The Strong Lottery Ticket Hypothesis and the Random Subset Sum Problem

Aalto University

Francesco d'Amore

Based on joint work with A. da Cunha and E. Natale [NeurIPS 2023]

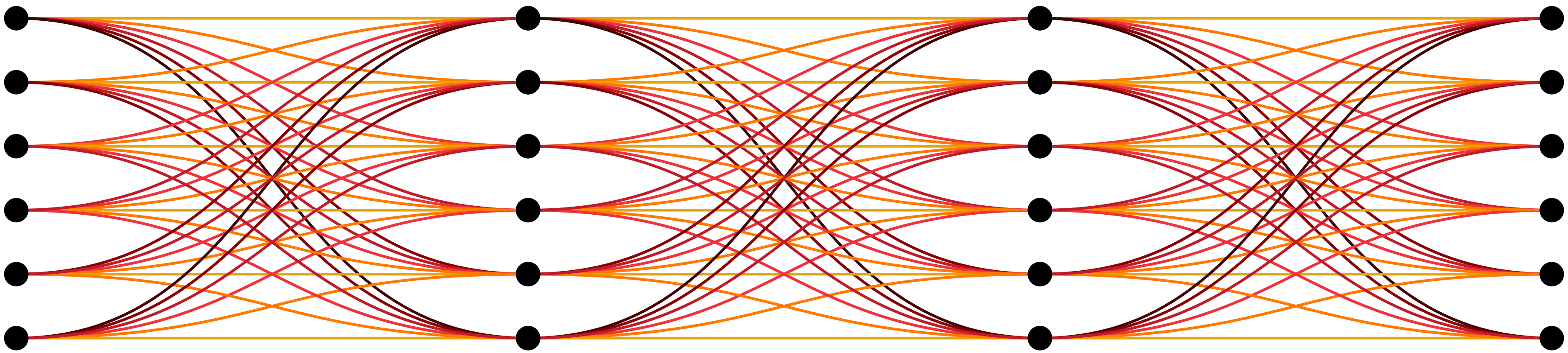
Aalto Theory Seminar

15 November 2023

Artificial neural networks are large

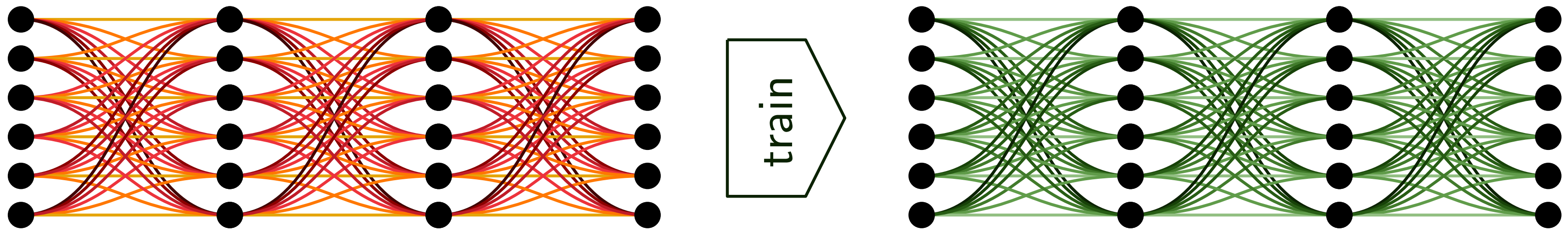
Usually ranging from millions to hundreds of billions parameters

- 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]



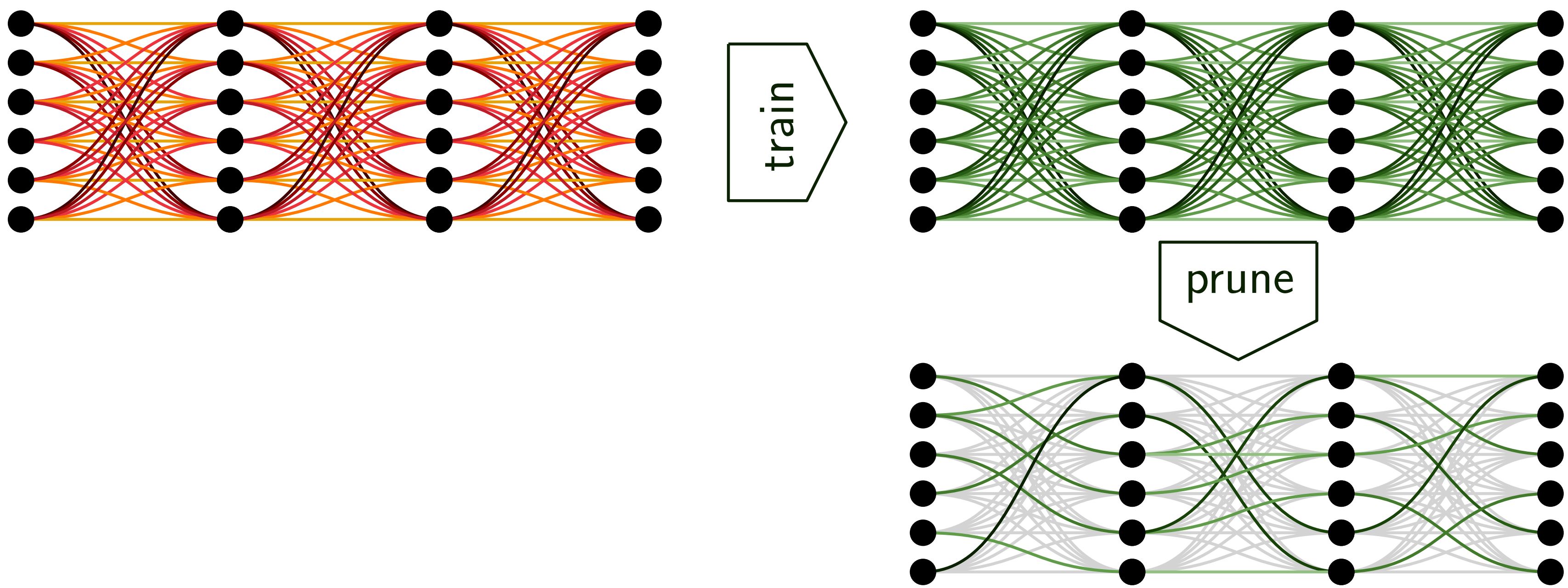
Training is expensive

- Resource intensive
- Good results
- Resulting network still large



Training is expensive

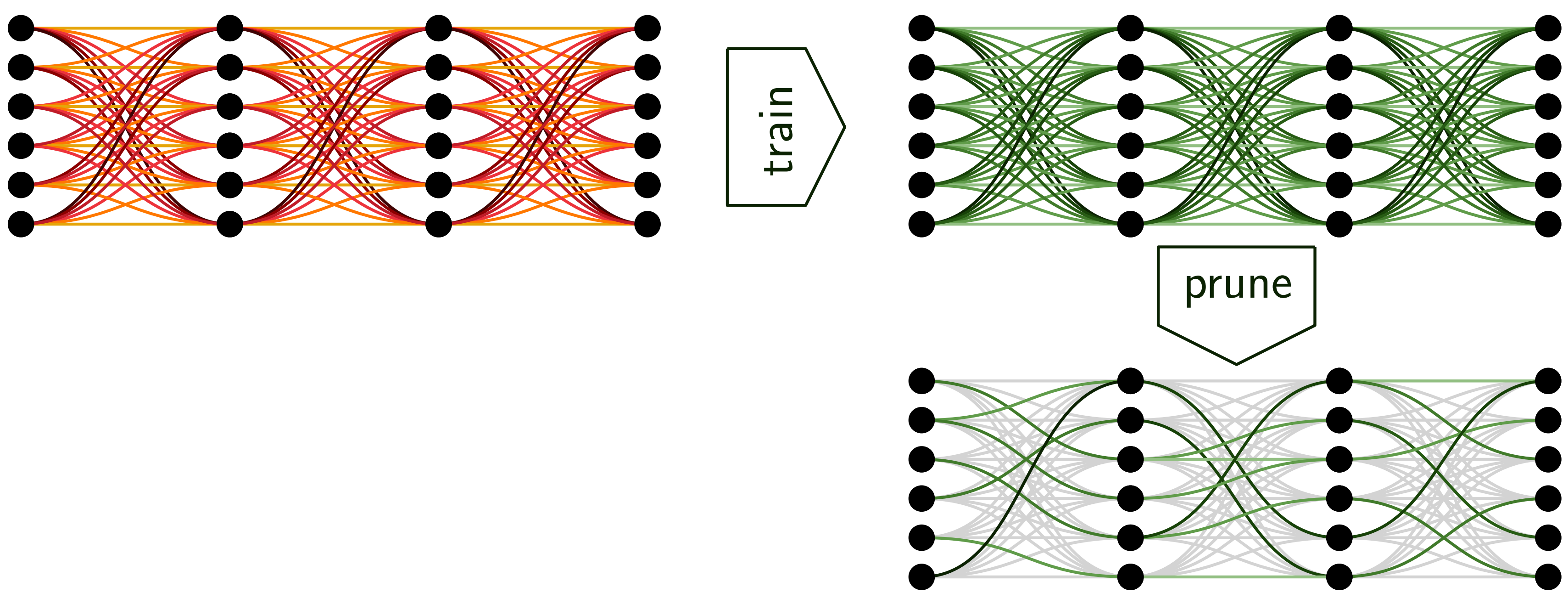
- Resource intensive
- Good results
- Resulting network still large
- Removing edges (pruning) works well



Training is expensive

- Resource intensive
- Good results
- Resulting network still large

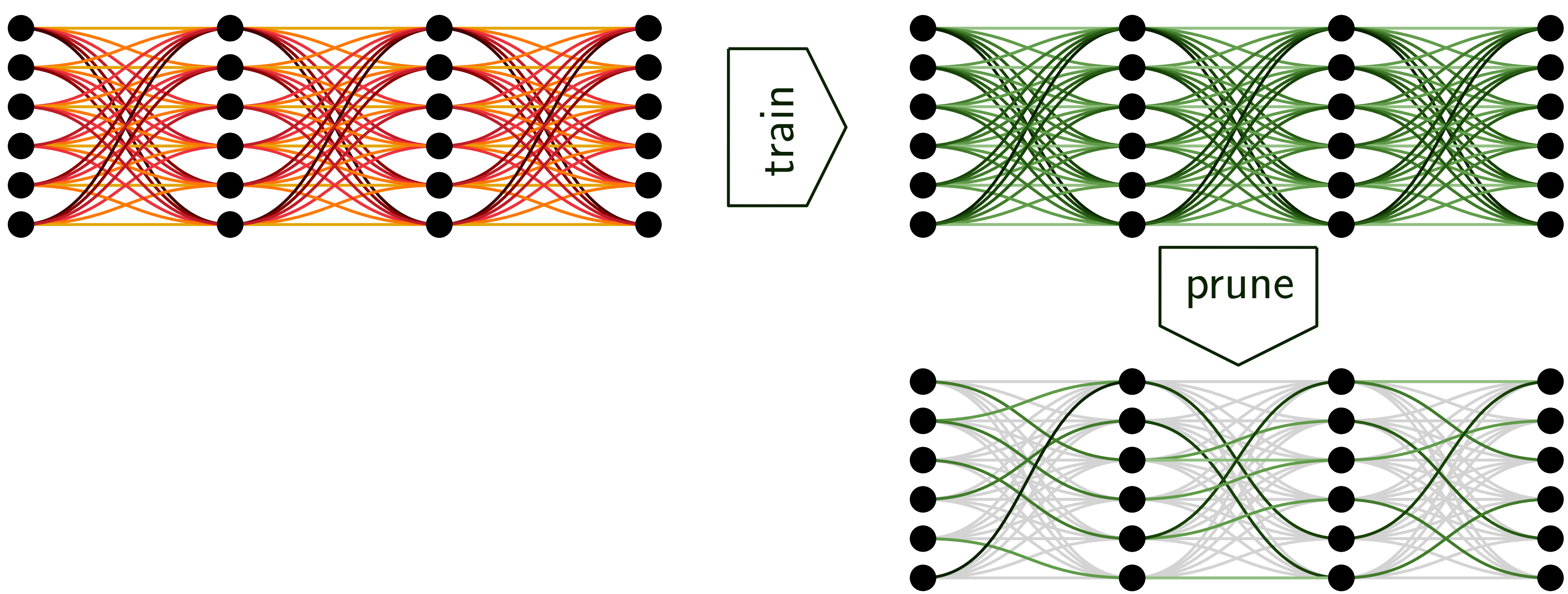
- Removing edges (pruning) works well
- Pruning $\sim 60 - 80\%$ of the edges can lead to better accuracies [Diffenderfer and Kailkhura 2021]



Training is expensive

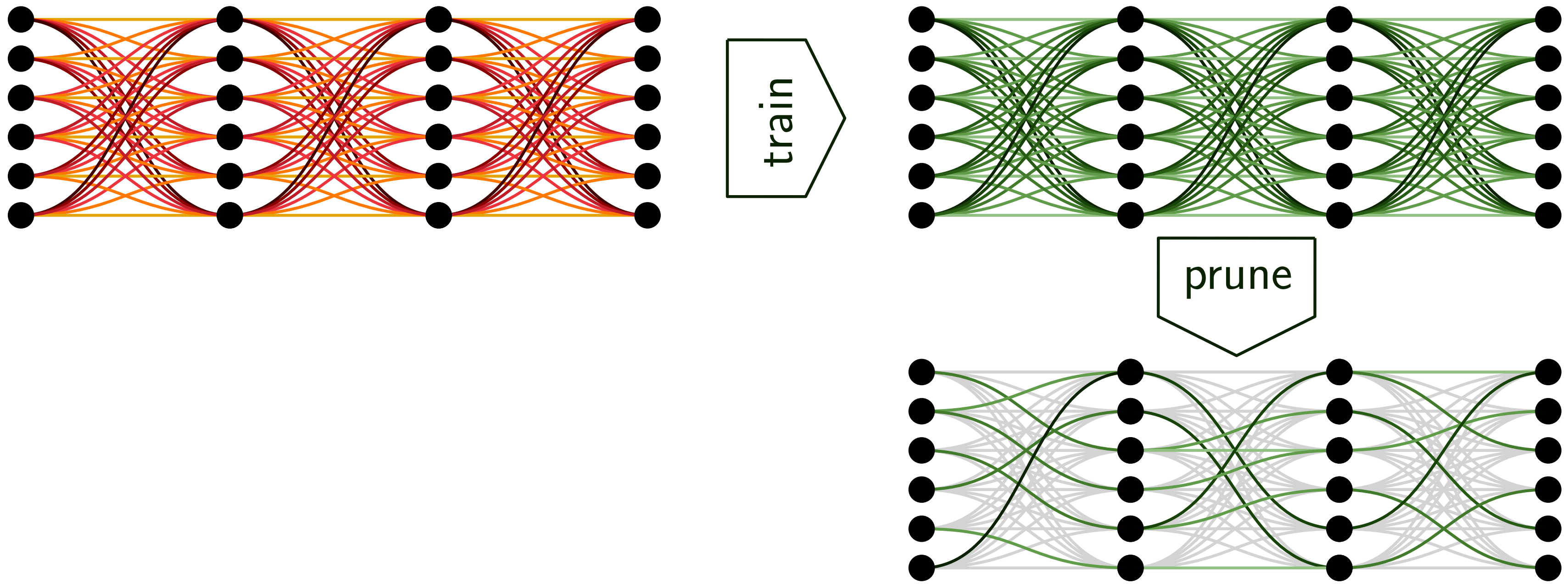
- Resource intensive
- Good results
- Resulting network still large

- Removing edges (pruning) 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]



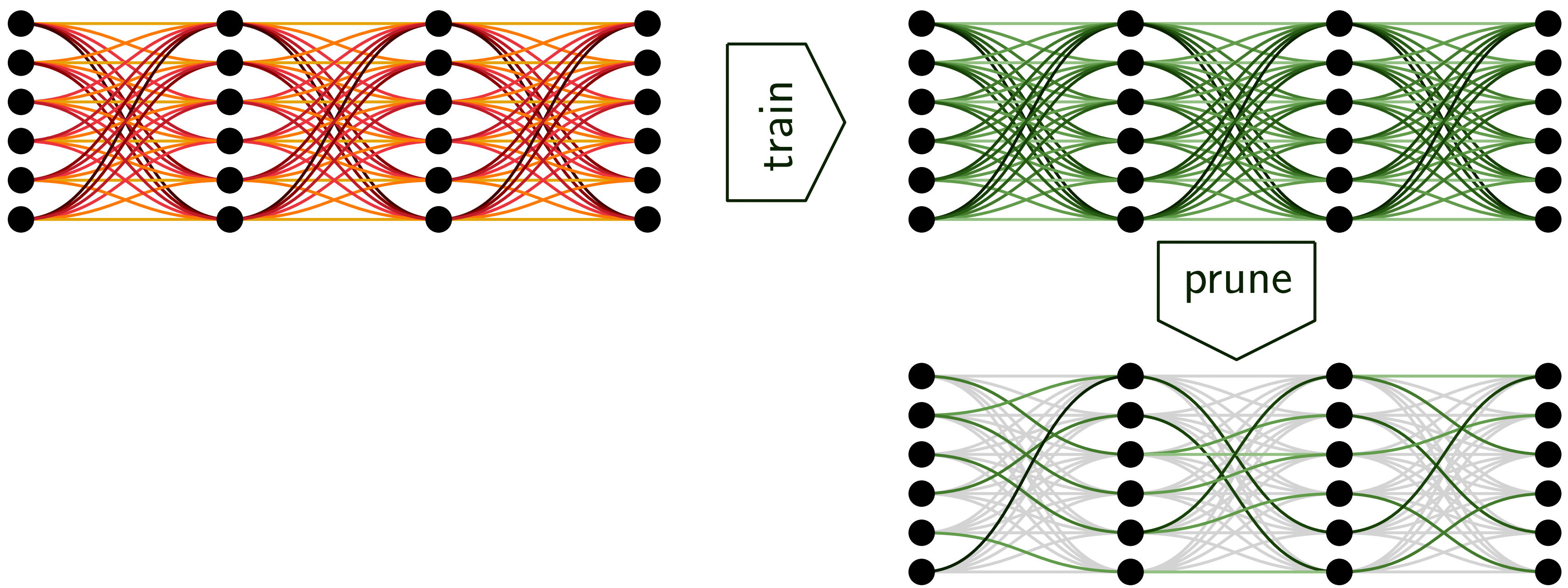
What if we train the tiny one?

- Maybe, we can avoid the effort of dense training



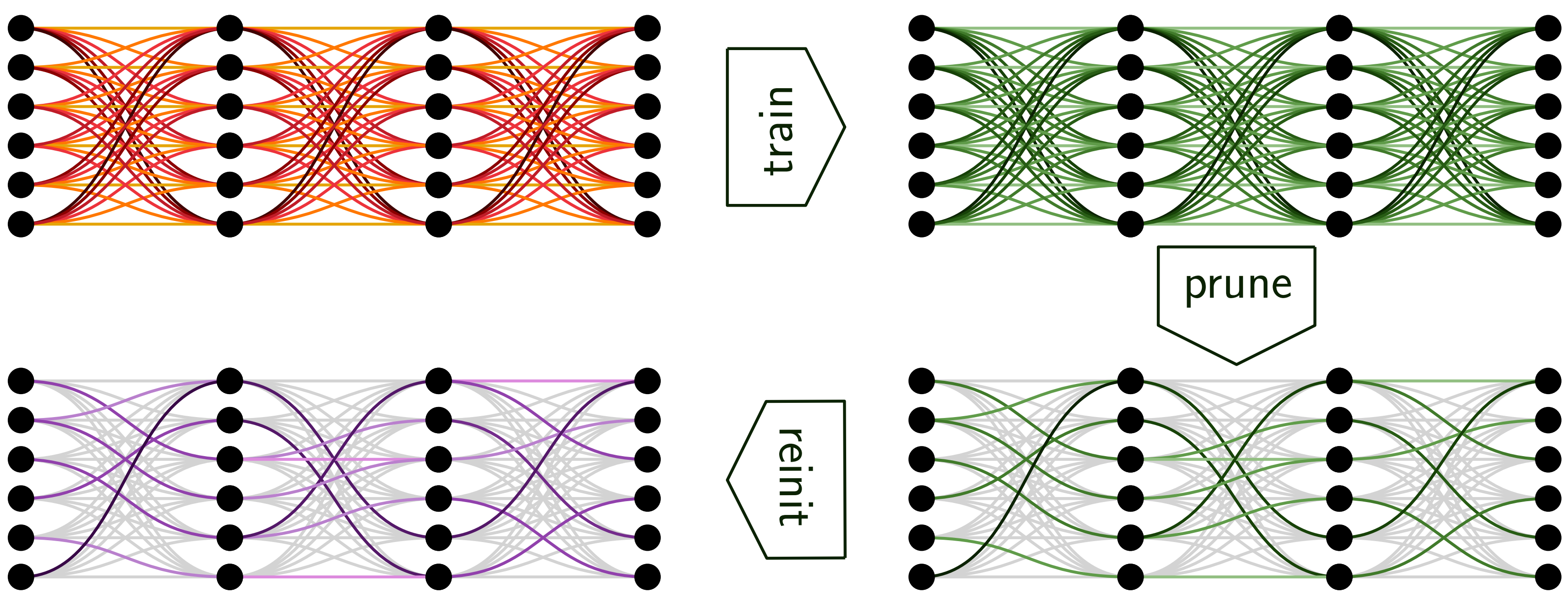
What if we train the tiny one?

- Maybe, we can avoid the effort of dense training
- Let's test the subnetwork by retraining it



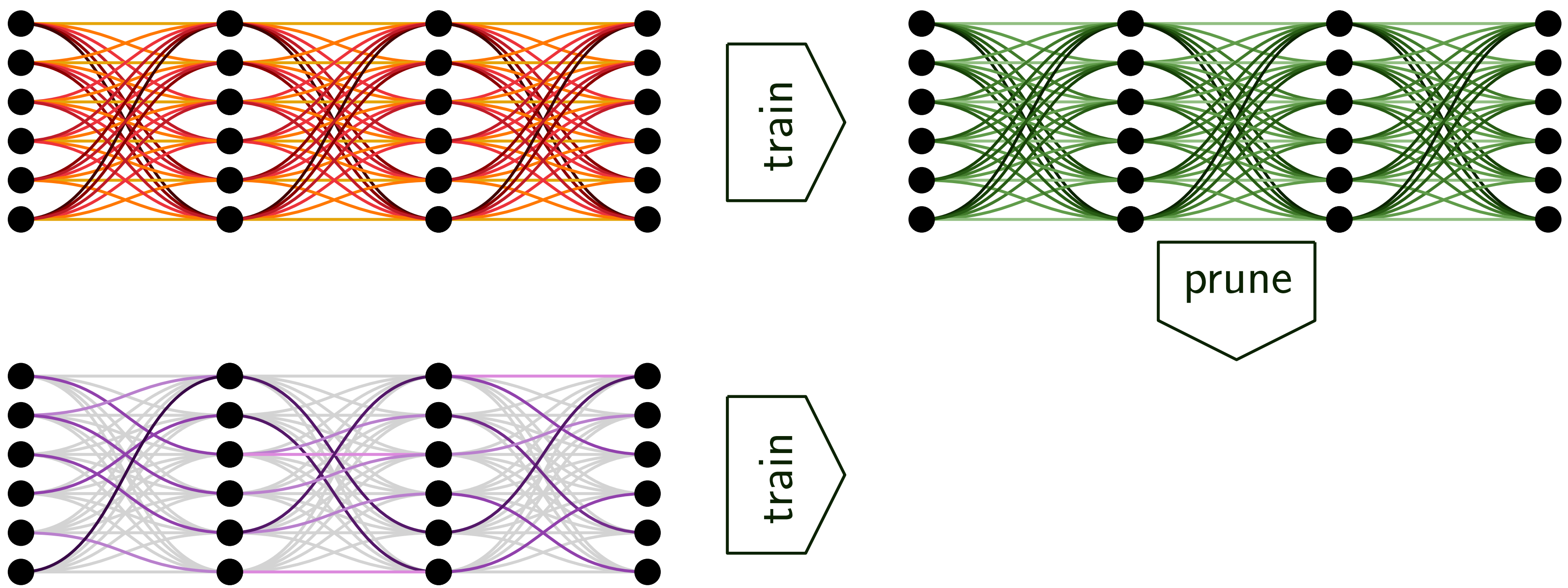
What if we train the tiny one?

- Maybe, we can avoid the effort of dense training
- Let's test the subnetwork by retraining it
    - Reinitialize



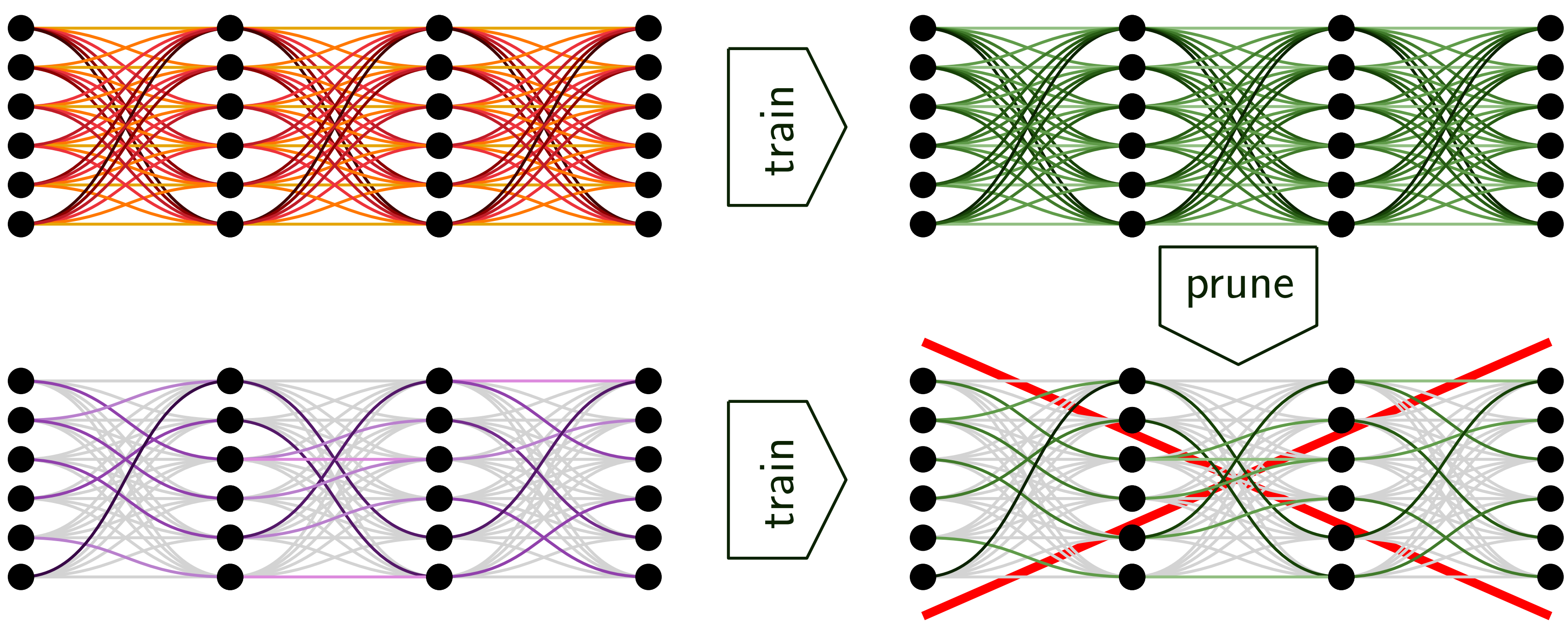
What if we train the tiny one?

- Maybe, we can avoid the effort of dense training
- Let's test the subnetwork by retraining it
    - Reinitialize
    - Train



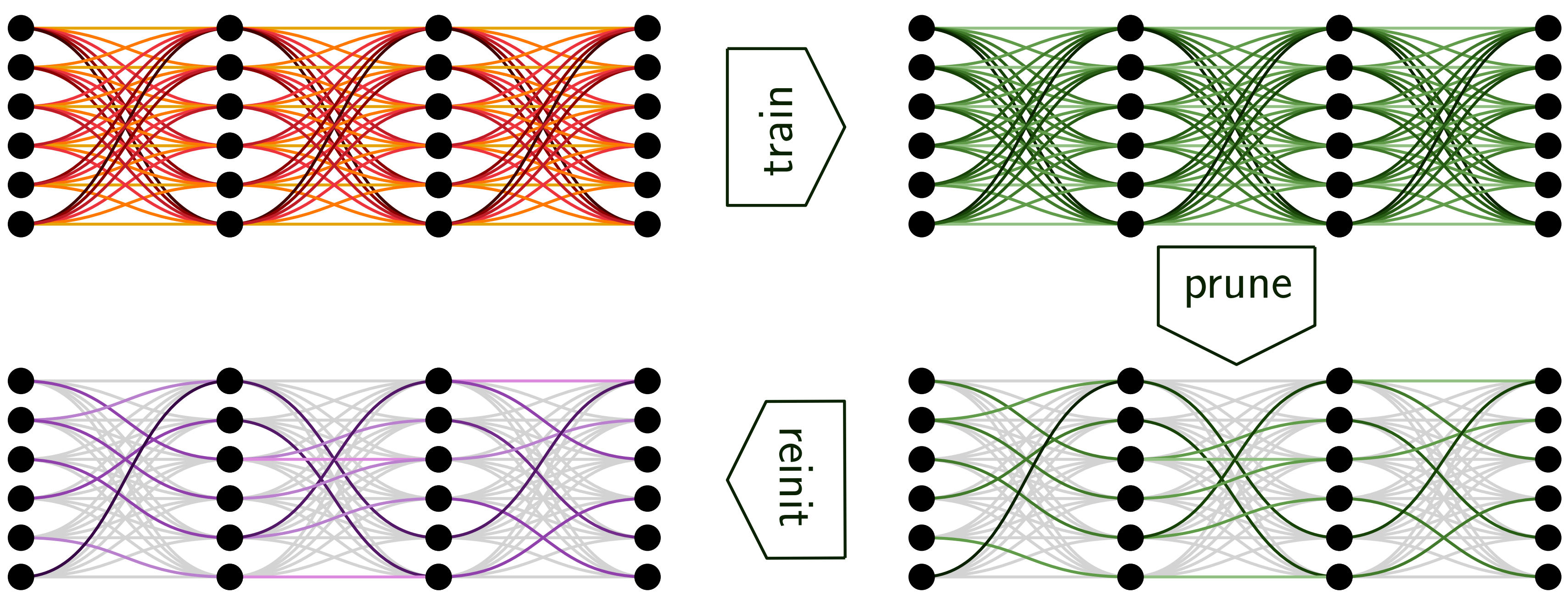
What if we train the tiny one?

