Scaling Up Exact Neural Network Compression by ReLU Stability

https://www.claimcompass.eu/blog/how-to-pack-a-carry-on/

Thiago Serra
Bucknell University

Joint work with:

Abhinav Kumar
Michigan State University

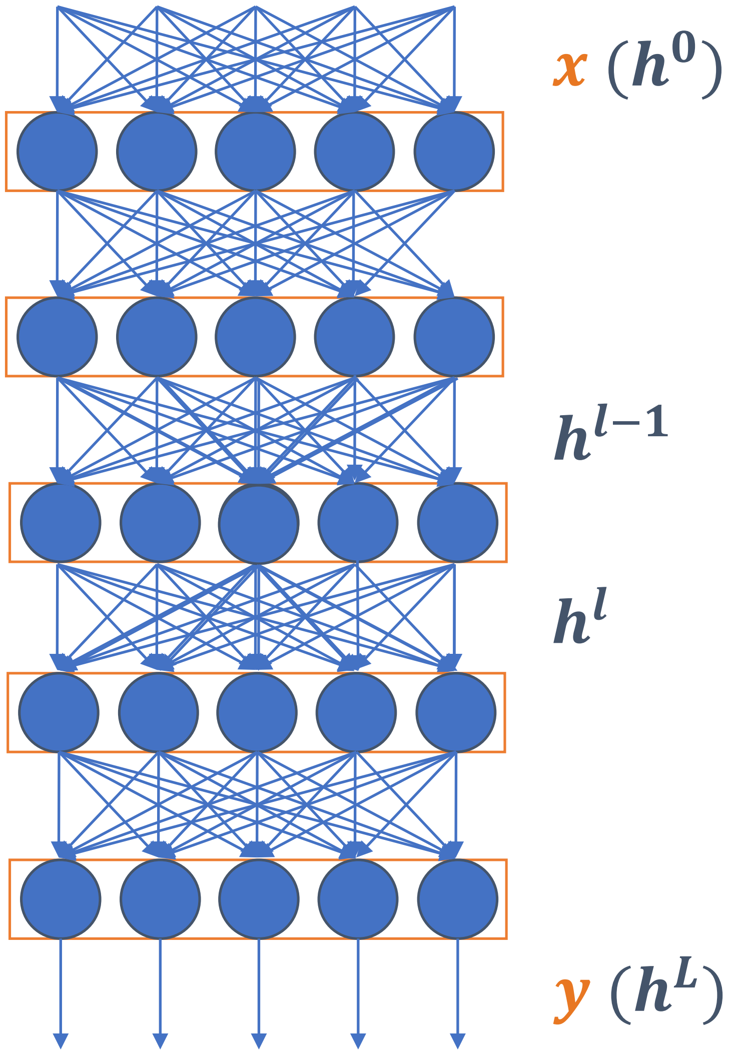
Srikumar Ramalingam
Google Research

Notation and Scope

A feedforward neural network models a function from **x** to **y**

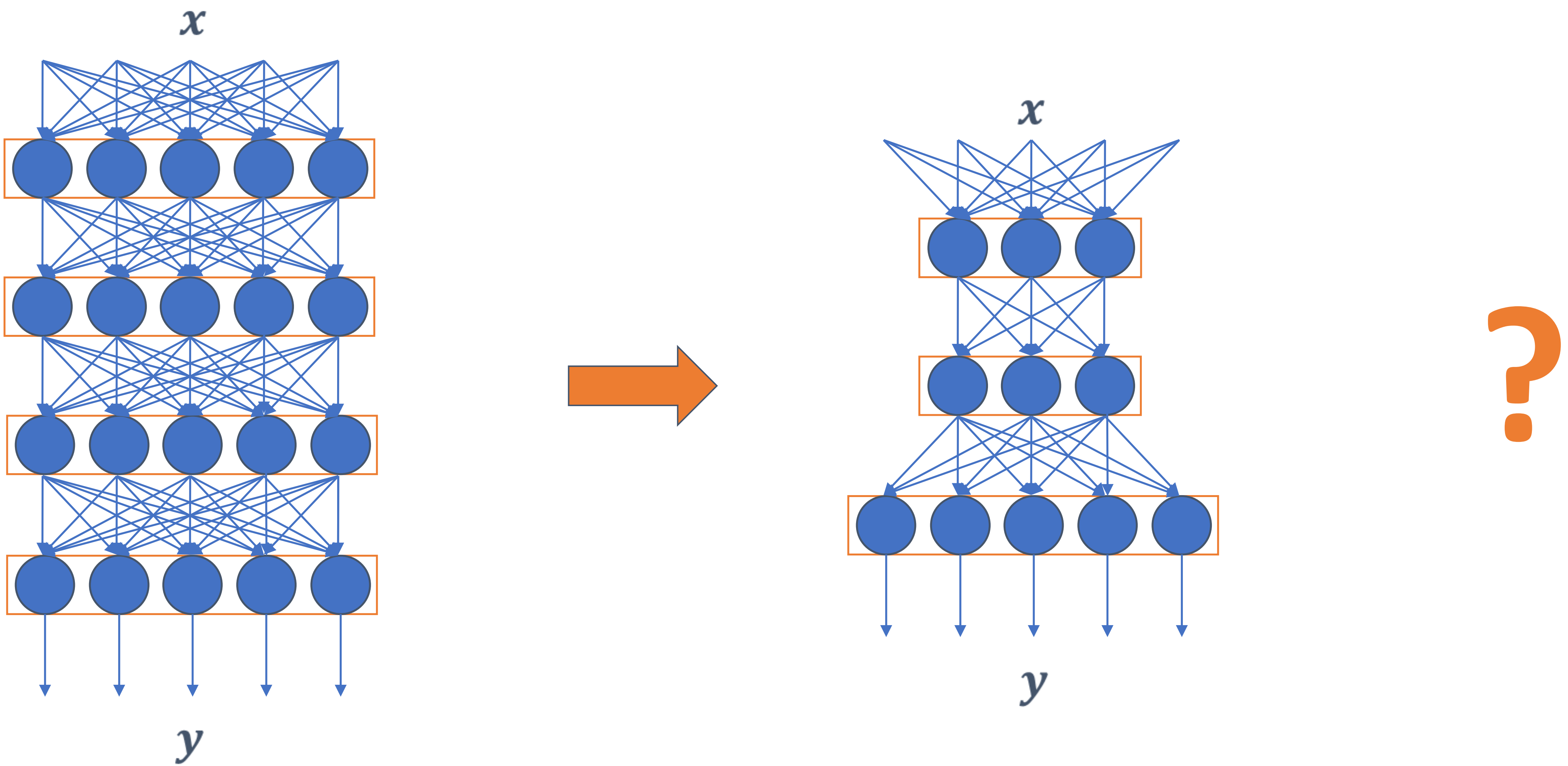
Let us assume that every neuron is a
Rectified Linear Unit (ReLU)

