ICIAP 2021

PRUNING DEEP NEURAL NETWORKS: TOWARDS EFFICIENT MODELS ON THE EDGE

T7 – ICIAP 2021

Enzo Tartaglione

Andrea Bragagnolo

Attilio Fiandrotti

Iacopo Colonnelli

Marco Grangetto

UNIVERSITÀ DEGLI STUDI DI TORINO

TELECOM Paris

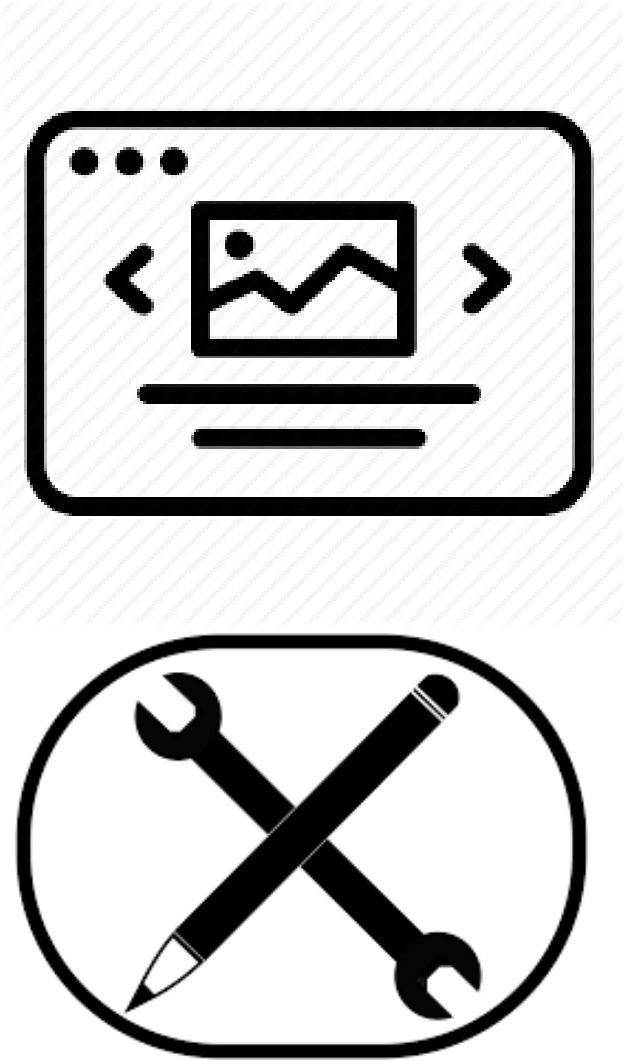
IP PARIS

Powered by HPC4AI

Material & practical infos

- Both these slides AND the Jupyter notebook for the practical session are downloadable at **https://github.com/EIDOSlab/ICIAP2021-T7-pruning**

- At the beginning of the practical session, some credentials will be distributed in order to access to HPC4AI computing resources.



Agenda

- **Part 1**: theory and background (14:00-15:30)
  - Get familiarity with fundamentals of pruning
  - Learn some of the most common techniques
  - Foresight the next challenges and future research and industrial objectives

  Coffee break (15:30-16:00)

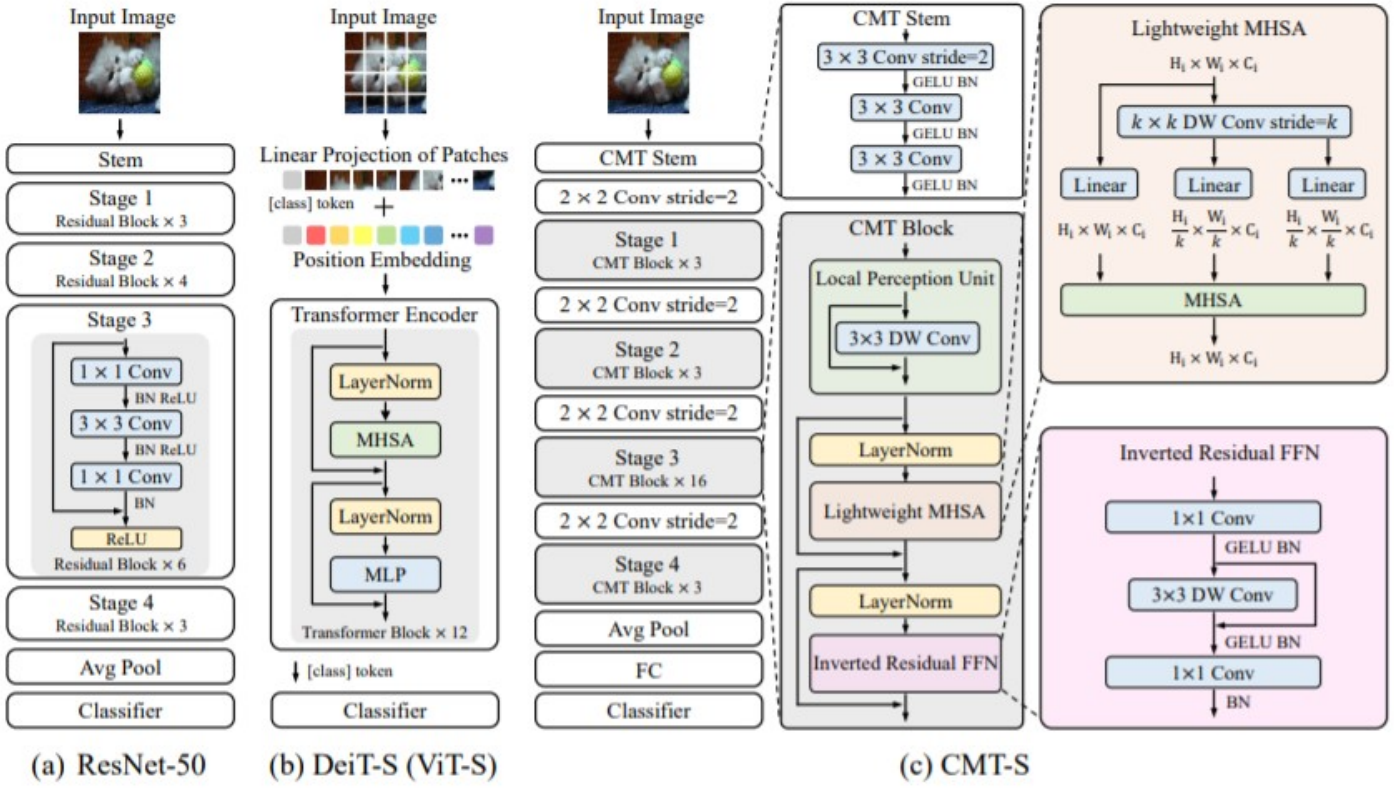
- **Part 2**: hands-on-code session (16:00-18:00)
  - Learn about what we have at home: HPC4AI- an italian supercomputing infrastructure
  - Experiment pruning by yourself! (a .ipynb notebook is provided)

Outline for PART 1

- Why pruning? Motivations and aims
- Voices from the past: pruning in the ‘90s
  - Skeletonization [Mozer and Smolensky, NIPS 1989]
  - Optimal brain damage [Le Cun et al., NIPS 1990]
- A revival of pruning (2015 & beyond)
  - Learning both weights and connections [Han et al., NIPS 2015]
  - Variational dropout sparsifies deep neural networks [Molchanlov et al., ICML 2017]
  - Learning sparse networks using targeted dropout [Gomez et al., NeurIPS 2019]
  - Learning sparse neural networks via sensitivity-driven regularization [Tartaglione et al., NeurIPS 2018]
  - Learning sparse neural networks through L0 regularization [Louizos et al., ICLR 2018]
- The lottery ticket hypothesis [Frankle and Carbin, ICLR 2019]
  - SNIP: Single-shot network pruning based on connection sensitivity [Lee et al., ICLR 2019]
  - Can we prune at initialization?
- Structured vs unstructured pruning

Making deep models efficient

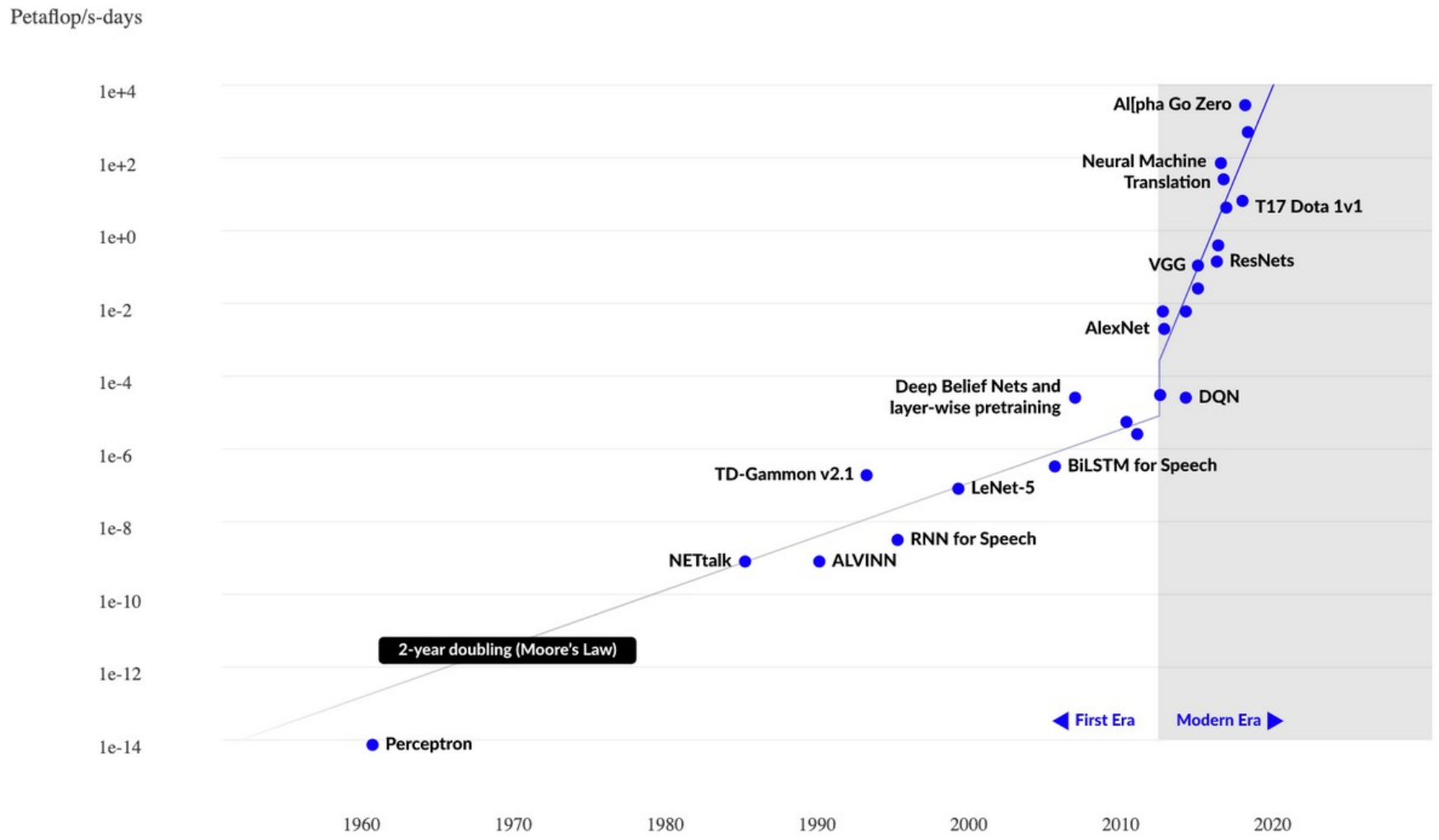
Deep networks



- High number of hidden layers
- More complex classification tasks (ImageNet)
- Use of convolutional layers, pooling layers, very large fully-connected layers
- Very high number of parameters (hundreds of millions and even more…)
- Is it possible to boost the performance making the ANN robust to noise?

Image from CMT: Convolutional Neural Networks Meet Vision Transformer Jianyuan Guo, Kai Han, Han Wu, Chang Xu, Yehui Tang, Chunjing, Yunhe Wang, arXiv 2021

Deeper means more complex

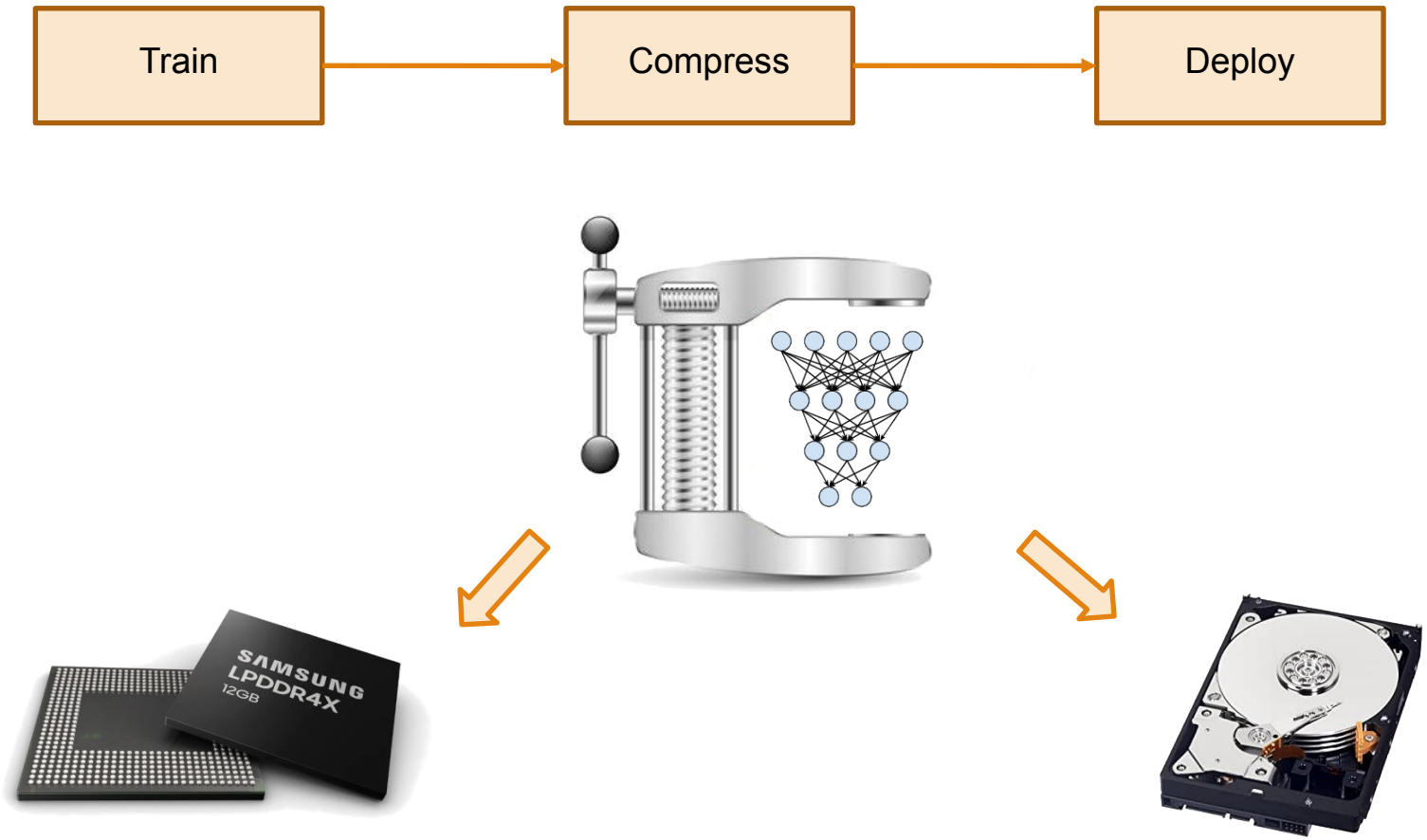


Deep models outside the datacenters

- Mobile phones, DL accelerators, FPGAs have little memory
- Also the bandwidth available on the channel is limited

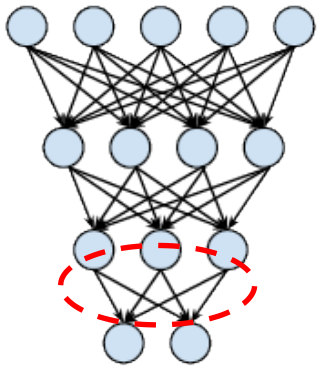
Can we make deep models compact?

Compressing deep models

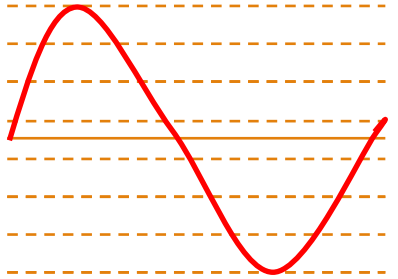


Quantization

Quantization



Neural network compression

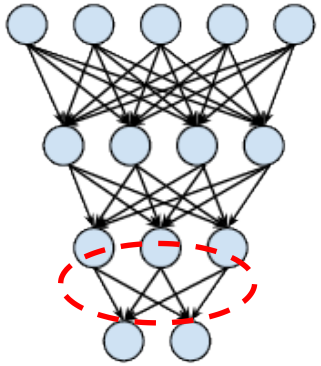


Quantization (w. entropy coding)

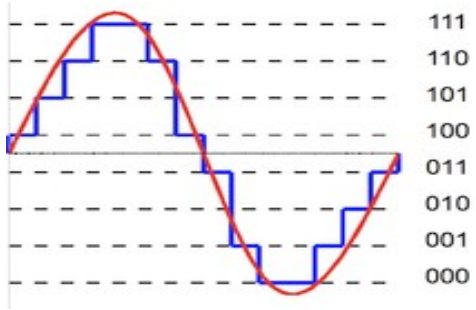
$$\begin{bmatrix} -2.3 & -.5 & -2.6 \\ 1.12 & 3 & .6 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

Floats weights matrix

Quantization



Neural network compression



Quantization
(w. entropy coding)

$$\begin{bmatrix} -2 & 0 & -3 \\ 1 & 3 & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

Quantized weights matrix

Quantization for deep learning

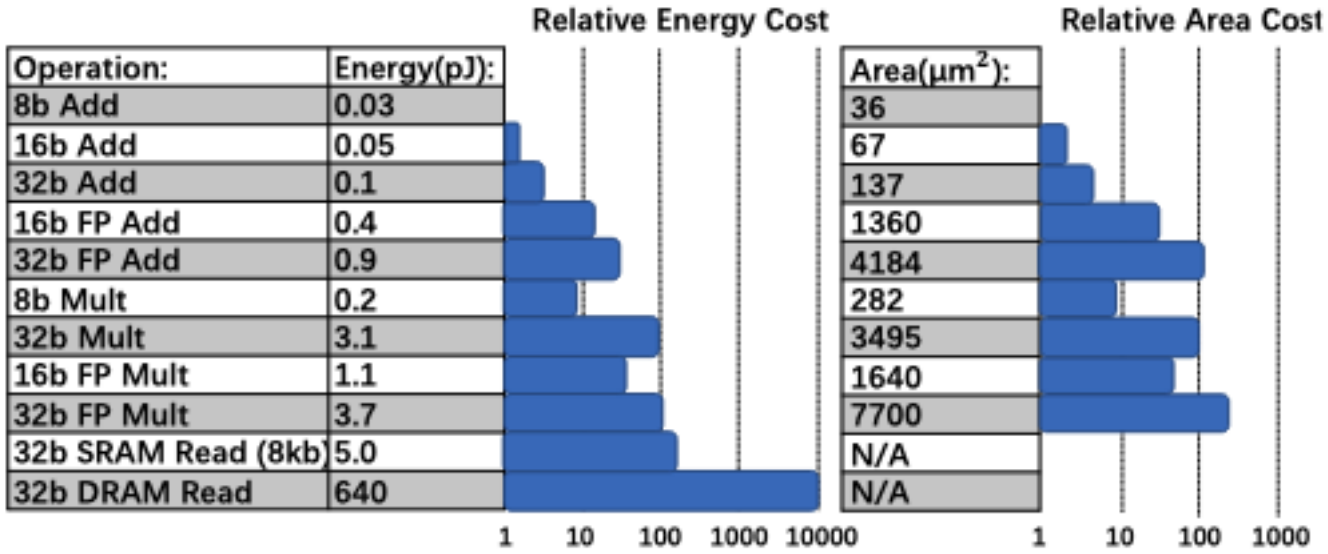
- Quantization for deep learning is the process of approximating a neural network that uses floating-point numbers by a neural network of low bit width numbers.
- This dramatically reduces both the memory requirement and computational cost of using neural networks.

- This dramatically reduces both the memory requirement and computational cost of using neural networks (and this is one approach we are going to deploy!), however, MANY techniques to take this into account AT TRAINING TIME are proposed every day!

If interested, here there are some (old yet valid!) references for quantization-aware training:

F. Li and B. Liu, "Ternary weight networks," CoRR, vol. abs/1605.04711, 2016. [Online]. Available: http://arxiv.org/abs/1605.04711
M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," CoRR, vol. abs/1602.02830, 2016. [Online]. Available: http://arxiv.org/abs/1602.02830
M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," CoRR, vol. abs/1603.05279, 2016. [Online]. Available: http://arxiv.org/abs/1603.05279
M. Ghasemzadeh, M. Samragh, and F. Koushanfar, "Resbinnet: Residual binary neural network," CoRR, vol. abs/1711.01243, 2017. [Online]. Available: http://arxiv.org/abs/1711.01243
S. Zhou, Z. Ni, X. Zhou, H. Wen, Y. Wu, and Y. Zou, "Dorefa-
net: Training low bitwidth convolutional neural networks with low bitwidth gradients," CoRR, vol. abs/1606.06160, 2016. [Online]. Available: http://arxiv.org/abs/1606.06160
Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," CoRR, vol. abs/1702.00953, 2017. [Online]. Available: http://arxiv.org/abs/1702.00953
D. Miyashita, E. H. Lee, and B. Murmann, "Convolutional neural networks using logarithmic data representation," CoRR, vol. abs/1603.01025, 2016. [Online]. Available: http://arxiv.org/abs/1603.01025
Y. Ding, J. Liu, and Y. Shi, "On the universal approximability of quantized relu neural networks," CoRR, vol. abs/1802.03646, 2018. [Online]. Available: http://arxiv.org/abs/1802.03646

From FP units to INT (quantized)



Reference: 45nm technology

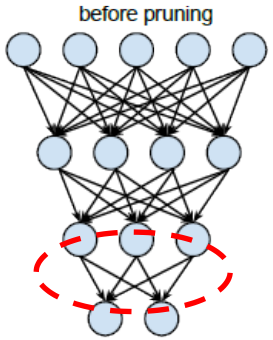
Gholami, Amir, et al. "A survey of quantization methods for efficient neural network inference." *arXiv preprint arXiv:2103.13630* (2021).

What is Pruning?

Sparsification / pruning

Neural network compression

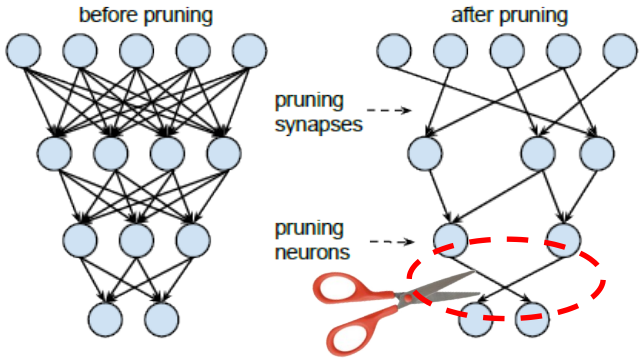
Sparsification via pruning



$$\begin{bmatrix} -2.3 & -.5 & -2.5 \\ 1.12 & 3 & .5 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

Sparsification / pruning

Neural network compression



Sparsification via pruning

$$\begin{bmatrix} 0 & 0 & -2.5 \\ 1.12 & 0 & 0 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

Sparse weights matrix

Pruning: a definition

- With pruning we refer to the process of removing parameters (synapses) or entire units from the deep learning model.