- Maybe, we can avoid the effort of dense training
- 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

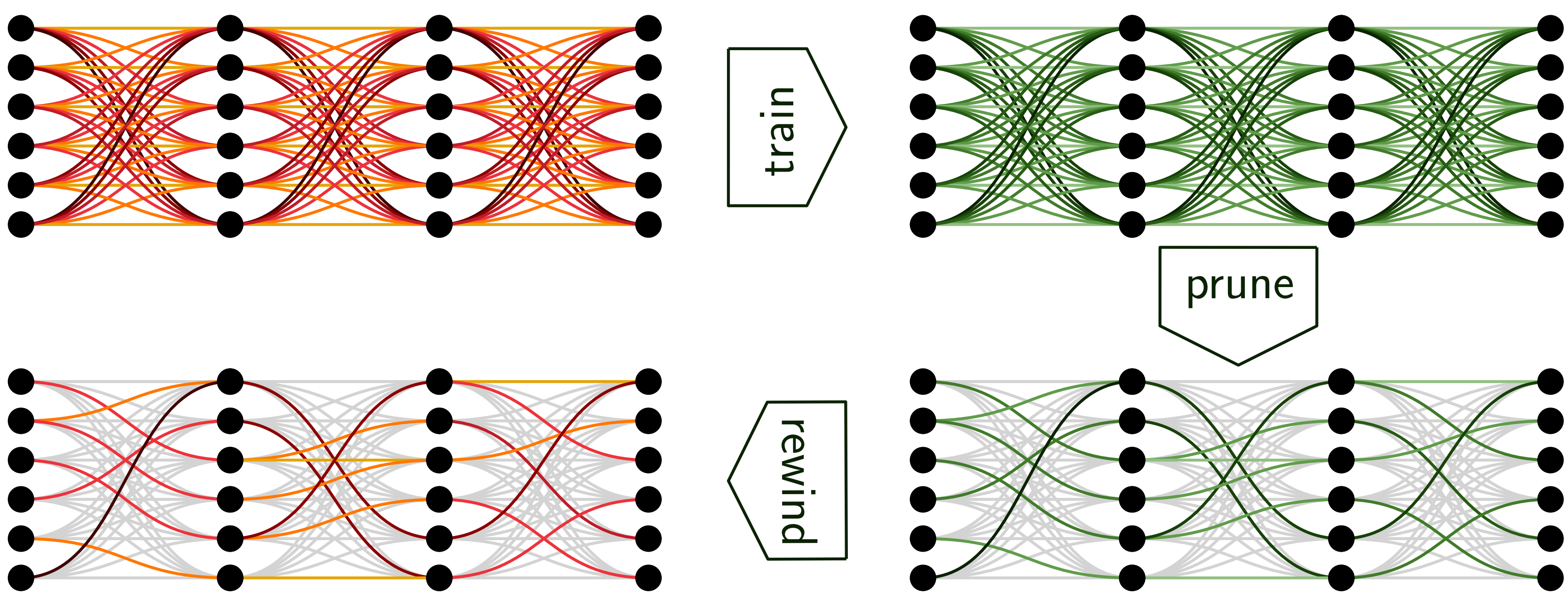


Not reinitialization, rewind instead

- Starting from a random point might be too much

[Frankle and Carbin ICLR '19]

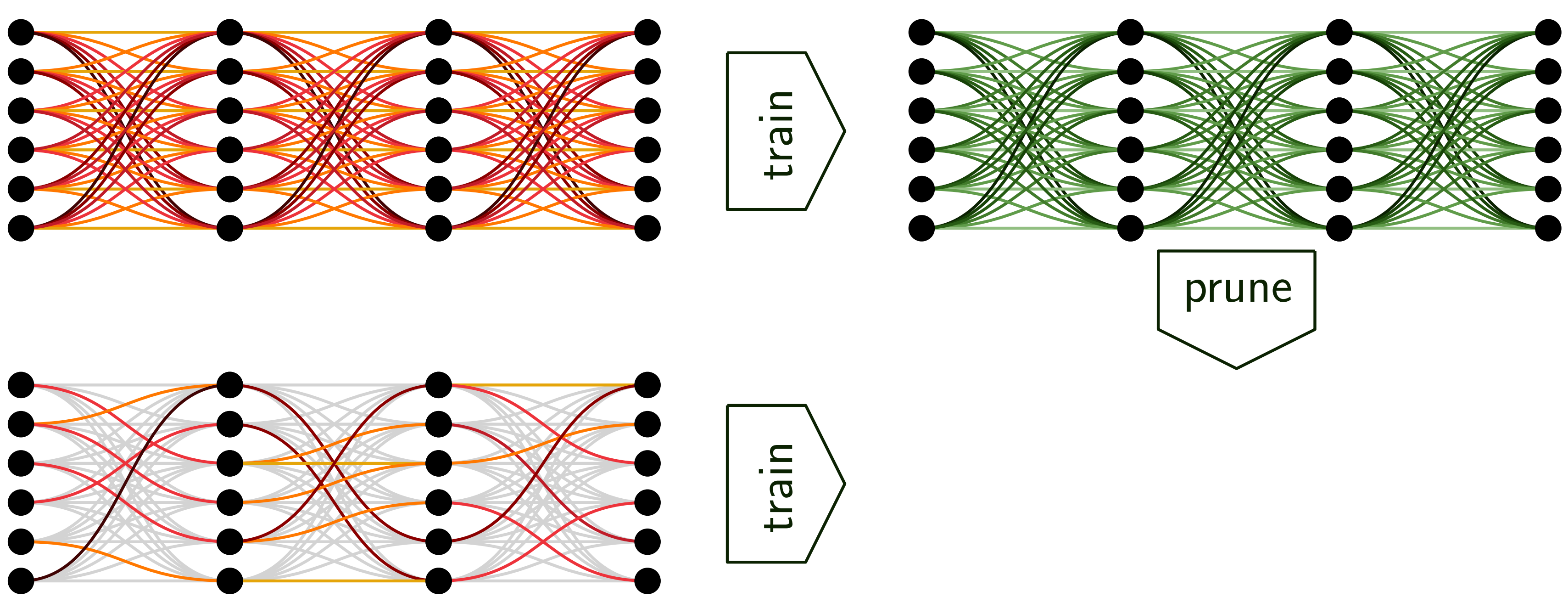
- Rewind instead



Not reinitialization, rewind instead

- Starting from a random point might be too much

[Frankle and Carbin ICLR '19]
- Rewind instead
- Training is efficient: 10%-20% of the original size

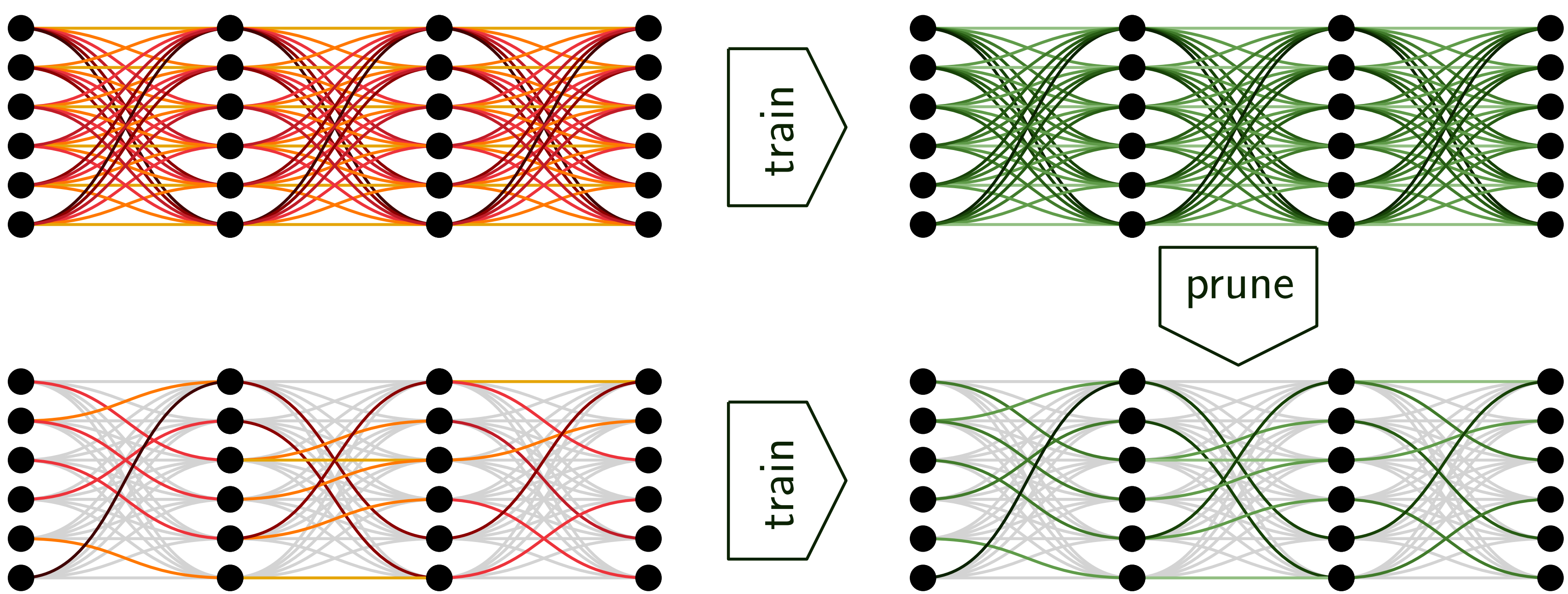


Not reinitialization, rewind instead

- Starting from a random point might be too much

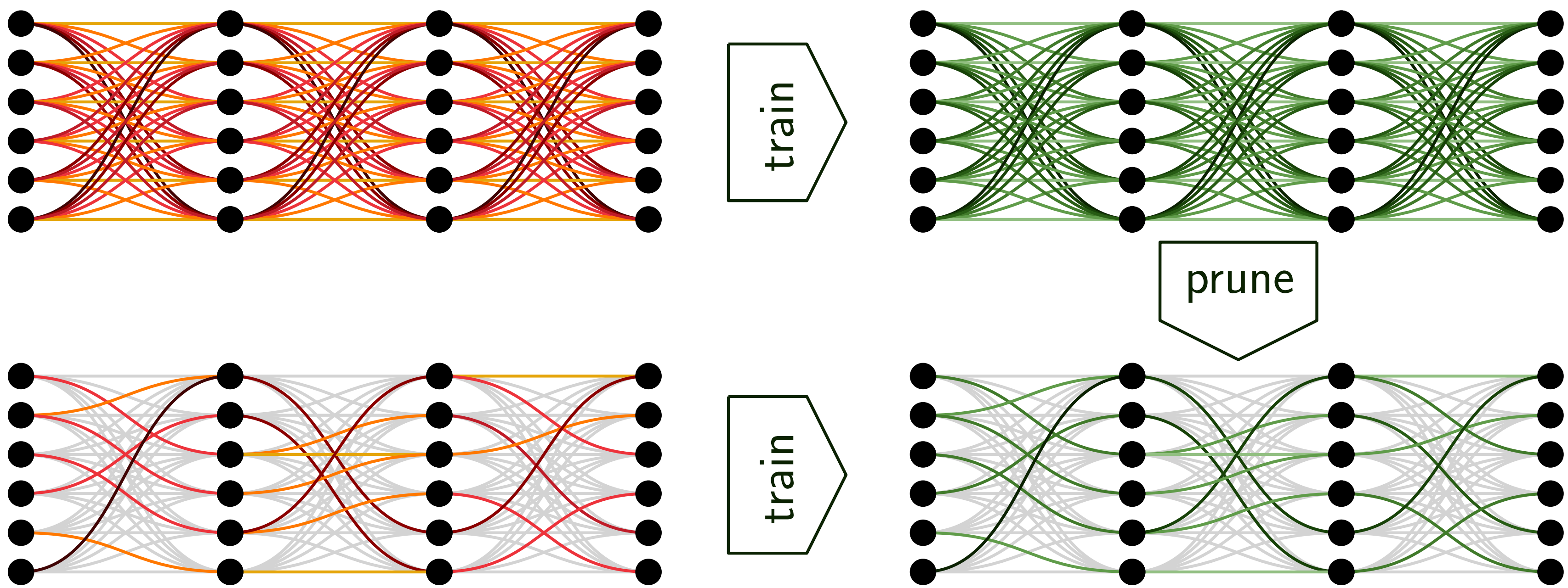
[Frankle and Carbin ICLR '19]

- Rewind instead
- Training is efficient: 10%-20% of the original size
- Similar accuracy



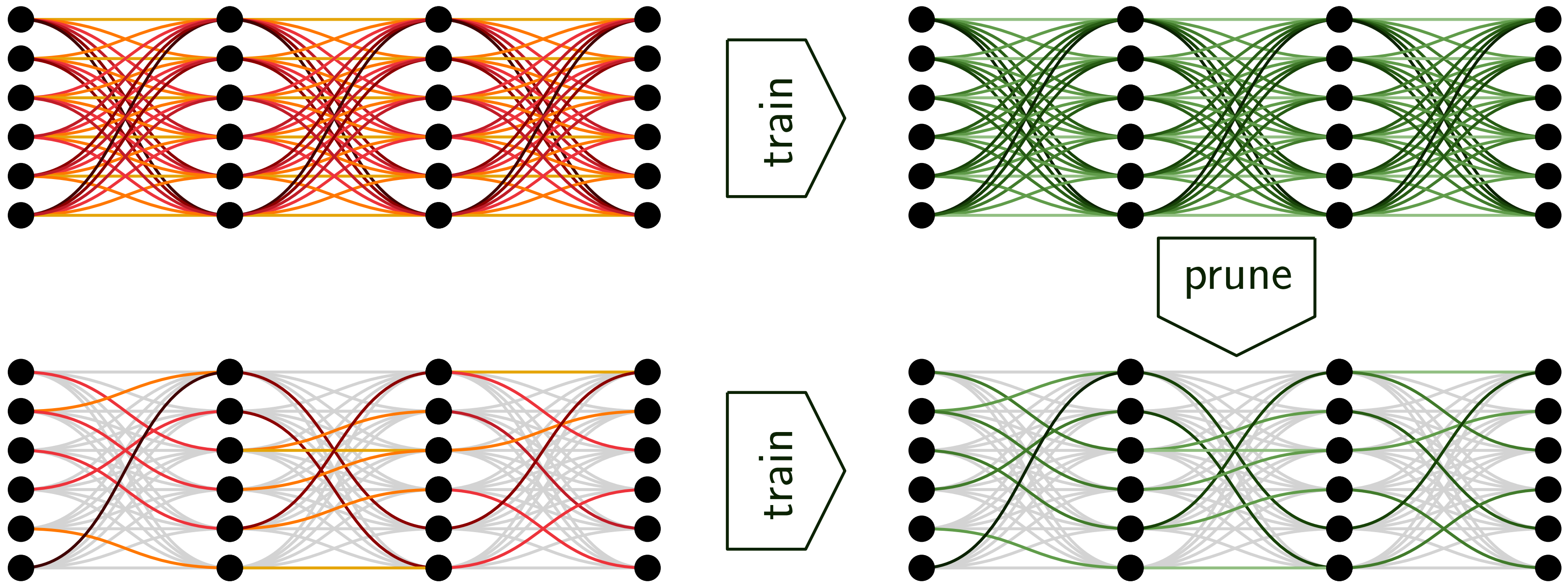
Lottery tickets

- What does it mean?



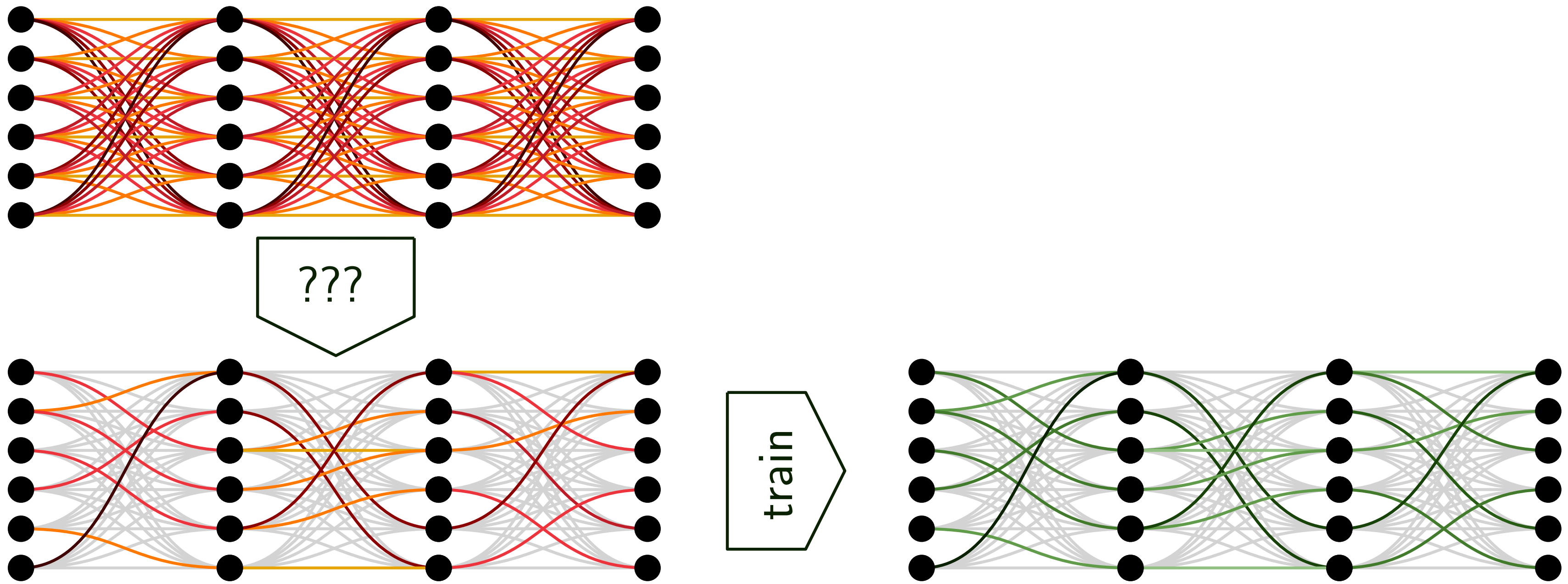
Lottery tickets

- What does it mean?
- This is not a good algorithm



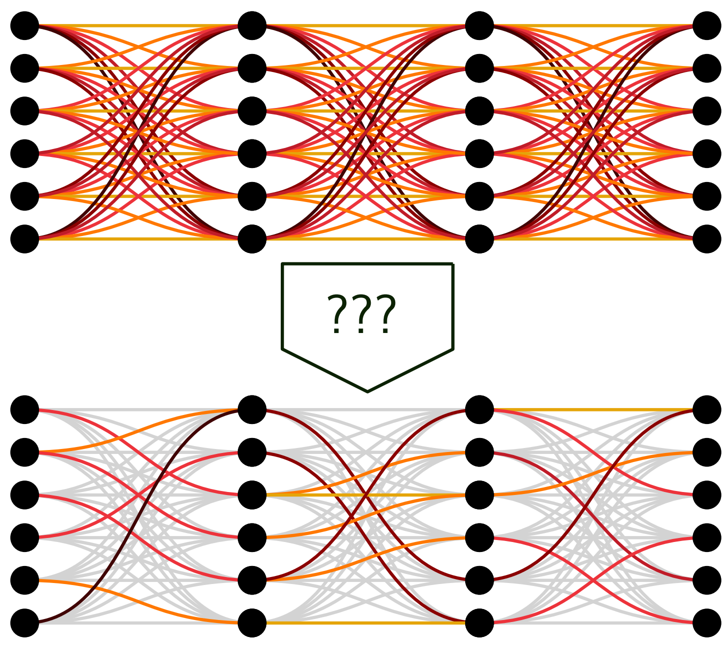
Lottery tickets

- What does it mean?
- This is not a good algorithm
- Existential result
    - Training is about topology + initialization



The Lottery Ticket Hypothesis (LTH)

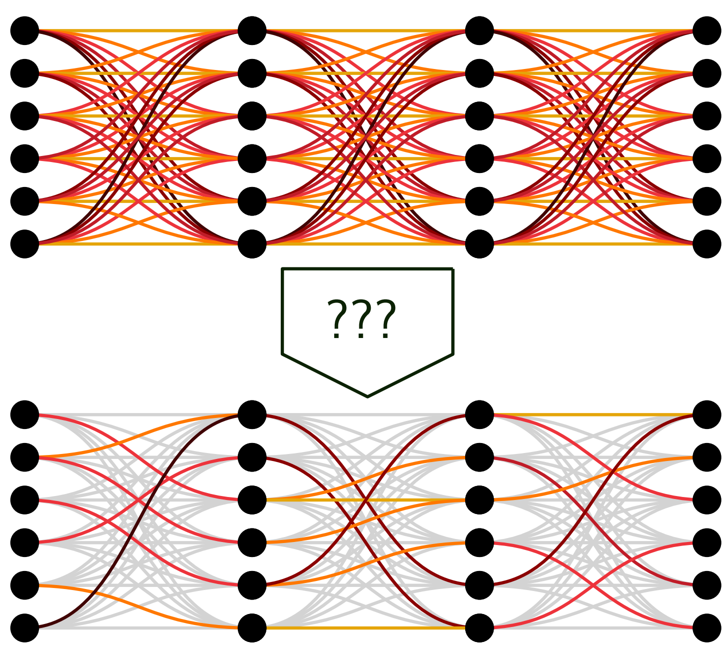
[Frankle and Carbin ICLR '19]: winning lottery tickets always exists



The Lottery Ticket Hypothesis (LTH)

[Frankle and Carbin ICLR '19]: winning lottery tickets always exists

Conjecture: *every randomly-initialized dense nework g contains a subnetwork f that matches the test accuracy of g once trained for at most the same number of iterations*

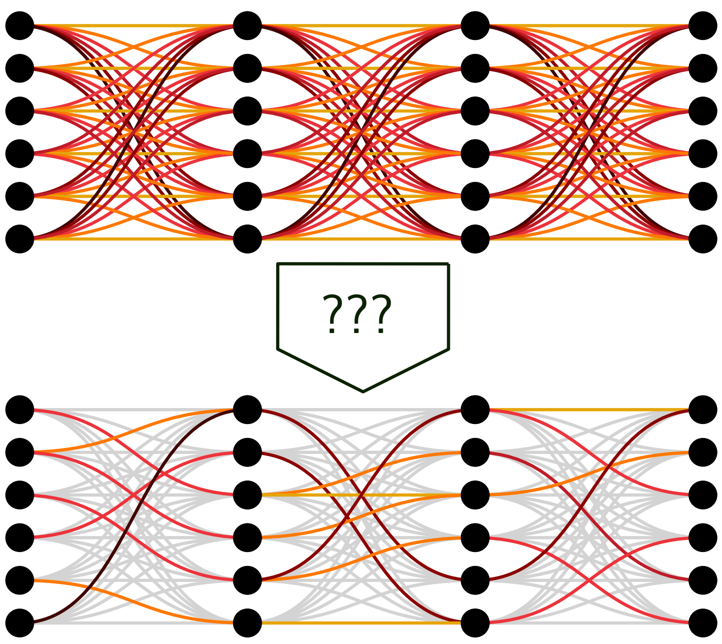


The Lottery Ticket Hypothesis (LTH)

[Frankle and Carbin ICLR '19]: winning lottery tickets always exists

Conjecture: *every randomly-initialized dense nework g contains a subnetwork f that matches the test accuracy of g once trained for at most the same number of iterations*

Lot of subsequent work . . .



The *Strong* Lottery Ticket Hypothesis (SLTH)

Intuition

- The considered networks are very large, and random

The *Strong* Lottery Ticket Hypothesis (SLTH)

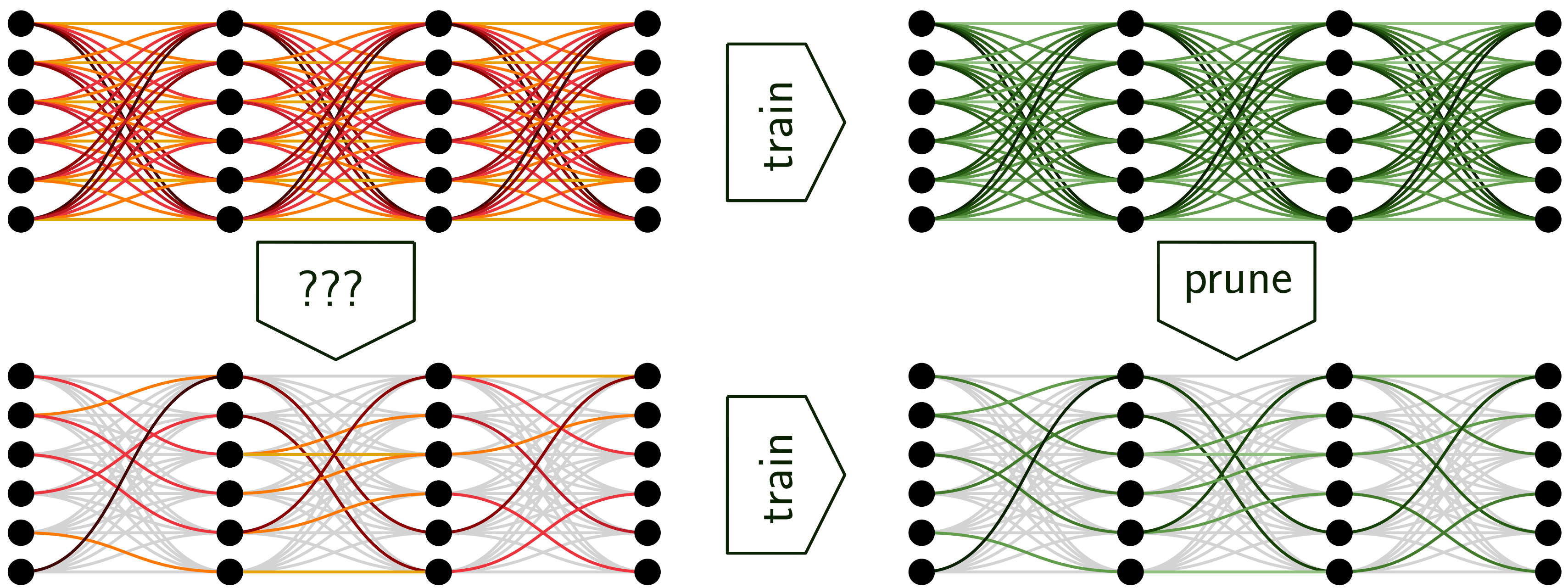
Intuition

- The considered networks are very large, and random
- They might already contain good subnetworks *from scratch*!

The *Strong* Lottery Ticket Hypothesis (SLTH)

Intuition

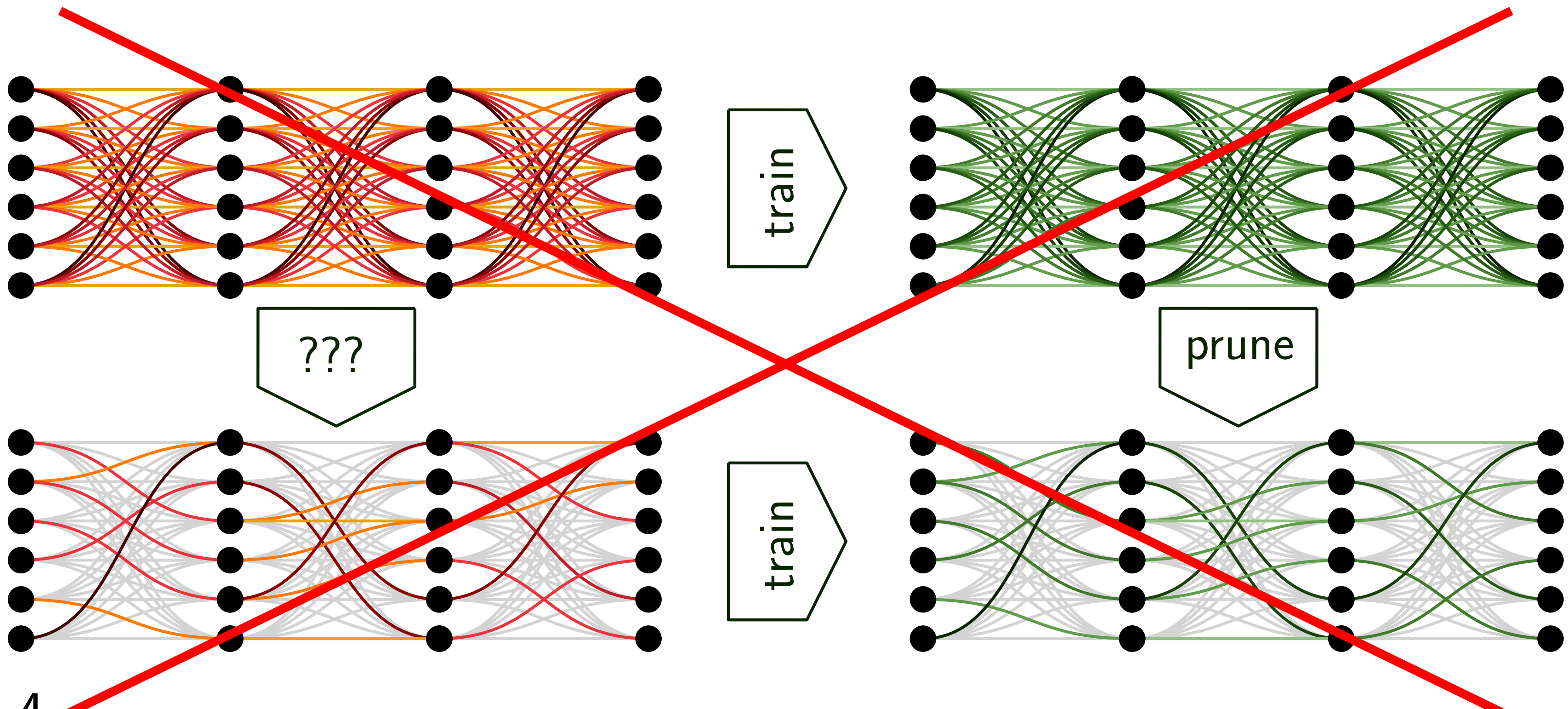
- The considered networks are very large, and random
- They might already contain good subnetworks *from scratch*!



The *Strong* Lottery Ticket Hypothesis (SLTH)

Intuition

- The considered networks are very large, and random
- They might already contain good subnetworks *from scratch*!

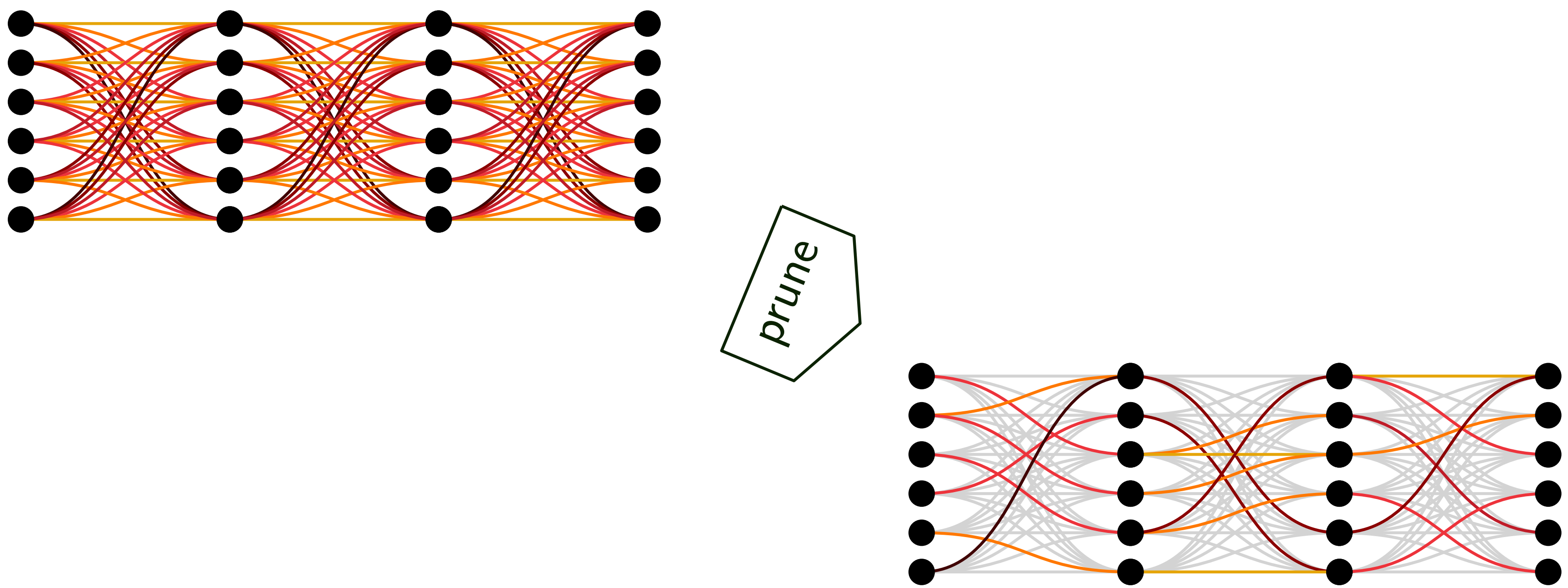


The *Strong* Lottery Ticket Hypothesis (SLTH)

Intuition

- The considered networks are very large, and random
- They might already contain good subnetworks *from scratch*!

Learn by pruning

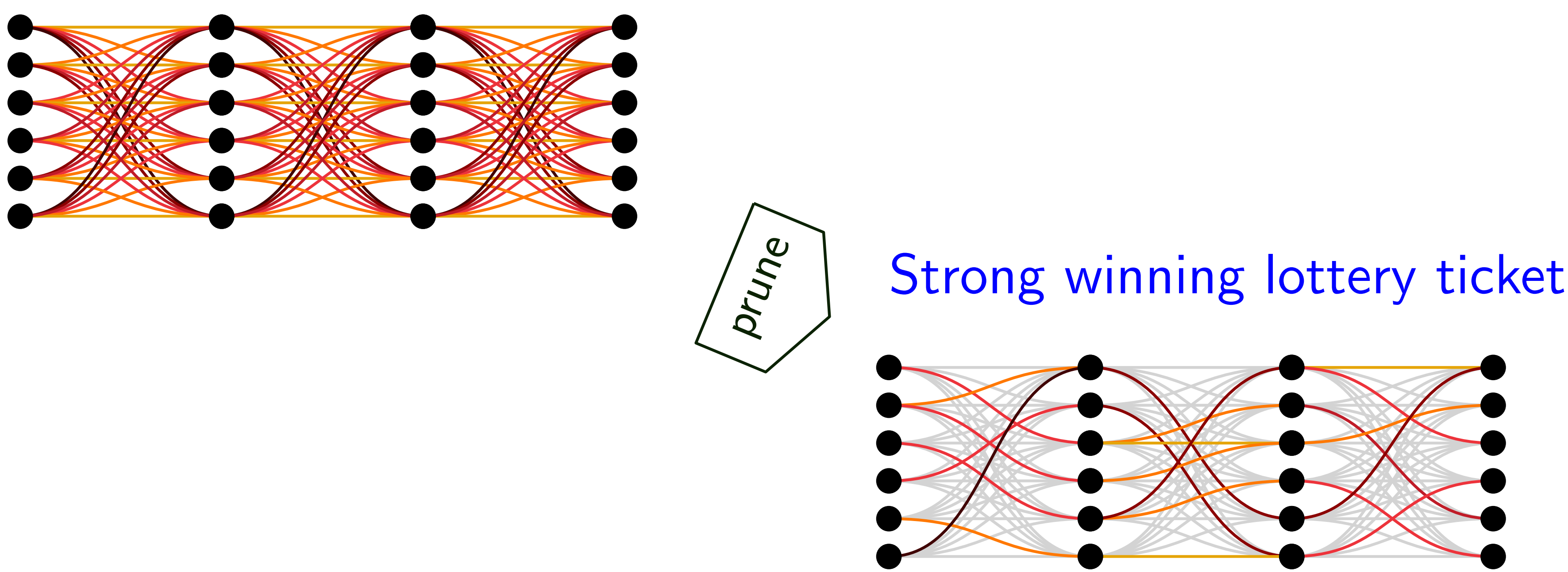


The *Strong* Lottery Ticket Hypothesis (SLTH)

Intuition

- The considered networks are very large, and random
- They might already contain good subnetworks *from scratch*!

Learn by pruning



The *Strong* Lottery Ticket Hypothesis (SLTH)

Conjecture: *every randomly-initialized and sufficiently large nework* g *contains a subnetwork* f *that matches the post-training test accuracy of* g *even without any training*

The *Strong* Lottery Ticket Hypothesis (SLTH)

Conjecture: *every randomly-initialized and sufficiently large nework* g *contains a subnetwork* f *that matches the post-training test accuracy of* g *even without any training*

[Zhou et al. NeurIPS '19] proposes a way to find f: prune weights according to some probability learned through stochastic gradient descent

The *Strong* Lottery Ticket Hypothesis (SLTH)

Conjecture: *every randomly-initialized and sufficiently large nework* g *contains a subnetwork* f *that matches the post-training test accuracy of* g *even without any training*

[Zhou et al. NeurIPS '19] 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)

Conjecture: *every randomly-initialized and sufficiently large nework g contains a subnetwork f that matches the post-training test accuracy of g even without any training*

[Zhou et al. NeurIPS '19] proposes a way to find f: prune weights according to some probability learned through stochastic gradient descent

- Decent accuracy

[Ramanujan et al. CVPR '20] improves on it: random ResNet-50 pruned to match ResNet-34 on ImageNet

The *Strong* Lottery Ticket Hypothesis (SLTH)

Conjecture: *every randomly-initialized and sufficiently large nework* g *contains a subnetwork* f *that matches the post-training test accuracy of* g *even without any training*

[Zhou et al. NeurIPS '19] proposes a way to find f: prune weights according to some probability learned through stochastic gradient descent

- Decent accuracy

[Ramanujan et al. CVPR '20] improves on it: random ResNet-50 pruned to match ResNet-34 on ImageNet

[Diffenderfer and Kailkhura ICLR '21]: works even with binary weights!

Do we have a theorem?

Target result: *Given a network g with random weights, with high probability, it is possible to prune g to approximate any sufficiently smaller network f*

Do we have a theorem?