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



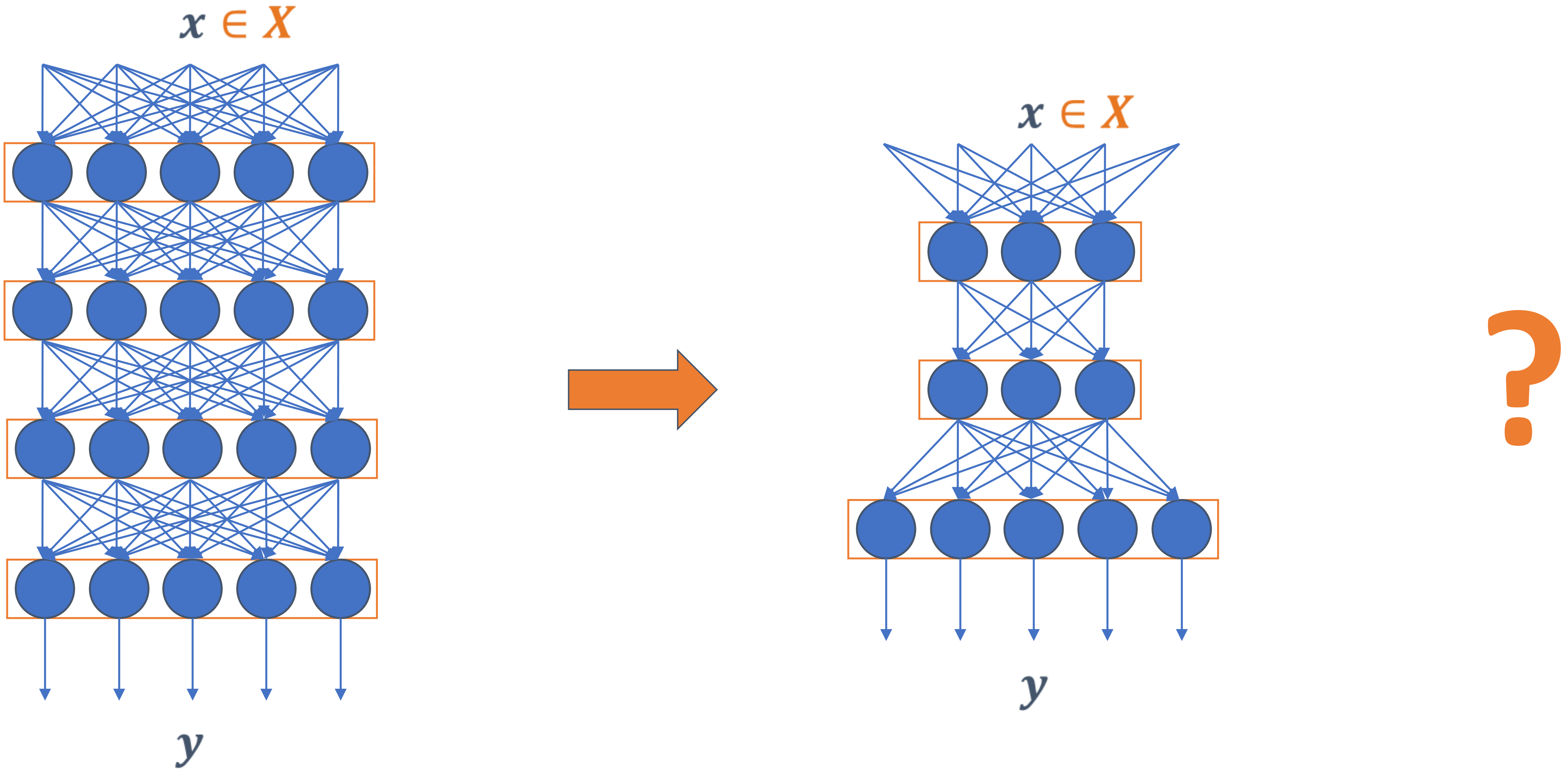
Exact Compression of Rectifier Networks

Can we find a smaller neural network that models the exact same function?



Exact Compression of Rectifier Networks

Can we find a smaller neural network that models the exact same function, **at least for the inputs that are relevant for a given application?**



Exact Compression of Rectifier Networks

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

Josef Steppan - Own work, ,
https://commons.wikimedia.org/w/index.php?curid=37508909

Related Work

Extensive literature on **inexact compression**

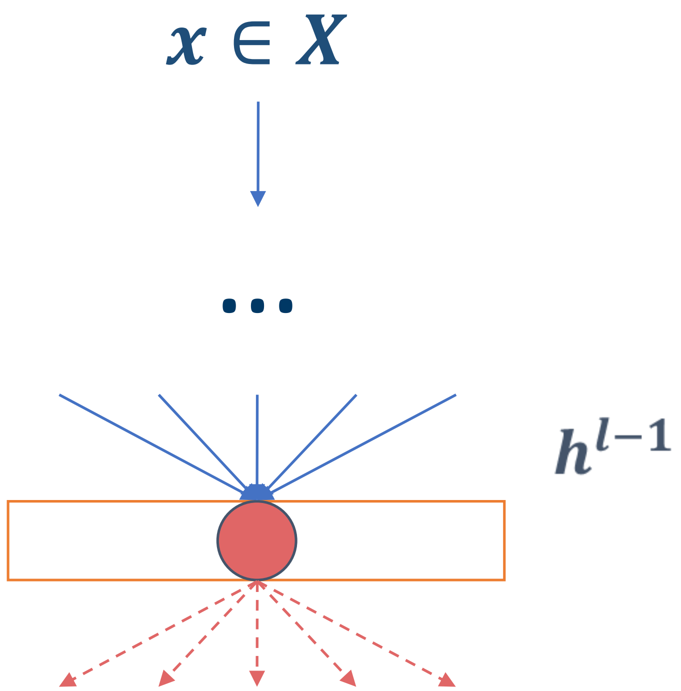
- Embedding neural networks in smaller devices
- **Denil et al. (2013)**: Redundancy between network parameters
- **Arora et al. (2018)**: Better generalization bounds
- **Blalock et al. (2020)**: Trade-off between compression and accuracy
- **Hooker et al. (2019)**: Compressed networks are less robust; loss in accuracy is disproportionally distributed across classes
- **ElAraby, Wolf & Carvalho (2020)**: MILP for inexact compression