Pruning: a definition

- With pruning we refer to the process of removing parameters (synapses) or entire units from the deep learning model.
- Pruning relates with sparsification: the weight matrix (representing a layer) becomes, indeed, sparse!

Pruning: a definition

- With pruning we refer to the process of removing parameters (synapses) or entire units from the deep learning model.
- Pruning relates with sparsification: the weight matrix (representing a layer) becomes, indeed, sparse!
- The removed parameters (if they are removed in an “unstructured” way) still need to be encoded, with a “0”.
- Hence, in general, the position with a “missing” connection still needs to be encoded in some way (while using general frameworks), producing some representation overhead.
- Do we have advantages with a pruned architecture?

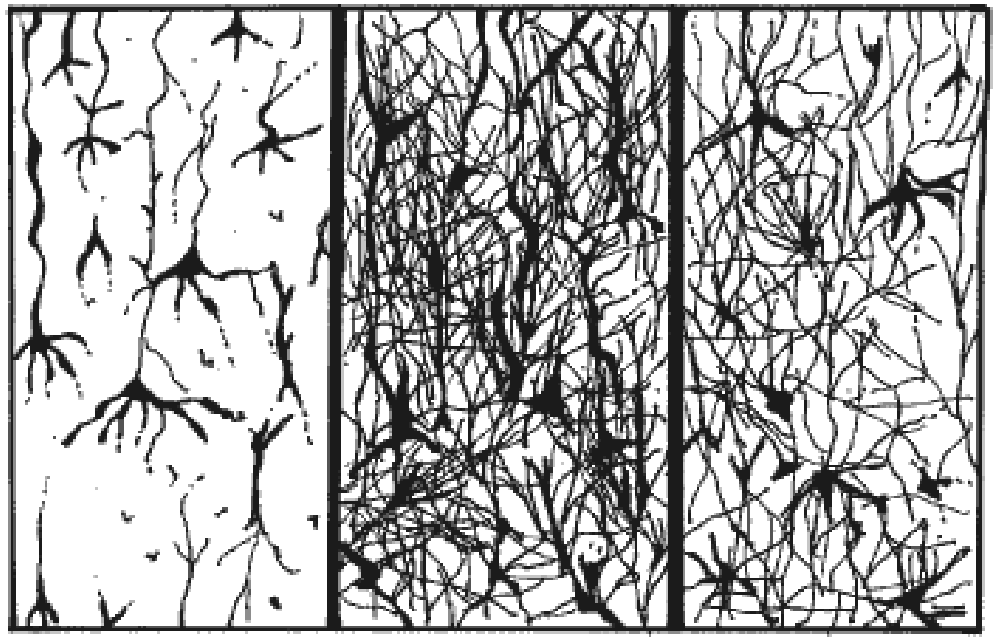
Pruning: a definition

- With pruning we refer to the process of removing parameters (synapses) or entire units from the deep learning model.
- Pruning relates with sparsification: the weight matrix (representing a layer) becomes, indeed, sparse!
- The removed parameters (if they are removed in an “unstructured” way) still need to be encoded, with a “0”.
- Hence, in general, the position with a “missing” connection still needs to be encoded in some way (while using general frameworks), producing some representation overhead.
- Do we have advantages with a pruned architecture?
  - The number of parameters is reduced. For special designs (like ASICs) the gain is real.

Pruning: a definition

- With pruning we refer to the process of removing parameters (synapses) or entire units from the deep learning model.
- Pruning relates with sparsification: the weight matrix (representing a layer) becomes, indeed, sparse!
- The removed parameters (if they are removed in an “unstructured” way) still need to be encoded, with a “0”.
- Hence, in general, the position with a “missing” connection still needs to be encoded in some way (while using general frameworks), producing some representation overhead.
- Do we have advantages with a pruned architecture?
  - The number of parameters is reduced. For special designs (like ASICs) the gain is real.
  - If they are removed in a “structured” way (entire blocks), there is no representation overhead, and the gain is real even in general frameworks!

Biological plausibility



[…] Synaptic density increased during infancy, reaching a maximum at age 1–2 years which was about 50% above the adult mean. The decline in synaptic density observed between ages 2–16 years was accompanied by a slight decrease in neuronal density. Human cerebral cortex is one of a number of neuronal systems in which loss of neurons and synapses appears to occur as a late developmental event.

Huttenlocher, Peter R. "Synaptic density in human frontal cortex-developmental changes and effects of aging." *Brain Res* 163, no. 2 (1979): 195-205.

Voices from the past… pruning in the ‘90s

Outline for PART 1

- Why pruning? Motivations and aims
- Voices from the past: pruning in the ‘90s
  - Skeletonization [Mozer and Smolensky, NIPS 1989]
  - Optimal brain damage [Le Cun et al., NIPS 1990]
- A revival of pruning (2015 & beyond)
  - Learning both weights and connections [Han et al., NIPS 2015]
  - Variational dropout sparsifies deep neural networks [Molchanlov et al., ICML 2017]
  - Learning sparse networks using targeted dropout [Gomez et al., NeurIPS 2019]
  - Learning sparse neural networks via sensitivity-driven regularization [Tartaglione et al., NeurIPS 2018]
  - Learning sparse neural networks through L0 regularization [Louizos et al., ICLR 2018]
- The lottery ticket hypothesis [Frankle and Carbin, ICLR 2019]
  - SNIP: Single-shot network pruning based on connection sensitivity [Lee et al., ICLR 2019]
  - Can we prune at initialization?
- Structured vs unstructured pruning

Skeletonization [Mozer and Smolensky, NIPS 1989]

The idea of the work is very simple…

1) Iteratively train the network to a certain performance criterion

Skeletonization [Mozer and Smolensky, NIPS 1989]

The idea of the work is very simple…

1) Iteratively train the network to a certain performance criterion

2) Compute a measure of relevance

$$\rho_i = E_{\text{without unit } i} - E_{\text{with unit } i}$$

Skeletonization [Mozer and Smolensky, NIPS 1989]

The idea of the work is very simple…

1) Iteratively train the network to a certain performance criterion

2) Compute a measure of relevance

$$\rho_i = E_{\text{without unit } i} - E_{\text{with unit } i}$$

3) Trim the least relevant units

Skeletonization [Mozer and Smolensky, NIPS 1989]

The idea of the work is very simple…

1) Iteratively train the network to a certain performance criterion

2) Compute a measure of relevance

$$\rho_i = E_{\text{without unit } i} - E_{\text{with unit } i}$$

3) Trim the least relevant units

THIS SCHEME WILL BE VERY COMMON IN MOST OF THE PRUNING APPROACHES!

Skeletonization [Mozer and Smolensky, NIPS 1989]

- Idea: how well does the network when the i-th unit is removed?

Skeletonization [Mozer and Smolensky, NIPS 1989]

- Idea: how well does the network when the i-th unit is removed?
- However many problems might rise: extremely high computational complexity, non-fixed training set etc…
- Definition of an attentional strength coefficient α which can be learned through back-propagation

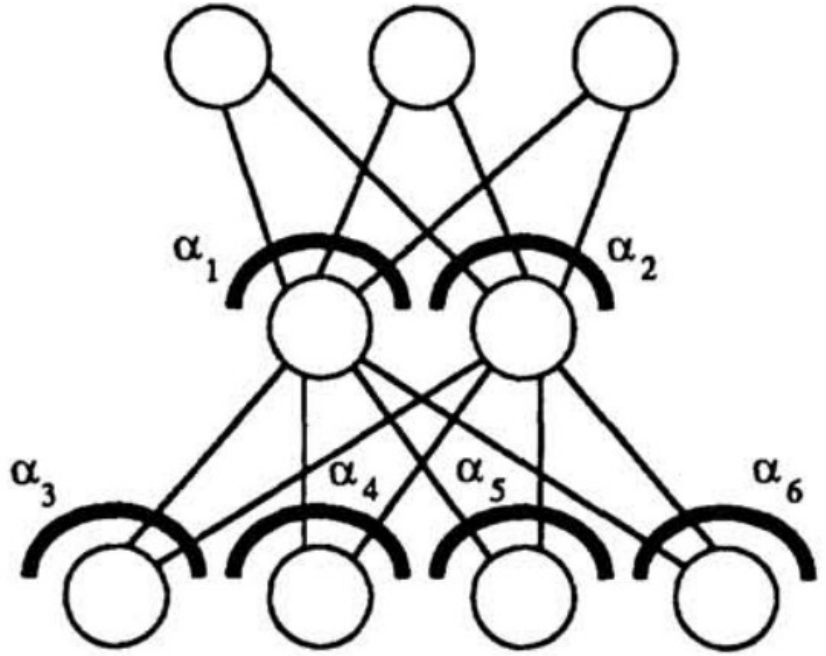
$$\rho_i = E_{\alpha_i=0} - E_{\alpha_i=1}$$

$$\lim_{\gamma \to 1} \frac{E_{\alpha_i=\gamma} - E_{\alpha_i=1}}{\gamma - 1} = \left. \frac{\partial E}{\partial \alpha_i} \right|_{\alpha_i=1}$$

$$\frac{E_{\alpha_i=0} - E_{\alpha_i=1}}{-1} \approx \left. \frac{\partial E}{\partial \alpha_i} \right|_{\alpha_i=1}$$

$$-\rho_i \approx \left. \frac{\partial E}{\partial \alpha_i} \right|_{\alpha_i=1}$$

Skeletonization [Mozer and Smolensky, NIPS 1989]



- Problem: this derivative fluctuates a lot during training, and exponentially-decaying time average can relieve the problem.

Outline for PART 1

- Why pruning? Motivations and aims
- Voices from the past: pruning in the ‘90s
  - Skeletonization [Mozer and Smolensky, NIPS 1989]
  - Optimal brain damage [Le Cun et al., NIPS 1990]
- A revival of pruning (2015 & beyond)
  - Learning both weights and connections [Han et al., NIPS 2015]
  - Variational dropout sparsifies deep neural networks [Molchanlov et al., ICML 2017]
  - Learning sparse networks using targeted dropout [Gomez et al., NeurIPS 2019]
  - Learning sparse neural networks via sensitivity-driven regularization [Tartaglione et al., NeurIPS 2018]
  - Learning sparse neural networks through L0 regularization [Louizos et al., ICLR 2018]
- The lottery ticket hypothesis [Frankle and Carbin, ICLR 2019]
  - SNIP: Single-shot network pruning based on connection sensitivity [Lee et al., ICLR 2019]
  - Can we prune at initialization?
- Structured vs unstructured pruning

Optimal brain damage [Le Cun et al., NIPS 1990]

- Use of second-order derivative information to find a trade-off between:
  - complexity;
  - training error.

- The focus here switches from units to single parameters.

Optimal brain damage [Le Cun et al., NIPS 1990]

- Let the error function being approximated through Taylor expansion
- A perturbation over the parameter vector will produce a perturbation on the error as

$$\delta E = \sum_i g_i \delta u_i + \frac{1}{2} \sum_i h_{ii} \delta u_i^2 + \frac{1}{2} \sum_{i \neq j} h_{ij} \delta u_i \delta u_j + O(||\delta \vec{u}||^3)$$

$$g_i = \frac{\partial E}{\partial u_i} \qquad h_{ij} = \frac{\partial^2 E}{\partial u_i \partial u_j}$$

Optimal brain damage [Le Cun et al., NIPS 1990]

- Let the error function being approximated through Taylor expansion
- A perturbation over the parameter vector will produce a perturbation on the error as

$$\delta E = \sum_i g_i \delta u_i + \frac{1}{2} \sum_i h_{ii} \delta u_i^2 + \frac{1}{2} \sum_{i \neq j} h_{ij} \delta u_i \delta u_j + O(||\delta \vec{u}||^3)$$

$$g_i = \frac{\partial E}{\partial u_i} \qquad h_{ij} = \frac{\partial^2 E}{\partial u_i \partial u_j}$$

- The objective here is to find the largest subset of parameters whose pruning will cause the least increase of E

Optimal brain damage [Le Cun et al., NIPS 1990]

- Let the error function being approximated through Taylor expansion
- A perturbation over the parameter vector will produce a perturbation on the error as

$$\delta E = \sum_i g_i \delta u_i + \frac{1}{2} \sum_i h_{ii} \delta u_i^2 + \frac{1}{2} \sum_{i \neq j} h_{ij} \delta u_i \delta u_j + O(||\delta \vec{u}||^3)$$

$$g_i = \frac{\partial E}{\partial u_i} \qquad h_{ij} = \frac{\partial^2 E}{\partial u_i \partial u_j}$$

- The objective here is to find the largest subset of parameters whose pruning will cause the least increase of E
- Intractable (for 2.6k parameters network, the Hessian would be 6.5 x 10^6 – imagine with a more recent model!)

Optimal brain damage [Le Cun et al., NIPS 1990]

- We can just work with the diagonal (deleting one parameter will not cause impact on the others – which we know in general it is not true).

$$\delta E = \sum_i g_i \delta u_i + \frac{1}{2} \sum_i h_{ii} \delta u_i^2 + \frac{1}{2} \sum_{i \neq j} h_{ij} \delta u_i \delta u_j + O(||\mathbf{\mathcal{U}}||^3)$$

Optimal brain damage [Le Cun et al., NIPS 1990]

- We can just work with the diagonal (deleting one parameter will not cause impact on the others – which we know in general it is not true).

$$\delta E = \sum_i g_i \delta u_i + \frac{1}{2} \sum_i h_{ii} \delta u_i^2 + \frac{1}{2} \sum_{i \neq j} h_{ij} \delta u_i \delta u_j + O(\|\delta U\|^3)$$

ZERO

Optimal brain damage [Le Cun et al., NIPS 1990]

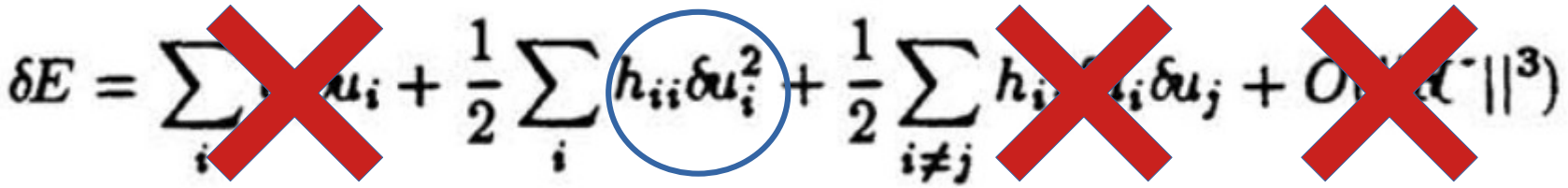
- We can just work with the diagonal (deleting one parameter will not cause impact on the others – which we know in general it is not true).

$$\delta E = \sum_i g_i \delta u_i + \frac{1}{2} \sum_i h_{ii} \delta u_i^2 + \frac{1}{2} \sum_{i \neq j} h_{ij} \delta u_i \delta u_j + O(\|\delta \mathbf{u}\|^3)$$

NEGLECTED

Optimal brain damage [Le Cun et al., NIPS 1990]

- We can just work with the diagonal (deleting one parameter will not cause impact on the others – which we know in general it is not true).



SALIENCY

- We ignore the first term as we assume we are in a minimum; hence, the gradient values zero.

Optimal brain damage [Le Cun et al., NIPS 1990]

- Assumption: the model has been already trained and we are in a local minimum
- All the h_ii terms are non-negative, hence every perturbation will cause the error to go up or stay the same

- The algorithm is composed of the following steps

  1. Choose a reasonable network architecture

Optimal brain damage [Le Cun et al., NIPS 1990]

- Assumption: the model has been already trained and we are in a local minimum
- All the h_ii terms are non-negative, hence every perturbation will cause the error to go up or stay the same

- The algorithm is composed of the following steps

  1. Choose a reasonable network architecture
  2. Train the network until a reasonable solution is obtained (train until convergency)

Optimal brain damage [Le Cun et al., NIPS 1990]

- Assumption: the model has been already trained and we are in a local minimum
- All the h_ii terms are non-negative, hence every perturbation will cause the error to go up or stay the same

- The algorithm is composed of the following steps

1. Choose a reasonable network architecture
2. Train the network until a reasonable solution is obtained (train until convergency)
3. Compute the second derivatives hu for each parameter
4. Compute the saliencies for each parameter: Sk = h_u/2

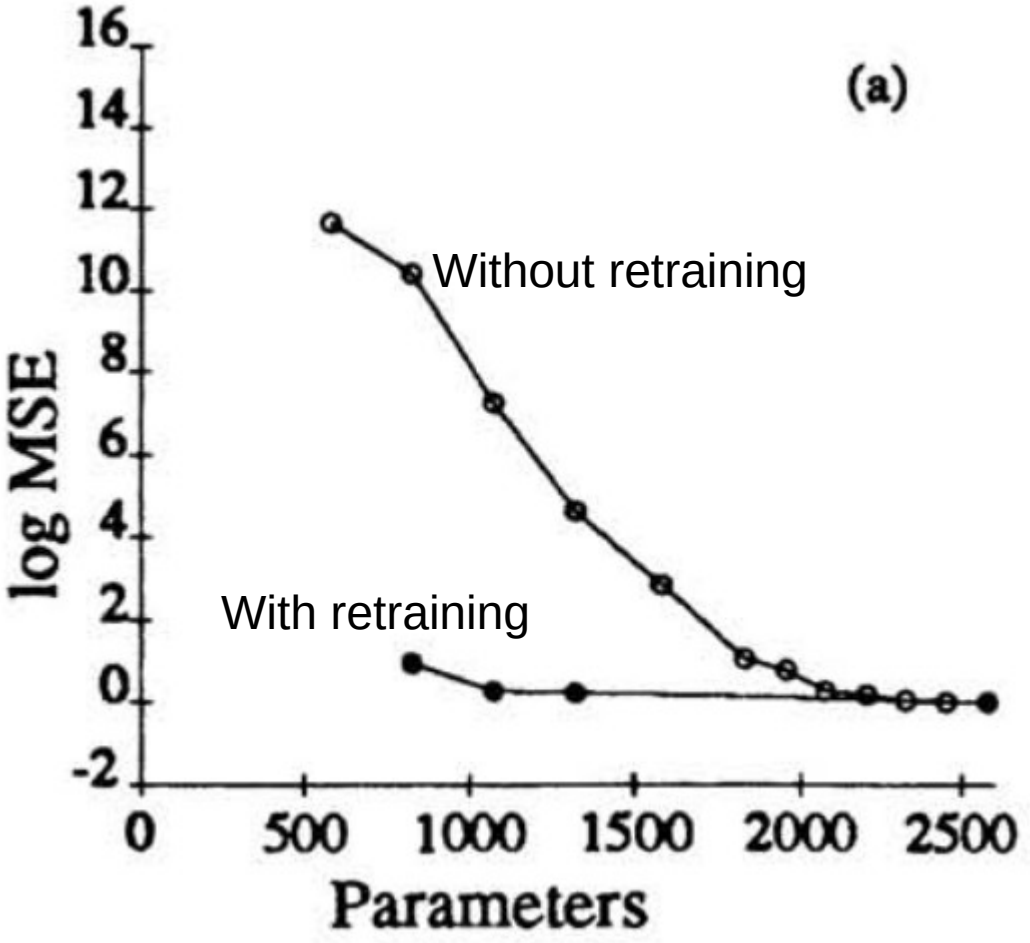
Optimal brain damage [Le Cun et al., NIPS 1990]

- Assumption: the model has been already trained and we are in a local minimum
- All the h_ii terms are non-negative, hence every perturbation will cause the error to go up or stay the same

- The algorithm is composed of the following steps

1. Choose a reasonable network architecture
2. Train the network until a reasonable solution is obtained (train until convergency)
3. Compute the second derivatives hu for each parameter
4. Compute the saliencies for each parameter: Sk = h_u/2
5. Sort the parameters by saliency and delete some low-saliency parameters (thresholding)
6. Iterate to step 2

Results [Le Cun et al., NIPS 1990]



Limits for these approaches

- Computationally extremely expensive
- They do not work so well in multi-layer architectures
- Slow (a lot of iterations to converge)

THESE ARE JUST FEW OF MANY OTHER WORKS IN THE ‘90s (and even before)

A revival of pruning (2015 & beyond)

A revival of pruning

- Possible to scale to “deep” architectures like VGG, GoogLeNet, ResNet etc…
- The number of parameters becomes huge!

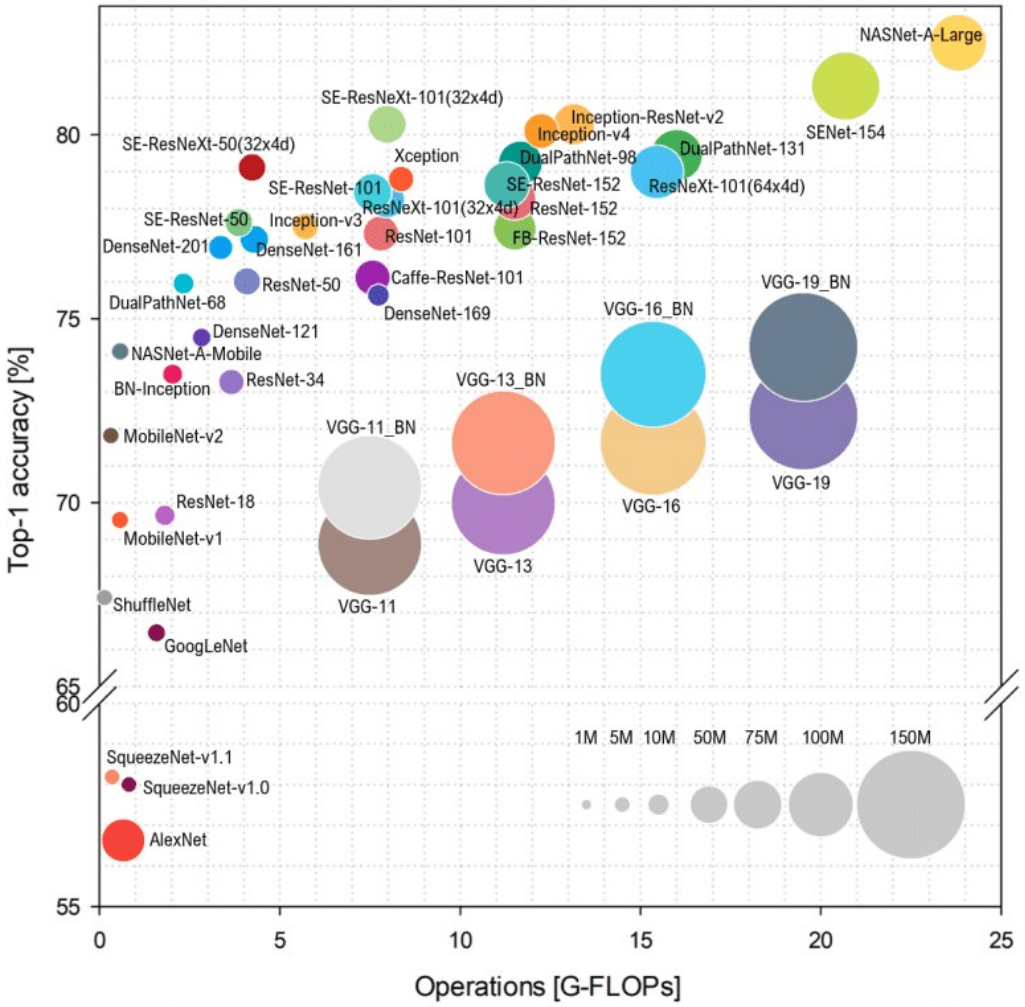


Image from https://theaisummer.com/cnn-architectures/

Outline for PART 1

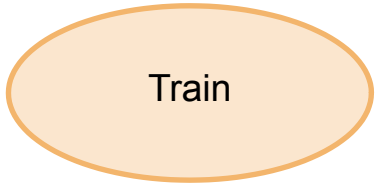
- Why pruning? Motivations and aims
- Voices from the past: pruning in the ‘90s
  - Skeletonization [Mozer and Smolensky, NIPS 1989]
  - Optimal brain damage [Le Cun et al., NIPS 1990]
- A revival of pruning (2015 & beyond)
  - Learning both weights and connections [Han et al., NIPS 2015]
  - Variational dropout sparsifies deep neural networks [Molchanlov et al., ICML 2017]
  - Learning sparse networks using targeted dropout [Gomez et al., NeurIPS 2019]
  - Learning sparse neural networks via sensitivity-driven regularization [Tartaglione et al., NeurIPS 2018]
  - Learning sparse neural networks through L0 regularization [Louizos et al., ICLR 2018]
- The lottery ticket hypothesis [Frankle and Carbin, ICLR 2019]
  - SNIP: Single-shot network pruning based on connection sensitivity [Lee et al., ICLR 2019]
  - Can we prune at initialization?
- Structured vs unstructured pruning

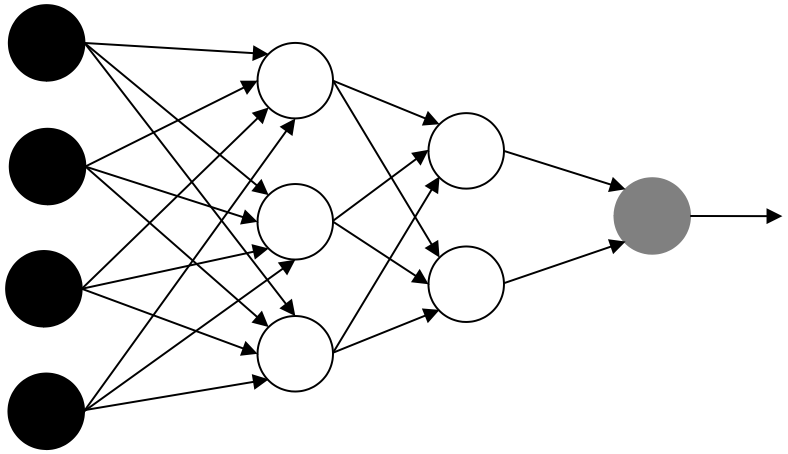
Learning both weights and connections [Han et al., NIPS 2015]

- Simpler idea than from the past: magnitude pruning (this work is not the first proposing this approach though)
- If a parameter has a very small value, it is pruned from the network
- No concept of saliency or other estimators, so much simpler…
  …but the model is REGULARIZED!