Target result: *Given a network g with random weights, with high probability, it is possible to prune g to approximate any sufficiently smaller network f*

Target result (equivalent): *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: *Given a network g with random weights, with high probability, it is possible to prune g to approximate any sufficiently smaller network f*

Target result (equivalent): *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: *Given a network g with random weights, with high probability, it is possible to prune g to approximate any sufficiently smaller network f*

Target result (equivalent): *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: *Given a network g with random weights, with high probability, it is possible to prune g to approximate any sufficiently smaller network f*

Target result (equivalent): *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. ICML '20]: polynomially overparameterized dense networks with ReLU activation functions

Overview of the (theoretical) results

SLTH holds for:

- [Malach et al. ICML '20]: polynomially overparameterized dense networks with ReLU activation functions
- [Pensia et al. NeurIPS '20]: logarithmically overparameterized dense networks with ReLU activation functions

Overview of the (theoretical) results

SLTH holds for:

- [Malach et al. ICML '20]: polynomially overparameterized dense networks with ReLU activation functions
- [Pensia et al. NeurIPS '20]: logarithmically overparameterized dense networks with ReLU activation functions
- [Diffenderfer and Kailkhura ICLR '21]: polynomially overparameterized binary dense networks

Overview of the (theoretical) results

SLTH holds for:

- [Malach et al. ICML '20]: polynomially overparameterized dense networks with ReLU activation functions
- [Pensia et al. NeurIPS '20]: logarithmically overparameterized dense networks with ReLU activation functions
- [Diffenderfer and Kailkhura ICLR '21]: polynomially overparameterized binary dense networks
- [Sreenivasan et al. AIStat '22]: polylogarithmically overparameterized binary dense networks

Overview of the (theoretical) results

SLTH holds for:

- [Malach et al. ICML '20]: polynomially overparameterized dense networks with ReLU activation functions
- [Pensia et al. NeurIPS '20]: logarithmically overparameterized dense networks with ReLU activation functions
- [Diffenderfer and Kailkhura ICLR '21]: polynomially overparameterized binary dense networks
- [Sreenivasan et al. AIStat '22]: polylogarithmically overparameterized binary dense networks
- [da Cunha et al. ICLR '22]: 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. ICML '20]: polynomially overparameterized dense networks with ReLU activation functions
- [Pensia et al. NeurIPS '20]: logarithmically overparameterized dense networks with ReLU activation functions
- [Diffenderfer and Kailkhura ICLR '21]: polynomially overparameterized binary dense networks
- [Sreenivasan et al. AIStat '22]: polylogarithmically overparameterized binary dense networks
- [da Cunha et al. ICLR '22]: logarithmically overparameterized convolutional neural networks (CNNs) with ReLU activation functions and non-negative inputs
- [Burkholz NeurIPS, ICML '22]: 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. ICML '20]: polynomially overparameterized dense networks with ReLU activation functions
- [Pensia et al. NeurIPS '20]: logarithmically overparameterized dense networks with ReLU activation functions
- [Diffenderfer and Kailkhura ICLR '21]: polynomially overparameterized binary dense networks
- [Sreenivasan et al. AIStat '22]: polylogarithmically overparameterized binary dense networks
- [da Cunha et al. ICLR '22]: logarithmically overparameterized convolutional neural networks (CNNs) with ReLU activation functions and non-negative inputs
- [Burkholz NeurIPS, ICML '22]: logarithmically overparameterized dense networks, CNNs, and residual architectures with a wider class of activation functions and less depth overhead
- [Ferbach et al. ICLR '22]: logarithmically overparameterized equivariant networks with ReLU activation functions

Review: SLTH in dense networks

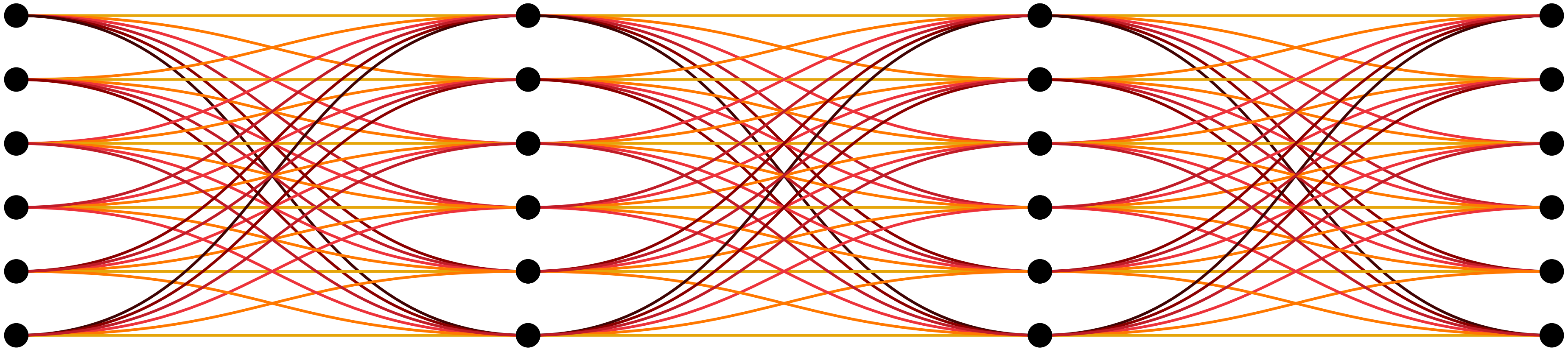
Dense network: $f(\mathbf{x}) = \mathbf{W}_\ell \sigma(\mathbf{W}_{\ell-1} \ldots \sigma(\mathbf{W}_1 \mathbf{x}))$

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

Review: SLTH in dense networks

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

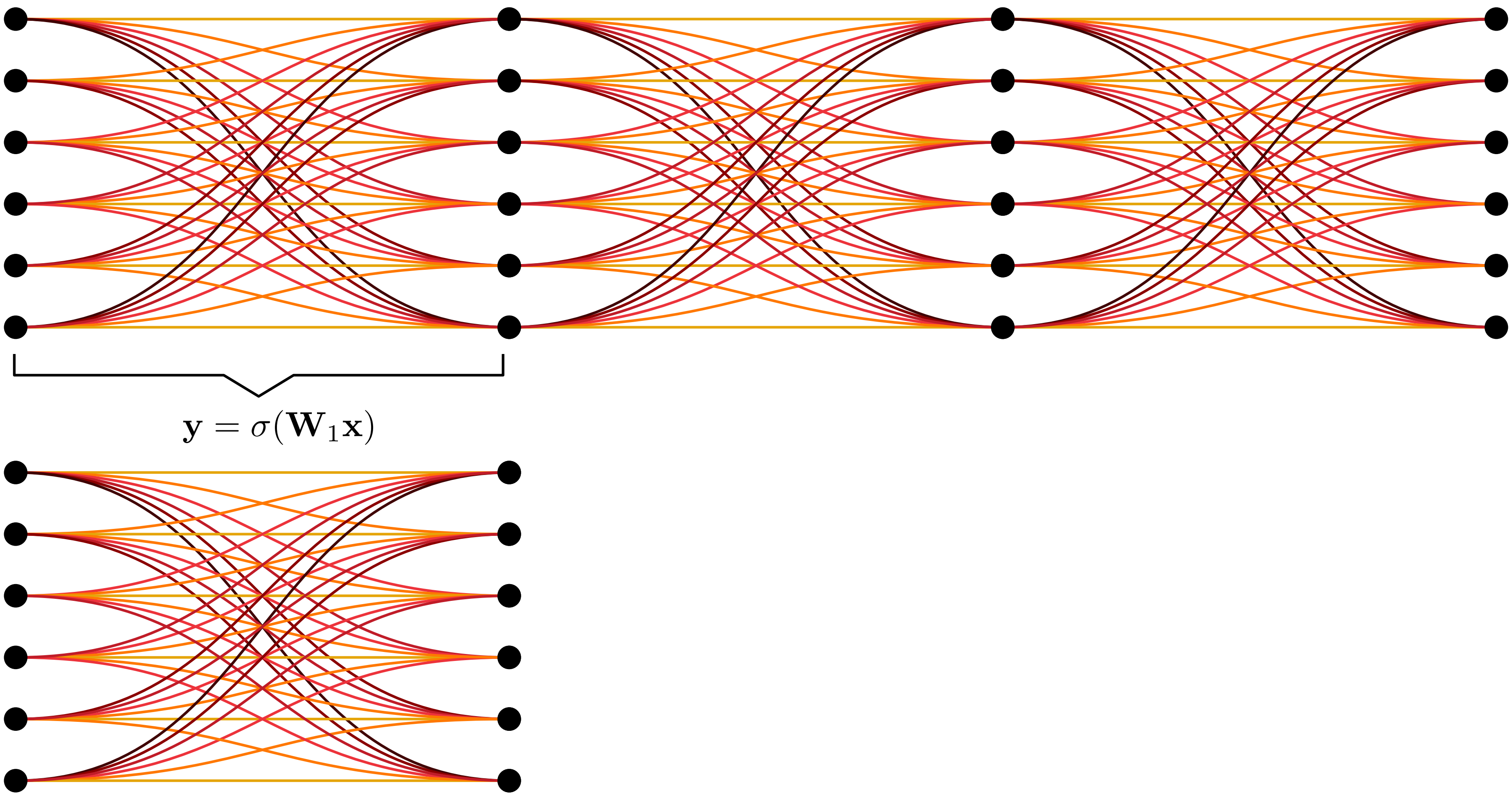
- $\mathbf{x} \in \mathbb{R}^{d_0}$, $\mathbf{W}_i \in \mathbb{R}^{d_{i-1} \times d_i}$
- $\sigma(x) = \max(0, x)$ (ReLU)



Review: SLTH in dense networks

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

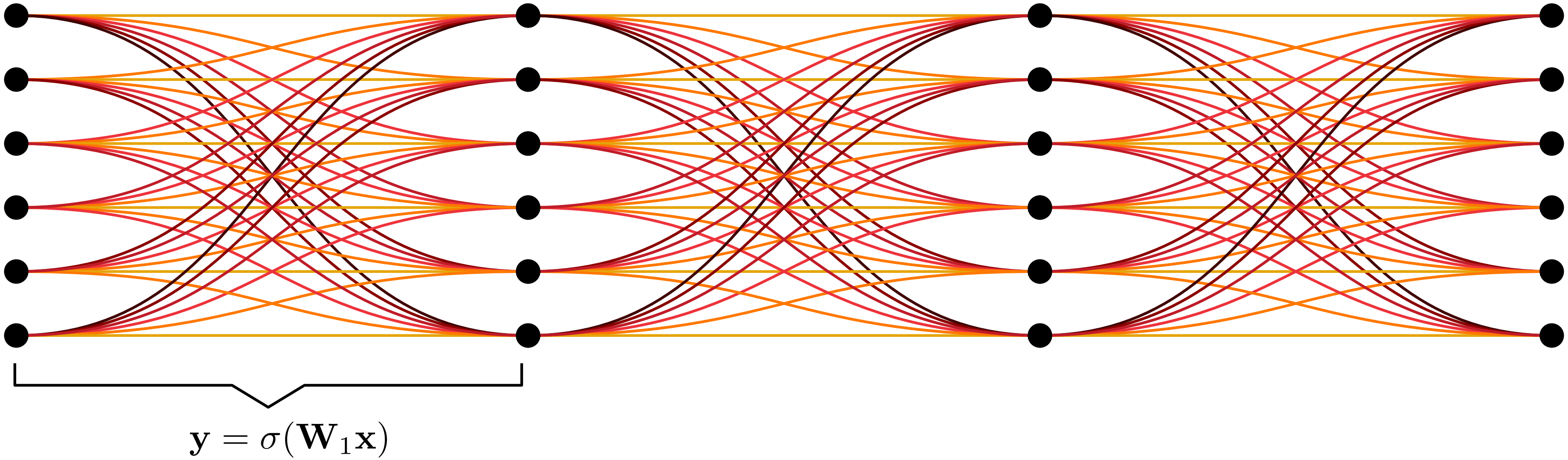
- $\mathbf{x} \in \mathbb{R}^{d_0}$, $\mathbf{W}_i \in \mathbb{R}^{d_{i-1} \times d_i}$
- $\sigma(x) = \max(0, x)$ (ReLU)

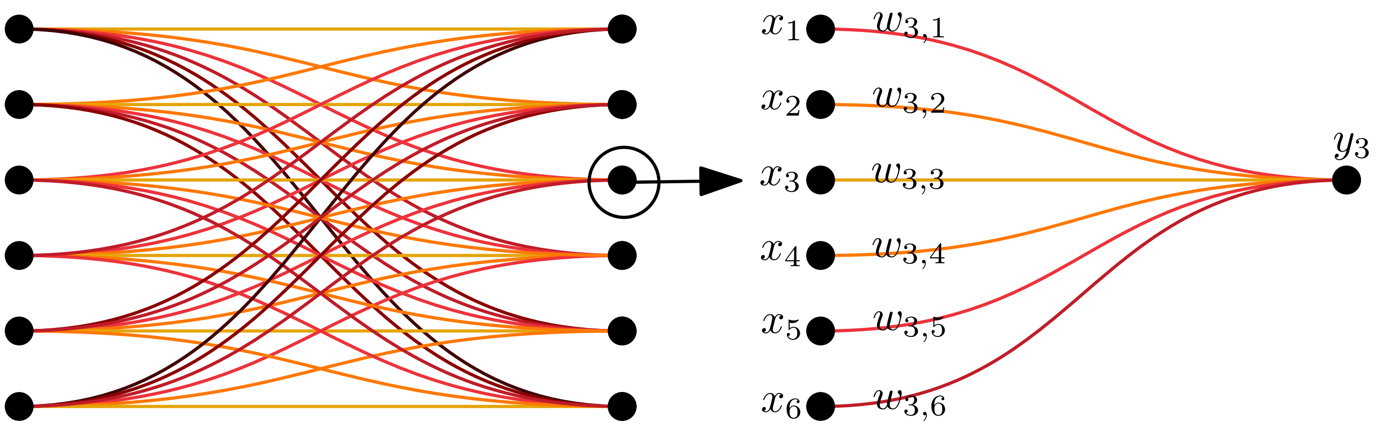


Review: SLTH in dense networks

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)

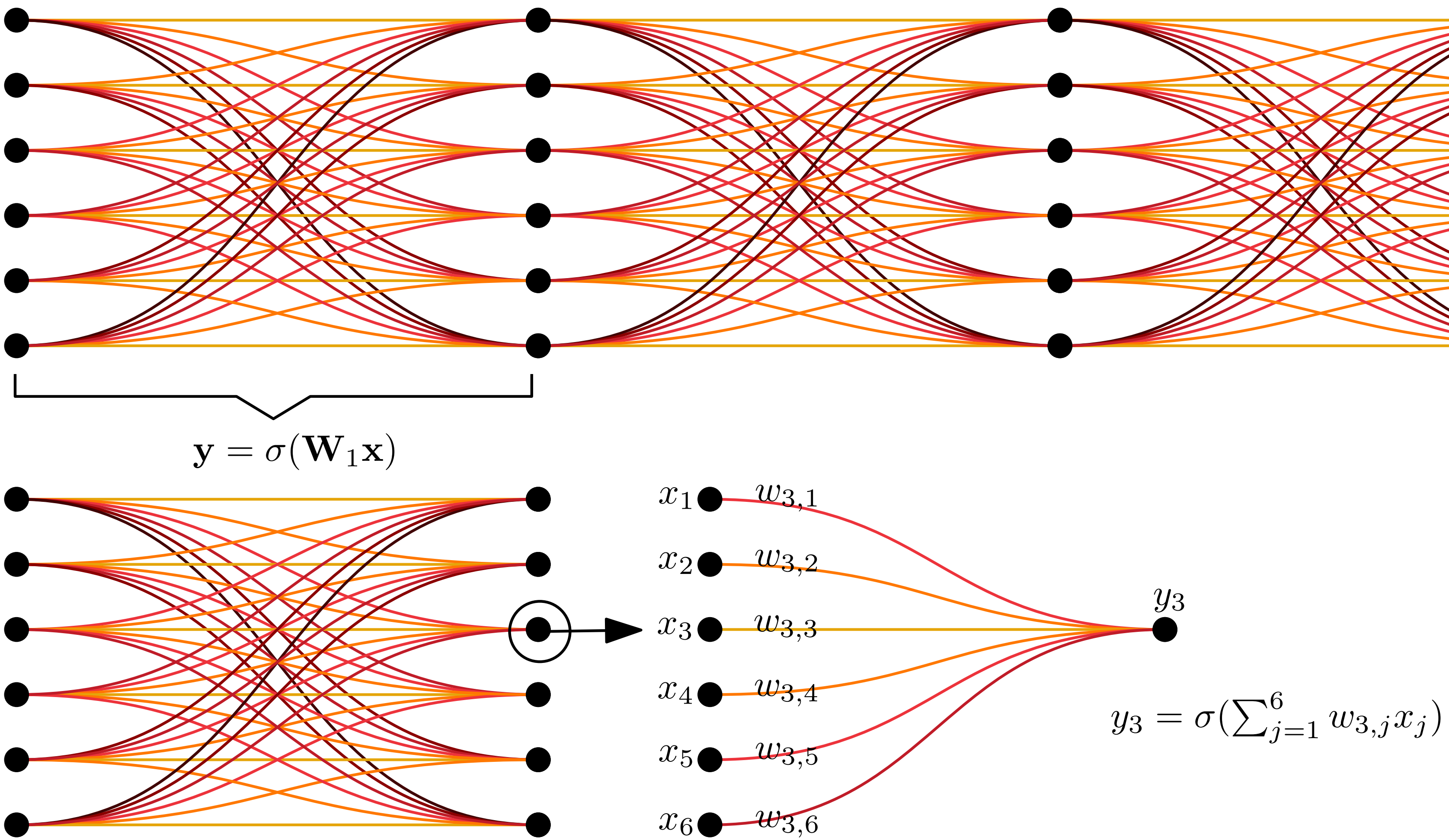




Review: SLTH in dense networks

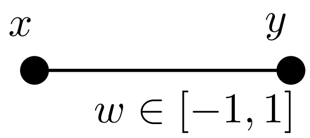
Dense network: $f(\mathbf{x}) = \mathbf{W}_\ell \sigma(\mathbf{W}_{\ell-1} \ldots \sigma(\mathbf{W}_1 \mathbf{x}))$

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



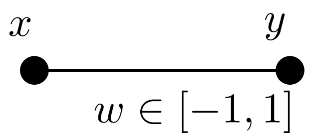
Review: SLTH in dense networks

- First target: approx $y = wx$ within error ε (no ReLU, one edge only)



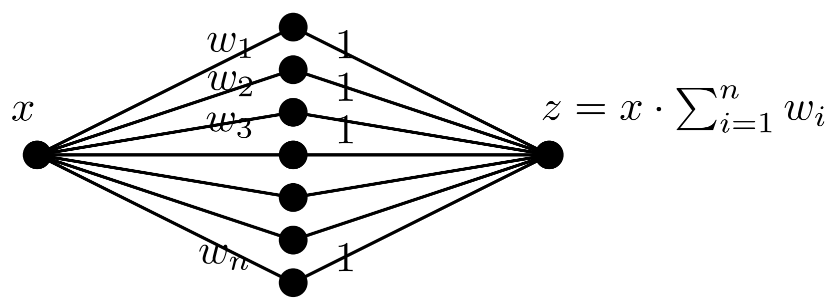
Review: SLTH in dense networks

- First target: approx $y = wx$ within error ε (no ReLU, one edge only)



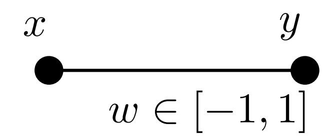
  - Original approach

    add intermediate layer, sample
    $w_i \sim$ Unif$[-1, 1]$ until getting $w \pm \varepsilon$



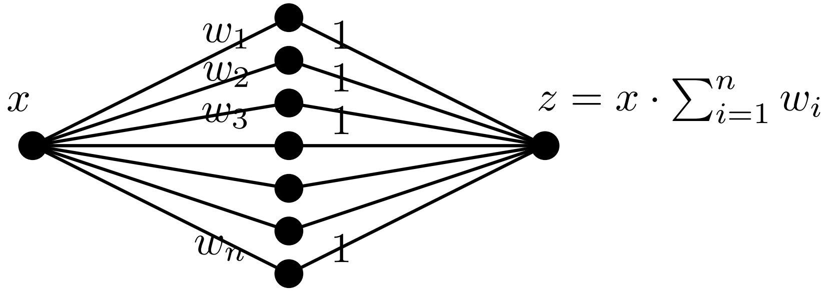
Review: SLTH in dense networks

- First target: approx $y = wx$ within error ε (no ReLU, one edge only)



  - Original approach

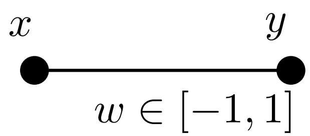
    add intermediate layer, sample
    $w_i \sim$ Unif$[-1, 1]$ until getting $w \pm \varepsilon$



    roughly $1/\varepsilon$ samples

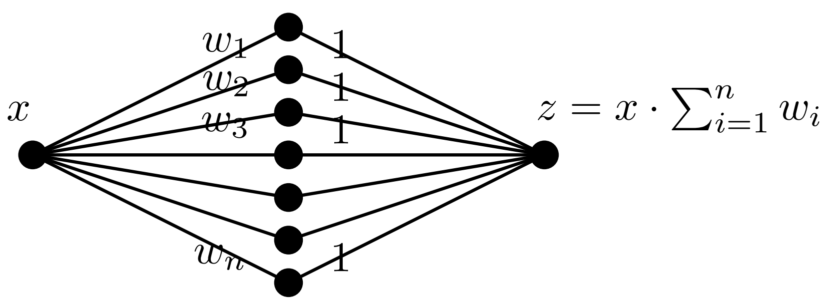
Review: SLTH in dense networks

- First target: approx $y = wx$ within error ε (no ReLU, one edge only)



- Original approach

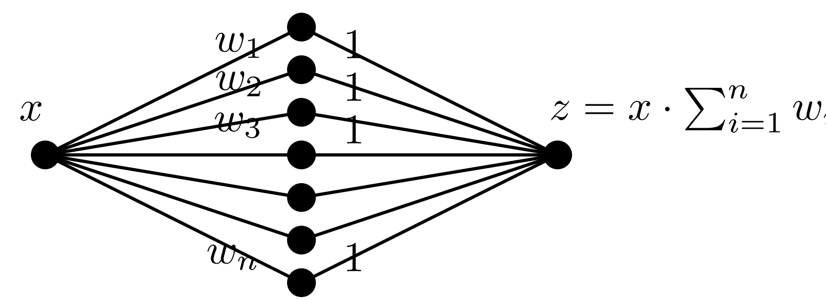
  add intermediate layer, sample
  $w_i \sim$ Unif$[-1, 1]$ until getting $w \pm \varepsilon$



  roughly $1/\varepsilon$ samples

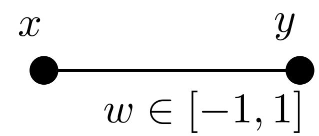
- Random subset sum (RSS) approach:

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



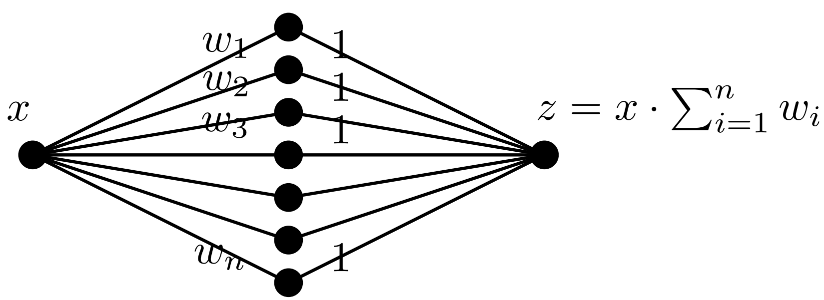
Review: SLTH in dense networks

- First target: approx $y = wx$ within error ε (no ReLU, one edge only)



- Original approach

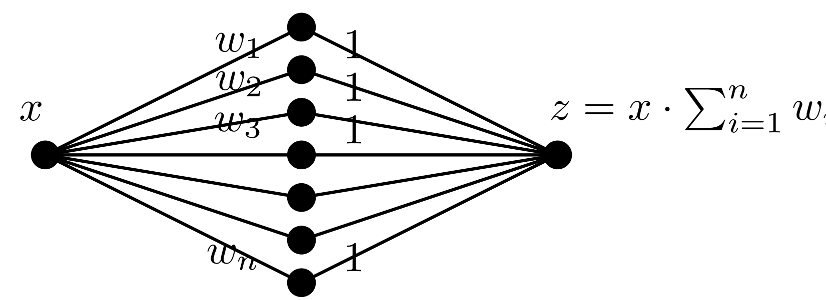
  add intermediate layer, sample
  $w_i \sim$ Unif$[-1, 1]$ until getting $w \pm \varepsilon$



  roughly $1/\varepsilon$ samples

- Random subset sum (RSS) approach:

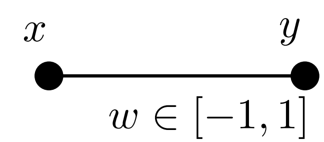
  add intermediate layer, sample
  $w_i \sim$ Unif$[-1, 1]$ and find a good subset



  How many?

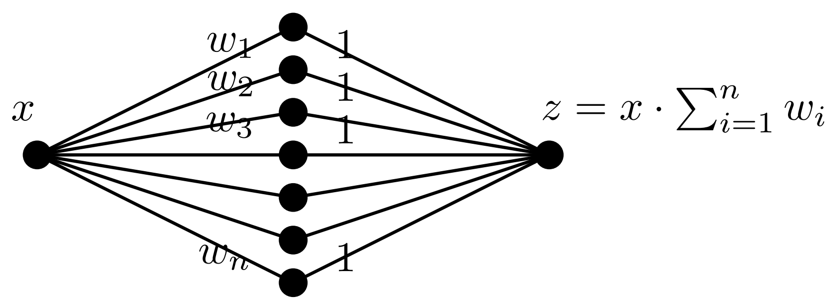
Review: SLTH in dense networks

- First target: approx $y = wx$ within error ε (no ReLU, one edge only)



- Original approach

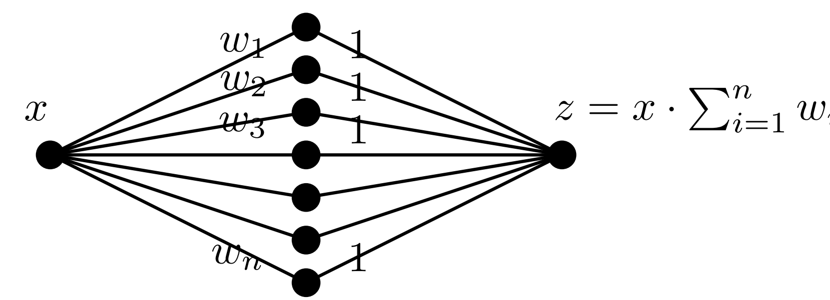
  add intermediate layer, sample $w_i \sim$ Unif$[-1, 1]$ until getting $w \pm \varepsilon$



  roughly $1/\varepsilon$ samples

- Random subset sum (RSS) approach:

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

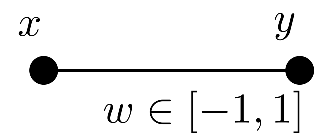


  How many?

Theorem [Lueker 1998; da Cunha et al. ESA '23]: *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 $\left|z - \sum_{i \in S} x_i\right| < 2\varepsilon$*

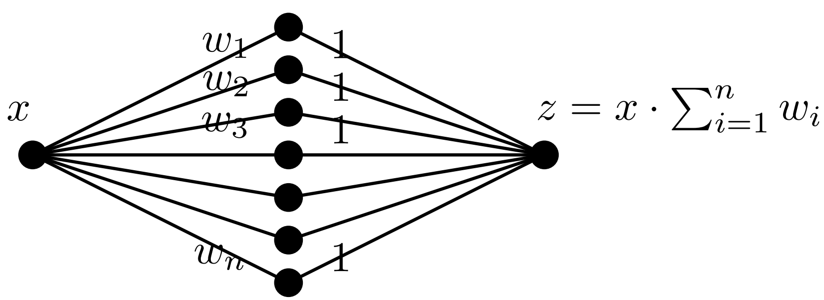
Review: SLTH in dense networks

- First target: approx $y = wx$ within error ε (no ReLU, one edge only)



- Original approach

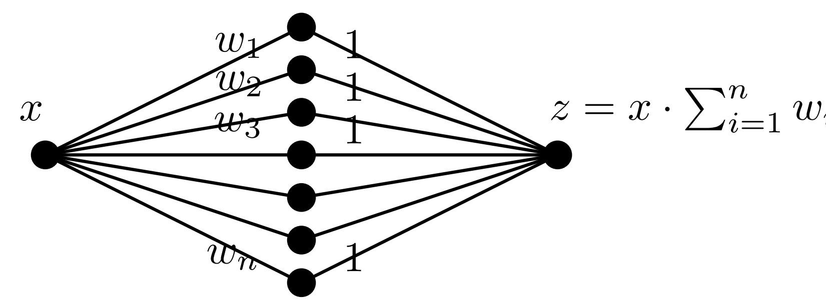
  add intermediate layer, sample
  $w_i \sim$ Unif$[-1, 1]$ until getting $w \pm \varepsilon$



  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?

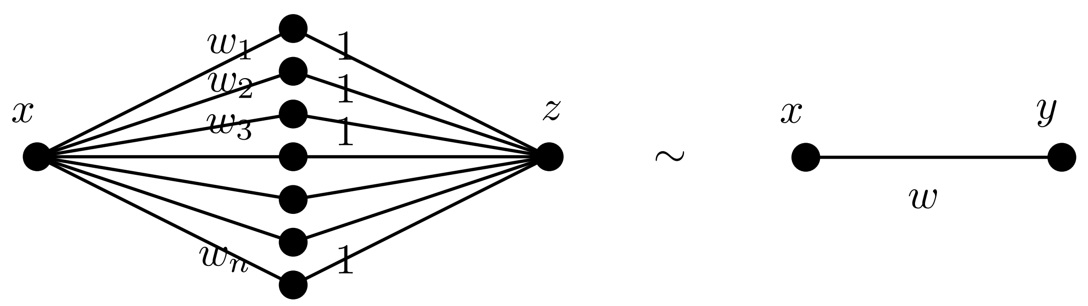
Theorem [Lueker 1998; da Cunha et al. ESA '23]: *Let* $x_1, \ldots, x_n \in [-1, 1]$ *be i.i.d. uniform random variables. Given any error parameter* $\varepsilon > 0$, *there exists a constant* $C > 0$ *such that if* $n \geq C \log 1/\varepsilon$ *then, with probability* $1 - \exp\left[(n - C \log 1/\varepsilon)^2 / 4n\right]$, *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"

RSS approach

- Random subset sum (RSS) approach:

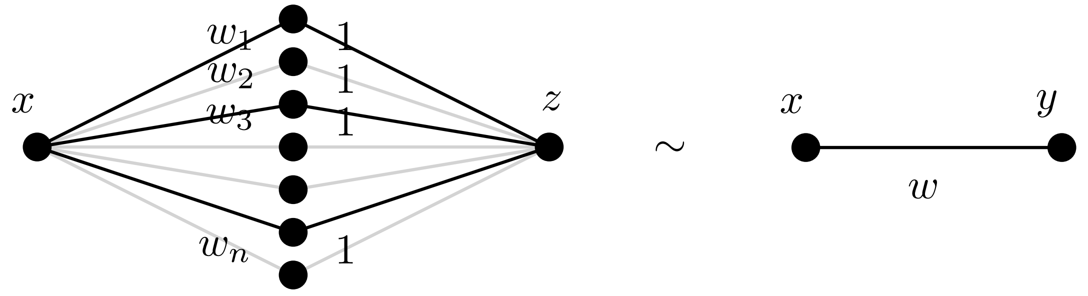
  add intermediate layer, sample $w_i \sim \text{Unif}[-1, 1]$ and find a good subset



RSS approach

- Random subset sum (RSS) approach:

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

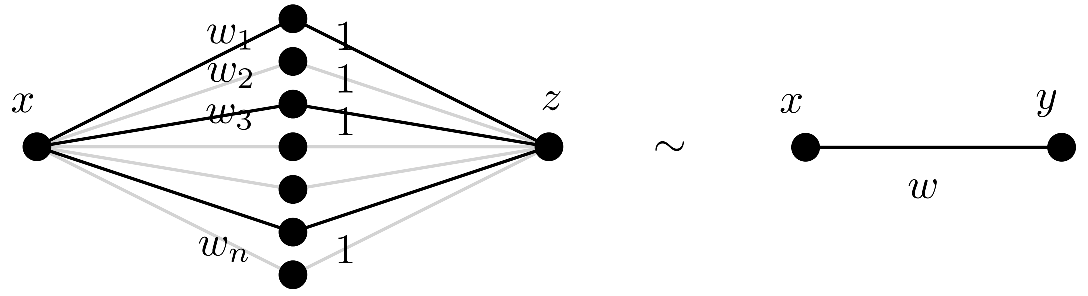


$n \geq C \log 1/\varepsilon \implies \exists S \subseteq [n] : \left| w - \sum_{i \in S} w_i \right| < 2\varepsilon$

RSS approach

- Random subset sum (RSS) approach:

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



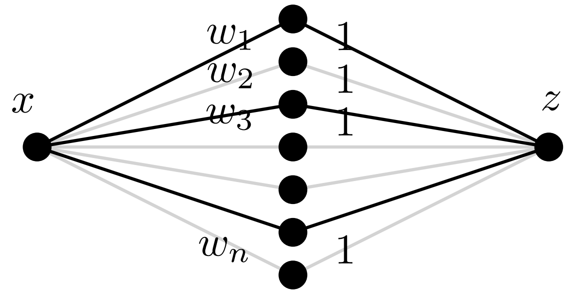
$n \geq C \log 1/\varepsilon \implies \exists S \subseteq [n] : \left| w - \sum_{i \in S} w_i \right| < 2\varepsilon$

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

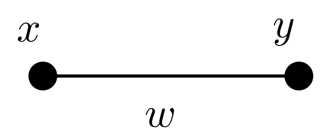
RSS approach

- Random subset sum (RSS) approach:

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



$\sim$

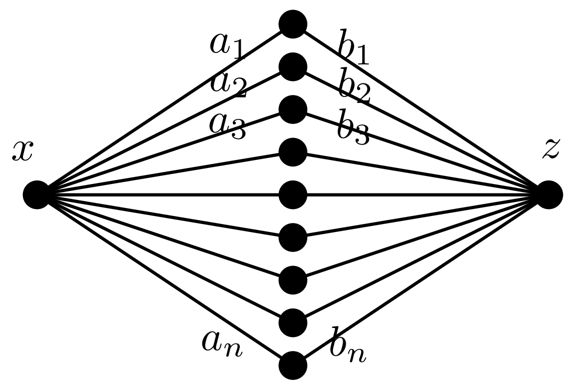


$$n \geq C \log 1/\varepsilon \implies \exists S \subseteq [n] : \left|w - \sum_{i \in S} w_i\right| < 2\varepsilon$$

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

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

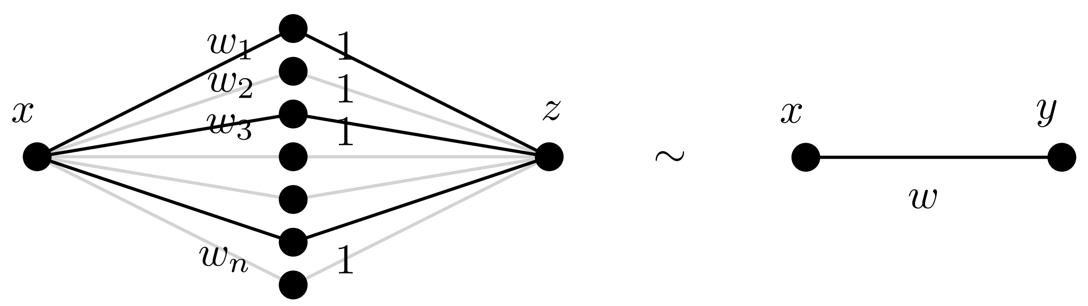
ReLU:
$\sigma(x) = \max(0, x)$



RSS approach

- Random subset sum (RSS) approach:

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

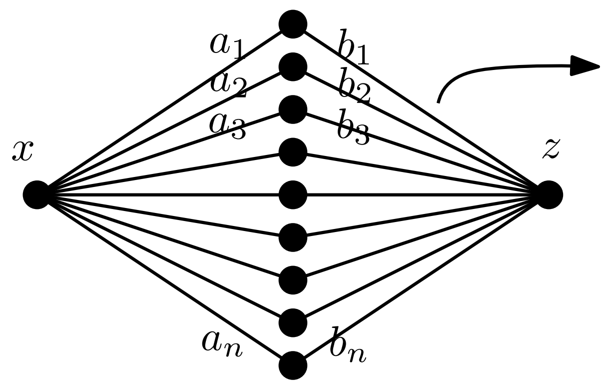


$n \geq C \log 1/\varepsilon \implies \exists S \subseteq [n] : \left|w - \sum_{i \in S} w_i\right| < 2\varepsilon$

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

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

  how to deal with non-linearity?



