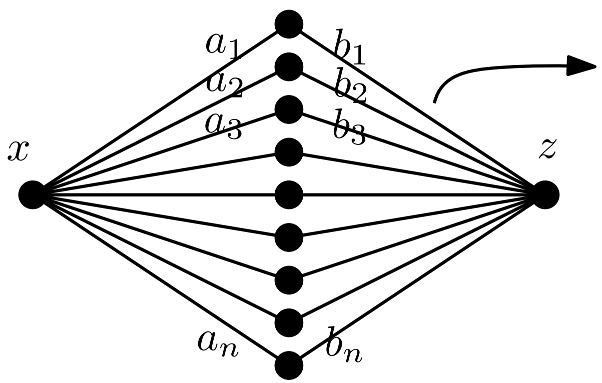


$$n \geq C \log 1/\varepsilon \implies \exists S \subseteq [n] : \left|w - \sum_{i \in S} w_i\right| < 2\varepsilon$$

$$\implies \left|wx - \sum_{i \in S} w_i x\right| \leq |x|\left|w - \sum_{i \in S} w_i\right| < 2\varepsilon|x|$$

- Completely random initialization + ReLU (non-linearity):

  how to deal with non-linearity?



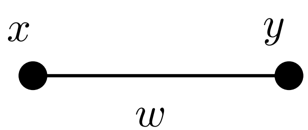
$$\left|wx - \sum_{i \in S} b_i \sigma(a_i x)\right|$$

Exploiting properties of the ReLU

- Completely random initialization + ReLU (non-linearity)

  Property of ReLU: $wx = \sigma(wx) - \sigma(-wx)$

ReLU:
$\sigma(x) = \max(0, x)$

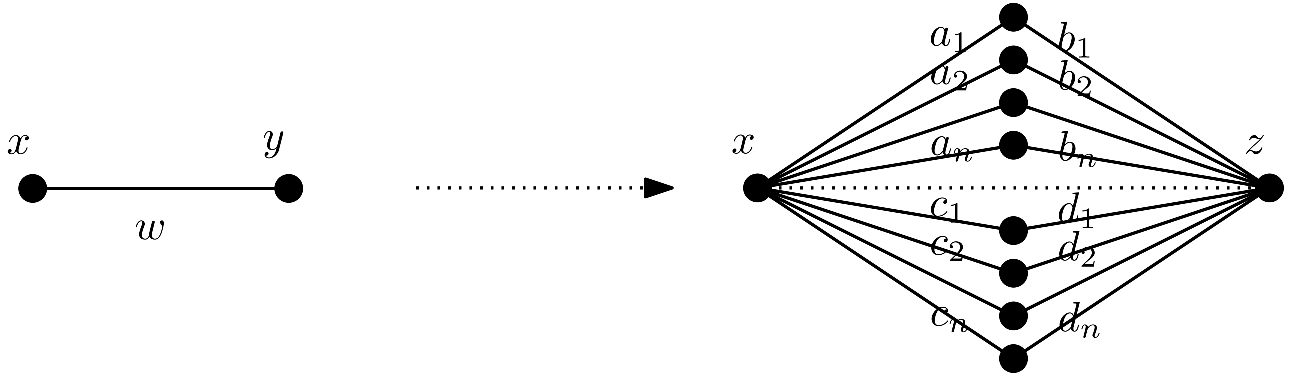


Exploiting properties of the ReLU

- Completely random initialization + ReLU (non-linearity)

  Property of ReLU: $wx = \sigma(wx) - \sigma(-wx)$

ReLU:
$\sigma(x) = \max(0, x)$

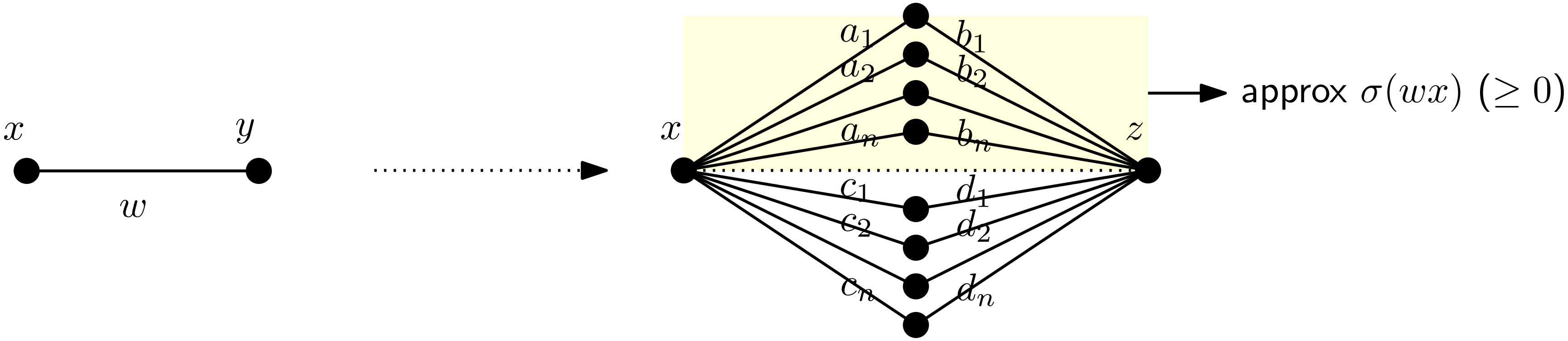


Exploiting properties of the ReLU

- Completely random initialization + ReLU (non-linearity)

  Property of ReLU: $wx = \sigma(wx) - \sigma(-wx)$

ReLU:
$\sigma(x) = \max(0, x)$

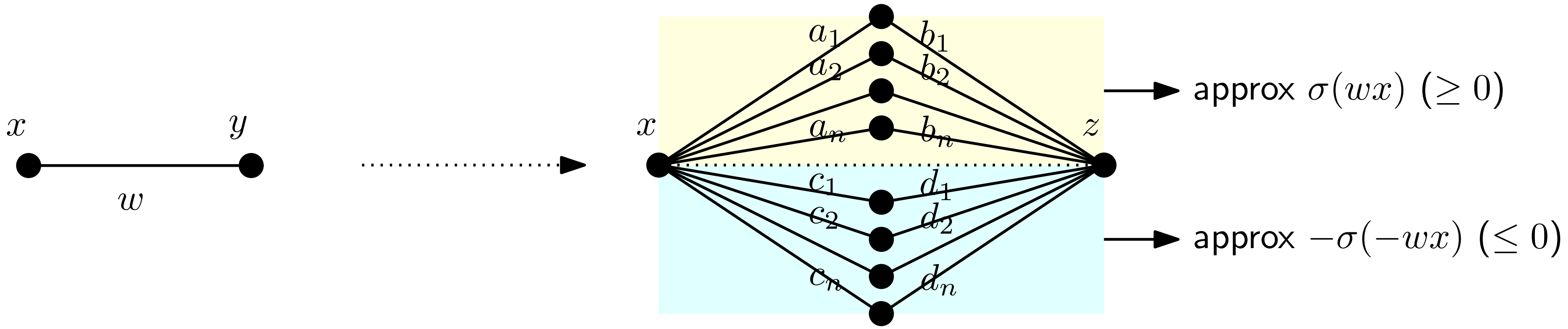


Exploiting properties of the ReLU

- Completely random initialization + ReLU (non-linearity)

  Property of ReLU: $wx = \sigma(wx) - \sigma(-wx)$

ReLU:
$\sigma(x) = \max(0, x)$

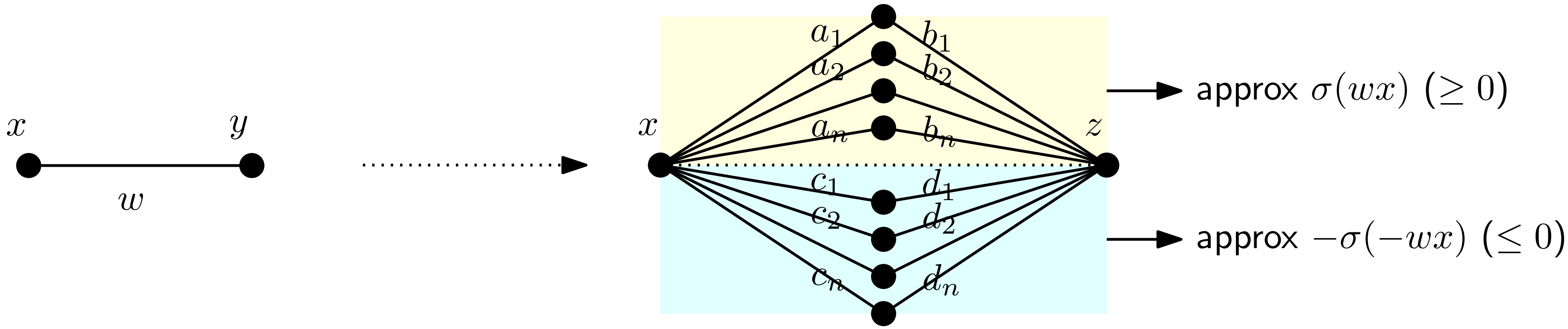


Exploiting properties of the ReLU

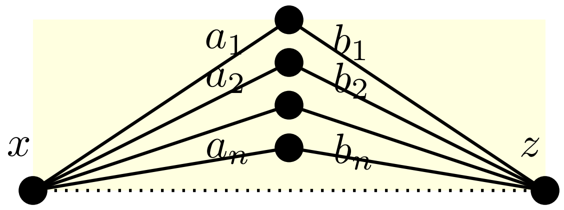
- Completely random initialization + ReLU (non-linearity)

  Property of ReLU: $wx = \sigma(wx) - \sigma(-wx)$

ReLU:
$\sigma(x) = \max(0, x)$



- How? Wlog, assume $w \geq 0$

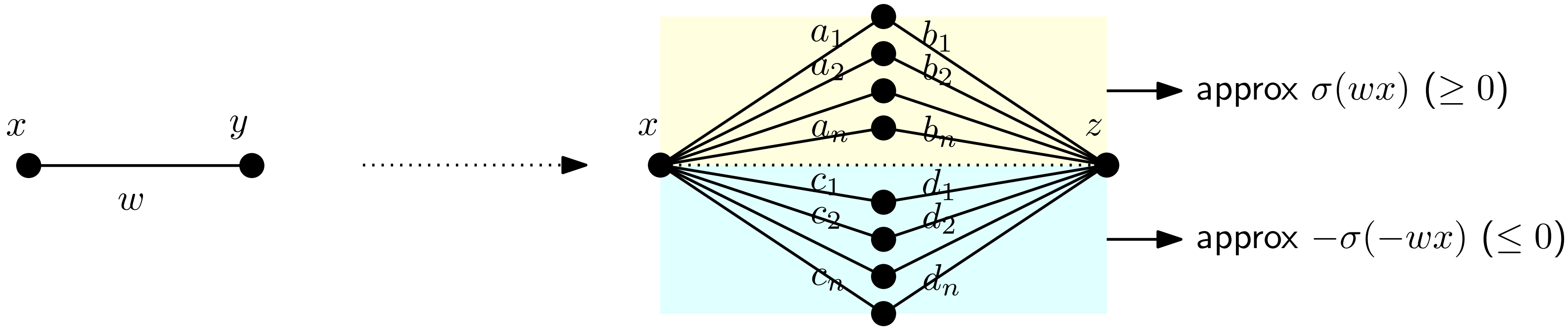


Exploiting properties of the ReLU

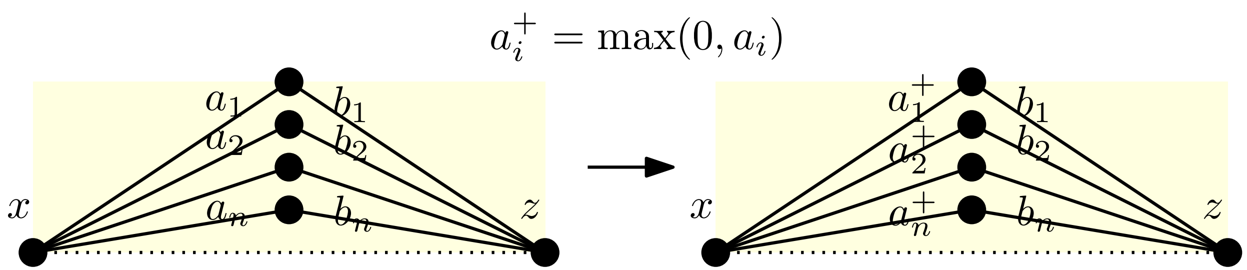
- Completely random initialization + ReLU (non-linearity)

  Property of ReLU: $wx = \sigma(wx) - \sigma(-wx)$

> ReLU:
> $\sigma(x) = \max(0, x)$



- How? Wlog, assume $w \geq 0$

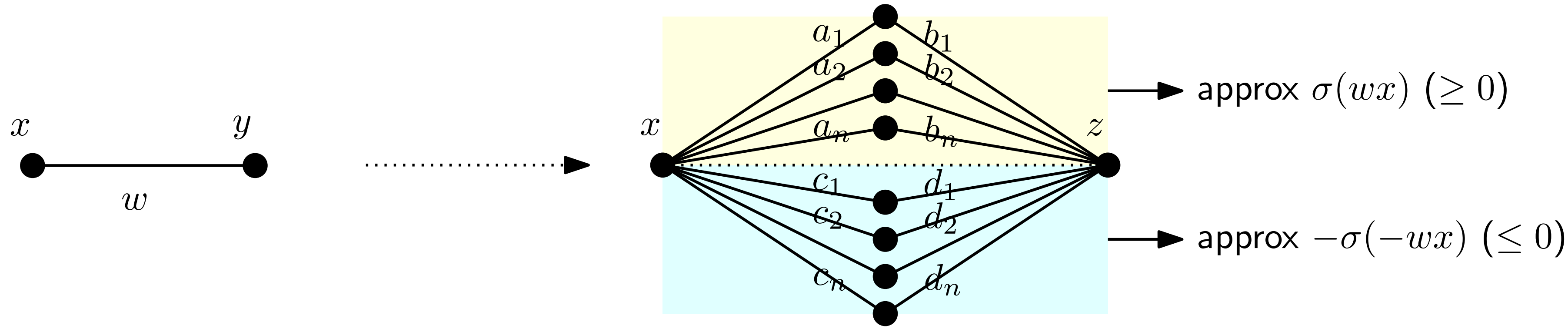


Exploiting properties of the ReLU

ReLU:
$\sigma(x) = \max(0, x)$

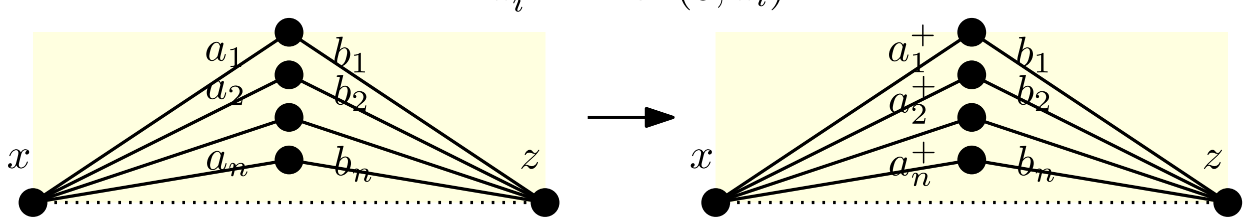
- Completely random initialization + ReLU (non-linearity)

  Property of ReLU: $wx = \sigma(wx) - \sigma(-wx)$



- How? Wlog, assume $w \geq 0$

$a_i^+ = \max(0, a_i)$



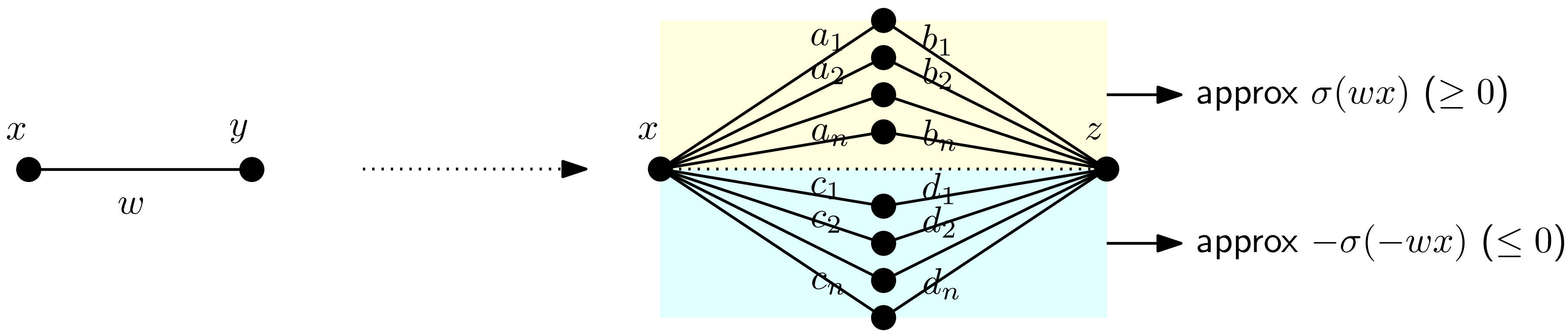
$$\left|\sigma(wx) - \sum_{i\in[n]} b_i \sigma(a_i x)\right| = \left|\sigma(wx) - \sum_{i\in[n]} b_i a_i^+ \sigma(x)\right|$$

Exploiting properties of the ReLU

- Completely random initialization + ReLU (non-linearity)

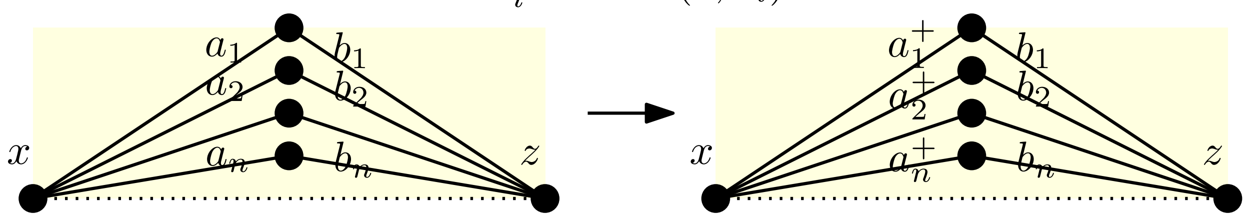
  Property of ReLU: $wx = \sigma(wx) - \sigma(-wx)$

ReLU:
$\sigma(x) = \max(0, x)$



- How? Wlog, assume $w \geq 0$

$a_i^+ = \max(0, a_i)$



$$\left|\sigma(wx) - \sum_{i\in[n]} b_i \sigma(a_i x)\right| = \left|\sigma(wx) - \sum_{i\in[n]} b_i a_i^+ \sigma(x)\right|$$

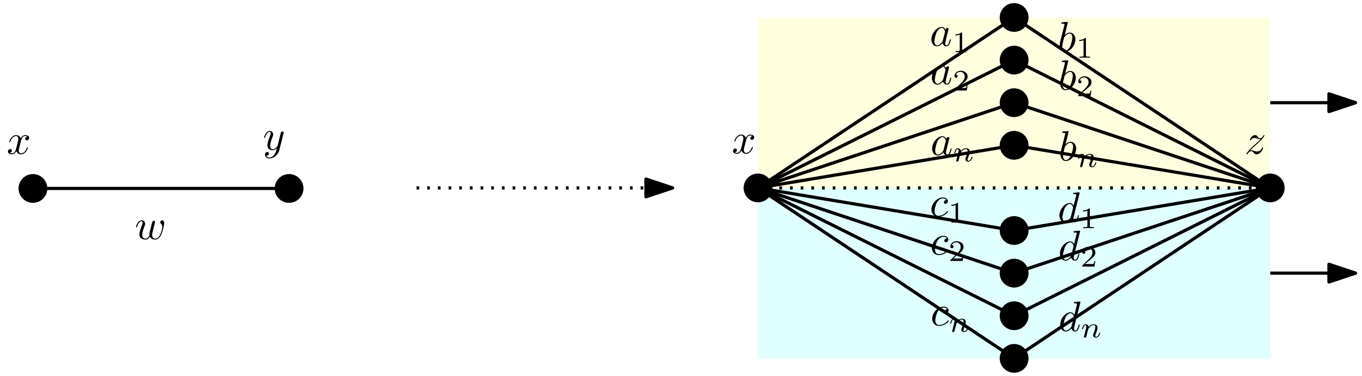
if $x \leq 0$, easy

Exploiting properties of the ReLU

> **ReLU:**
> $\sigma(x) = \max(0, x)$

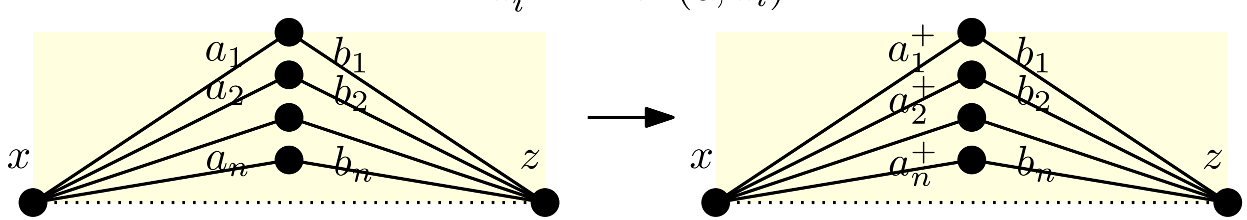
- Completely random initialization + ReLU (non-linearity)

  Property of ReLU: $wx = \sigma(wx) - \sigma(-wx)$



- How? Wlog, assume $w \geq 0$

$a_i^+ = \max(0, a_i)$



$$\left|\sigma(wx) - \sum_{i\in[n]} b_i\sigma(a_i x)\right| = \left|\sigma(wx) - \sum_{i\in[n]} b_i a_i^+ \sigma(x)\right| = x\left|w - \sum_{i\in[n]} b_i a_i^+\right|$$

if $x \leq 0$, easy

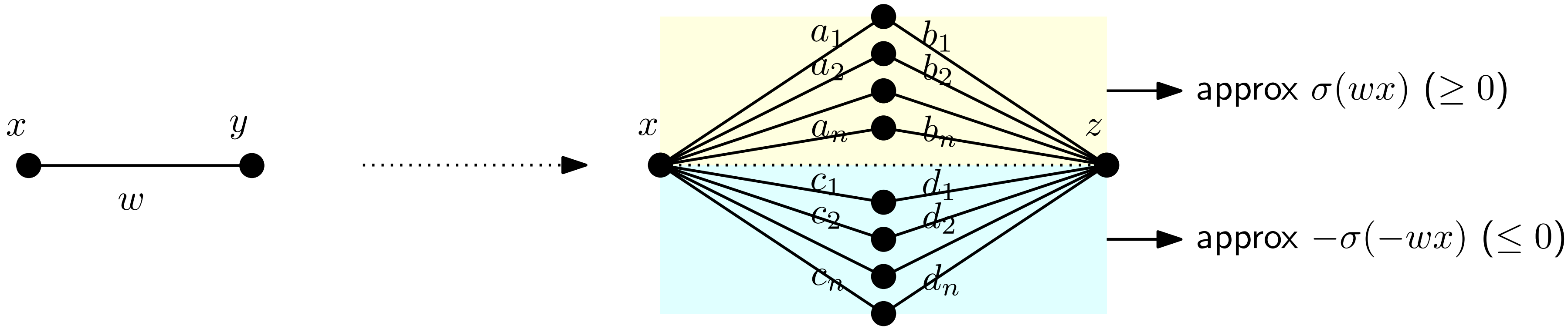
if $x > 0$, then RSS holds!