Learning both weights and connections [Han et al., NIPS 2015]

- Parameters are randomy initialized



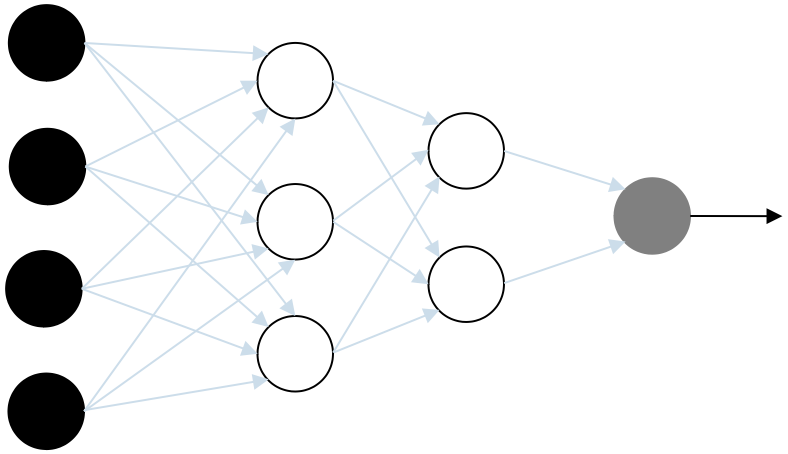


Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*.

Learning both weights and connections [Han et al., NIPS 2015]

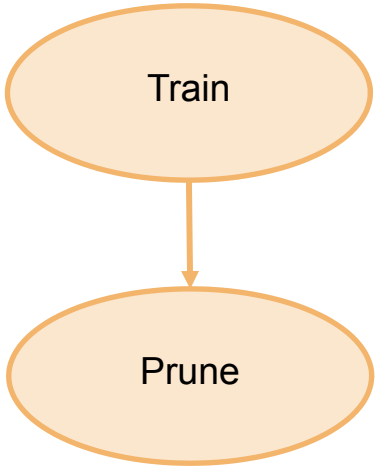
Train

- Parameters are randomy initialized
- Parameters are updated then trained with standard gradient descent until performance is achieved (training stage)

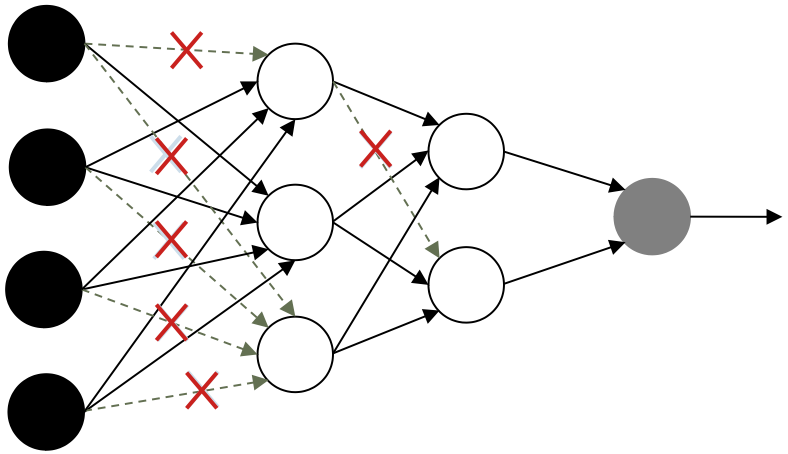


Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*.

Learning both weights and connections [Han et al., NIPS 2015]

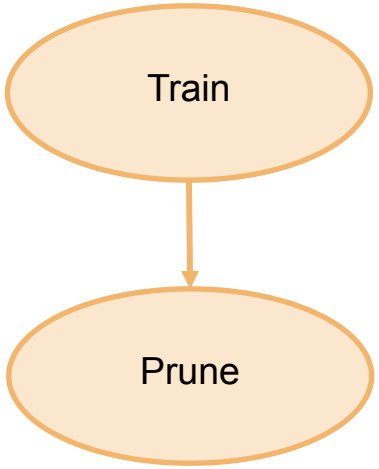


- Parameters are randomy initialized
- Parameters are updated then trained with standard gradient descent until performance is achieved (training stage)
- Parameters below threshold T are removed, pruning connections (parameter sparsification)

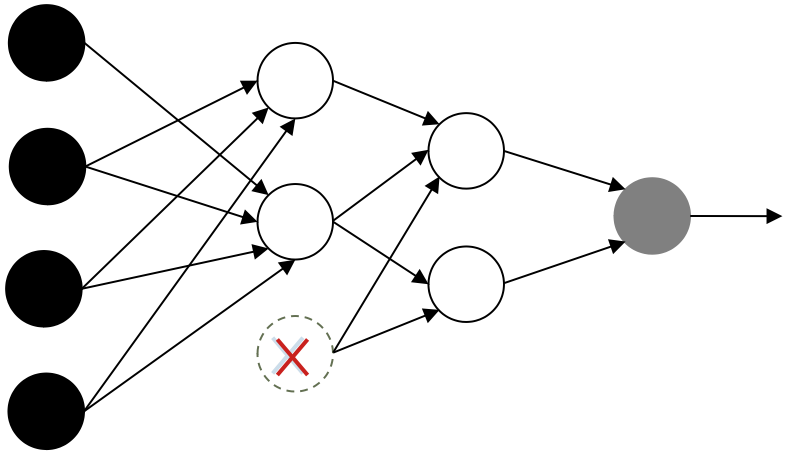


Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*.

Learning both weights and connections [Han et al., NIPS 2015]

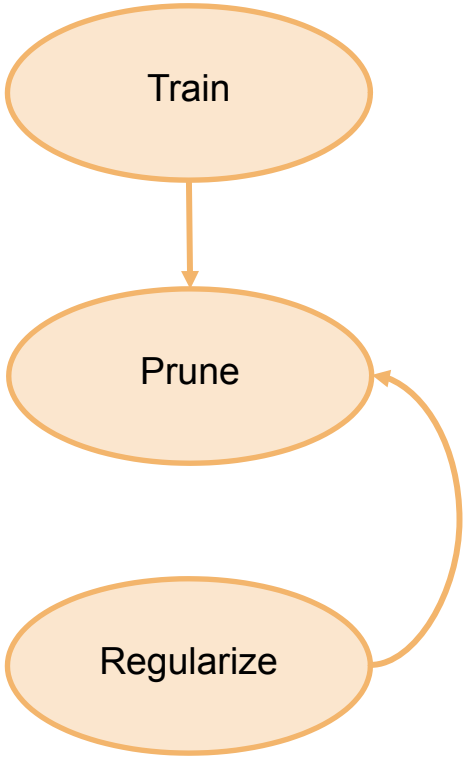


- Parameters are randomy initialized
- Parameters are updated then trained with standard gradient descent until performance is achieved (training stage)
- Parameters below threshold T are removed, pruning connections (parameter sparsification)
- Neurons without input arcs input are pruned from the network (neuron sparsification) ->Degrades network performance

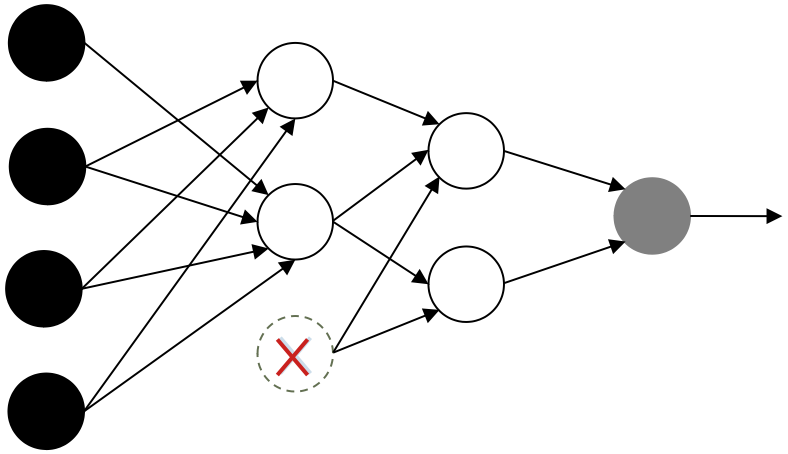


Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*.

Learning both weights and connections [Han et al., NIPS 2015]



- Parameters are randomy initialized
- Parameters are updated then trained with standard gradient descent until performance is achieved (training stage)
- Parameters below threshold T are removed, pruning connections (parameter sparsification)
- Neurons without input arcs input are pruned from the network (neuron sparsification) ->Degrades network performance
- Fine-tune the model, recovering the performance and iteratively prune again.



Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*.

Regularization in deep learning

- Add extra constraints to the objective function minimized.

$$J(x, \hat{y}, w) = \eta L[y(x, w), \hat{y}] + \lambda R(w)$$

Regularization in deep learning

- Add extra constraints to the objective function minimized.

$$J(x, \hat{y}, w) = \eta L[y(x, w), \hat{y}] + \lambda R(w)$$

- Some very common regularizations are dropout, L2 (weight-decay)…

Regularization in deep learning

- Which regularizer would you put to enforce sparsity?
- From the optimization theory, we know LASSO (L1) is the most common choice. Why?

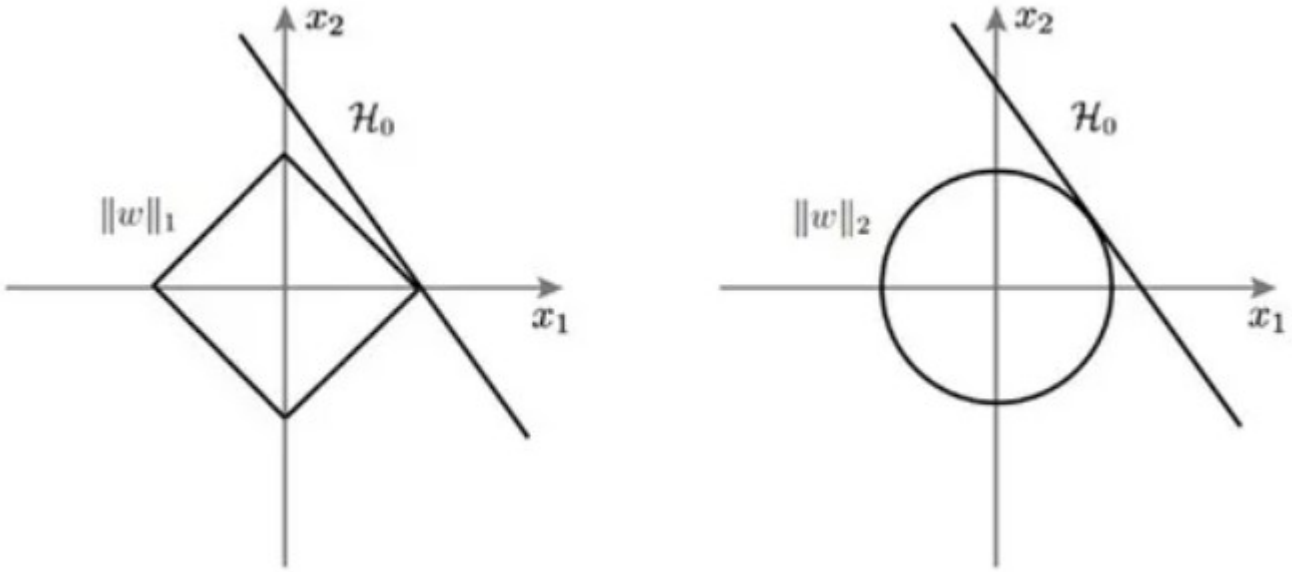


Image from https://www.quora.com/When-would-you-chose-L1-norm-over-L2-norm

Regularization in deep learning

- Can we enforce directly an L0 regularization?

Image from https://kevinbinz.com/2019/06/09/regularization/

Regularization in deep learning

- Can we enforce directly an L0 regularization?
- NO, it is non-differentiable. There are proxies…

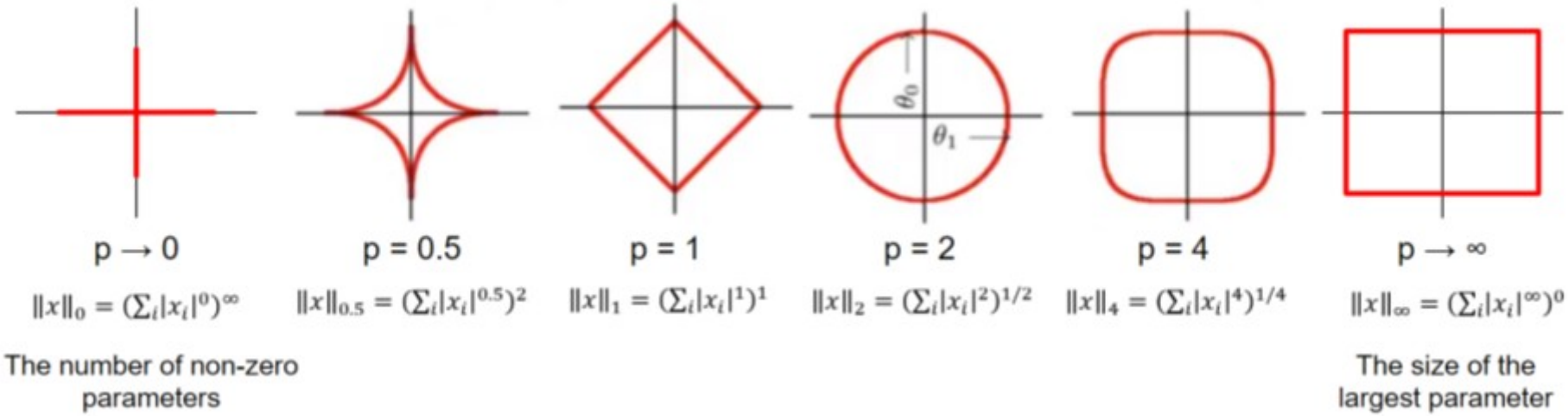
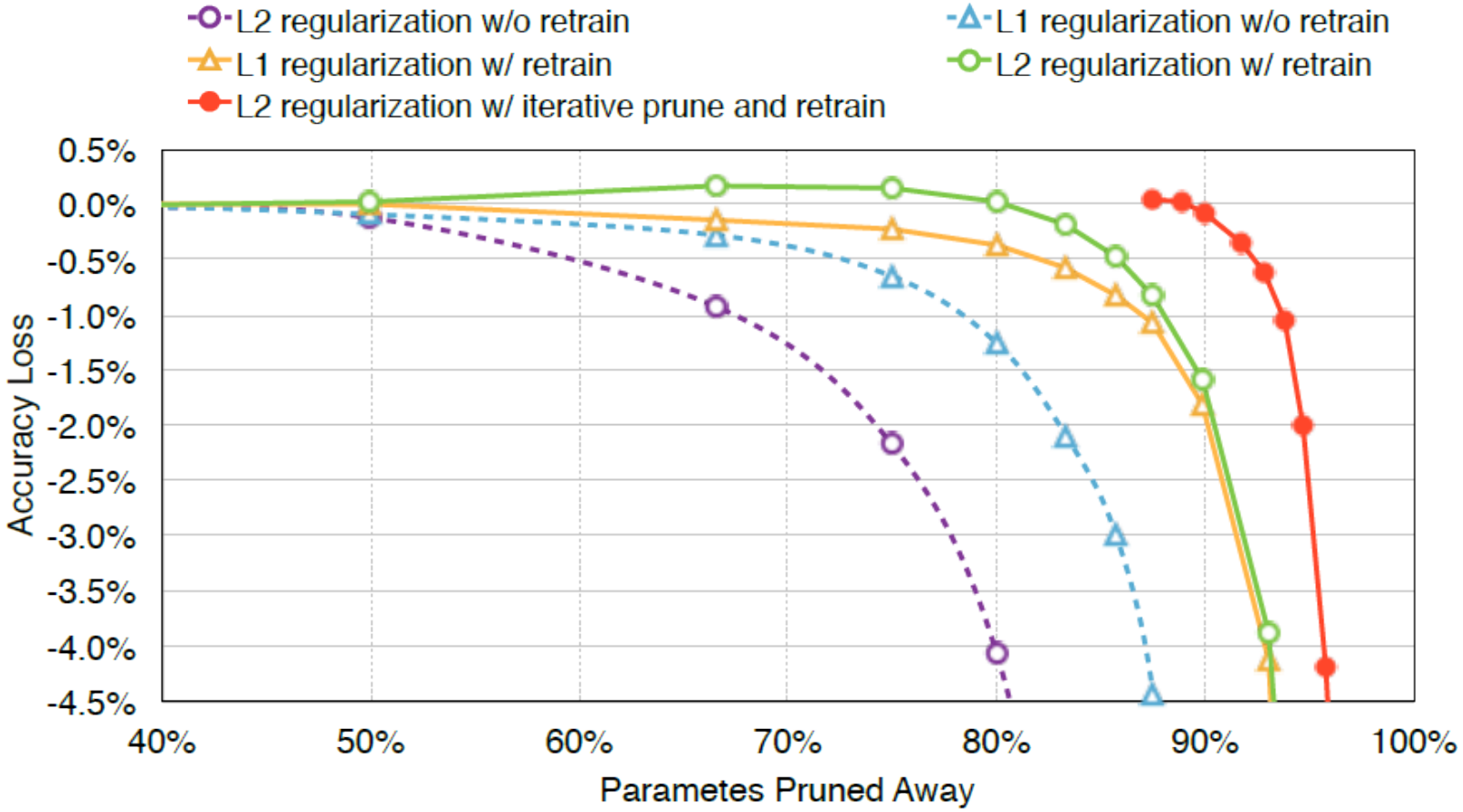


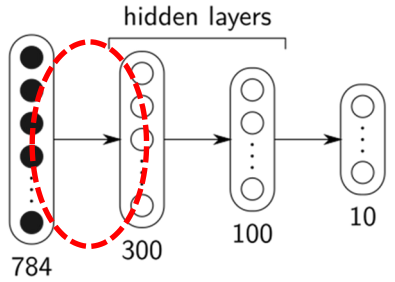
Image from https://kevinbinz.com/2019/06/09/regularization/

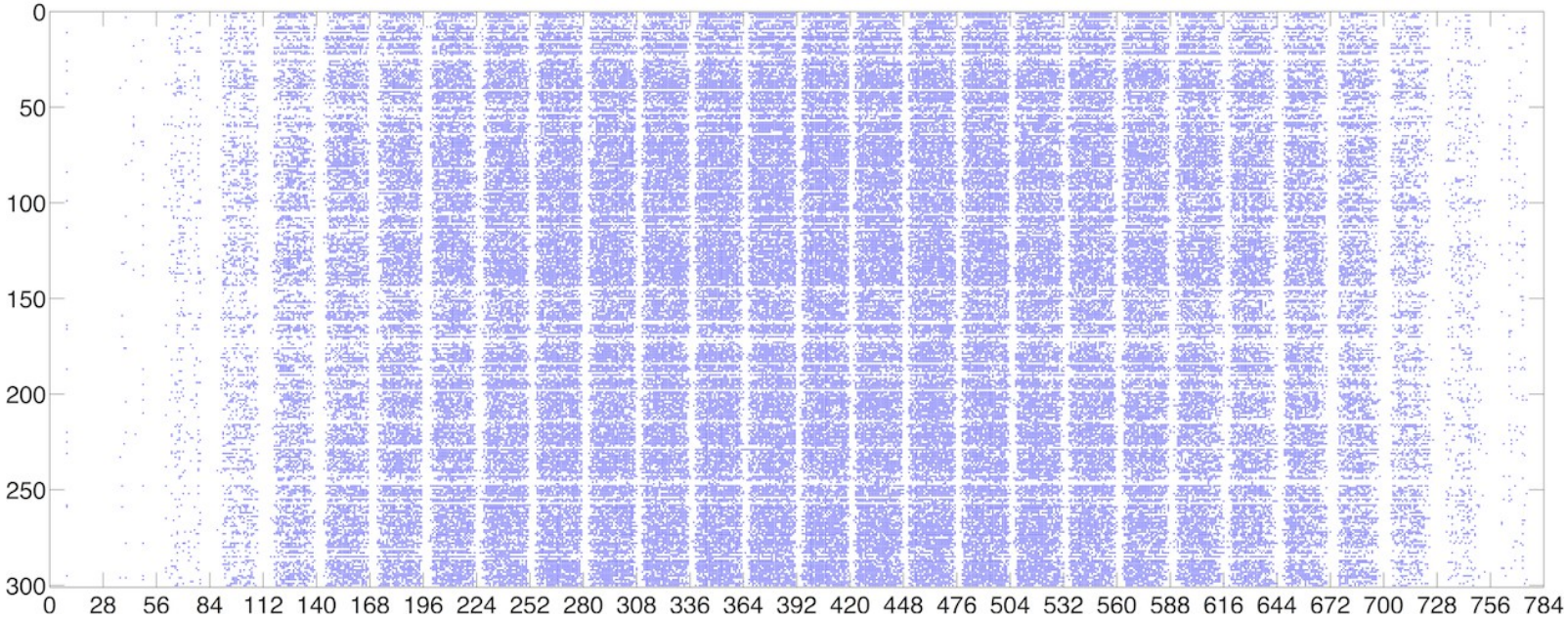
Learning both weights and connections [Han et al., NIPS 2015]

- SURPRISE! L2 leads to sparser models (under same performance constraints)!
- Why? Fine-tuning is the key



Learning both weights and connections [Han et al., NIPS 2015]





Han, Song, Jeff Pool, John Tran, and William Dally. "Learning both weights and connections for efficient neural network." In Advances in neural information processing systems, pp. 1135-1143. 2015.