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

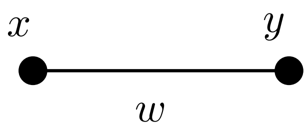
ReLU:
$\sigma(x) = \max(0, x)$

Exploiting properties of the ReLU

- Completely random initialization + ReLU (non-linearity)

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

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

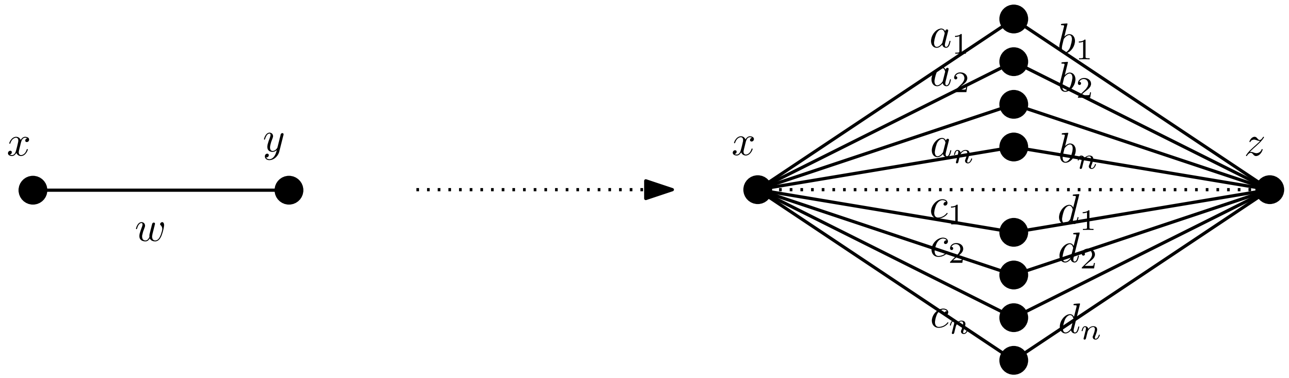


Exploiting properties of the ReLU

- Completely random initialization + ReLU (non-linearity)

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

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

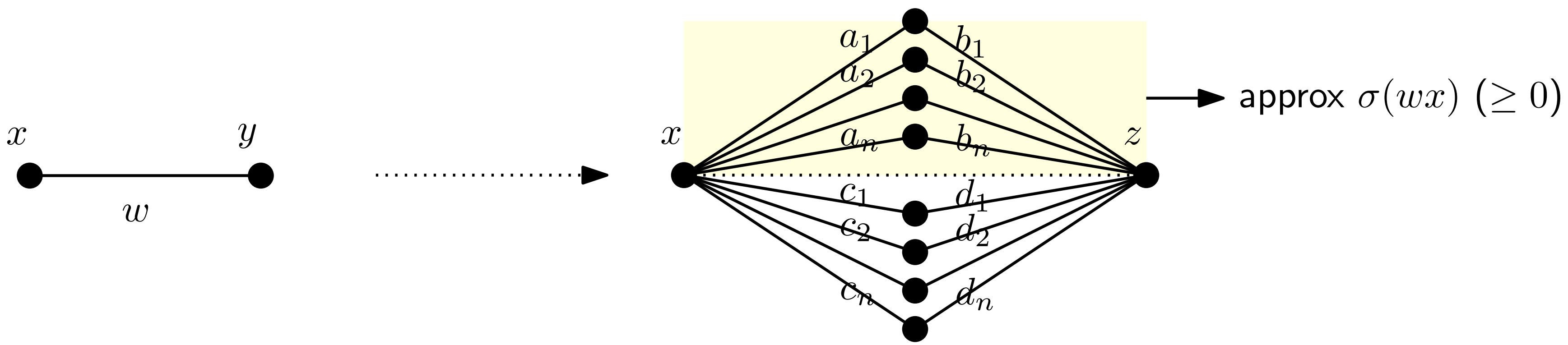


Exploiting properties of the ReLU

- Completely random initialization + ReLU (non-linearity)

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

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

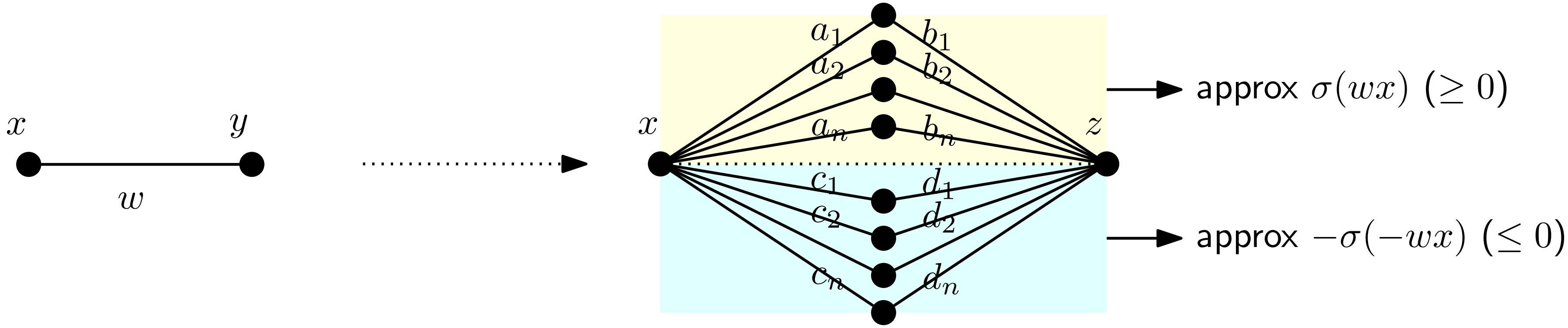


Exploiting properties of the ReLU

- Completely random initialization + ReLU (non-linearity)

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

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

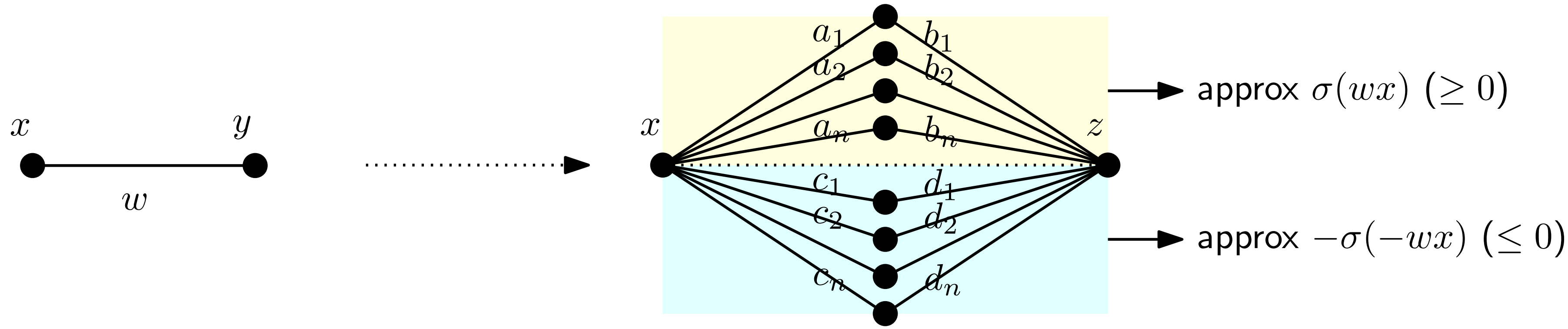


Exploiting properties of the ReLU

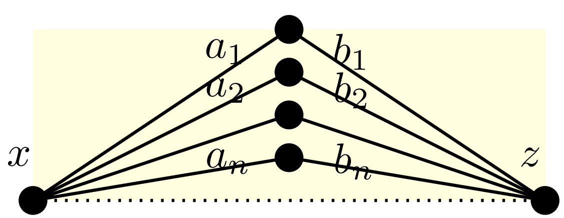
- Completely random initialization + ReLU (non-linearity)

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

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



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

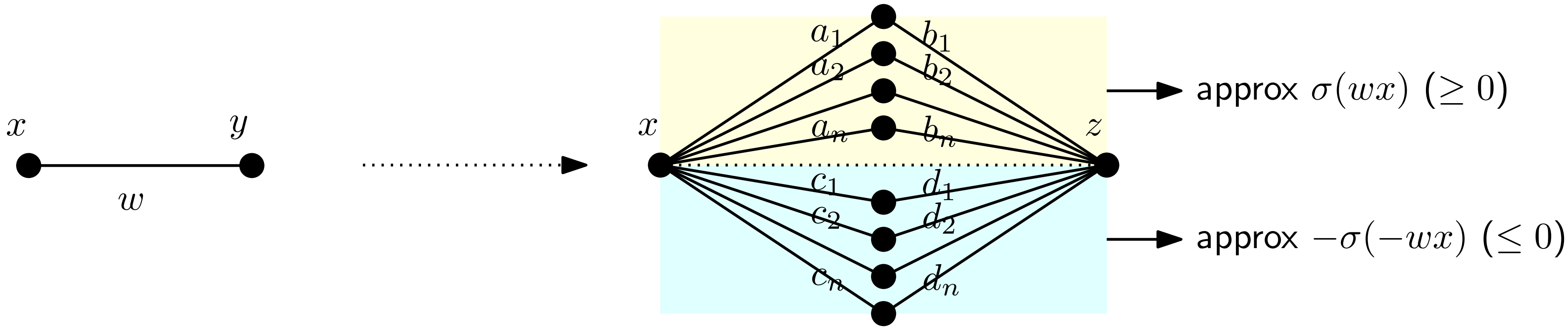


Exploiting properties of the ReLU

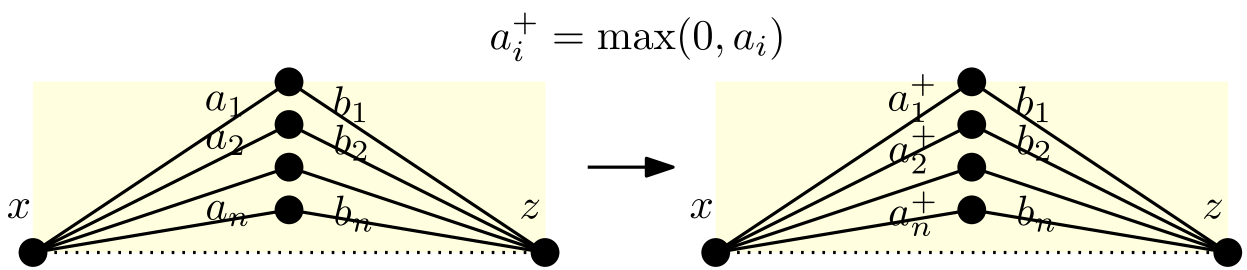
- Completely random initialization + ReLU (non-linearity)

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

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



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

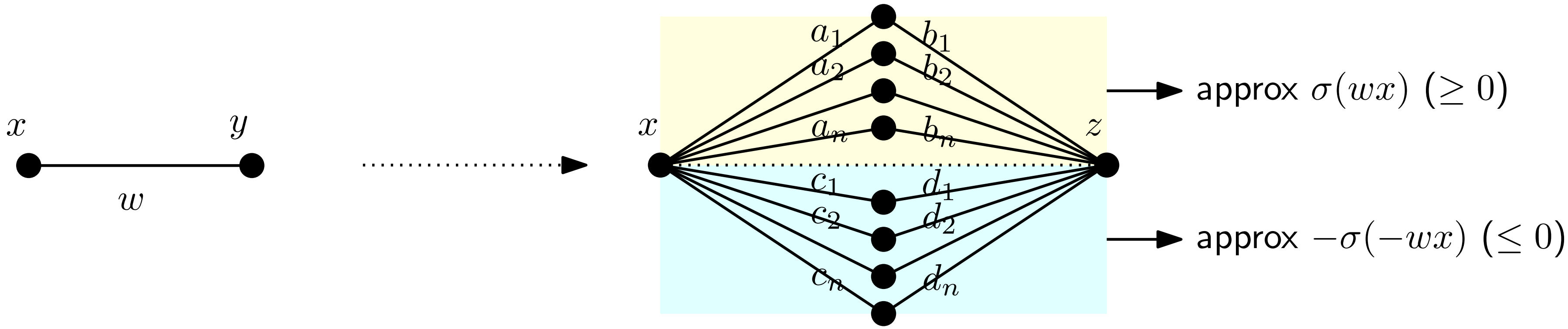


Exploiting properties of the ReLU

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

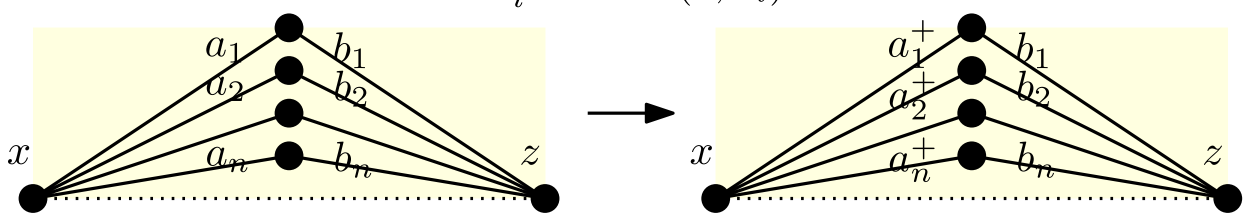
- Completely random initialization + ReLU (non-linearity)

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



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

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



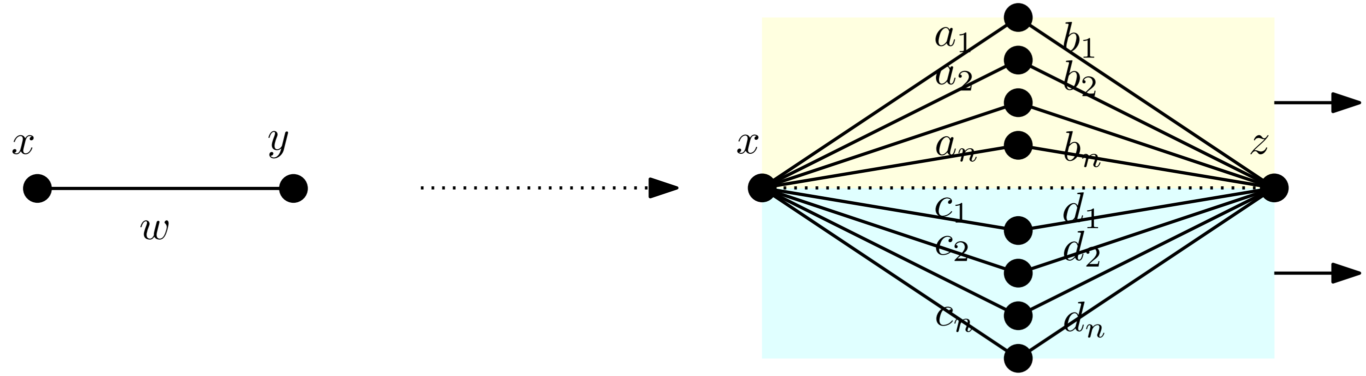
$$\left|\sigma(wx) - \textstyle\sum_{i\in[n]} b_i \sigma(a_i x)\right| = \left|\sigma(wx) - \textstyle\sum_{i\in[n]} b_i a_i^+ \sigma(x)\right|$$

Exploiting properties of the ReLU

- Completely random initialization + ReLU (non-linearity)

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

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

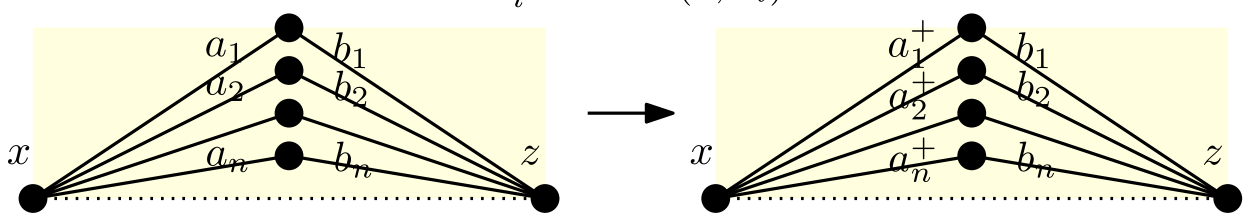


approx $\sigma(wx)$ (≥ 0)

approx $-\sigma(-wx)$ (≤ 0)

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

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



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

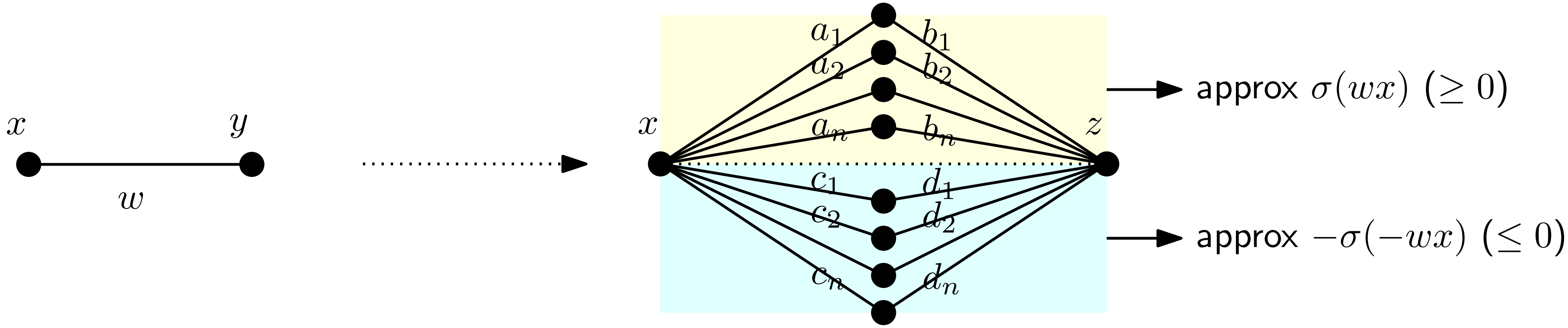
if $x \leq 0$, easy

Exploiting properties of the ReLU

- Completely random initialization + ReLU (non-linearity)

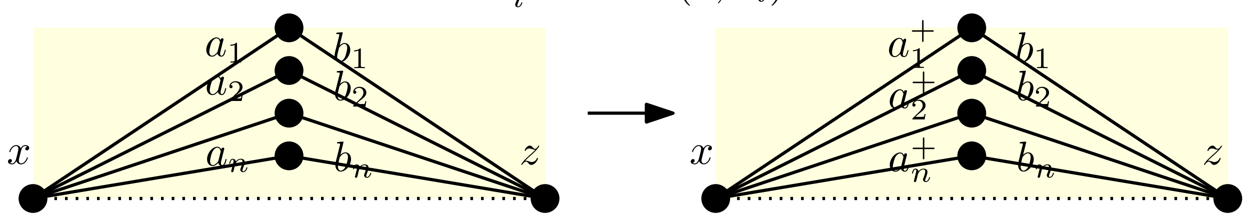
  Property of ReLU: $wx = \sigma(wx) - \sigma(-wx)$

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



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

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



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

if $x \leq 0$, easy

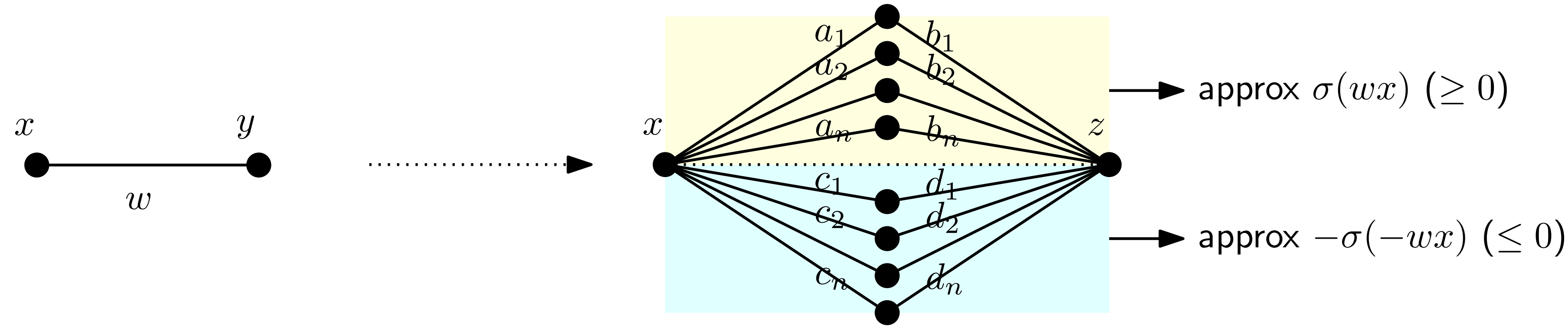
if $x > 0$, then RSS holds!

Exploiting properties of the ReLU

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

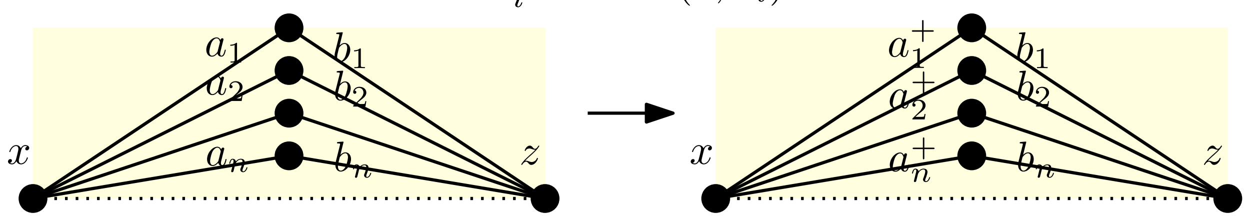
- Completely random initialization + ReLU (non-linearity)

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



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

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



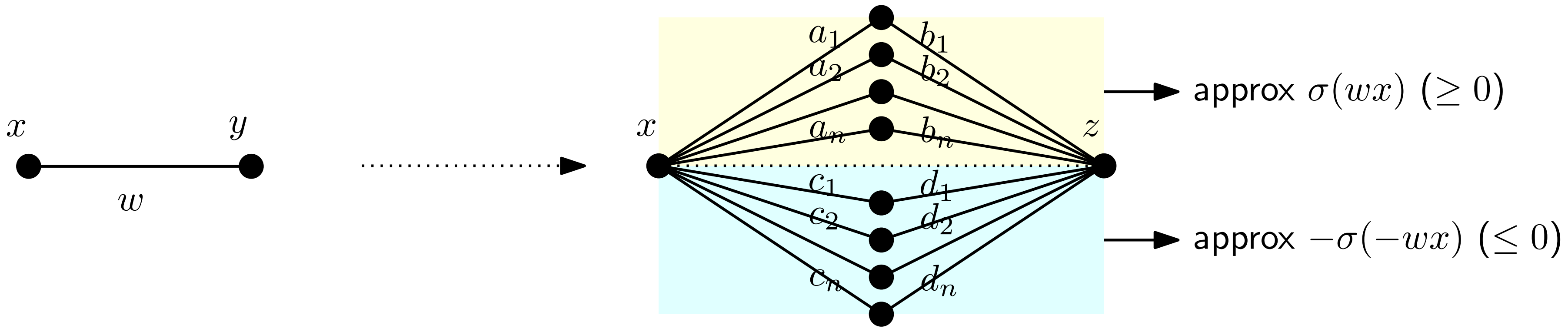
$$\left|\sigma(wx) - \sum_{i\in[n]} b_i\sigma(a_i x)\right| = \left|\sigma(wx) - \sum_{i\in[n]} b_i a_i^+ \sigma(x)\right| = x\left|w - \sum_{i\in[n]} b_i a_i^+\right|$$

if $x \leq 0$, easy

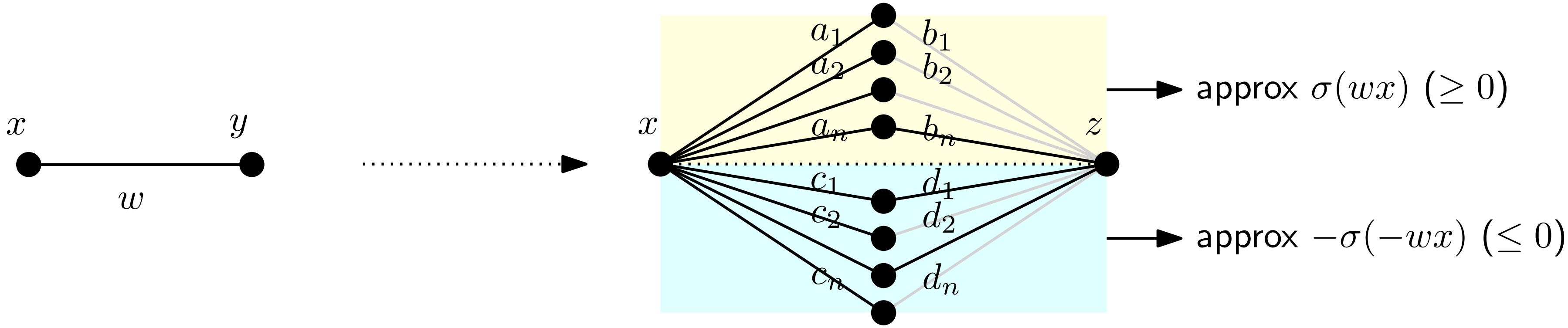
if $x > 0$, then RSS holds!

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

Putting everything together

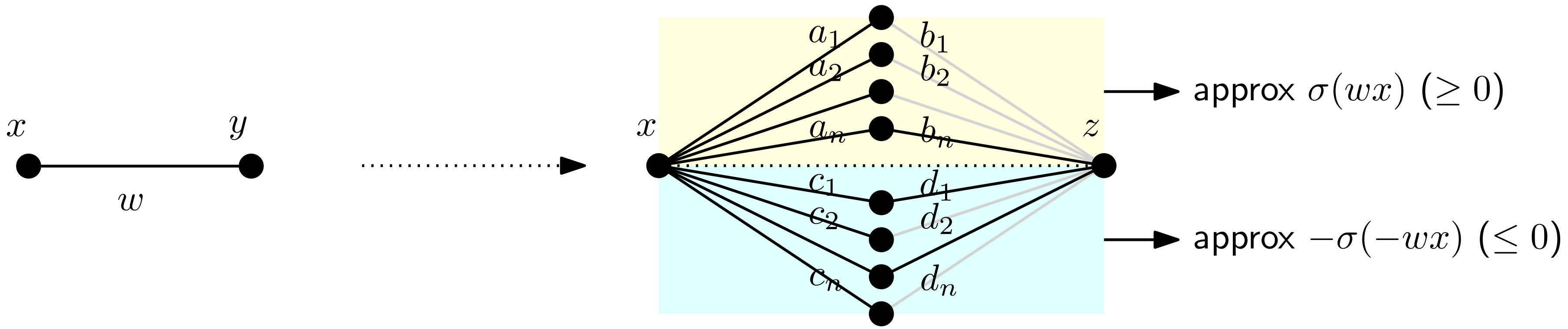


Putting everything together

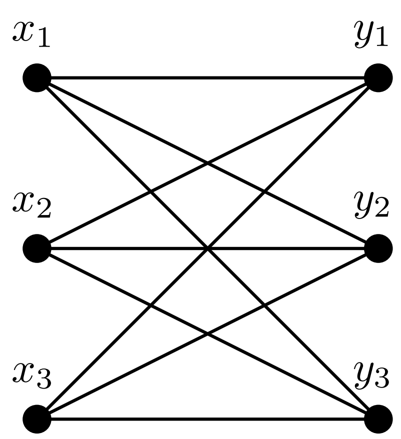


prune only the right layer: reuse the left layer

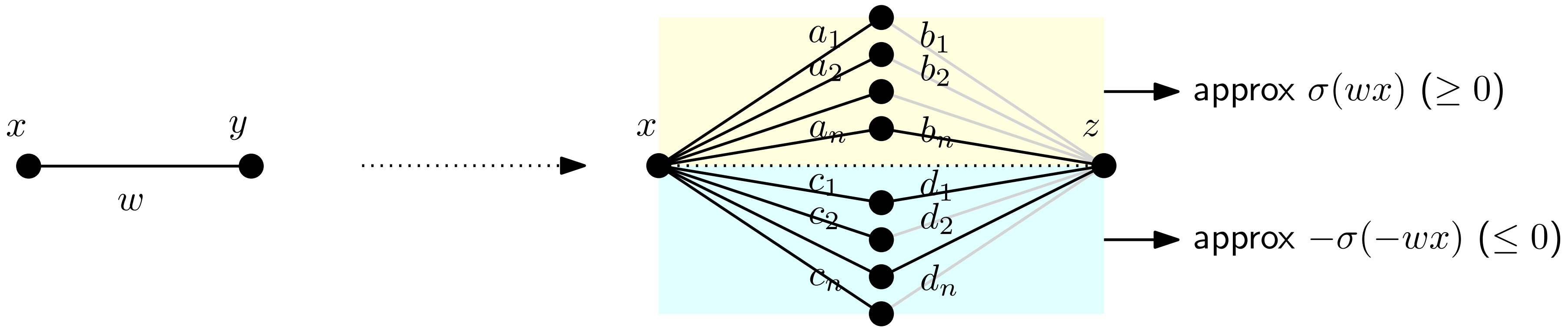
Putting everything together



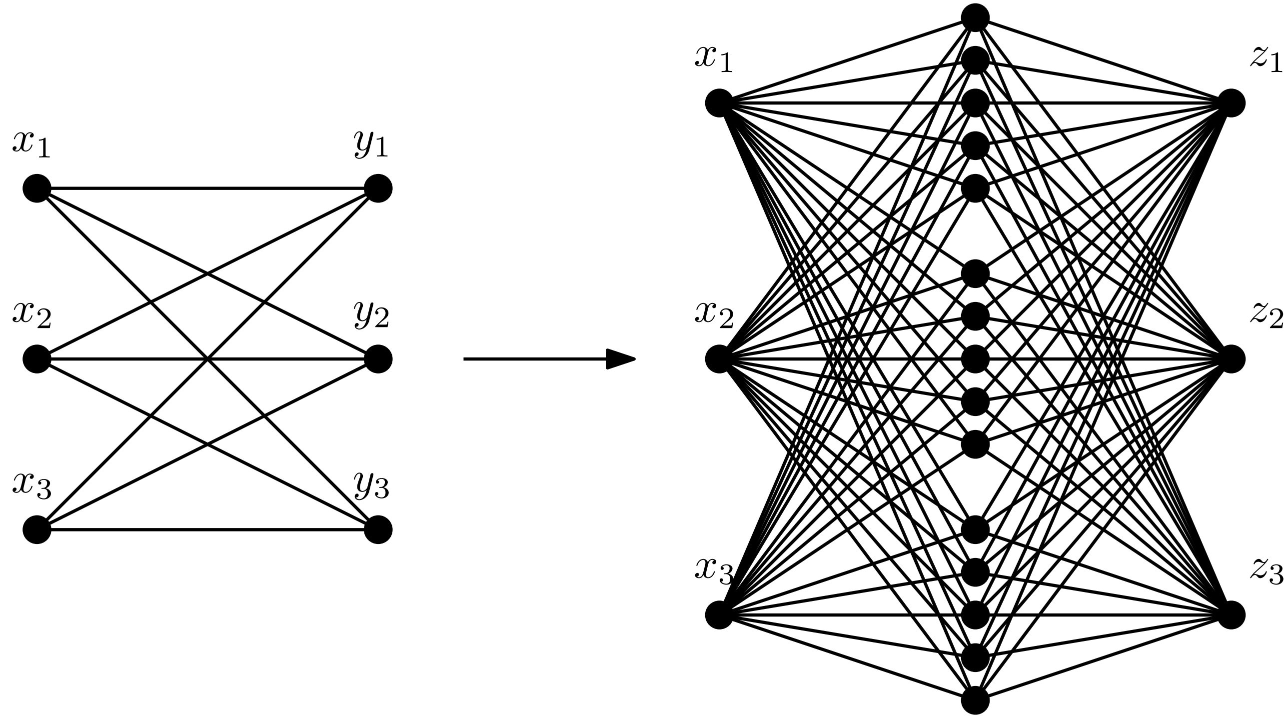
prune only the right layer: reuse the left layer



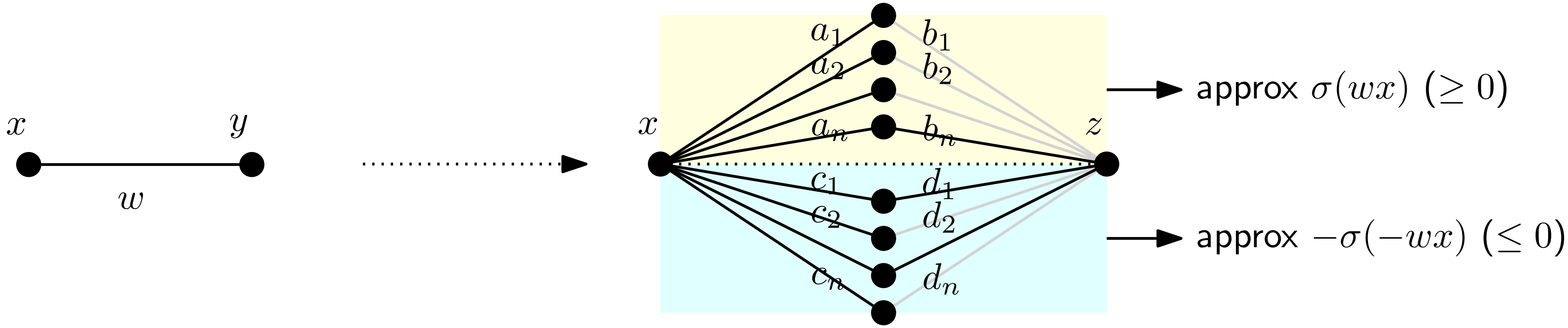
Putting everything together



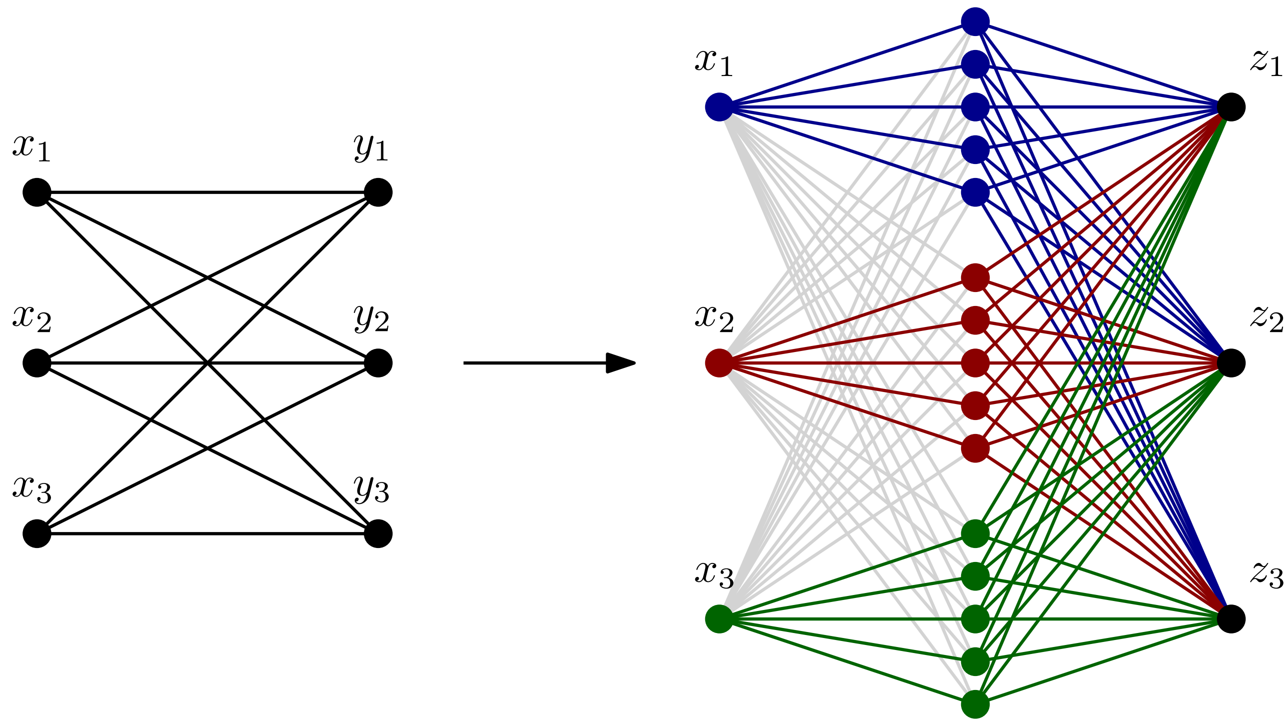
prune only the right layer: reuse the left layer