Exploiting properties of the ReLU

ReLU:
$\sigma(x) = \max(0, x)$

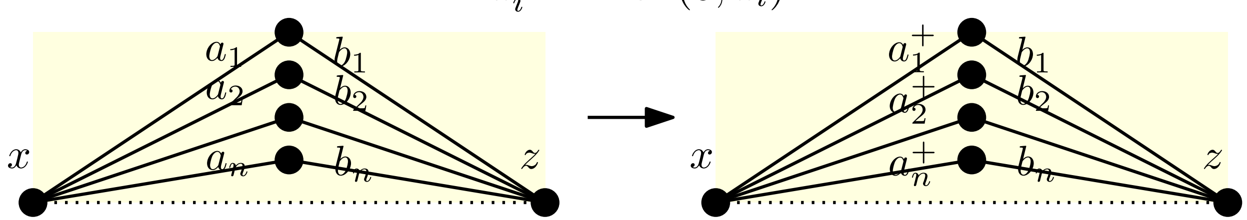
- Completely random initialization + ReLU (non-linearity)

  Property of ReLU: $wx = \sigma(wx) - \sigma(-wx)$



- How? Wlog, assume $w \geq 0$

$a_i^+ = \max(0, a_i)$



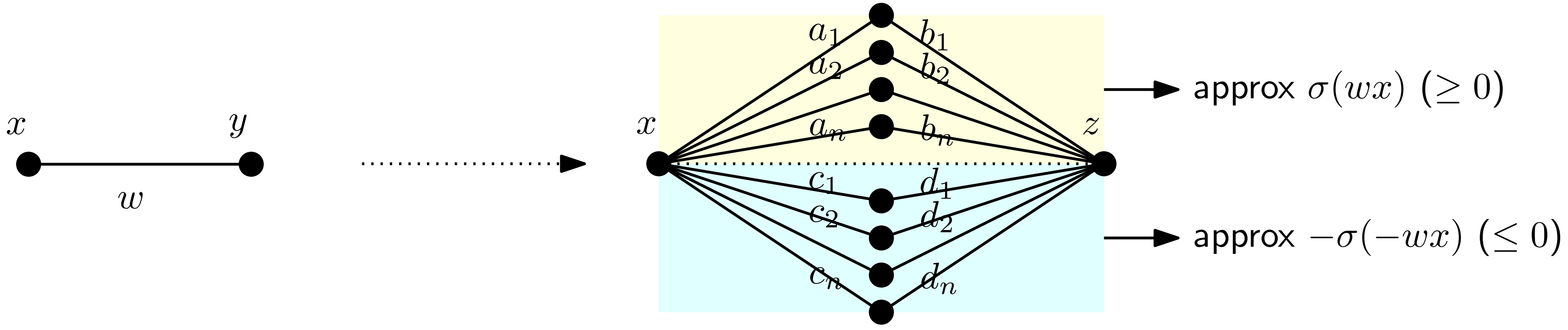
$$\left|\sigma(wx) - \sum_{i\in[n]} b_i\sigma(a_i x)\right| = \left|\sigma(wx) - \sum_{i\in[n]} b_i a_i^+ \sigma(x)\right| = x\left|w - \sum_{i\in[n]} b_i a_i^+\right|$$

if $x \leq 0$, easy

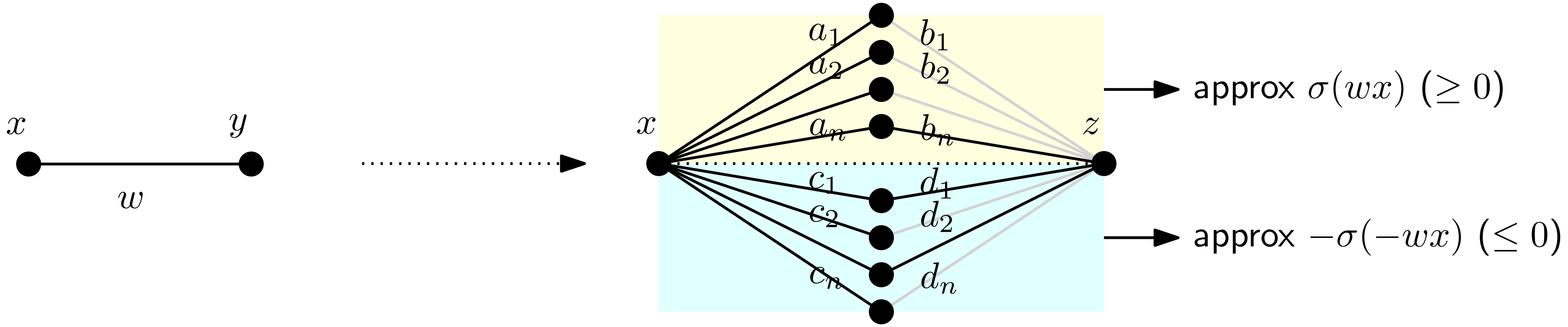
if $x > 0$, then RSS holds!

$n \geq C \log 1/\varepsilon$

Putting everything together

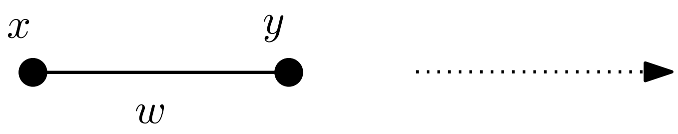


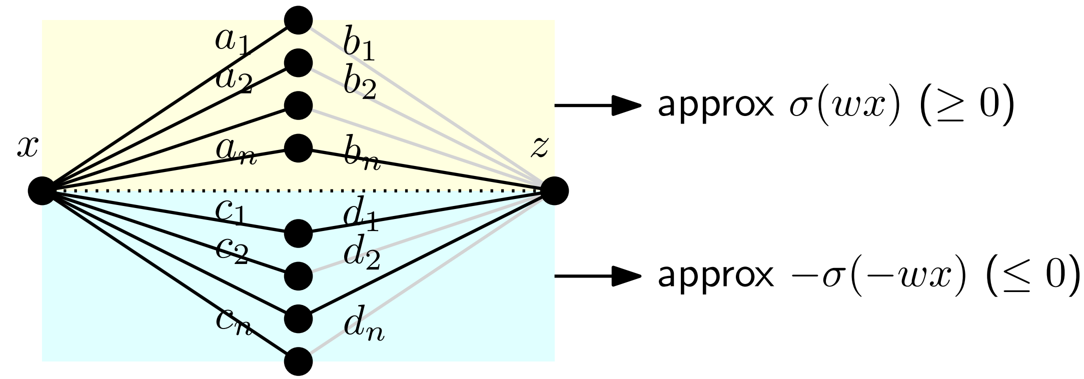
Putting everything together



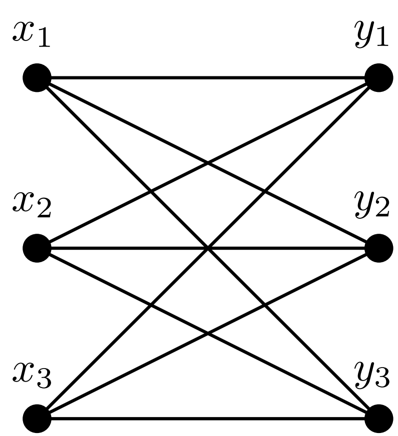
prune only the right layer: reuse the left layer

Putting everything together

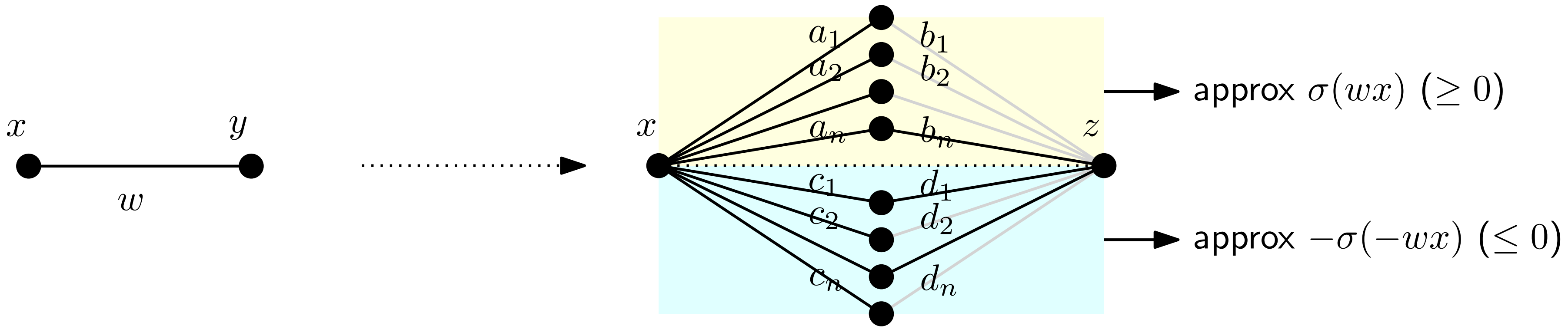




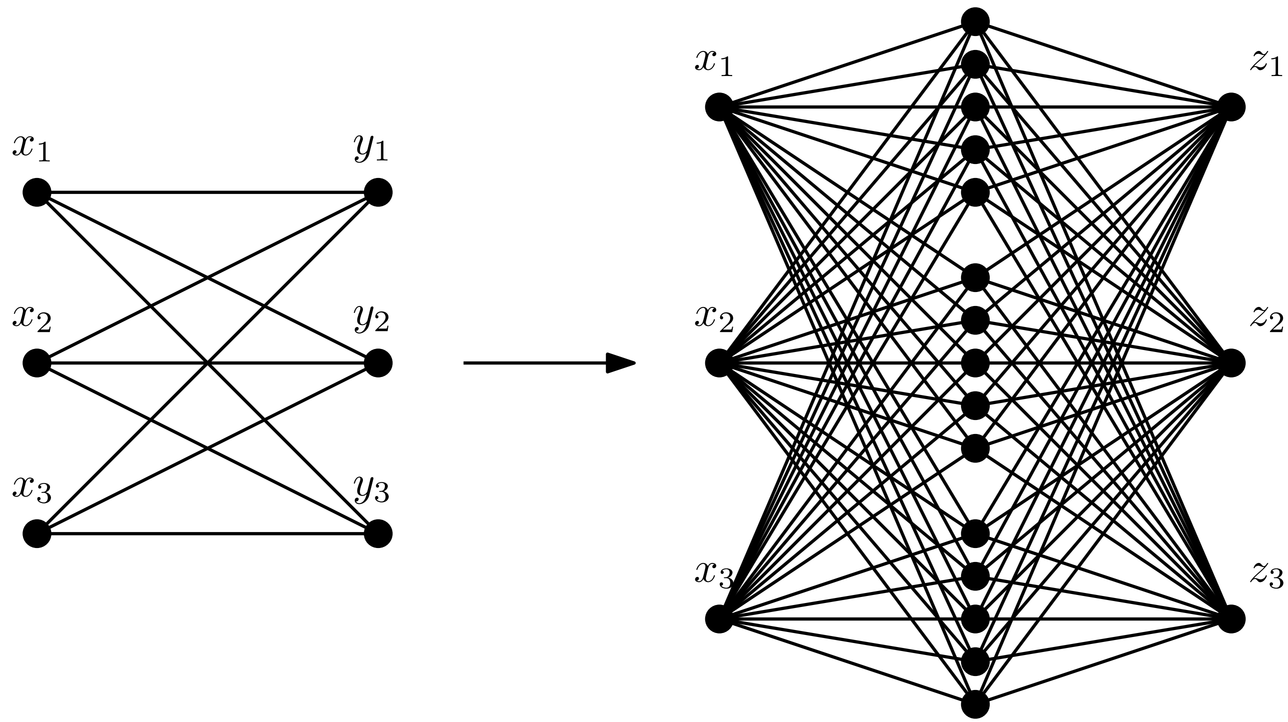
prune only the right layer: reuse the left layer



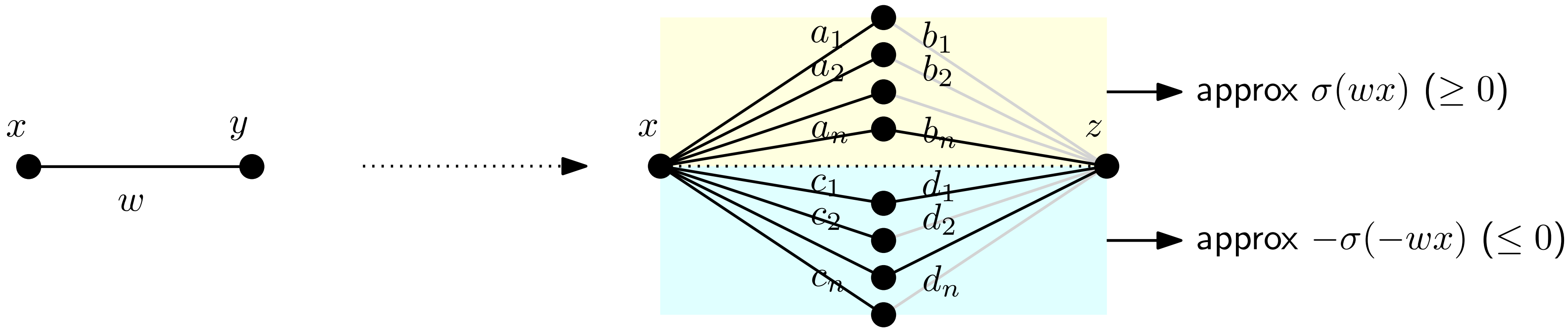
Putting everything together



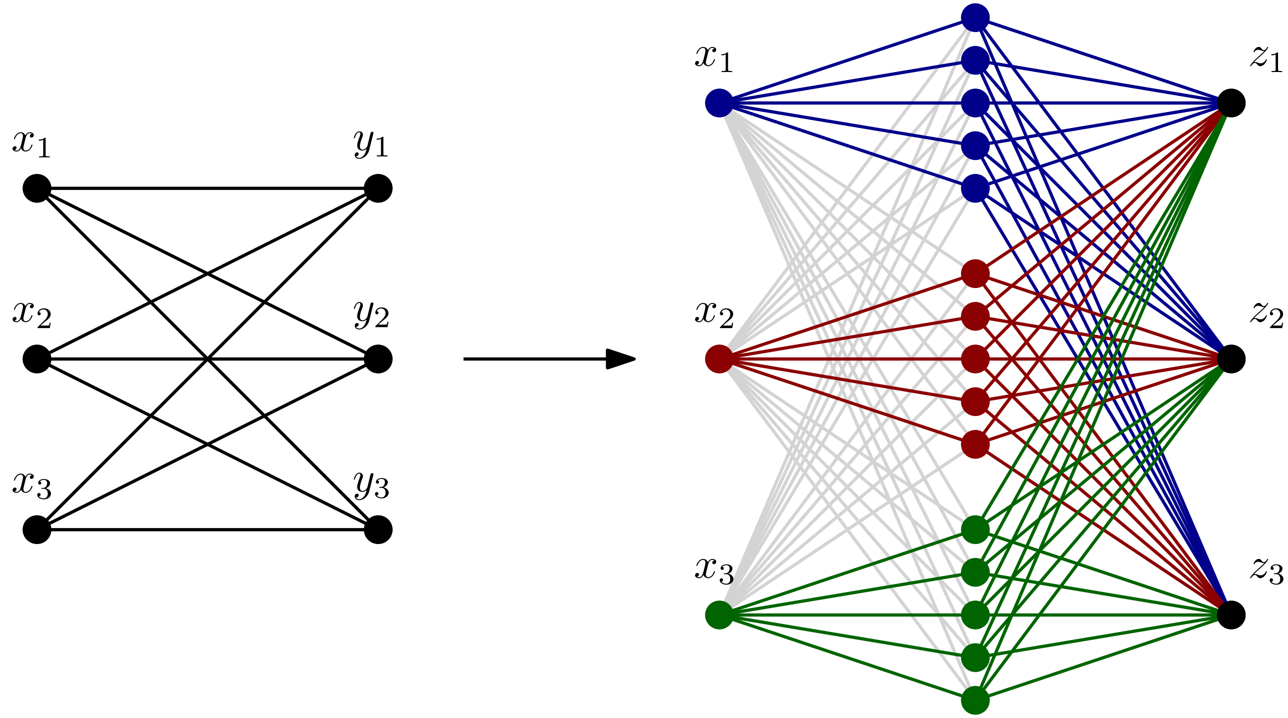
prune only the right layer: reuse the left layer



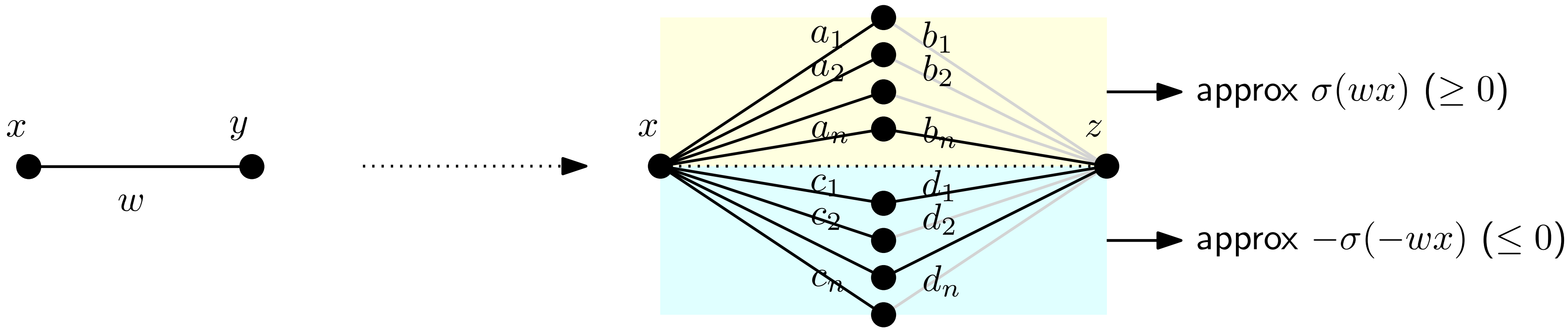
Putting everything together



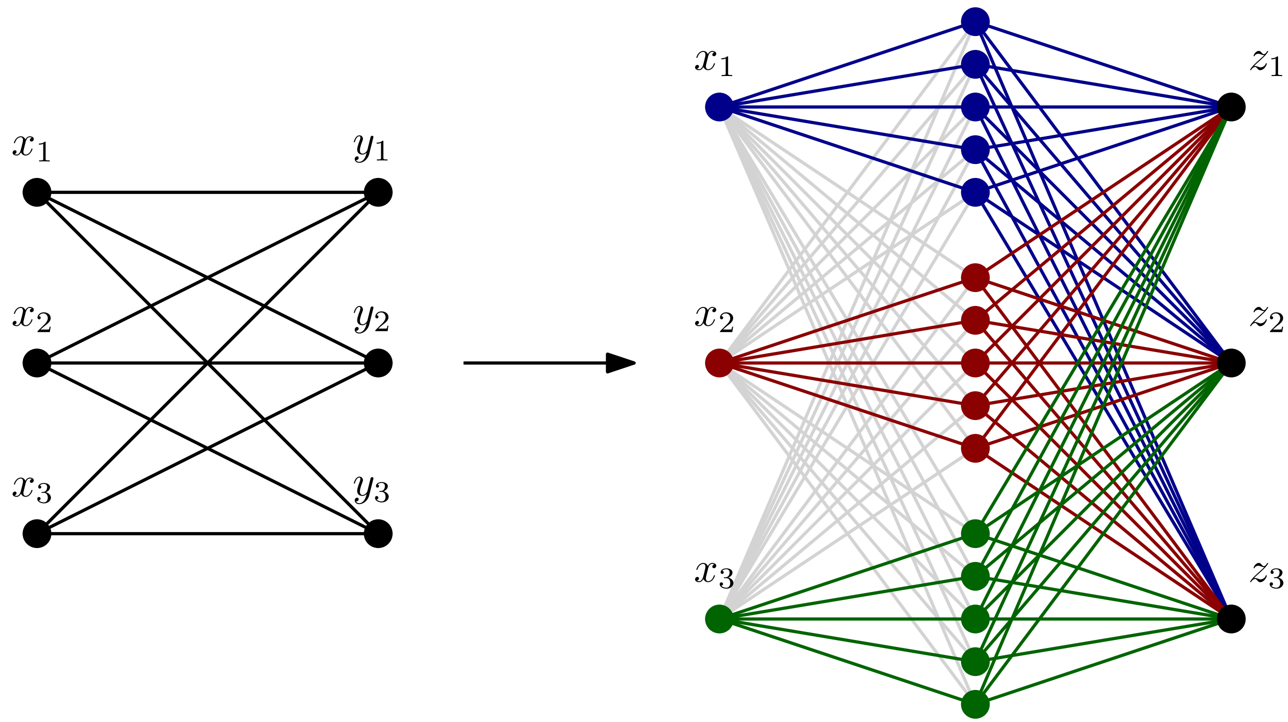
prune only the right layer: reuse the left layer



Putting everything together



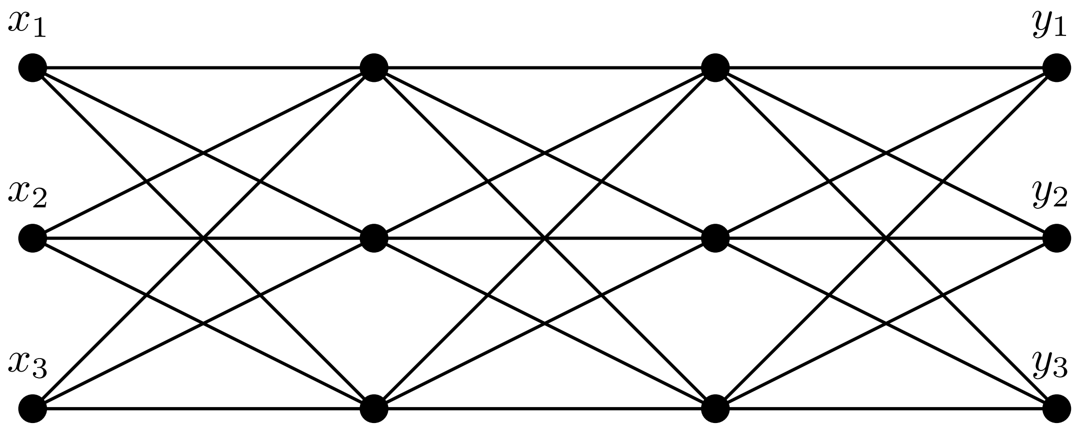
prune only the right layer: reuse the left layer