Learning both weights and connections [Han et al., NIPS 2015]

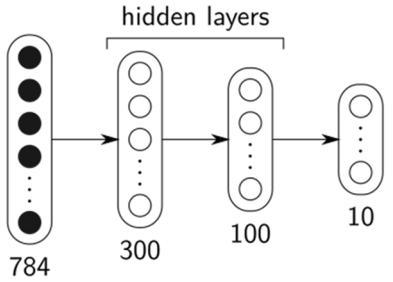


Table 1: Network pruning can save 9× to 13× parameters with no drop in predictive performance.

Network	Top-1 Error	Top-5 Error	Parameters	Compression Rate
LeNet-300-100 Ref	1.64%	-	267K	
LeNet-300-100 Pruned	1.59%	-	**22K**	**12×**
LeNet-5 Ref	0.80%	-	431K	
LeNet-5 Pruned	0.77%	-	**36K**	**12×**
AlexNet Ref	42.78%	19.73%	61M	
AlexNet Pruned	42.77%	19.67%	**6.7M**	**9×**
VGG-16 Ref	31.50%	11.32%	138M	
VGG-16 Pruned	31.34%	10.88%	**10.3M**	**13×**

Han, Song, Jeff Pool, John Tran, and William Dally. "Learning both weights and connections for efficient neural network." In Advances in neural information processing systems, pp. 1135-1143. 2015.

Outline for PART 1

- Why pruning? Motivations and aims
- Voices from the past: pruning in the ‘90s
  - Skeletonization [Mozer and Smolensky, NIPS 1989]
  - Optimal brain damage [Le Cun et al., NIPS 1990]
- A revival of pruning (2015 & beyond)
  - Learning both weights and connections [Han et al., NIPS 2015]
  - Variational dropout sparsifies deep neural networks [Molchanlov et al., ICML 2017]
  - Learning sparse networks using targeted dropout [Gomez et al., NeurIPS 2019]
  - Learning sparse neural networks via sensitivity-driven regularization [Tartaglione et al., NeurIPS 2018]
  - Learning sparse neural networks through L0 regularization [Louizos et al., ICLR 2018]
- The lottery ticket hypothesis [Frankle and Carbin, ICLR 2019]
  - SNIP: Single-shot network pruning based on connection sensitivity [Lee et al., ICLR 2019]
  - Can we prune at initialization?
- Structured vs unstructured pruning

Variational Dropout Sparsifies Deep Neural Networks
[Molchanov et al., ICML 2017]

- Proposed a variational approach to drive pruning of single parameters
- “Dropout” here is intended as “DropConnect”

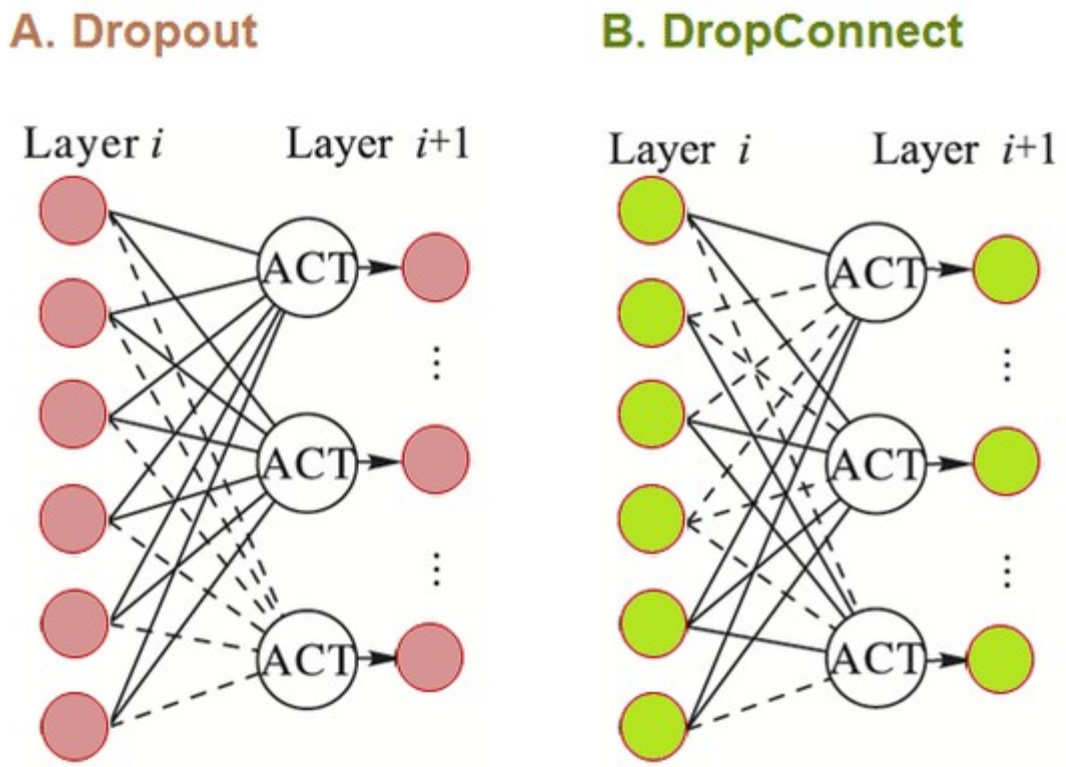


Image from Leonard, Juan. (2019). Image Classification and Object Detection Algorithm Based on Convolutional Neural Network. Science Insights. 31. 85-100. 10.15354/si.19.re117.

The reparametrization trick [Kingma and Welling, ICLR 2014]

- Key element of the “Variational autoencoder” architectures (VAEs)
- Let some mean μ and standard deviation σ for a gaussian distribution. In case we need to sample some z from this distribution, this will make the whole sampling process stochastic. How can we make back-propagation passing through this stochastic node?

- At sampling time, we inject some gaussian noise, which will be our stochastic element! This allows us to have μ and σ as deterministic node, which allows us to optimize!

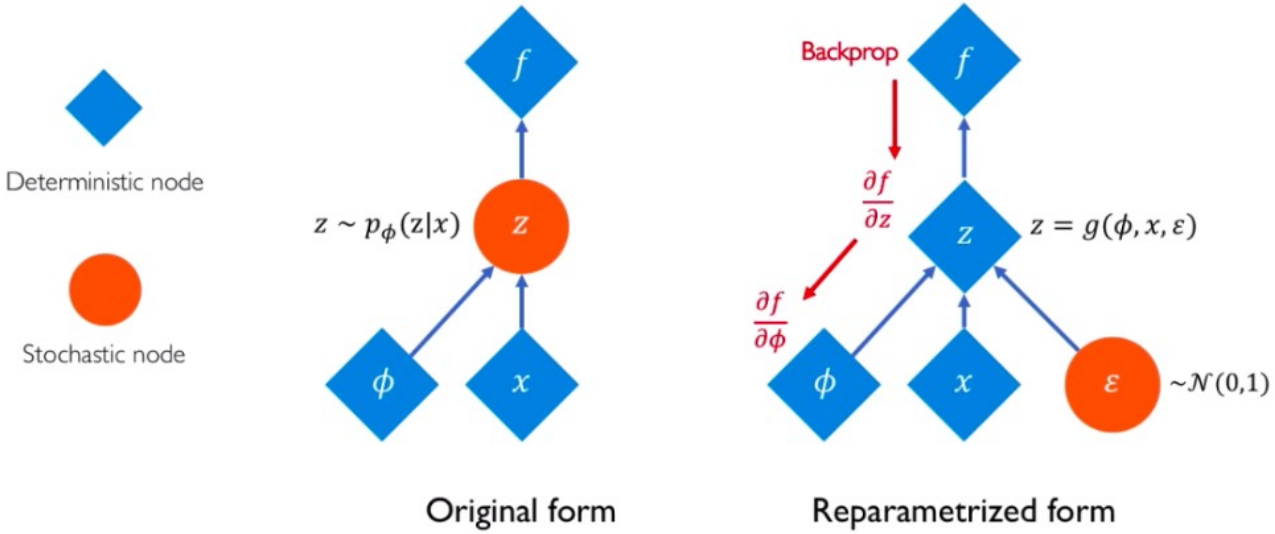


Image from https://www.youtube.com/watch?v=rZufA635dq4

Variational Dropout Sparsifies Deep Neural Networks
[Molchanov et al., ICML 2017]

- The parameter is modeled as the real value θ and the variationally-learned parameter α
- In order to make the gradient towards θ unitary, the variance can be modeled as

making

$$\sigma_{ij}^2 = \alpha_{ij}\theta_{ij}^2$$

$$w_{ij} = \theta_{ij}(1 + \sqrt{\alpha_{ij}} \cdot \epsilon_{ij}) = \theta_{ij} + \sigma_{ij} \cdot \epsilon_{ij}$$

$$\frac{\partial w_{ij}}{\partial \theta_{ij}} = 1, \qquad \epsilon_{ij} \sim \mathcal{N}(0, 1)$$

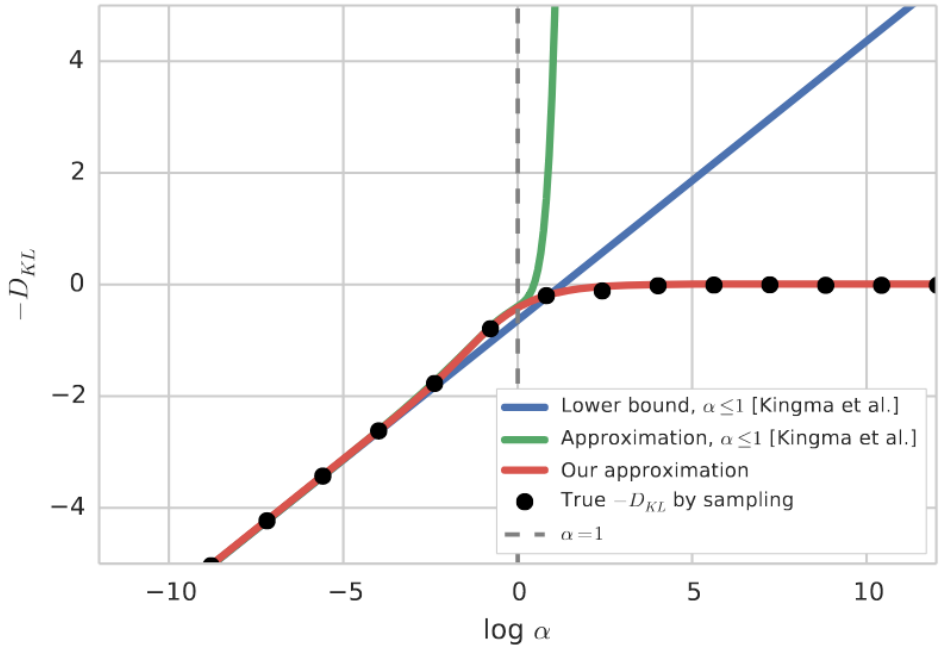
Variational Dropout Sparsifies Deep Neural Networks
[Molchanov et al., ICML 2017]

- The KL-divergence term in this approach is

$$-D_{KL}(q(w_{ij} \mid \theta_{ij}, \alpha_{ij}) \,\|\, p(w_{ij})) =$$
$$= \frac{1}{2} \log \alpha_{ij} - \mathbb{E}_{\epsilon \sim \mathcal{N}(1, \alpha_{ij})} \log |\epsilon| + \mathrm{C}$$

- The second term is intractable; however, it is approximated with

$$-D_{KL}(q(w_{ij} \mid \theta_{ij}, \alpha_{ij}) \,\|\, p(w_{ij})) \approx$$
$$\approx k_1 \sigma(k_2 + k_3 \log \alpha_{ij})) - 0.5 \log(1 + \alpha_{ij}^{-1}) + \mathrm{C}$$
$$k_1 = 0.63576 \quad k_2 = 1.87320 \quad k_3 = 1.48695$$



Variational Dropout Sparsifies Deep Neural Networks
[Molchanov et al., ICML 2017]

- On simple architectures/datasets (MNIST), this approach can sparsify well!

Network	Method	Error %	Sparsity per Layer %	$\frac{\lvert\mathbf{W}\rvert}{\lvert\mathbf{W}_{\neq 0}\rvert}$
	Original	1.64		1
	Pruning	1.59	92.0 – 91.0 – 74.0	12
LeNet-300-100	DNS	1.99	98.2 – 98.2 – 94.5	56
	SWS	1.94		23
(ours)	Sparse VD	1.92	98.9 – 97.2 – 62.0	**68**
	Original	0.80		1
	Pruning	0.77	34 – 88 – 92.0 – 81	12
LeNet-5-Caffe	DNS	0.91	86 – 97 – 99.3 – 96	111
	SWS	0.97		200
(ours)	Sparse VD	0.75	67 – 98 – 99.8 – 95	**280**

Outline for PART 1

- Why pruning? Motivations and aims
- Voices from the past: pruning in the ‘90s
  - Skeletonization [Mozer and Smolensky, NIPS 1989]
  - Optimal brain damage [Le Cun et al., NIPS 1990]
- A revival of pruning (2015 & beyond)
  - Learning both weights and connections [Han et al., NIPS 2015]
  - Variational dropout sparsifies deep neural networks [Molchanlov et al., ICML 2017]
  - Learning sparse networks using targeted dropout [Gomez et al., NeurIPS 2019]
  - Learning sparse neural networks via sensitivity-driven regularization [Tartaglione et al., NeurIPS 2018]
  - Learning sparse neural networks through L0 regularization [Louizos et al., ICLR 2018]
- The lottery ticket hypothesis [Frankle and Carbin, ICLR 2019]
  - SNIP: Single-shot network pruning based on connection sensitivity [Lee et al., ICLR 2019]
  - Can we prune at initialization?
- Structured vs unstructured pruning

Learning Sparse Networks Using Targeted Dropout
[Gomez et al., NeurIPS 2019]

- Formulation adapting to both neurons and single parameters (dropout and/or dropconnect).
- The work compares solutions with variational, lasso (L1) and L0 (Louizos et al.) approaches against the proposed “targeted” dropout.
- In this case, the “targeted” term implies a dropout guided by the **dropout probability α** for the **γ lowest-valued parameters** (this is why “targeted”). These are two hyper-parameters for the model.

Learning Sparse Networks Using Targeted Dropout
[Gomez et al., NeurIPS 2019]

- NO PRUNING at training time, just enforcing the model's parameters removal after training
- Very dependent on how to tune the two hyper-parameters
- Pruning entire units is more difficult than removing single parameters!

Weight Dropout/Pruning

prune percentage	none	dropout α=0.25	targeted α=0.5,γ=0.5	targeted α=0.33,γ=0.75	targeted α=0.66,γ=0.75	targeted α=0.75,γ=0.90	variational	$L^1_{0.1}$	$L^0_{0.1}$
0 %	93.71	93.62	93.03	89.88	92.64	92.53	92.09	92.80	88.83
10%	93.72	93.63	93.04	89.80	92.62	92.55	92.00	92.72	90.66
20%	93.77	93.66	93.02	89.93	92.63	92.48	92.02	92.84	88.64
30%	93.59	93.58	92.98	89.89	92.66	92.53	92.07	92.63	87.16
40%	93.09	93.45	93.03	89.75	92.70	92.63	92.12	92.80	85.31
50%	92.20	93.07	92.99	89.72	92.65	92.54	91.84	92.29	80.94
60%	90.46	90.81	92.66	89.84	92.70	92.55	91.48	91.20	69.48
70%	81.88	72.29	92.22	89.80	92.66	92.56	90.23	86.30	46.19
80%	32.02	19.84	84.03	85.80	91.86	92.54	83.44	63.00	23.71
90%	14.63	10.05	28.27	27.04	67.58	92.48	15.16	21.08	12.55

Unit Dropout/Pruning

prune percentage	none	dropout α=0.25	targeted α=0.5,γ=0.5	targeted α=0.33,γ=0.75	targeted α=0.66,γ=0.75	targeted α=0.90,γ=0.75	variational	$L^1_{0.01}$	$L^0_{0.01}$
0 %	93.69	92.43	92.21	90.46	89.38	89.78	93.14	93.31	93.35
10%	90.05	67.52	91.96	88.44	89.48	90.18	92.91	91.03	83.01
20%	80.34	25.05	91.63	83.55	88.89	89.79	90.38	85.63	54.59
30%	59.94	13.47	91.30	69.82	88.84	89.88	86.38	72.19	21.34
40%	35.40	10.02	89.89	54.42	87.54	89.98	83.59	46.41	10.82
50%	12.63	9.97	88.41	28.88	84.86	90.05	65.79	26.72	15.04
60%	10.65	9.99	26.55	18.55	81.98	90.08	41.05	12.11	9.46
70%	11.70	10.01	17.41	17.84	75.47	90.03	19.36	11.81	10.02
80%	9.99	9.95	10.63	10.87	28.99	34.18	9.56	14.73	14.88
90%	9.85	9.98	9.30	10.29	9.97	10.04	10.41	10.22	9.98

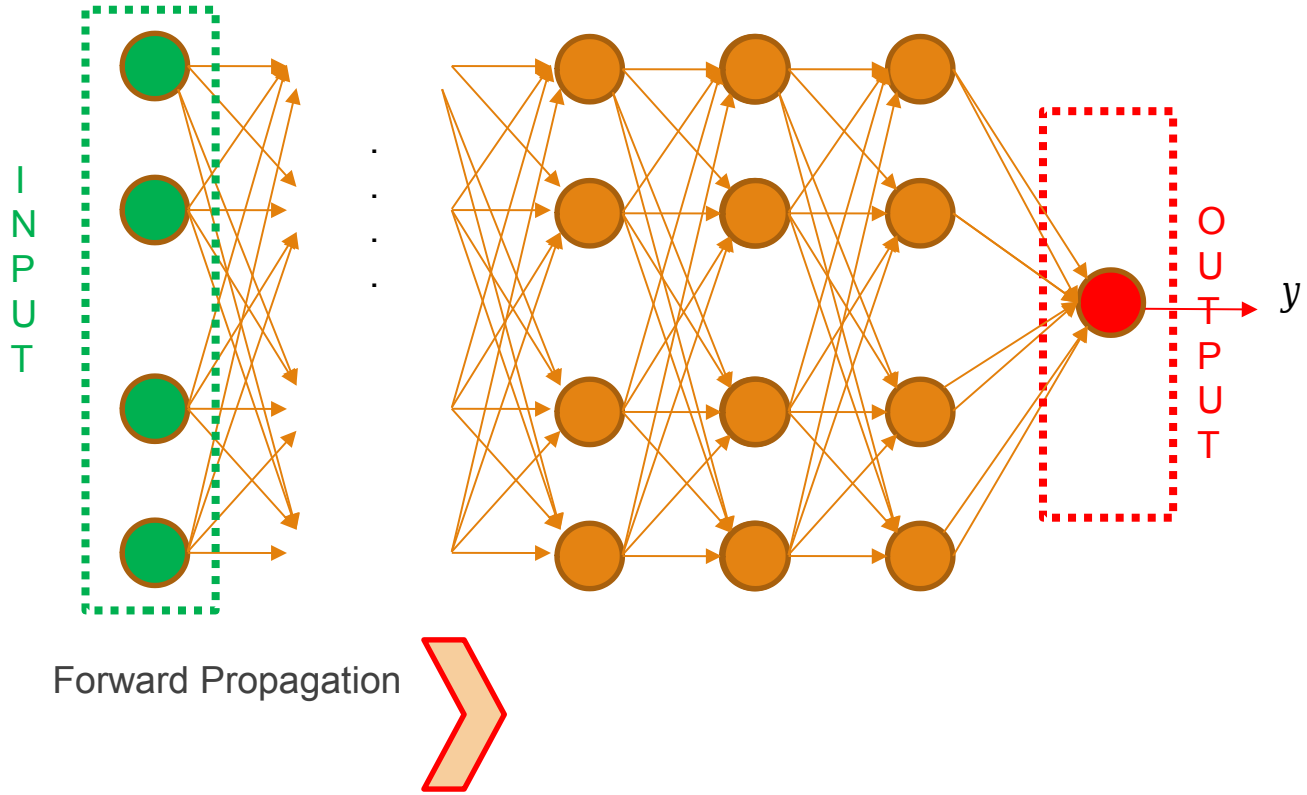
Outline for PART 1

- Why pruning? Motivations and aims
- Voices from the past: pruning in the ‘90s
  - Skeletonization [Mozer and Smolensky, NIPS 1989]
  - Optimal brain damage [Le Cun et al., NIPS 1990]
- A revival of pruning (2015 & beyond)
  - Learning both weights and connections [Han et al., NIPS 2015]
  - Variational dropout sparsifies deep neural networks [Molchanlov et al., ICML 2017]
  - Learning sparse networks using targeted dropout [Gomez et al., NeurIPS 2019]
  - Learning sparse neural networks via sensitivity-driven regularization [Tartaglione et al., NeurIPS 2018]
  - Learning sparse neural networks through L0 regularization [Louizos et al., ICLR 2018]
- The lottery ticket hypothesis [Frankle and Carbin, ICLR 2019]
  - SNIP: Single-shot network pruning based on connection sensitivity [Lee et al., ICLR 2019]
  - Can we prune at initialization?
- Structured vs unstructured pruning

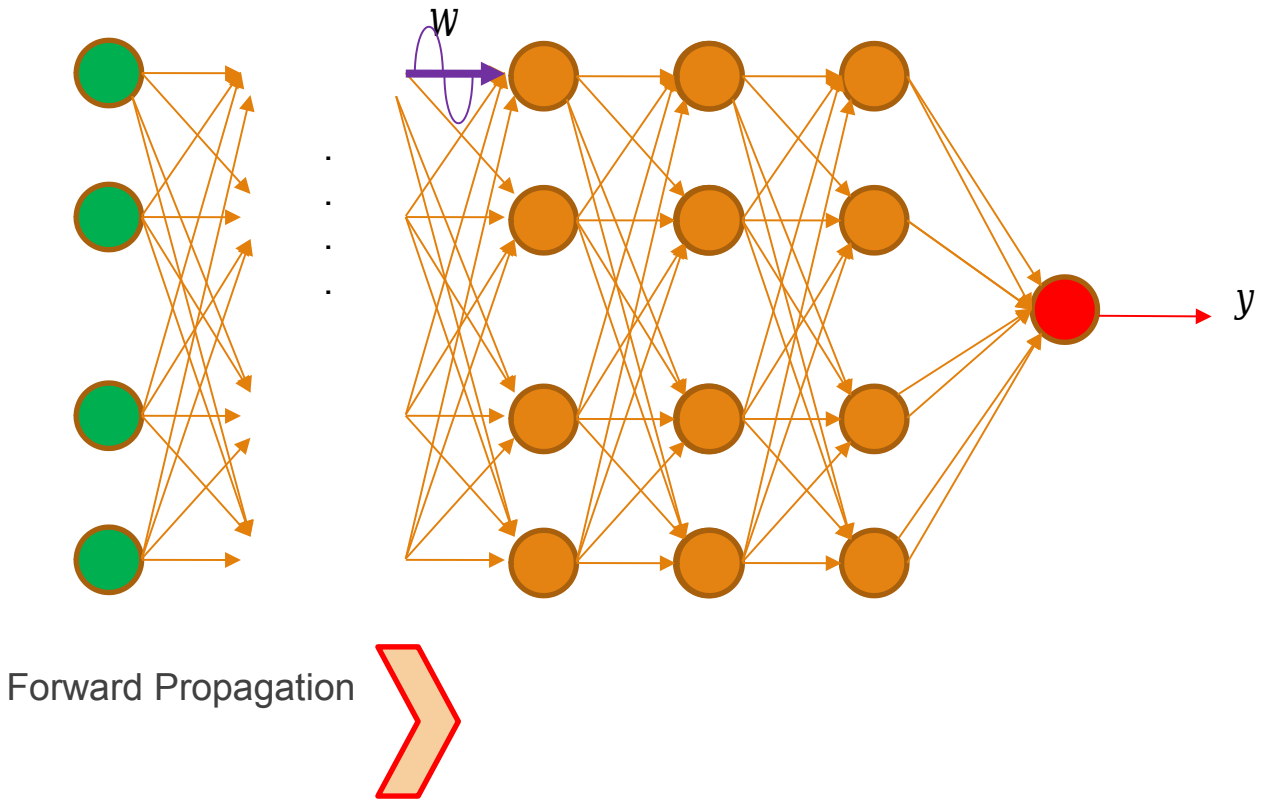
Learning sparse neural networks via sensitivity-driven regularization [Tartaglione et al., NeurIPS 2018]