Putting everything together



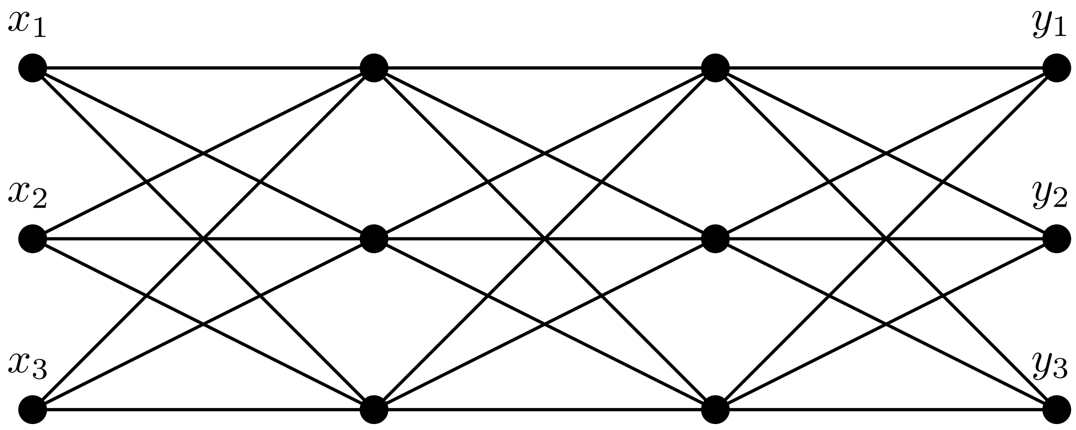
prune only the right layer: reuse the left layer



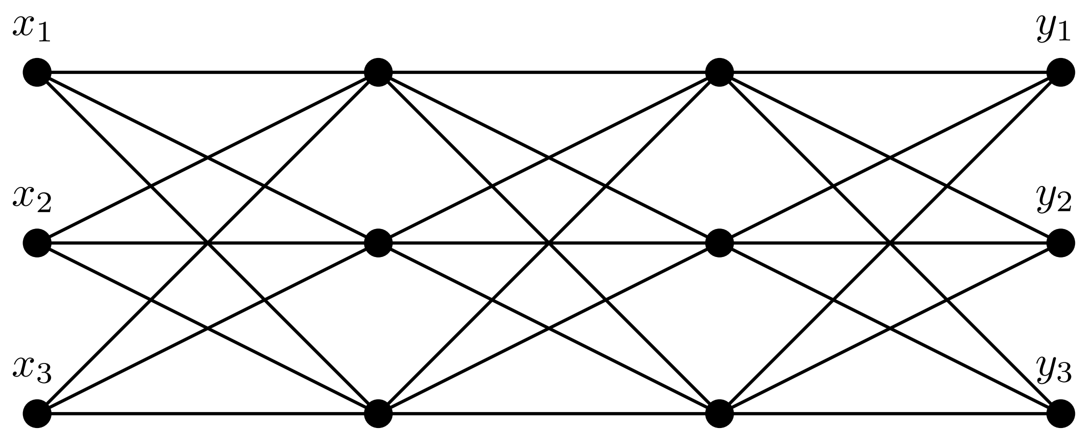
$n \geq C \log d^2/\varepsilon$

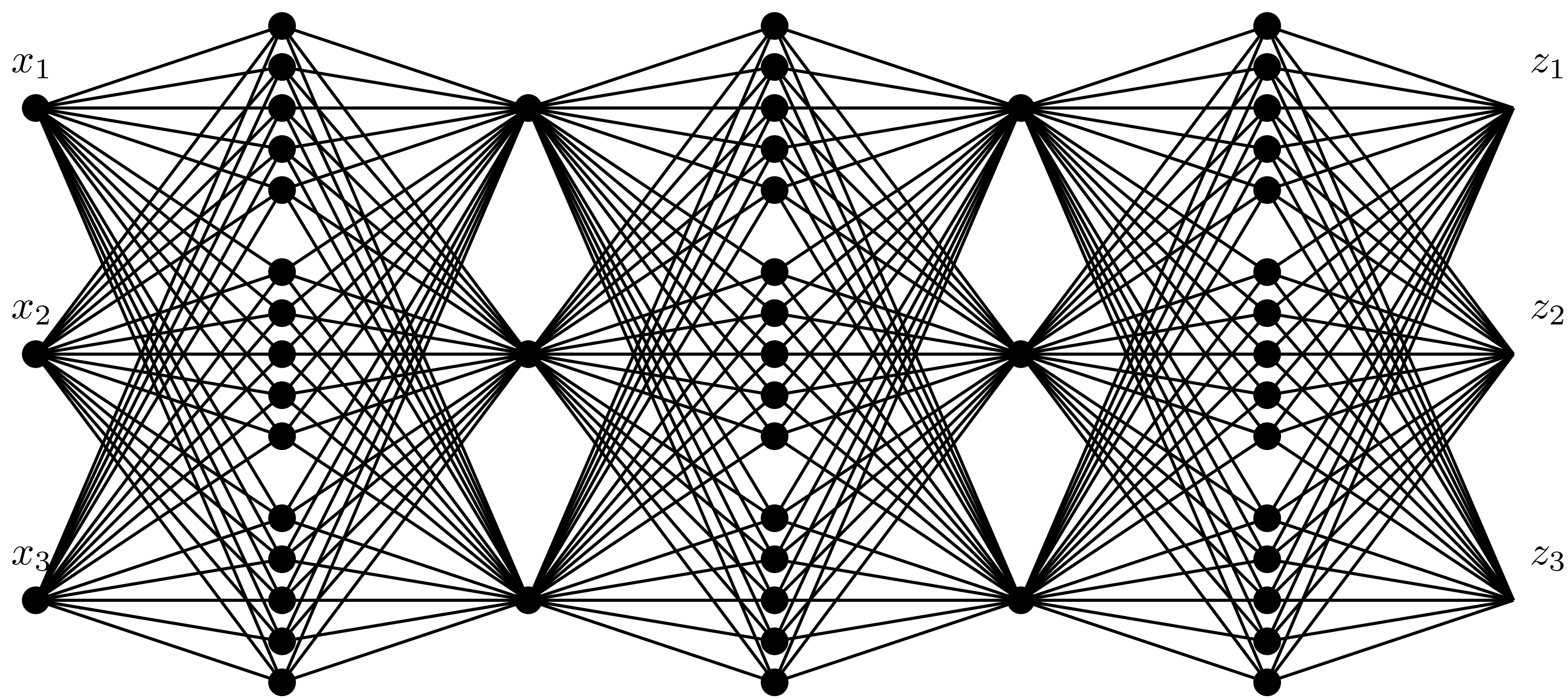
$d =$ width original layer

More layers together



More layers together



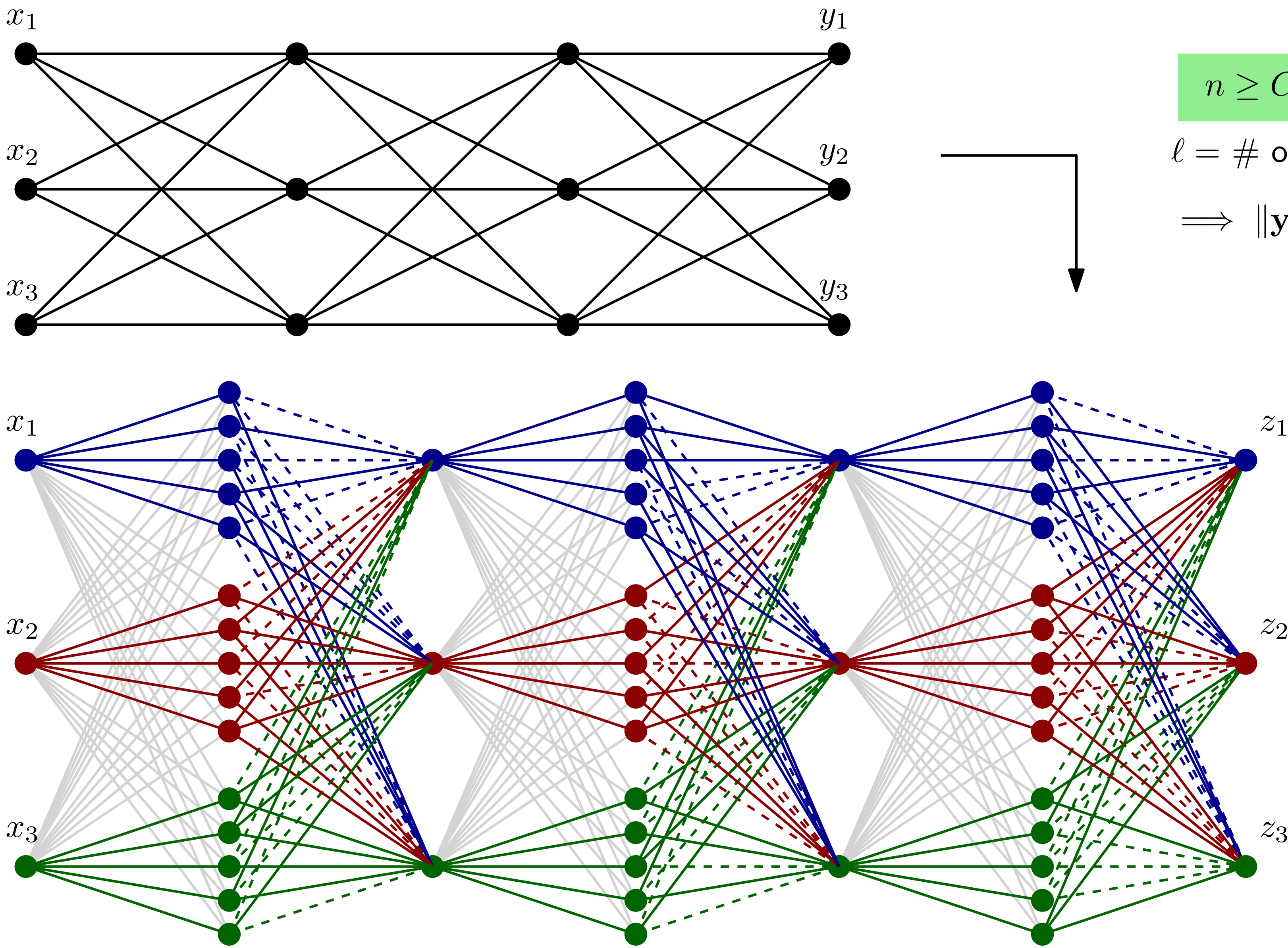


$n \geq C \log \ell d^2/\varepsilon$

$\ell = \#$ original layers

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

More layers together



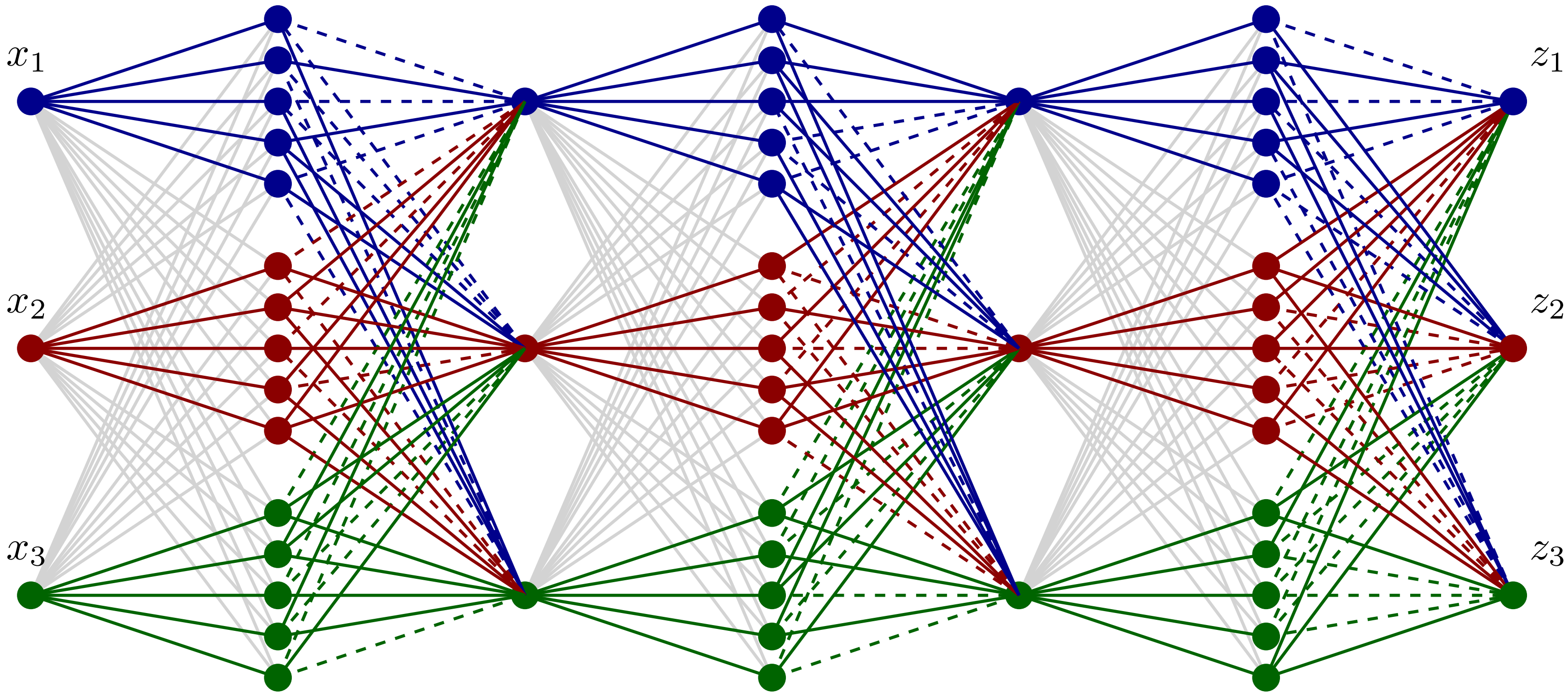
$n \geq C \log \ell d^2/\varepsilon$

$\ell = \#$ original layers

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

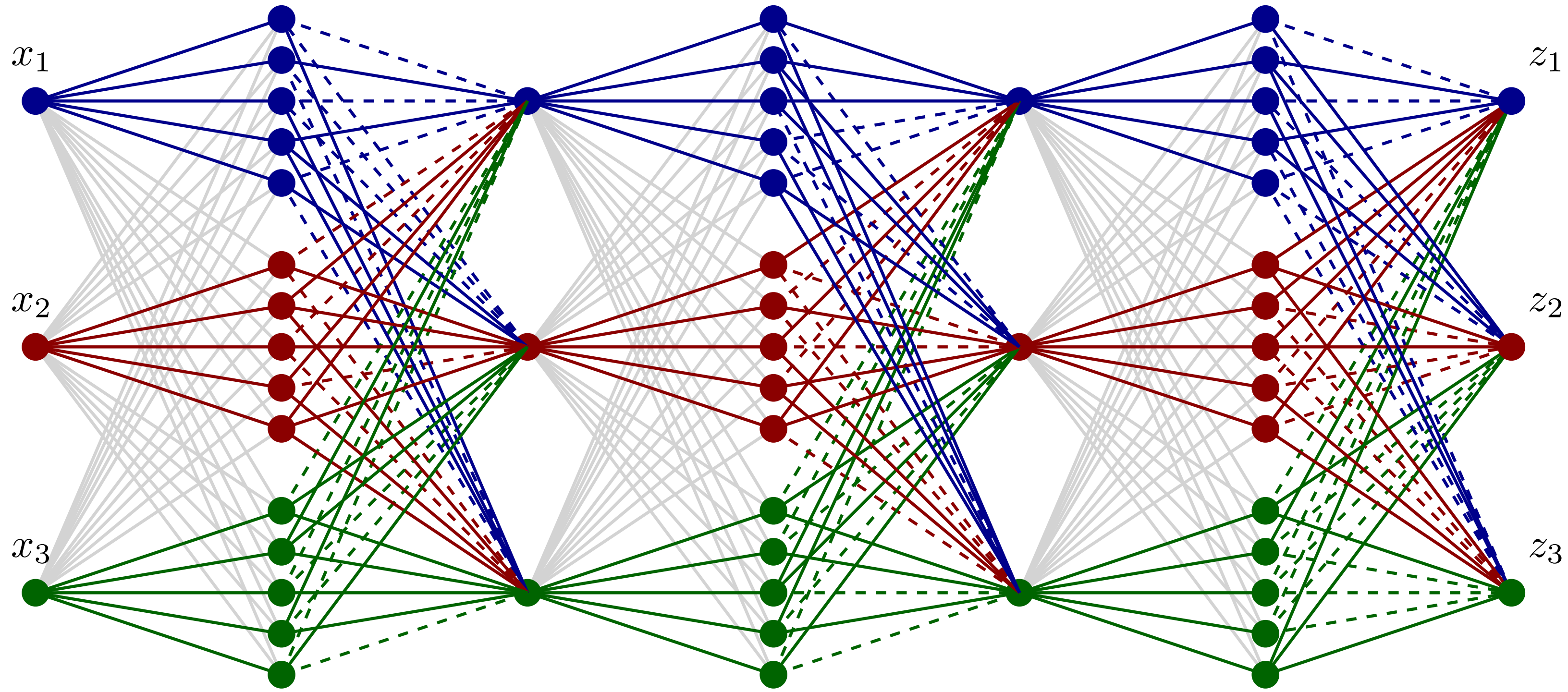
Unstructured pruning

- Removed edges can be everywhere



Unstructured pruning

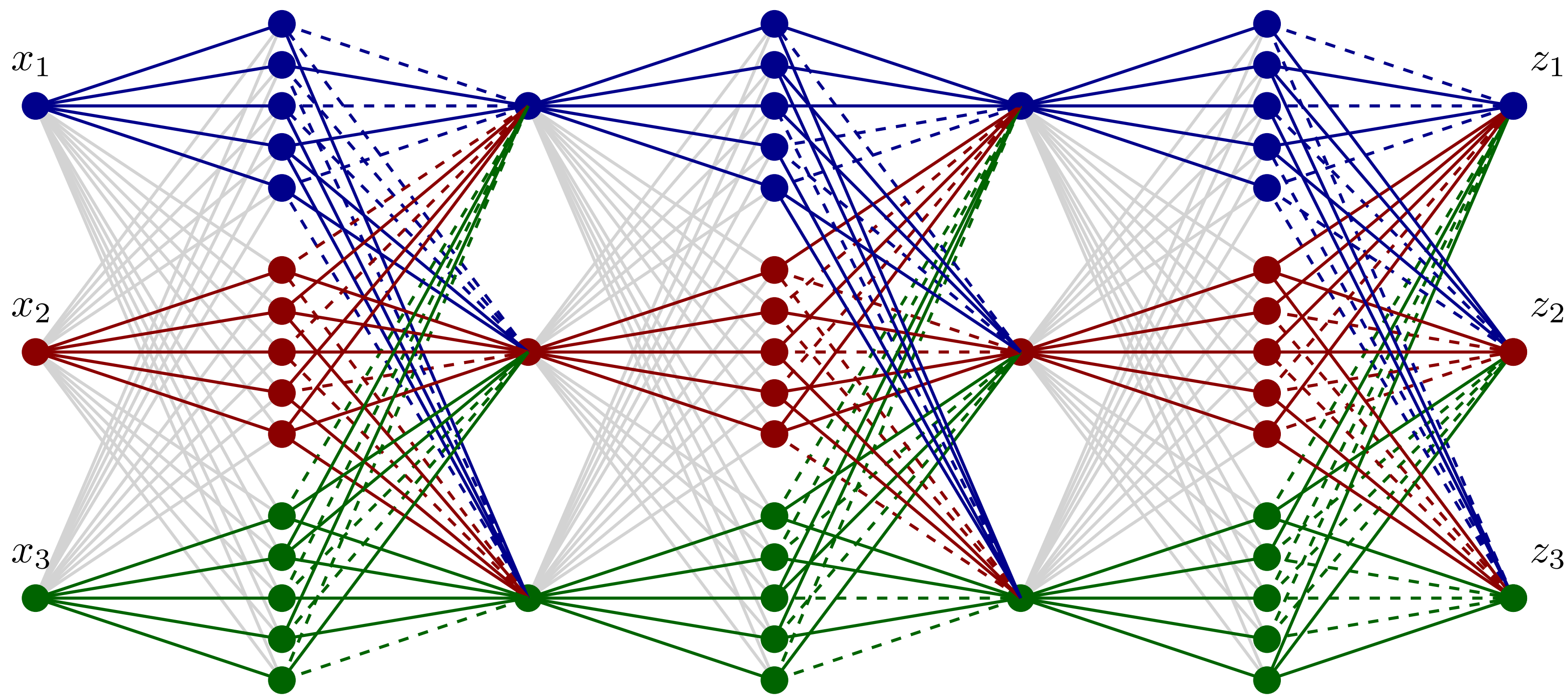
- Removed edges can be everywhere



- No structure usually implies slower processes

Unstructured pruning

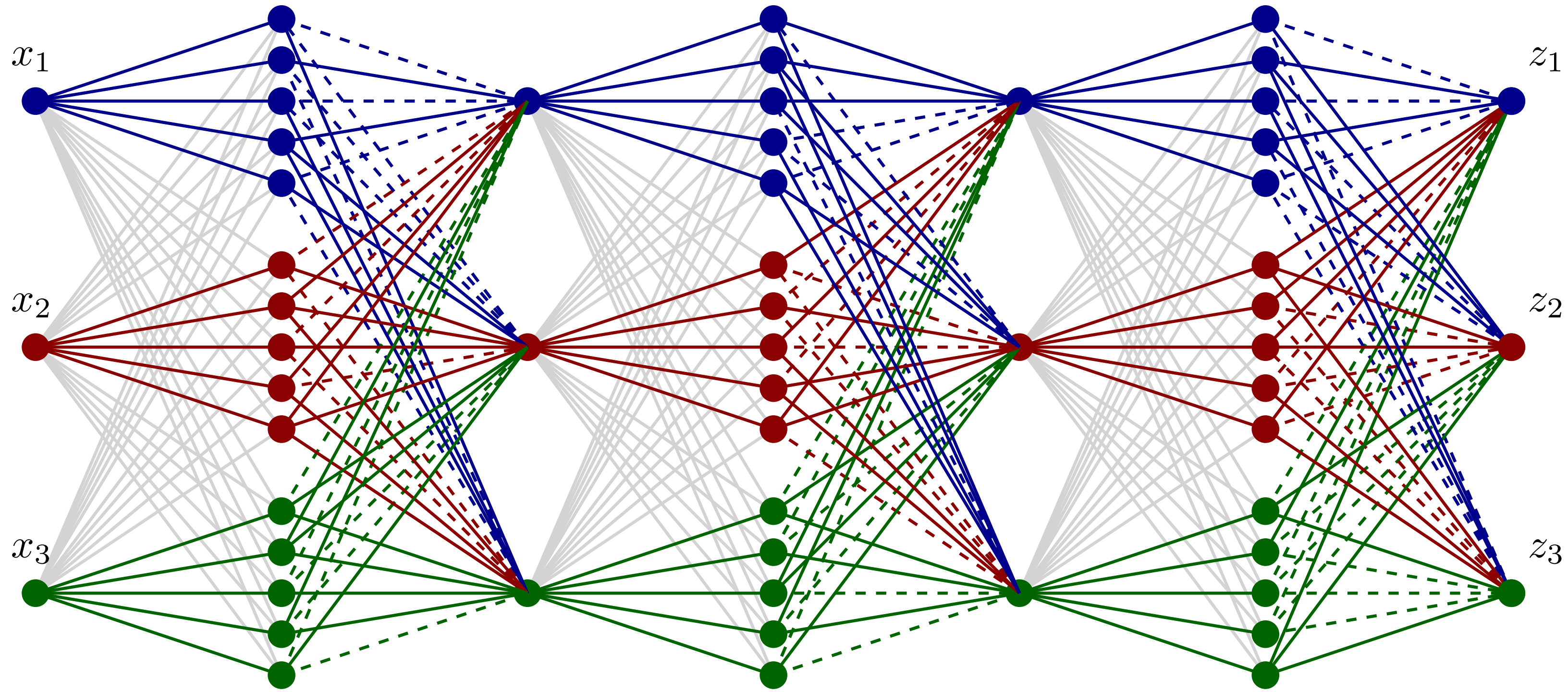
- Removed edges can be everywhere



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

Unstructured pruning

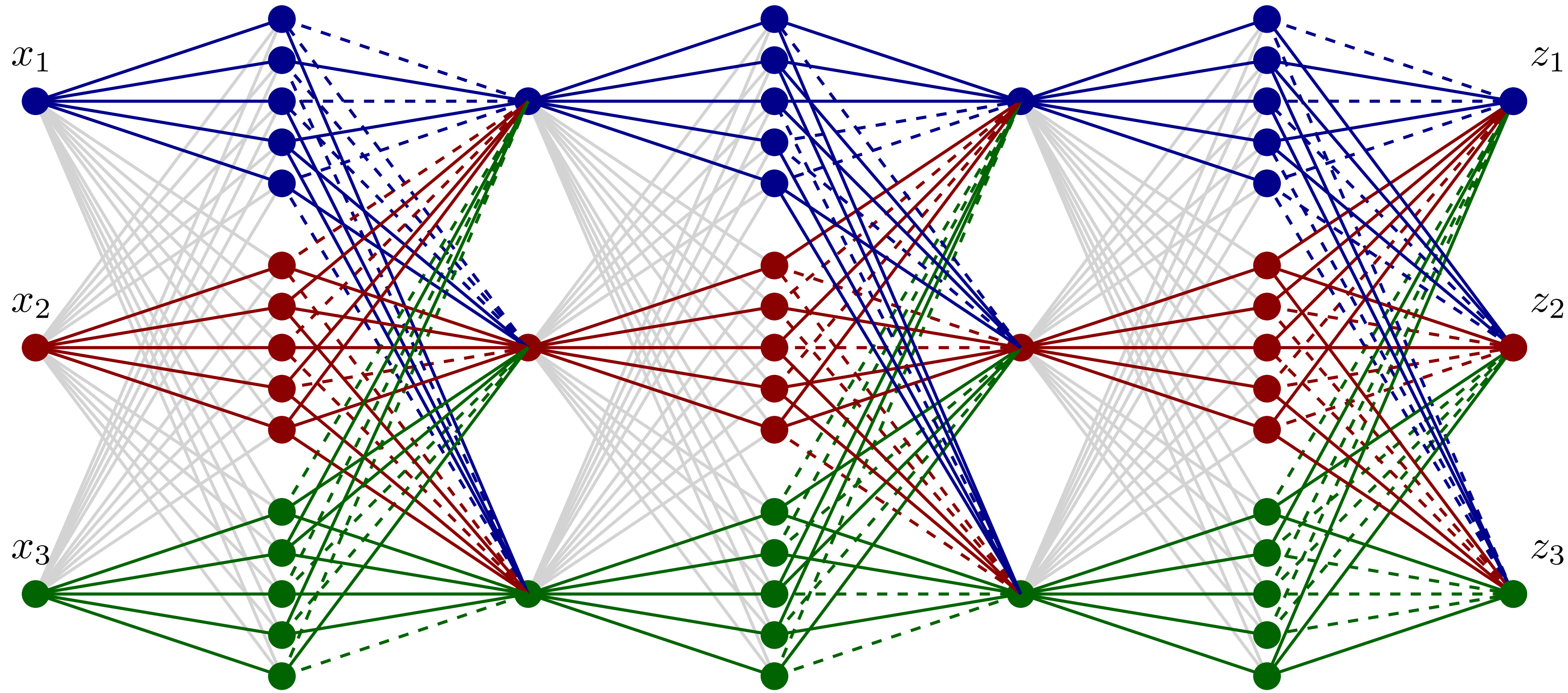
- Removed edges can be everywhere



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

Unstructured pruning

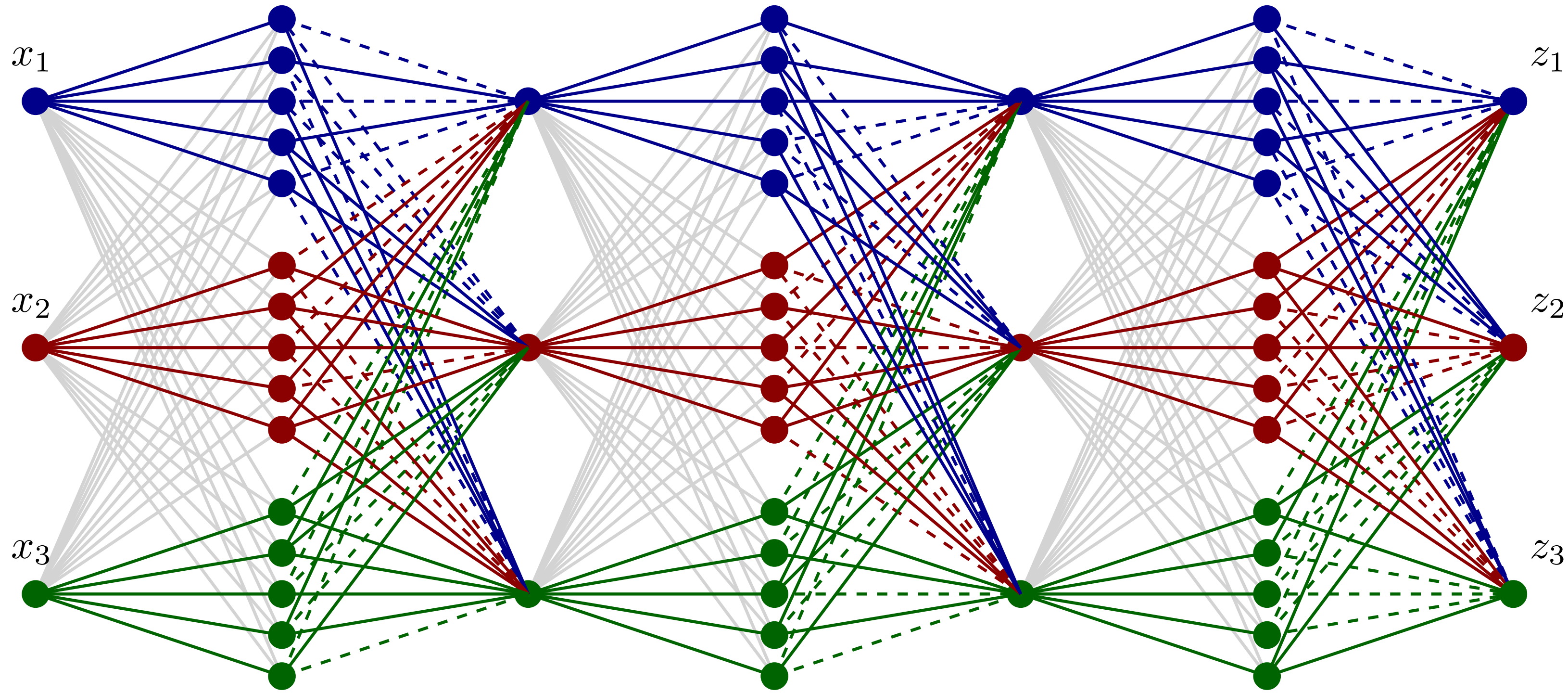
- Removed edges can be everywhere



- No structure usually implies slower processes
  - difficulty encoding unstructured sparsity
  - accessing data is more time consuming than processing
  - the processor register allows parallel operations for blocks of memory

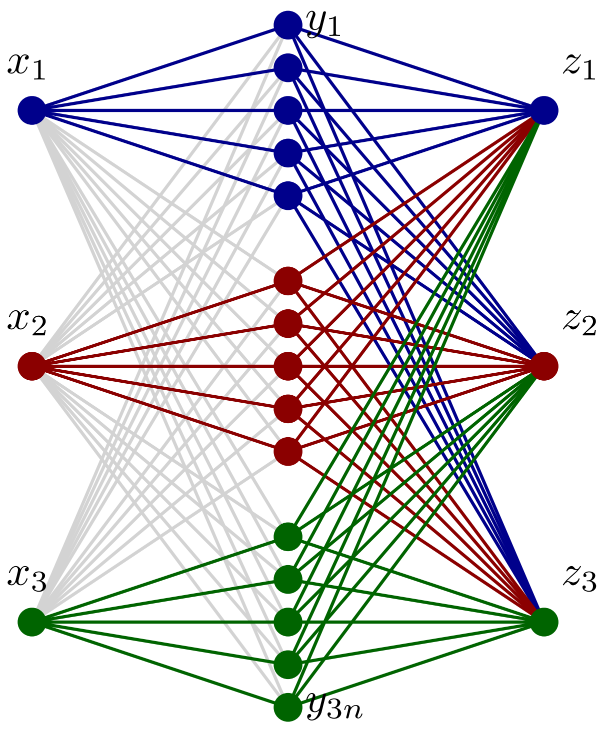
Unstructured pruning

- Removed edges can be everywhere

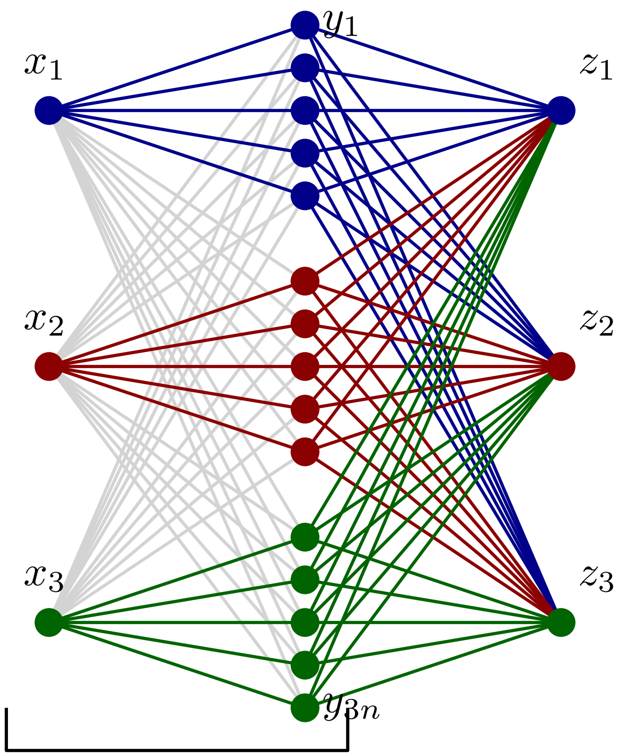


- No structure usually implies slower processes
  - difficulty encoding unstructured sparsity
  - accessing data is more time consuming than processing
  - the processor register allows parallel operations for blocks of memory
- [Malach et al. ICML '20]: pruning neurons alone requires exponential overparameterization