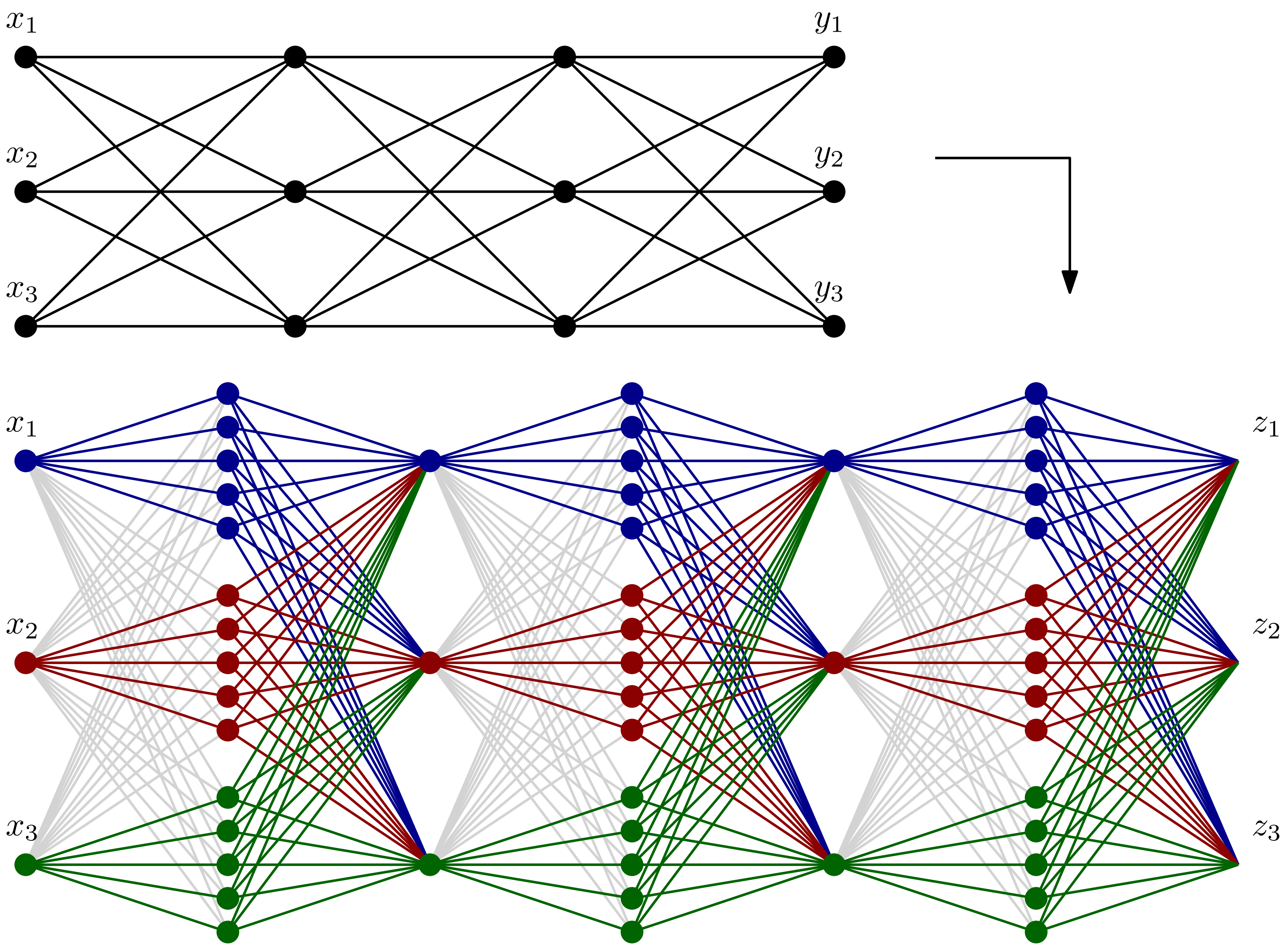
$n \geq C \log d^2/\varepsilon$

d = width original layer

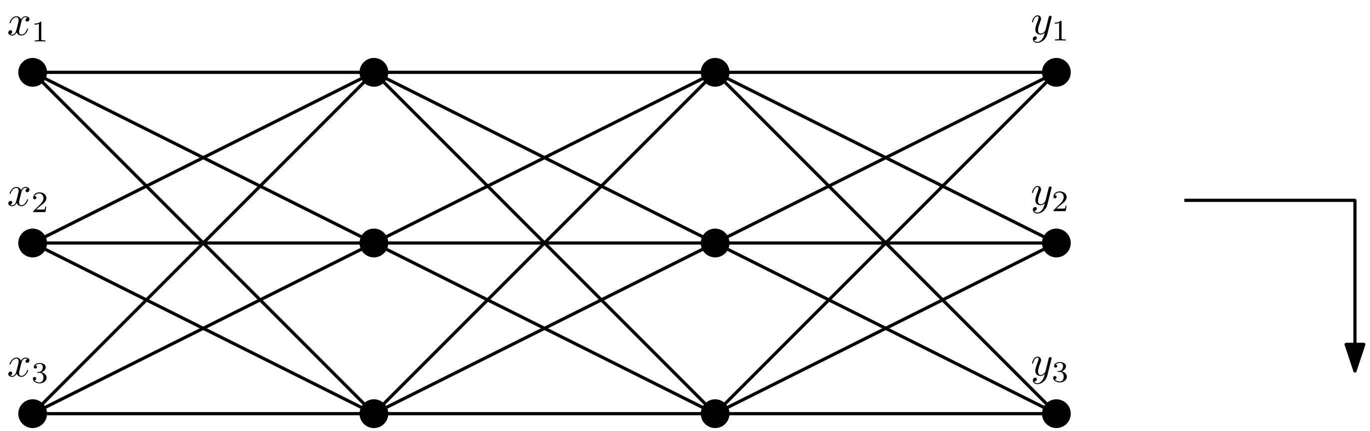
More layers together



More layers together



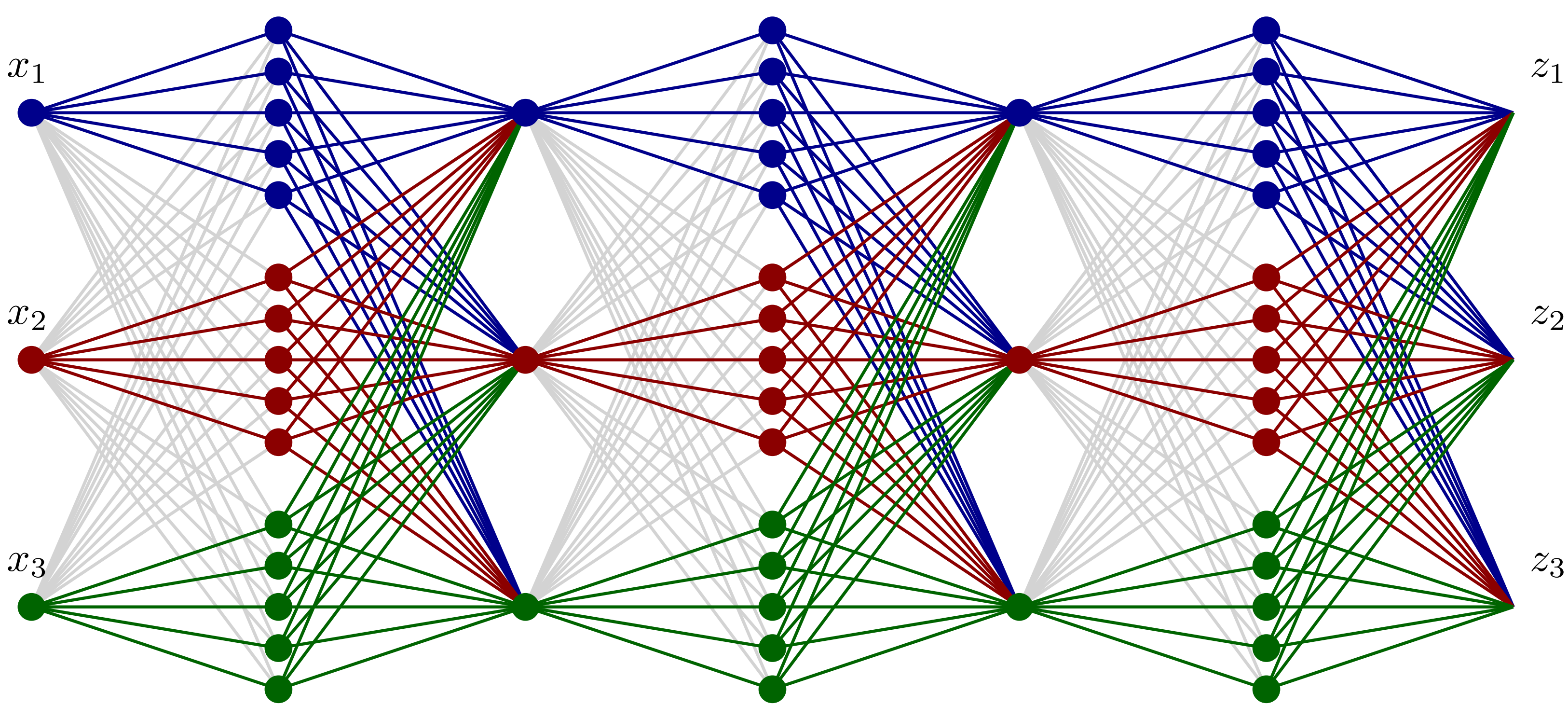
More layers together



$n \geq C \log \ell d^2/\varepsilon$

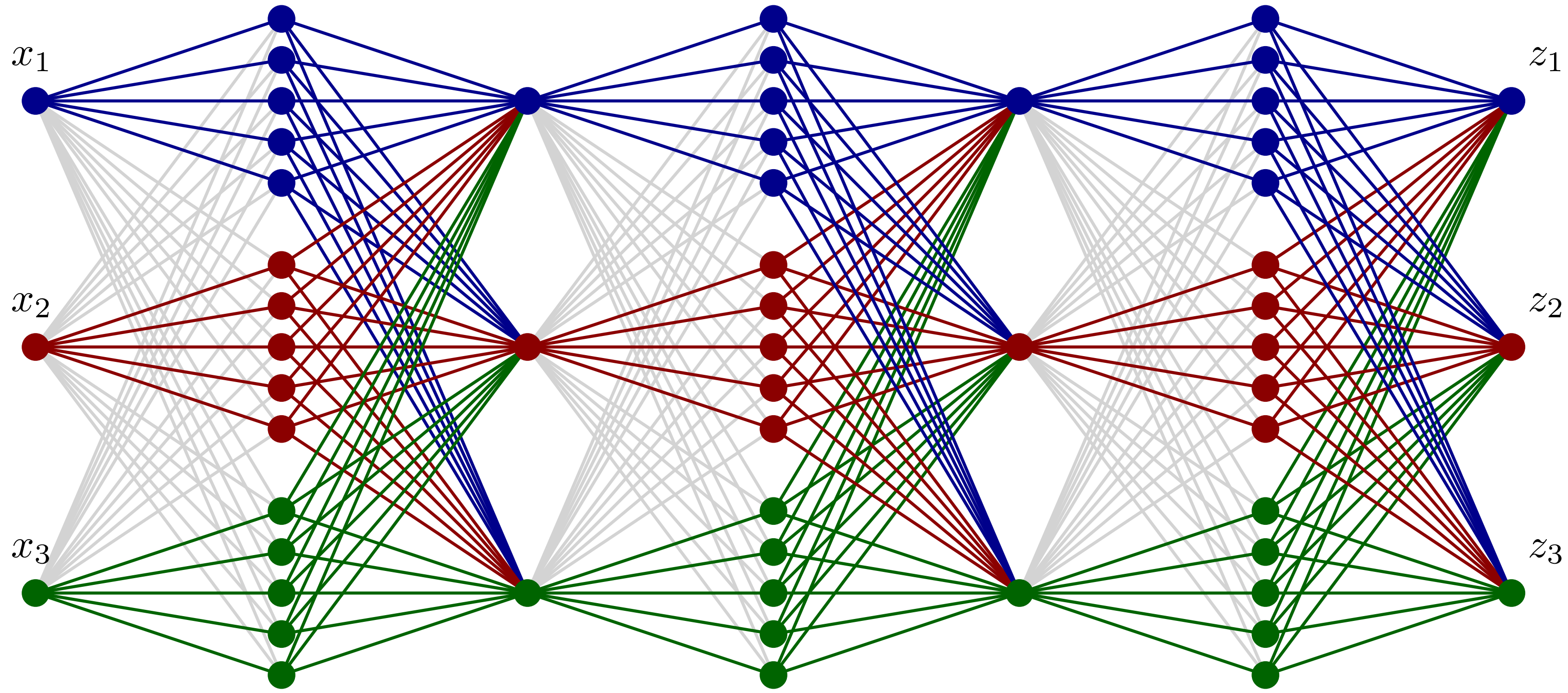
$\ell = \#$ original layers

$\implies \|\mathbf{y} - \mathbf{z}\| \leq 2\varepsilon$



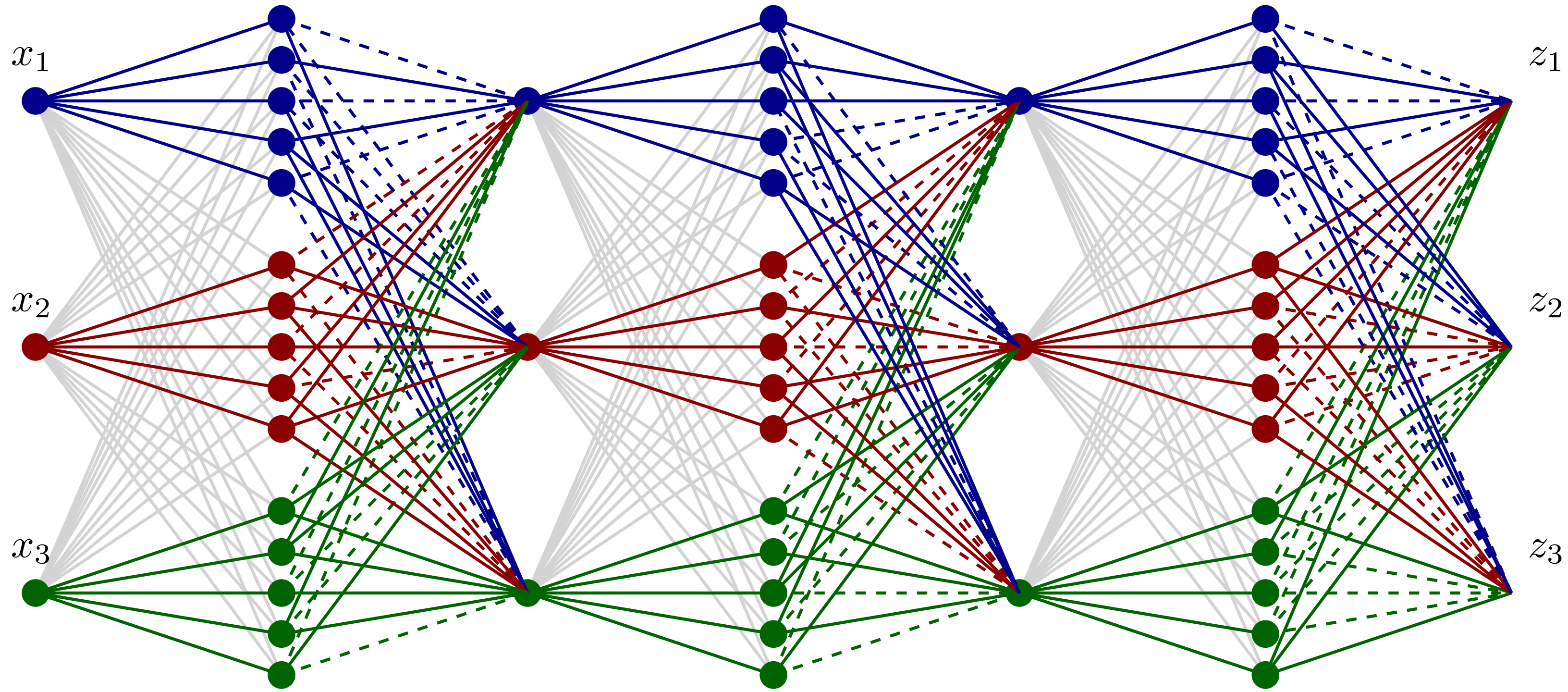
Issues with Unstructured pruning

- Removed edges can be everywhere



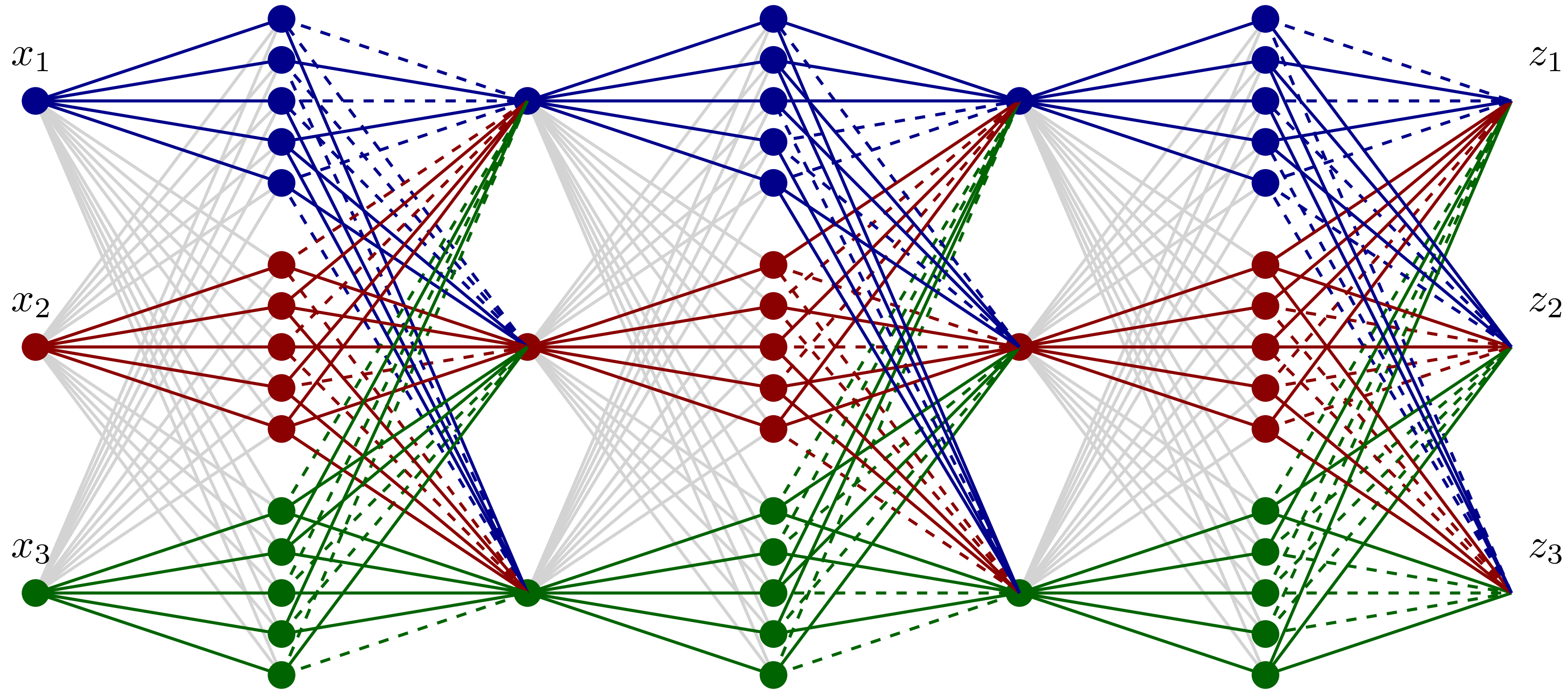
Issues with Unstructured pruning

- Removed edges can be everywhere



Issues with Unstructured pruning

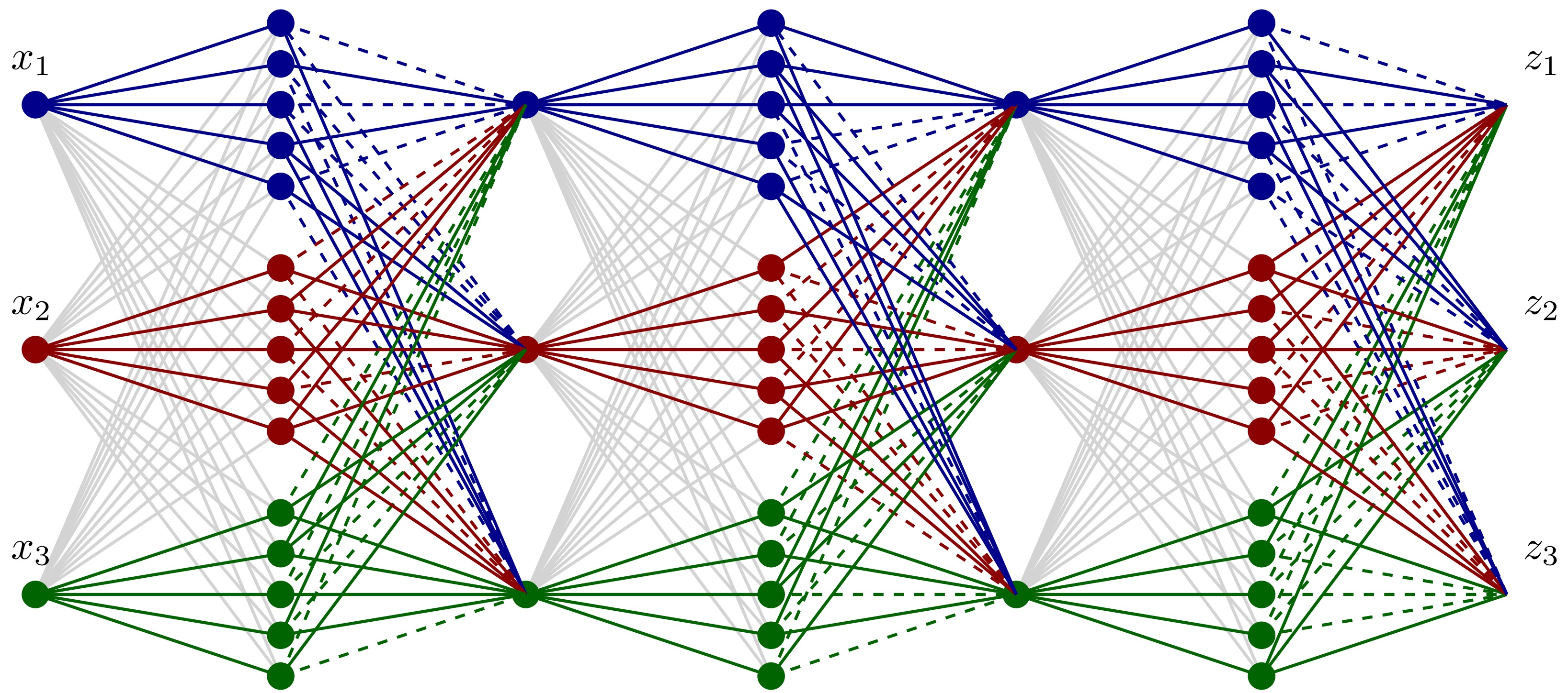
- Removed edges can be everywhere



- No structure usually implies slower processes

Issues with Unstructured pruning

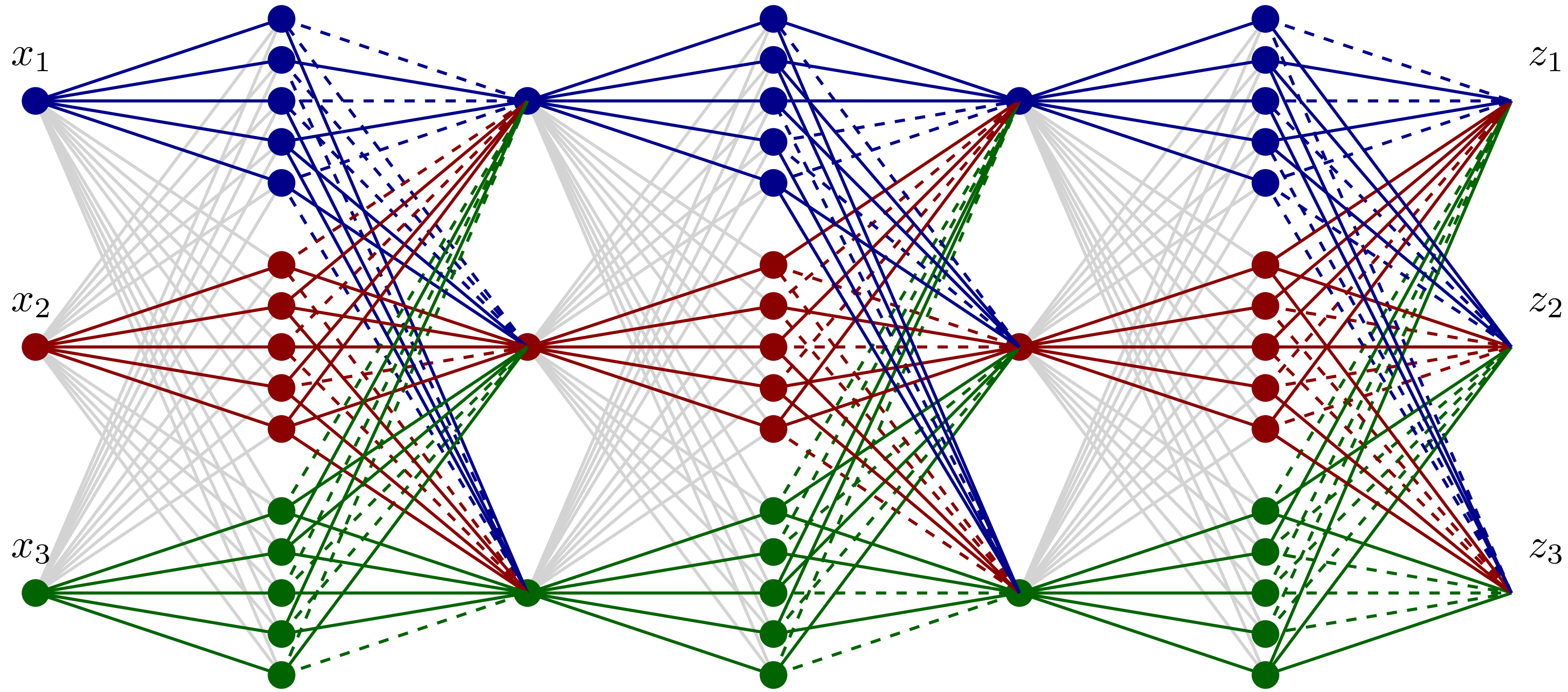
- Removed edges can be everywhere



- No structure usually implies slower processes
  - difficulty encoding unstructured sparsity

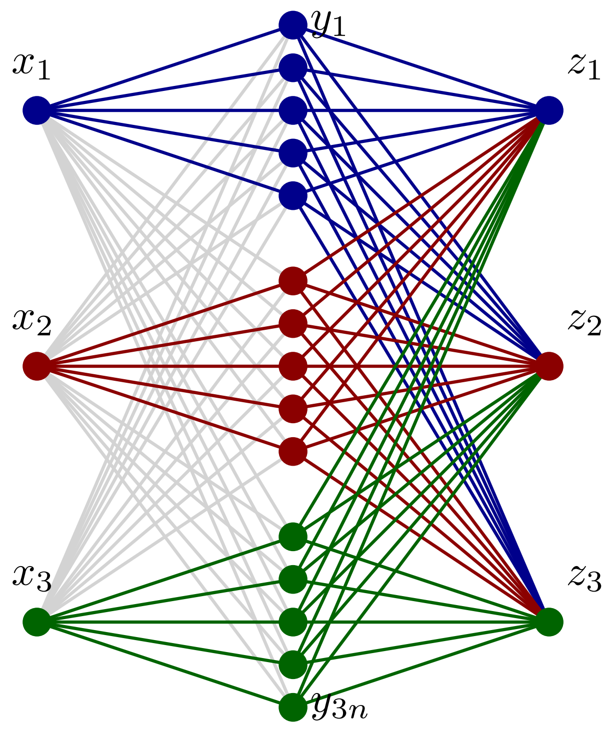
Issues with Unstructured pruning

- Removed edges can be everywhere

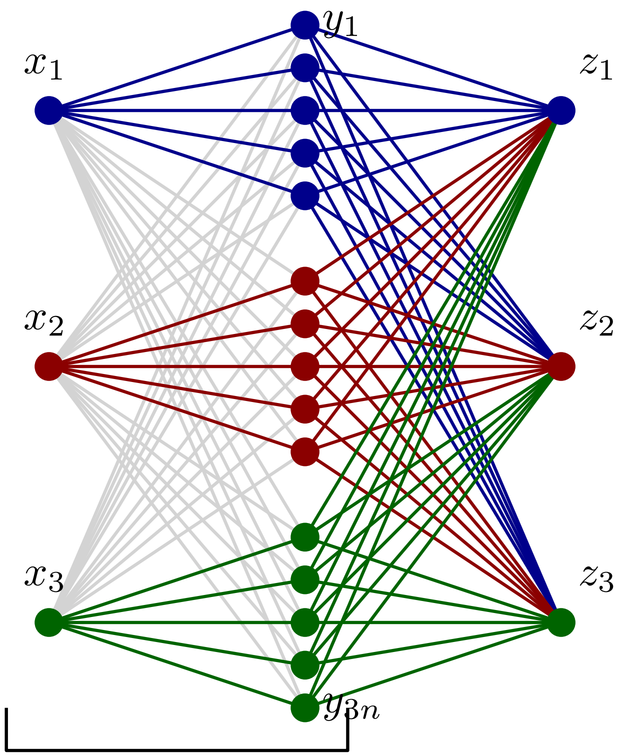


- No structure usually implies slower processes
  - difficulty encoding unstructured sparsity
  - accessing data is more time consuming than processing

Structured pruning

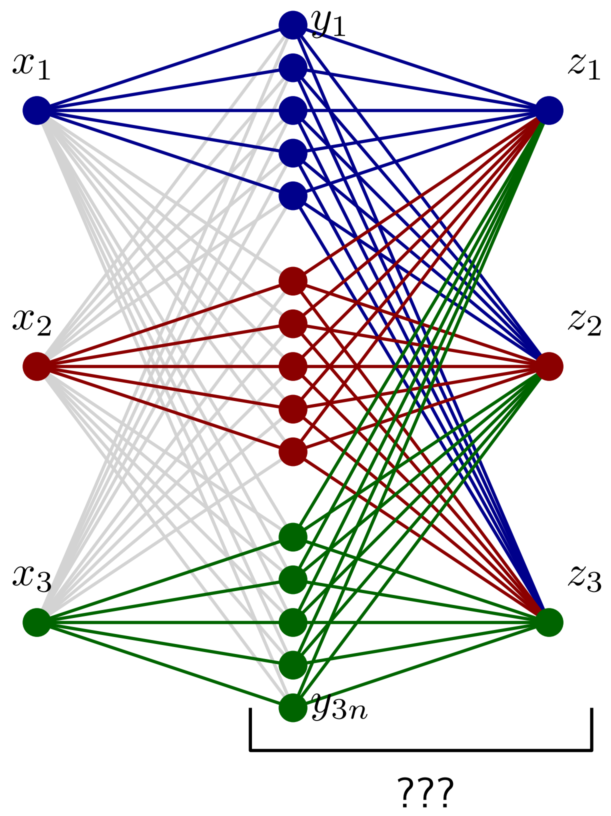


Structured pruning

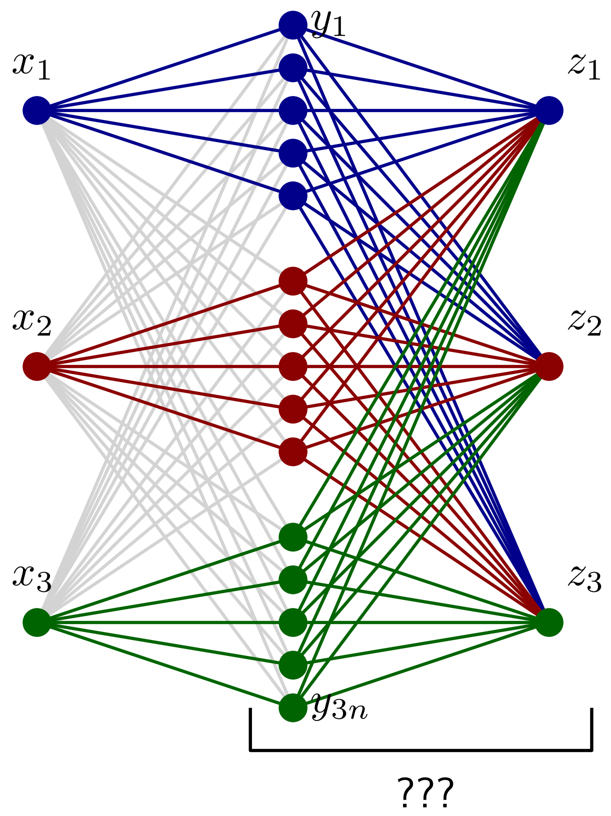


$$\begin{bmatrix} w_{1,1} & 0 & 0 \\ \vdots & \vdots & \vdots \\ w_{n,1} & 0 & 0 \\ 0 & w_{1,2} & 0 \\ \vdots & \vdots & \vdots \\ 0 & w_{n,2} & 0 \\ 0 & 0 & w_{1,3} \\ \vdots & \vdots & \vdots \\ 0 & 0 & w_{n,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Structured pruning

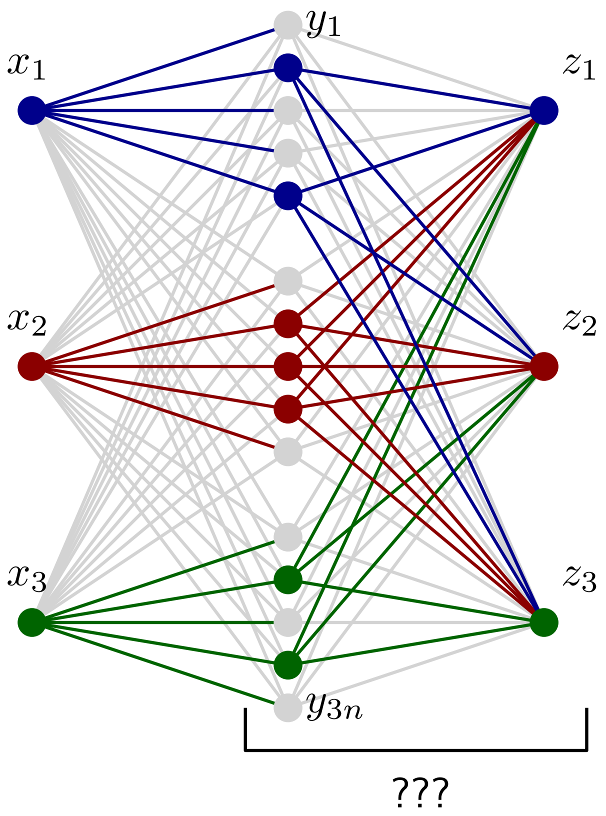


