What Makes Neural Networks So Expressive, and What Could Make Them Smaller?

Thiago Serra

Bucknell University

Joint work with:

Abhinav Kumar

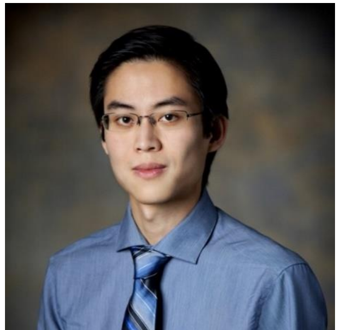
Michigan State University

Srikumar Ramalingam

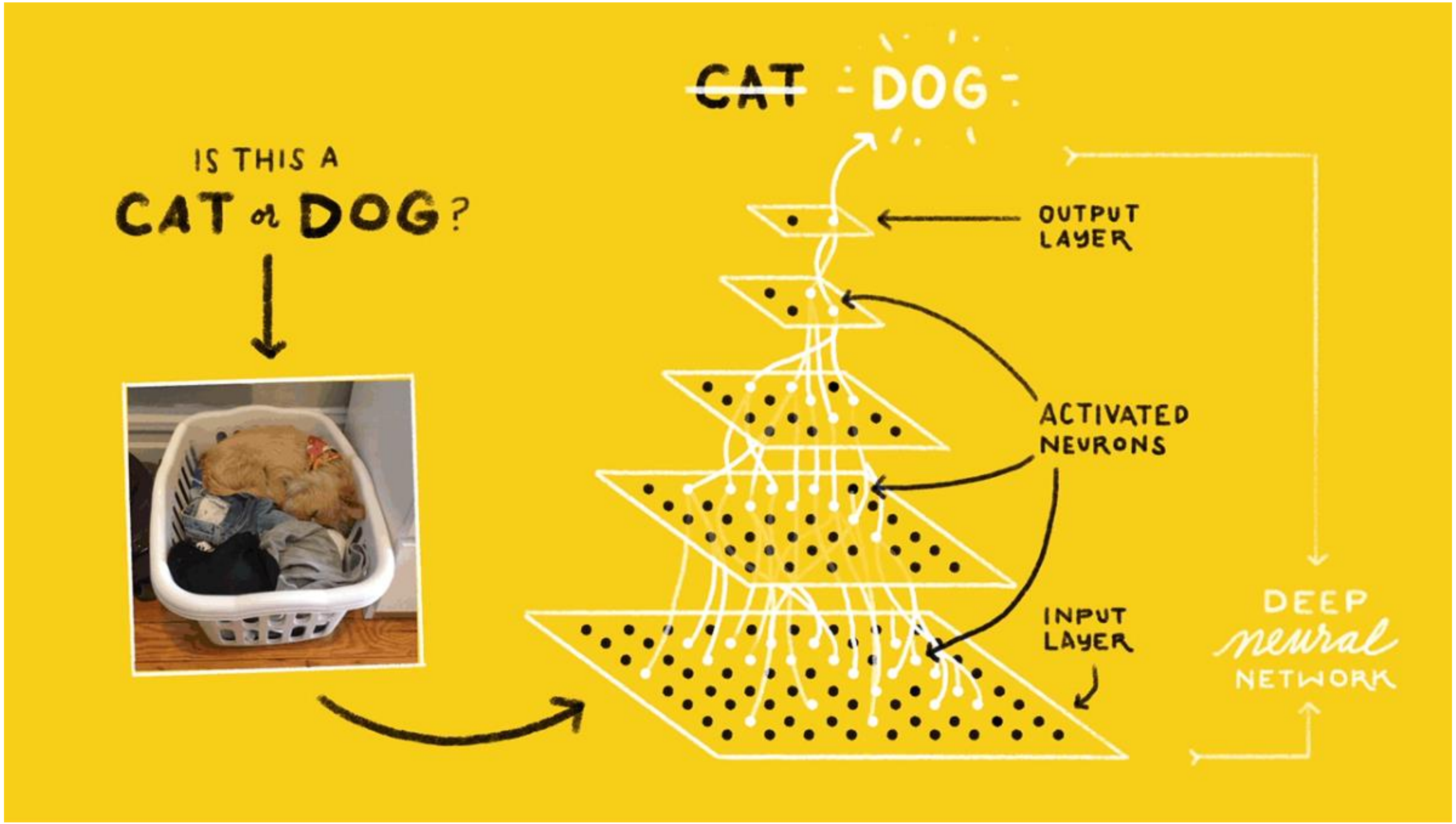
Google Research / The University of Utah

Christian Tjandraatmadja

Google Research



The Answer to Life, the Universe, and Everything



By Matteo Kofler
https://towardsdatascience.com/deep-learning-with-tensorflow-part-1-b19ce7803428

Or, Sometimes, Maybe Not…

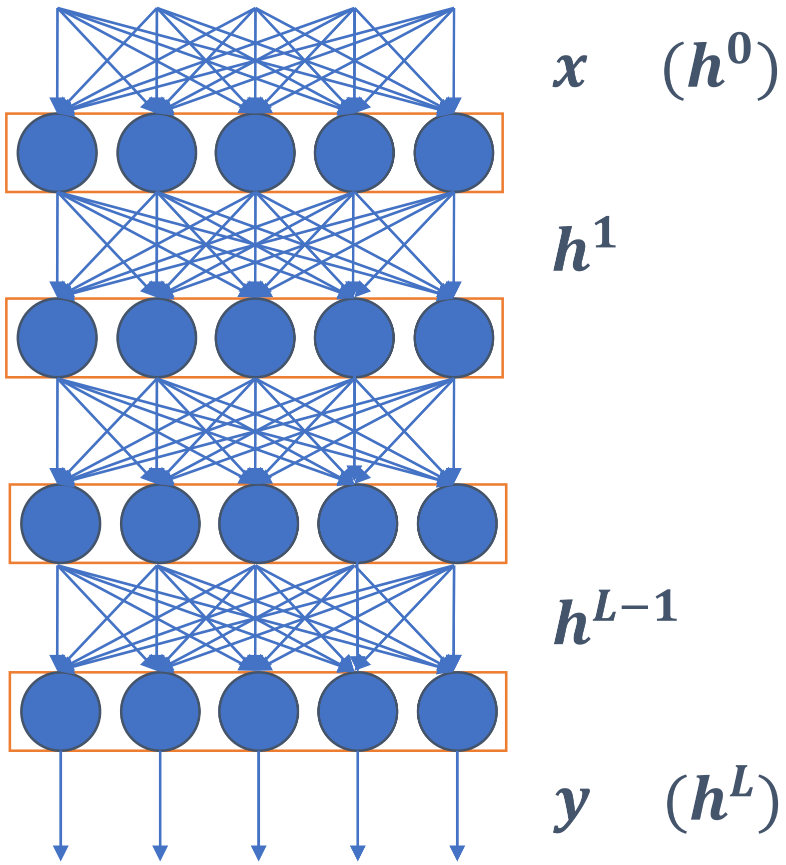
By Matt Hopkins
https://www.pedestrian.tv/tech/the-computer-from-hitchhikers-guide-to-the-galaxy-is-being-made-irl/

Notation

A feedforward neural network models a function from an input space **x** to an output space **y**

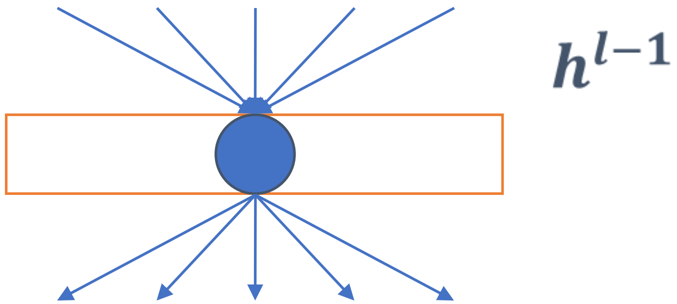
- Number of layers: $\boldsymbol{L}$
- Width of layer $\boldsymbol{l}$: $\boldsymbol{n^l}$
- Output of layer $\boldsymbol{l}$: $\boldsymbol{h^l} \in \mathbb{R}^{\boldsymbol{n^l}}$

- Input vector: $\boldsymbol{x}\ (\boldsymbol{h^0})$
- Input dimension: $\boldsymbol{n^0}$



The Scope of This Talk

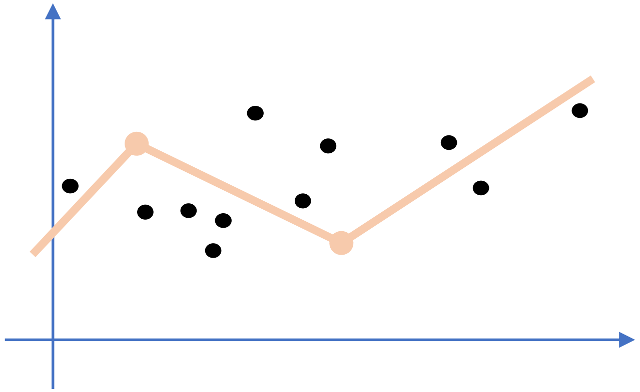
We study rectifier networks – those with only Rectified Linear Units (ReLUs)

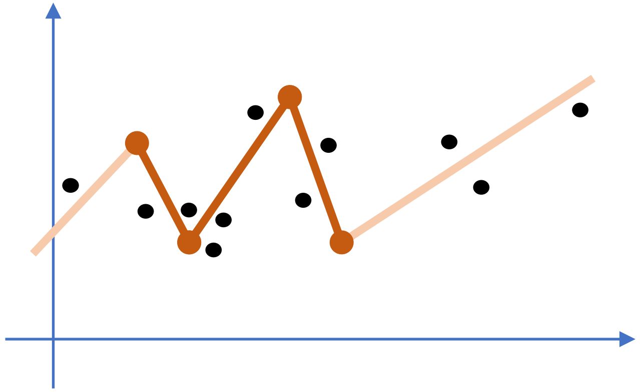


$$h_i^l = max\,\{\,\underbrace{0}_{\textbf{Inactive}},\,\underbrace{W_i^l h^{l-1} + b_i^l}_{\textbf{Active}}\,\}$$

Rectifier networks model piecewise linear functions

What Piecewise Linear Regression?





In other words, training a rectifier network is the same as performing a piecewise linear regression, but we do not know:

1. The **family of piecewise linear functions** of such regression

2. If a **smaller neural network** could define the same function

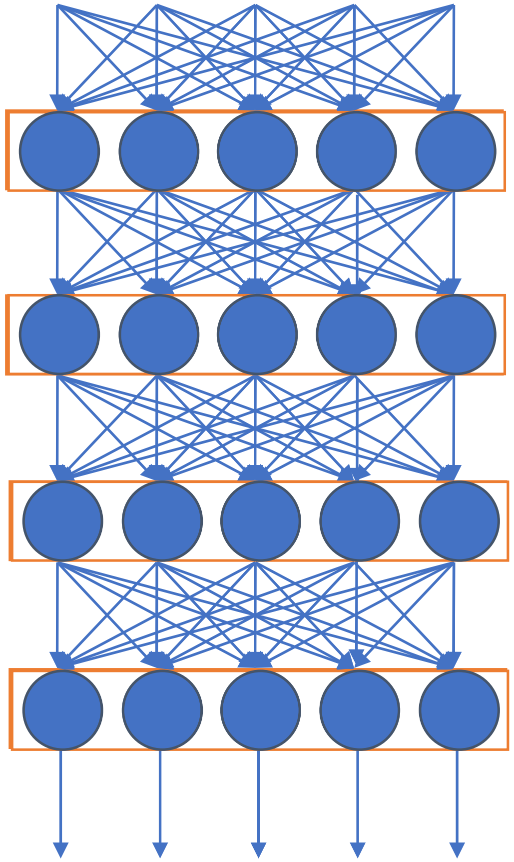
Each piece of the function domain is called a linear region

What makes neural networks so expressive?

https://www.kpbs.org/news/2019/aug/20/ironic-informal-and-expressive-new-rules-of/

Activation Patterns and Linear Regions

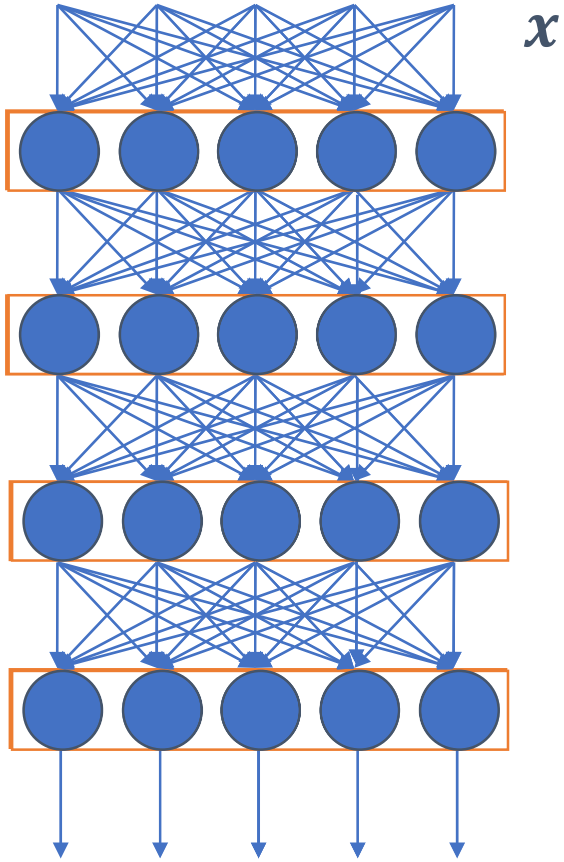
For ReLUs, we characterize these regions using the concept of activation patterns (Raghu et al., 2017; Montufar, 2017):



Activation Patterns and Linear Regions

For ReLUs, we characterize these regions using the concept of <u>activation patterns</u> (Raghu et al., 2017; Montufar, 2017):

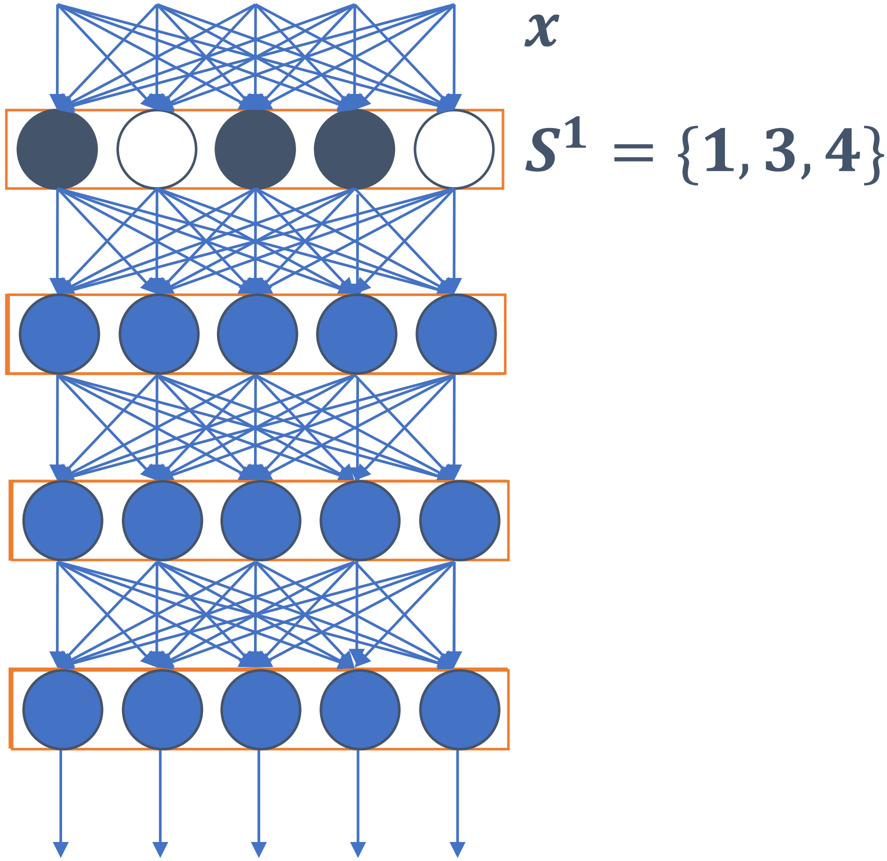
- For a given input $\boldsymbol{x}$



Activation Patterns and Linear Regions

For ReLUs, we characterize these regions using the concept of activation patterns (Raghu et al., 2017; Montufar, 2017):

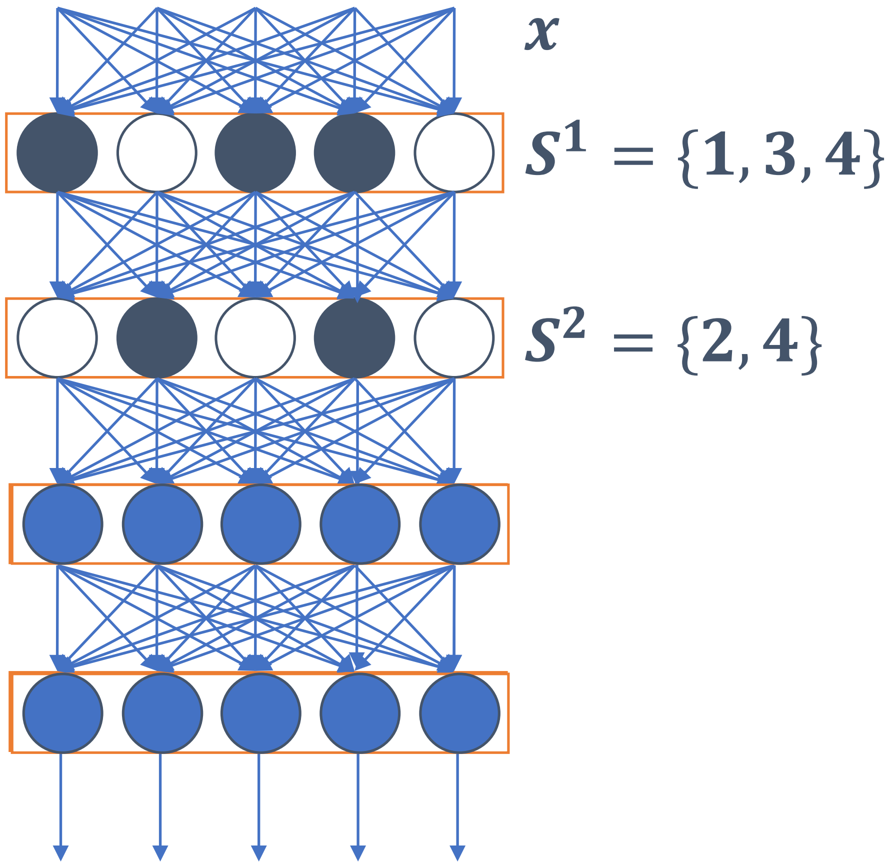
- For a given input $\boldsymbol{x}$
- There is an activation set $\boldsymbol{S^l} \subseteq \{\mathbf{1}, \mathbf{2}, \dots, \boldsymbol{n^l}\}$ for each layer l such that $\boldsymbol{i} \in \boldsymbol{S^l}$ iff $\mathbf{h_i^l} > \mathbf{0}$