Structured pruning

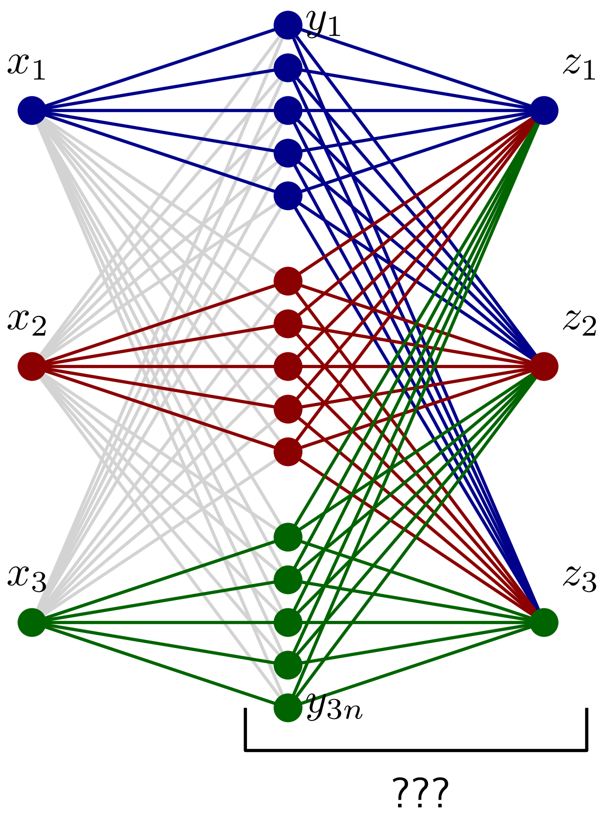


Structured pruning

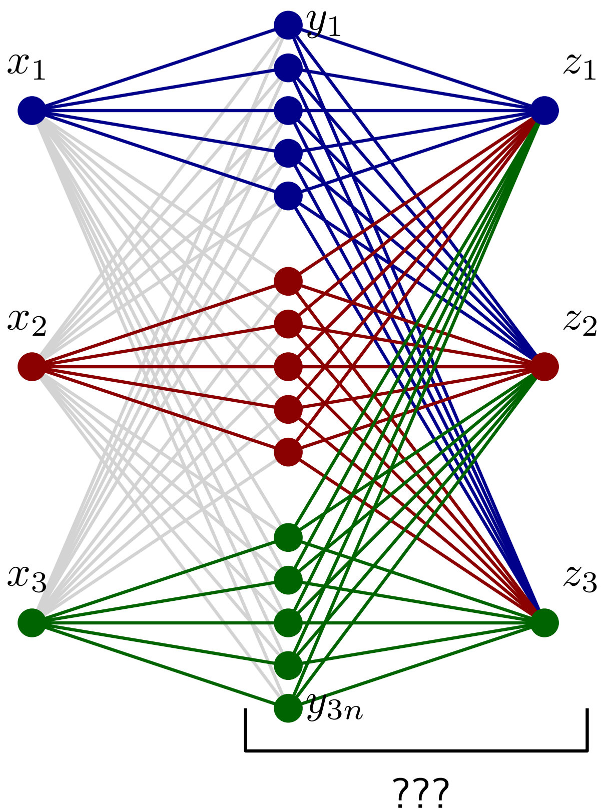


$$\begin{bmatrix} w_{1,1} & 0 & 0 \\ \vdots & \vdots & \vdots \\ w_{n,1} & 0 & 0 \\ 0 & w_{1,2} & 0 \\ \vdots & \vdots & \vdots \\ 0 & w_{n,2} & 0 \\ 0 & 0 & w_{1,3} \\ \vdots & \vdots & \vdots \\ 0 & 0 & w_{n,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Structured pruning

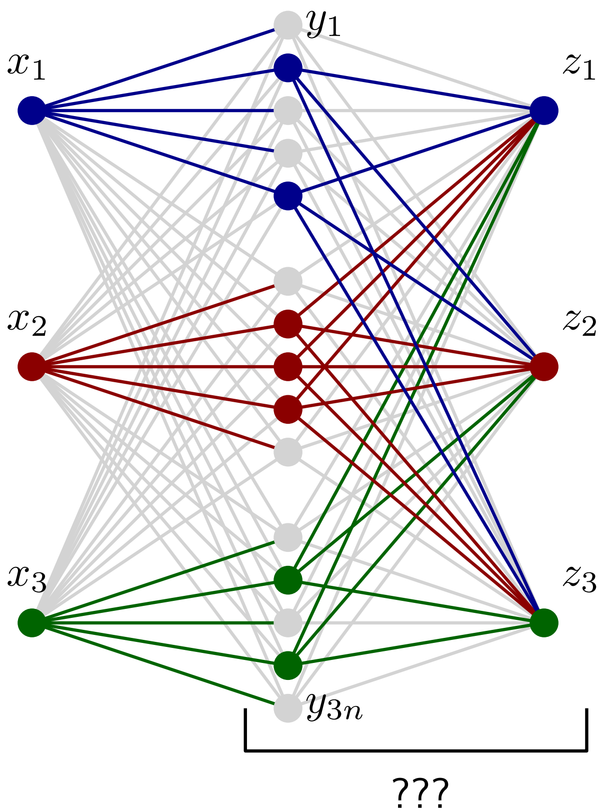


Structured pruning



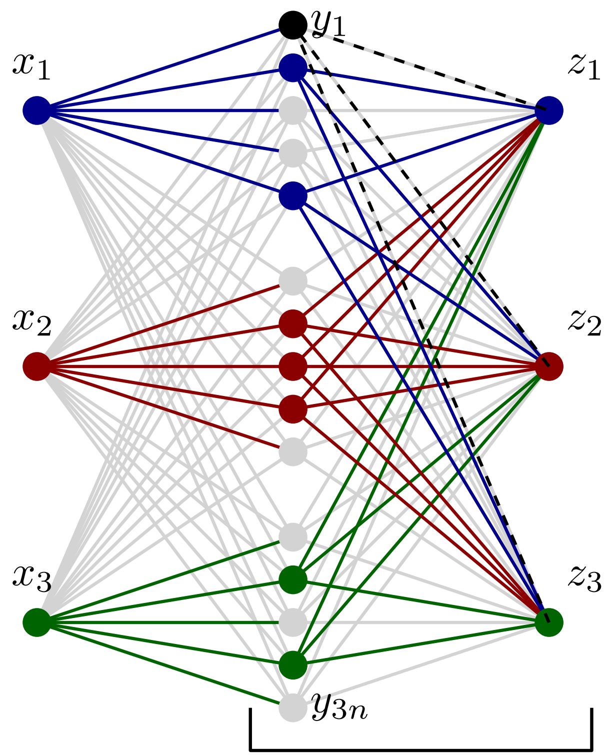
- Removing entire neurons from the middle layer!

Structured pruning



- Removing entire neurons from the middle layer!

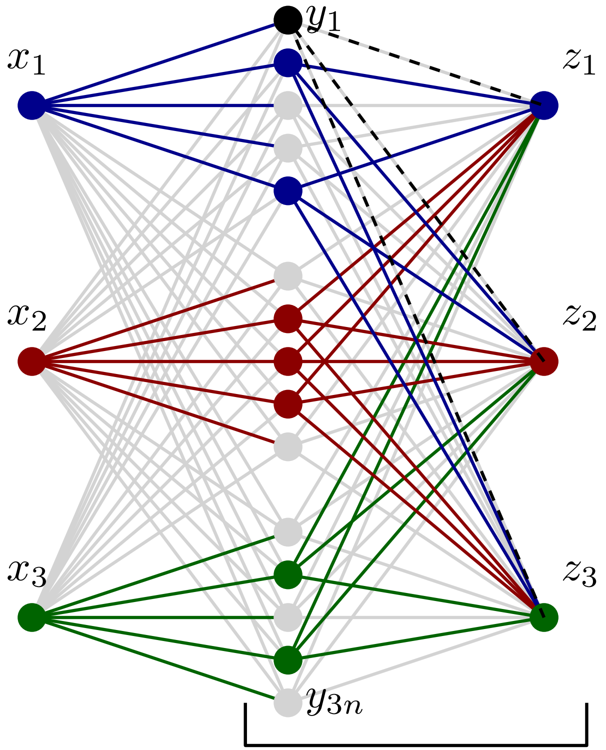
Structured pruning



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

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

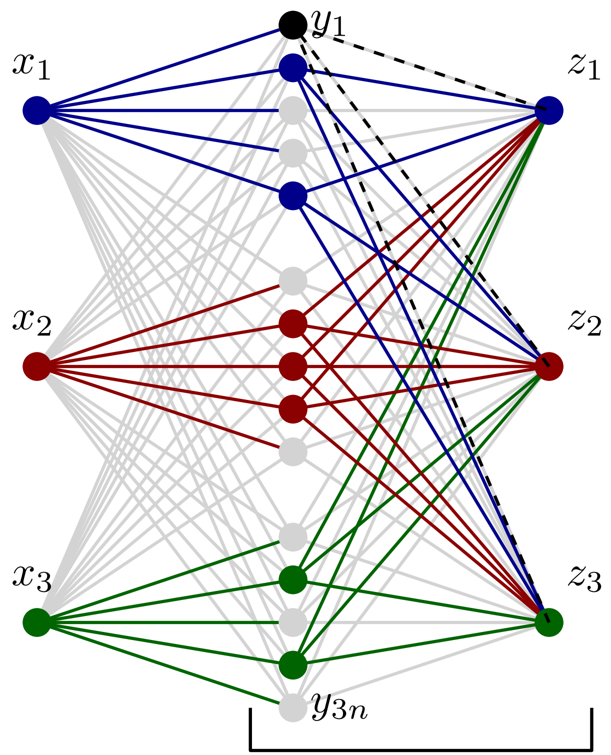
Structured pruning



- 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

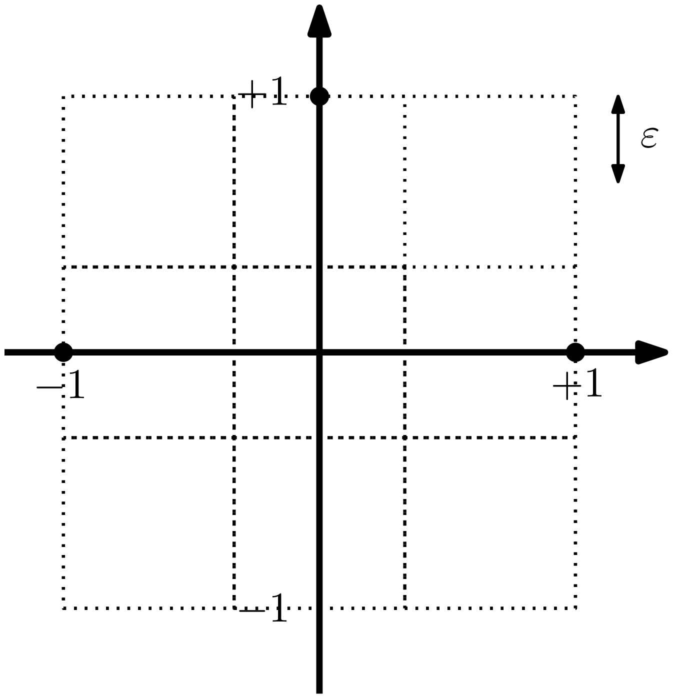


- Removing entire neurons from the middle layer!
  - removes columns!
- The one-dimensional RSS result does not work
  - leads to exponential bounds
- A multidimensional RSS result is required

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

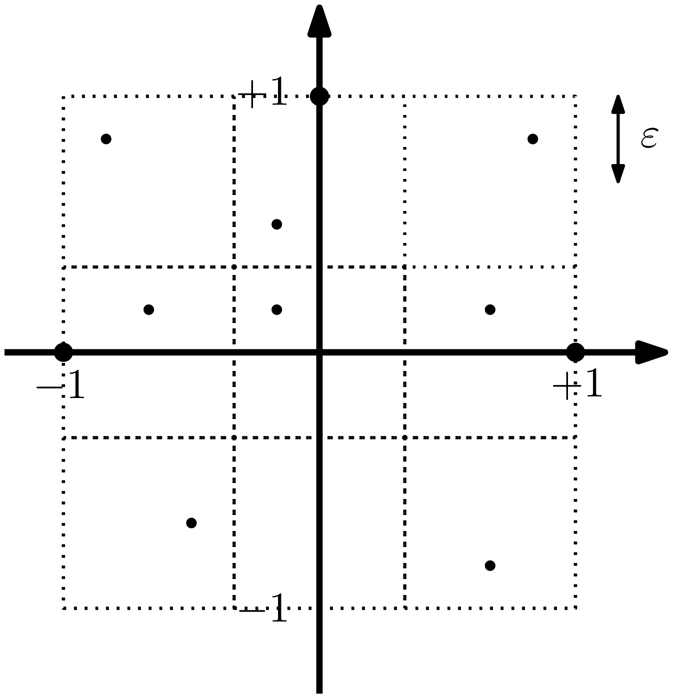
The multidimensional RSS problem

- Natural generalization



The multidimensional RSS problem

- Natural generalization



Input:

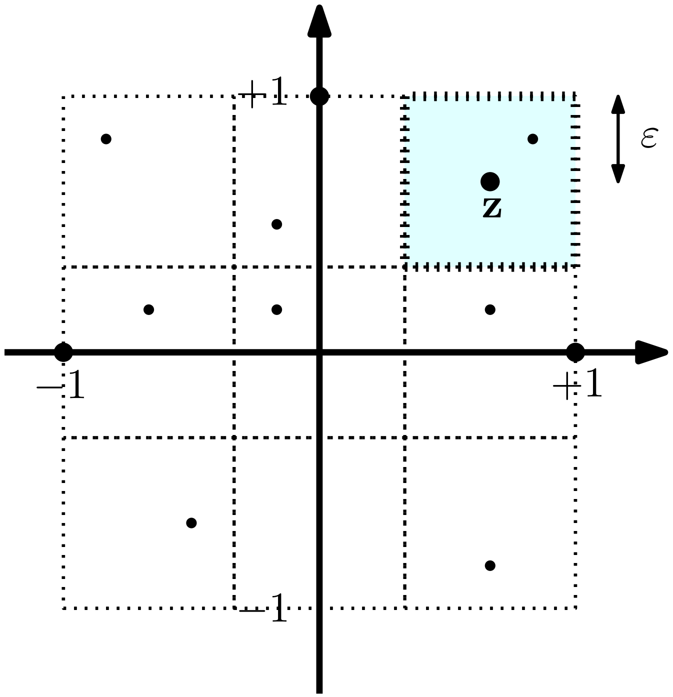
- Sequence of n i.i.d. random vectors $X_1, \dots, X_n$

The multidimensional RSS problem

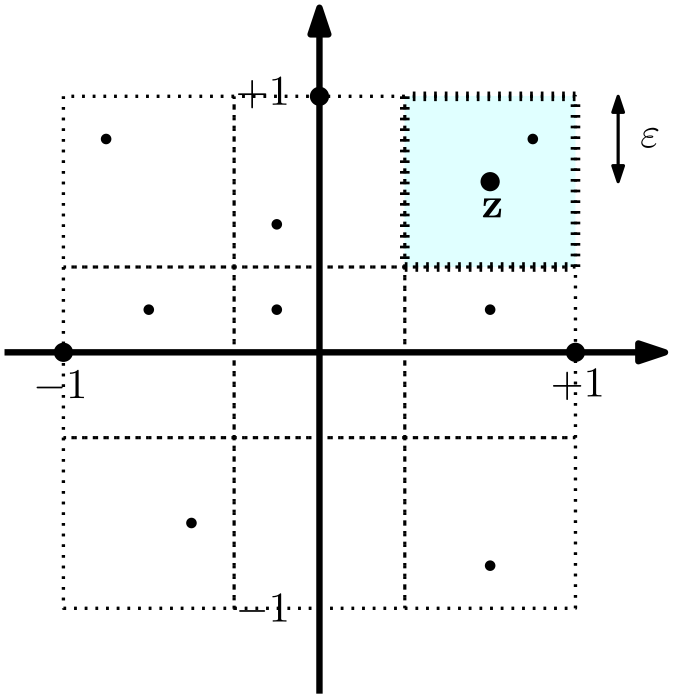
- Natural generalization

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:

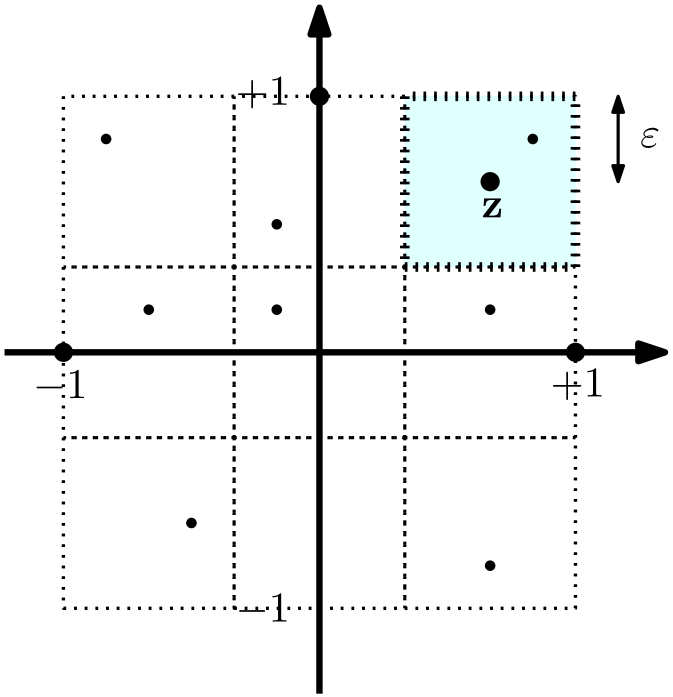
• Sequence of n i.i.d. random vectors $X_1, \ldots, X_n$

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

• Error parameter $\varepsilon > 0$

The multidimensional RSS problem

- Natural generalization



Input:

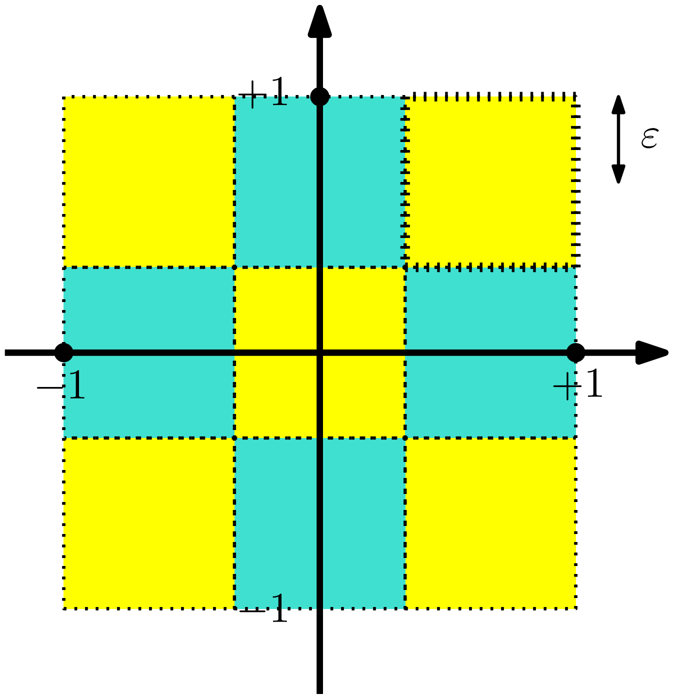
- Sequence of n i.i.d. random vectors $X_1, \ldots, X_n$
- Target vector $\mathbf{z} \in [-1, +1]^d$
- Error parameter $\varepsilon > 0$

Question:

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

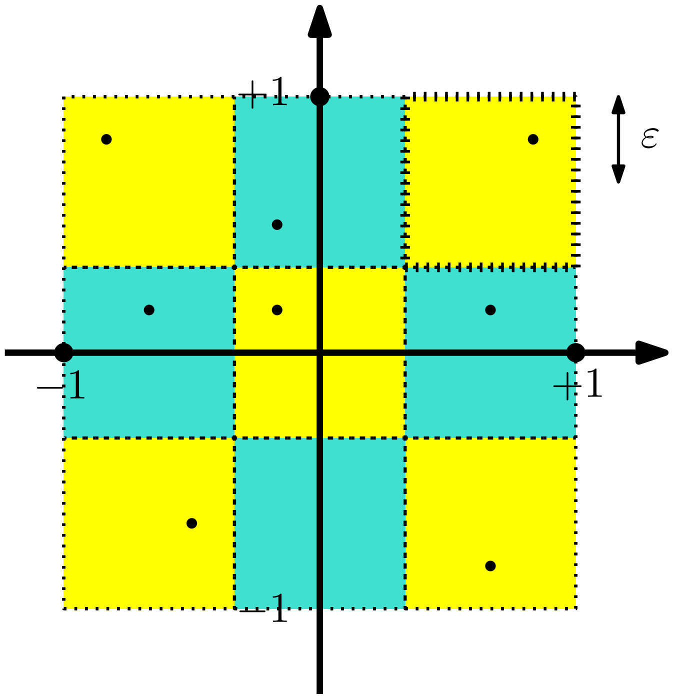
MRSS in expectation

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



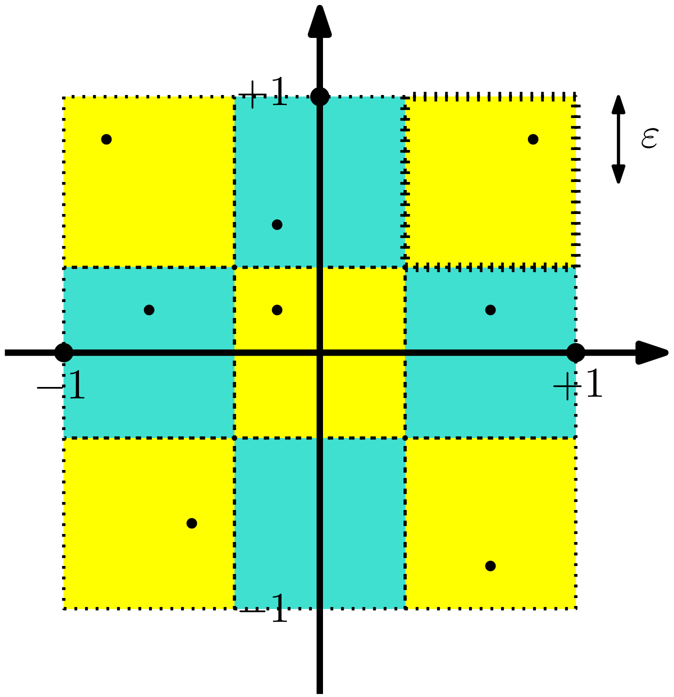
MRSS in expectation

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



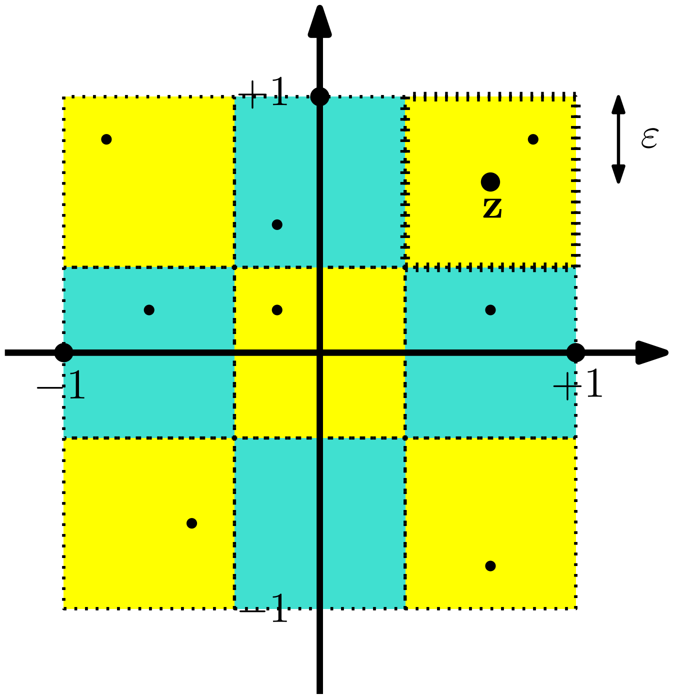
MRSS in expectation

- 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

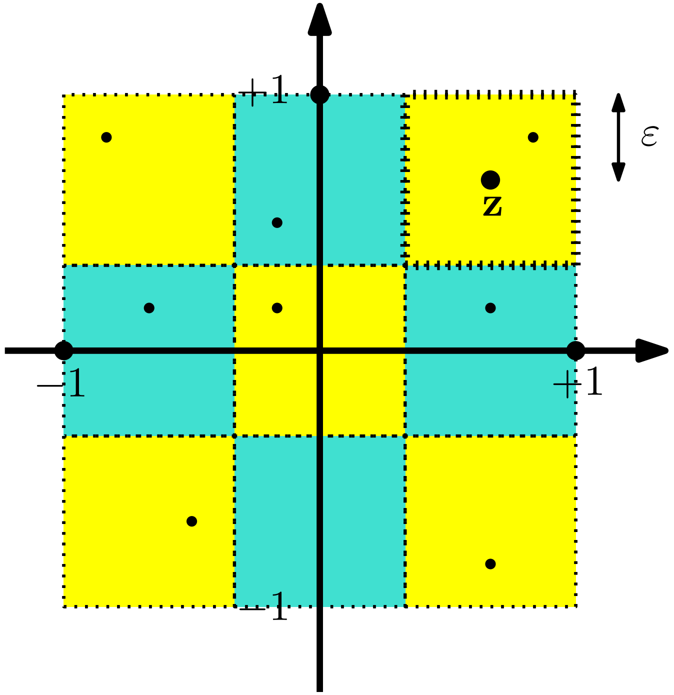


MRSS in expectation

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



MRSS in expectation

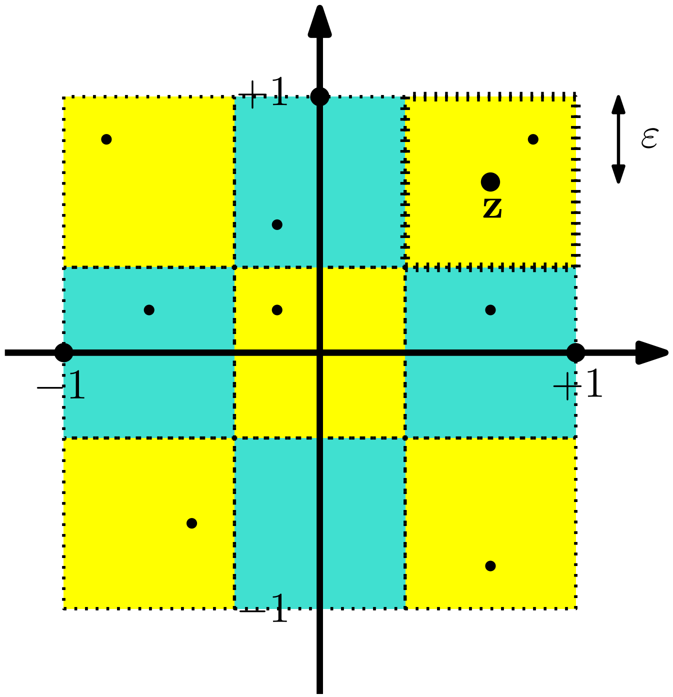


- Number of ε-cubes: $1/\varepsilon^d = 2^{d \log 1/\varepsilon}$
- Sequence of n i.i.d. random vectors $X_1, \ldots, X_n \sim \mathcal{N}(\mathbf{0}, I_d)$
- 2^n possible subsets
- Target $\mathbf{z} \in [-1, 1]^d$

Upper bound

MRSS in expectation

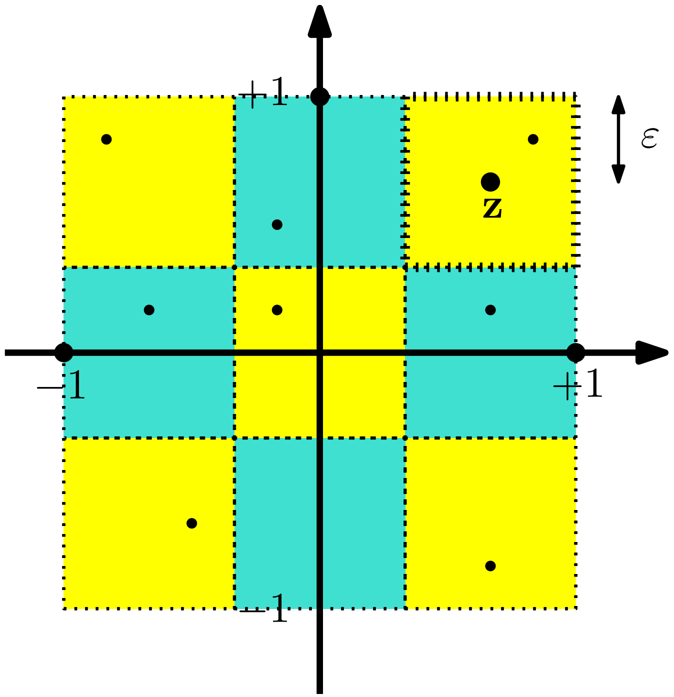
- Number of ε-cubes: $1/\varepsilon^d = 2^{d \log 1/\varepsilon}$
- Sequence of n i.i.d. random vectors $X_1, \ldots, X_n \sim \mathcal{N}(\mathbf{0}, I_d)$
- 2^n possible subsets
- Target $\mathbf{z} \in [-1, 1]^d$



Upper bound

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

MRSS in expectation

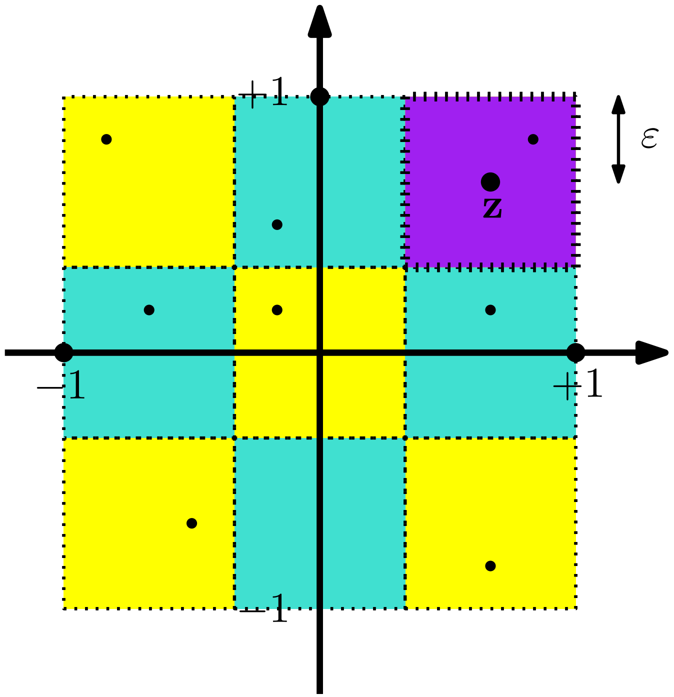


- Number of ε-cubes: $1/\varepsilon^d = 2^{d \log 1/\varepsilon}$
- Sequence of n i.i.d. random vectors $X_1, \ldots, X_n \sim \mathcal{N}(\mathbf{0}, I_d)$
- 2^n possible subsets
- Target $\mathbf{z} \in [-1, 1]^d$

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

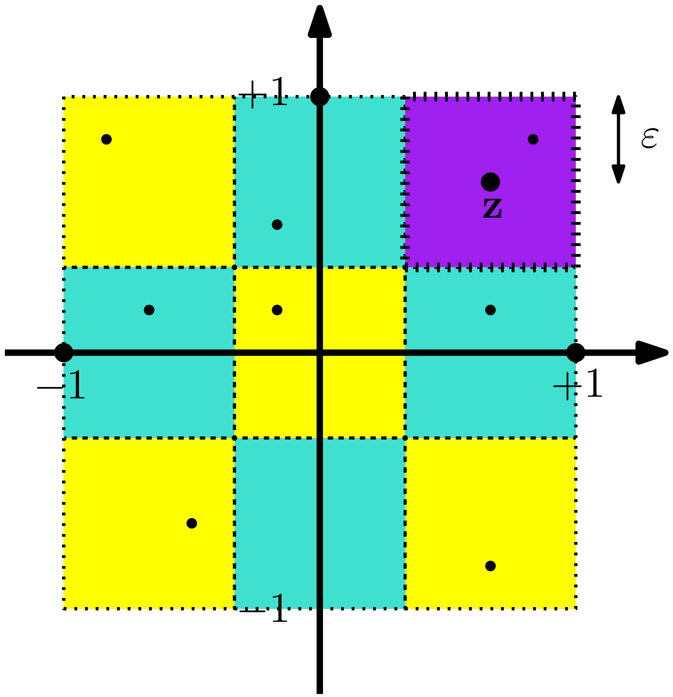


- Number of ε-cubes: $1/\varepsilon^d = 2^{d \log 1/\varepsilon}$
- Sequence of n i.i.d. random vectors $X_1, \ldots, X_n \sim \mathcal{N}(\mathbf{0}, I_d)$
- 2^n possible subsets
- Target $\mathbf{z} \in [-1, 1]^d$

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

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: $\binom{n}{k} \le (en/k)^k$
- 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

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

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

< 1 if $n \le cd\log 1/\varepsilon$ 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

- $Z_n = \sum_{S \subseteq [n]} Y_S$ 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

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

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

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

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

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

- **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

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

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

- **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

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

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

- **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

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

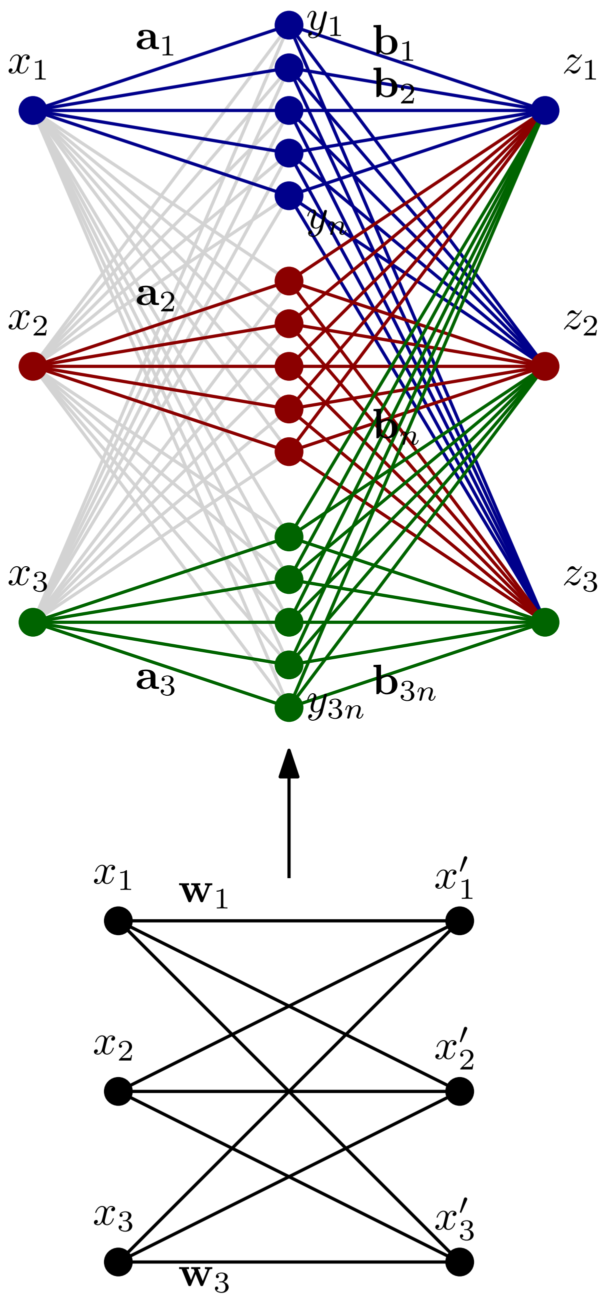
- $\mathsf{P}\left[Z_n \geq 1\right] \geq \left(\mathbb{E}\left[Z_n\right]\right)^2 / \mathbb{E}\left[Z_n^2\right]$