- Iterative strategy relying in a regularization term favoring sparsity, for deep learning.
- This approach can focus on the single parameters (either considering the outputs or the loss function – LOBSTER) or on entire units (SeReNe).
- Follows the presentation of LOBSTER (the basic concept is the same in all of the formulations)

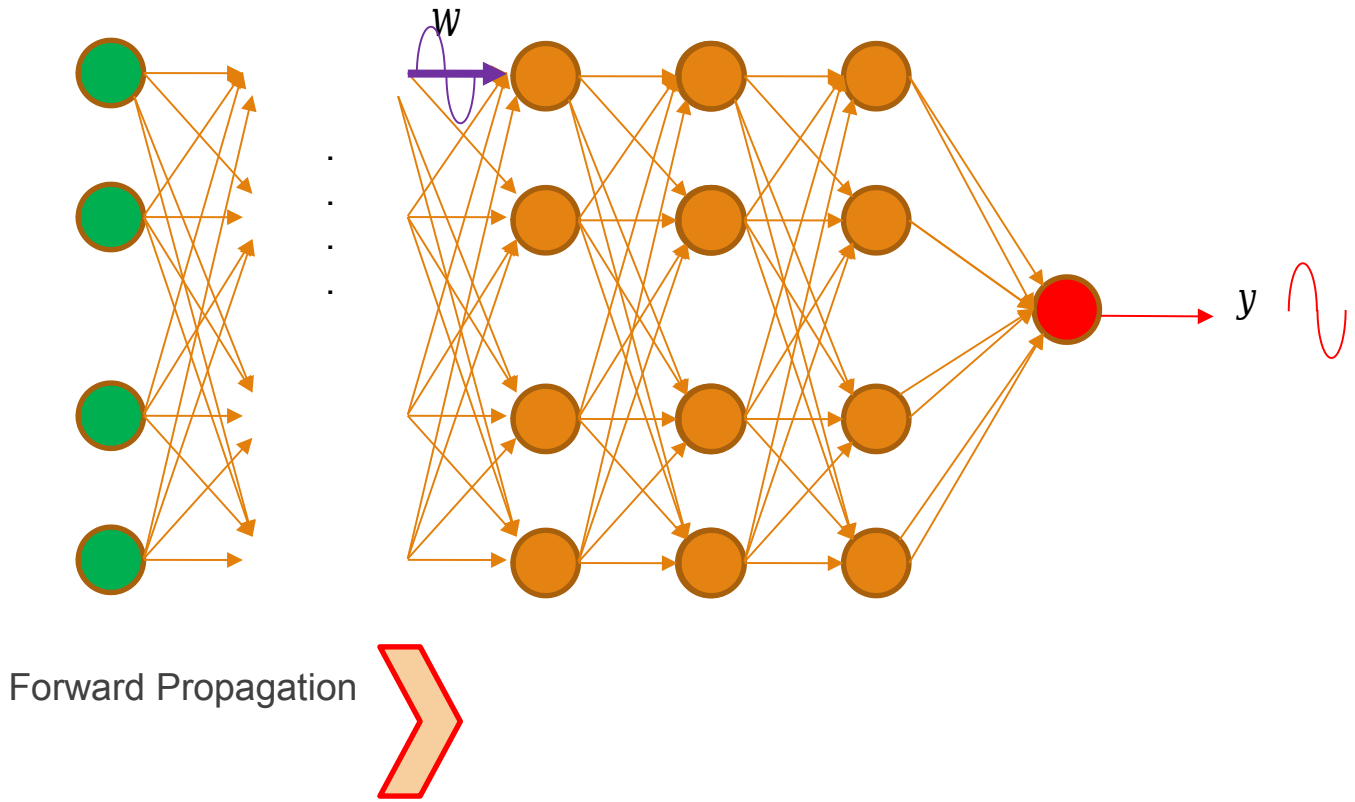
Learning sparse neural networks via sensitivity-driven regularization [Tartaglione et al., NeurIPS 2018]



Learning sparse neural networks via sensitivity-driven regularization [Tartaglione et al., NeurIPS 2018]

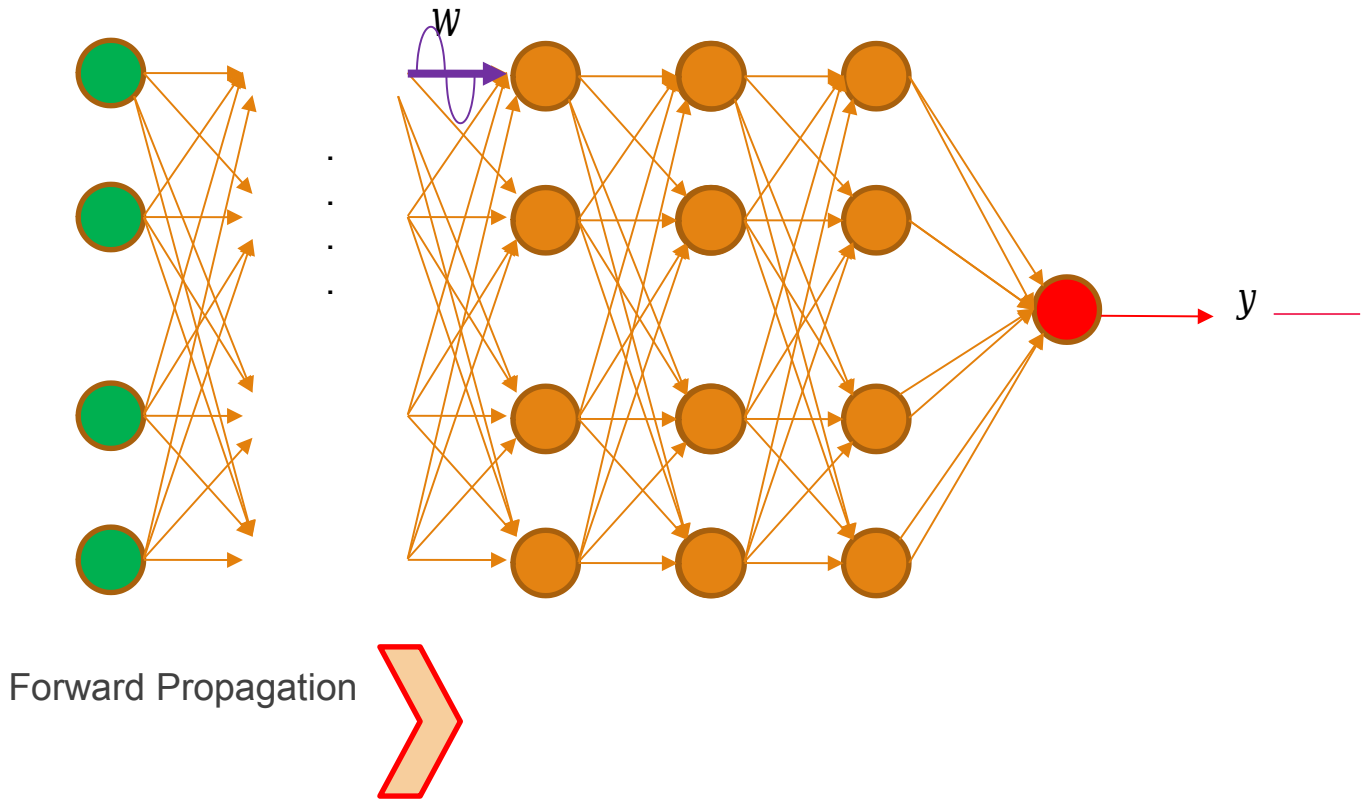


Learning sparse neural networks via sensitivity-driven regularization [Tartaglione et al., NeurIPS 2018]



Changing w we change the output of the network… we don't want to modify it!

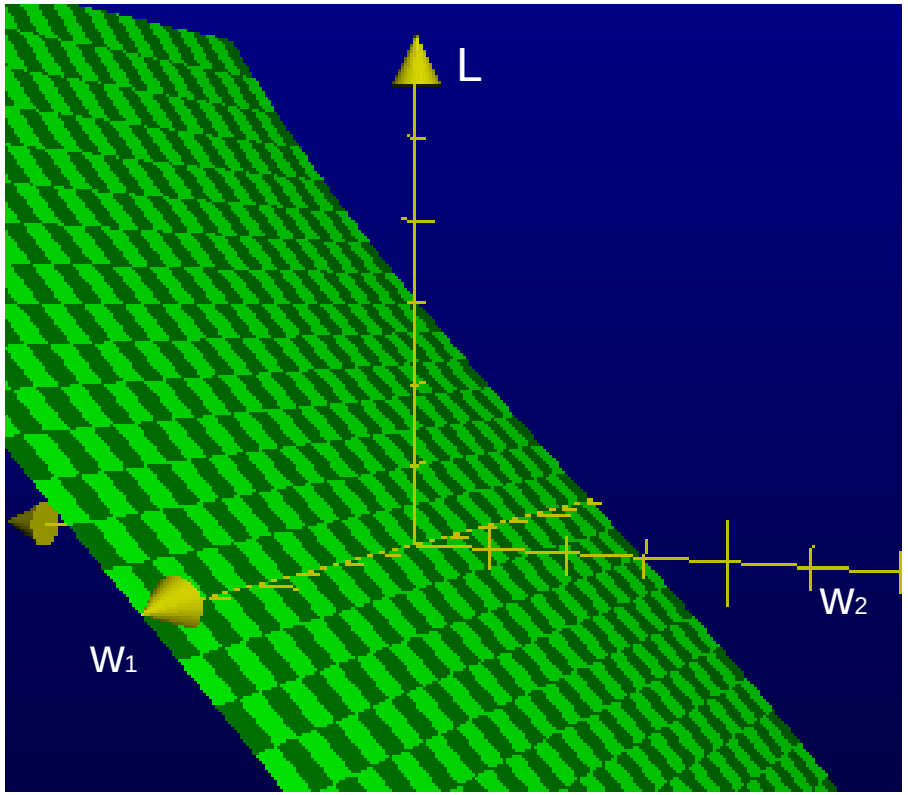
Learning sparse neural networks via sensitivity-driven regularization [Tartaglione et al., NeurIPS 2018]



Changing w we do not change the output of the network… we are free to change it!

LOss-Based SensiTivity rEgulaRization: Towards deep sparse neural networks [Tartaglione et al., Neural Networks 2022]

- We include a penalty term in the update rule.
- Intuition: if a parameter is in a flat region of the loss landscape, it is pushed towards lower and lower values. If the loss landscape remains flat, it is further pushed, otherwise the gradient descent dynamics will drive it.
  - This procedure works with gradients, but it approximates the hessian!



LOss-Based SensiTivity rEgulaRization: Towards deep sparse neural networks [Tartaglione et al., Neural Networks 2022]

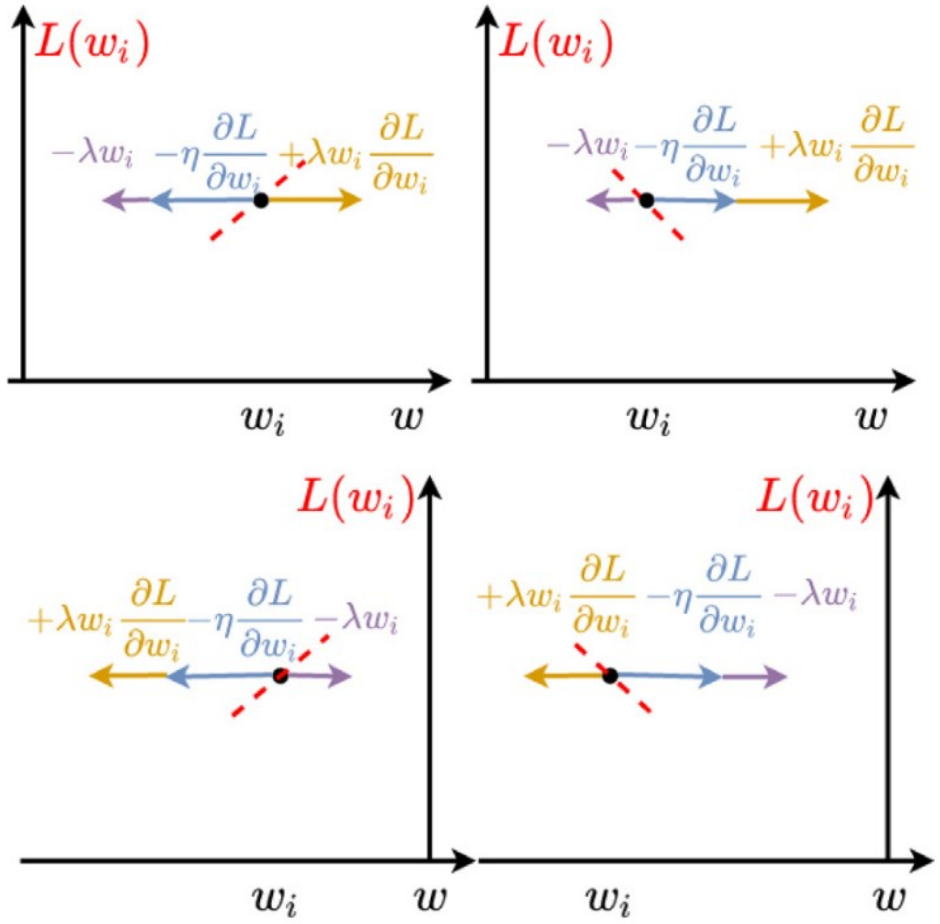
- During the training, we re-weight the L2 regularization by the strength of the gradient

$$w_i^{t+1} := w_i^t - \eta \frac{\partial L}{\partial w_i^t} - \lambda w_i^t \left[1 - S(L, w_i^t)\right] P\left[S(L, w_i^t)\right]$$

  defining sensitivity

$$S(L, w_i) = \left| \frac{\partial L}{\partial w_i} \right|$$

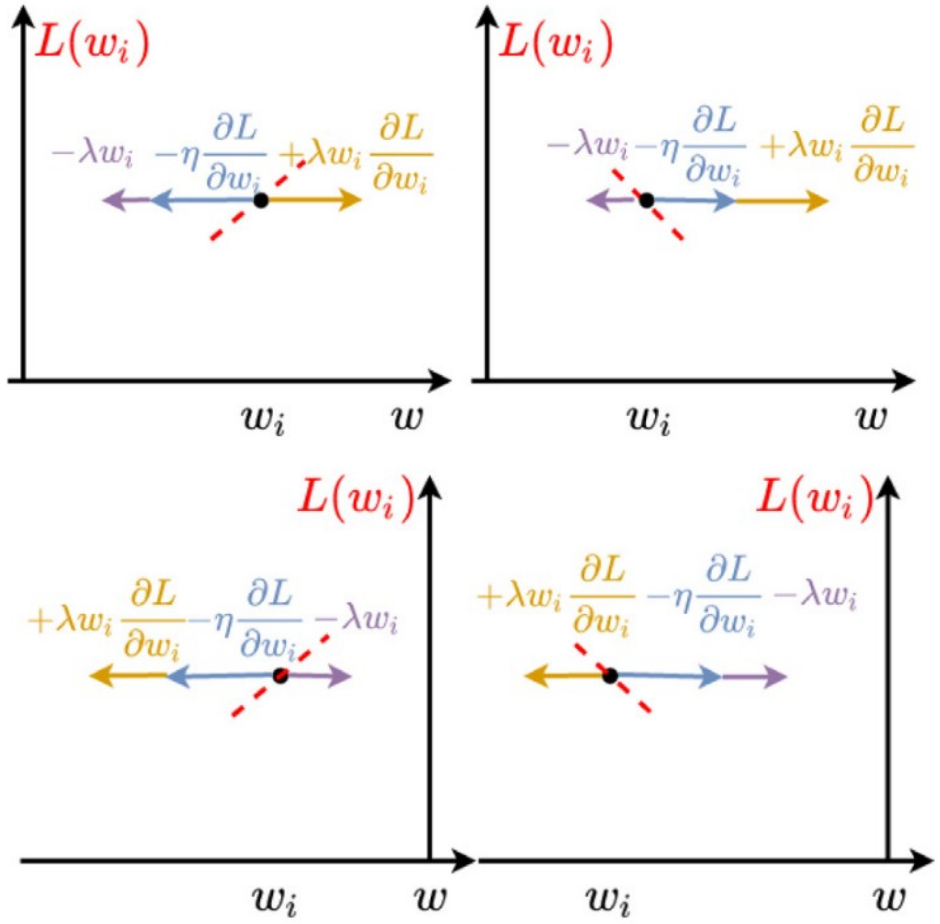
- Effect: if the gradient is zero (flat region) the parameter is penalized towards zero; otherwise the learning dynamics is favored



LOss-Based SensiTivity rEgulaRization: Towards deep sparse neural networks [Tartaglione et al., Neural Networks 2022]

- What if the model is in a local minimum for the loss?

  The model IS PENALIZED; however, if the derivative increases, the regularization term will be shut down!!

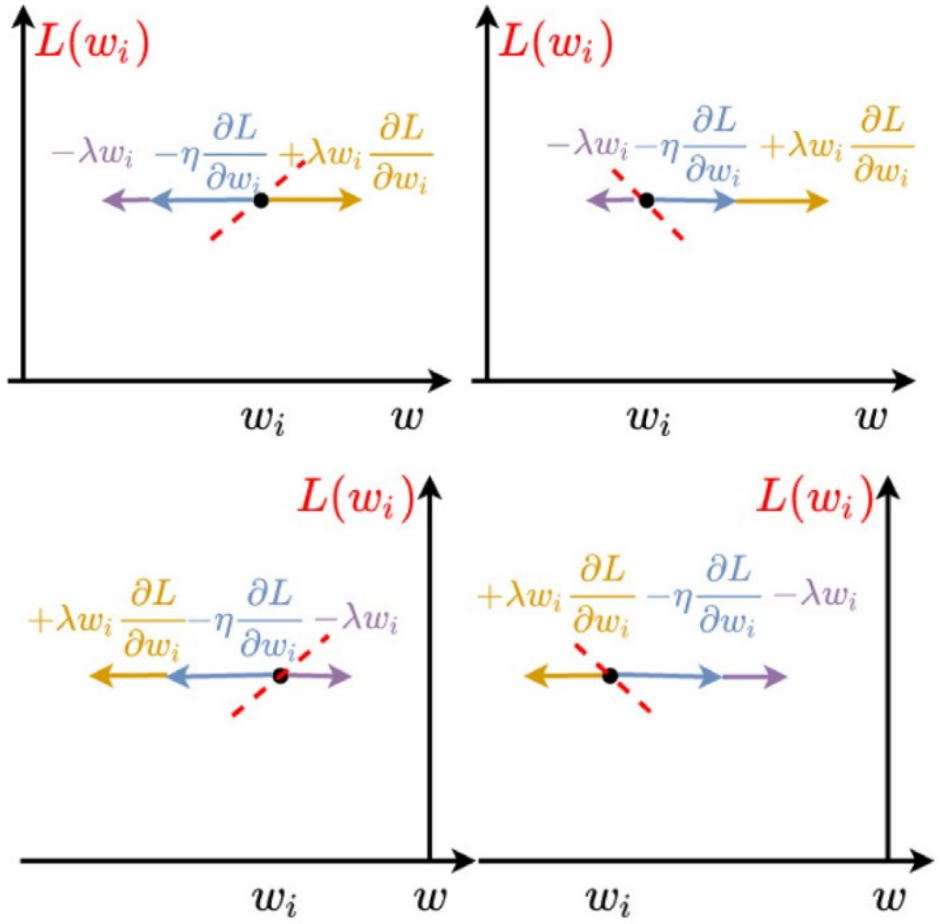


LOss-Based SensiTivity rEgulaRization: Towards deep sparse neural networks [Tartaglione et al., Neural Networks 2022]

- What if the model is in a local minimum for the loss?

  The model IS PENALIZED; however, if the derivative increases, the regularization term will be shut down!!
- As it is an iterative algorithm, it implicitly evaluates the second derivative (Hessian).

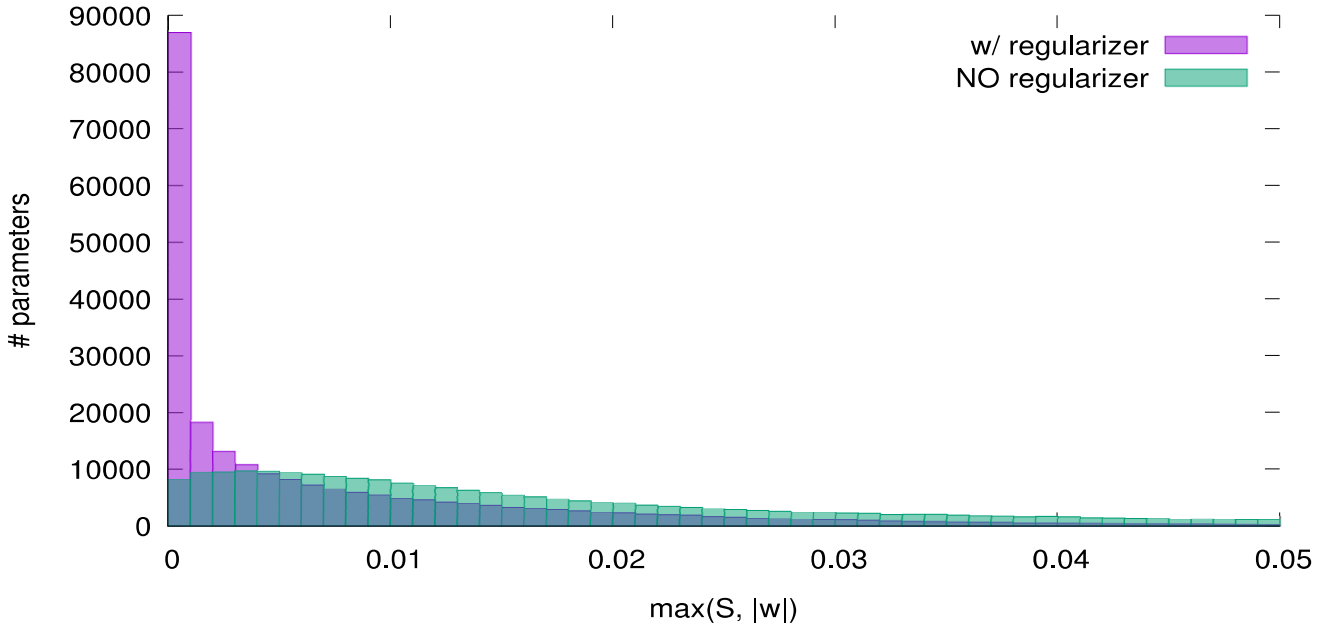
- The equilibrium point will be reached according to the balancing between the hyper-parameters.