Activation Patterns and Linear Regions

For ReLUs, we characterize these regions using the concept of activation patterns (Raghu et al., 2017; Montufar, 2017):

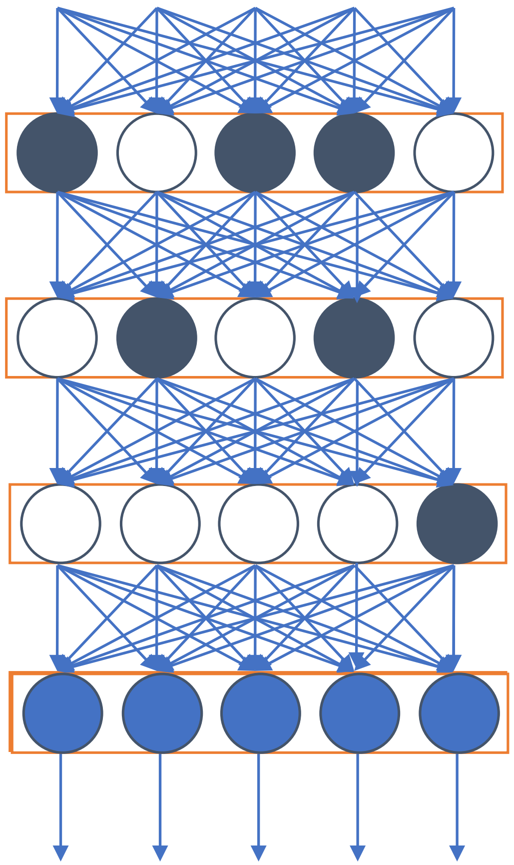
- For a given input $\boldsymbol{x}$
- There is an activation set $\boldsymbol{S^l} \subseteq \{\mathbf{1}, \mathbf{2}, \dots, \boldsymbol{n^l}\}$ for each layer l such that $\boldsymbol{i} \in \boldsymbol{S^l}$ iff $\mathbf{h_i^l} > \mathbf{0}$



Activation Patterns and Linear Regions

For ReLUs, we characterize these regions using the concept of activation patterns (Raghu et al., 2017; Montufar, 2017):

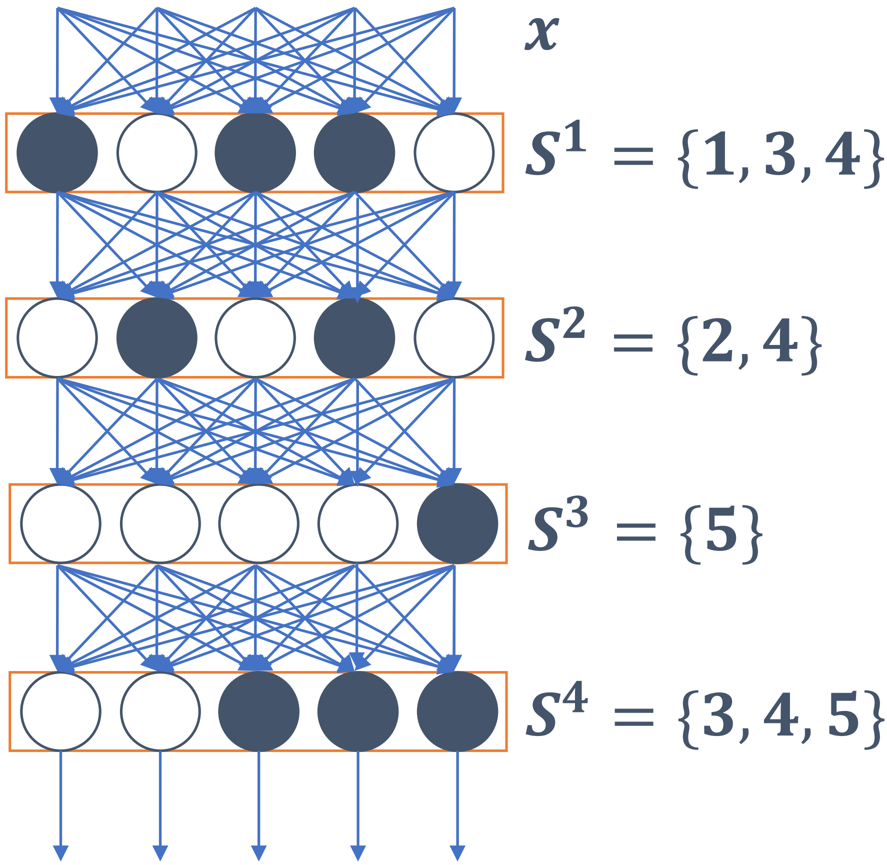
- For a given input $\boldsymbol{x}$
- There is an activation set $\boldsymbol{S^l} \subseteq \{\mathbf{1}, \mathbf{2}, \dots, \boldsymbol{n^l}\}$ for each layer l such that $\boldsymbol{i} \in \boldsymbol{S^l}$ iff $\mathbf{h_i^l} > \mathbf{0}$



Activation Patterns and Linear Regions

For ReLUs, we characterize these regions using the concept of activation patterns (Raghu et al., 2017; Montufar, 2017):

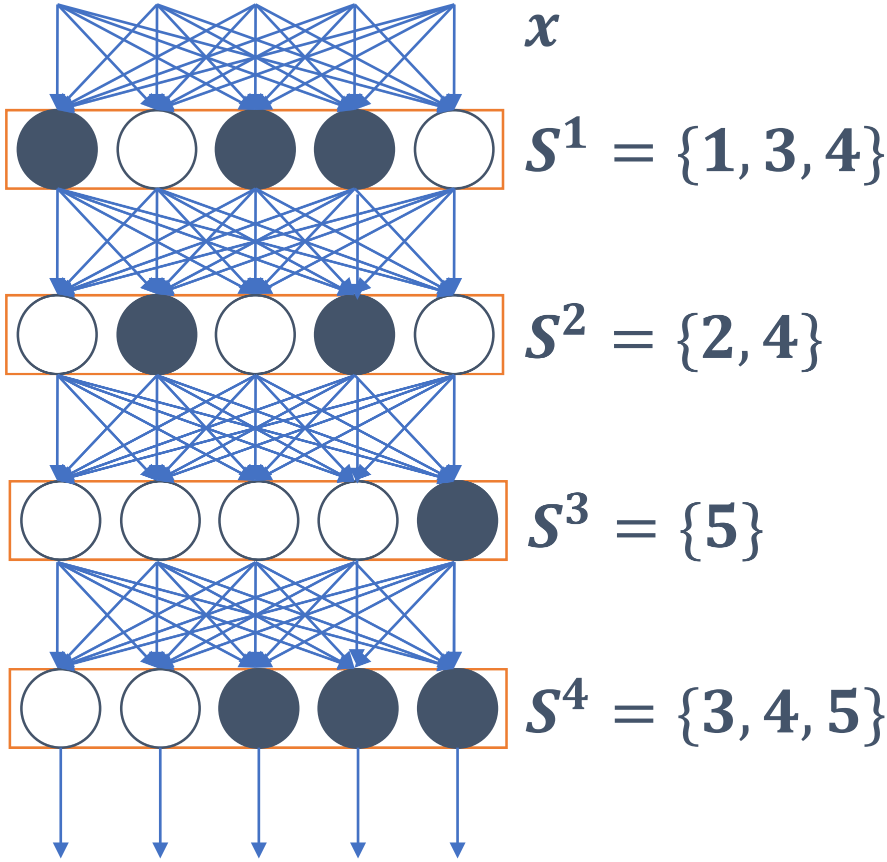
- For a given input $\boldsymbol{x}$
- There is an activation set $\boldsymbol{S}^{l} \subseteq \{\mathbf{1}, \mathbf{2}, \dots, \boldsymbol{n}^{l}\}$ for each layer l such that $\boldsymbol{i} \in \boldsymbol{S}^{l}$ iff $\mathbf{h}_{\mathbf{i}}^{\mathbf{l}} > \mathbf{0}$



Activation Patterns and Linear Regions

For ReLUs, we characterize these regions using the concept of activation patterns (Raghu et al., 2017; Montufar, 2017):

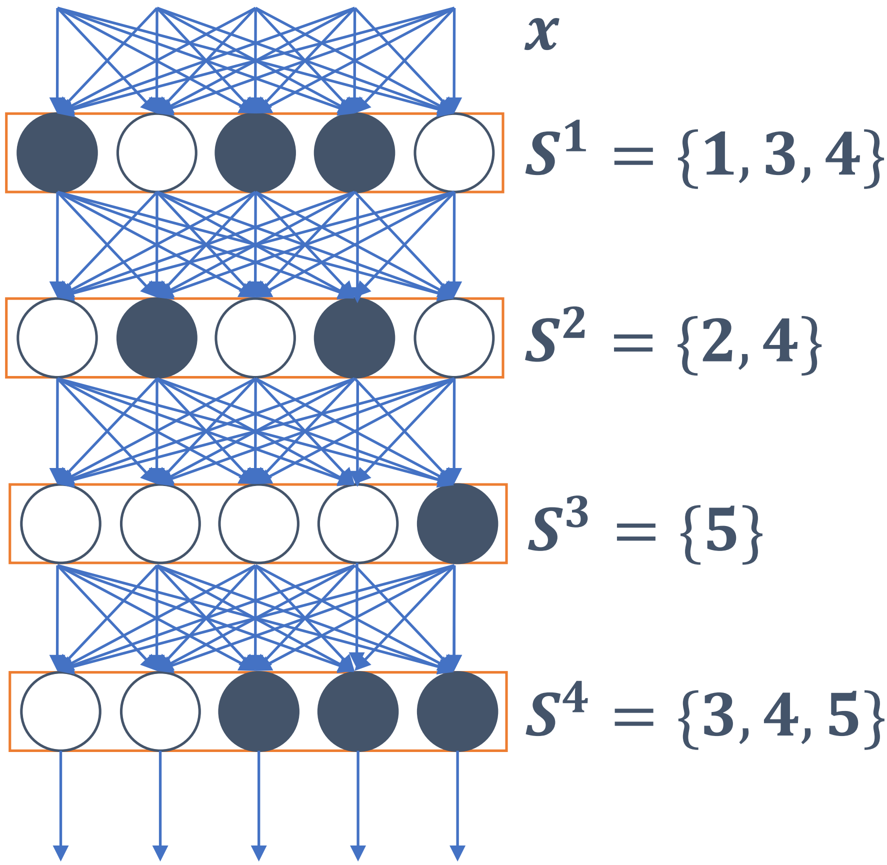
- For a given input $\boldsymbol{x}$
- There is an activation set $\boldsymbol{S}^l \subseteq \{1, 2, \ldots, n^l\}$ for each layer l such that $\boldsymbol{i} \in \boldsymbol{S}^l$ iff $\mathbf{h}_\mathbf{i}^\mathbf{l} > \mathbf{0}$
- The activation pattern of $\boldsymbol{x}$ is $\boldsymbol{\mathcal{S}} = (\boldsymbol{S}^1, \ldots, \boldsymbol{S}^l)$



Activation Patterns and Linear Regions

For ReLUs, we characterize these regions using the concept of activation patterns (Raghu et al., 2017; Montufar, 2017):

- For a given input $\boldsymbol{x}$
- There is an activation set $\boldsymbol{S^l} \subseteq \{\mathbf{1}, \mathbf{2}, \dots, \boldsymbol{n^l}\}$ for each layer l such that $\boldsymbol{i} \in \boldsymbol{S^l}$ iff $\mathbf{h_i^l} > \mathbf{0}$
- The activation pattern of $\boldsymbol{x}$ is $\boldsymbol{\mathcal{S}} = (\boldsymbol{S^1}, \dots, \boldsymbol{S^l})$



A linear region is the set of all points with a same activation pattern

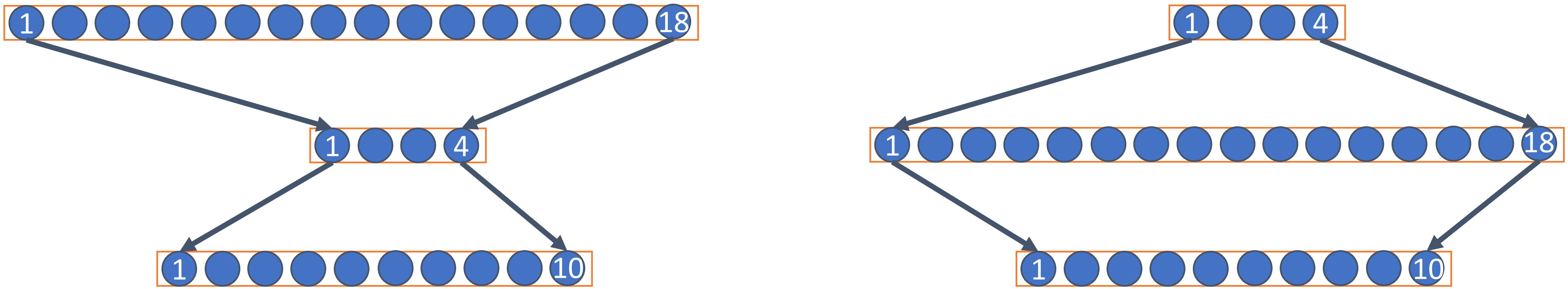
The number of activation patterns bounds the number of linear regions (Montufar et al., 2014):

$$\mathbf{2}^{\boldsymbol{n_1} + \dots + \boldsymbol{n_L}}$$

Bounding

Negatives through bounds are important

- We can find limits to what functions can be <u>approximated</u>
- We can compare <u>different configurations</u>

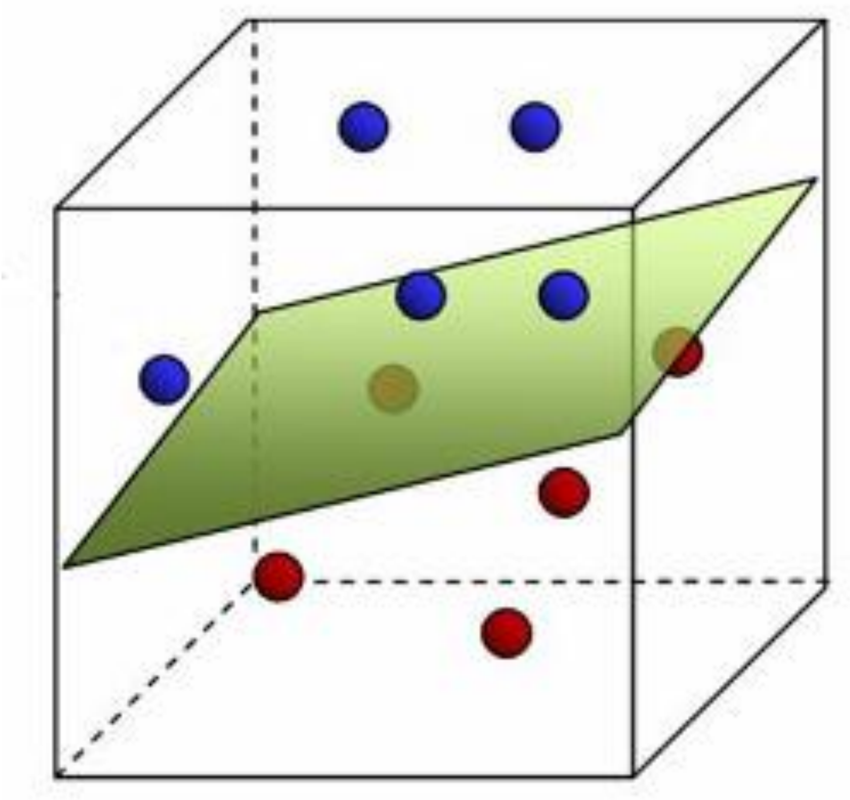


The Geometry of Linear Regions

For each unit *i* in layer *l*, $\boldsymbol{W_i^l}$ and $\boldsymbol{b_i^l}$ define an <u>activation hyperplane</u> on $\boldsymbol{h^{l-1}}$:

$$\boldsymbol{W_i^l h^{l-1} + b_i^l = 0}$$

Active points:
$\boldsymbol{W_i^l h^{l-1} + b_i^l > 0}$



Inactive points:
$\boldsymbol{W_i^l h^{l-1} + b_i^l \leq 0}$

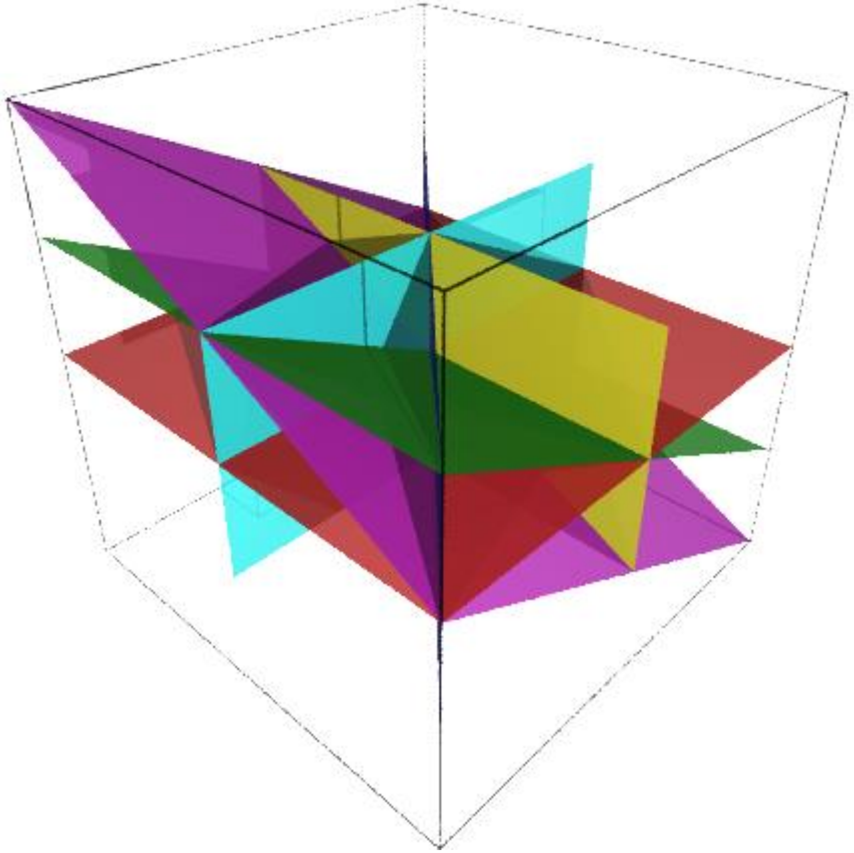
By Arvind Narayanan
https://33bits.wordpress.com/2010/12/20/the-unsung-success-of-can-spam/

The Geometry of Linear Regions

When put together, the activation hyperplanes of units in a given layer break the input space of the layer with a hyperplane arrangement

The number of regions depends on:

- **Number of hyperplanes**
- **Dimension of the space**



Bounding Shallow Networks

The number of regions of a shallow network is at most

$$\sum_{i=0}^{n_0} \binom{n_1}{i}$$

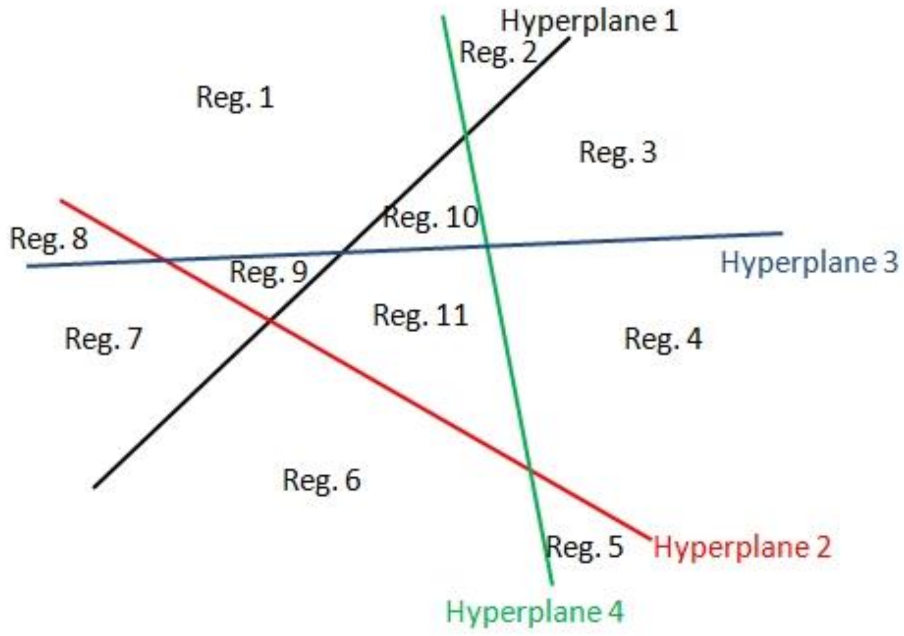
Bounding Shallow Networks

The number of regions of a shallow network is at most

$$\sum_{i=0}^{n_0} \binom{n_1}{i}$$

With 4 hyperplanes in 2 dimensions, we have:

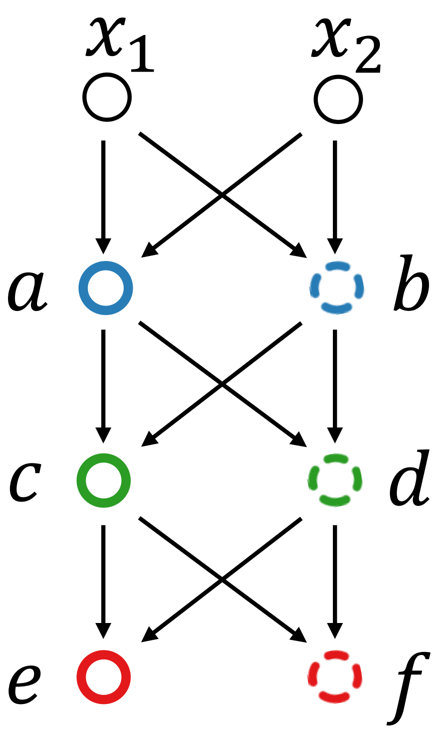
$$\binom{4}{0} + \binom{4}{1} + \binom{4}{2} = 1 + 4 + 6 = 11$$

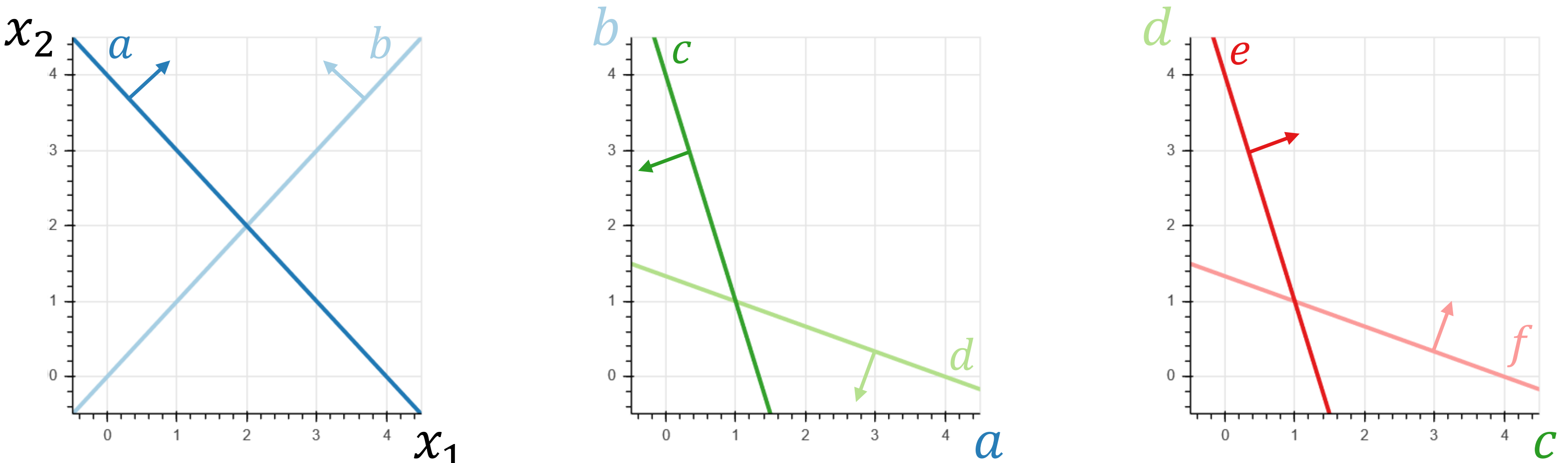


We can always reach that bound

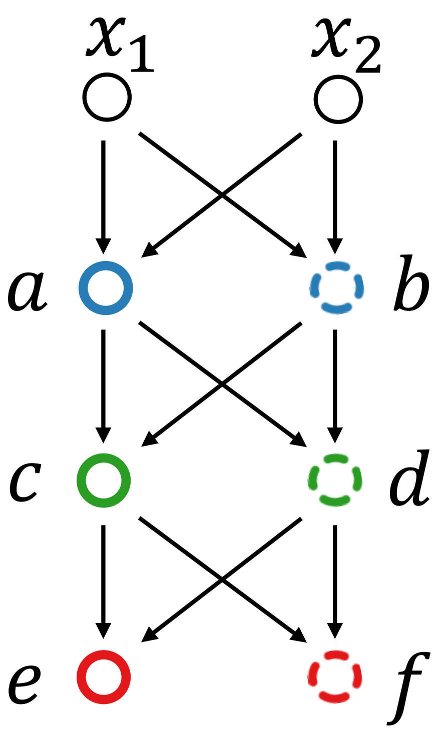
From https://onionesquereality.wordpress.com/2012/11/23/maximum-number-of-regions-in-arrangement-of-hyperplanes/