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

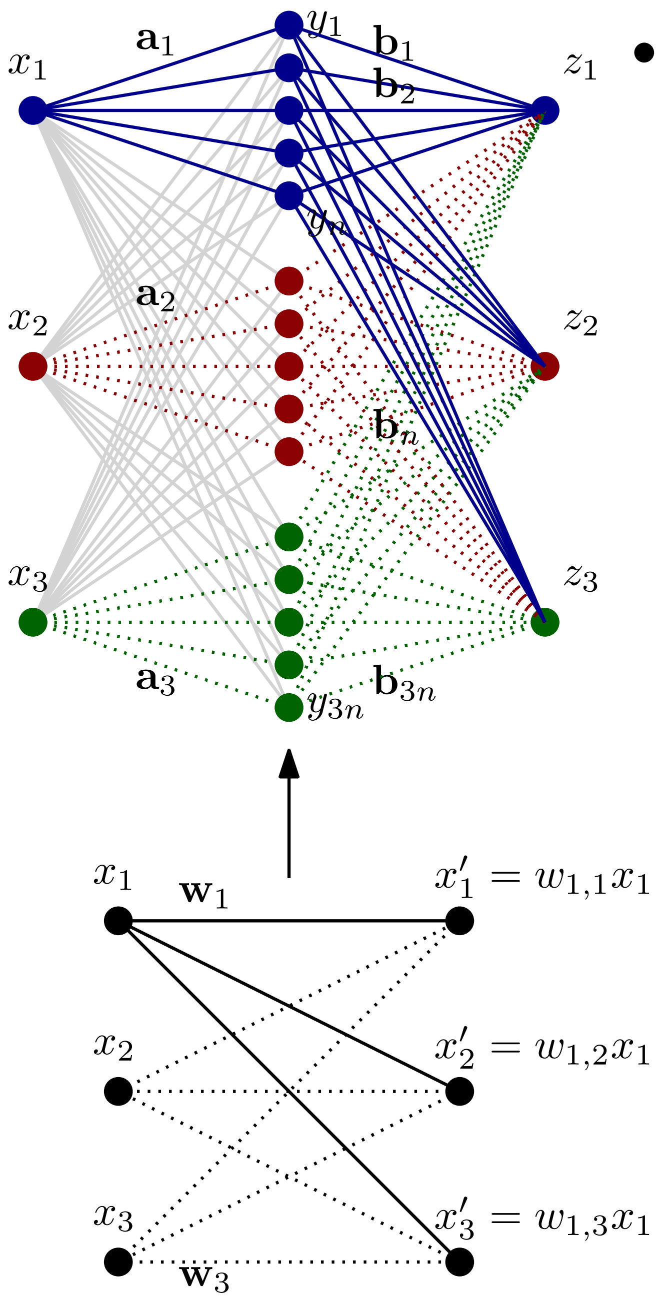
- choose only subsets of size αn so that the "average intersection" concentrates around $\alpha^2 n$

- **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

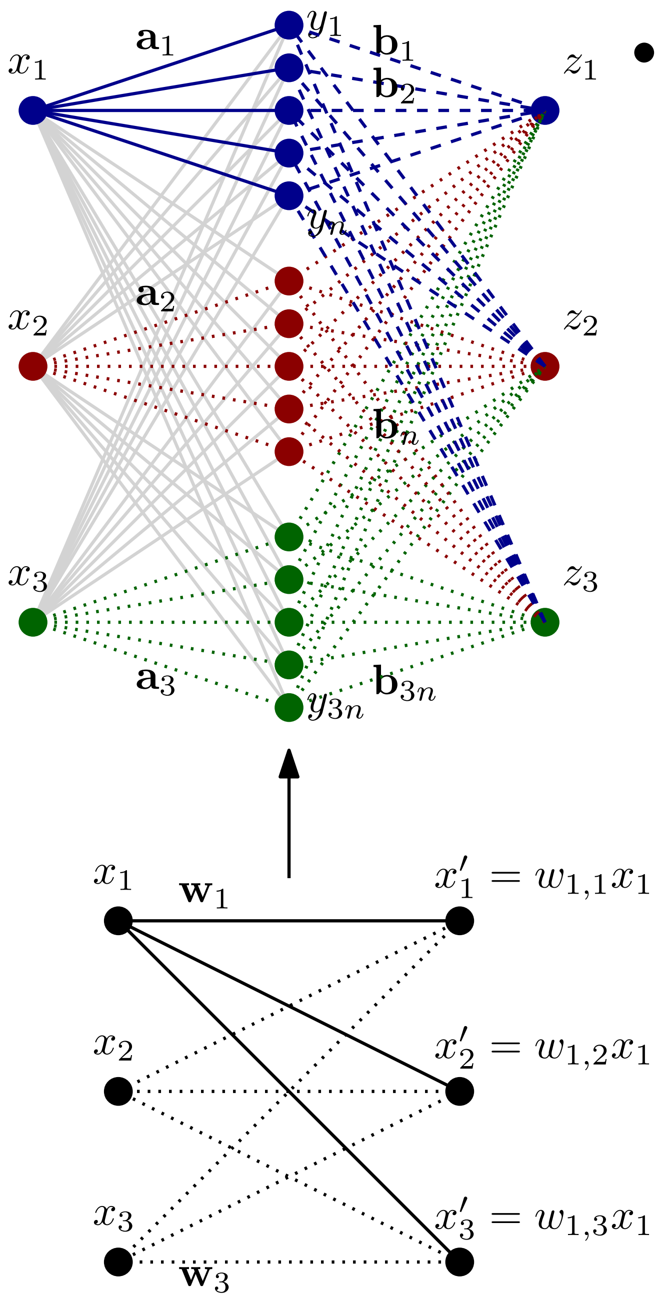


Apply MRSS for structured pruning



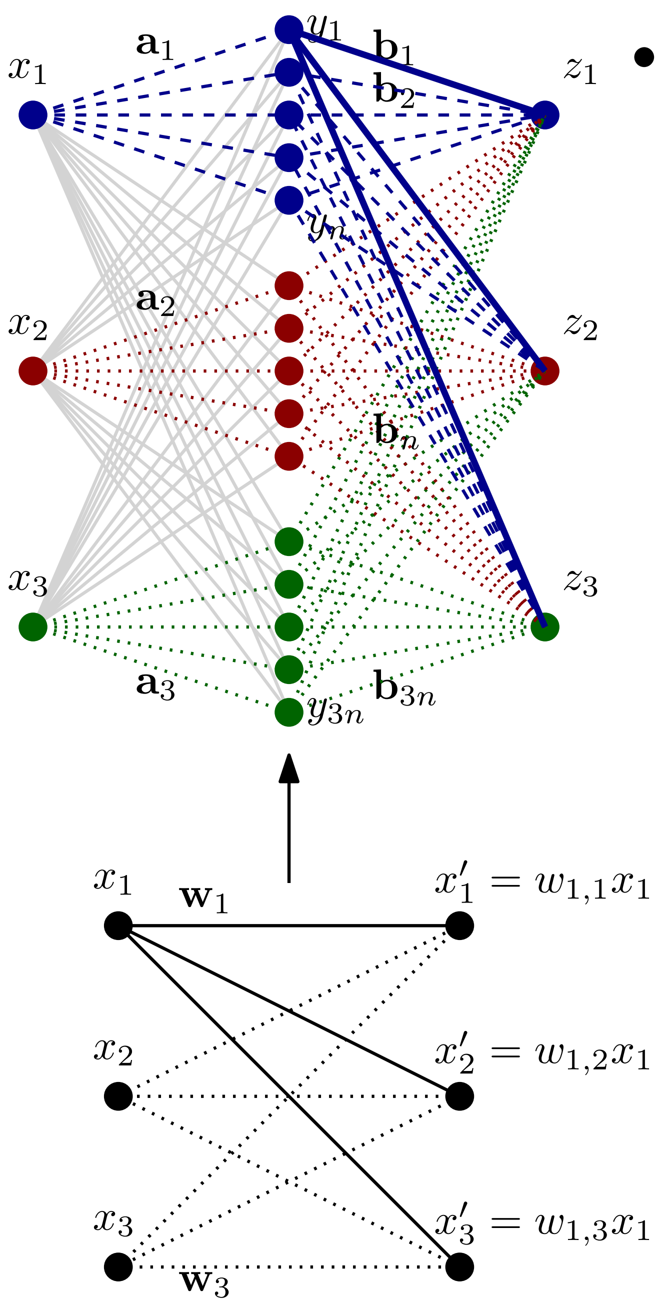
- $\mathbf{a}_i \sim \mathcal{N}(\mathbf{0}, I_n)$, $\mathbf{b}_i \sim \mathcal{N}(\mathbf{0}, I_d)$ (here, $d = 3$)

Apply MRSS for structured pruning



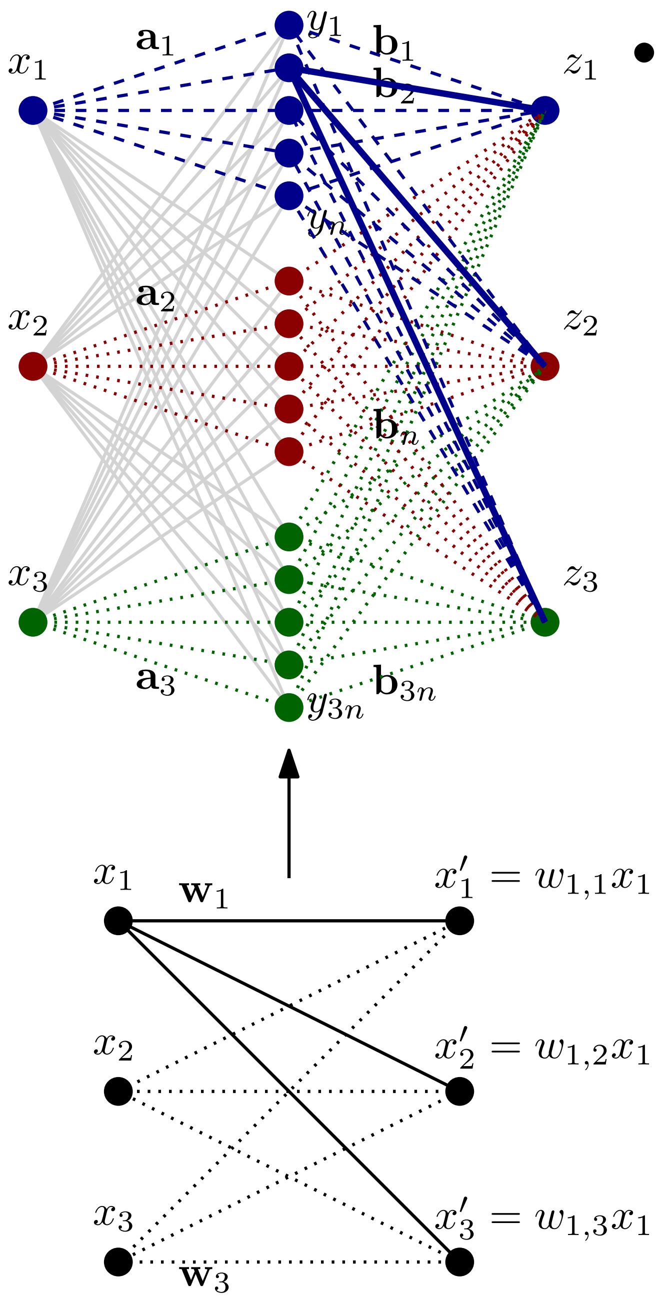
- $\mathbf{a}_i \sim \mathcal{N}(\mathbf{0}, I_n)$

Apply MRSS for structured pruning



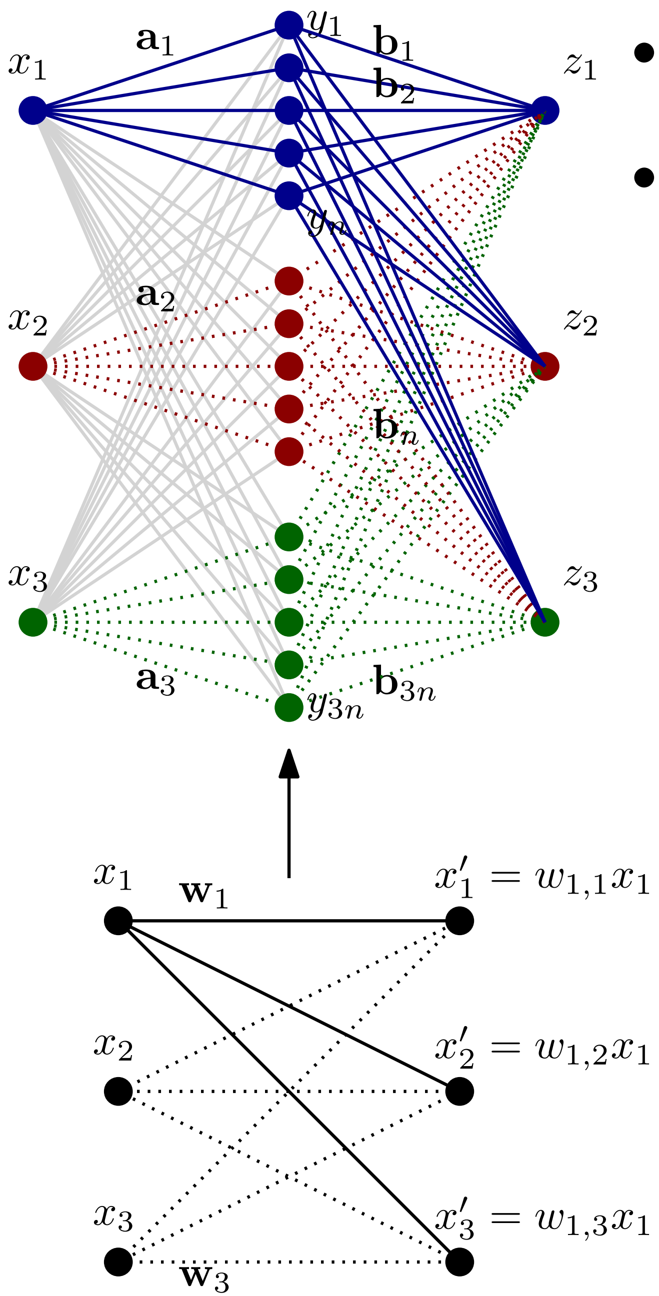
- $\mathbf{a}_i \sim \mathcal{N}(\mathbf{0}, I_n)$, $\mathbf{b}_i \sim \mathcal{N}(\mathbf{0}, I_d)$ (here, $d = 3$)

Apply MRSS for structured pruning



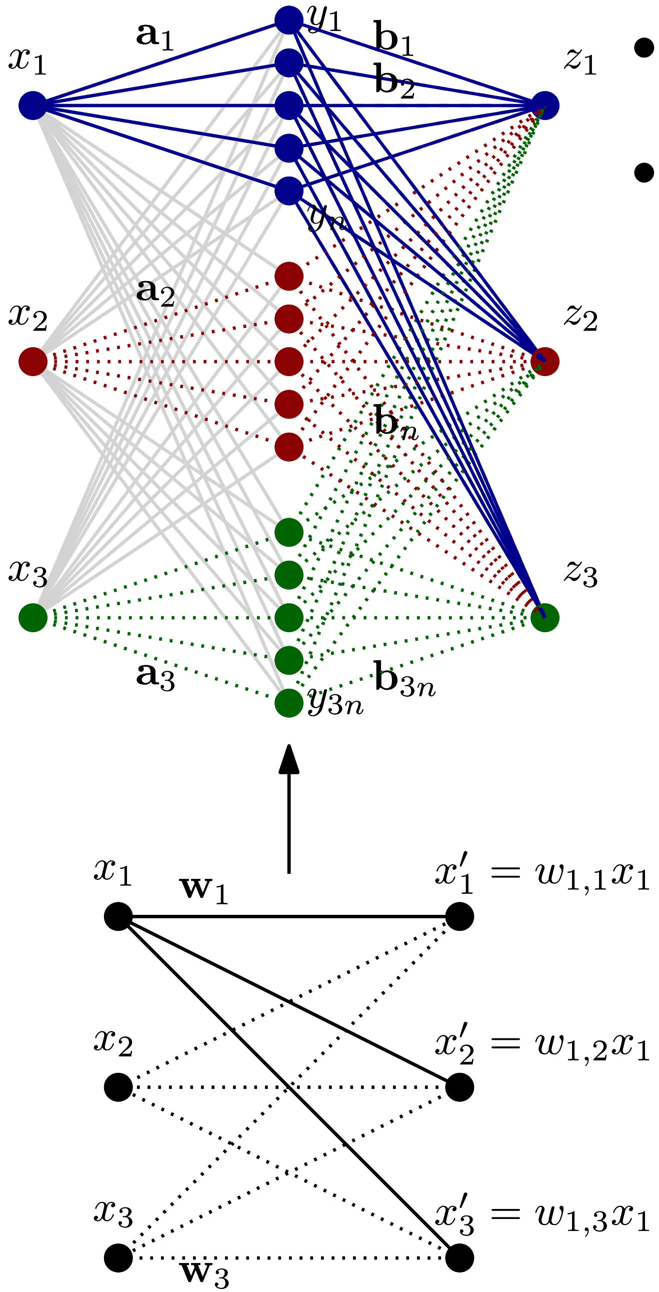
- $\mathbf{a}_i \sim \mathcal{N}(\mathbf{0}, I_n)$, $\mathbf{b}_i \sim \mathcal{N}(\mathbf{0}, I_d)$ (here, $d = 3$)

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**

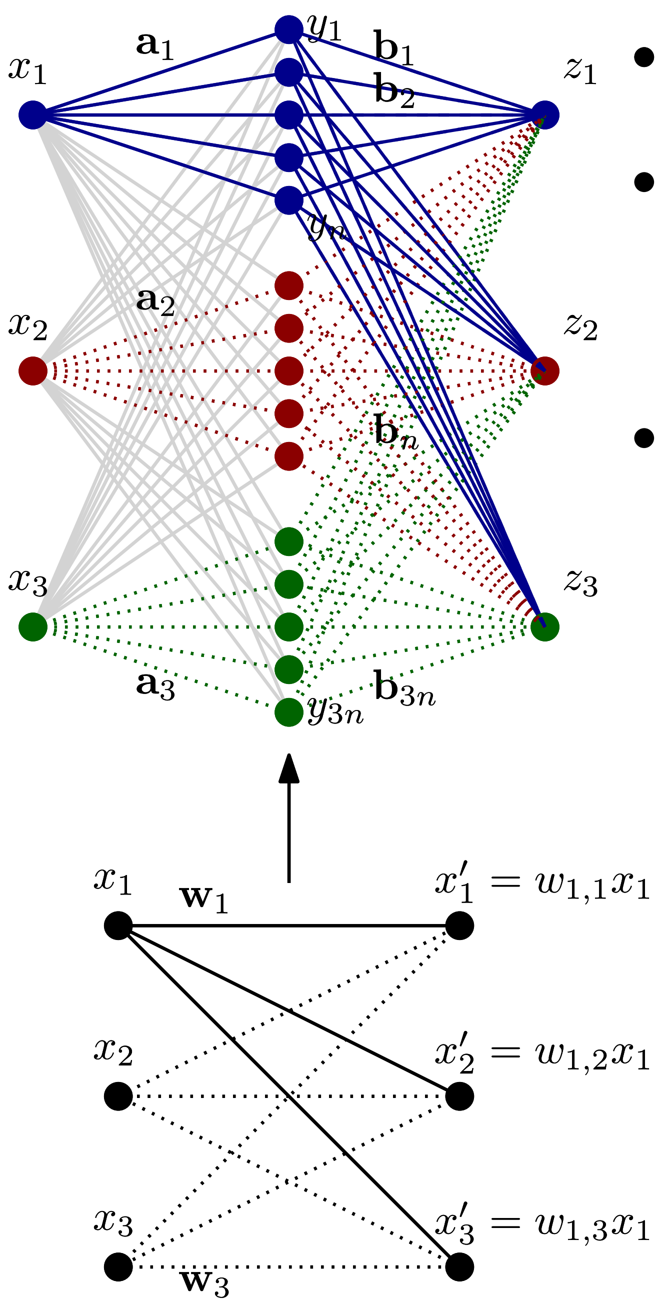
Apply MRSS for structured pruning



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

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

Apply MRSS for structured pruning

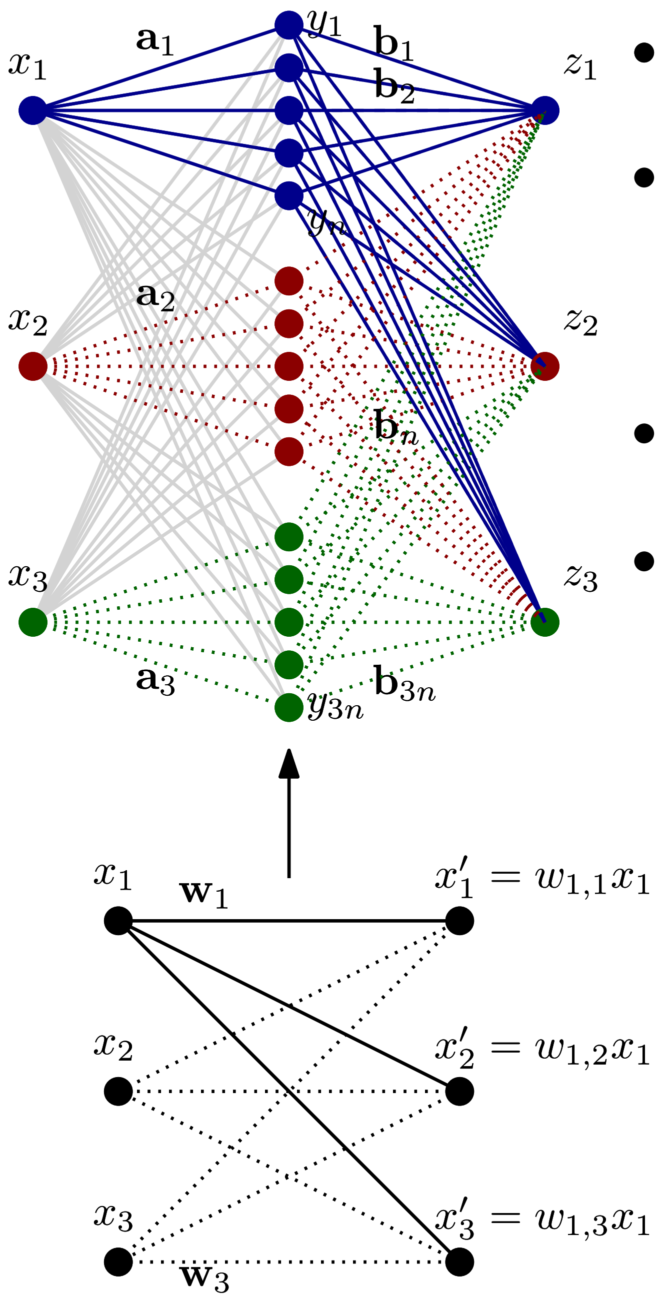


- $\mathbf{a}_i \sim \mathcal{N}(\mathbf{0}, I_n)$, $\mathbf{b}_i \sim \mathcal{N}(\mathbf{0}, I_d)$ (here, $d = 3$)
- For simplicity: **no ReLU**

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

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

Apply MRSS for structured pruning



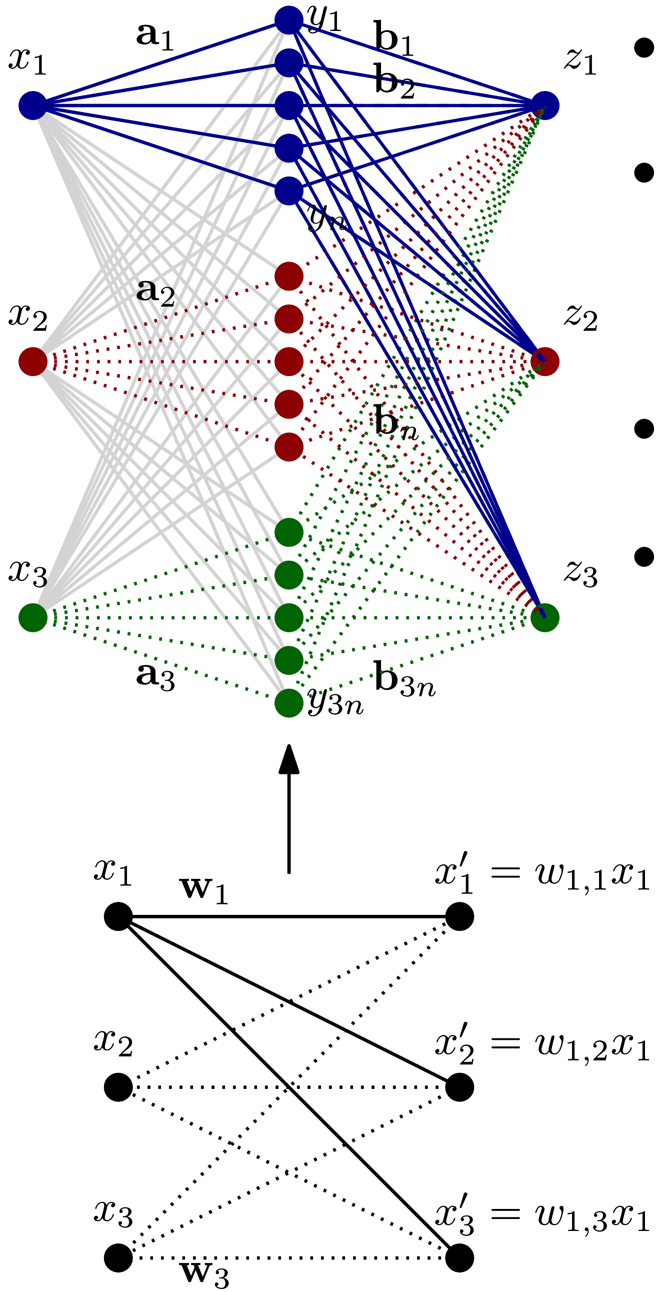
- $\mathbf{a}_i \sim \mathcal{N}(\mathbf{0}, I_n)$, $\mathbf{b}_i \sim \mathcal{N}(\mathbf{0}, I_d)$ (here, $d = 3$)
- For simplicity: **no ReLU**

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

- **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, $d = 3$)

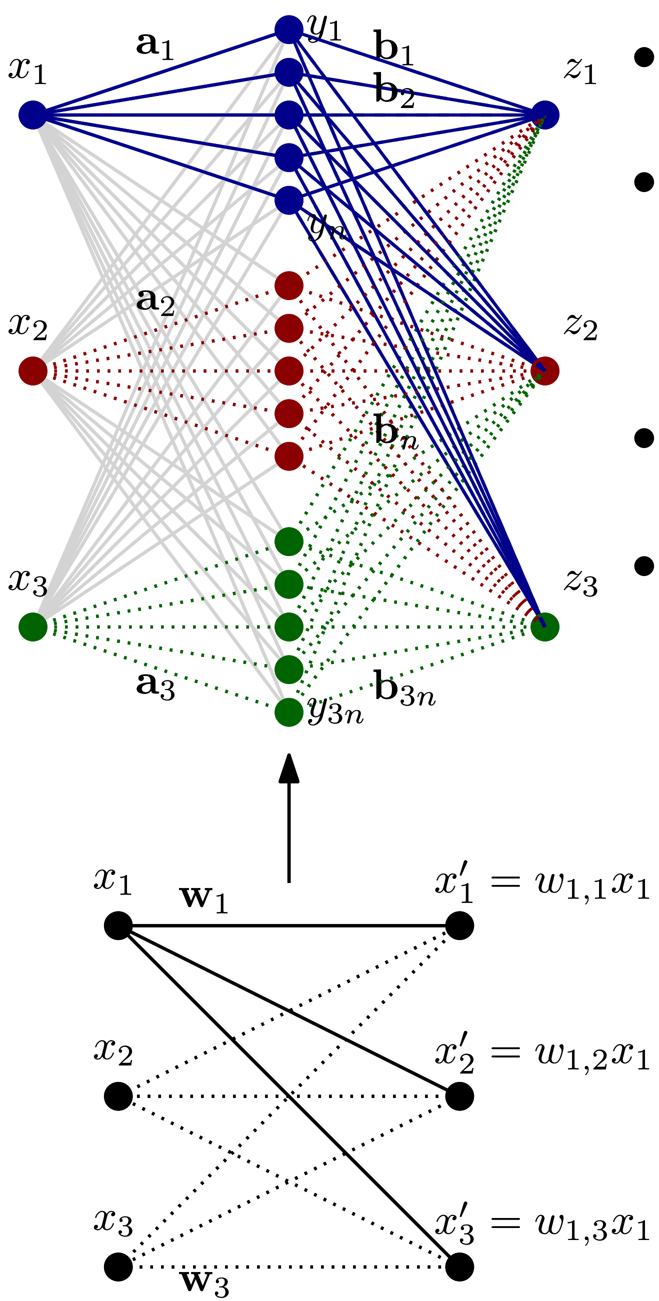
- For simplicity: **no ReLU**

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

- **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 $\mathcal{N}(\mathbf{0}, \sum_{i \in S} a_{1,i}^2 \cdot I_d)$

Apply MRSS for structured pruning

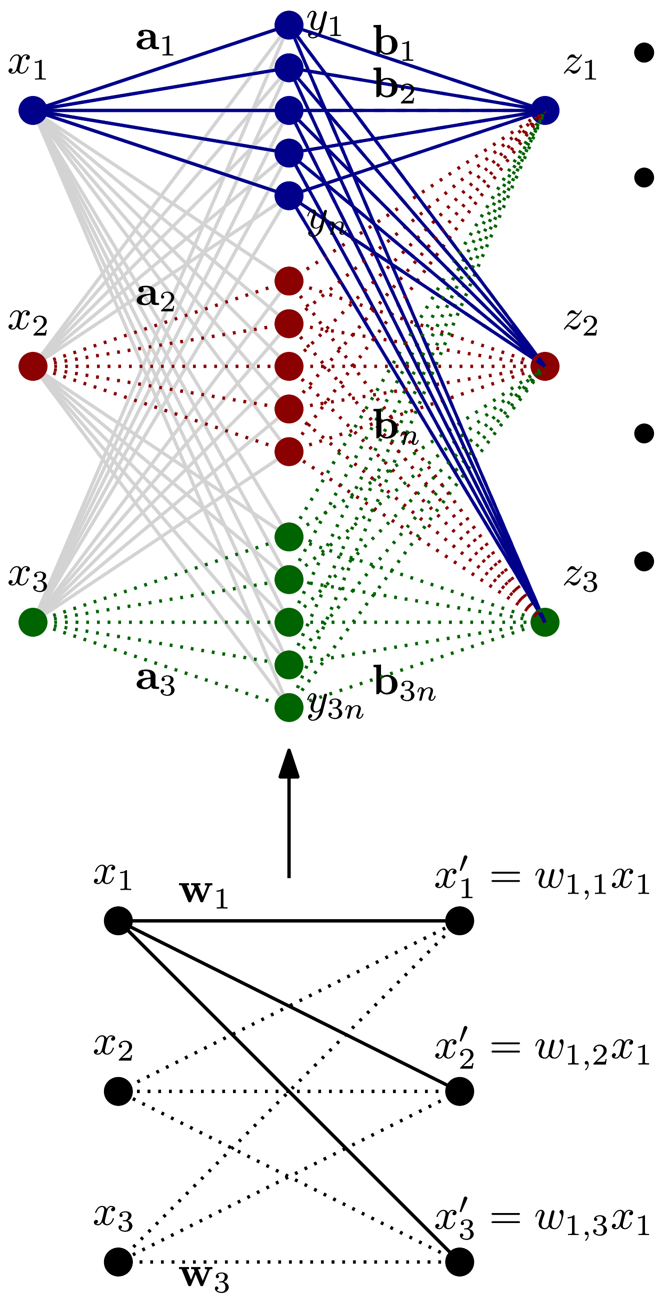


- $\mathbf{a}_i \sim \mathcal{N}(\mathbf{0}, I_n)$, $\mathbf{b}_i \sim \mathcal{N}(\mathbf{0}, I_d)$ (here, $d = 3$)
- For simplicity: **no ReLU**

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

- **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 $\mathcal{N}(\mathbf{0}, \sum_{i\in S} a_{1,i}^2 \cdot I_d)$
  - $\sum_{i \in S} a_{1,i}^2$ is a Chi-squared distribution: concentration inequalities!