LOss-Based SensiTivity rEgulaRization: Towards deep sparse neural networks [Tartaglione et al., Neural Networks 2022]

- During training, due to numerical errors and asymptotic behaviors it might happen that its value never reaches zero.

- Thanks to sensitivity-based regularization (or to other contemporary approaches), the convergency to sparser solutions is faster.



LOss-Based SensiTivity rEgulaRization: Towards deep sparse neural networks [Tartaglione et al., Neural Networks 2022]

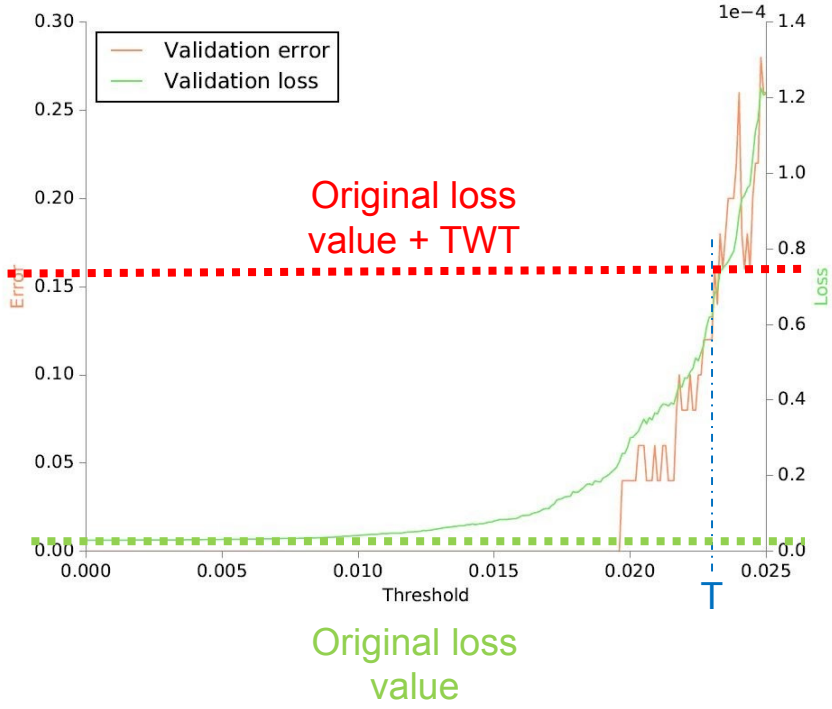
- When we prune, we typically set a fixed pruning rate (how many parameters to be removed, which is typically a percentage)
- We lose some performance! We need to recover it back!
- …what if we are not able anymore to recover it…?
- Can we design a more controlled way to prune?

LOss-Based SensiTivity rEgulaRization: Towards deep sparse neural networks [Tartaglione et al., Neural Networks 2022]

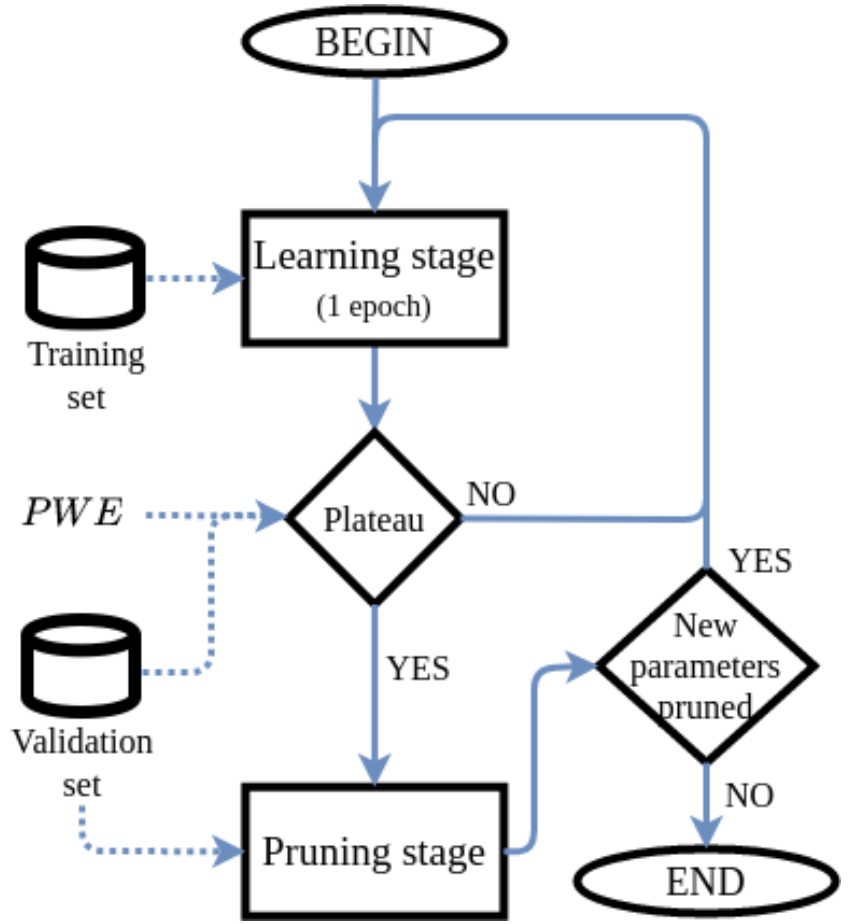
- After 1 epoch the performance is evaluated on the **validation set**.
- In case we reach a plateau in the performances for the last PWE epochs, we proceed to the **thresholding**.
- T if found allowing a maximum relative worsening in the performance (TWT) of the model.
- Parameters are thresholded. Once a parameter is removed, it is **pinned**.

LOss-Based SensiTivity rEgulaRization: Towards deep sparse neural networks [Tartaglione et al., Neural Networks 2022]

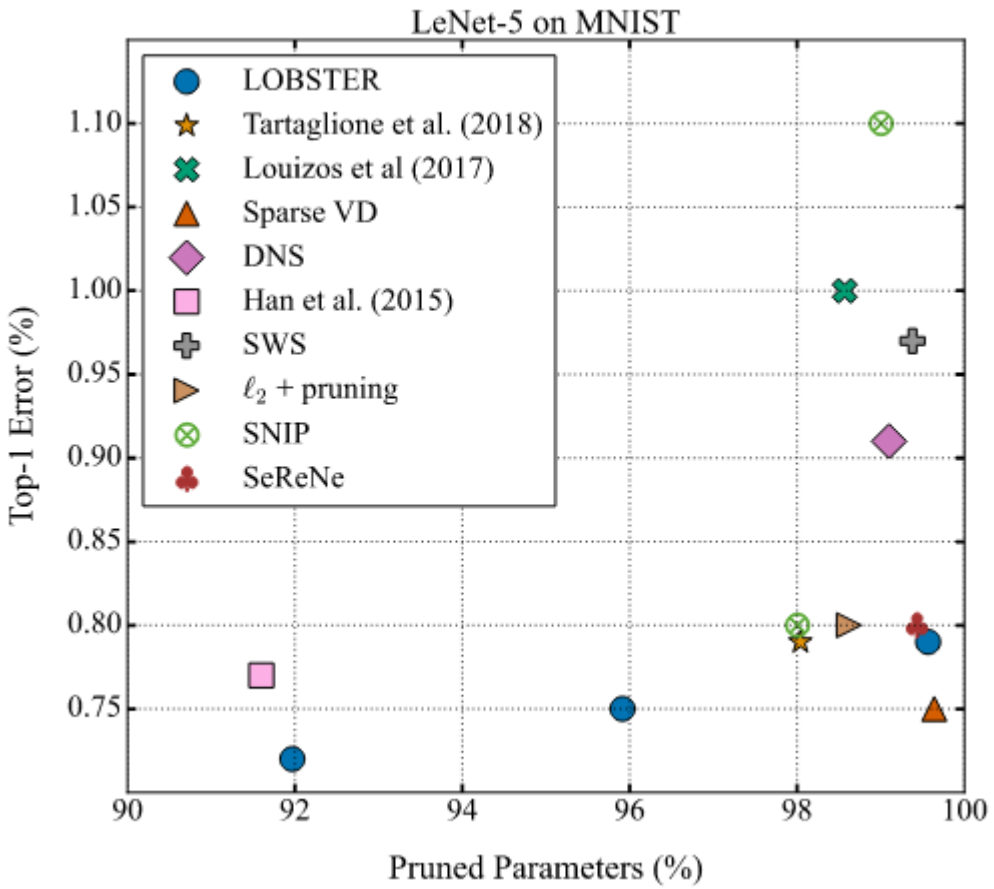
- We fix a given relative performance worsening tolerance (TWT).
- We find T such that the loss value never exceeds the original loss value * (1 +TWT).
- Thresholding is on the parameters, but the regularizer makes most of the pruned parameters belonging to the same neuron.

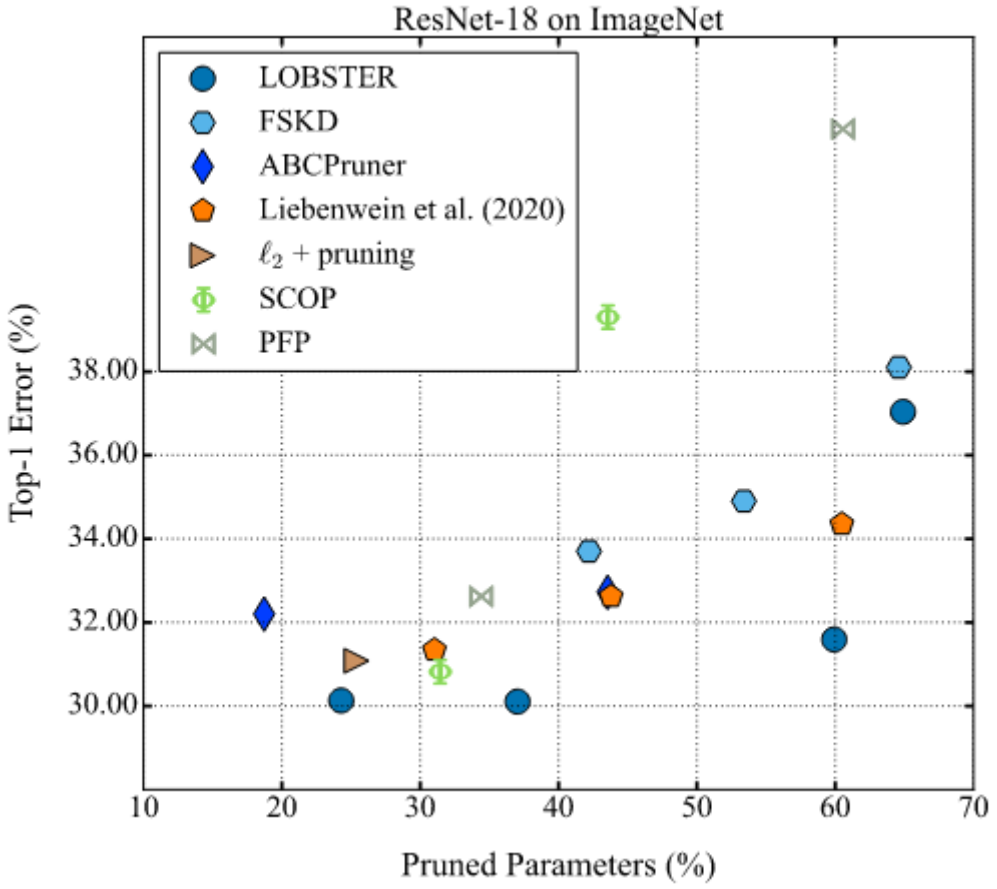


Iterative pruning algorithm



LOss-Based SensiTivity rEgulaRization: Towards deep sparse neural networks [Tartaglione et al., Neural Networks 2022]





Outline for PART 1

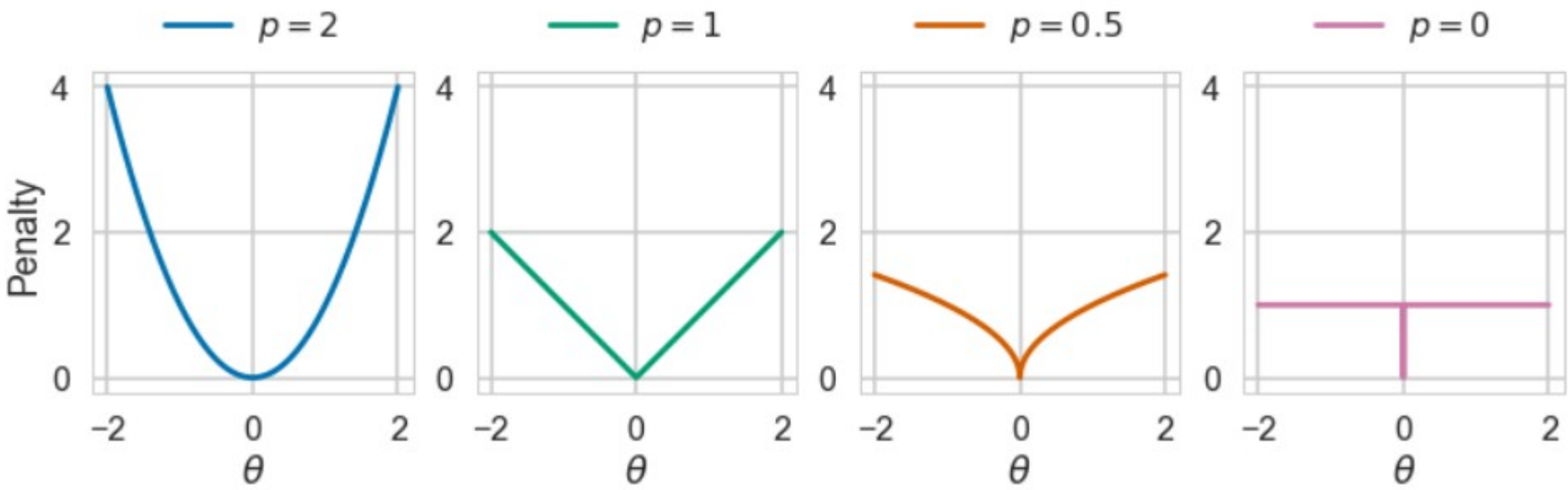
- Why pruning? Motivations and aims
- Voices from the past: pruning in the ‘90s
  - Skeletonization [Mozer and Smolensky, NIPS 1989]
  - Optimal brain damage [Le Cun et al., NIPS 1990]
- A revival of pruning (2015 & beyond)
  - Learning both weights and connections [Han et al., NIPS 2015]
  - Variational dropout sparsifies deep neural networks [Molchanlov et al., ICML 2017]
  - Learning sparse networks using targeted dropout [Gomez et al., NeurIPS 2019]
  - Learning sparse neural networks via sensitivity-driven regularization [Tartaglione et al., NeurIPS 2018]
  - Learning sparse neural networks through L0 regularization [Louizos et al., ICLR 2018]
- The lottery ticket hypothesis [Frankle and Carbin, ICLR 2019]
  - SNIP: Single-shot network pruning based on connection sensitivity [Lee et al., ICLR 2019]
  - Can we prune at initialization?
- Structured vs unstructured pruning

Learning sparse neural networks through L0 regularization [Louizos et al., 2018]

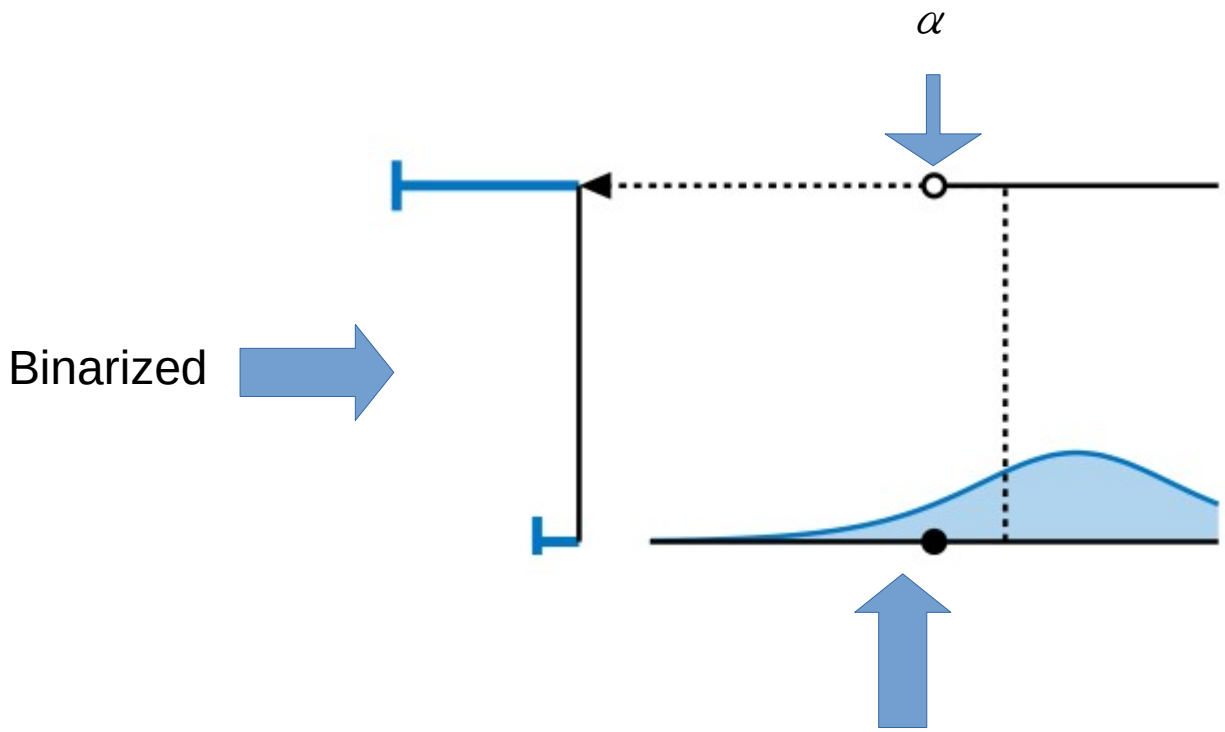
- L0 is a non-differentiable metric… how to deal with it?

- Introduced here the concept of **non-negative stochastic gates.**
- For a specific distribution of the gates (hard concrete) the measure becomes differentiable
- The objective here is to find the largest subset of parameters whose pruning will cause the least increase of E.

Learning sparse neural networks through L0 regularization [Louizos et al., 2018]

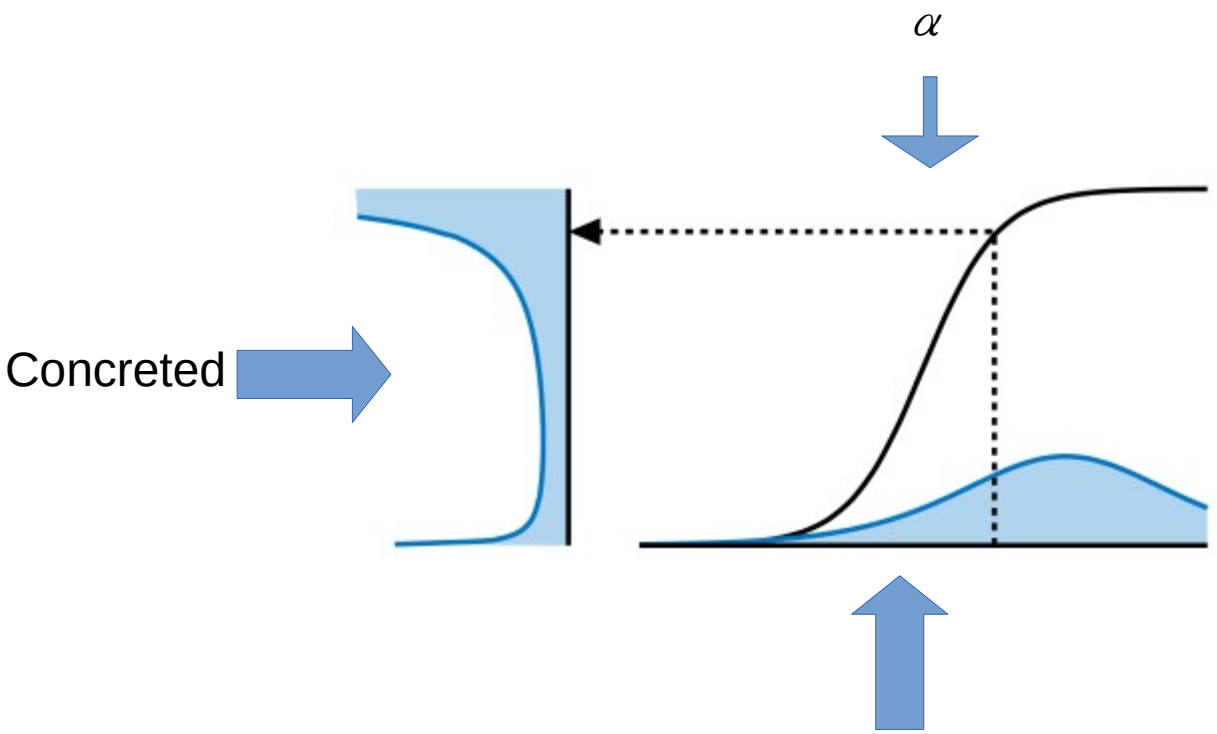


The concrete distribution [Maddison et al., ICLR 2017]



Case zero temperature (λ=0)

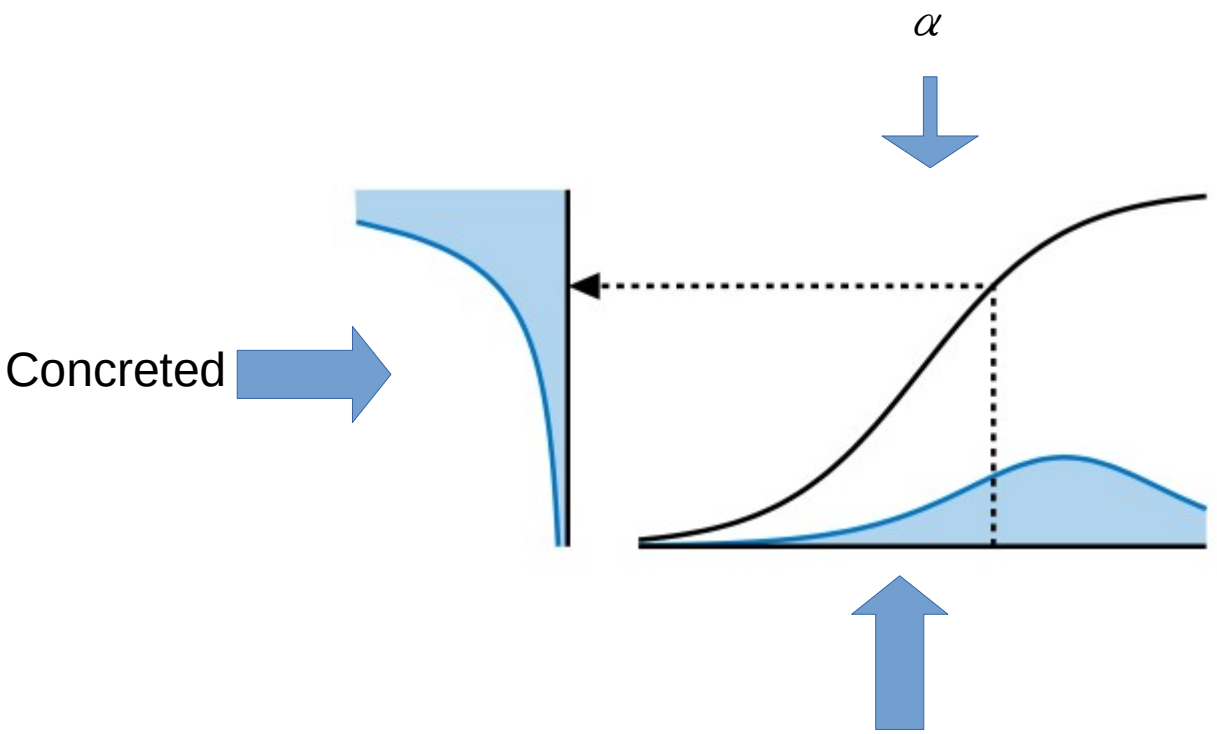
The concrete distribution [Maddison et al., ICLR 2017]