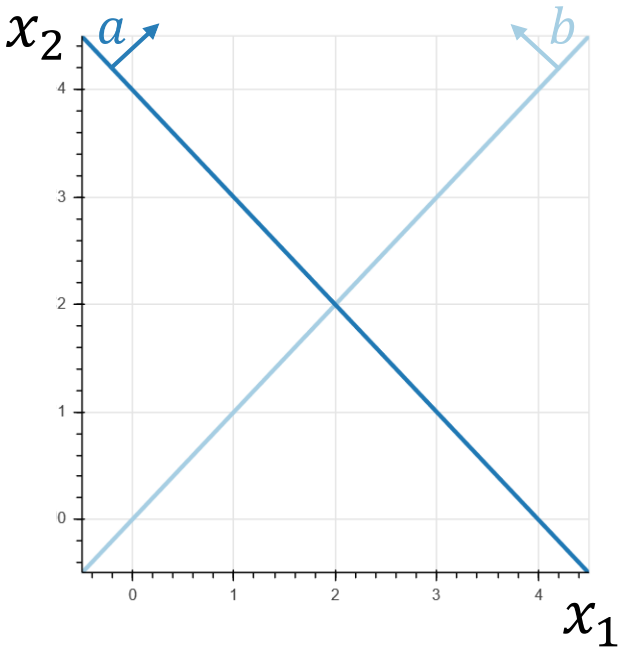
The Geometry of Linear Regions

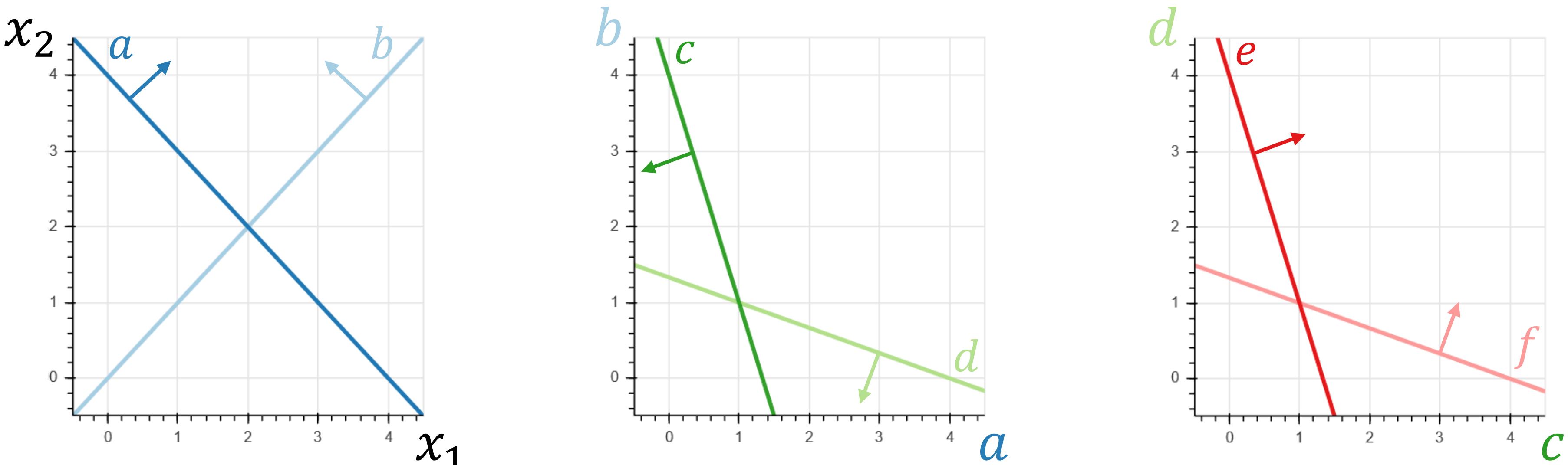




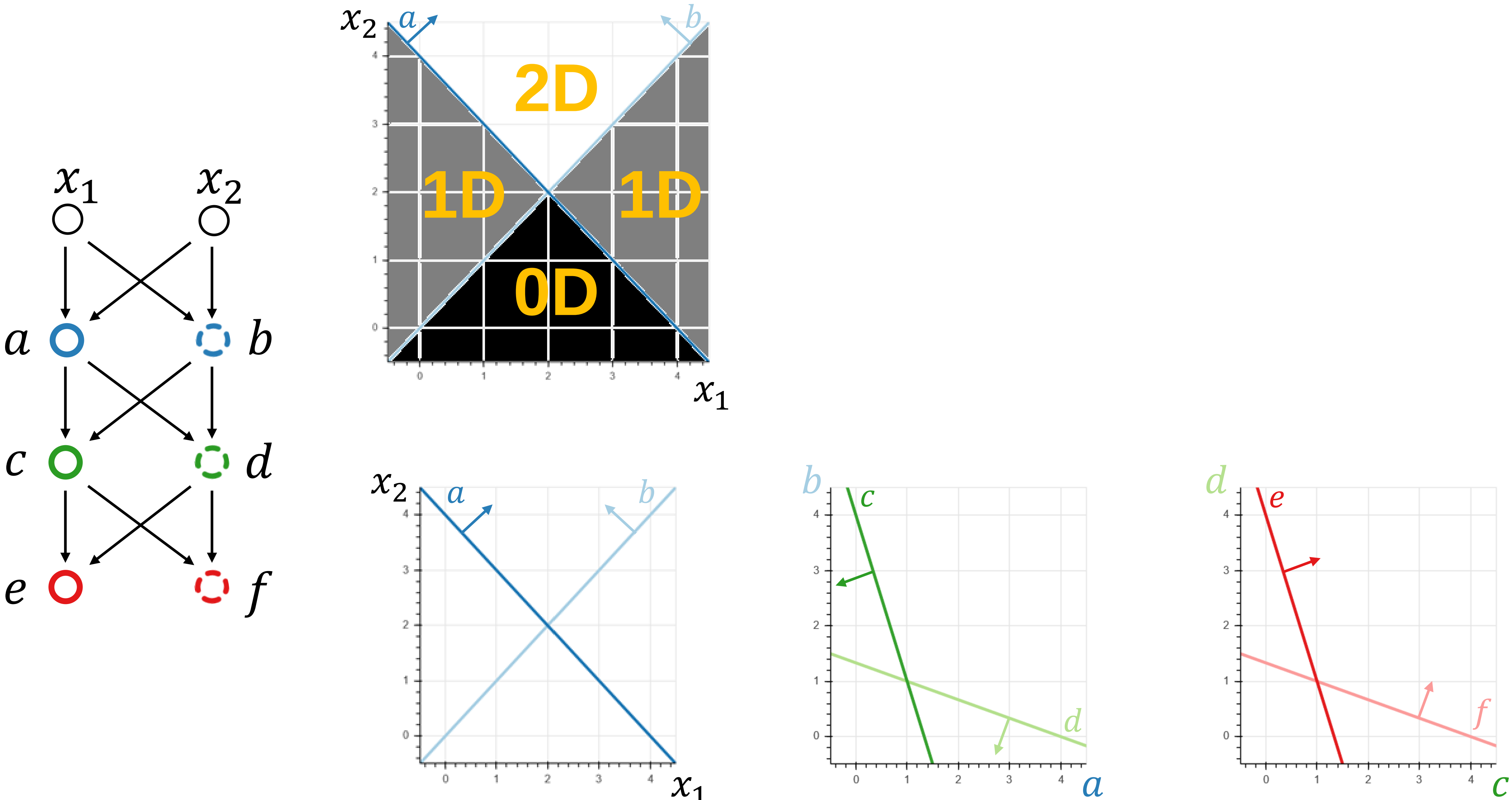
The Geometry of Linear Regions



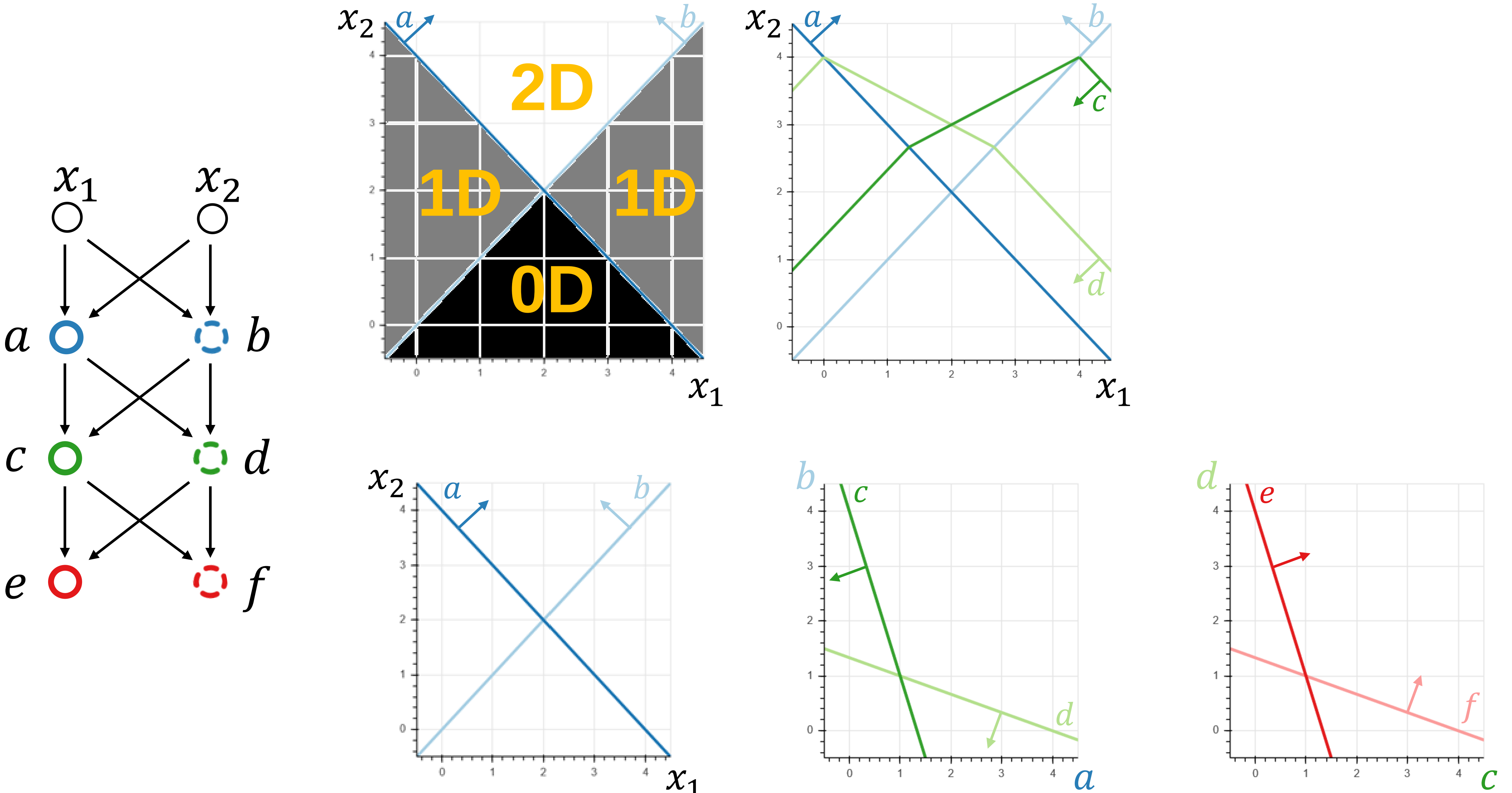




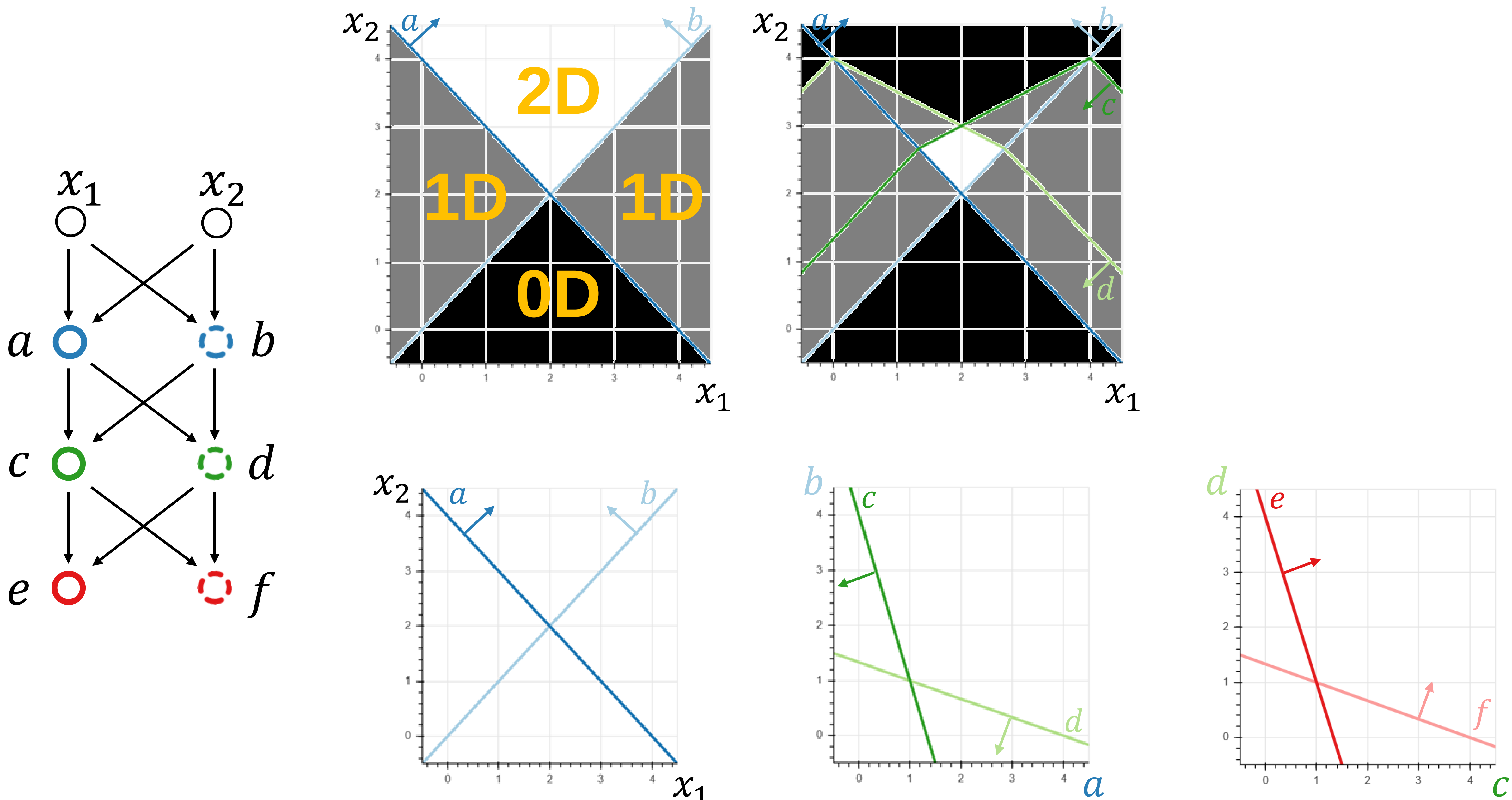
The Geometry of Linear Regions



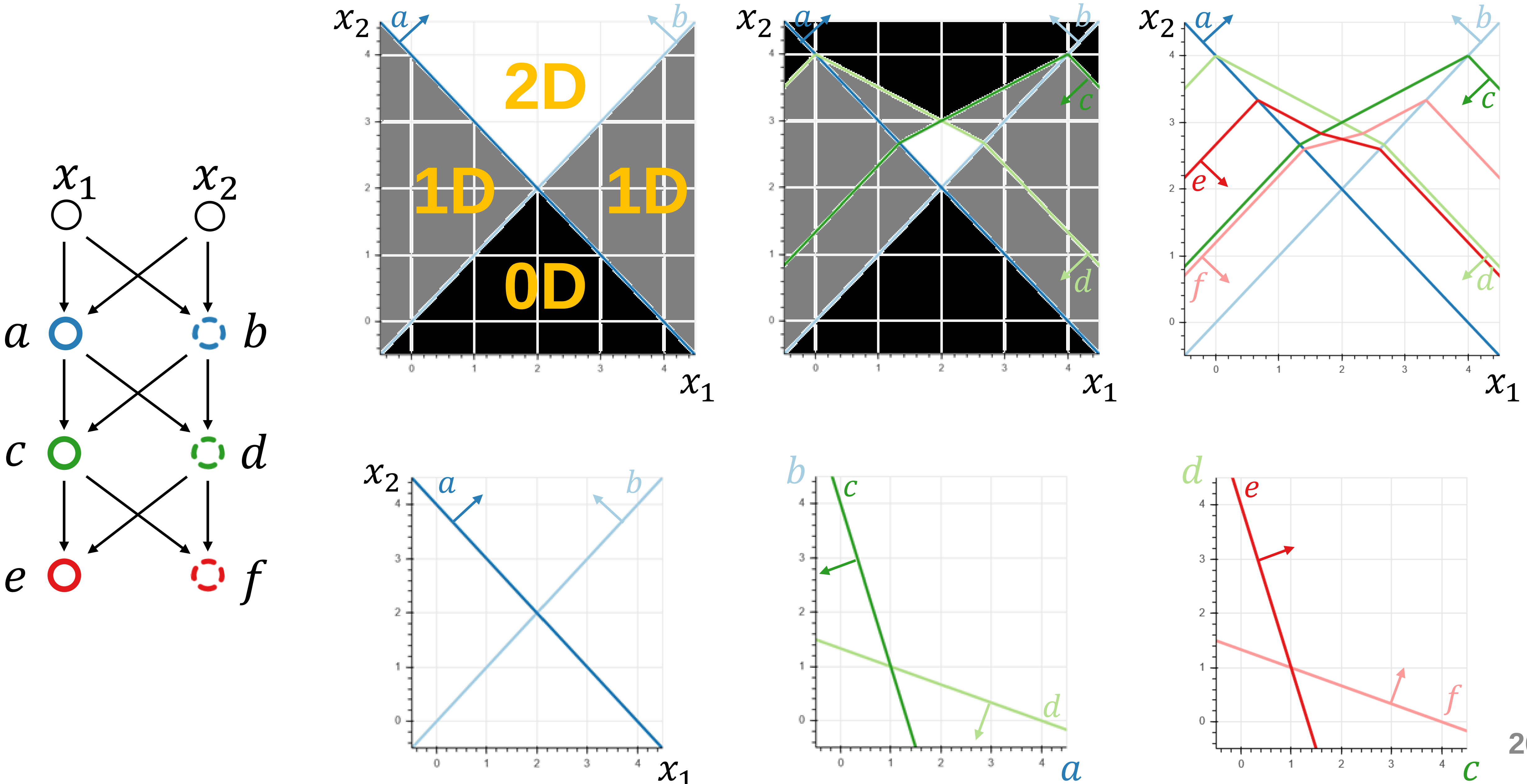
The Geometry of Linear Regions



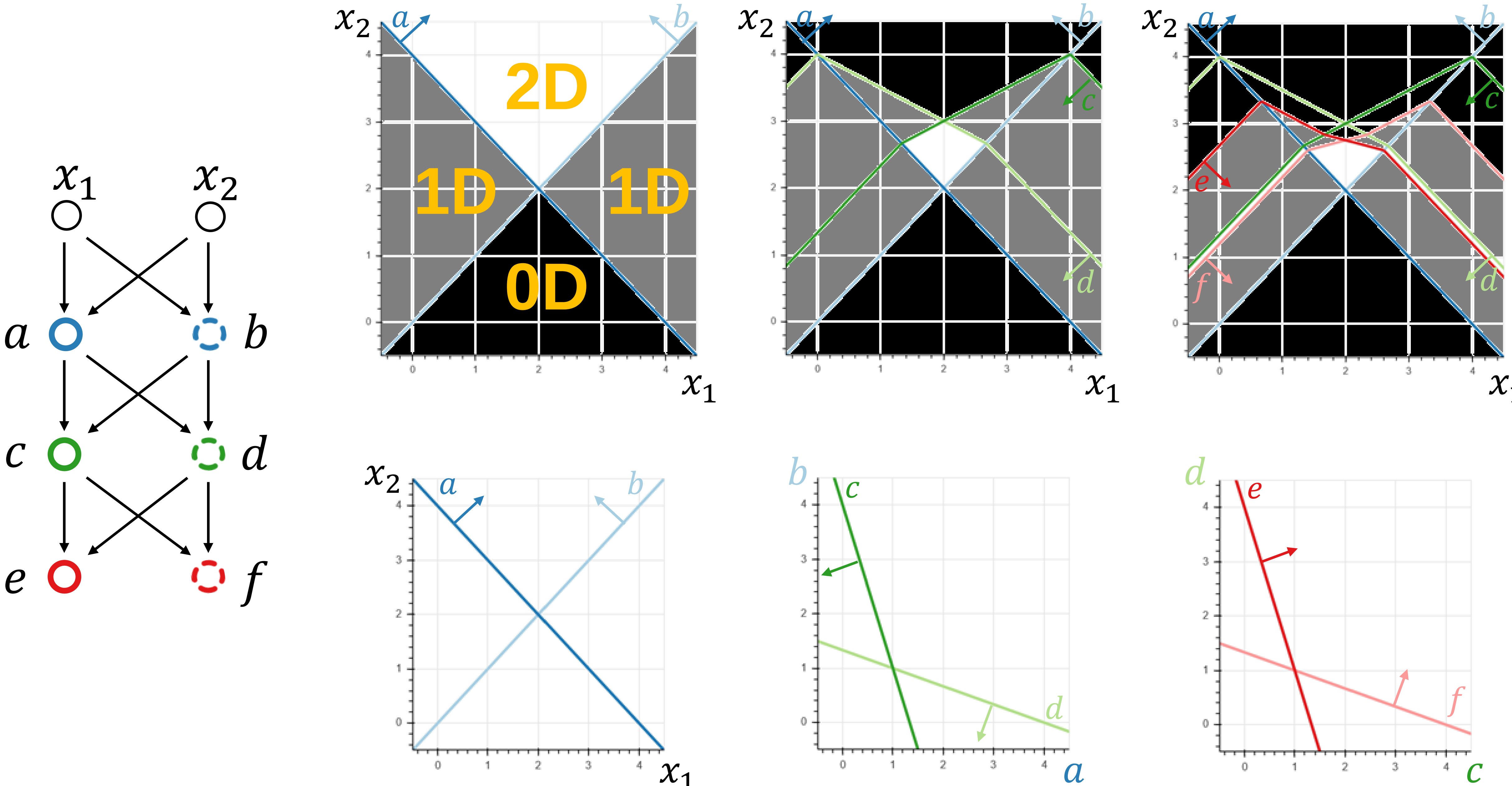
The Geometry of Linear Regions



The Geometry of Linear Regions



The Geometry of Linear Regions



The Number of Linear Regions

Theorem 1 **(S., Tjandraatmadja, Ramalingam 2018)**: For a rectifier DNN, there are at most

$$\sum_{(j_1,\ldots,j_L)\in J} \prod_{l=1}^{L} \binom{n_l}{j_l}$$

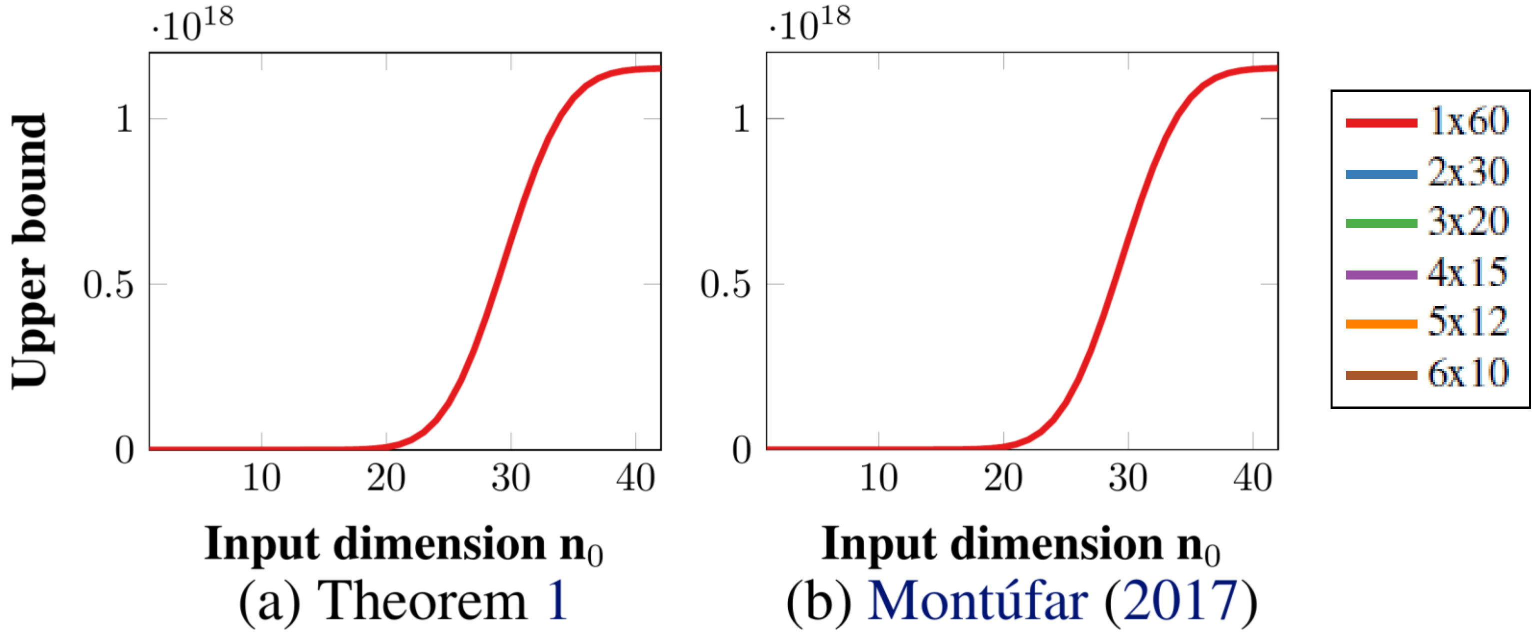
linear regions, where

$J = \{(j_1, \ldots, j_L) \in \mathbb{Z}^L : 0 \le j_l \le \min\{n_0, n_1 - j_1, \ldots, n_{l-1} - j_{l-1}, n_l\}\ \forall l = 1, \ldots, L\}$.

This bound is tight when $n_0 = 1$

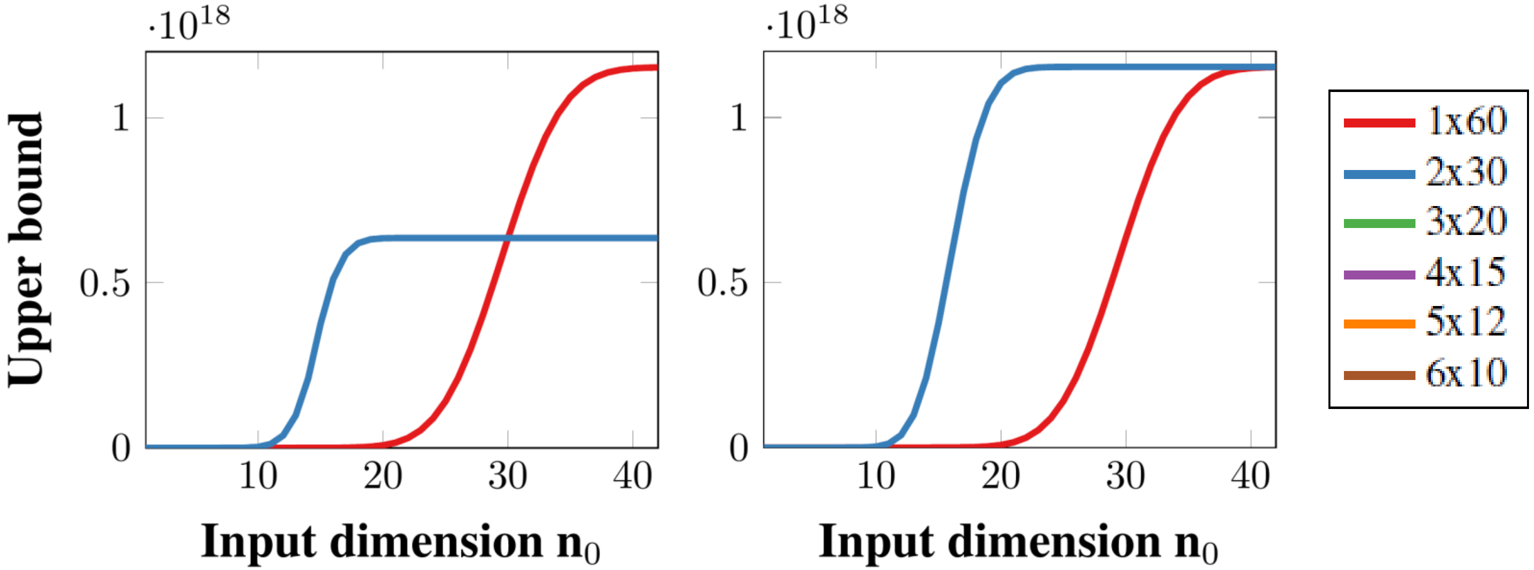
The Number of Linear Regions

We uniformly distribute 60 units in 1 to 6 layers and vary input dimension



The Number of Linear Regions

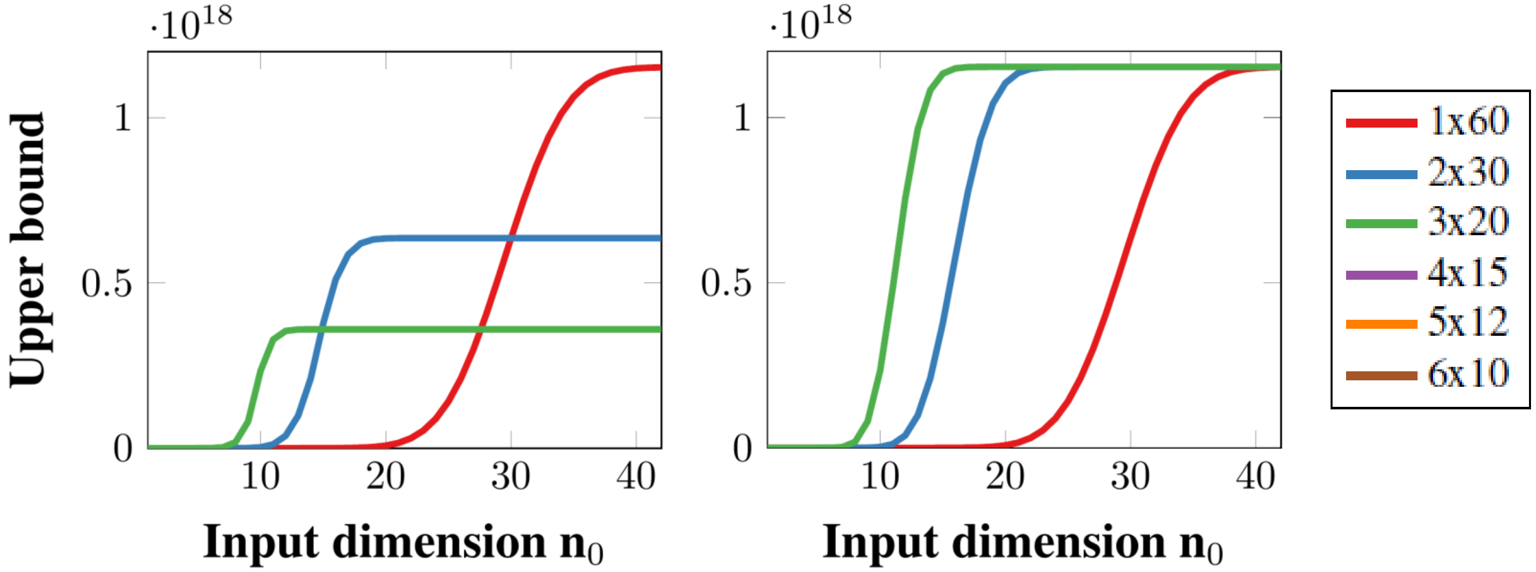
We uniformly distribute 60 units in 1 to 6 layers and vary input dimension