Exact compression is **relatively unexplored**

- Avoids side effects above; and does not require retraining
- **Sourek, Zelezny, Kuzelka (2021)**: Symmetry in graph neural networks
- **Serra, Kumar & Ramalingam (2020)**: Small rectifier networks

Neuron Stability with Respect to a Domain

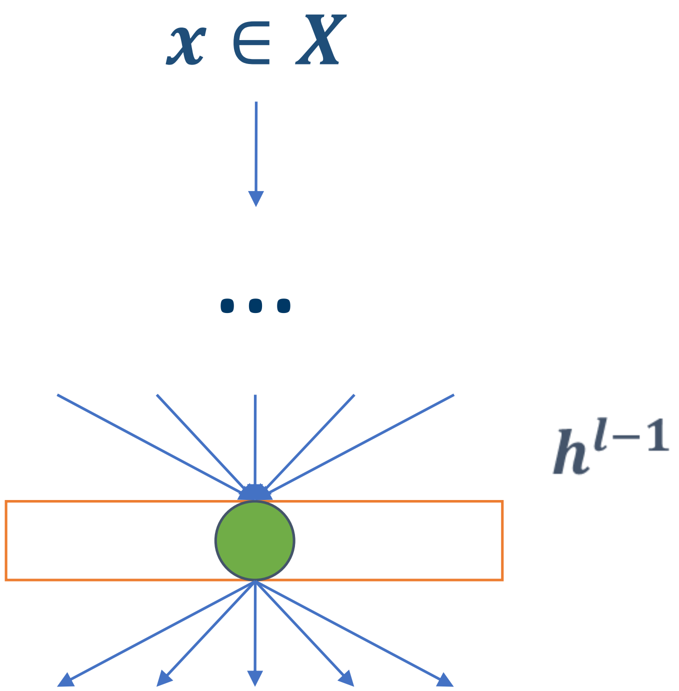
A neuron is **stably inactive** if it never produces a positive output



$$g_i^l := W^l h^{l-1} + b_i^l \leq 0 \quad \rightarrow h_i^l = 0$$

Neuron Stability with Respect to a Domain

A neuron is **stably active** if it <u>always</u> produces a nonnegative output



$$g_i^l := W^l h^{l-1} + b_i^l \geq 0 \quad \rightarrow h_i^l = g_i^l$$

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

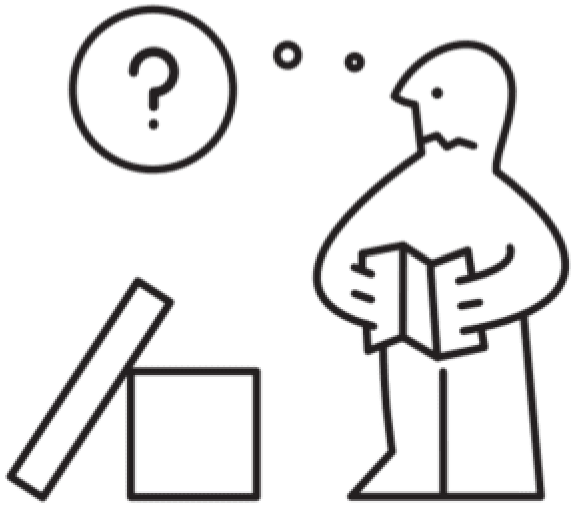
But… How!?

We can identify stable neurons with **optimization**!

If $\max_{x \in X} g_i^l := W_i^l h^{l-1} + b_i^l \leq 0$, then the neuron is **stably inactive**

If $\min_{x \in X} g_i^l := W_i^l h^{l-1} + b_i^l \geq 0$, then the neuron is **stably active**

We formulate a **Mixed-Integer Linear Program (MILP)**
to map inputs to outputs of a trained neural network



ikea.com

Mapping Inputs to Outputs

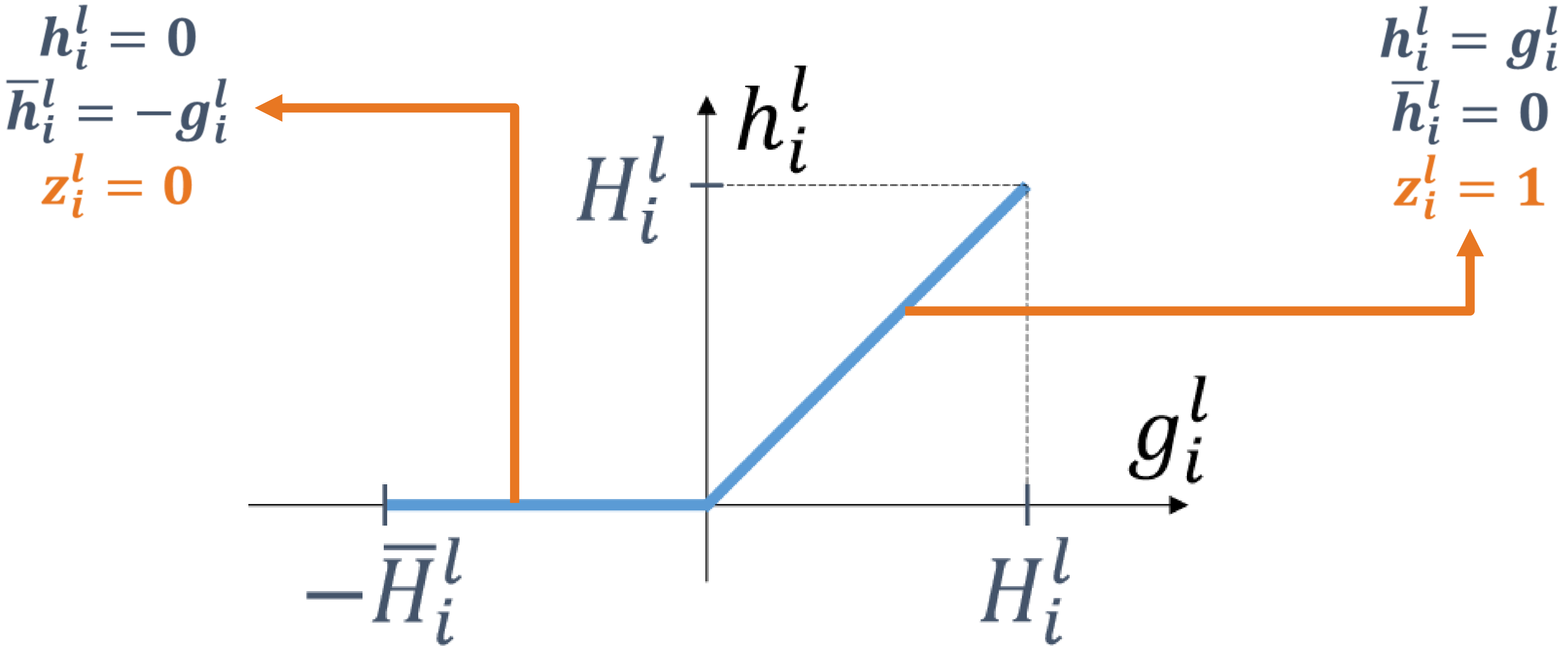
The following constraints represent a ReLU $\boldsymbol{i}$ in layer $\boldsymbol{l}$:

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

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

Mapping Inputs to Outputs

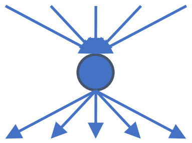
The binary variable changes the mapping according to unit activity



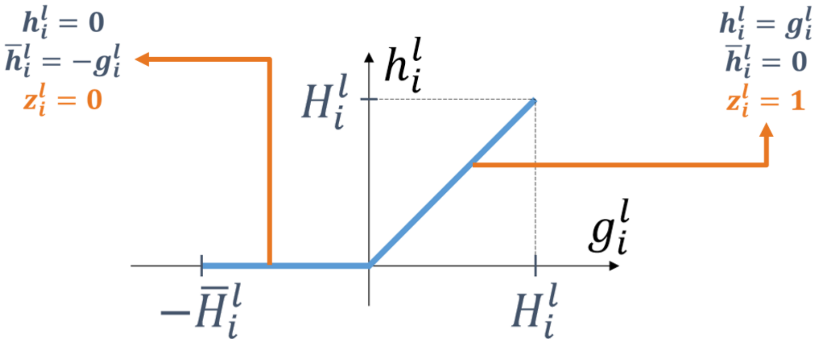
Wait...

If $\max\limits_{x \in X} g_i^l := W_i^l h^{l-1} + b_i^l \leq 0$,

If $\min\limits_{x \in X} g_i^l := W_i^l h^{l-1} + b_i^l \geq 0$,



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



https://giphy.com/gifs/Friends-friends-season-5-episode-106-TglfmNLgdMPE5NfLKS

Solving two MILPs per neuron is not that cheap!

Well…

Stopping with **negative upper bounds for max** or **positive lower bounds for min**, this is the **runtime to identify all stable neurons**:

Hidden Layers	Runtime (~ s)
2 x 25	30
2 x 50	100
2 x 100	400

https://giphy.com/gifs/friends-ross-geller-i-know-XZ0yPco3eynUAGU0i3

A Couple of Insights on the Compression Problem

1. **We are solving all these problems over the same feasible set**

   I.e., every network input is mapped to the corresponding output

2. **It is easy to certify that a neuron is not stable**

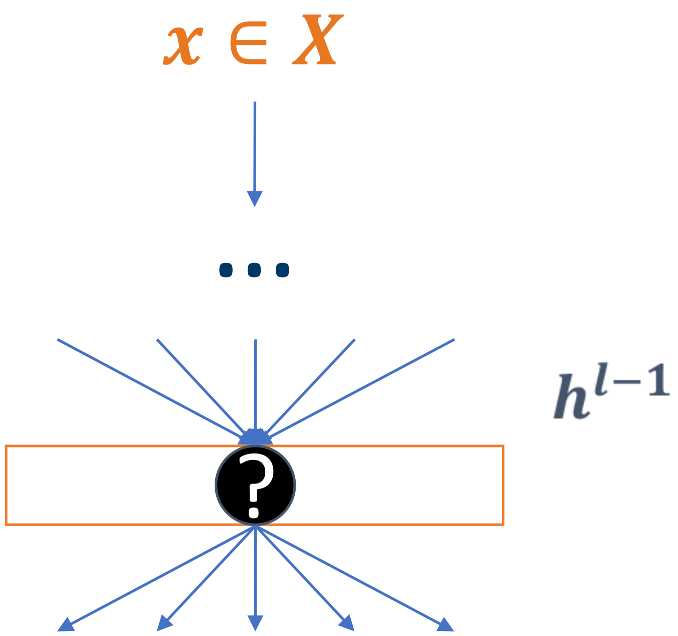
We just need two network inputs:

- One for which the neuron is active
- Another one for which the neuron is inactive

In fact, one network input may be used to certify multiple neurons

Rethinking the Optimization Problem

Find an input that certifies as many neurons of unknown status as possible



$g_i^l \coloneqq W^l h^{l-1} + b_i^l \geq 0$? → P: Neurons that have not been active yet

$g_i^l \coloneqq W^l h^{l-1} + b_i^l \leq 0$? → Q: Neurons that have not been inactive yet

Rethinking the Optimization Problem

Find an input that certifies as many neurons of unknown status as possible

- $\boldsymbol{P}$: Neurons that have not been active yet
- $\boldsymbol{Q}$: Neurons that have not been inactive yet
- Binary variables $\boldsymbol{p_i^l}$ and $\boldsymbol{q_i^l}$ for every neuron in $\boldsymbol{P}$ and $\boldsymbol{Q}$

$$\boldsymbol{c(P,Q) := \max_{x \in X} \sum_{(l,i) \in P} p_i^l + \sum_{(l,i) \in Q} q_i^l}$$

If $\boldsymbol{c(P,Q) = 0}$, then **every neuron in P is stably inactive** and **every neuron in Q is stably active**

If $\boldsymbol{c(P,Q) > 0}$, we are one step closer to identifying stable neurons

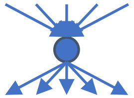
One Solve to Find Them All!