Structured pruning



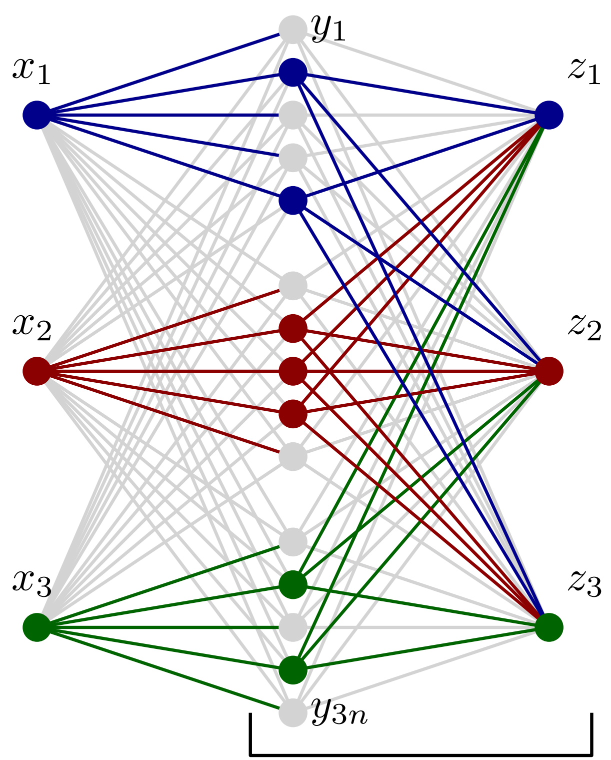
- Removing entire neurons from the middle layer!

Structured pruning



- Removing entire neurons from the middle layer!

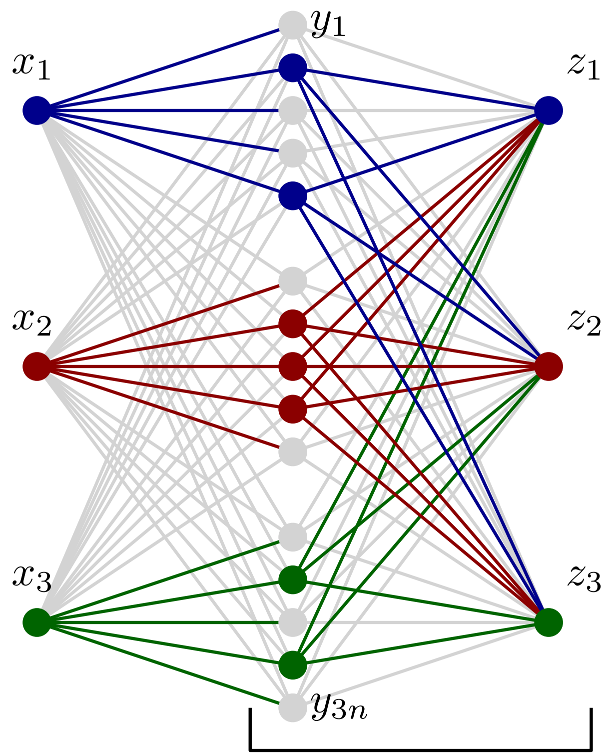
Structured pruning



- Removing entire neurons from the middle layer!
  - removes columns!

$$\begin{bmatrix} 0 & v_{1,2} & 0 & \dots & 0 & v_{i,1} & 0 & \dots \\ 0 & v_{2,2} & 0 & \dots & 0 & v_{i,2} & 0 & \dots \\ 0 & v_{3,2} & 0 & \dots & 0 & v_{i,2} & 0 & \dots \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{3n} \end{bmatrix}$$

Structured pruning

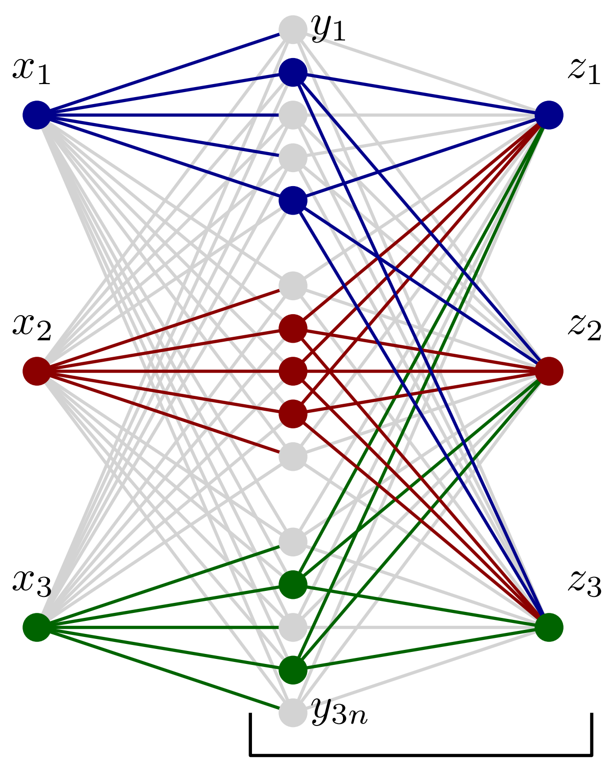


- Removing entire neurons from the middle layer!
  - removes columns!

- The one-dimensional RSS result does not work
  - leads to exponential bounds

$$\begin{bmatrix} 0 & v_{1,2} & 0 & \dots & 0 & v_{i,1} & 0 & \dots \\ 0 & v_{2,2} & 0 & \dots & 0 & v_{i,2} & 0 & \dots \\ 0 & v_{3,2} & 0 & \dots & 0 & v_{i,2} & 0 & \dots \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{3n} \end{bmatrix}$$

Structured pruning

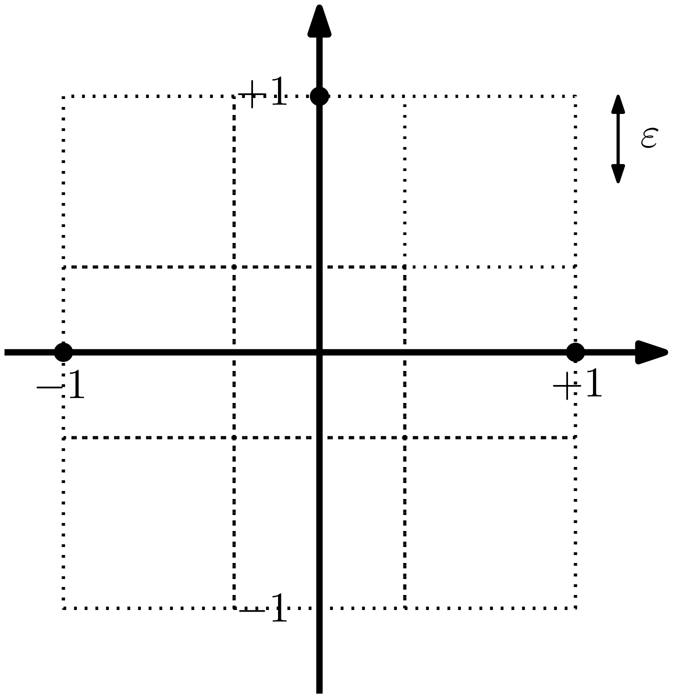


- Removing entire neurons from the middle layer!
  - removes columns!
- The one-dimensional RSS result does not work
  - leads to exponential bounds
- A multidimensional RSS result is required

$$\begin{bmatrix} 0 & v_{1,2} & 0 & \dots & 0 & v_{i,1} & 0 & \dots \\ 0 & v_{2,2} & 0 & \dots & 0 & v_{i,2} & 0 & \dots \\ 0 & v_{3,2} & 0 & \dots & 0 & v_{i,2} & 0 & \dots \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{3n} \end{bmatrix}$$

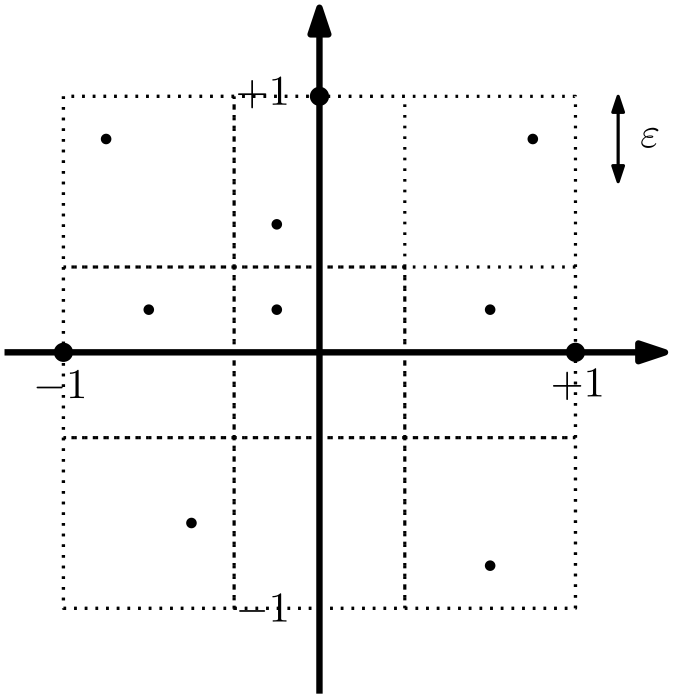
The multidimensional RSS problem

- Natural generalization



The multidimensional RSS problem

• Natural generalization

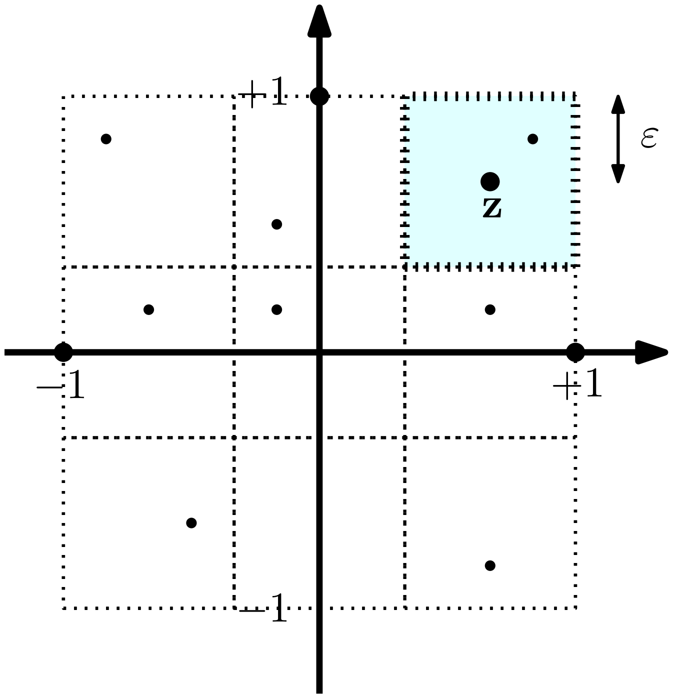


Input:

• Sequence of n i.i.d. random vectors $X_1, \ldots, X_n$

The multidimensional RSS problem

• Natural generalization



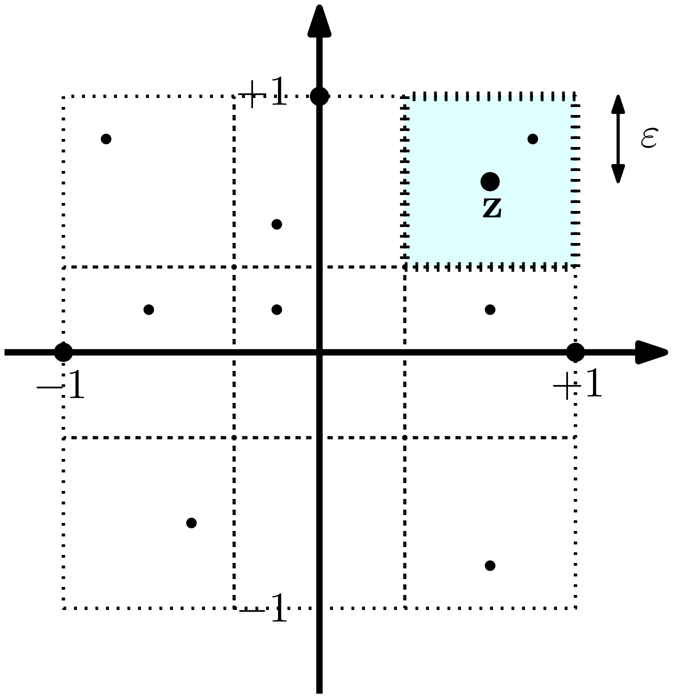
Input:

• Sequence of n i.i.d. random vectors $X_1, \ldots, X_n$

• Target vector $\mathbf{z} \in [-1, +1]^d$

The multidimensional RSS problem

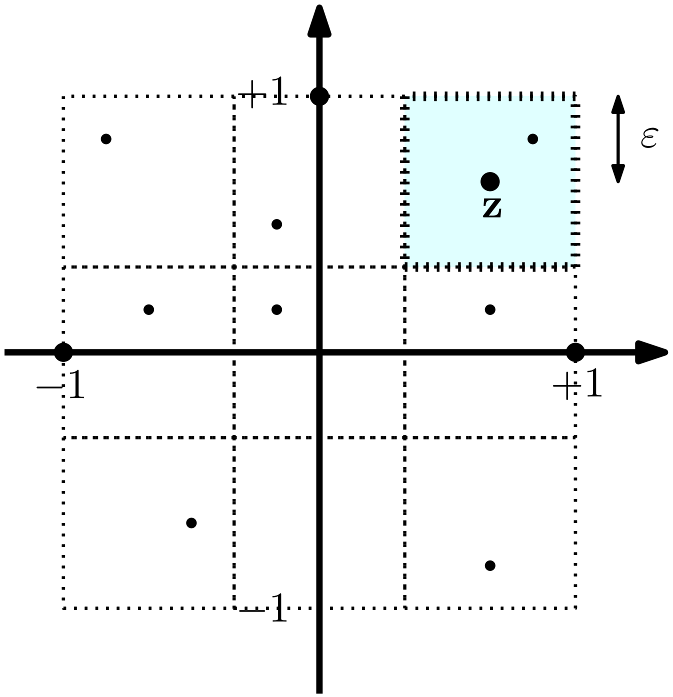
- Natural generalization



Input:

- Sequence of n i.i.d. random vectors $X_1, \ldots, X_n$
- Target vector $\mathbf{z} \in [-1, +1]^d$
- Error parameter $\varepsilon > 0$

The multidimensional RSS problem



• Natural generalization

Input:

• Sequence of n i.i.d. random vectors $X_1, \ldots, X_n$

• Target vector $\mathbf{z} \in [-1, +1]^d$

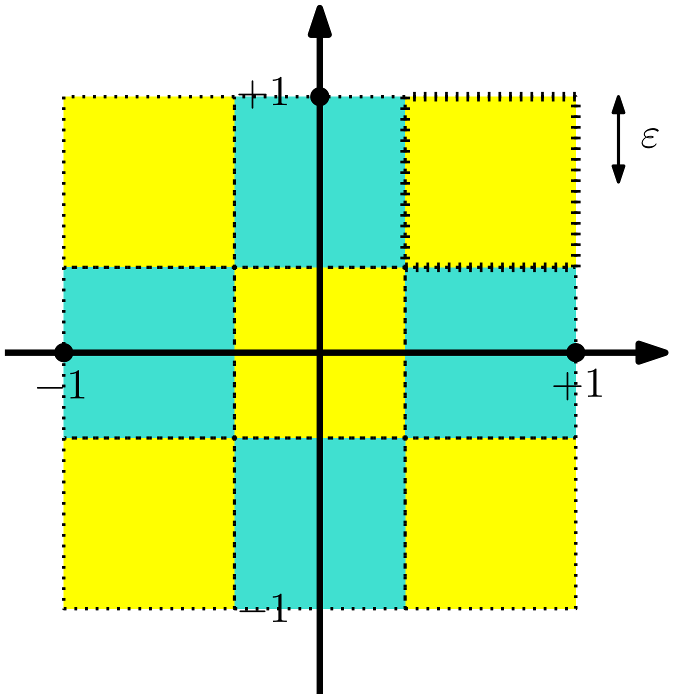
• Error parameter $\varepsilon > 0$

Question:

• Estimate n such that, with high probability, a subset $S \subseteq [n]$ exists with $\|\mathbf{z} - \sum_{i \in S} X_i\|_\infty \leq 2\varepsilon$

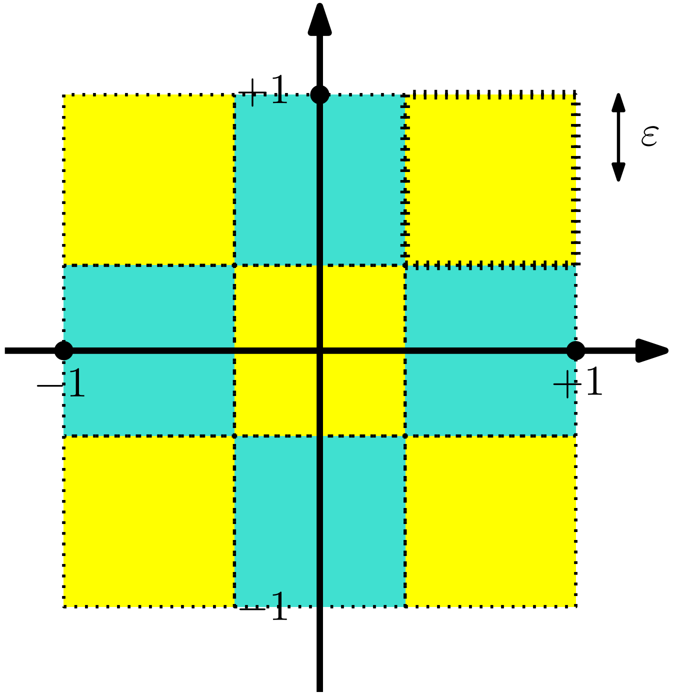
MRSS in expectation

- Number of ε-cubes: $1/\varepsilon^d = 2^{d \log 1/\varepsilon}$



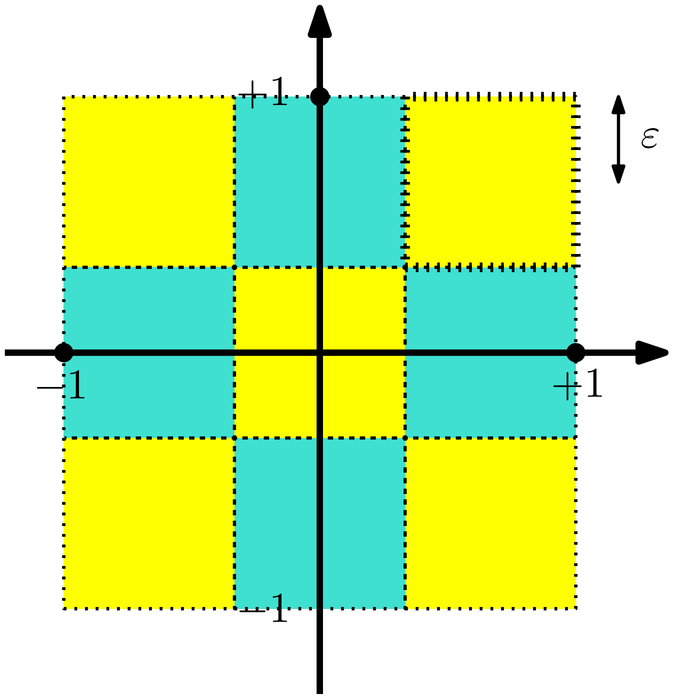
MRSS in expectation

- Number of ε-cubes: $1/\varepsilon^d = 2^{d \log 1/\varepsilon}$
- Sequence of n i.i.d. random vectors $X_1, \ldots, X_n \sim \mathcal{N}(\mathbf{0}, I_d)$



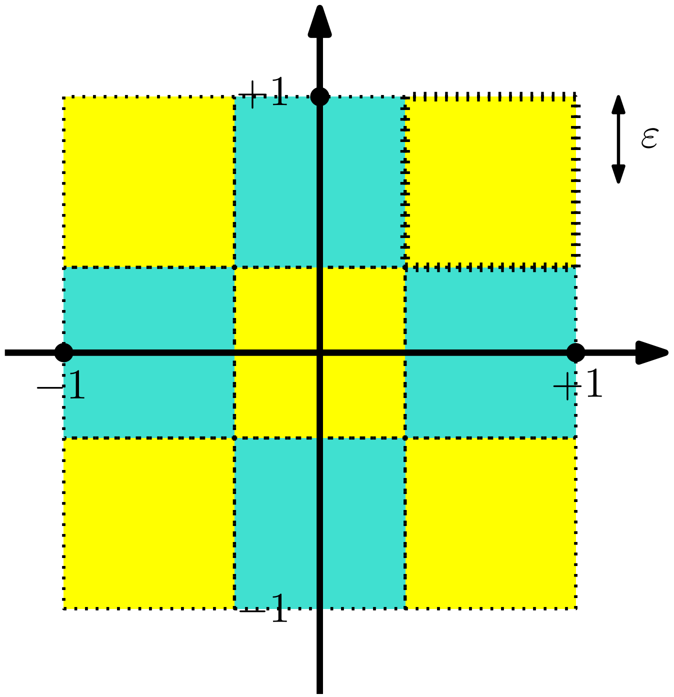
MRSS in expectation

- Number of ε-cubes: $1/\varepsilon^d = 2^{d \log 1/\varepsilon}$
- Sequence of n i.i.d. random vectors $X_1, \ldots, X_n \sim \mathcal{N}(\mathbf{0}, I_d)$
- 2^n possible subsets



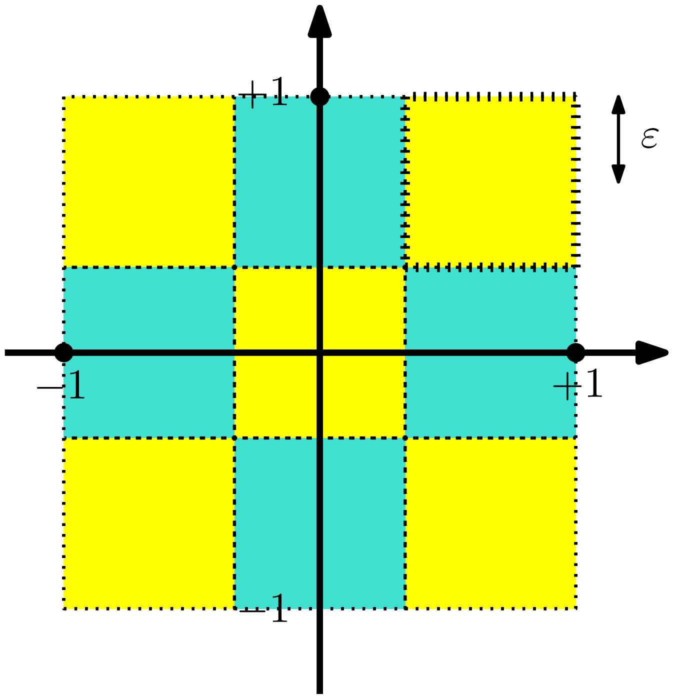
MRSS in expectation

- Number of ε-cubes: $1/\varepsilon^d = 2^{d \log 1/\varepsilon}$
- Sequence of n i.i.d. random vectors $X_1, \ldots, X_n \sim \mathcal{N}(\mathbf{0}, I_d)$
- 2^n possible subsets



Upper bound

MRSS in expectation

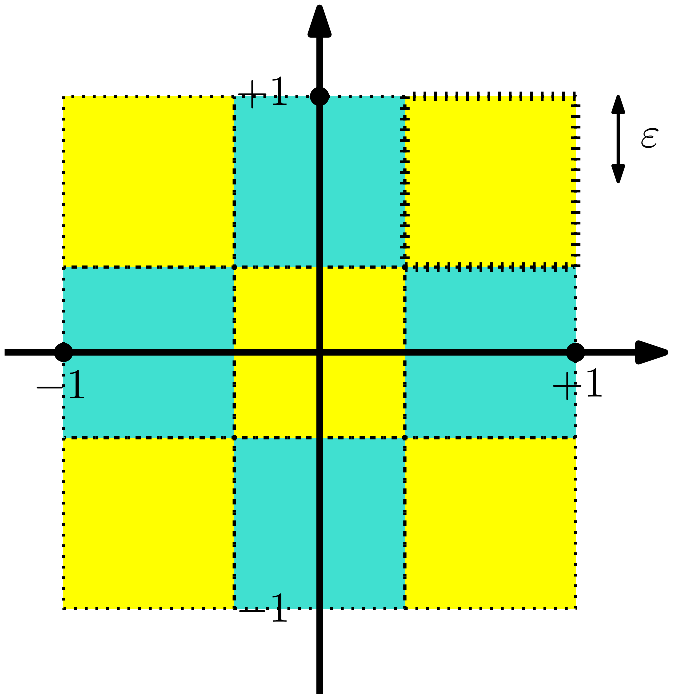


- Number of ε-cubes: $1/\varepsilon^d = 2^{d \log 1/\varepsilon}$
- Sequence of n i.i.d. random vectors $X_1, \ldots, X_n \sim \mathcal{N}(\mathbf{0}, I_d)$
- 2^n possible subsets

Upper bound

- If subset size $k = \frac{n}{2}$, possible subsets: $\binom{n}{n/2} \geq 2^{n/2}$

MRSS in expectation



- Number of ε-cubes: $1/\varepsilon^d = 2^{d \log 1/\varepsilon}$

- Sequence of n i.i.d. random vectors $X_1, \ldots, X_n \sim \mathcal{N}(\mathbf{0}, I_d)$

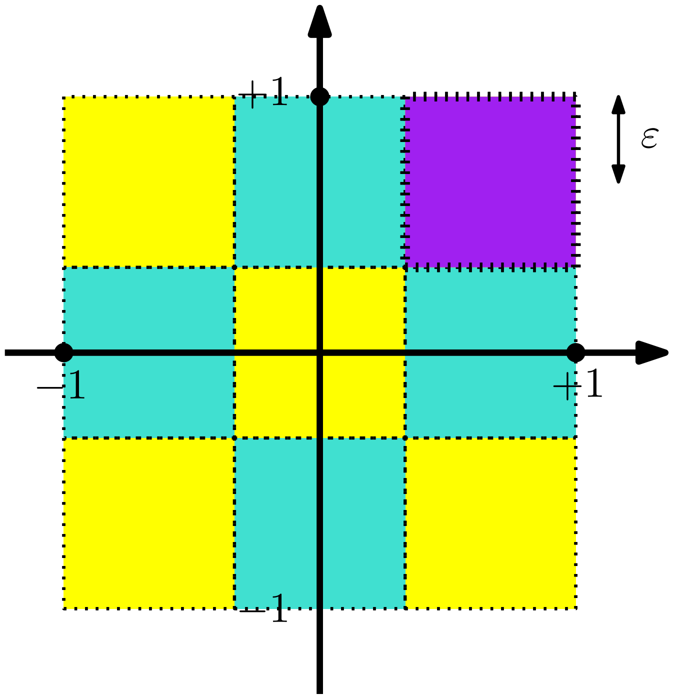
- 2^n possible subsets

Upper bound

- If subset size $k = \frac{n}{2}$, possible subsets: $\binom{n}{n/2} \geq 2^{n/2}$

- Each subset $S \subseteq [n]$, $|S| = \frac{n}{2}$, gives a Gaussian $Y_S \sim \mathcal{N}(\mathbf{0}, \frac{n}{2} I_d)$

MRSS in expectation

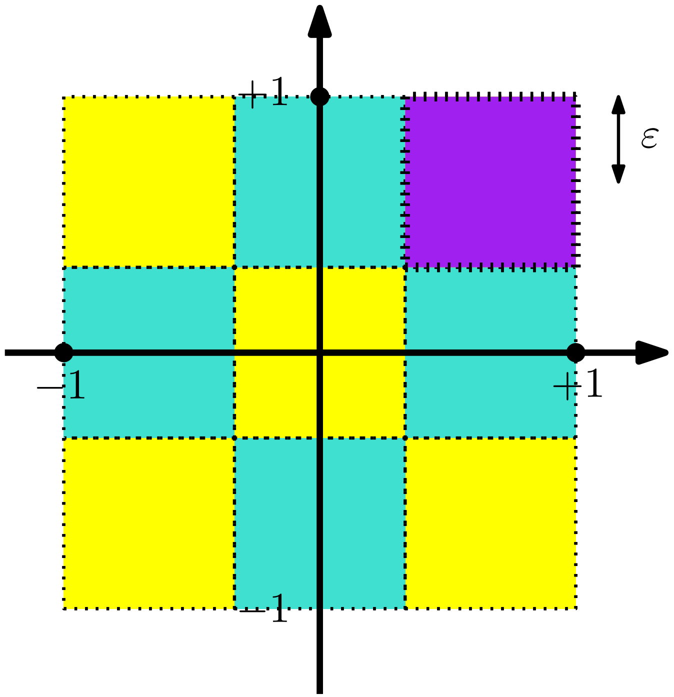


- Number of ε-cubes: $1/\varepsilon^d = 2^{d \log 1/\varepsilon}$
- Sequence of n i.i.d. random vectors $X_1, \ldots, X_n \sim \mathcal{N}(\mathbf{0}, I_d)$
- 2^n possible subsets

Upper bound

- If subset size $k = \frac{n}{2}$, possible subsets: $\binom{n}{n/2} \geq 2^{n/2}$
- Each subset $S \subseteq [n]$, $|S| = \frac{n}{2}$, gives a Gaussian $Y_S \sim \mathcal{N}(\mathbf{0}, \frac{n}{2} I_d)$
- Probability roughly $(\varepsilon/\sqrt{n/2})^d$ to hit any ε-cube

MRSS in expectation



- Number of ε-cubes: $1/\varepsilon^d = 2^{d \log 1/\varepsilon}$
- Sequence of n i.i.d. random vectors $X_1, \ldots, X_n \sim \mathcal{N}(\mathbf{0}, I_d)$
- 2^n possible subsets

Upper bound

- If subset size $k = \frac{n}{2}$, possible subsets: $\binom{n}{n/2} \geq 2^{n/2}$
- Each subset $S \subseteq [n]$, $|S| = \frac{n}{2}$, gives a Gaussian $Y_S \sim \mathcal{N}(\mathbf{0}, \frac{n}{2} I_d)$
- Probability roughly $(\varepsilon/\sqrt{n/2})^d$ to hit any ε-cube

$$\mathbb{E}\left[\#\text{ subsets approximating any cube}\right] \geq 2^{n/2} \cdot \left(\frac{\varepsilon}{\sqrt{n/2}}\right)^d$$

$$= 2^{n/2 - d \log 1/\varepsilon - d/2 \log n/2} = 2^{O(n)} \text{ if } n \geq C d \log 1/\varepsilon$$

MRSS in expectation

Lower bound

- If subset size k, possible subsets: $\binom{n}{k} \leq (en/k)^k$

MRSS in expectation

Lower bound

- If subset size k, possible subsets: $\binom{n}{k} \leq (en/k)^k$
- Each subset $S \subseteq [n]$, $|S| = k$, gives a Gaussian $Y_S \sim \mathcal{N}(\mathbf{0}, kI_d)$

MRSS in expectation

Lower bound

- If subset size k, possible subsets: $\binom{n}{k} \leq (en/k)^k$
- Each subset $S \subseteq [n]$, $|S| = k$, gives a Gaussian $Y_S \sim \mathcal{N}(\mathbf{0}, kI_d)$
- Probability roughly $(\varepsilon/\sqrt{k})^d$ to hit any ε-cube

MRSS in expectation

Lower bound

- If subset size k, possible subsets: $\binom{n}{k} \leq (en/k)^k$
- Each subset $S \subseteq [n]$, $|S| = k$, gives a Gaussian $Y_S \sim \mathcal{N}(\mathbf{0}, kI_d)$
- Probability roughly $(\varepsilon/\sqrt{k})^d$ to hit any ε-cube

$$\mathbb{E}\left[\# \text{ subsets approximating any cube}\right] \leq \sum_{k=1}^{n} (en/k)^k \cdot \left(\frac{\varepsilon}{\sqrt{k}}\right)^d$$

$$= \sum_{k=1}^{n} 2^{k \log(en/k) - d \log 1/\varepsilon - d/2 \log k} \leq n \cdot 2^{n/2 \log(2e) - d \log 1/\varepsilon - d/2 \log n/2}$$