(a) Theorem 1 (b) Montúfar (2017)

The Number of Linear Regions

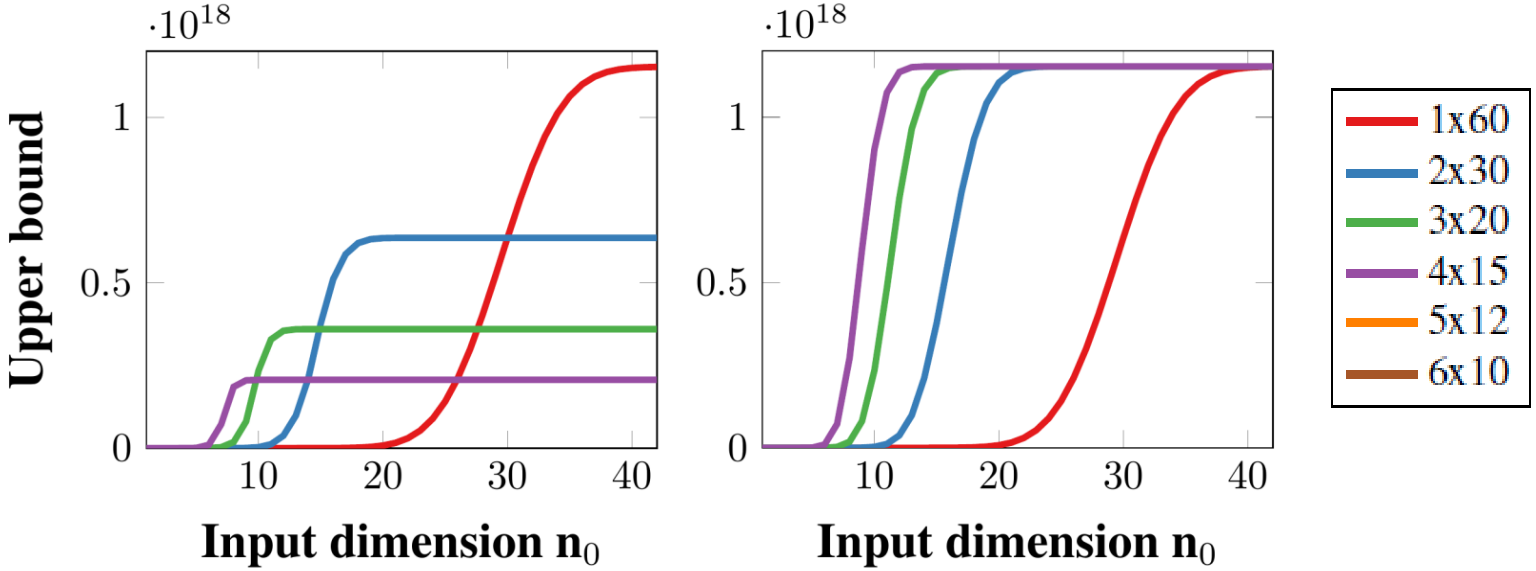
We uniformly distribute 60 units in 1 to 6 layers and vary input dimension



(a) Theorem 1 (b) Montúfar (2017)

The Number of Linear Regions

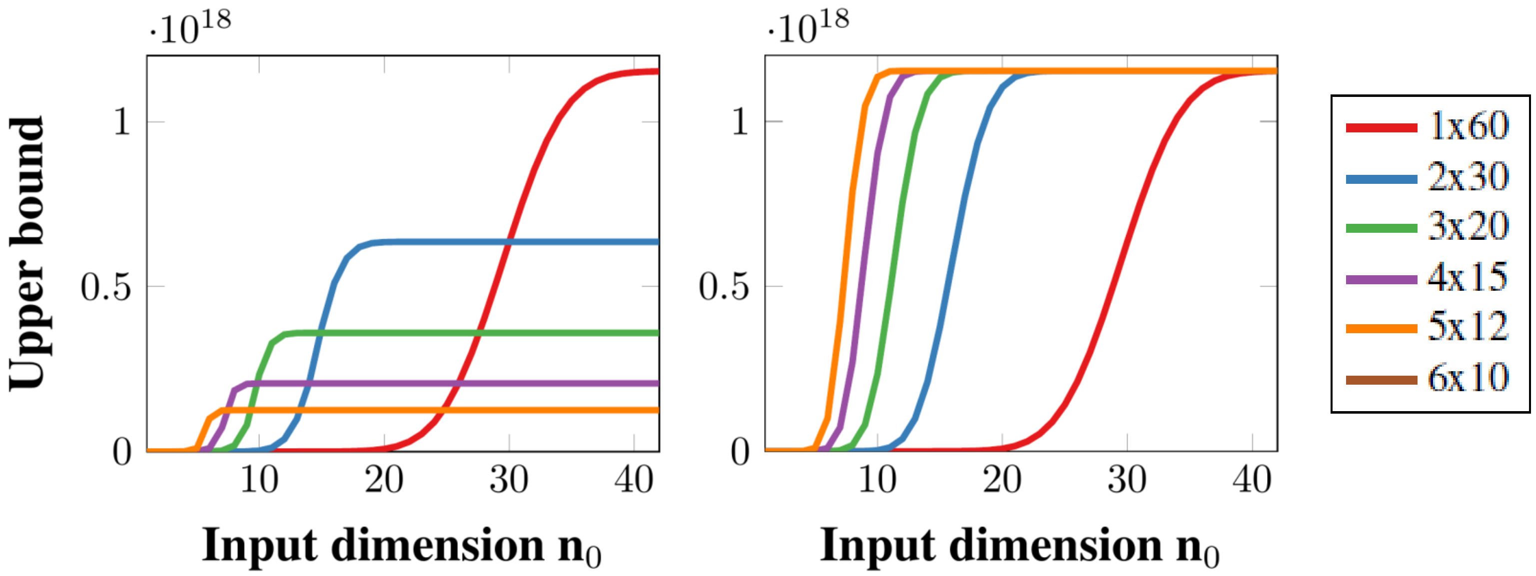
We uniformly distribute 60 units in 1 to 6 layers and vary input dimension



(a) Theorem 1 (b) Montúfar (2017)

The Number of Linear Regions

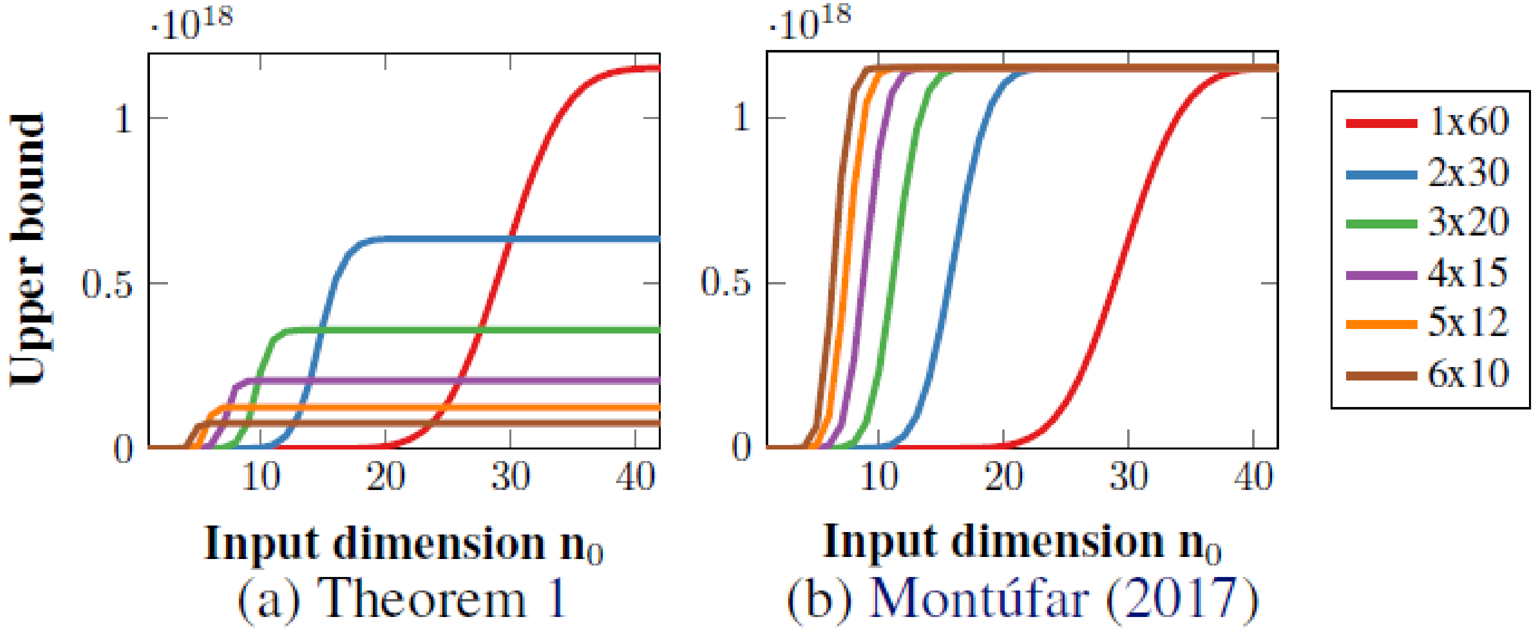
We uniformly distribute 60 units in 1 to 6 layers and vary input dimension



(a) Theorem 1 (b) Montúfar (2017)

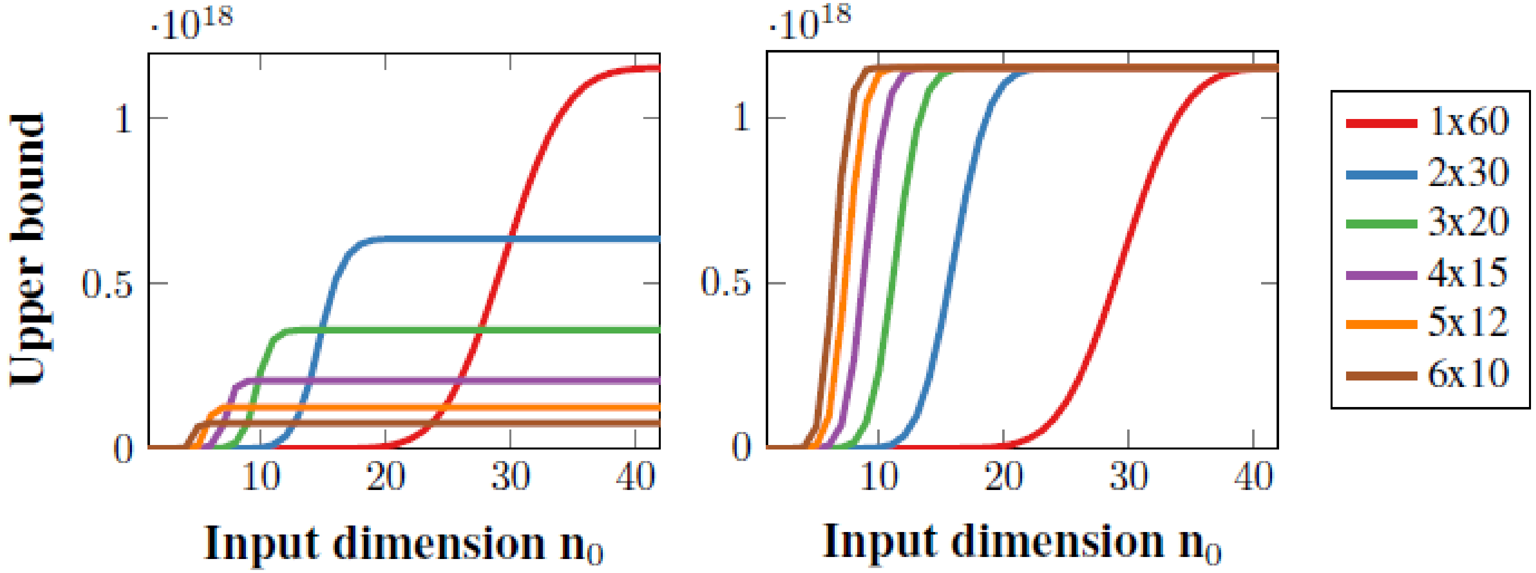
The Number of Linear Regions

We uniformly distribute 60 units in 1 to 6 layers and vary input dimension



The Number of Linear Regions

We uniformly distribute 60 units in 1 to 6 layers and vary input dimension

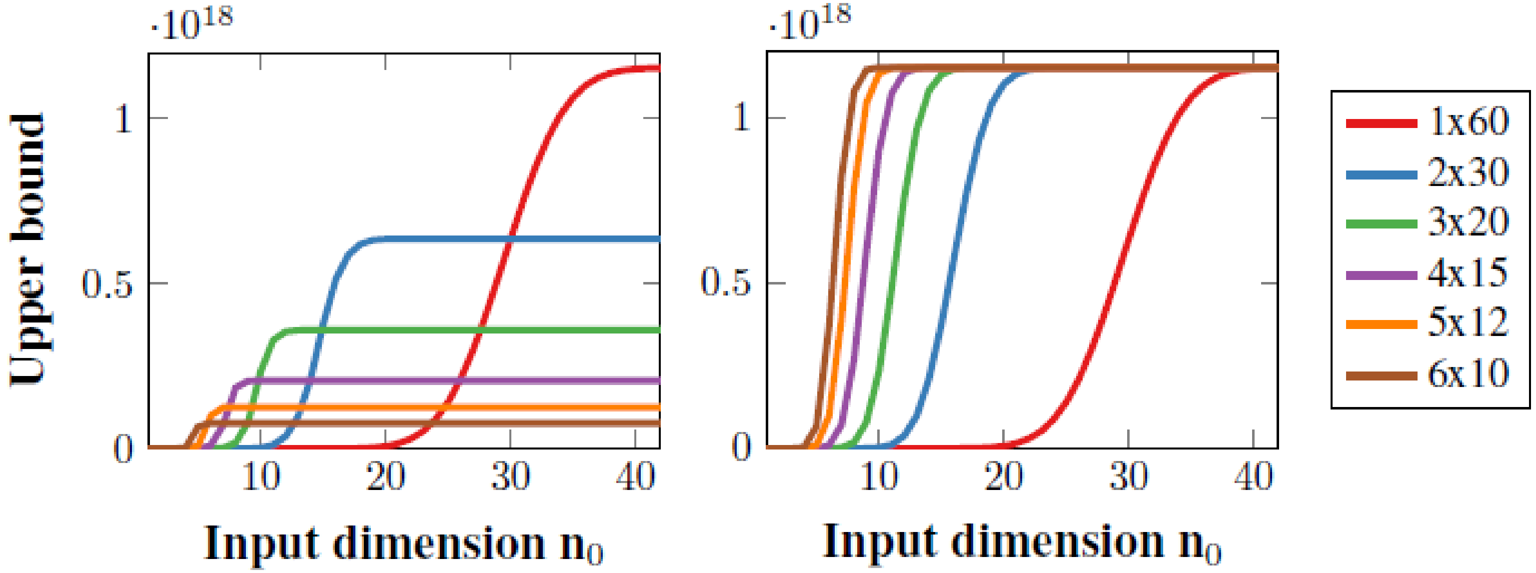


(a) Theorem 1 (b) Montúfar (2017)

- When the <u>input dimension is very large</u>, shallow networks have more LRs

The Number of Linear Regions

We uniformly distribute 60 units in 1 to 6 layers and vary input dimension



(a) Theorem 1 (b) Montúfar (2017)

- When the input dimension is very large, shallow networks have more LRs
- For a fixed input dimension, there is a depth that maximizes the bound

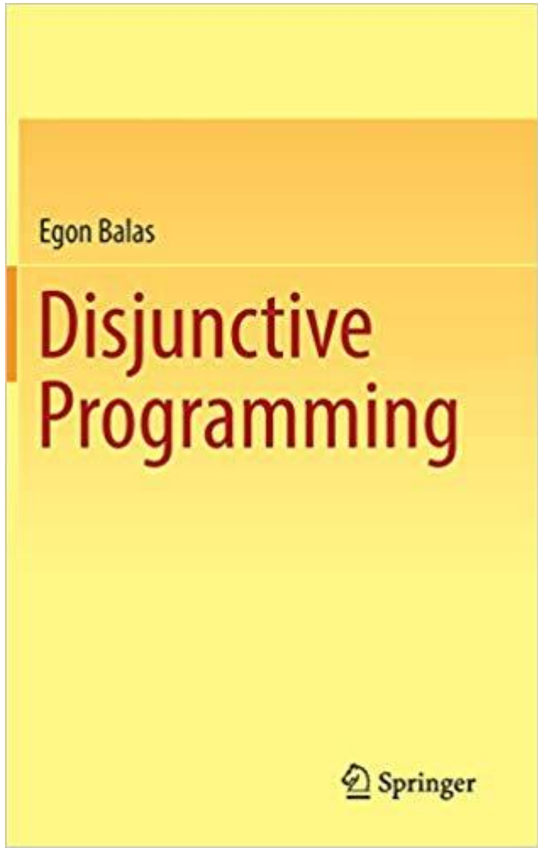
A Disjunctive Program

The union of the polyhedra corresponding to the sets of activation patterns is a disjunctive program, which can be translated to a MILP formulation

$$\bigvee_{(S^1,\ldots,S^L)\in\mathcal{S}} \left(\begin{array}{ll} h_i^l = W_i^l h^{l-1} + b_i^l \geq 0 & \forall\, i \in S^l, l \in \{1,\ldots,L\} \\ W_i^l h^{l-1} + b_i^l \leq 0 & \forall\, i \notin S^l, l \in \{1,\ldots,L\} \\ h_i^l = 0 & \forall\, i \notin S^l, l \in \{1,\ldots,L\} \end{array} \right)$$