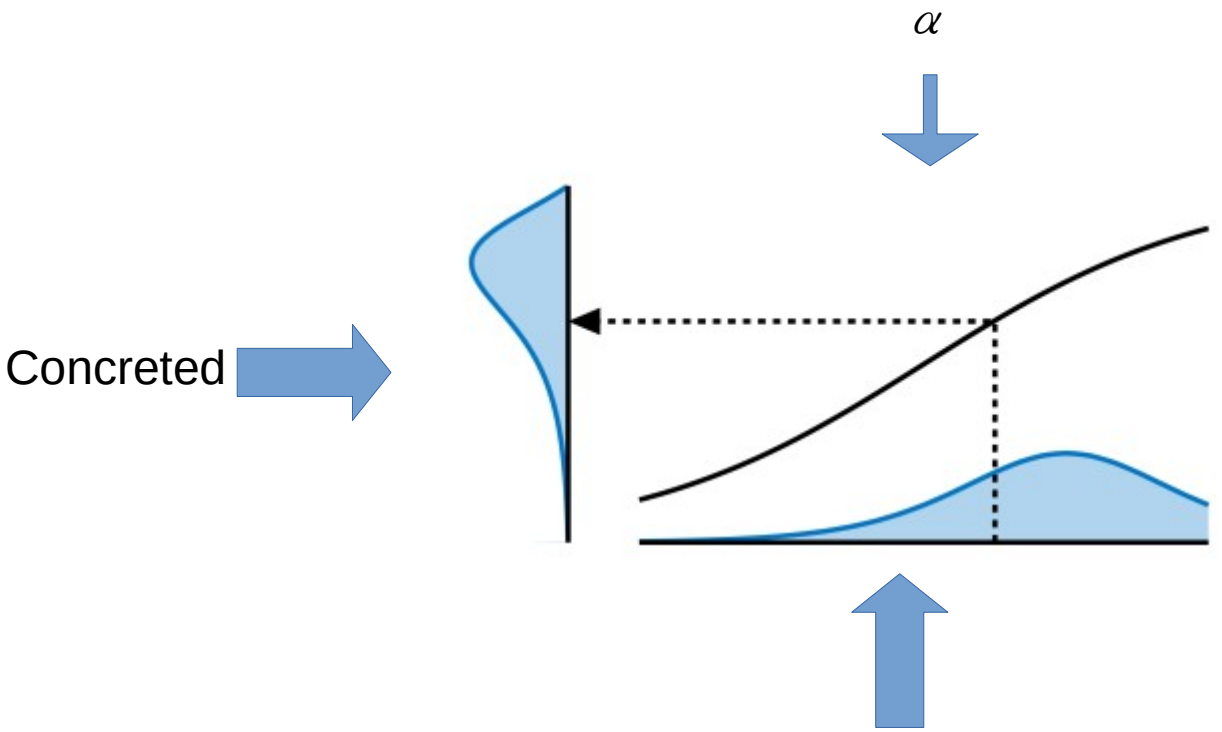
Continuous distribution

λ=0.5

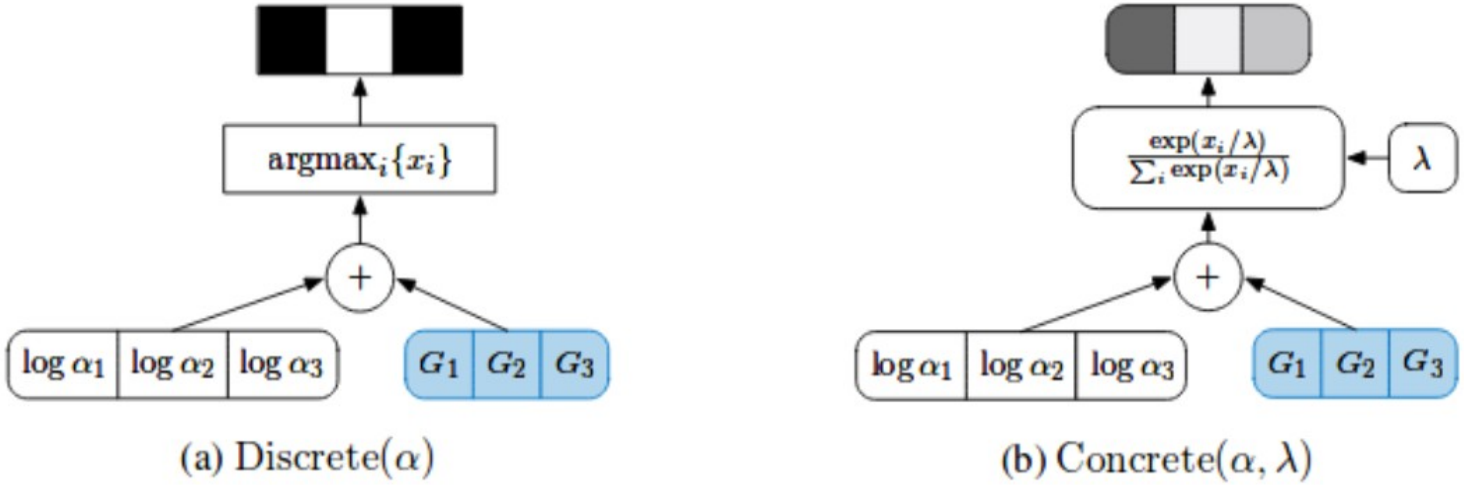
The concrete distribution [Maddison et al., ICLR 2017]



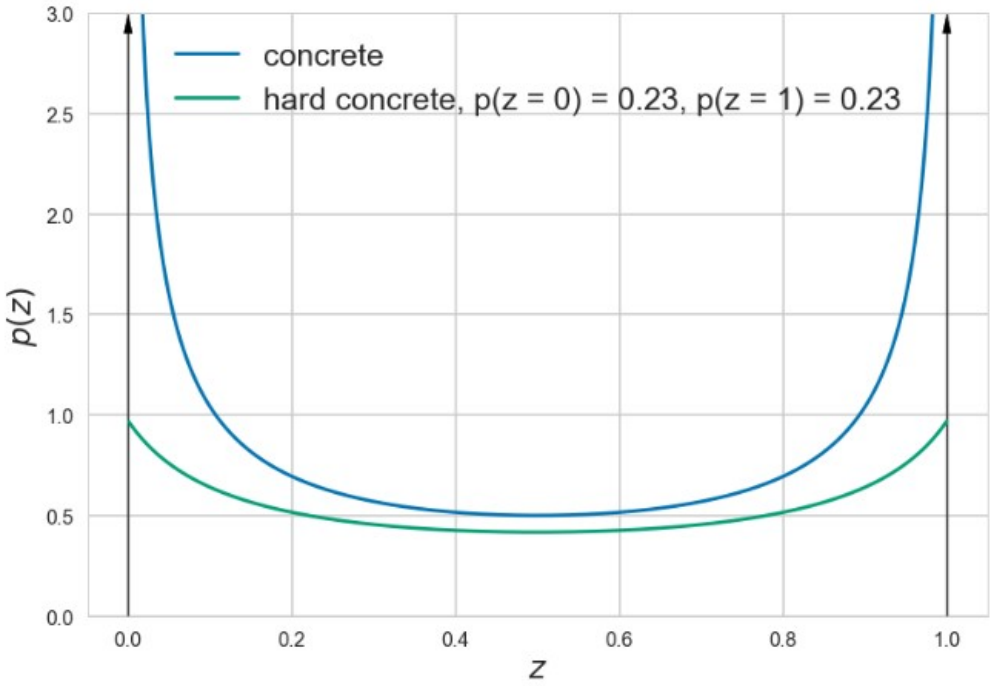
The concrete distribution [Maddison et al., ICLR 2017]

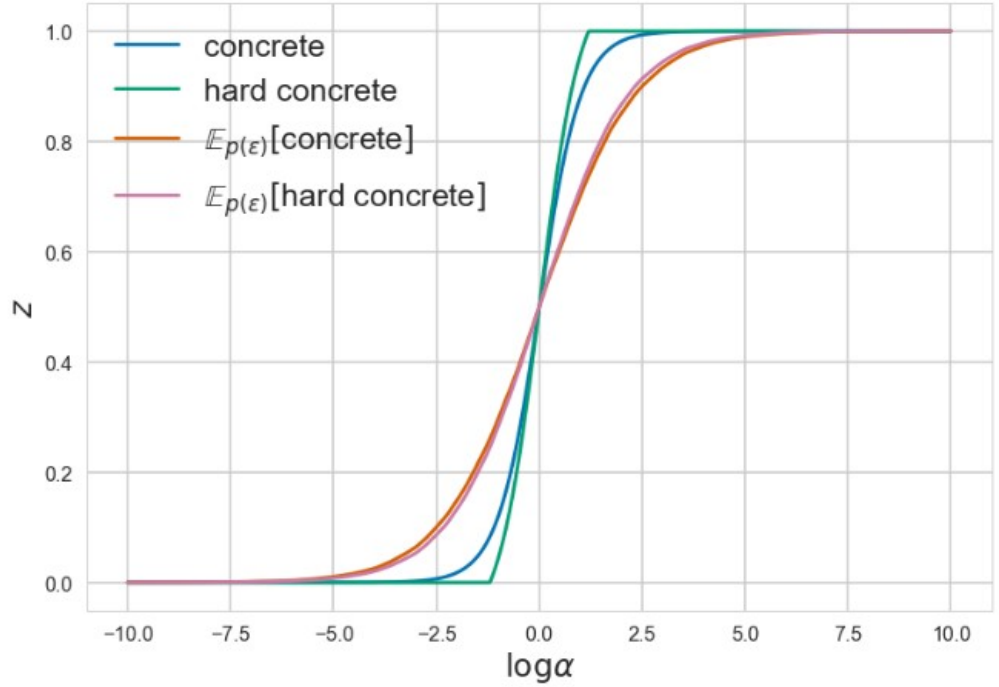


The concrete distribution [Maddison et al., ICLR 2017]



The hard concrete distribution [Louizos et al., ICLR 2018]





$$\mathcal{L}_C = \sum_{j=1}^{\cdot\cdot\cdot} \left(1 - Q_{\bar{s}_j}(0|\phi)\right) = \sum_{j=1}^{\cdot\cdot\cdot} \text{Sigmoid}\left(\log \alpha_j - \beta \log \frac{-\gamma}{\zeta}\right).$$

Cumulative distribution

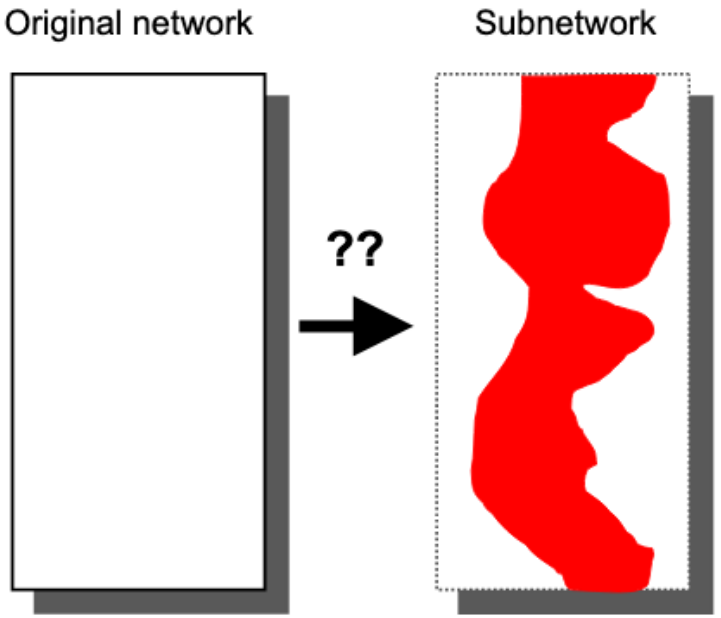
Learning sparse neural networks through L0 regularization [Louizos et al., 2018]

Network & size	Method	Pruned architecture	Error (%)
MLP 784-300-100	Sparse VD (Molchanov et al., 2017)	512-114-72	1.8
	BC-GNJ (Louizos et al., 2017)	278-98-13	1.8
	BC-GHS (Louizos et al., 2017)	311-86-14	1.8
	$L_{0_{hc}}, \lambda = 0.1/N$	219-214-100	1.4
	$L_{0_{hc}}, \lambda$ sep.	266-88-33	1.8
LeNet-5-Caffe 20-50-800-500	Sparse VD (Molchanov et al., 2017)	14-19-242-131	1.0
	GL (Wen et al., 2016)	3-12-192-500	1.0
	GD (Srinivas & Babu, 2016)	7-13-208-16	1.1
	SBP (Neklyudov et al., 2017)	3-18-284-283	0.9
	BC-GNJ (Louizos et al., 2017)	8-13-88-13	1.0
	BC-GHS (Louizos et al., 2017)	5-10-76-16	1.0
	$L_{0_{hc}}, \lambda = 0.1/N$	20-25-45-462	0.9
	$L_{0_{hc}}, \lambda$ sep.	9-18-65-25	1.0

Pruning at initialization??
The lottery ticket hypothesis

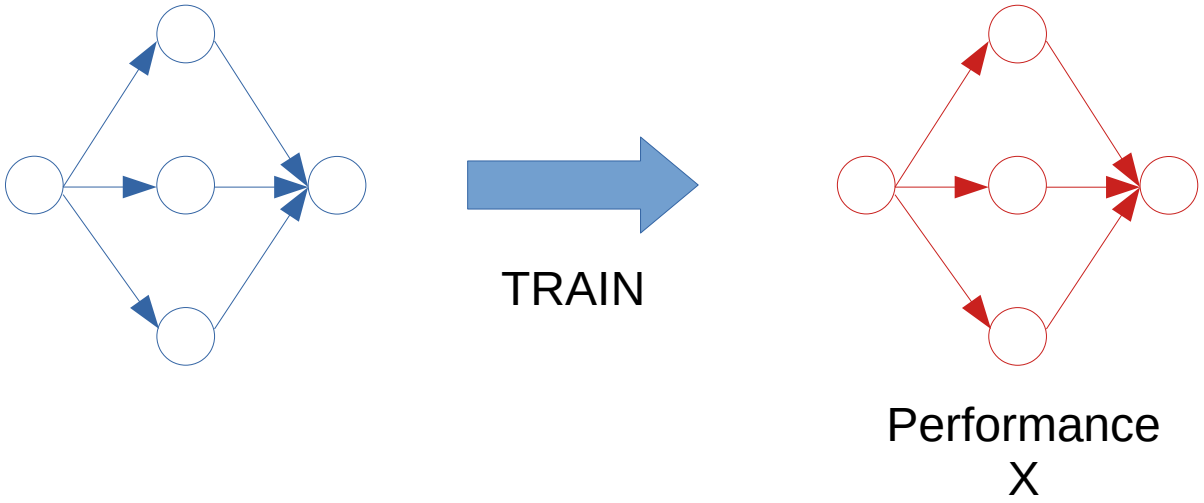
The lottery ticket hypothesis:
finding sparse, trainable neural networks [Frankle and Carbin, ICLR 2019]

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



https://users.aalto.fi/~falconr1/RecentAdvances2019/The%20Lottery%20Ticket%20Hypothesis%20slides.pdf

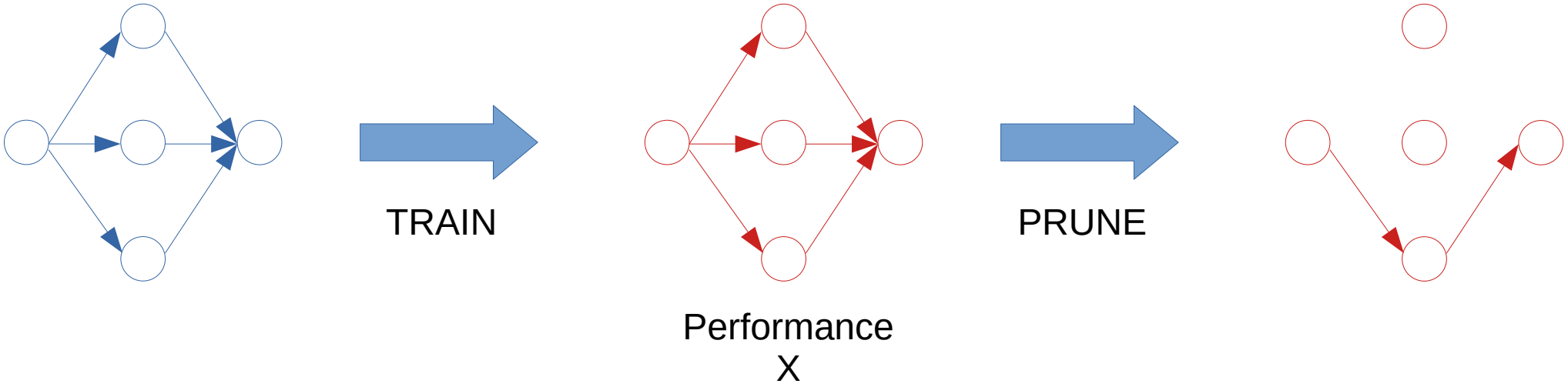
The lottery ticket hypothesis:
finding sparse, trainable neural networks [Frankle and Carbin, ICLR 2019]



BEFORE TRANING

AFTER TRAINING

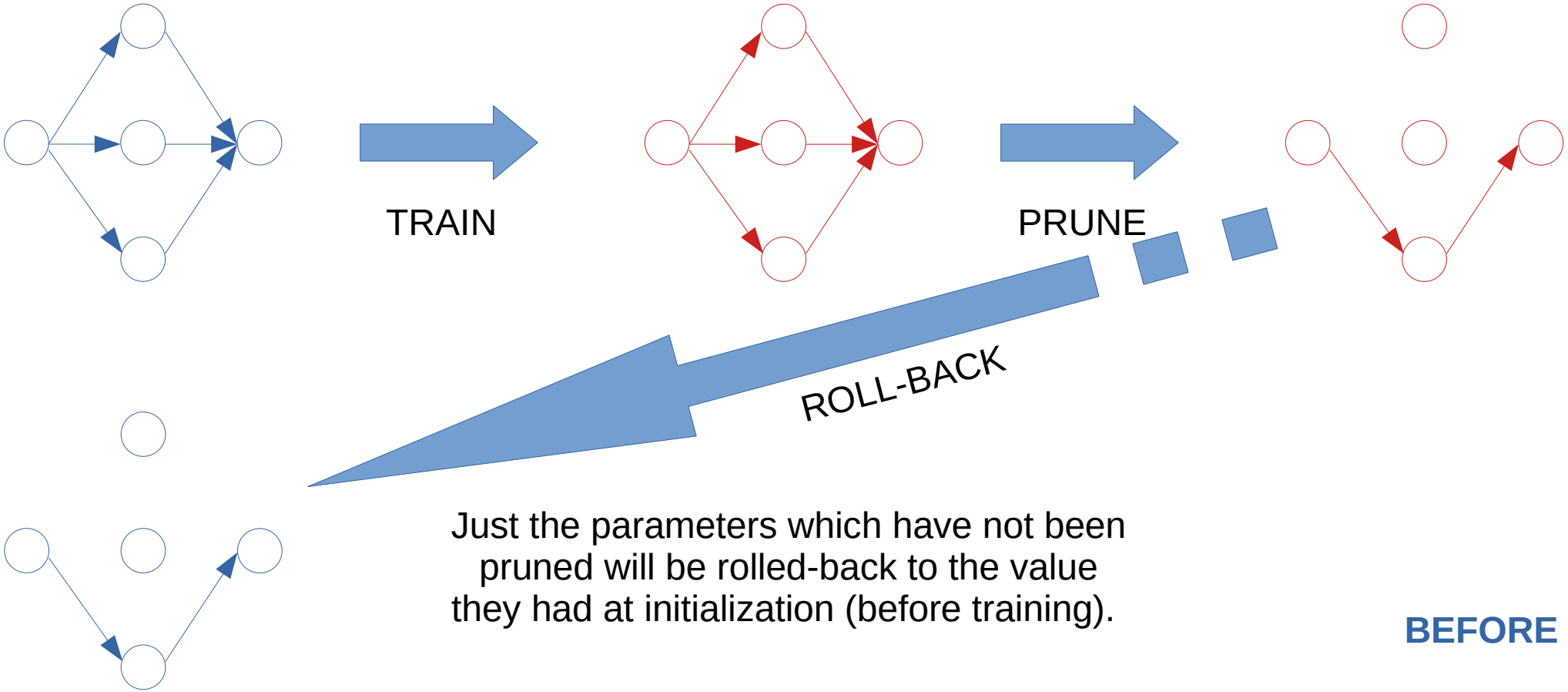
The lottery ticket hypothesis: finding sparse, trainable neural networks [Frankle and Carbin, ICLR 2019]



BEFORE TRANING

AFTER TRAINING

The lottery ticket hypothesis: finding sparse, trainable neural networks [Frankle and Carbin, ICLR 2019]

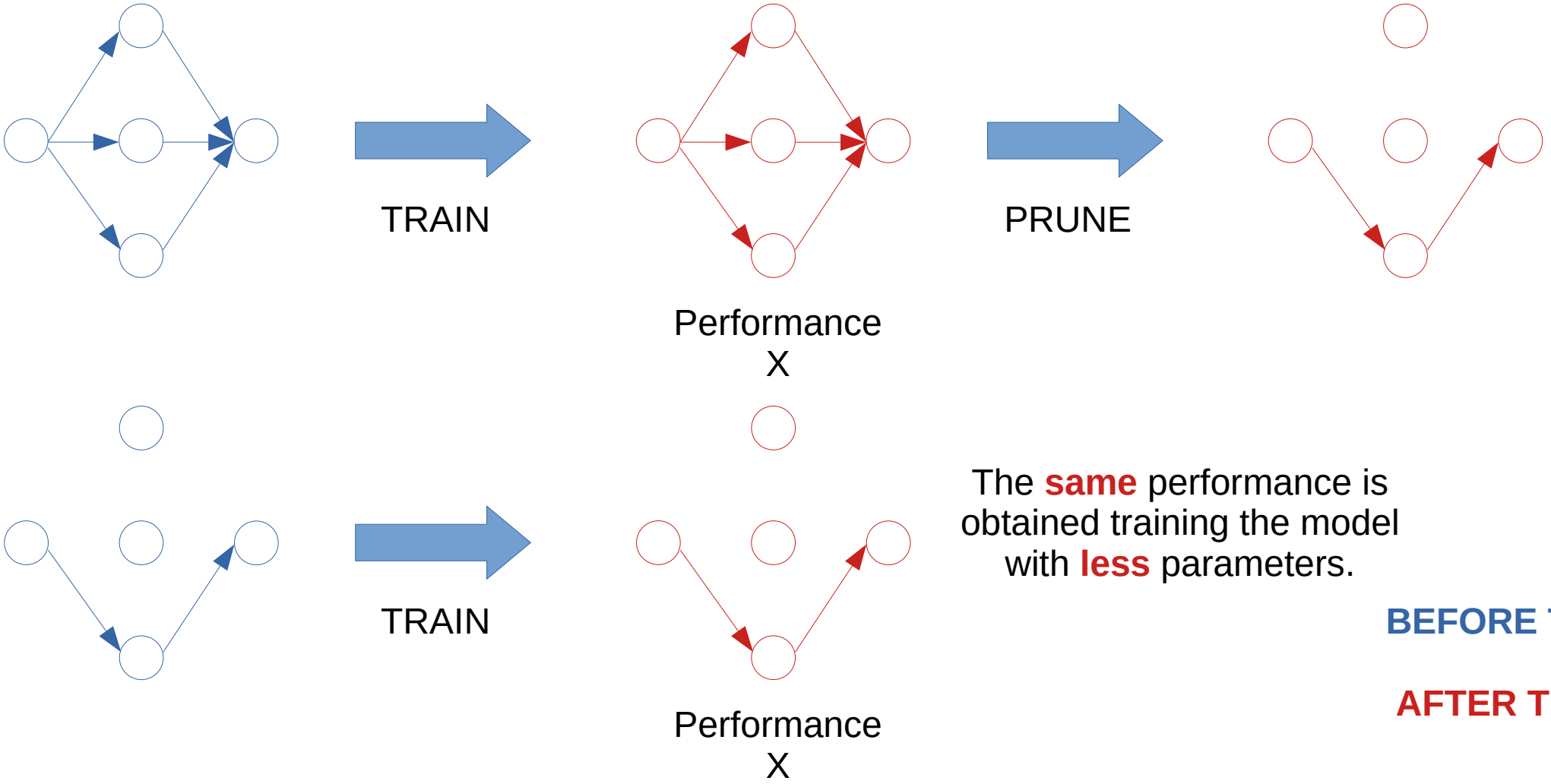


Just the parameters which have not been pruned will be rolled-back to the value they had at initialization (before training).