MRSS in expectation

Lower bound

- If subset size k, possible subsets: $\binom{n}{k} \leq (en/k)^k$
- Each subset $S \subseteq [n]$, $|S| = k$, gives a Gaussian $Y_S \sim \mathcal{N}(\mathbf{0}, kI_d)$
- Probability roughly $(\varepsilon/\sqrt{k})^d$ to hit any ε-cube

$$\mathbb{E}\left[\#\text{ subsets approximating any cube}\right] \leq \sum_{k=1}^{n} (en/k)^k \cdot \left(\frac{\varepsilon}{\sqrt{k}}\right)^d$$

$$= \sum_{k=1}^{n} 2^{k\log(en/k) - d\log 1/\varepsilon - d/2\log k} \leq n \cdot 2^{n/2\log(2e) - d\log 1/\varepsilon - d/2\log n/2}$$

< 1 if $n \leq cd\log 1/\varepsilon$ for c small enough

MRSS: current results

- [Borst et al. 2022; Becchetti et al. 2022] use the 2nd moment method to derive bounds

MRSS: current results

- [Borst et al. 2022; Becchetti et al. 2022] use the 2nd moment method to derive bounds

- for $S \subseteq [n]$, $Y_S = 1$ if $\sum_{i \in S} X_i$ approximates target $\mathbf{z}$ and 0 otherwise

MRSS: current results

- [Borst et al. 2022; Becchetti et al. 2022] use the 2nd moment method to derive bounds

- for $S \subseteq [n]$, $Y_S = 1$ if $\sum_{i \in S} X_i$ approximates target $\mathbf{z}$ and 0 otherwise

- $Z_n = \sum_{S \subseteq [n]} Y_S$ random variable yielding number of subsets approximating target $\mathbf{z}$

MRSS: current results

- [Borst et al. 2022; Becchetti et al. 2022] use the 2nd moment method to derive bounds

- for $S \subseteq [n]$, $Y_S = 1$ if $\sum_{i \in S} X_i$ approximates target $\mathbf{z}$ and 0 otherwise

- $Z_n = \sum_{S \subseteq [n]} Y_S$ random variable yielding number of subsets approximating target $\mathbf{z}$

- $\mathsf{P}[Z_n \geq 1] \geq (\mathbb{E}[Z_n])^2 / \mathbb{E}[Z_n^2]$

MRSS: current results

- [Borst et al. 2022; Becchetti et al. 2022] use the 2nd moment method to derive bounds

- for $S \subseteq [n]$, $Y_S = 1$ if $\sum_{i \in S} X_i$ approximates target $\mathbf{z}$ and 0 otherwise

- $Z_n = \sum_{S \subseteq [n]} Y_S$ random variable yielding number of subsets approximating target $\mathbf{z}$

- $\mathrm{P}\left[Z_n \geq 1\right] \geq \left(\mathbb{E}\left[Z_n\right]\right)^2 / \mathbb{E}\left[Z_n^2\right]$

- **Challenge**: dealing with dependencies to estimate $\mathbb{E}\left[Z_n^2\right]$

MRSS: current results

- [Borst et al. 2022; Becchetti et al. 2022] use the 2nd moment method to derive bounds

  - for $S \subseteq [n]$, $Y_S = 1$ if $\sum_{i \in S} X_i$ approximates target $\mathbf{z}$ and 0 otherwise

  - $Z_n = \sum_{S \subseteq [n]} Y_S$ random variable yielding number of subsets approximating target $\mathbf{z}$

  - $\mathrm{P}\left[Z_n \geq 1\right] \geq (\mathbb{E}\left[Z_n\right])^2 / \mathbb{E}\left[Z_n^2\right]$

- **Challenge**: dealing with dependencies to estimate $\mathbb{E}\left[Z_n^2\right]$

  - choose only subsets of size αn so that the "average intersection" concentrates around $\alpha^2 n$

MRSS: current results

- [Borst et al. 2022; Becchetti et al. 2022] use the 2nd moment method to derive bounds
  - for $S \subseteq [n]$, $Y_S = 1$ if $\sum_{i \in S} X_i$ approximates target $\mathbf{z}$ and 0 otherwise
  - $Z_n = \sum_{S \subseteq [n]} Y_S$ random variable yielding number of subsets approximating target $\mathbf{z}$
  - $\mathrm{P}\left[Z_n \geq 1\right] \geq (\mathbb{E}\left[Z_n\right])^2 / \mathbb{E}\left[Z_n^2\right]$

- **Challenge**: dealing with dependencies to estimate $\mathbb{E}\left[Z_n^2\right]$
  - choose only subsets of size αn so that the "average intersection" concentrates around $\alpha^2 n$

- **Result**: $n \geq \text{poly}(d) \log(d/\varepsilon)$ ($\alpha = 1/\sqrt{d}$)

MRSS: current results

- [Borst et al. 2022; Becchetti et al. 2022] use the 2nd moment method to derive bounds

  - for $S \subseteq [n]$, $Y_S = 1$ if $\sum_{i \in S} X_i$ approximates target $\mathbf{z}$ and 0 otherwise

  - $Z_n = \sum_{S \subseteq [n]} Y_S$ random variable yielding number of subsets approximating target $\mathbf{z}$

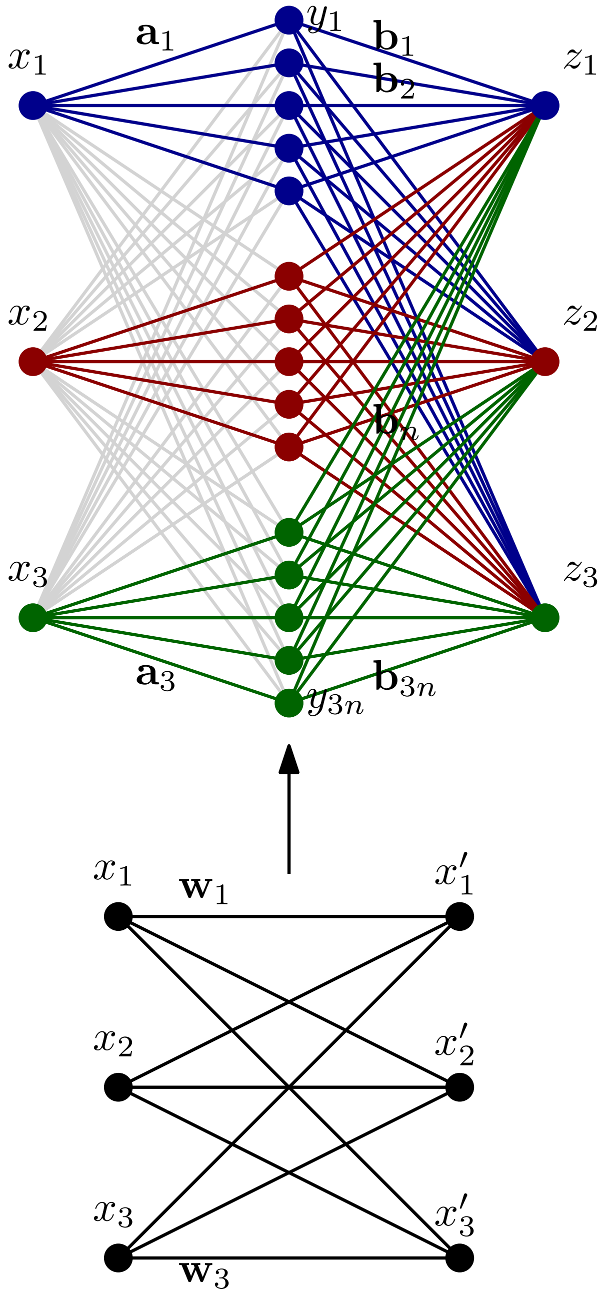
  - $\mathrm{P}\left[Z_n \geq 1\right] \geq \left(\mathbb{E}\left[Z_n\right]\right)^2 / \mathbb{E}\left[Z_n^2\right]$

- **Challenge**: dealing with dependencies to estimate $\mathbb{E}\left[Z_n^2\right]$

  - choose only subsets of size αn so that the "average intersection" concentrates around $\alpha^2 n$

- **Result**: $n \geq \text{poly}(d) \log(d/\varepsilon)$ $(\alpha = 1/\sqrt{d})$
- What about approximating all the hypercube $[-1,1]^d$? The **union bound** is highly non-optimal

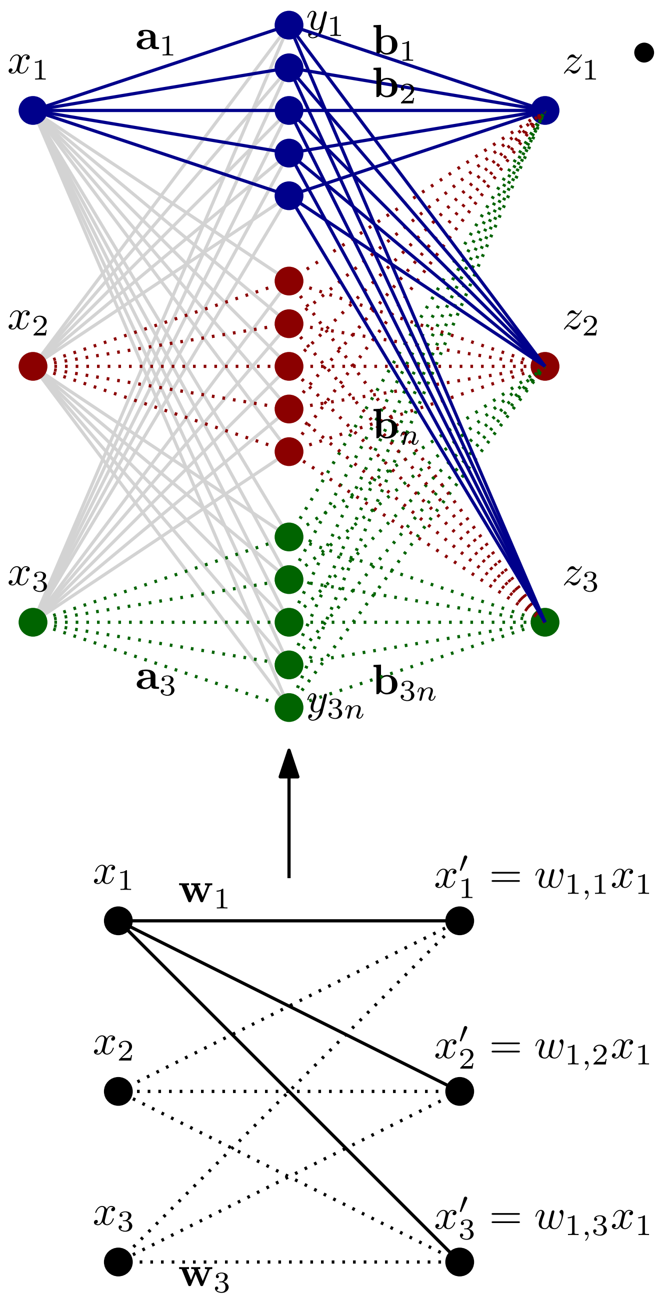
Apply MRSS for structured pruning



- $\mathbf{a}_i \sim \mathcal{N}(\mathbf{0}, I_n)$, $\mathbf{b}_i \sim \mathcal{N}(\mathbf{0}, I_d)$ (here, $d = 3$)

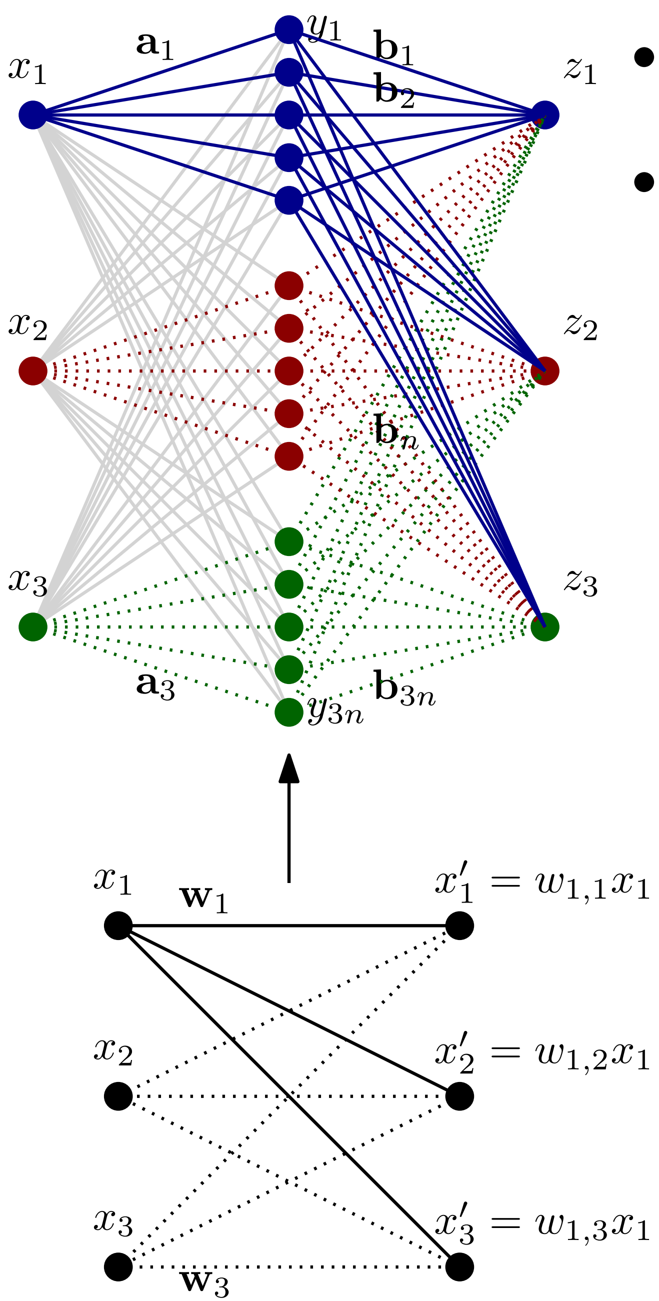
  (each neuron i has one $\mathbf{b}_i$)

Apply MRSS for structured pruning



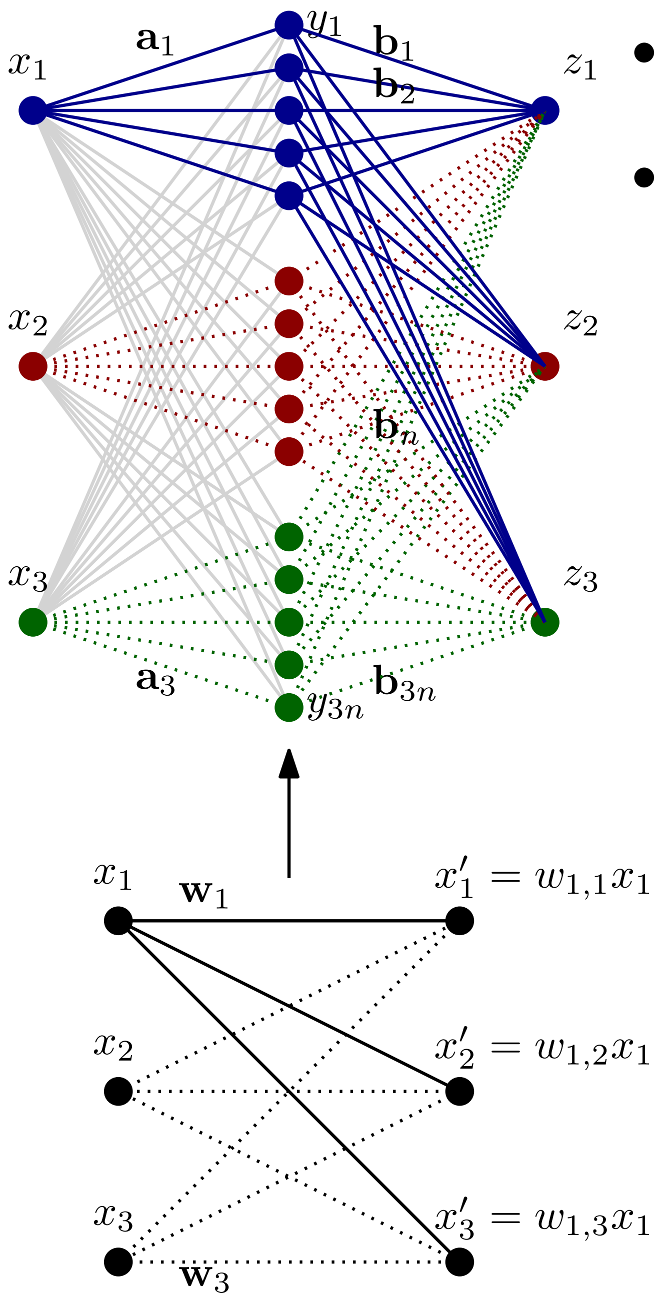
- $\mathbf{a}_i \sim \mathcal{N}(\mathbf{0}, I_n)$, $\mathbf{b}_i \sim \mathcal{N}(\mathbf{0}, I_d)$ (here, $d = 3$)

Apply MRSS for structured pruning



- $\mathbf{a}_i \sim \mathcal{N}(\mathbf{0}, I_n)$, $\mathbf{b}_i \sim \mathcal{N}(\mathbf{0}, I_d)$ (here, $d = 3$)
- For simplicity: **no ReLU**

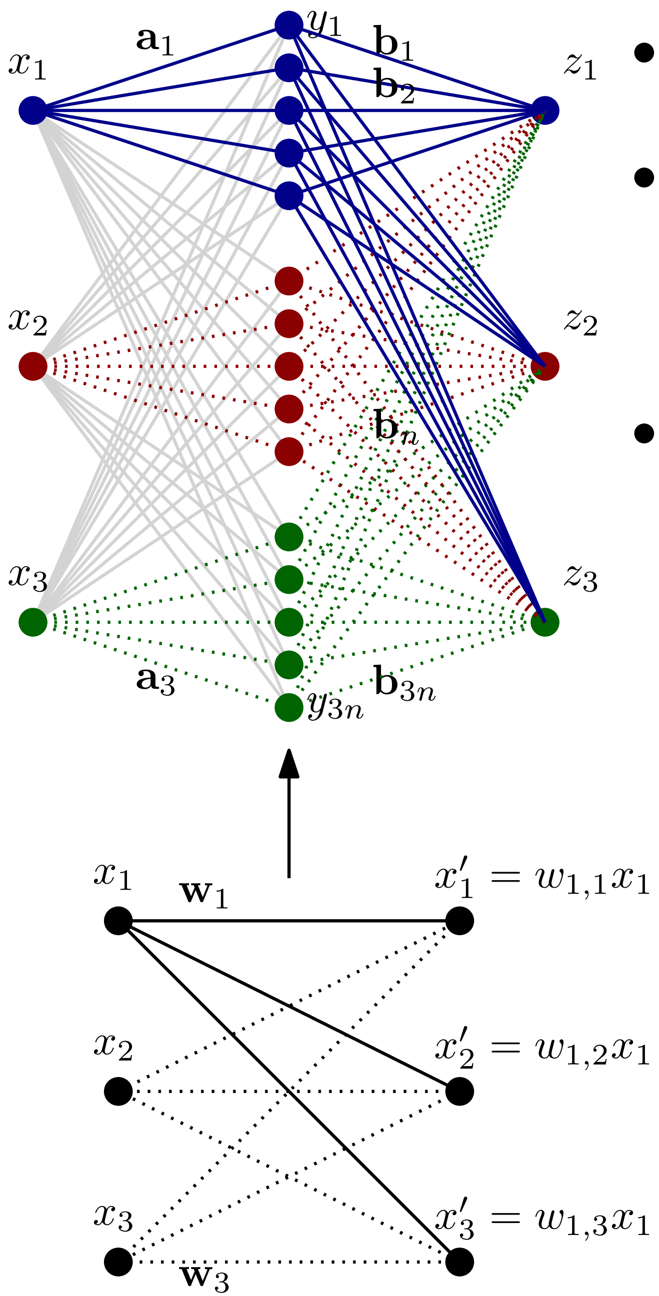
Apply MRSS for structured pruning



- $\mathbf{a}_i \sim \mathcal{N}(\mathbf{0}, I_n)$, $\mathbf{b}_i \sim \mathcal{N}(\mathbf{0}, I_d)$ (here, $d = 3$)
- For simplicity: **no ReLU**

$$\left\|x_1\mathbf{w}_1 - \sum_{i=1}^{n} x_1 a_{1,i}\mathbf{b}_i\right\|_\infty \leq |x_1|\left\|\mathbf{w}_1 - \sum_{i=1}^{n} a_{1,i}\mathbf{b}_i\right\|_\infty$$

Apply MRSS for structured pruning



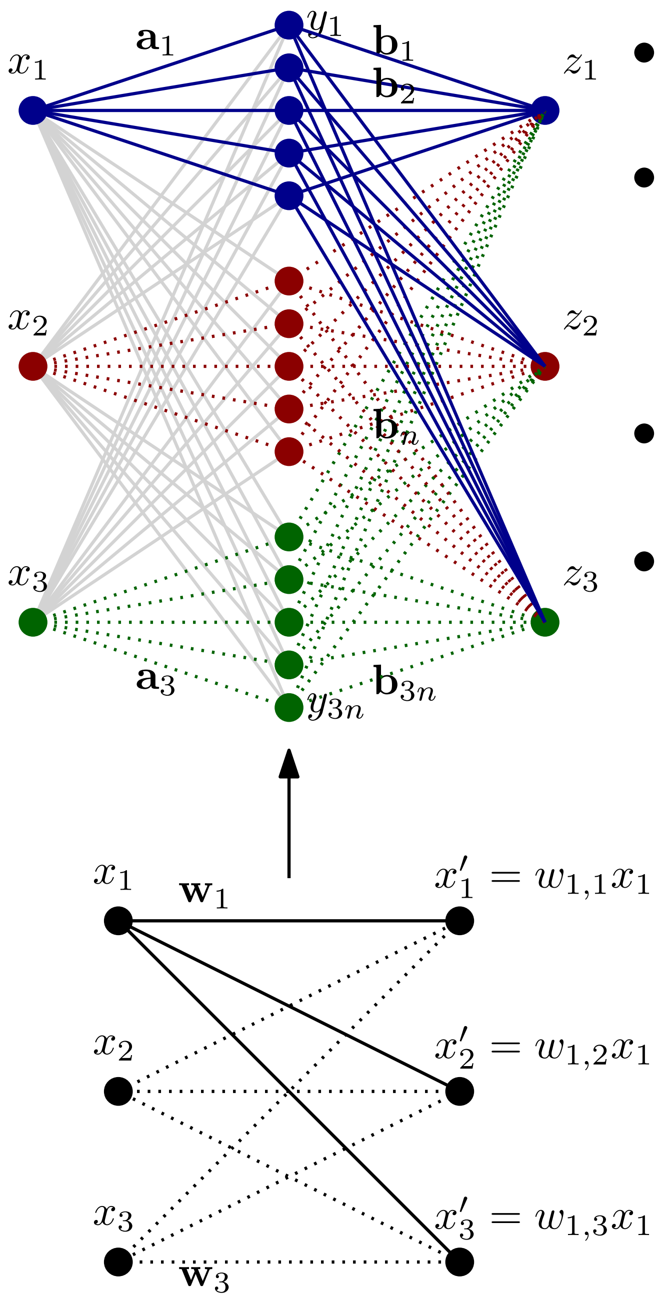
- $\mathbf{a}_i \sim \mathcal{N}(\mathbf{0}, I_n)$, $\mathbf{b}_i \sim \mathcal{N}(\mathbf{0}, I_d)$ (here, $d = 3$)

- For simplicity: **no ReLU**

$$\|x_1\mathbf{w}_1 - \sum_{i=1}^n x_1 a_{1,i}\mathbf{b}_i\|_\infty \leq |x_1| \|\mathbf{w}_1 - \sum_{i=1}^n a_{1,i}\mathbf{b}_i\|_\infty$$

- **Issue**: dependencies among entries of $a_{1,i}\mathbf{b}_i$!