We obtain the polyhedron in $\boldsymbol{x}$ by Fourier-Motzkin elimination

We find all linear regions using a mixed-integer formulation



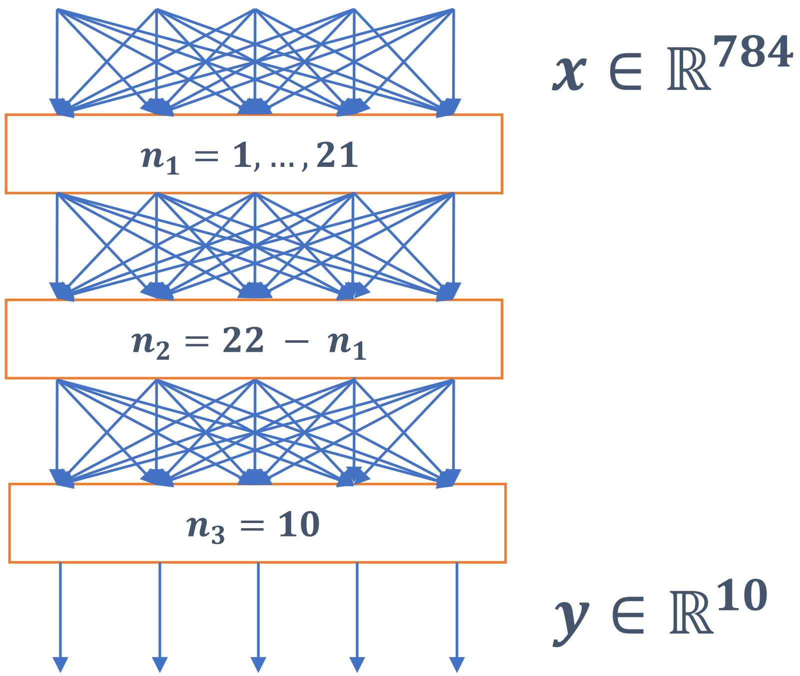
Experimental Setup

We trained rectifier networks on the MNIST benchmark

Experimental Setup

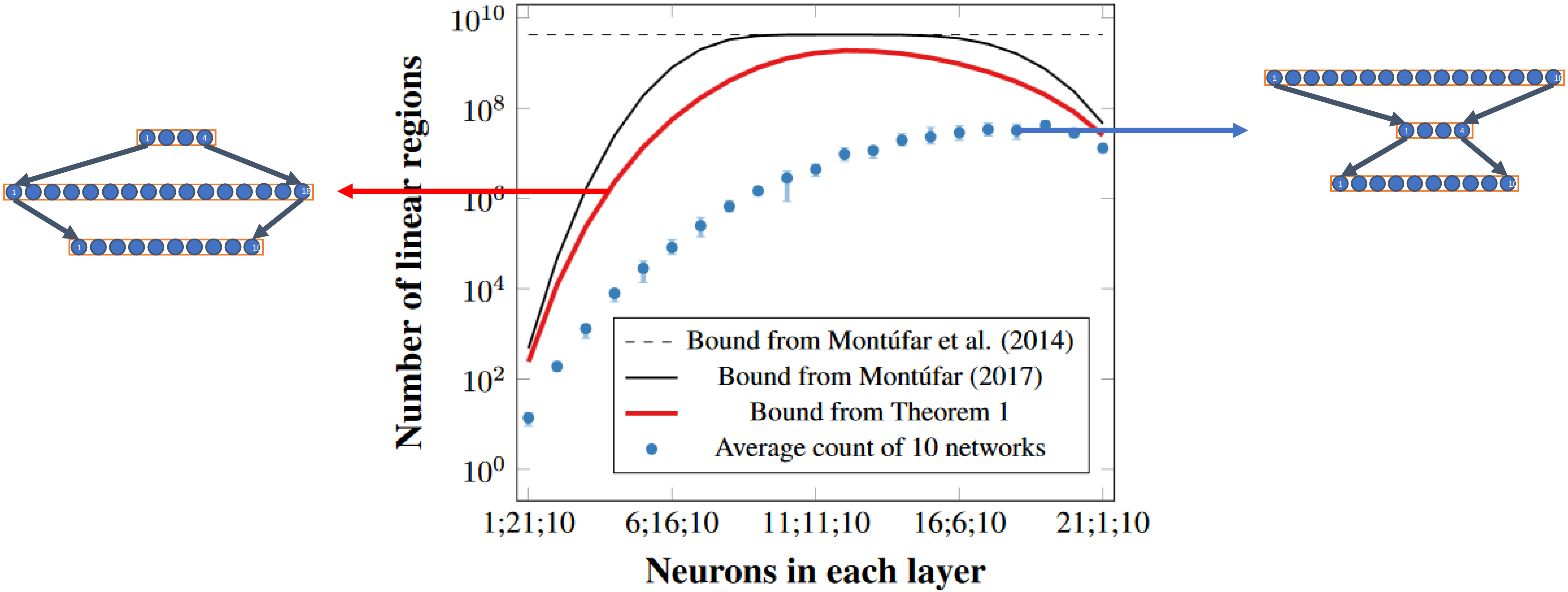
We trained rectifier networks on the MNIST benchmark

- Input is 28x28, final layer has 10 units (one per digit)
- Two other layers share 22 units
- For each possible configuration, 10 networks were trained and counted



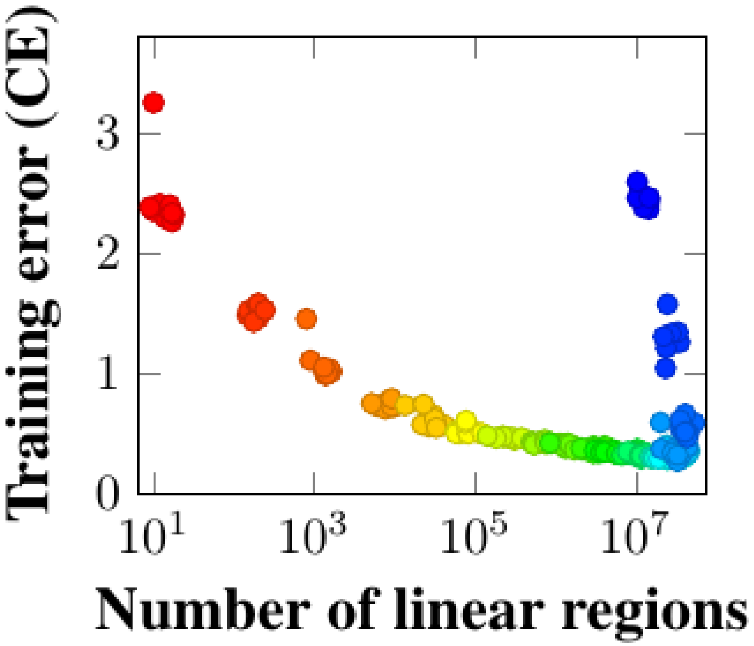
Bounding vs. Counting Results

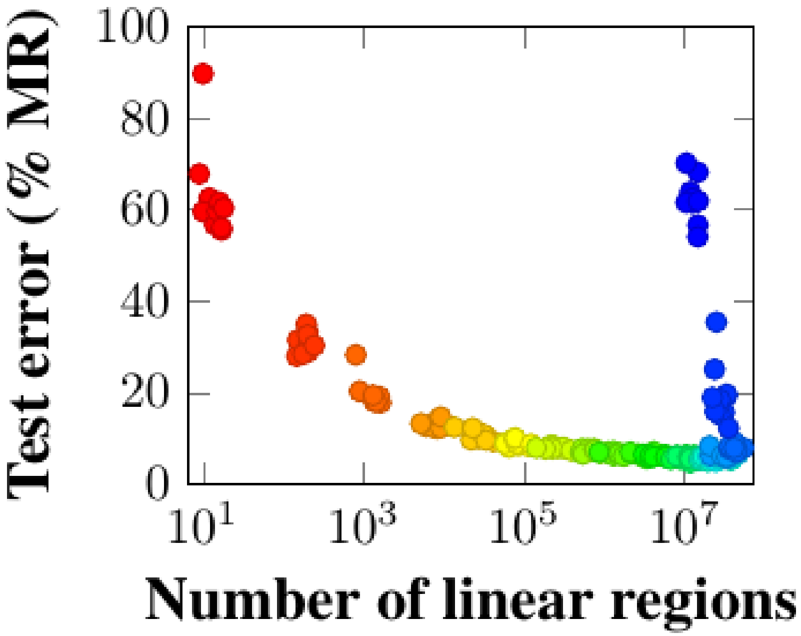
Comparison of bounds with average of 10 networks and min-max bars



Linear Regions and Accuracy

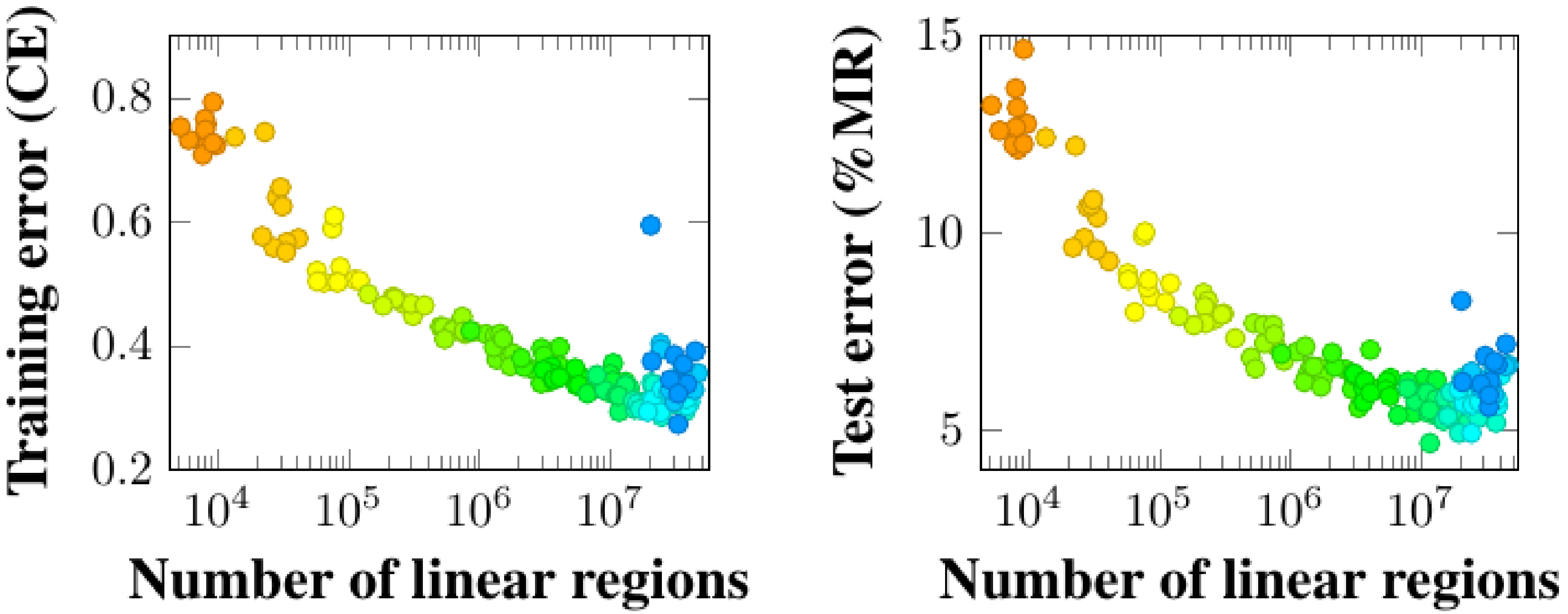
Plot with all points in heat scale by width, from 1,21,10 to 21,1,10





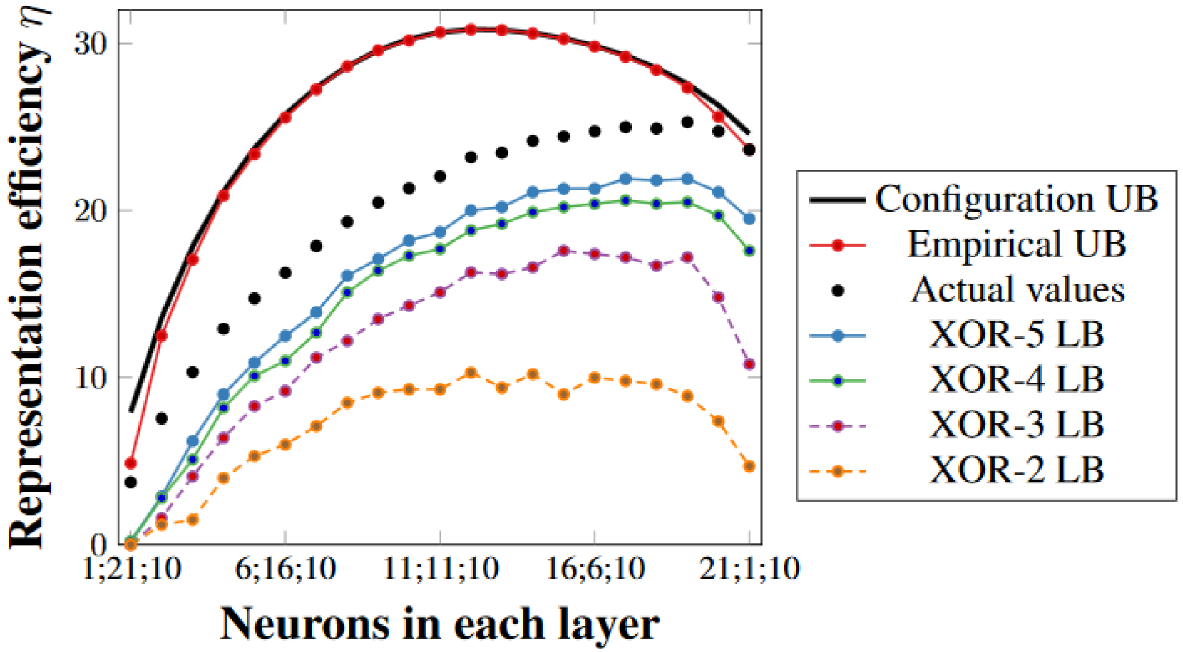
Linear Regions and Accuracy

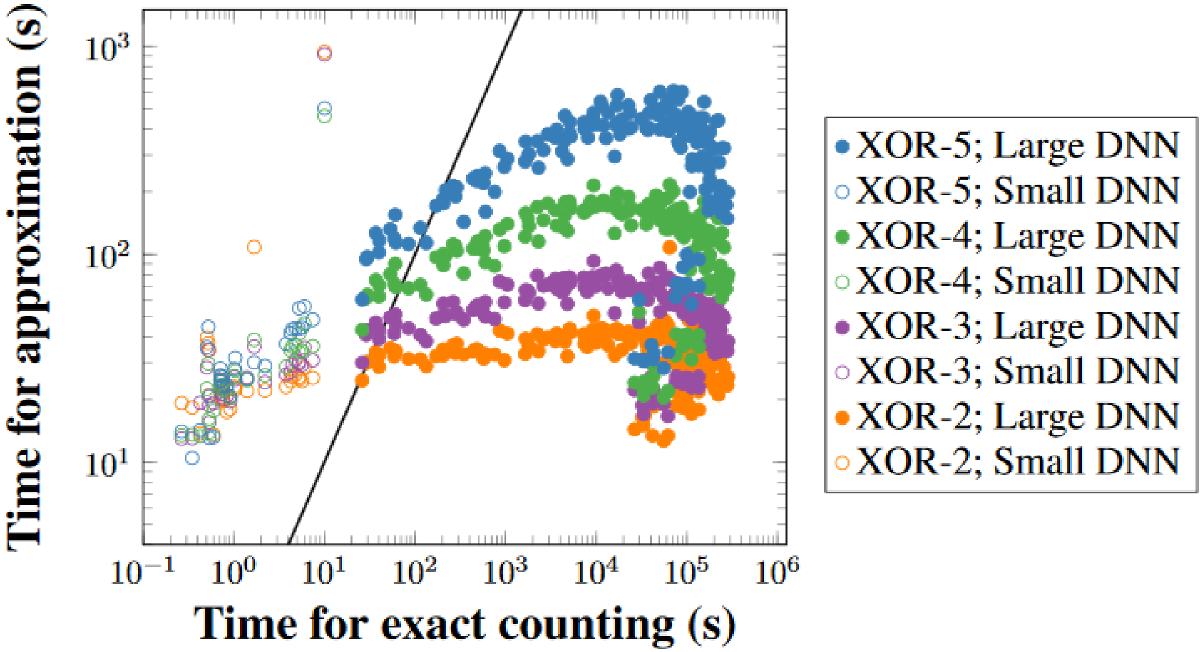
Same plot, but configurations are limited from 4,18,10 to 18,4,10



Empirical Bounding Results

Comparison of bound with coefficients and approximate counting

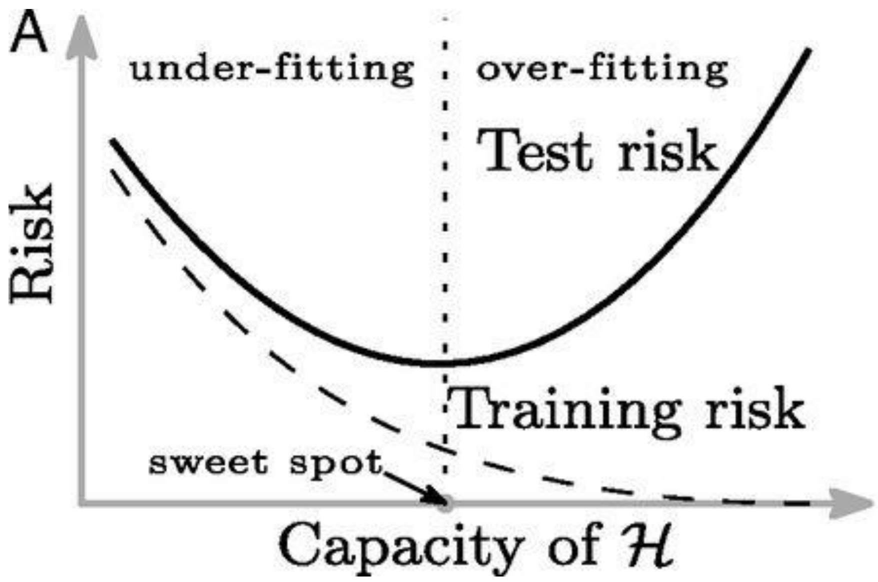




What About Overfitting?

In traditional ML, we aim for a **trade-off between training and test** errors

- In **sufficient large neural networks**, it if often possible to obtain a good generalization with training error approaching zero

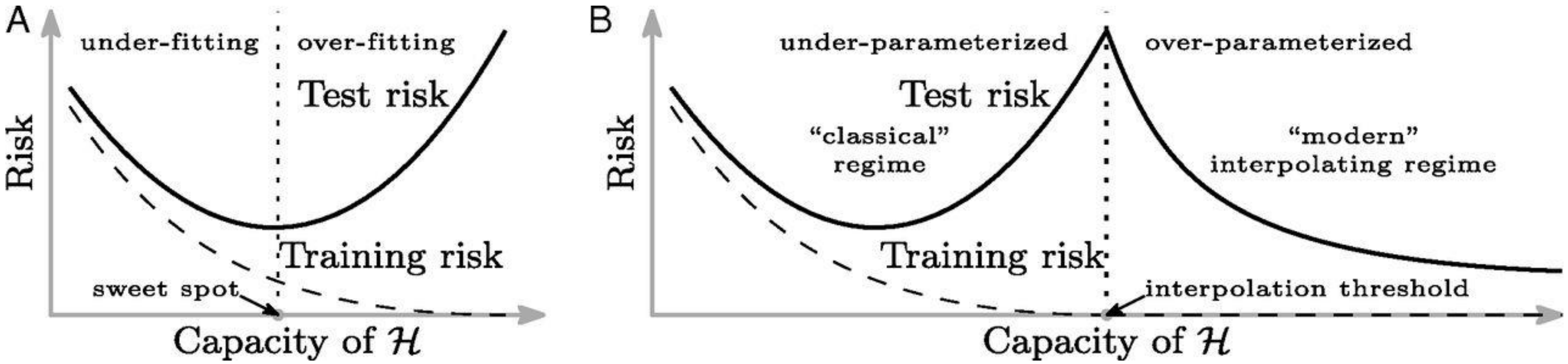


https://www.pnas.org/content/116/32/15849

What About Overfitting?