We only need to solve this problem to optimality if $\boldsymbol{c(P, Q) = 0}$:

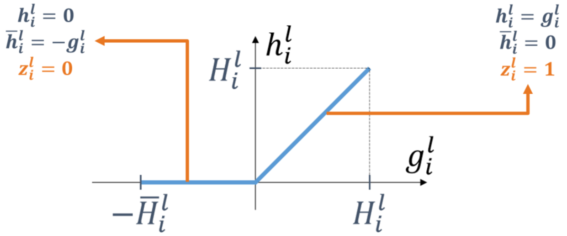
- Any solution with a positive value helps
- For every solution $(\overline{\boldsymbol{p}}, \overline{\boldsymbol{q}})$, fix $\boldsymbol{p_i^l = 0}$ if $\overline{\boldsymbol{p}}_i^l = \mathbf{1}$ (and likewise with $\boldsymbol{q}$)

Relax new binary variables by using the integrality of $\boldsymbol{z}$

$$\boldsymbol{c(P, Q) := \max_{x \in X} \sum_{(l,i) \in P} p_i^l + \sum_{(l,i) \in Q} q_i^l}$$



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



$$\boldsymbol{0 \leq p_i^l \leq z_i^l}$$

$$\boldsymbol{0 \leq q_i^l \leq 1 - z_i^l}$$

An Insight on the MILP Formulation

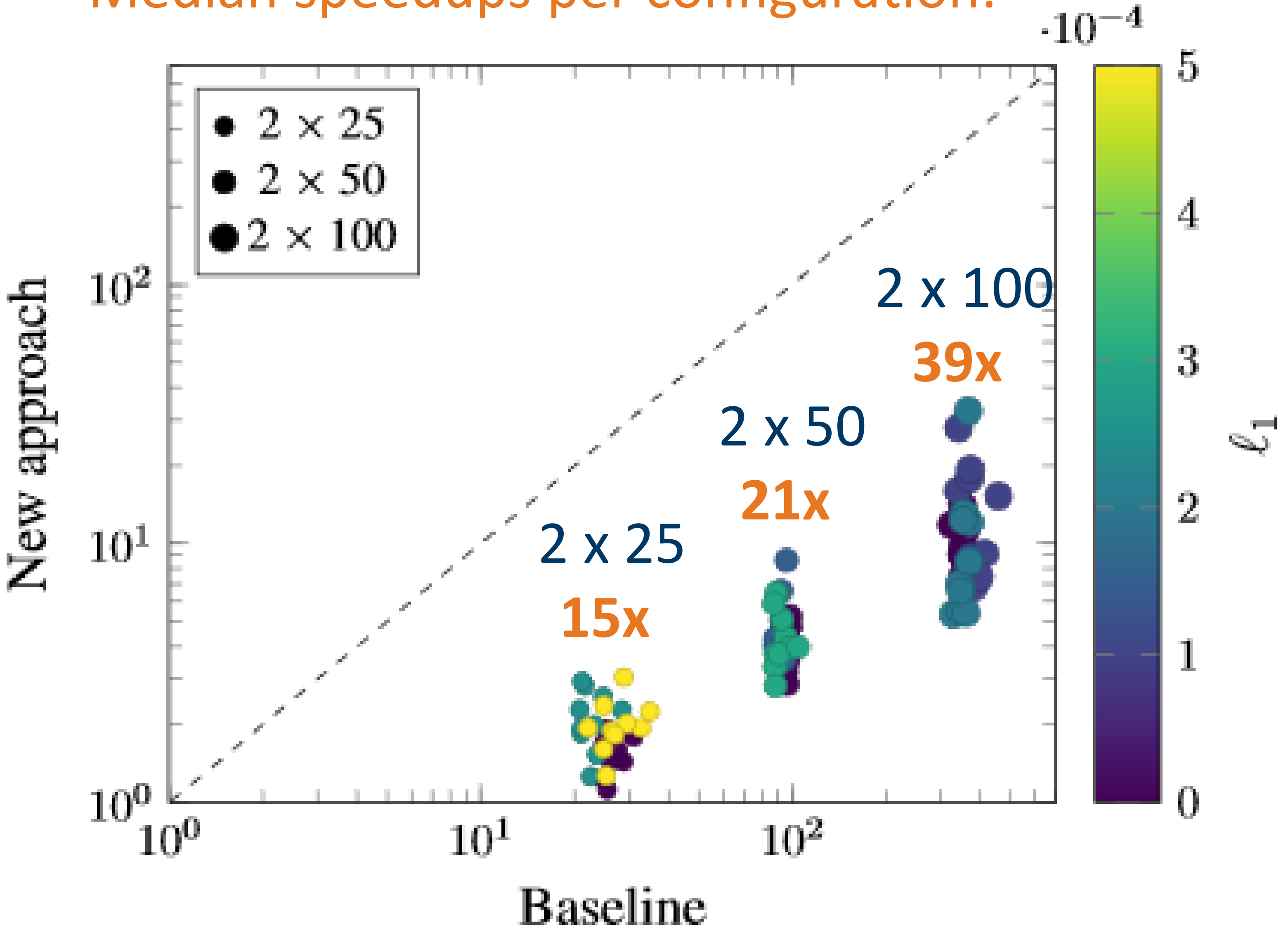
3. **Finding feasible solutions for these MILPs is easy**
   Fischetti & Jo (2018): Any valid input defines a feasible solution

Use the input $\tilde{\boldsymbol{x}}$ associated with solving the linear relaxation

- The restriction $\boldsymbol{x} = \tilde{\boldsymbol{x}}$ yields one feasible MILP solution
- This solution is somewhat guided by the objective function
- We can get one of those at every node of the branch-and-bound tree

How About the Runtimes Now?

Median speedups per configuration:



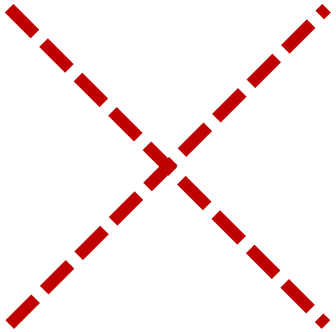
https://giphy.com/gifs/Friends-season-5-episode-7-friends-tv-iHskdY9SMLFZuQ2u5c

But how can we make these networks smaller?