Apply MRSS for structured pruning

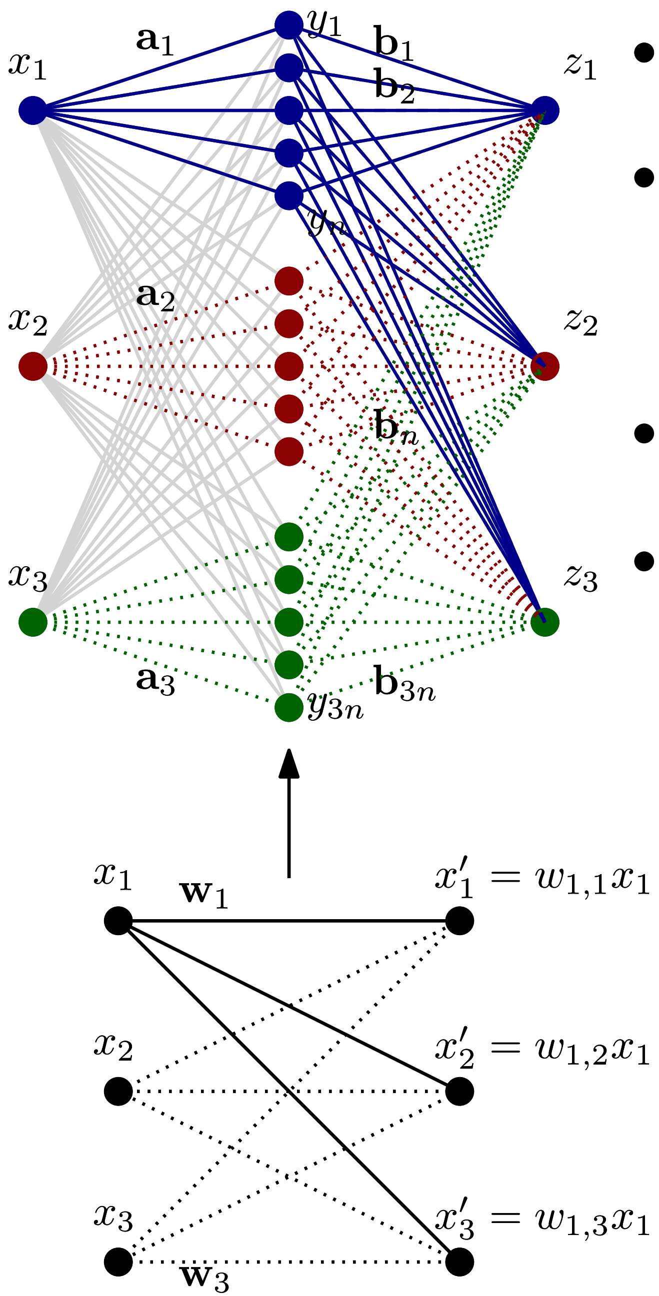


- $\mathbf{a}_i \sim \mathcal{N}(\mathbf{0}, I_n)$, $\mathbf{b}_i \sim \mathcal{N}(\mathbf{0}, I_d)$ (here, $d = 3$)
- For simplicity: **no ReLU**

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

- **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**

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

- **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 $\mathcal{N}(\mathbf{0}, \sum_{i\in S} a_{1,i}^2 \cdot I_d)$
  - $\sum_{i\in S} a_{1,i}^2$ is a Chi-squared distribution: concentration inequalities!
  - *things do not change too much*

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

Convolutional neural networks (CNNs)

- **Generality**: we actually prove the result in CNNs

Convolutional neural networks (CNNs)

- **Generality**: we actually prove the result in CNNs
- **Discrete convolution**:

$$\begin{bmatrix} 1 & 3 & -1 \\ 0 & -2 & 0 \\ 4 & -1 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 3 & 2 & 1 \\ 1 & 0 & 0 & -2 \\ 4 & -2 & 3 & 0 \\ 0 & 1 & 2 & -5 \end{bmatrix} = \begin{bmatrix} 19 & -3 \\ 2 & 3 \end{bmatrix}$$

Convolutional neural networks (CNNs)

- **Generality**: we actually prove the result in CNNs

- **Discrete convolution**:

$$\begin{array}{|c|c|c|}\hline 1 & 3 & -1 \\ \hline 0 & -2 & 0 \\ \hline 4 & -1 & -1 \\ \hline \end{array} * \begin{array}{|c|c|c|c|}\hline 1 & 3 & 2 & 1 \\ \hline 1 & 0 & 0 & -2 \\ \hline 4 & -2 & 3 & 0 \\ \hline 0 & 1 & 2 & -5 \\ \hline \end{array} = \begin{array}{|c|c|}\hline 19 & -3 \\ \hline 2 & 3 \\ \hline \end{array}$$

Convolutional neural networks (CNNs)

- **Generality**: we actually prove the result in CNNs

- **Discrete convolution**:

$$\begin{array}{|c|c|c|}\hline 1 & 3 & -1 \\ \hline 0 & -2 & 0 \\ \hline 4 & -1 & -1 \\ \hline \end{array} * \begin{array}{|c|c|c|c|}\hline 1 & 3 & 2 & 1 \\ \hline 1 & 0 & 0 & -2 \\ \hline 4 & -2 & 3 & 0 \\ \hline 0 & 1 & 2 & -5 \\ \hline \end{array} = \begin{array}{|c|c|}\hline 19 & -3 \\ \hline 2 & 3 \\ \hline \end{array}$$

Convolutional neural networks (CNNs)

- **Generality**: we actually prove the result in CNNs

- **Discrete convolution**:

$$\begin{array}{|c|c|c|} \hline 1 & 3 & -1 \\ \hline 0 & -2 & 0 \\ \hline 4 & -1 & -1 \\ \hline \end{array} * \begin{array}{|c|c|c|c|} \hline 1 & 3 & 2 & 1 \\ \hline 1 & 0 & 0 & -2 \\ \hline 4 & -2 & 3 & 0 \\ \hline 0 & 1 & 2 & -5 \\ \hline \end{array} = \begin{array}{|c|c|} \hline 19 & -3 \\ \hline 2 & 3 \\ \hline \end{array}$$

Convolutional neural networks (CNNs)

- **Generality**: we actually prove the result in CNNs
- **Discrete convolution**: keep dimension by padding with zeroes

$$\begin{array}{|c|c|c|}\hline 1 & 3 & -1 \\ \hline 0 & -2 & 0 \\ \hline 4 & -1 & -1 \\ \hline \end{array} * \begin{array}{|c|c|c|c|}\hline 1 & 3 & 2 & 1 \\ \hline 1 & 0 & 0 & -2 \\ \hline 4 & -2 & 3 & 0 \\ \hline 0 & 1 & 2 & -5 \\ \hline \end{array} = \begin{array}{|c|c|}\hline 19 & -3 \\ \hline 2 & 3 \\ \hline \end{array}$$

Convolutional neural networks (CNNs)

- **Generality**: we actually prove the result in CNNs
- **Discrete convolution**: keep dimension by padding with zeroes

$$\begin{array}{|c|c|c|}\hline 1 & 3 & -1 \\ \hline 0 & -2 & 0 \\ \hline 4 & -1 & -1 \\ \hline \end{array} * \begin{array}{|c|c|c|c|}\hline 1 & 3 & 2 & 1 \\ \hline 1 & 0 & 0 & -2 \\ \hline 4 & -2 & 3 & 0 \\ \hline 0 & 1 & 2 & -5 \\ \hline \end{array} = \begin{array}{|c|c|}\hline 19 & -3 \\ \hline 2 & 3 \\ \hline \end{array}$$

- **Generalization of matrix multiplication**: tensors can have higher dimensions

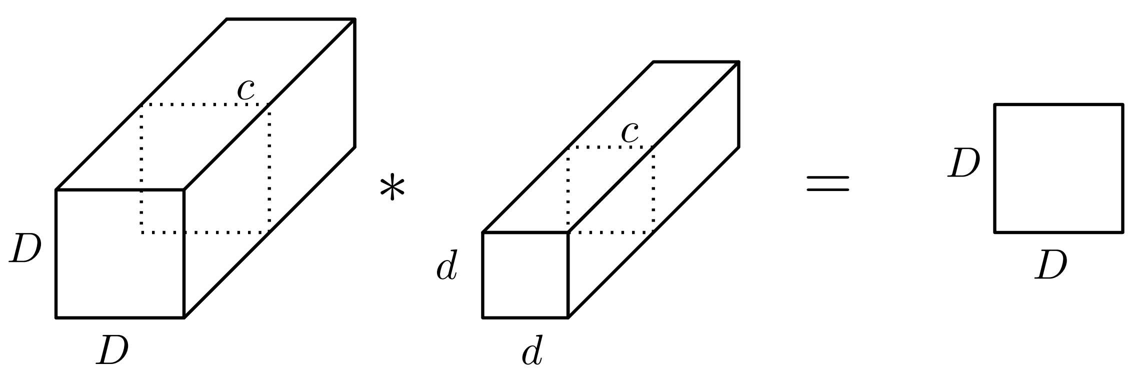
Convolutional neural networks (CNNs)

- **Generality**: we actually prove the result in CNNs

- **Discrete convolution**: keep dimension by padding with zeroes

$$\begin{array}{|c|c|c|}\hline 1 & 3 & -1 \\ \hline 0 & -2 & 0 \\ \hline 4 & -1 & -1 \\ \hline \end{array} * \begin{array}{|c|c|c|c|}\hline 1 & 3 & 2 & 1 \\ \hline 1 & 0 & 0 & -2 \\ \hline 4 & -2 & 3 & 0 \\ \hline 0 & 1 & 2 & -5 \\ \hline \end{array} = \begin{array}{|c|c|}\hline 19 & -3 \\ \hline 2 & 3 \\ \hline \end{array}$$

- **Generalization of matrix multiplication**: tensors can have higher dimensions

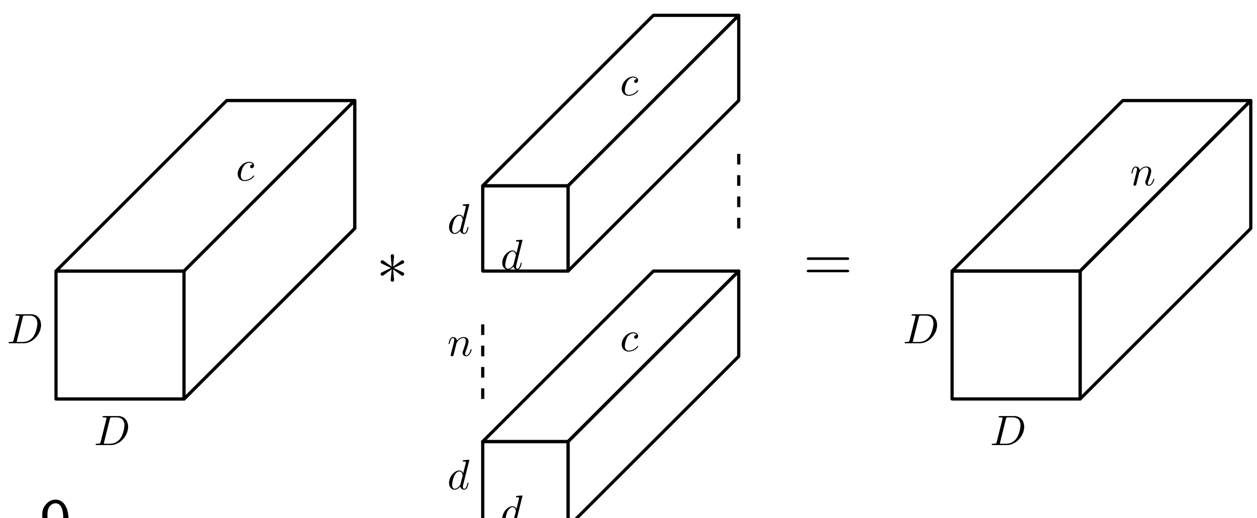


Convolutional neural networks (CNNs)

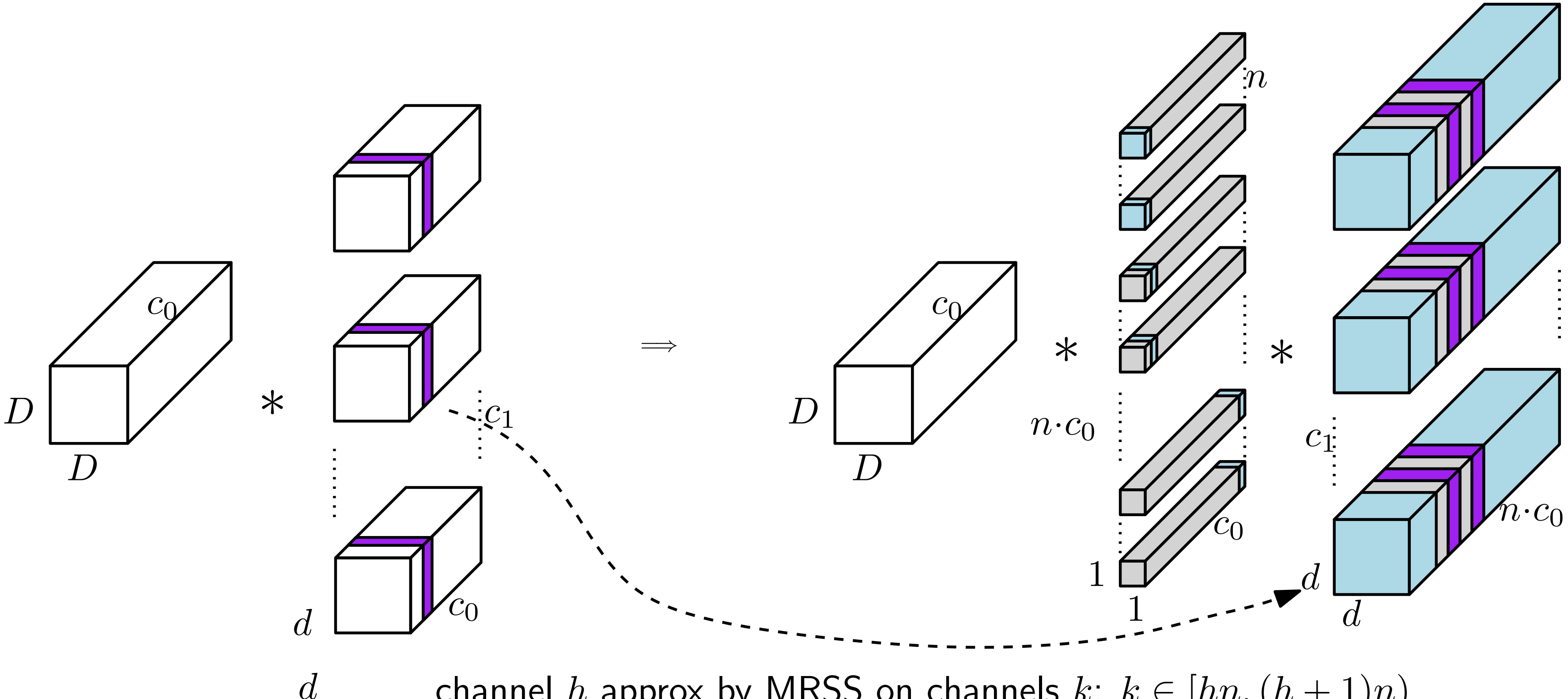
- **Generality**: we actually prove the result in CNNs
- **Discrete convolution**: keep dimension by padding with zeroes

$$\begin{array}{|c|c|c|}\hline 1 & 3 & -1 \\ \hline 0 & -2 & 0 \\ \hline 4 & -1 & -1 \\ \hline \end{array} * \begin{array}{|c|c|c|c|}\hline 1 & 3 & 2 & 1 \\ \hline 1 & 0 & 0 & -2 \\ \hline 4 & -2 & 3 & 0 \\ \hline 0 & 1 & 2 & -5 \\ \hline \end{array} = \begin{array}{|c|c|}\hline 19 & -3 \\ \hline 2 & 3 \\ \hline \end{array}$$

- **Generalization of matrix multiplication**: tensors can have higher dimensions

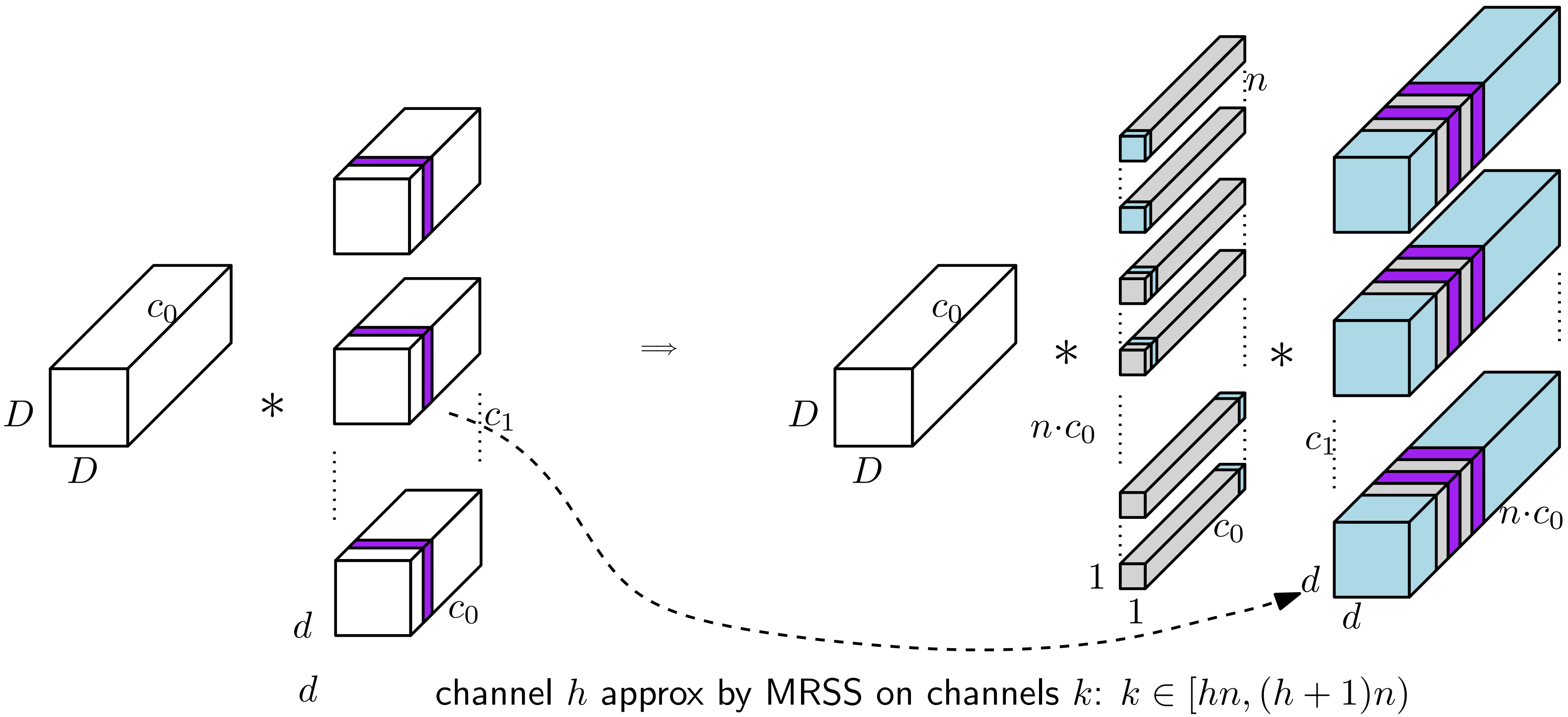


SLTH construction in CNNs



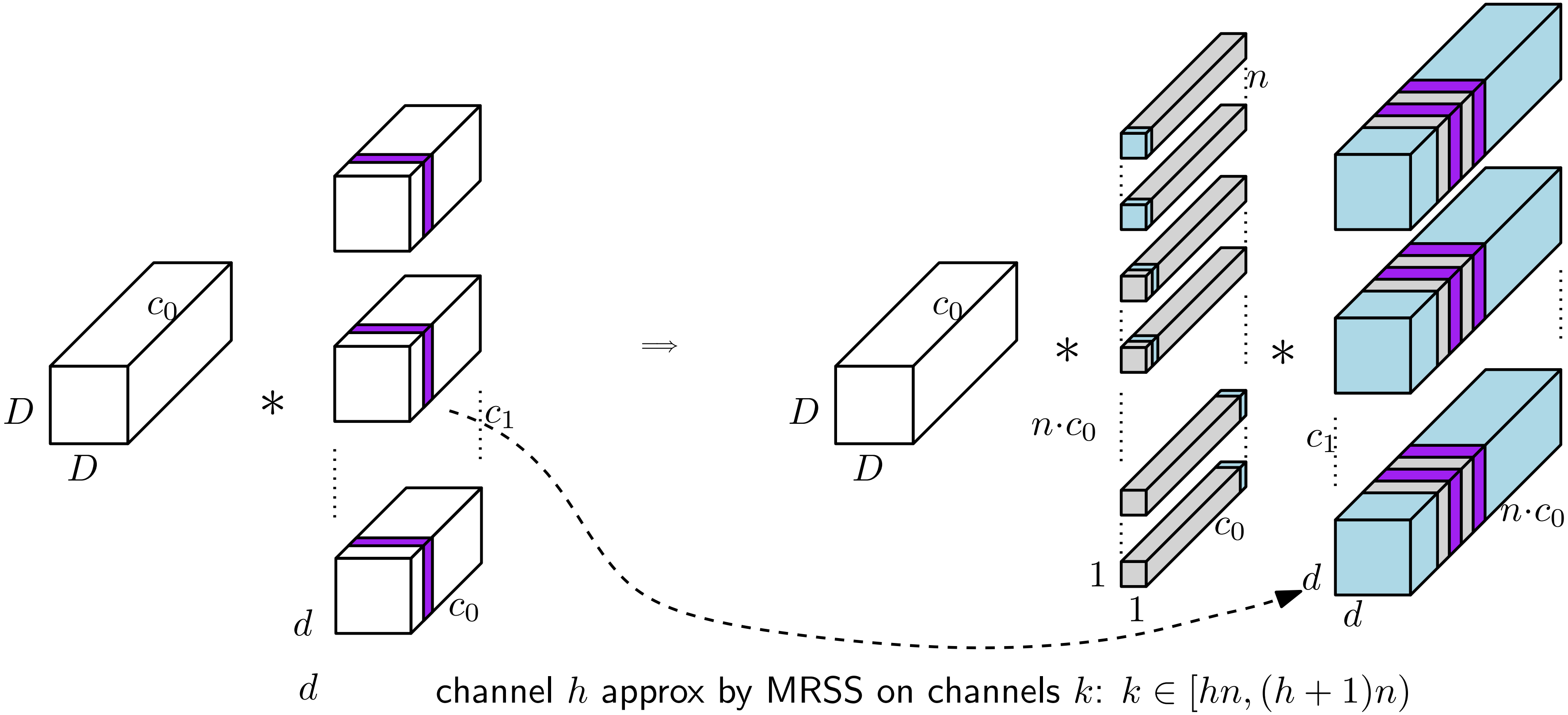
channel h approx by MRSS on channels k: $k \in [hn, (h+1)n)$

SLTH construction in CNNs



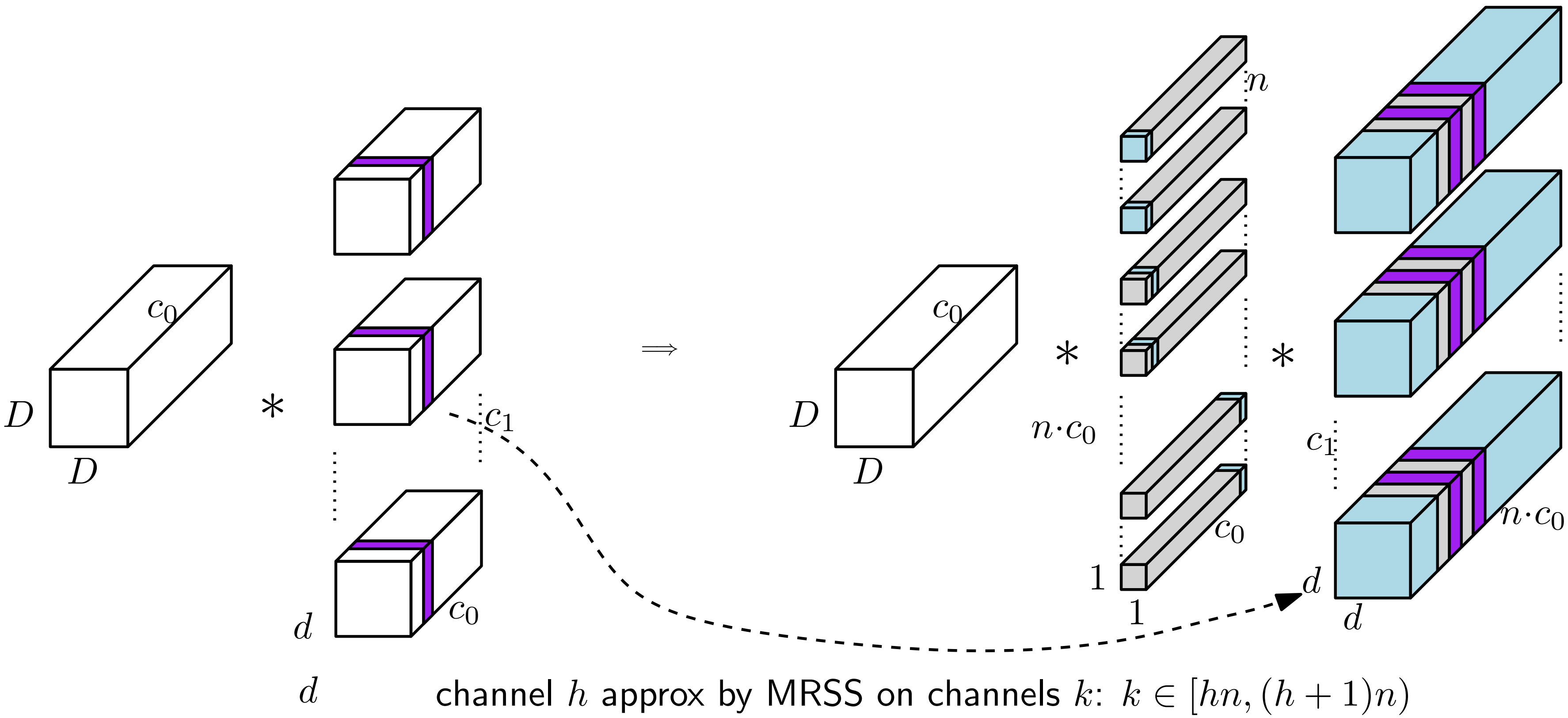
- Restrictions on the structure of the CNN

SLTH construction in CNNs



- Restrictions on the structure of the CNN
- Only ReLU activation function

SLTH construction in CNNs



channel h approx by MRSS on channels k: $k \in [hn, (h+1)n)$

- Restrictions on the structure of the CNN
- Only ReLU activation function
- $n \geq$ poly$(d) \cdot$ polylog$(d\ell/\varepsilon)$ is sufficient

Conclusions

- **Previously**: SLTH holds via unstructured pruning in dense networks, CNNs, etc., with logarithmic overhead

Conclusions

- **Previously**: SLTH holds via unstructured pruning in dense networks, CNNs, etc., with logarithmic overhead
- **Tool**: the one-dimensional RSS problem is heavily exploited

Conclusions

- **Previously**: SLTH holds via unstructured pruning in dense networks, CNNs, etc., with logarithmic overhead
- **Tool**: the one-dimensional RSS problem is heavily exploited
- **Issue**: it leads to exponential bounds when trying to achieve structured pruning

Conclusions

- **Previously**: SLTH holds via unstructured pruning in dense networks, CNNs, etc., with logarithmic overhead
- **Tool**: the one-dimensional RSS problem is heavily exploited
- **Issue**: it leads to exponential bounds when trying to achieve structured pruning
- **Solution**: multidimensional RSS

Conclusions

- **Previously**: SLTH holds via unstructured pruning in dense networks, CNNs, etc., with logarithmic overhead
- **Tool**: the one-dimensional RSS problem is heavily exploited
- **Issue**: it leads to exponential bounds when trying to achieve structured pruning
- **Solution**: multidimensional RSS
- **Modifications**: adaptation of MRSS to random vectors with dependent entries

Conclusions

- **Previously**: SLTH holds via unstructured pruning in dense networks, CNNs, etc., with logarithmic overhead
- **Tool**: the one-dimensional RSS problem is heavily exploited
- **Issue**: it leads to exponential bounds when trying to achieve structured pruning
- **Solution**: multidimensional RSS
- **Modifications**: adaptation of MRSS to random vectors with dependent entries
- **Our result**: SLTH holds in CNNs via structured pruning with polynomial overparameterization

Conclusions

- **Previously**: SLTH holds via unstructured pruning in dense networks, CNNs, etc., with logarithmic overhead
- **Tool**: the one-dimensional RSS problem is heavily exploited
- **Issue**: it leads to exponential bounds when trying to achieve structured pruning
- **Solution**: multidimensional RSS
- **Modifications**: adaptation of MRSS to random vectors with dependent entries
- **Our result**: SLTH holds in CNNs via structured pruning with polynomial overparameterization
- **Open (1)**: tightness of MRSS ($n \geq d \log 1/\varepsilon$?)

Conclusions

- **Previously**: SLTH holds via unstructured pruning in dense networks, CNNs, etc., with logarithmic overhead
- **Tool**: the one-dimensional RSS problem is heavily exploited
- **Issue**: it leads to exponential bounds when trying to achieve structured pruning
- **Solution**: multidimensional RSS
- **Modifications**: adaptation of MRSS to random vectors with dependent entries
- **Our result**: SLTH holds in CNNs via structured pruning with polynomial overparameterization
- **Open (1)**: tightness of MRSS ($n \geq d \log 1/\varepsilon$?)
- **Open (2)**: how to replace the union bound?

Conclusions

- **Previously**: SLTH holds via unstructured pruning in dense networks, CNNs, etc., with logarithmic overhead
- **Tool**: the one-dimensional RSS problem is heavily exploited
- **Issue**: it leads to exponential bounds when trying to achieve structured pruning
- **Solution**: multidimensional RSS
- **Modifications**: adaptation of MRSS to random vectors with dependent entries
- **Our result**: SLTH holds in CNNs via structured pruning with polynomial overparameterization
- **Open (1)**: tightness of MRSS ($n \geq d \log 1/\varepsilon$?)
- **Open (2)**: how to replace the union bound?
- **Open (3)**: generalization of CNN structure, activation function, etc.

Conclusions

- **Previously**: SLTH holds via unstructured pruning in dense networks, CNNs, etc., with logarithmic overhead
- **Tool**: the one-dimensional RSS problem is heavily exploited
- **Issue**: it leads to exponential bounds when trying to achieve structured pruning
- **Solution**: multidimensional RSS
- **Modifications**: adaptation of MRSS to random vectors with dependent entries
- **Our result**: SLTH holds in CNNs via structured pruning with polynomial overparameterization
- **Open (1)**: tightness of MRSS ($n \geq d \log 1/\varepsilon$?)
- **Open (2)**: how to replace the union bound?
- **Open (3)**: generalization of CNN structure, activation function, etc.

Thank you!

RSS proof overview

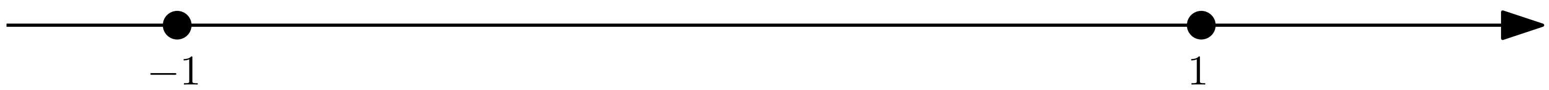
- [Lueker 1998; da Cunha et al. 2023]

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 $\varepsilon > 0$

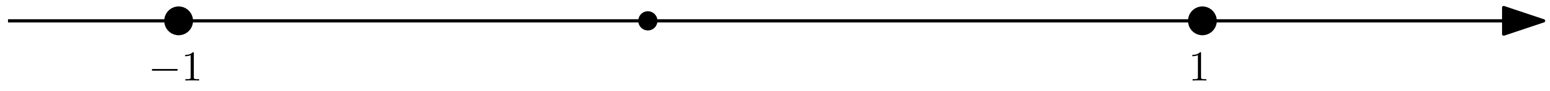


RSS proof overview

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

Specific instance of RSSP

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

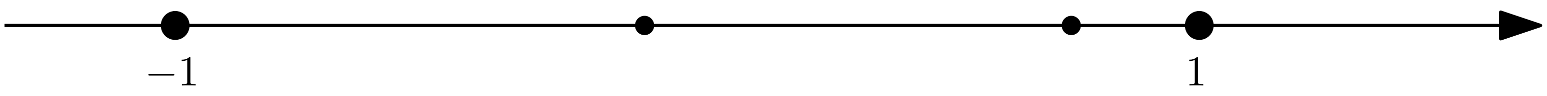


RSS proof overview

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

Specific instance of RSSP

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

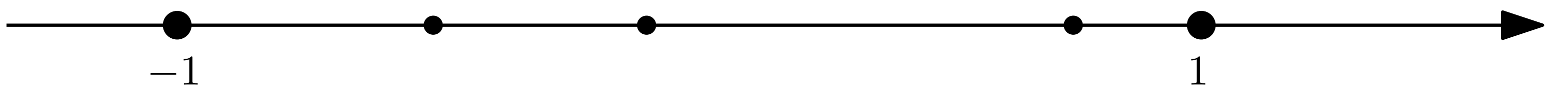


RSS proof overview

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

Specific instance of RSSP

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

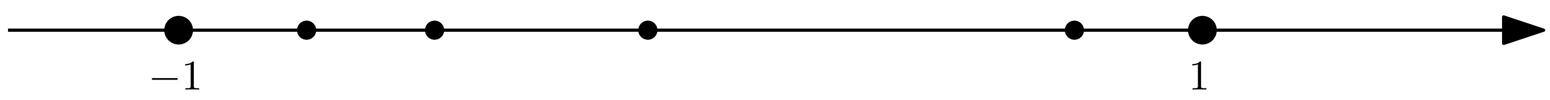


RSS proof overview

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

Specific instance of RSSP

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

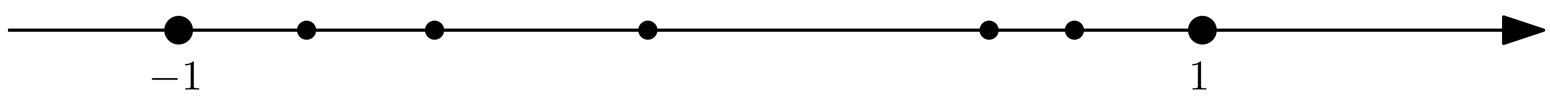


RSS proof overview

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

Specific instance of RSSP

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

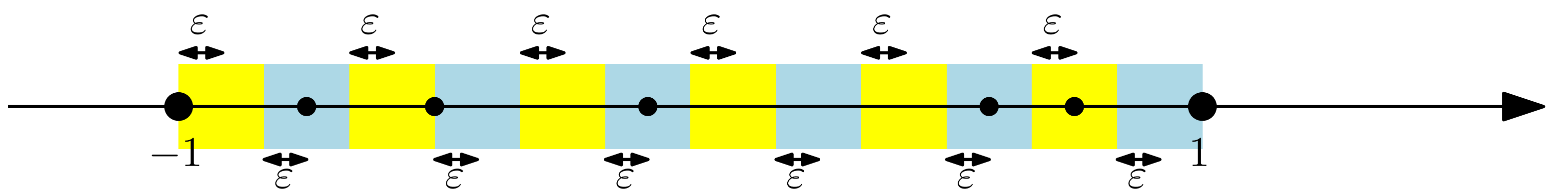


RSS proof overview

- [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

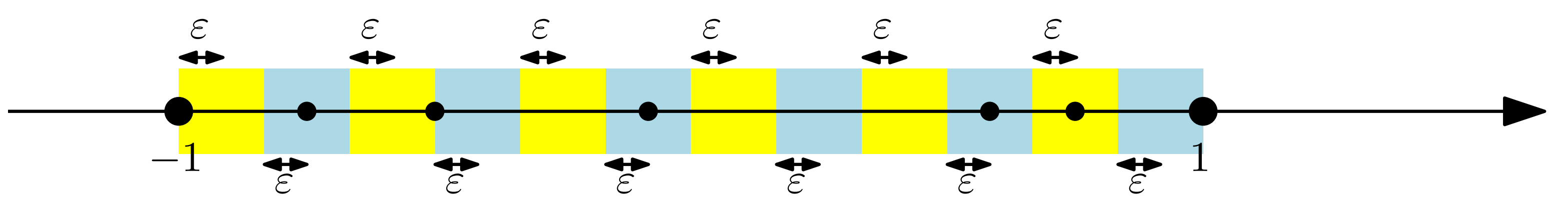


RSS proof overview

- [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



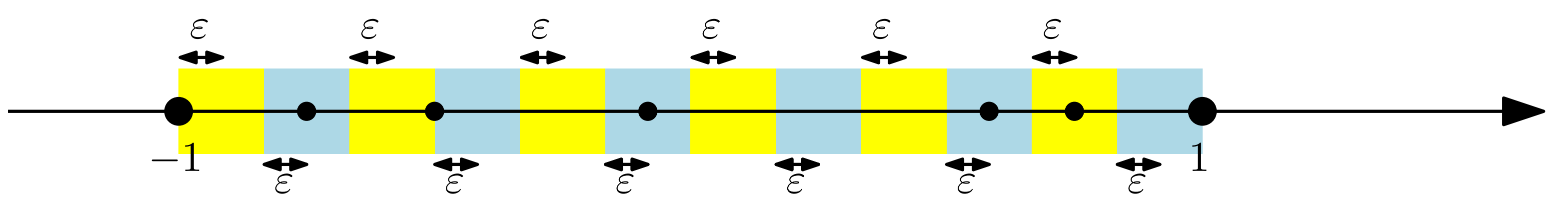
Consider $f_t(x) = \begin{cases} 1 & \text{if } x \in [-1,1] \text{ and } \exists S \subseteq [t] : \left| x - \sum_{i \in S} X_i \right| < 2\varepsilon \\ 0 & \text{otherwise} \end{cases}$

RSS proof overview

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

Specific instance of RSSP

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



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

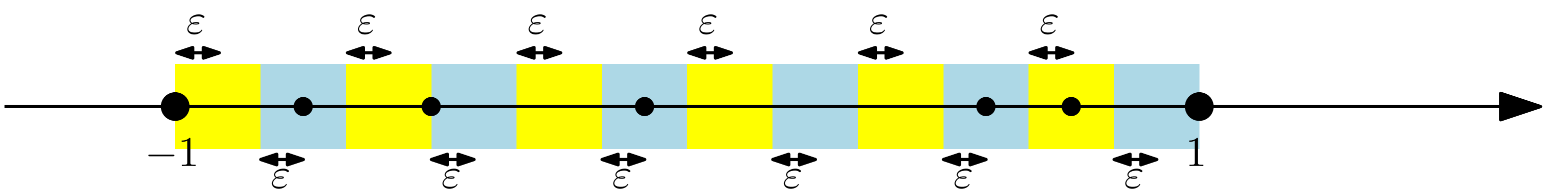
$v_t = \frac{1}{2} \int_{-1}^{1} f_t(x) \, \mathrm{d}x$ keeps track of the approximated volume

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

By restricting f_t, v_t becomes a sub-martingale