In traditional ML, we aim for a **trade-off between training and test** errors

- In **sufficient large neural networks**, it if often possible to obtain a good generalization with training error approaching zero



https://www.pnas.org/content/116/32/15849

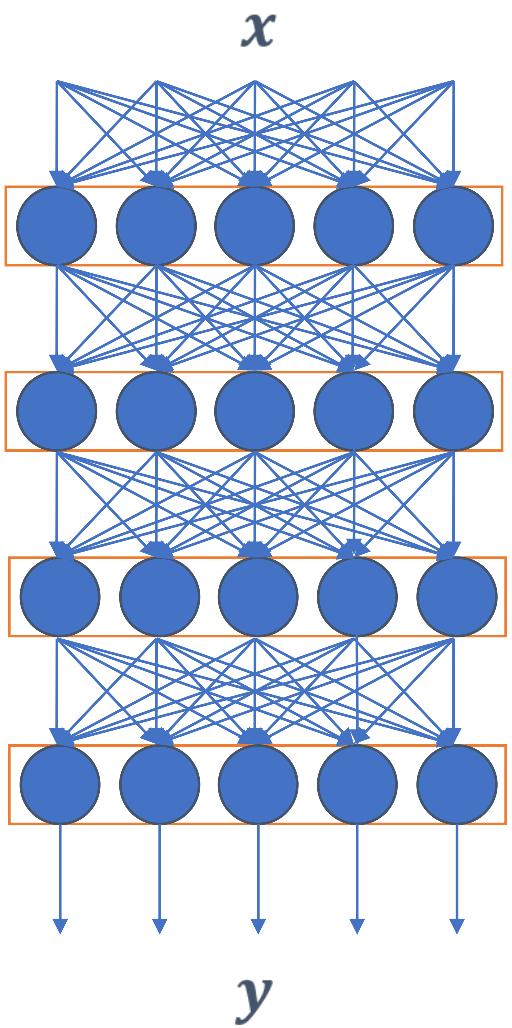
What could make them smaller?

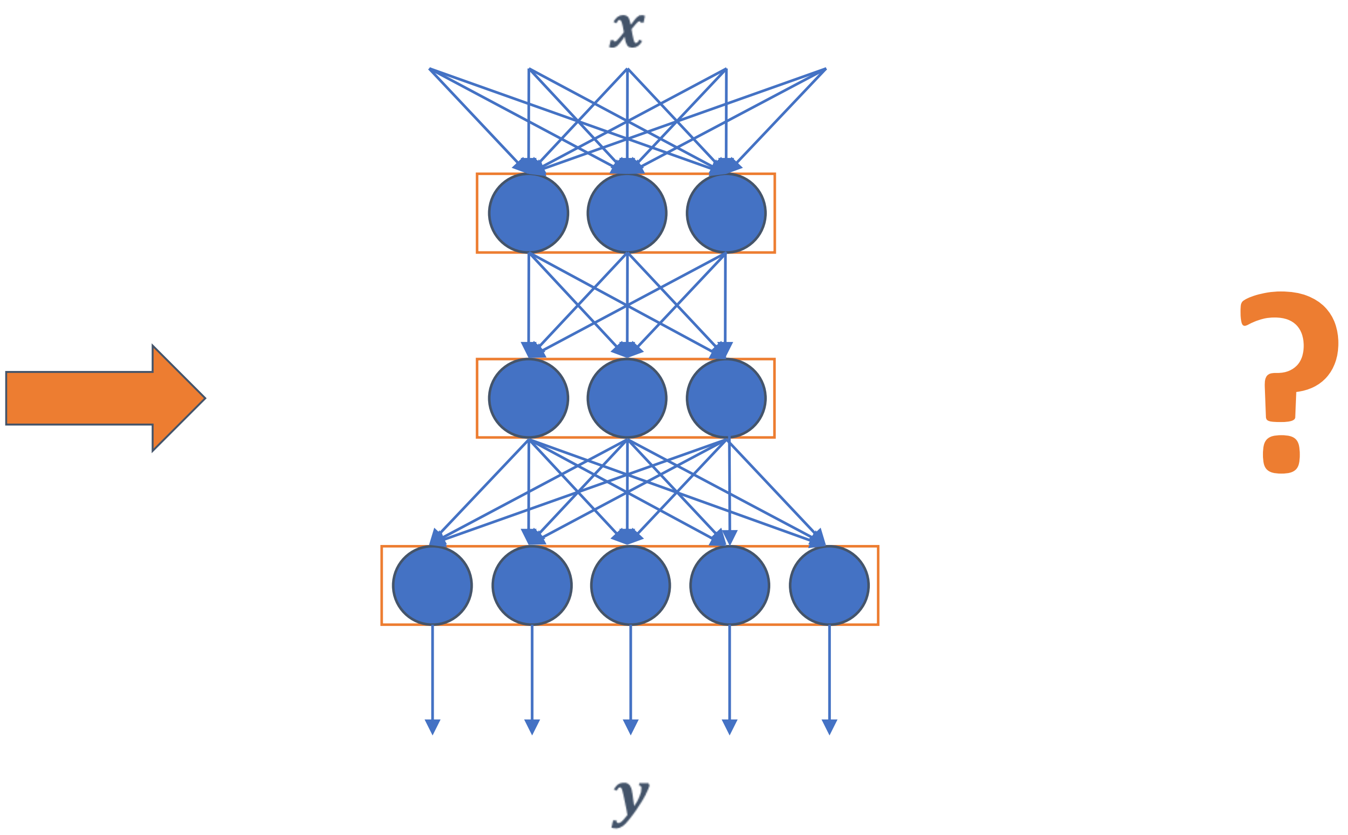
https://deadline.com/2020/02/honey-i-shrunk-the-kids-reboot-rick-moranis-1202858344/

Part 2

Lossless Compression of Rectifier Networks

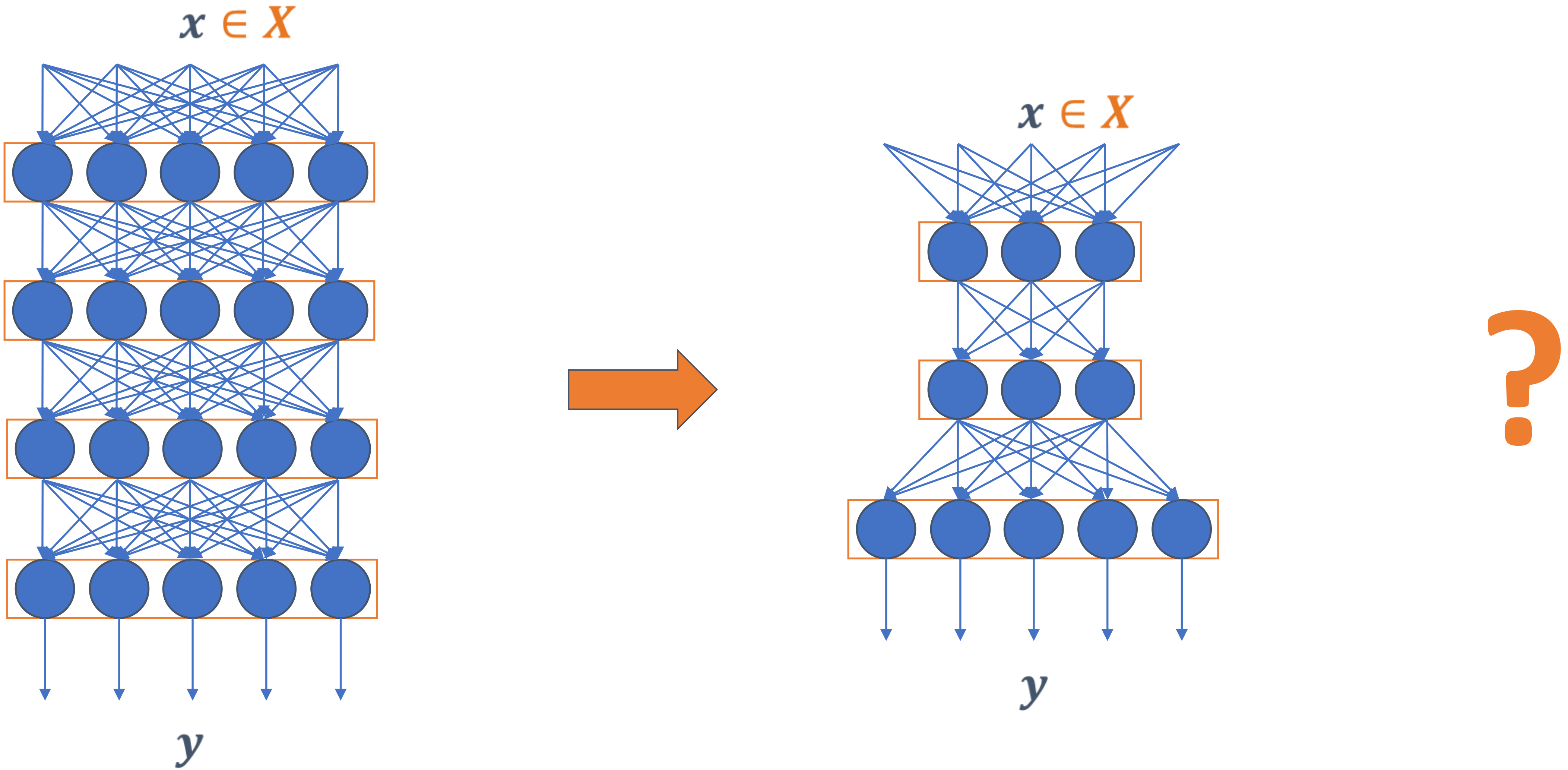
Can we find a smaller neural network that models the same $x \to y$ function?





Lossless Compression of Rectifier Networks

Can we find a smaller neural network that models the same $x \to y$ function, **at least for the inputs that are relevant for a given application?**

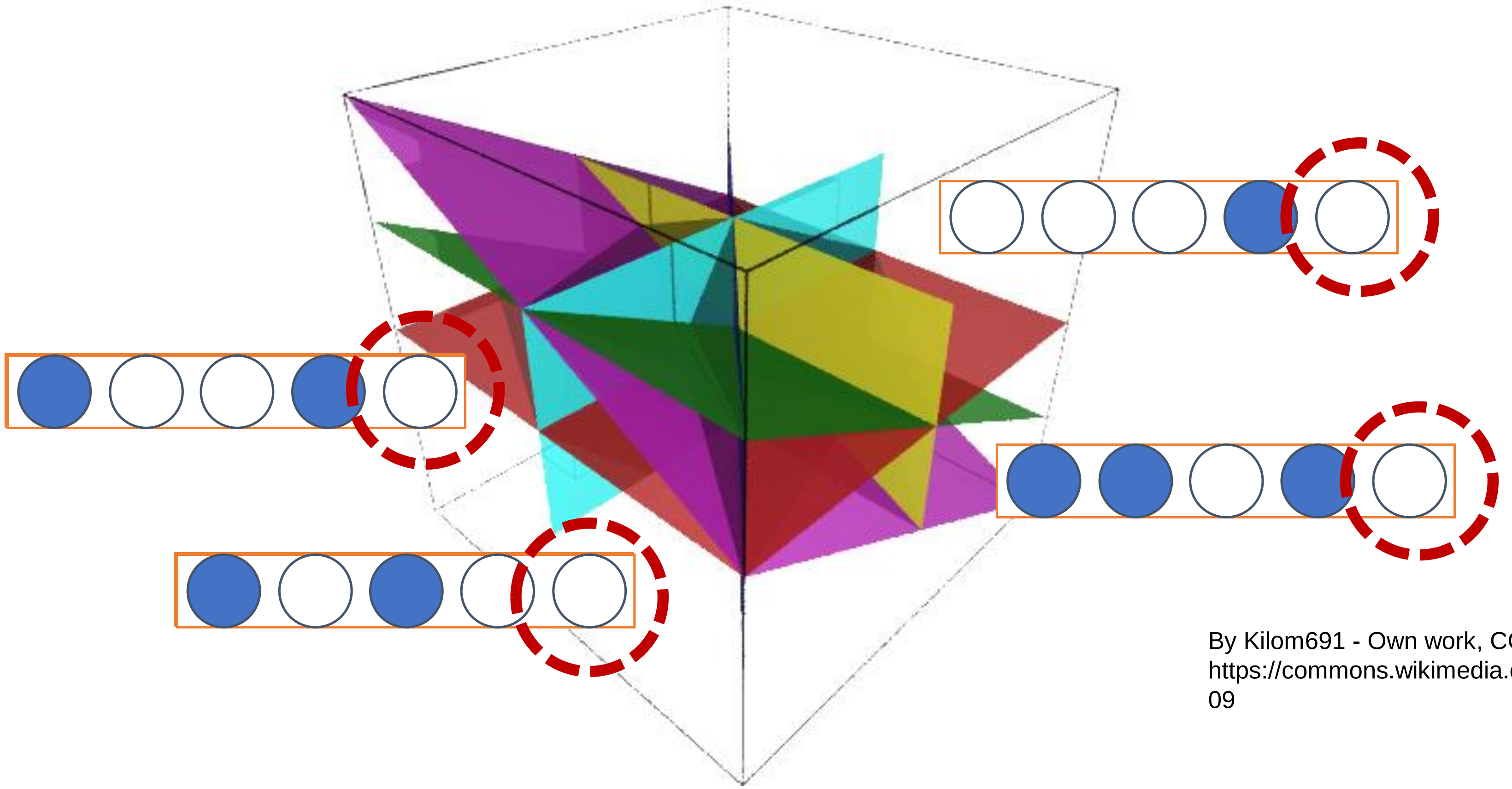


Lossless Compression of Rectifier Networks

For networks trained on **MNIST**, we only need equivalence for $\boldsymbol{x} \in [\mathbf{0}, \mathbf{1}]^{\mathbf{784}}$

Linear Regions and Stability

Each linear region is defined as the set of inputs yielding the same pattern of active units on each layer of the network

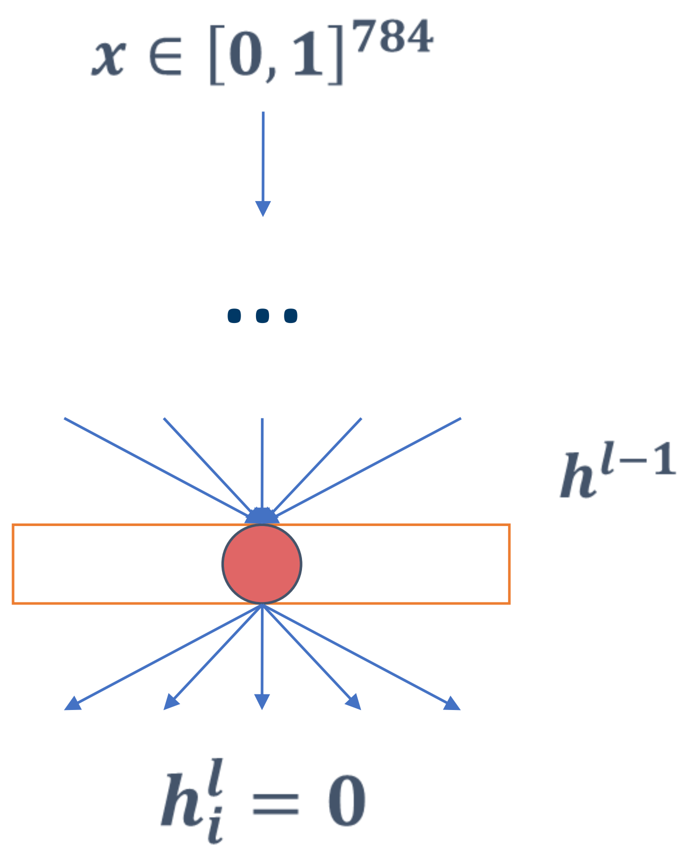


By Kilom691 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=37508909

We may be able to simplify neural networks if their units are stable

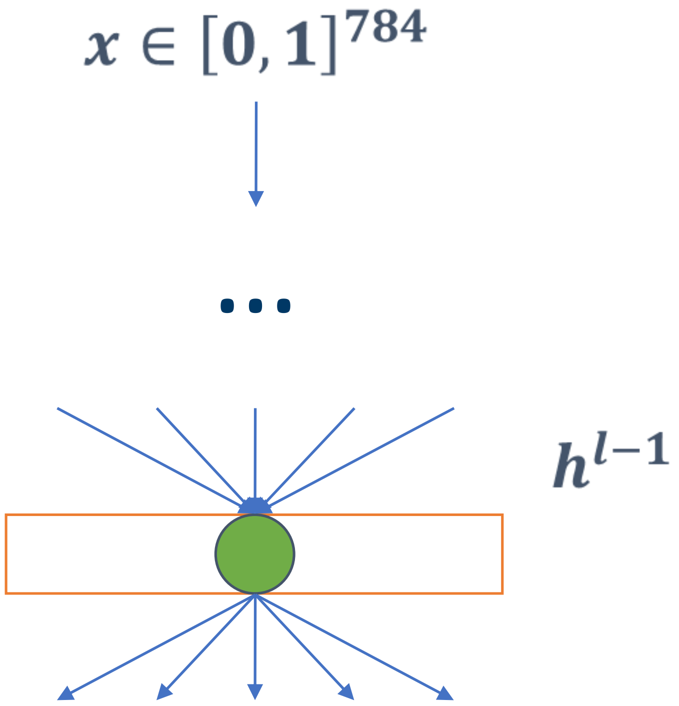
Unit Stability with Respect to a Domain

A unit is **stably inactive** if it never produces a positive output



Unit Stability with Respect to a Domain

A unit is **stably active** if it <u>always</u> produces a positive output

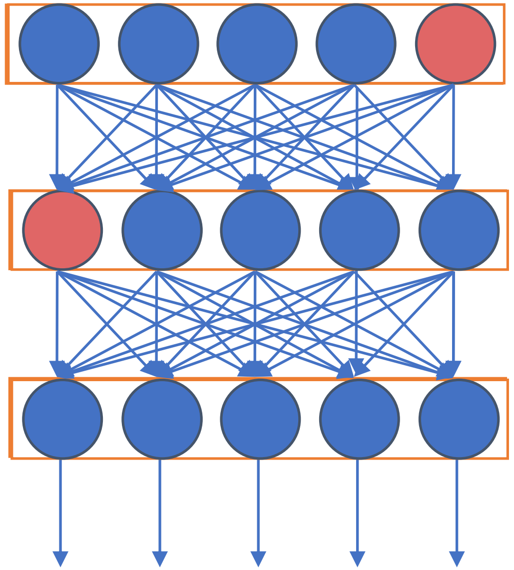


$$h_i^l = W_i^l h^{l-1} + b_i^l > 0$$

In both cases, the absence of nonlinearity may help us simplify the network without changing the function that it models