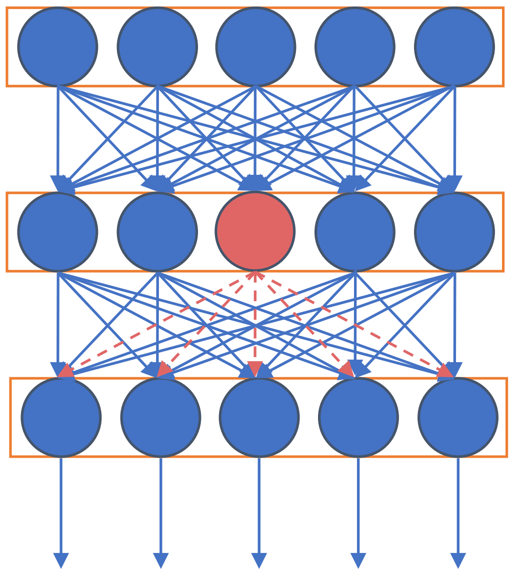
https://deadline.com/2020/02/honey-i-shrunk-the-kids-reboot-rick-moranis-1202858344/

Conditions for Exact Compression

What happens when a **neuron is stably inactive**?



$h_i^l = 0$



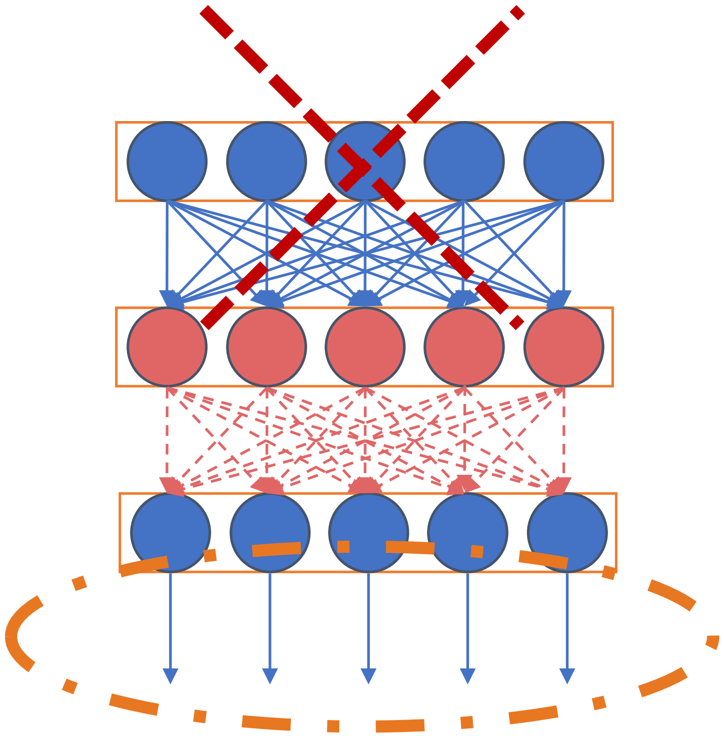
Since the output of the neuron is always zero,

we can easily remove it from the neural network

Conditions for Exact Compression

What happens when an entire **layer is stably inactive**?

$$h^l = 0$$



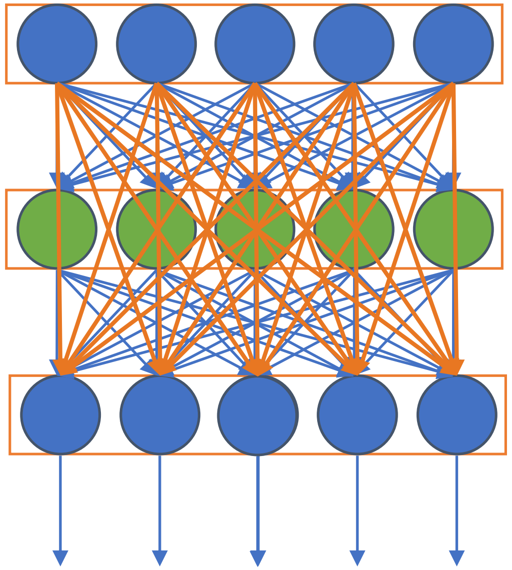
The network output is constant and defined by the parameters in subsequent layers

All hidden layers can be removed

Conditions for Exact Compression

What happens when an entire **layer is stably active**?

$$\boldsymbol{h}^l = \boldsymbol{W}^l\,\boldsymbol{h}^{l-1} + \boldsymbol{b}^l$$

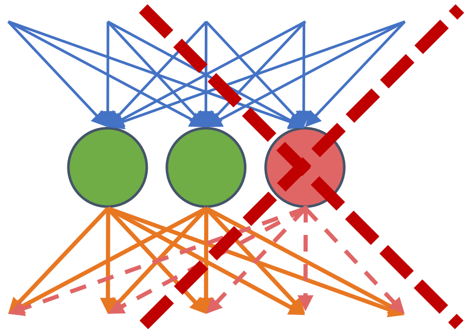


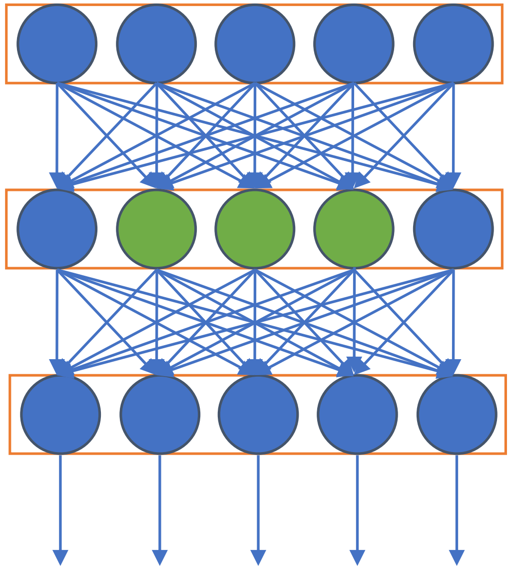
$$\boldsymbol{g}_j^{l+1} = \boldsymbol{W}^{l+1}\,(\boldsymbol{W}^l\boldsymbol{h}^{l-1} + \boldsymbol{b}^l) + \boldsymbol{b}_j^{l+1}$$

$$= (\boldsymbol{W}^{l+1}\,\boldsymbol{W}^l)\boldsymbol{h}^{l-1} + (\boldsymbol{W}^{l+1}\boldsymbol{b}^l + \boldsymbol{b}_j^{l+1})$$

Conditions for Exact Compression

What happens when some **neurons are stably active**?