Apply MRSS for structured pruning

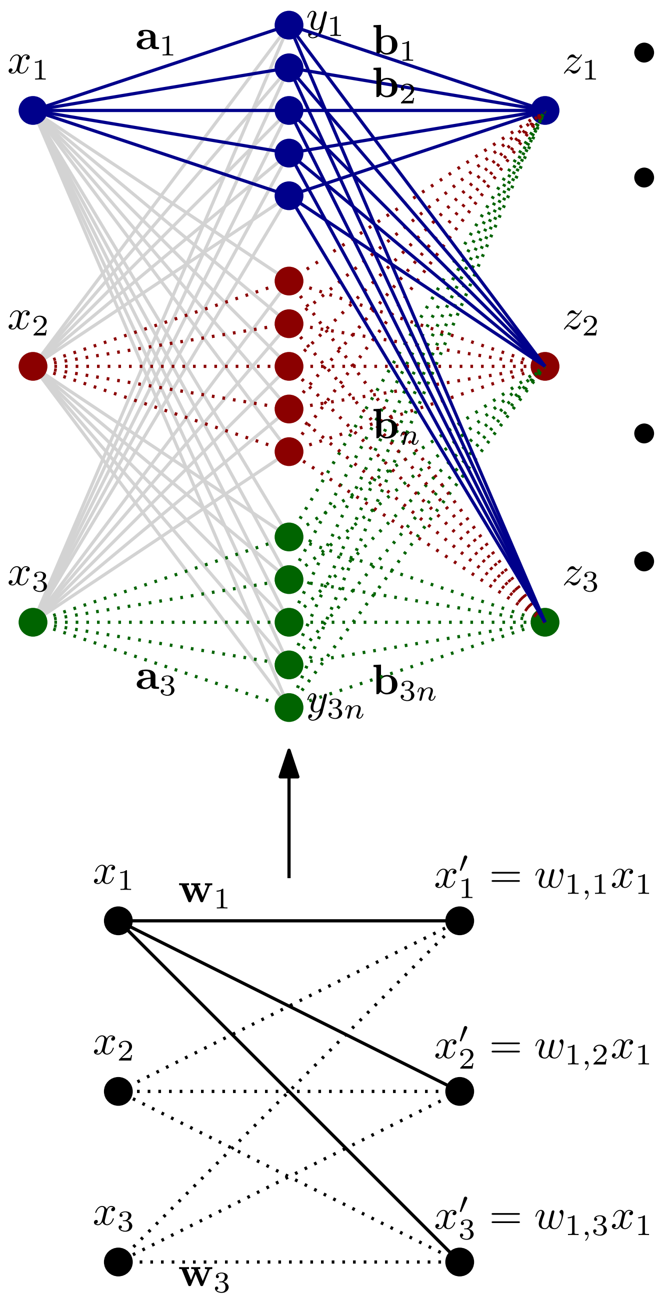


- $\mathbf{a}_i \sim \mathcal{N}(\mathbf{0}, I_n)$, $\mathbf{b}_i \sim \mathcal{N}(\mathbf{0}, I_d)$ (here, $d = 3$)
- For simplicity: **no ReLU**

$$\|x_1\mathbf{w}_1 - \sum_{i=1}^{n} x_1 a_{1,i}\mathbf{b}_i\|_\infty \leq |x_1| \|\mathbf{w}_1 - \sum_{i=1}^{n} a_{1,i}\mathbf{b}_i\|_\infty$$

- **Issue**: dependencies among entries of $a_{1,i}\mathbf{b}_i$!
- **Solution**:
  - for $S \subseteq [n]$, $X_S = \sum_{i \in S} a_{1,i}\mathbf{b}_i$

Apply MRSS for structured pruning

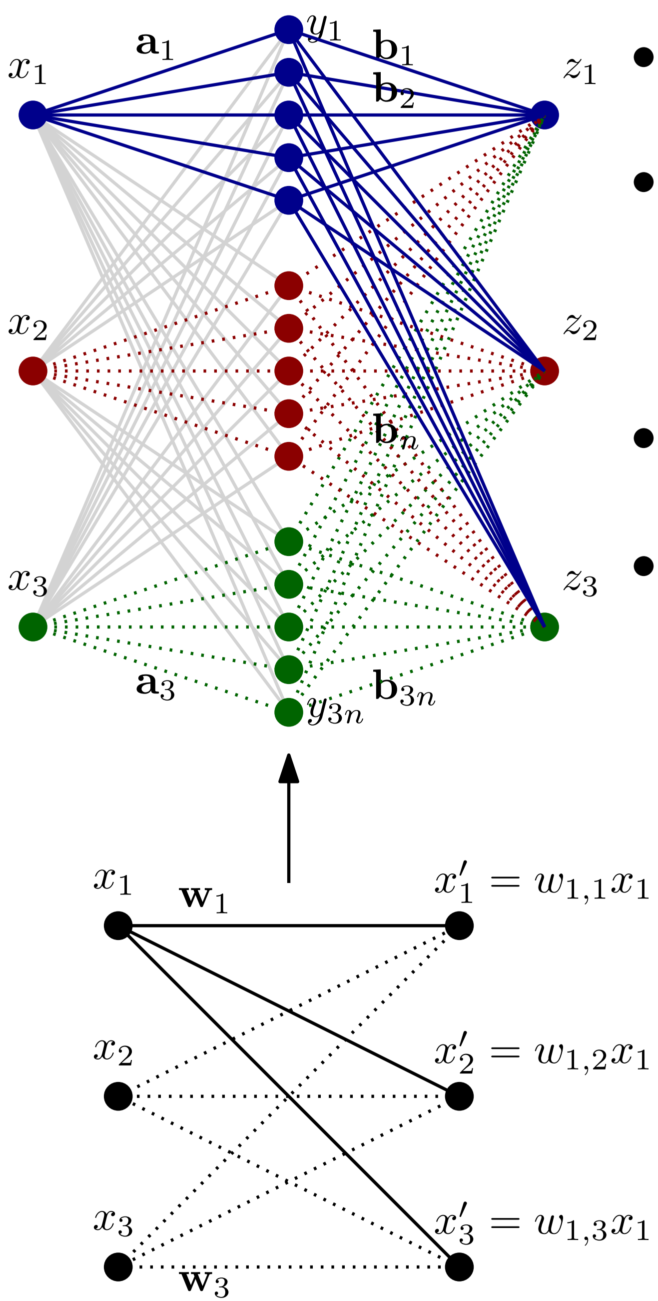


- $\mathbf{a}_i \sim \mathcal{N}(\mathbf{0}, I_n)$, $\mathbf{b}_i \sim \mathcal{N}(\mathbf{0}, I_d)$ (here, $d = 3$)
- For simplicity: **no ReLU**

$$\|x_1\mathbf{w}_1 - \textstyle\sum_{i=1}^n x_1 a_{1,i}\mathbf{b}_i\|_\infty \le |x_1|\|\mathbf{w}_1 - \textstyle\sum_{i=1}^n a_{1,i}\mathbf{b}_i\|_\infty$$

- **Issue**: dependencies among entries of $a_{1,i}\mathbf{b}_i$!
- **Solution**:
  - for $S \subseteq [n]$, $X_S = \sum_{i\in S} a_{1,i}\mathbf{b}_i$
  - conditional on $a_{1,i}$ for each $i \in S$, X_S is distributed as $\mathcal{N}(\mathbf{0}, \sum_{i\in S} a_{1,i}^2 \cdot I_d)$

Apply MRSS for structured pruning



- $\mathbf{a}_i \sim \mathcal{N}(\mathbf{0}, I_n)$, $\mathbf{b}_i \sim \mathcal{N}(\mathbf{0}, I_d)$ (here, $d = 3$)

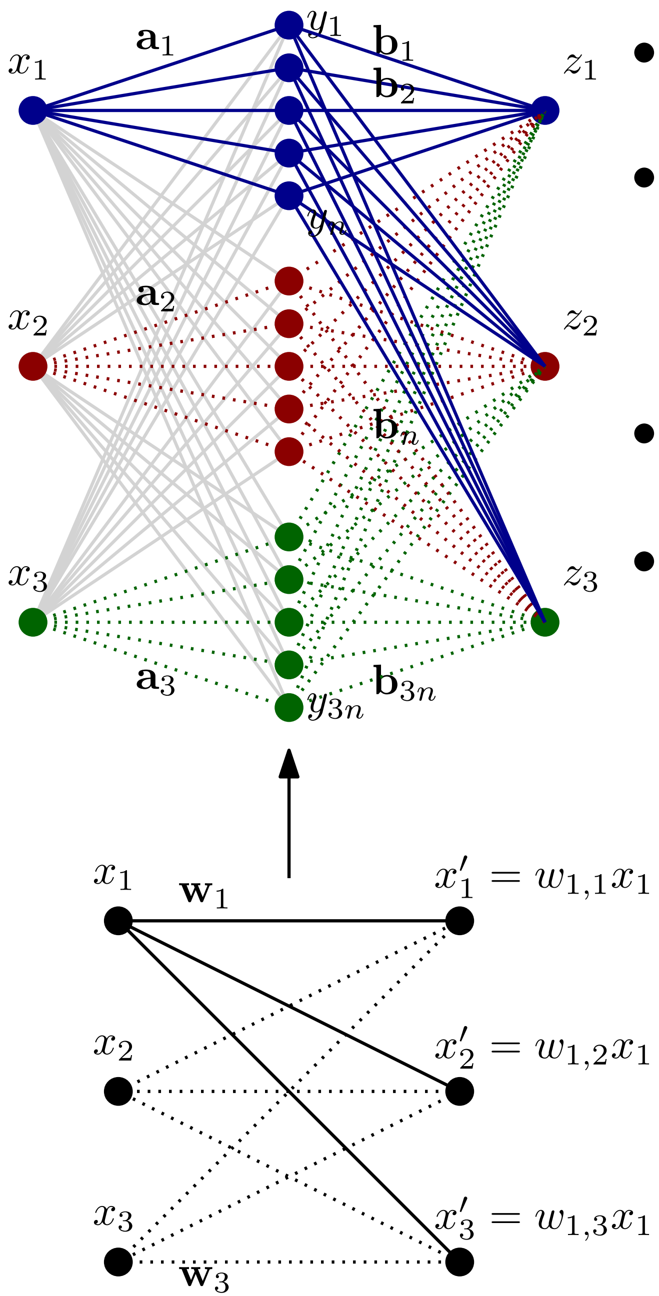
- For simplicity: **no ReLU**

$$\|x_1\mathbf{w}_1 - \sum_{i=1}^{n} x_1 a_{1,i}\mathbf{b}_i\|_\infty \leq |x_1|\|\mathbf{w}_1 - \sum_{i=1}^{n} a_{1,i}\mathbf{b}_i\|_\infty$$

- **Issue**: dependencies among entries of $a_{1,i}\mathbf{b}_i$!

- **Solution**:
  - for $S \subseteq [n]$, $X_S = \sum_{i \in S} a_{1,i}\mathbf{b}_i$
  - conditional on $a_{1,i}$ for each $i \in S$, X_S is distributed as $\mathcal{N}(\mathbf{0}, \sum_{i \in S} a_{1,i}^2 \cdot I_d)$
  - $\sum_{i \in S} a_{1,i}^2$ is a Chi-squared distribution: concentration inequalities!

Apply MRSS for structured pruning

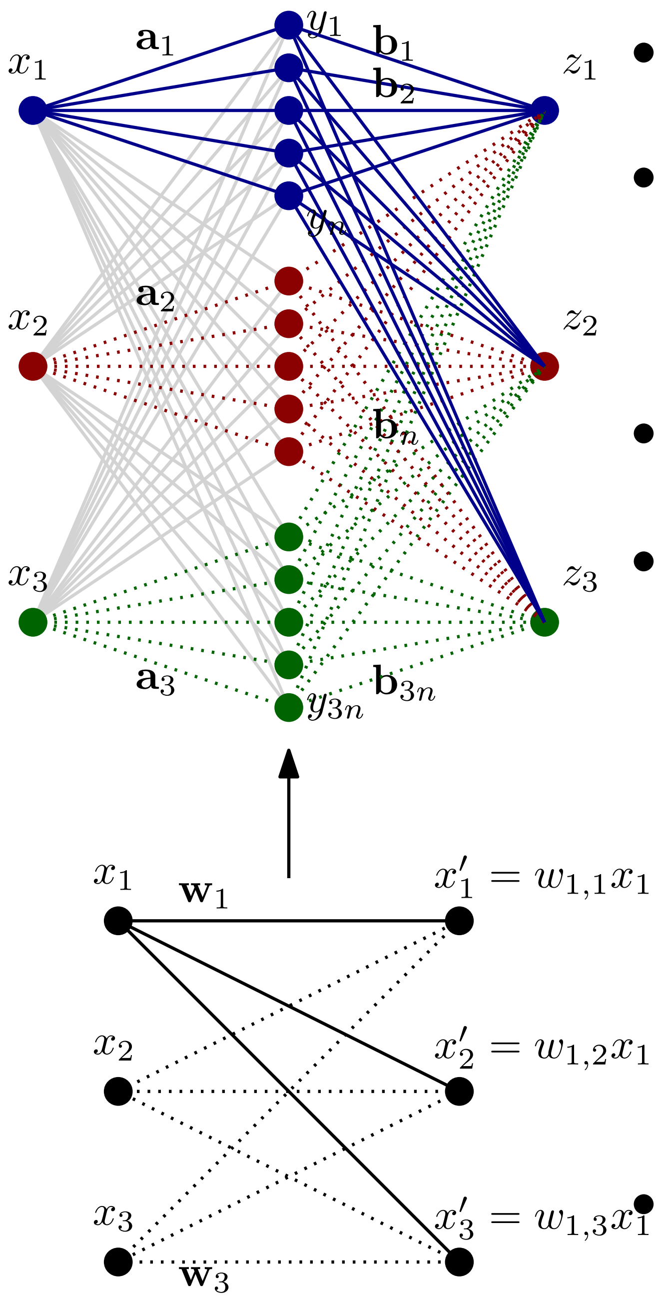


- $\mathbf{a}_i \sim \mathcal{N}(\mathbf{0}, I_n)$, $\mathbf{b}_i \sim \mathcal{N}(\mathbf{0}, I_d)$ (here, $d = 3$)
- For simplicity: **no ReLU**

$$\|x_1\mathbf{w}_1 - \sum_{i=1}^n x_1 a_{1,i}\mathbf{b}_i\|_\infty \leq |x_1|\|\mathbf{w}_1 - \sum_{i=1}^n a_{1,i}\mathbf{b}_i\|_\infty$$

- **Issue**: dependencies among entries of $a_{1,i}\mathbf{b}_i$!
- **Solution**:
  - for $S \subseteq [n]$, $X_S = \sum_{i\in S} a_{1,i}\mathbf{b}_i$
  - conditional on $a_{1,i}$ for each $i \in S$, X_S is distributed as $\mathcal{N}(\mathbf{0}, \sum_{i\in S} a_{1,i}^2 \cdot I_d)$
  - $\sum_{i\in S} a_{1,i}^2$ is a Chi-squared distribution: concentration inequalities!
  - *things do not change too much*

Apply MRSS for structured pruning



- $\mathbf{a}_i \sim \mathcal{N}(\mathbf{0}, I_n)$, $\mathbf{b}_i \sim \mathcal{N}(\mathbf{0}, I_d)$ (here, $d = 3$)

- For simplicity: **no ReLU**

$$\|x_1\mathbf{w}_1 - \sum_{i=1}^n x_1 a_{1,i}\mathbf{b}_i\|_\infty \leq |x_1|\|\mathbf{w}_1 - \sum_{i=1}^n a_{1,i}\mathbf{b}_i\|_\infty$$

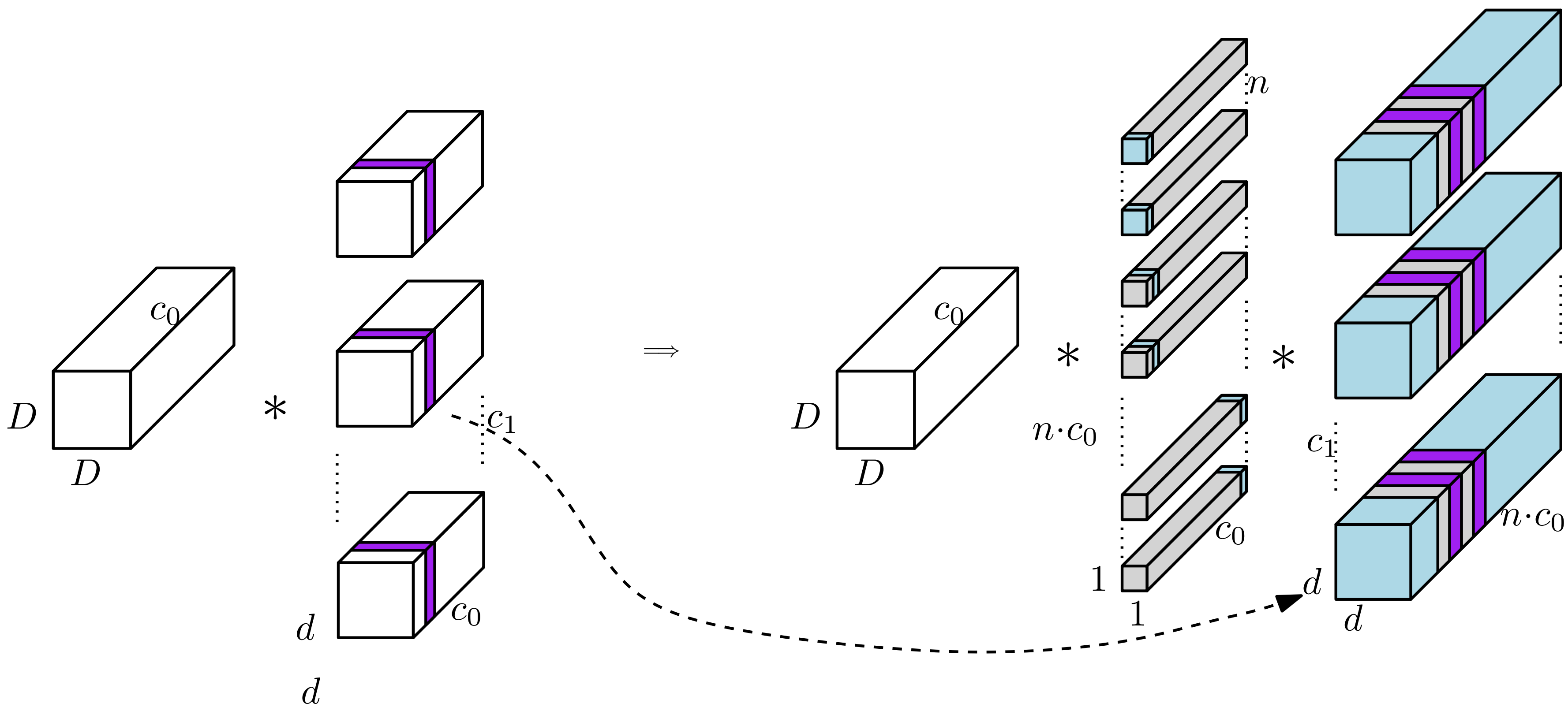
- **Issue**: dependencies among entries of $a_{1,i}\mathbf{b}_i$!

- **Solution**:
  - for $S \subseteq [n]$, $X_S = \sum_{i\in S} a_{1,i}\mathbf{b}_i$
  - conditional on $a_{1,i}$ for each $i \in S$, X_S is distributed as $\mathcal{N}(\mathbf{0}, \sum_{i\in S} a_{1,i}^2 \cdot I_d)$
  - $\sum_{i\in S} a_{1,i}^2$ is a Chi-squared distribution: concentration inequalities!
  - *things do not change too much*

- **Result**: $n \geq \text{poly}(d) \cdot \text{polylog}(d\ell/\varepsilon)$

Convolutional neural networks (CNNs)

- **Generality**: There are even some results for CNNs. What other architectures can the SLTH be applied to?