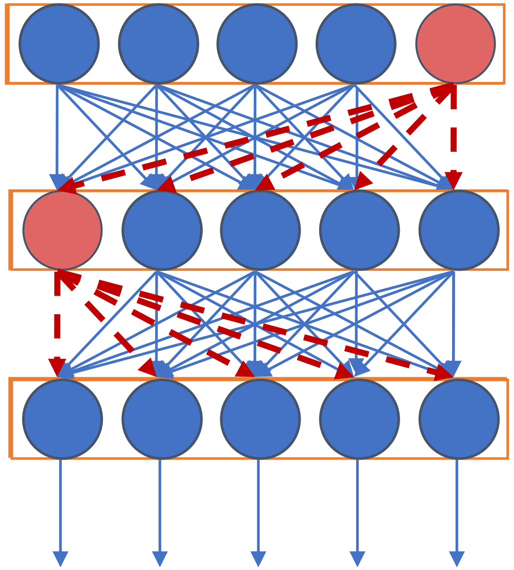
Lossless Removal of Units

Let us suppose that the units in red are **stably inactive**



Lossless Removal of Units

Let us suppose that the units in red are **stably inactive**

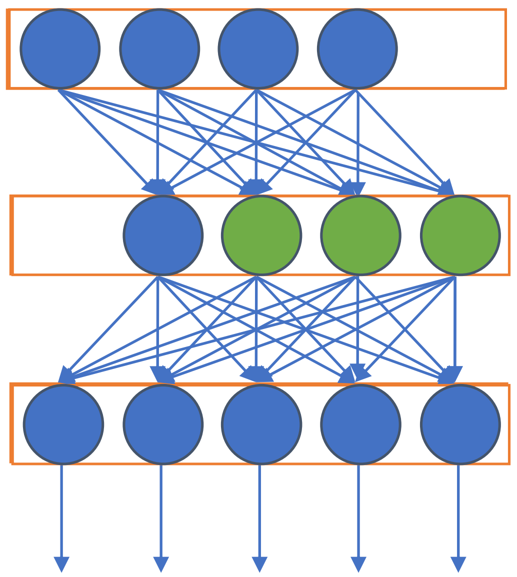


Their outputs can be ignored

and the units can be removed

Lossless Removal of Units

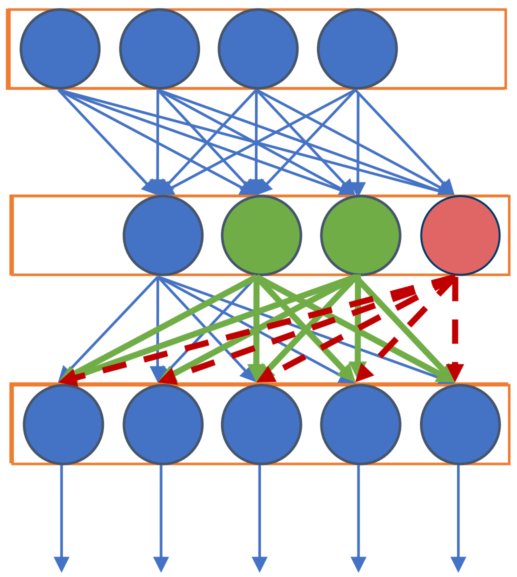
Let us suppose that the 3 units in green are **stably active** and they define a set S for which $\mathrm{rank}(\boldsymbol{W}_S^l) = 2$



The output of one unit can be defined using the other two

Lossless Removal of Units

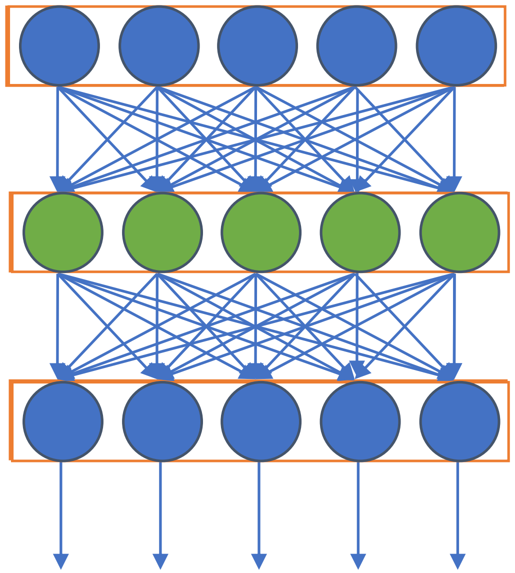
Let us suppose that the 3 units in green are **stably active** and they define a set S for which $\text{rank}\left(\boldsymbol{W}_S^l\right) = 2$



We can remove one unit by adjusting the weights in the following layer accordingly

Lossless Removal of Layers

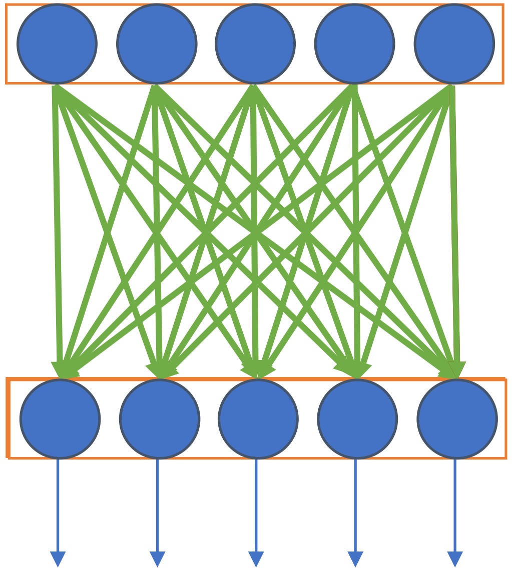
Now let us suppose that all units in the middle layer are **stably active**



Without nonlinearities, we have an affine transformation of the output of the previous layer to the input of the next layer

Lossless Removal of Layers

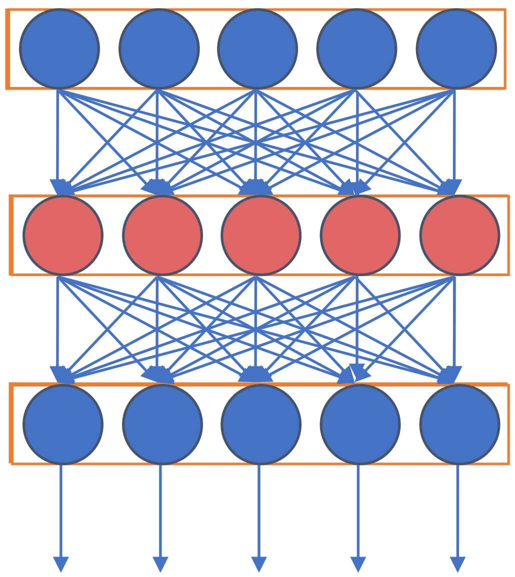
Now let us suppose that all units in the middle layer are **stably active**



We can fold that layer by directly connecting the other layers and adjusting the weights

Lossless Removal of Layers

Likewise, let us suppose that all units in the middle layer are **stably inactive**

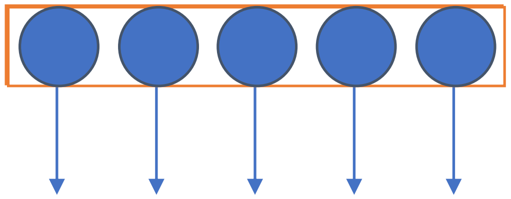


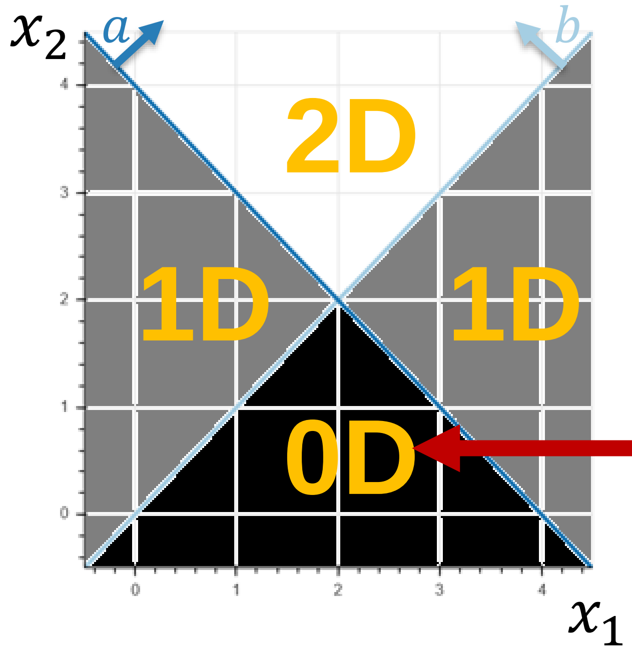
The input values no longer matter, since they all map to a single intermediary point (0,0,0,0,0)

Lossless Removal of Layers

Likewise, let us suppose that all units in the middle layer are **stably inactive**

We can collapse the neural network to a single output layer since it models a constant function





Mapping Inputs to Outputs

The following constraints represent a ReLU $\boldsymbol{i}$ in layer $\boldsymbol{l}$:

$$\boldsymbol{W_i^l h^{l-1} + b_i^l = g_i^l}$$

$$g_i^l = h_i^l - \bar{h}_i^l$$

$$h_i^l \geq 0$$

$$\bar{h}_i^l \geq 0$$

$$h_i^l \leq H_i^l z_i^l$$

$$\bar{h}_i^l \leq \bar{H}_i^l (1 - z_i^l)$$

$$z_i^l \in \{0, 1\}$$

$\bar{h}_i^l$ is the output of a fictitious complementary unit

z_i^l is a binary variable modeling if the neuron is active

H_i^l and $\bar{H}_i^l$ are sufficiently large and positive constants (bounded inputs)

Mapping Inputs to Outputs

The following constraints represent a ReLU $\boldsymbol{i}$ in layer $\boldsymbol{l}$:

$$\boldsymbol{W_i^l h^{l-1} + b_i^l = g_i^l}$$
$$\boldsymbol{g_i^l = h_i^l - \overline{h}_i^l}$$
$$h_i^l \geq 0$$
$$\overline{h}_i^l \geq 0$$
$$h_i^l \leq H_i^l z_i^l$$
$$\overline{h}_i^l \leq \overline{H}_i^l (1 - z_i^l)$$
$$z_i^l \in \{0, 1\}$$

- $\boldsymbol{\overline{h}_i^l}$ is the output of a fictitious complementary unit
- z_i^l is a binary variable modeling if the neuron is active
- H_i^l and $\overline{H}_i^l$ are sufficiently large and positive constants (bounded inputs)

Mapping Inputs to Outputs

The following constraints represent a ReLU $\boldsymbol{i}$ in layer $\boldsymbol{l}$:

$$\boldsymbol{W_i^l h^{l-1} + b_i^l = g_i^l}$$
$$\boldsymbol{g_i^l = h_i^l - \bar{h}_i^l}$$
$$\boldsymbol{h_i^l \geq 0}$$
$$\boldsymbol{\bar{h}_i^l \geq 0}$$

- $\boldsymbol{\bar{h}_i^l}$ is the output of a fictitious complementary unit

Mapping Inputs to Outputs

The following constraints represent a ReLU $\boldsymbol{i}$ in layer $\boldsymbol{l}$:

$$\boldsymbol{W_i^l h^{l-1} + b_i^l = g_i^l}$$
$$\boldsymbol{g_i^l = h_i^l - \bar{h}_i^l}$$
$$\boldsymbol{h_i^l \geq 0}$$
$$\boldsymbol{\bar{h}_i^l \geq 0}$$
$$\boldsymbol{z_i^l \in \{0, 1\}}$$

- $\boldsymbol{\bar{h}_i^l}$ is the output of a fictitious complementary unit
- $\boldsymbol{z_i^l}$ is a binary variable modeling if the neuron is active

Mapping Inputs to Outputs

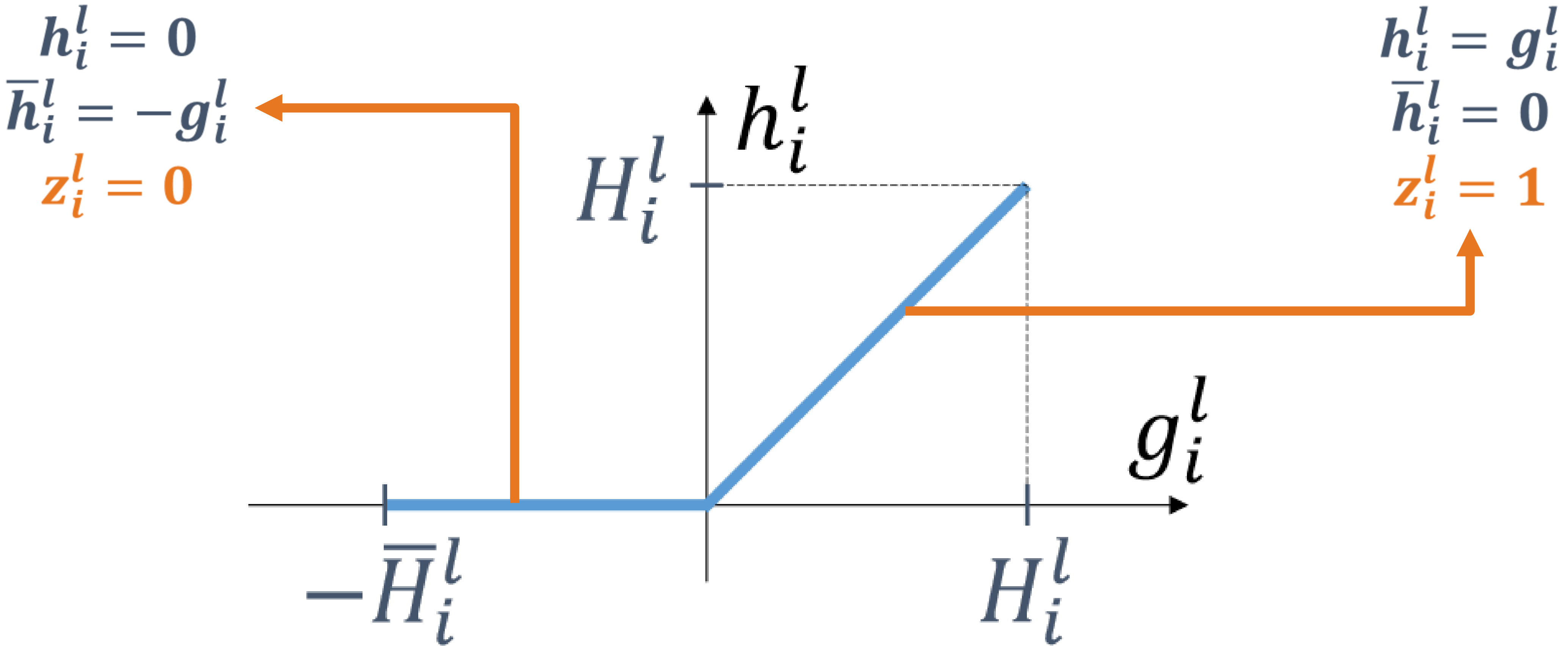
The following constraints represent a ReLU $\boldsymbol{i}$ in layer $\boldsymbol{l}$:

$$\boldsymbol{W_i^l h^{l-1} + b_i^l = g_i^l}$$
$$\boldsymbol{g_i^l = h_i^l - \bar{h}_i^l}$$
$$\boldsymbol{h_i^l \geq 0}$$
$$\boldsymbol{\bar{h}_i^l \geq 0}$$
$$\boldsymbol{z_i^l \in \{0, 1\}}$$
$$\boldsymbol{h_i^l \leq H_i^l\, z_i^l}$$
$$\boldsymbol{\bar{h}_i^l \leq \bar{H}_i^l (1 - z_i^l)}$$

- $\boldsymbol{\bar{h}_i^l}$ is the output of a fictitious complementary unit
- $\boldsymbol{z_i^l}$ is a binary variable modeling if the unit is active
- $\boldsymbol{H_i^l}$ and $\boldsymbol{\bar{H}_i^l}$ are sufficiently large and positive constants (bounded inputs)

Mapping Inputs to Outputs

The binary variable changes the mapping according to unit activity



Identifying Stable Units

If $\max\{g_i^l \mid$ (input/output mapping of previous units), $x \in D \} < 0$,

the unit is **stably inactive** and never produces a positive output

And we don't need an optimal solution:

A negative upper bound suffices

Identifying Stable Units

If $\min\{g_i^l \mid$ (input/output mapping of previous units), $x \in D \quad\} > 0$,

the unit is **stably active** and the output is an affine transformation

Once more, we don't need an optimal solution:

A positive lower bound suffices

We can use Mixed-Integer Linear Programming (MILP) solvers to determine unit stability

Finding Stability

Lossless compression is possible in practice if:

1. Domain is restricted (global vs. local stability)
2. DNN trained to induce sparsity (e.g., ℓ_1 regularization)

$$h_i^l = max\,\{\,0, W_i^l h^{l-1} + b_i^l\,\}$$

If the bias is in a higher order of magnitude than the weights, the weights have limited effect on whether the unit is active or not

Results

For each layer width and ℓ_1 regularization weight, we train 31 networks with 2 hidden layers and 10 units of output on the MNIST dataset.

Layer Width	Regulari-zation	Accuracy (%)	Compression (%)	Runtime (s)
25	0.001	95.76 ± 0.05	22 ± 1	27.9 ± 0.3
25	**0.0002**	**97.24 ± 0.02**	**8.3 ± 0.7**	**29 ± 1**
25	0.0	96.68 ± 0.03	0 ± 0	28.4 ± 0.3

The amount of regularization that improves accuracy also induces compressibility!

Results

For each layer width and ℓ_1 regularization weight, we train 31 networks with 2 hidden layers and 10 units of output on the MNIST dataset.

Layer Width	Regulari-zation	Accuracy (%)	Compression (%)	Runtime (s)
50	0.001	96.05 ± 0.04	29.4 ± 0.6	103 ± 2
50	**0.0002**	**97.81 ± 0.02**	**7.5 ± 0.5**	**106 ± 3**
50	0.0	97.62 ± 0.02	0 ± 0	112 ± 3

The amount of regularization that improves accuracy also induces compressibility!

Results

For each layer width and ℓ_1 regularization weight, we train 31 networks with 2 hidden layers and 10 units of output on the MNIST dataset.

Layer Width	Regulari-zation	Accuracy (%)	Compression (%)	Runtime (s)
100	0.0005	97.14 ± 0.02	30.8 ± 0.5	421 ± 4
100	**0.0001**	**98.14 ± 0.01**	**14.9 ± 0.4**	**456 ± 8**
100	0.0	98.00 ± 0.01	0 ± 0	385 ± 2

The amount of regularization that improves accuracy also induces compressibility!

References

Serra, Tjandraatmadja, Ramalingam: "**Bounding and Counting Linear Regions of Deep Neural Networks**" – **ICML 2018**

https://arxiv.org/abs/1711.02114

Serra, Ramalingam: "**Empirical Bounds on Linear Regions of Deep Rectifier Networks**" – **AAAI 2020**

https://arxiv.org/abs/1810.03370

Serra, Kumar, Ramalingam: "**Lossless Compression of Deep Neural Networks**" – **CPAIOR 2020**

https://arxiv.org/abs/2001.00218

Kumar, Serra, Ramalingam: "**Equivalent and Approximate Transformation of Deep Neural Networks**" – **arXiv preprint**

https://arxiv.org/abs/1905.11428

"**My Neural Network is a Piecewise Linear Regression, but Which One?**" – **ThiagoSerra.com/blog**

https://thiagoserra.com/2020/02/05/my-neural-network-is-a-piecewise-linear-regression-but-which-one-a-glimpse-of-our-aaai-20-paper-on-empirical-bounds/