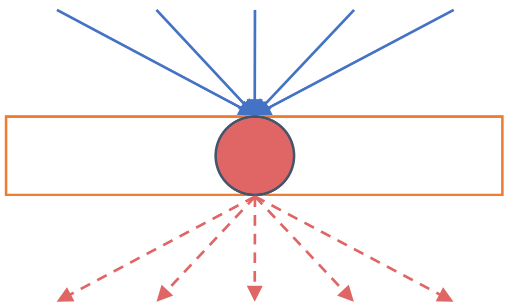
If $\mathbf{r} := rank(W_S^l) < |S|$, we can define the same affine transformation with r neurons

Are There Stable Neurons?

Tjeng, Xiao & Tedrake (2019): **Stability** can be induced with ℓ_1 **regularization**

$$\boldsymbol{W^l h^{l-1} + b_i^l}$$

$\downarrow$ −

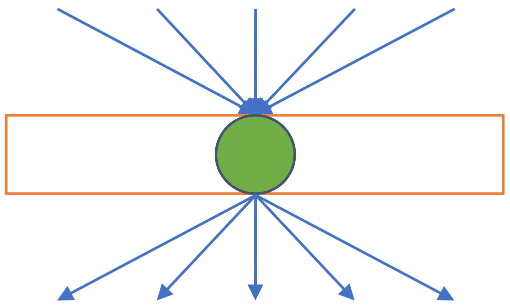


$\boldsymbol{h^{l-1}}$

$$\boldsymbol{g_i^l \leq 0}$$

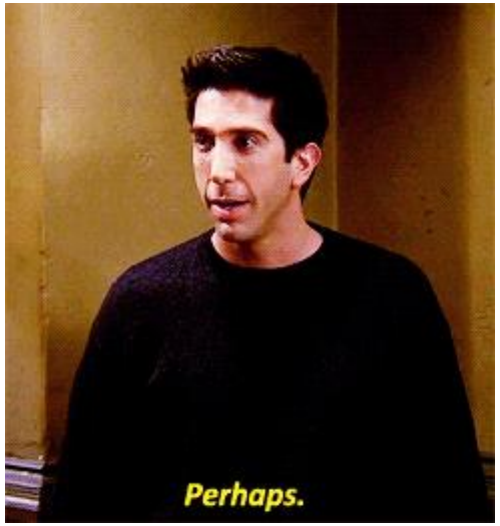
$$\boldsymbol{W^l h^{l-1} + b_i^l}$$

$\downarrow$ +



$\boldsymbol{h^{l-1}}$

$$\boldsymbol{g_i^l \geq 0}$$



https://wifflegif.com/tags/27060-ross-geller-gifs

Some (but not too much) training regularization: better accuracy

Neurons Removed in MNIST Classifiers

We chose $\ell_1 = \bar{\ell}$ to yield the same training accuracy as $\ell_1 = 0$

- $\ell_1 = \bar{\ell}/2$ has **better accuracy while being compressible**

Hidden Layers	$\ell_1 = 0$	Accuracy	$\ell_1 = \bar{\ell}/2$	Accuracy	$\ell_1 = \bar{\ell}$	Accuracy
2 x 100	0%	97.93%	**13%**	98.14%	23%	97.89%
3 x 100	0%	98.05%	**14%**	98.23%	26%	98.05%
4 x 100	0%	98.12%	**16%**	98.18%	25%	98.12%
5 x 100	0%	98.13%	**17%**	98.42%	27%	98.12%

Neurons Removed in MNIST Classifiers

We chose $\ell_1 = \bar{\ell}$ to yield the same training accuracy as $\ell_1 = 0$

- $\ell_1 = \bar{\ell}/2$ has **better accuracy while being compressible**

Hidden Layers	$\ell_1 = 0$	Accuracy	$\ell_1 = \bar{\ell}/2$	Accuracy	$\ell_1 = \bar{\ell}$	Accuracy
2 x 100	0%	97.93%	**13%**	98.14%	23%	97.89%
2 x 200	0%	98.17%	**13%**	98.33%	26%	98.17%
2 x 400	0%	98.25%	**8%**	98.35%	24%	98.24%
2 x 800	0%	98.28%	**–**	–	22%	98.29%

Runtime in MNIST Classifiers

We chose $\ell_1 = \bar{\ell}$ to yield the same training accuracy as $\ell_1 = 0$

- $\ell_1 = \bar{\ell}/2$ has **better accuracy while being compressible**

Hidden Layers	$\ell_1 = 0$	Accuracy	$\ell_1 = \bar{\ell}/2$	Accuracy	$\ell_1 = \bar{\ell}$	Accuracy
2 x 100	10 s	97.93%	**14 s**	98.14%	11 s	97.89%
3 x 100	54 s	98.05%	**80 s**	98.23%	50 s	98.05%
4 x 100	220 s	98.12%	**2,000 s**	98.18%	180 s	98.12%
5 x 100	840 s	98.13%	**3,370 s**	98.42%	510 s	98.12%

Runtime in MNIST Classifiers

We chose $\ell_1 = \bar{\ell}$ to yield the same training accuracy as $\ell_1 = 0$