Zhou et al. Algorithm

[Zhou et al 2019, NeurIPS]

For each weight w_i learn a probability p_i

Zhou et al. Algorithm

[Zhou et al 2019, NeurIPS]

For each weight w_i learn a probability p_i

For all i, set $w_i' = w_i \cdot Bern(p_i)$

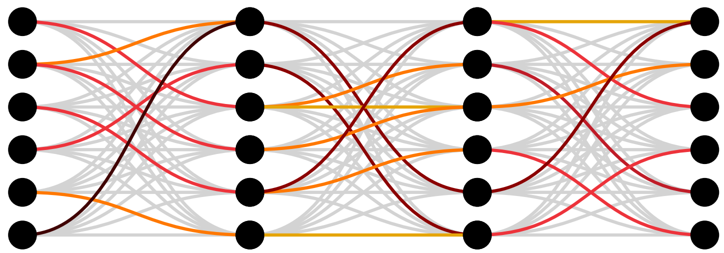
Zhou et al. Algorithm

[Zhou et al 2019, NeurIPS]

For each weight w_i learn a probability p_i

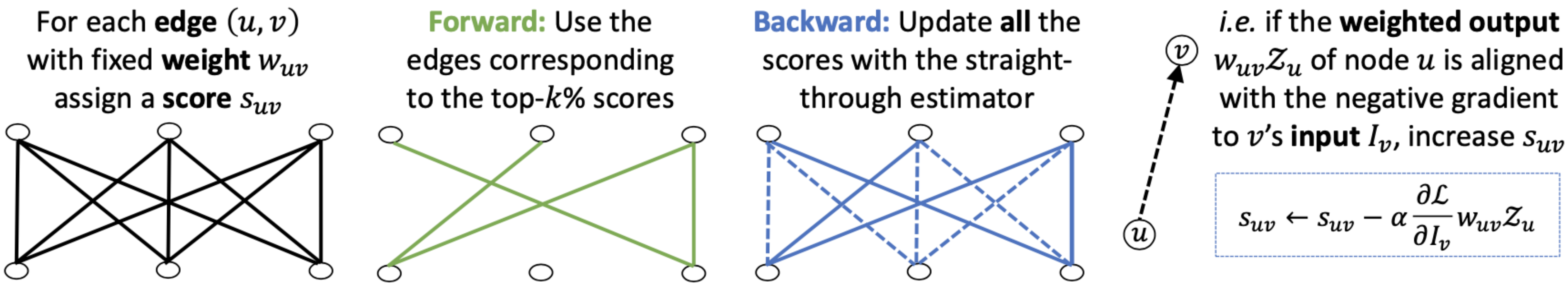
For all i, set $w_i' = w_i \cdot Bern(p_i)$

This leads to some robustness



Edge-Popup Algorithm

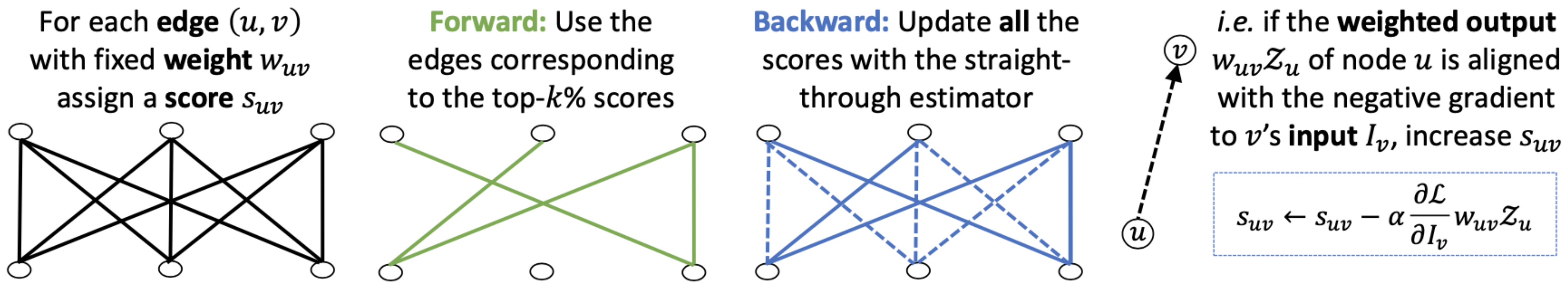
[Ramanujan et al 2020, CVPR]: Edge-popup Algorithm



I_v denotes the input to node v Z_v denote the output, $Z_v = \sigma(I_v)$

Edge-Popup Algorithm

[Ramanujan et al 2020, CVPR]: Edge-popup Algorithm

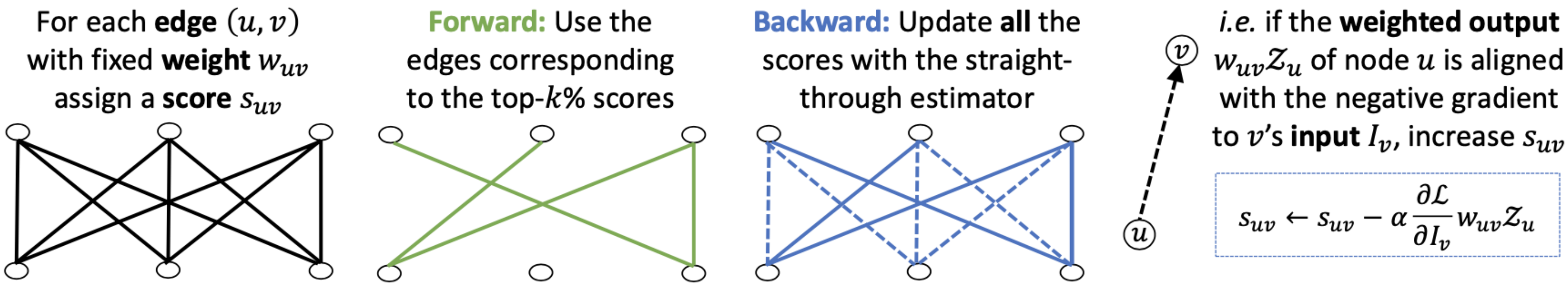


I_v denotes the input to node v Z_v denote the output, $Z_v = \sigma(I_v)$

Theorem: When edge (i, k) replaces (j, k) and the rest of the subnetwork remains fixed, then the loss decreases for the mini-batch (provided the loss is sufficiently smooth).

Edge-Popup Algorithm

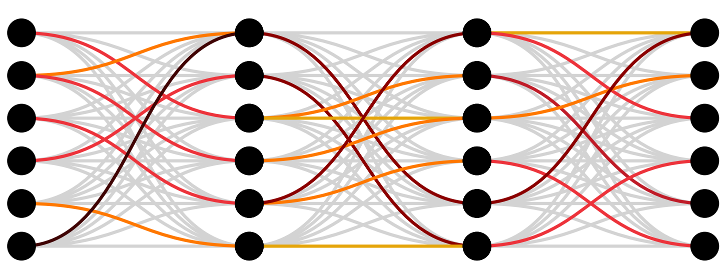
[Ramanujan et al 2020, CVPR]: Edge-popup Algorithm



I_v denotes the input to node v Z_v denote the output, $Z_v = \sigma(I_v)$

Theorem: When edge (i, k) replaces (j, k) and the rest of the subnetwork remains fixed, then the loss decreases for the mini-batch (provided the loss is sufficiently smooth).

Crucially, their the final network only has the a size of $k\%$



Lower bounds

- Almost all of the lower bounds focus on approximating a single neuron

Lower bounds

- Almost all of the lower bounds focus on approximating a single neuron
- At the core lies some packing argument: there are many linear functions that one might one to approximate. A network must be able to approximate any fixed function.

Lower bounds

- Almost all of the lower bounds focus on approximating a single neuron
- At the core lies some packing argument: there are many linear functions that one might one to approximate. A network must be able to approximate any fixed function.
- Even approximating the null-function seems ‘hard’

Lower bounds

- Almost all of the lower bounds focus on approximating a single neuron
- At the core lies some packing argument: there are many linear functions that one might one to approximate. A network must be able to approximate any fixed function.
- Even approximating the null-function seems ‘hard’
- **Open Problem**: None of the proofs have really moved beyond one layer. How much harder is it to approximate a deep neural network?

Overparameterization

- **SLTH**: Every sufficiently large network contains a subnetwork that does the job!

Overparameterization

- **SLTH**: Every sufficiently large network contains a subnetwork that does the job!
- **Issue**: All results consider a overparameterization (at least logarithmic)

Overparameterization

- **SLTH**: Every sufficiently large network contains a subnetwork that does the job!
- **Issue**: All results consider a overparameterization (at least logarithmic)
- **Unfair**: We aim to replicate a target network that is potentially optimal - real-world networks are not of optimal size.

Overparameterization

- **SLTH**: Every sufficiently large network contains a subnetwork that does the job!
- **Issue**: All results consider a overparameterization (at least logarithmic)
- **Unfair**: We aim to replicate a target network that is potentially optimal - real-world networks are not of optimal size.
- **Open Problem**: How can we make the comparison fairer?

Conclusions

- **LTH**: Every network contains a sub-network that can be trained in isolation to achieve the same test accuracy

Conclusions

- **LTH**: Every network contains a sub-network that can be trained in isolation to achieve the same test accuracy
- **SLTH**: Every sufficiently large network contains a subnetwork that does the job!

Conclusions

- **LTH**: Every network contains a sub-network that can be trained in isolation to achieve the same test accuracy
- **SLTH**: Every sufficiently large network contains a subnetwork that does the job!
- **Open Problem (1)**: We know the lottery tickets exist. But how can we find them efficiently?

Conclusions

- **LTH**: Every network contains a sub-network that can be trained in isolation to achieve the same test accuracy
- **SLTH**: Every sufficiently large network contains a subnetwork that does the job!
- **Open Problem (1)**: We know the lottery tickets exist. But how can we find them efficiently?
- **Open Problem (2)**: How hard is neuron-pruning?

Conclusions

- **LTH**: Every network contains a sub-network that can be trained in isolation to achieve the same test accuracy
- **SLTH**: Every sufficiently large network contains a subnetwork that does the job!
- **Open Problem (1)**: We know the lottery tickets exist. But how can we find them efficiently?
- **Open Problem (2)**: How hard is neuron-pruning?
- **Open Problem (3)**: How to apply the SLTH to transformers?

Conclusions

- **LTH**: Every network contains a sub-network that can be trained in isolation to achieve the same test accuracy
- **SLTH**: Every sufficiently large network contains a subnetwork that does the job!
- **Open Problem (1)**: We know the lottery tickets exist. But how can we find them efficiently?
- **Open Problem (2)**: How hard is neuron-pruning?
- **Open Problem (3)**: How to apply the SLTH to transformers?

I'm on sabbatical soon, if you want to work on this, let me know :)

Thank you!

RSS proof overview

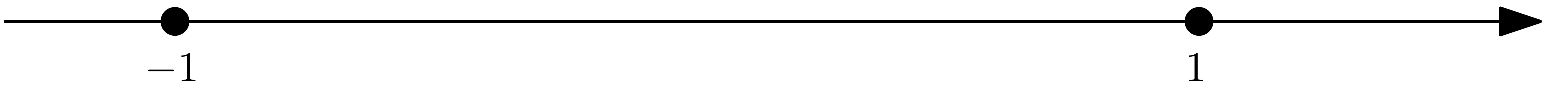
- [Lueker 1998; da Cunha et al. 2023]

RSS proof overview

- [Lueker 1998; da Cunha et al. 2023]

Specific instance of RSSP

- $X_1, \ldots, X_n$ uniform random variables over $[-1, 1]$
- Error parameter $\varepsilon > 0$

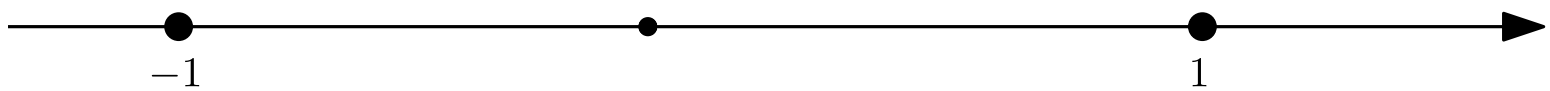


RSS proof overview

- [Lueker 1998; da Cunha et al. 2023]

Specific instance of RSSP

- $X_1, \ldots, X_n$ uniform random variables over $[-1, 1]$
- Error parameter $\varepsilon > 0$

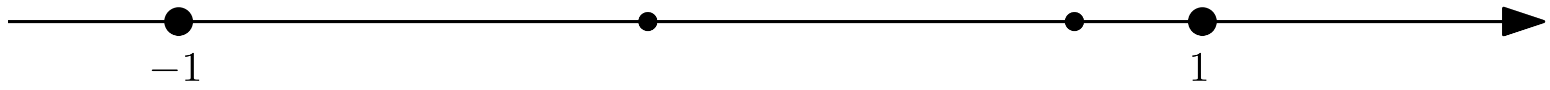


RSS proof overview

- [Lueker 1998; da Cunha et al. 2023]

Specific instance of RSSP

- $X_1, \ldots, X_n$ uniform random variables over $[-1, 1]$
- Error parameter $\varepsilon > 0$

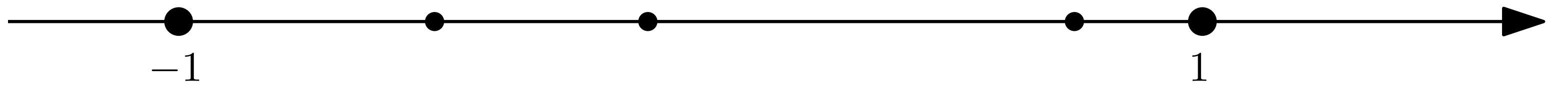


RSS proof overview

- [Lueker 1998; da Cunha et al. 2023]

Specific instance of RSSP

- $X_1, \ldots, X_n$ uniform random variables over $[-1, 1]$
- Error parameter $\varepsilon > 0$

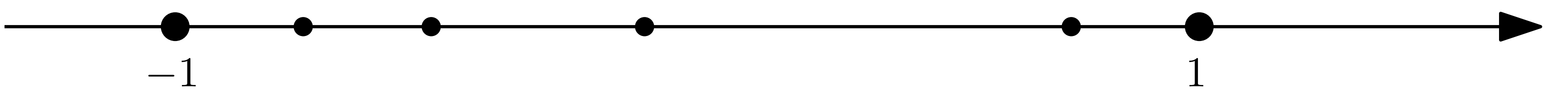


RSS proof overview

- [Lueker 1998; da Cunha et al. 2023]

Specific instance of RSSP

- $X_1, \ldots, X_n$ uniform random variables over $[-1, 1]$
- Error parameter $\varepsilon > 0$

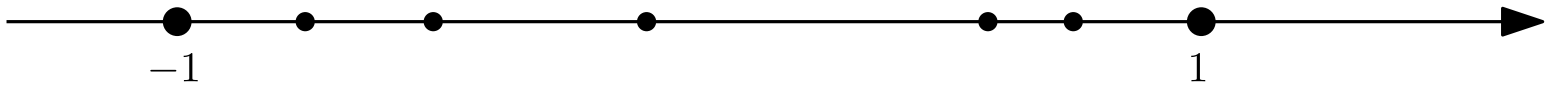


RSS proof overview

- [Lueker 1998; da Cunha et al. 2023]

Specific instance of RSSP

- $X_1, \ldots, X_n$ uniform random variables over $[-1, 1]$
- Error parameter $\varepsilon > 0$

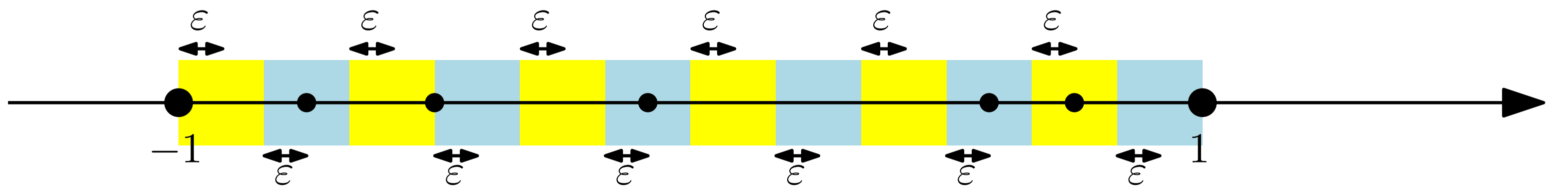


RSS proof overview

- [Lueker 1998; da Cunha et al. 2023]

Specific instance of RSSP

- $X_1, \ldots, X_n$ uniform random variables over $[-1, 1]$
- Error parameter $\varepsilon > 0$
- Approximate the whole interval

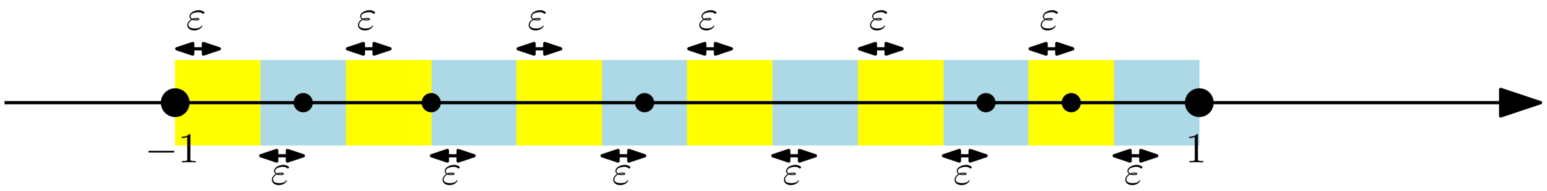


RSS proof overview

- [Lueker 1998; da Cunha et al. 2023]

Specific instance of RSSP

- $X_1, \ldots, X_n$ uniform random variables over $[-1, 1]$
- Error parameter $\varepsilon > 0$
- Approximate the whole interval



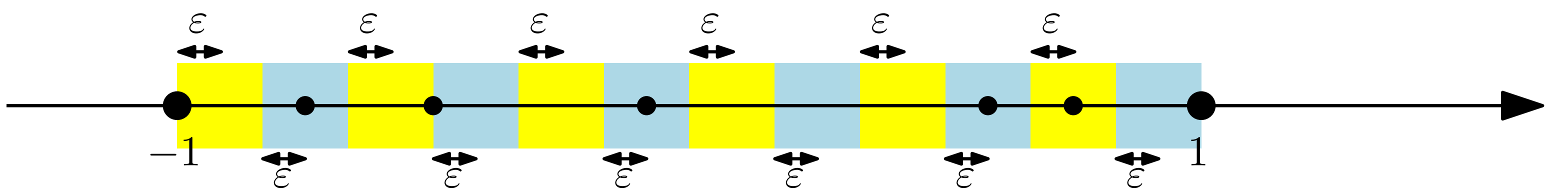
Consider $f_t(x) = \begin{cases} 1 & \text{if } x \in [-1,1] \text{ and } \exists S \subseteq [t] : \left| x - \sum_{i \in S} X_i \right| < 2\varepsilon \\ 0 & \text{otherwise} \end{cases}$

RSS proof overview

- [Lueker 1998; da Cunha et al. 2023]

Specific instance of RSSP

- $X_1, \ldots, X_n$ uniform random variables over $[-1, 1]$
- Error parameter $\varepsilon > 0$
- Approximate the whole interval



Consider $f_t(x) = \begin{cases} 1 & \text{if } x \in [-1, 1] \text{ and } \exists S \subseteq [t] : \left| x - \sum_{i \in S} X_i \right| < 2\varepsilon \\ 0 & \text{otherwise} \end{cases}$

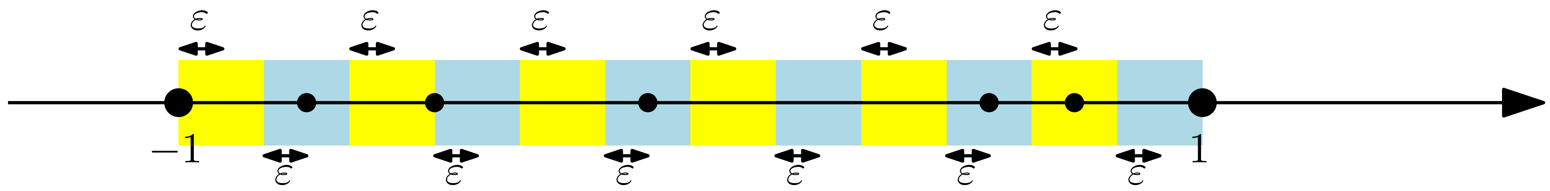
$v_t = \frac{1}{2} \int_{-1}^{1} f_t(x) \, \mathrm{d}x$ keeps track of the approximated volume

RSS proof overview

- [Lueker 1998; da Cunha et al. 2023]

Specific instance of RSSP

- $X_1, \ldots, X_n$ uniform random variables over $[-1, 1]$
- Error parameter $\varepsilon > 0$
- Approximate the whole interval



Consider $f_t(x) = \begin{cases} 1 & \text{if } x \in [-1,1] \text{ and } \exists S \subseteq [t] : \left| x - \sum_{i \in S} X_i \right| < 2\varepsilon \\ 0 & \text{otherwise} \end{cases}$

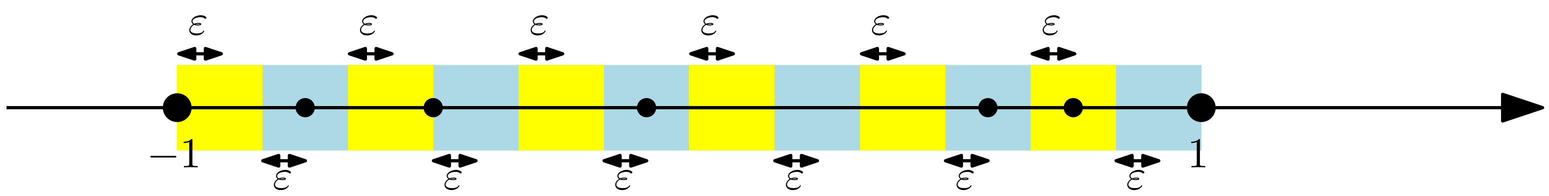
$v_t = \frac{1}{2} \int_{-1}^{1} f_t(x)\,\mathrm{d}x$ keeps track of the approximated volume

RSS proof overview

- [Lueker 1998; da Cunha et al. 2023]

Specific instance of RSSP

- $X_1, \ldots, X_n$ uniform random variables over $[-1, 1]$
- Error parameter $\varepsilon > 0$
- Approximate the whole interval



Consider $f_t(x) = \begin{cases} 1 & \text{if } x \in [-1,1] \text{ and } \exists S \subseteq [t] : \left|x - \sum_{i \in S} X_i\right| < 2\varepsilon \\ 0 & \text{otherwise} \end{cases}$

$v_t = \frac{1}{2} \int_{-1}^{1} f_t(x)\, \mathrm{d}x$ keeps track of the approximated volume

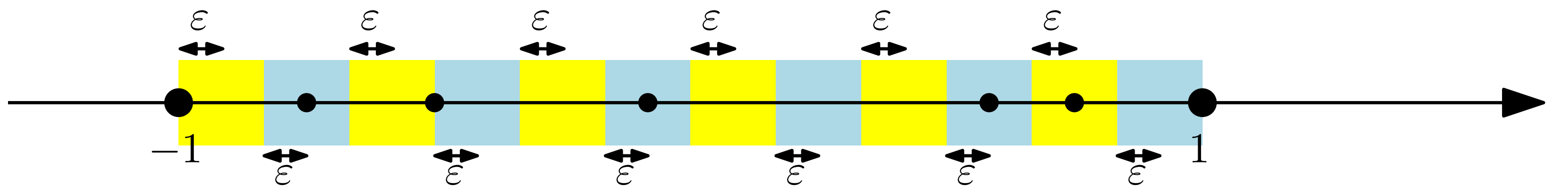
$$f_{t+1}(z) = f_t(z) + (1 - f_t(z))\, f_t(z - X_{t+1}).$$

RSS proof overview

- [Lueker 1998; da Cunha et al. 2023]

Specific instance of RSSP

- $X_1, \ldots, X_n$ uniform random variables over $[-1, 1]$
- Error parameter $\varepsilon > 0$
- Approximate the whole interval



Consider $f_t(x) = \begin{cases} 1 & \text{if } x \in [-1,1] \text{ and } \exists S \subseteq [t] : \left|x - \sum_{i \in S} X_i\right| < 2\varepsilon \\ 0 & \text{otherwise} \end{cases}$

$v_t = \frac{1}{2} \int_{-1}^{1} f_t(x)\,\mathrm{d}x$ keeps track of the approximated volume

$$f_{t+1}(z) = f_t(z) + (1 - f_t(z))\, f_t(z - X_{t+1}).$$

For all $0 \leq t < n$, it holds that $\mathbb{E}\left[v_{t+1} \mid X_1, \ldots, X_t\right] \geq v_t \left[1 + \frac{1}{4}(1 - v_t)\right].$