- $\ell_1 = \bar{\ell}/2$ has **better accuracy while being compressible**

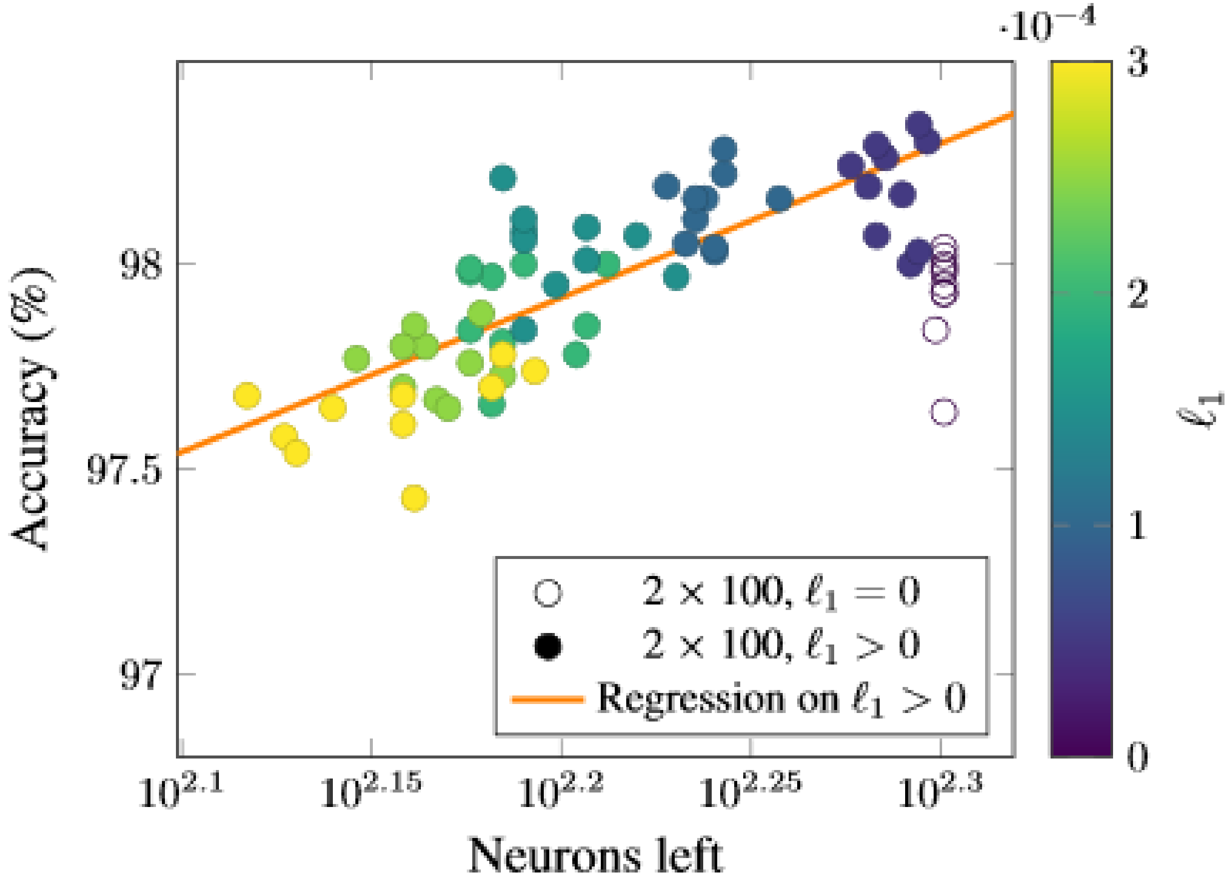
Hidden Layers	$\ell_1 = 0$	Accuracy	$\ell_1 = \bar{\ell}/2$	Accuracy	$\ell_1 = \bar{\ell}$	Accuracy
2 x 100	10 s	97.93%	**14 s**	98.14%	11 s	97.89%
2 x 200	45 s	98.17%	**120 s**	98.33%	26 s	98.17%
2 x 400	400 s	98.25%	**1,800 s**	98.35%	130 s	98.24%
2 x 800	3,500 s	98.28%	**–**	–	1,000 s	98.29%

The Impact of ℓ_1 Regularization on Runtimes

		Harder	Easier
Hidden Layers	$\ell_1 = 0$	$\ell_1 = \bar{\ell}/2$	$\ell_1 = \bar{\ell}$
2 x 100	10 s	**14 s**	11 s
2 x 200	45 s	**120 s**	26 s
2 x 400	400 s	**1,800 s**	130 s
2 x 800	3,500 s	**–**	1,000 s

The Impact of ℓ_1 Regularization on Compression

When regularization is used, **compression is related to accuracy** ($R^2 = 0.64$)



Neurons Removed in MNIST Autoencoders

With <u>way more</u> regularization:

- **First two hidden layers fold** before loss doubles

Hidden Layers	$\ell_1 = 0$	Loss	$\ell_1 = 0.00002$	Loss	$\ell_1 = 0.0002$	Loss
100 | 10 | 100	0%	0.045	13%	0.047	**95%**	0.077
100 | 25 | 100	0%	0.035	14%	0.047	**90%**	0.076
100 | 50 | 100	0%	0.031	17%	0.048	**90%**	0.071

Neurons Removed in MNIST Autoencoders

With <u>way more</u> regularization:

- **First two hidden layers fold** before loss doubles

Hidden Layers	$\ell_1 = 0$	Loss	$\ell_1 = 0.00002$	Loss	$\ell_1 = 0.0002$	Loss
50 | 10 | 50	0%	0.047	14%	0.051	**89%**	0.081
100 | 10 | 100	0%	0.045	13%	0.047	**95%**	0.077
200 | 10 | 200	0%	0.041	14%	0.043	**95%**	0.076
400 | 10 | 400	0%	0.040	15%	0.040	**89%**	0.073

Runtime in MNIST Autoencoders

With <u>way more</u> regularization:

- **First two hidden layers fold** before loss doubles
- **Runtimes drop abruptly**

Hidden Layers	$\ell_1 = 0$	Loss	$\ell_1 = 0.00002$	Loss	$\ell_1 = 0.0002$	Loss
100 | 10 | 100	130 s	0.045	120 s	0.047	**3 s**	0.077
100 | 25 | 100	500 s	0.035	800 s	0.047	**3 s**	0.076
100 | 50 | 100	230 s	0.031	600 s	0.048	**3 s**	0.071

Neurons Removed in MNIST Autoencoders

With <u>way more</u> regularization:

- **First two hidden layers fold** before loss doubles
- **Runtimes drop abruptly**

Easiest ⬆

Hidden Layers	$\ell_1 = 0$ (Loss)	$\ell_1 = 0.00002$ (Loss)	$\ell_1 = 0.0002$ (Loss)
50 | 10 | 50	33 s 0.047	50 s 0.051	**1 s** 0.081
100 | 10 | 100	130 s 0.045	120 s 0.047	**3 s** 0.077
200 | 10 | 200	1000 s 0.041	700 s 0.043	**5 s** 0.076
400 | 10 | 400	2700 s 0.040	1300 s 0.040	**10 s** 0.073

Fim

We presented a general-purpose exact compression method which:

- **Scales** to networks large enough **for practical use**
- **Runs faster than training** the network

In a nutshell:

- **What?** Remove and merge neurons, fold layers, or collapse the network
- **When?** Neural networks are trained with ℓ_1 regularization
- **Why?** They have stable neurons with linear behavior
- **How?** Solving an optimization problem

MIP is there for you!

https://giphy.com/gifs/Friends-season-9-episode-19-friends-tv-S6qkS0ETvel6EZat45

References

Serra, Kumar, Ramalingam: "**Scaling Up Exact Neural Network Compression by ReLU Stability**" – **arXiv preprint**

https://arxiv.org/abs/2102.07804

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

https://arxiv.org/abs/2001.00218

Related work on linear regions:

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

https://arxiv.org/abs/1810.03370

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

https://arxiv.org/abs/1711.02114

Serra: "**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/