BEFORE TRANING

AFTER TRAINING

The lottery ticket hypothesis: finding sparse, trainable neural networks [Frankle and Carbin, ICLR 2019]



The **same** performance is obtained training the model with **less** parameters.

BEFORE TRANING

AFTER TRAINING

The lottery ticket hypothesis: finding sparse, trainable neural networks [Frankle and Carbin, ICLR 2019]

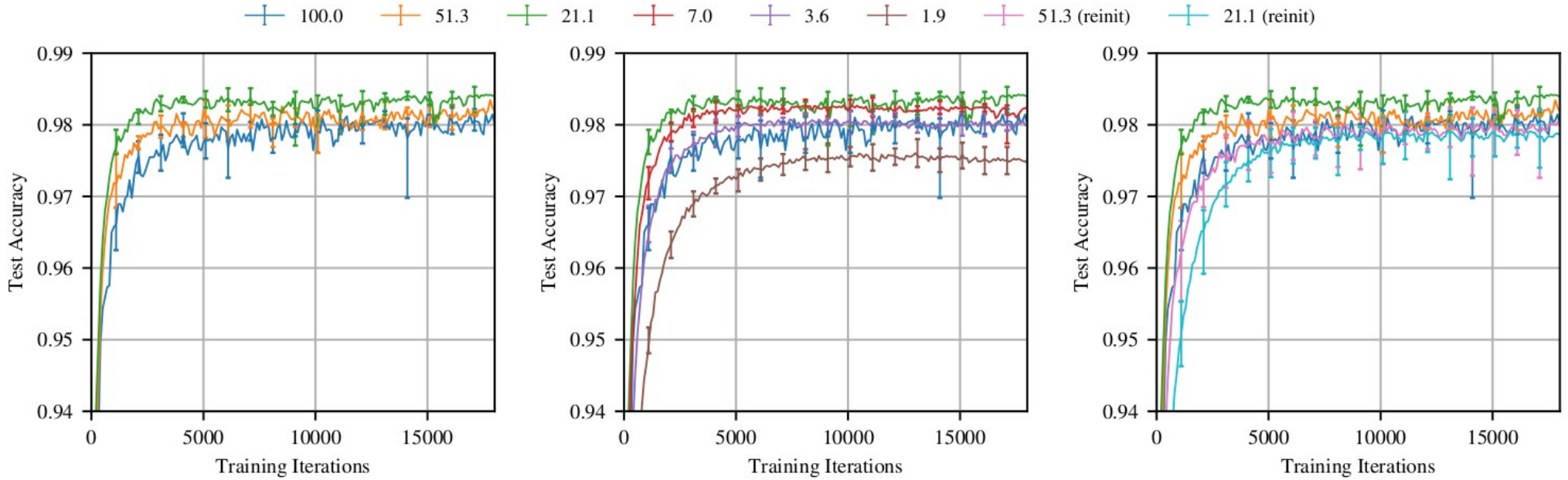


Figure 3: Test accuracy on Lenet (iterative pruning) as training proceeds. Each curve is the average of five trials. Labels are P_m—the fraction of weights remaining in the network after pruning. Error bars are the minimum and maximum of any trial.

The lottery ticket hypothesis: finding sparse, trainable neural networks [Frankle and Carbin, ICLR 2019]

- So, there are parameters “winning at the lottery of initialization” in the deep models…
- It is possible to successfully train a model from the initialization phase removing the largest part of the parameters and training just the remaining fraction!
- In this work, just their **existence** (with some a-posteriori evaluation) has been shown… however, the method on how to identify them at initialization (or in the fist learning stages) has not been proposed (yet in this work).
- **A RACE** for pruning models at initialization has begun…

Outline for PART 1

- Why pruning? Motivations and aims
- Voices from the past: pruning in the ‘90s
  - Skeletonization [Mozer and Smolensky, NIPS 1989]
  - Optimal brain damage [Le Cun et al., NIPS 1990]
- A revival of pruning (2015 & beyond)
  - Learning both weights and connections [Han et al., NIPS 2015]
  - Variational dropout sparsifies deep neural networks [Molchanlov et al., ICML 2017]
  - Learning sparse networks using targeted dropout [Gomez et al., NeurIPS 2019]
  - Learning sparse neural networks via sensitivity-driven regularization [Tartaglione et al., NeurIPS 2018]
  - Learning sparse neural networks through L0 regularization [Louizos et al., ICLR 2018]
- The lottery ticket hypothesis [Frankle and Carbin, ICLR 2019]
  - SNIP: Single-shot network pruning based on connection sensitivity [Lee et al., ICLR 2019]
  - Can we prune at initialization?
- Structured vs unstructured pruning

SNIP: Single-shot network pruning based on connection sensitivity [Lee et al., ICLR 2019]

- Instead of running an iterative algorithm involving regularization+pruning, we prune just once.
- They perform that at initialization using a very similar concept as [Tartaglione et al, NeurIPS 2018], with an additional softmax averaging term.

$$\Delta L_j(\mathbf{w};\mathcal{D}) \approx g_j(\mathbf{w};\mathcal{D}) = \left.\frac{\partial L(\mathbf{c}\odot\mathbf{w};\mathcal{D})}{\partial c_j}\right|_{\mathbf{c}=\mathbf{1}} \qquad s_j = \frac{|g_j(\mathbf{w};\mathcal{D})|}{\sum_{k=1}^{m}|g_k(\mathbf{w};\mathcal{D})|}$$

- Then, the ranking of the parameters to remove is not magnitude-based, but sensitivity-based

- This is performed right after initialization (hence, the main direction of the gradient is preserved).

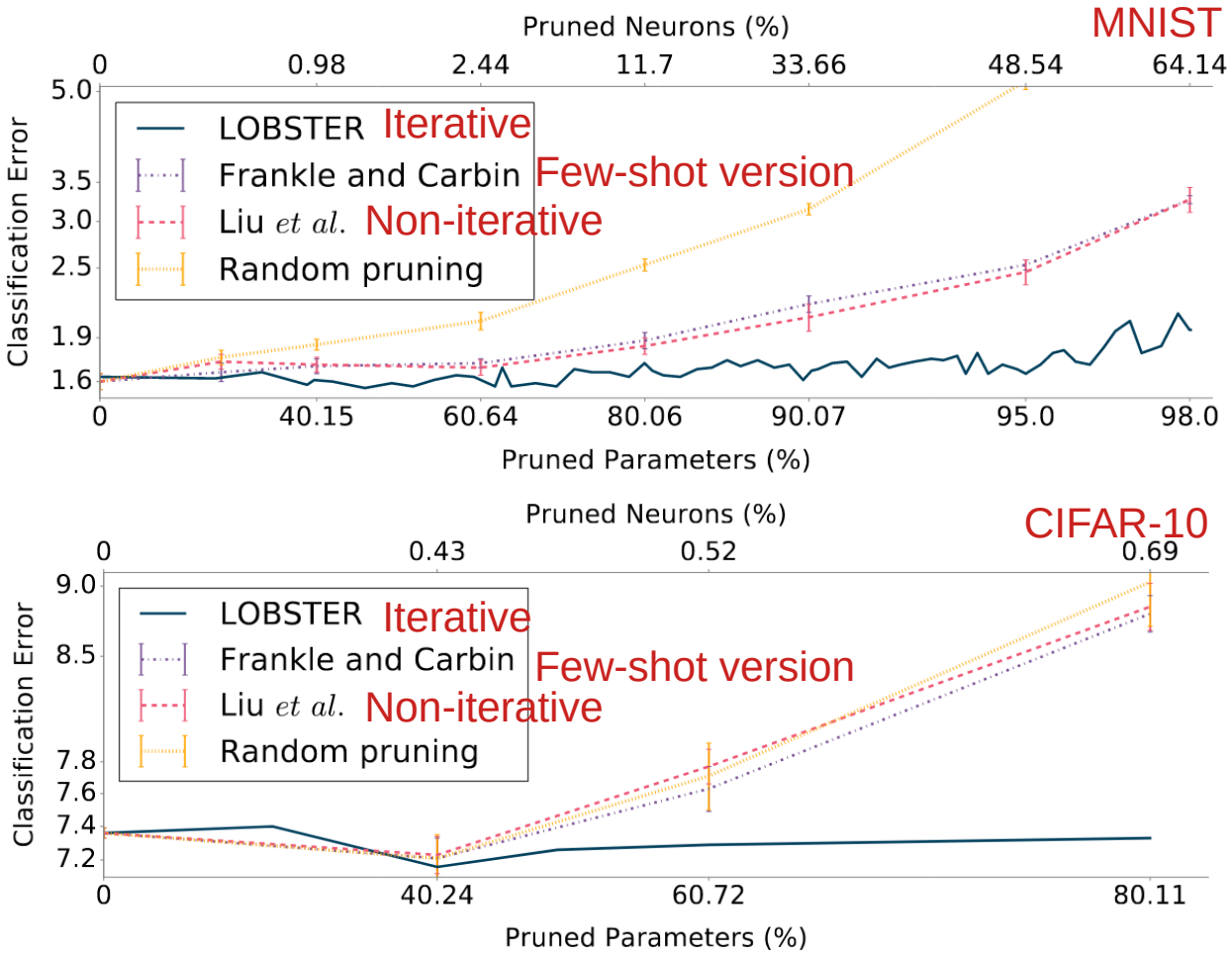
SNIP: Single-shot network pruning based on connection sensitivity [Lee et al., ICLR 2019]

- Also this technique works on simple datasets

Method	Criterion	LeNet-300-100 $\bar{\kappa}$ (%)	LeNet-300-100 err. (%)	LeNet-5-Caffe $\bar{\kappa}$ (%)	LeNet-5-Caffe err. (%)	Pretrain	# Prune	Additional hyperparam.	Augment objective	Arch. constraints
Ref.	–	–	1.7	–	0.9	–	–	–	–	–
LWC	Magnitude	91.7	**1.6**	91.7	**0.8**	✓	many	✓	✗	✓
DNS	Magnitude	98.2	2.0	99.1	0.9	✓	many	✓	✗	✓
LC	Magnitude	99.0	3.2	99.0	1.1	✓	many	✓	✓	✗
SWS	Bayesian	95.6	1.9	99.5	1.0	✓	soft	✓	✓	✗
SVD	Bayesian	98.5	1.9	99.6	**0.8**	✓	soft	✓	✓	✗
OBD	Hessian	92.0	2.0	92.0	2.7	✓	many	✓	✗	✗
L-OBS	Hessian	98.5	2.0	99.0	2.1	✓	many	✓	✗	✓
SNIP (ours)	Connection sensitivity	95.0 98.0	**1.6** 2.4	98.0 99.0	**0.8** 1.1	✗	**1**	✗	✗	✗

Iterative VS non-iterative algorithms: a comparison

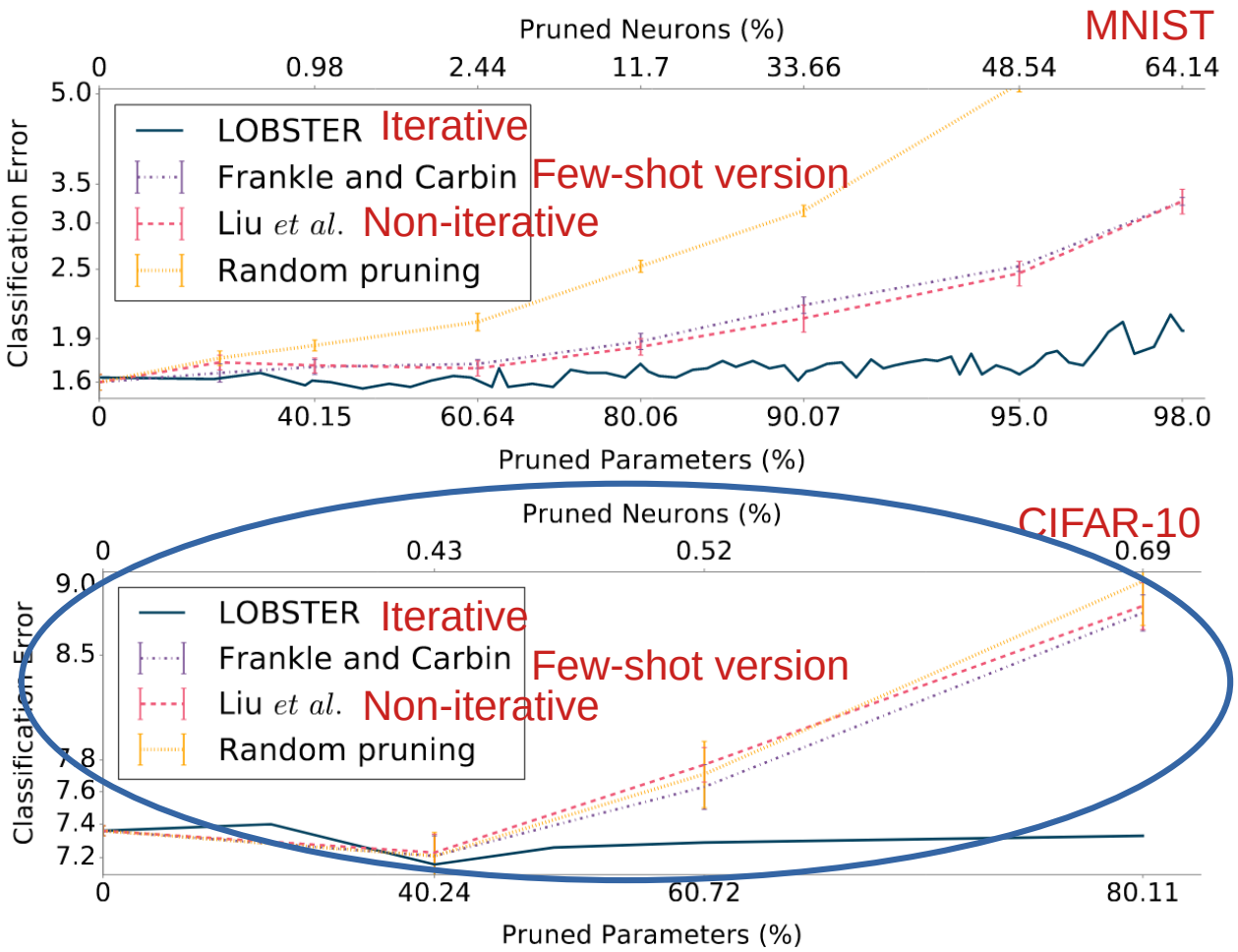
- LOBSTER as representant of iterative pruning algorithms
- Frankle and Carbin as representants of few-shot algorithms
- At high pruning rates, the iterative algorithms achieve better generalization (at higher training costs)



Images taken from Tartaglione, Enzo, Andrea Bragagnolo, and Marco Grangetto. "Pruning artificial neural networks: A way to find well-generalizing, high-entropy sharp minima." International Conference on Artificial Neural Networks. Springer, Cham, 2020.

Iterative VS non-iterative algorithms: a comparison

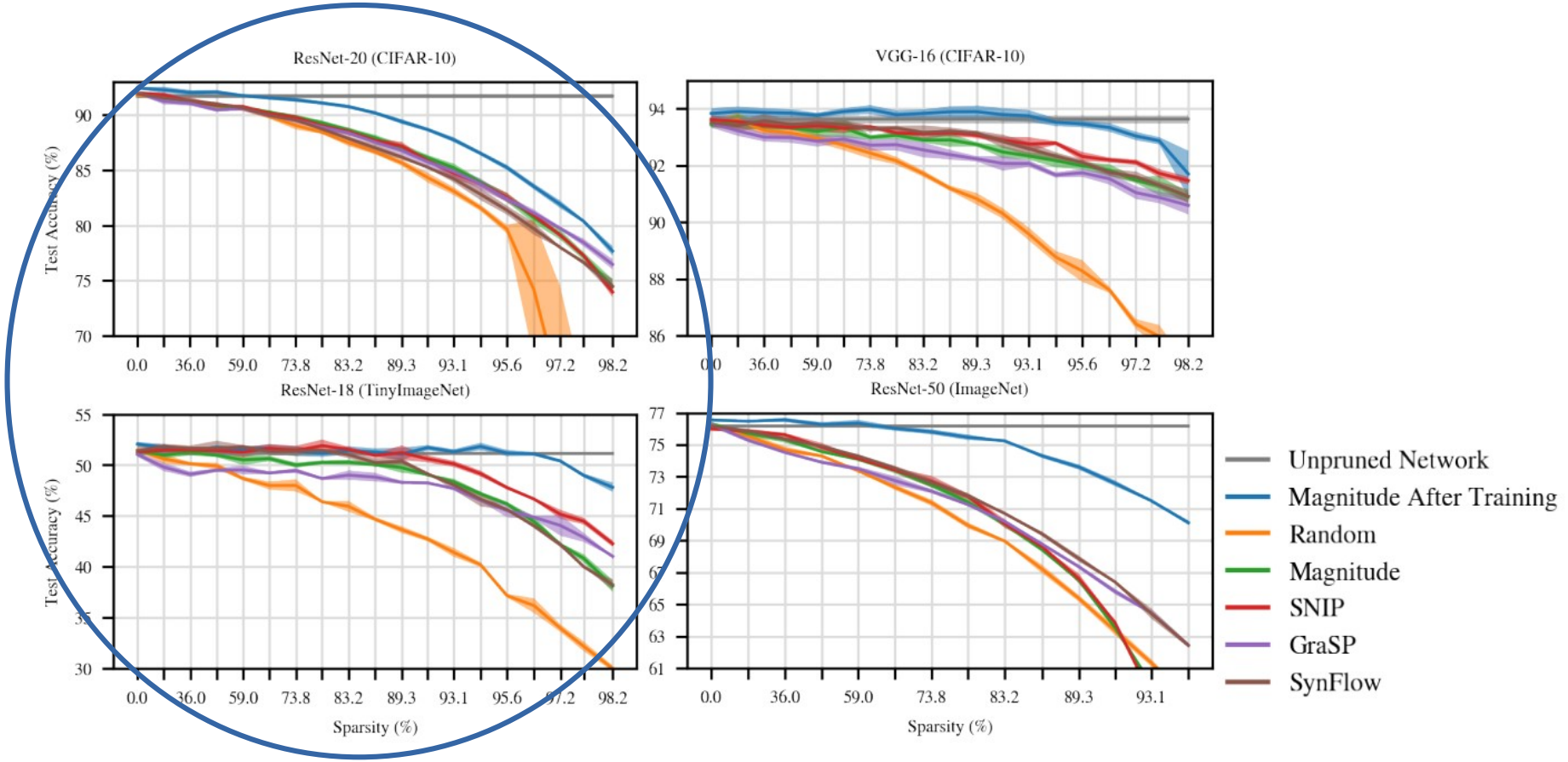
- LOBSTER as representant of iterative pruning algorithms
- Frankle and Carbin as representants of few-shot algorithms
- At high pruning rates, the iterative algorithms achieve better generalization (at higher training costs)



Images taken from Tartaglione, Enzo, Andrea Bragagnolo, and Marco Grangetto. "Pruning artificial neural networks: A way to find well-generalizing, high-entropy sharp minima." International Conference on Artificial Neural Networks. Springer, Cham, 2020.

Pruning the neural networks at initialization: why are we missing the mark? [Frankle et al., ICLR 2021]

- Benchmark of some approaches of pruning-at-initialization recently proposed…
- NONE is state-of-the-art as just one re-training is consistently better… why?



The early phase of neural network training [Frankle et al., ICLR 2020]

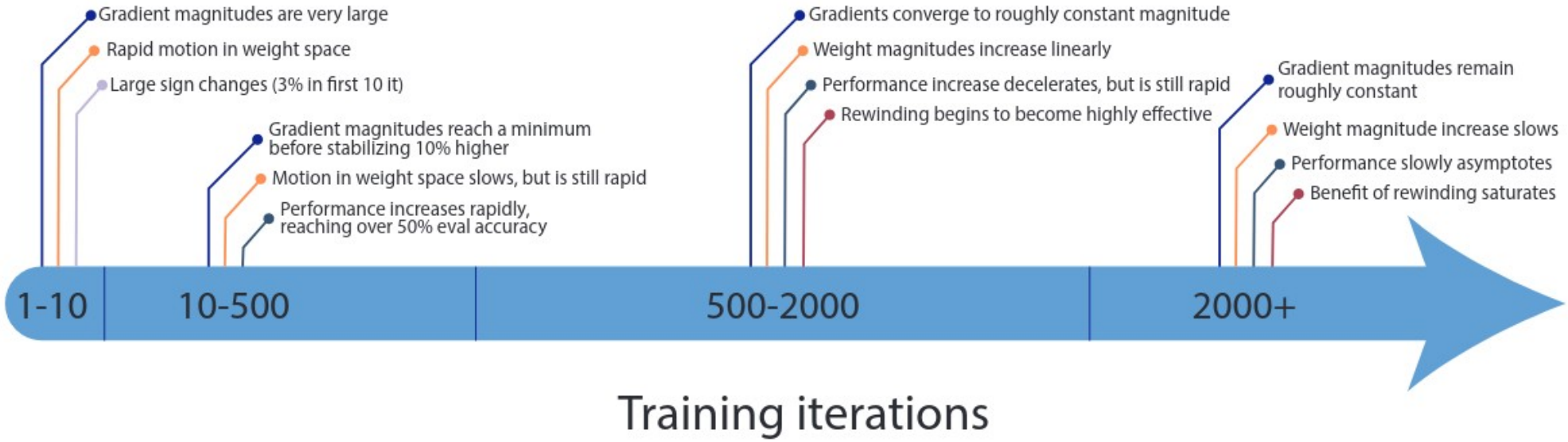


Figure 2: Rough timeline of the early phase of training for ResNet-20 on CIFAR-10.

The rise of the lottery heroes: why zero-shot pruning is hard [Tartaglione, 2022]

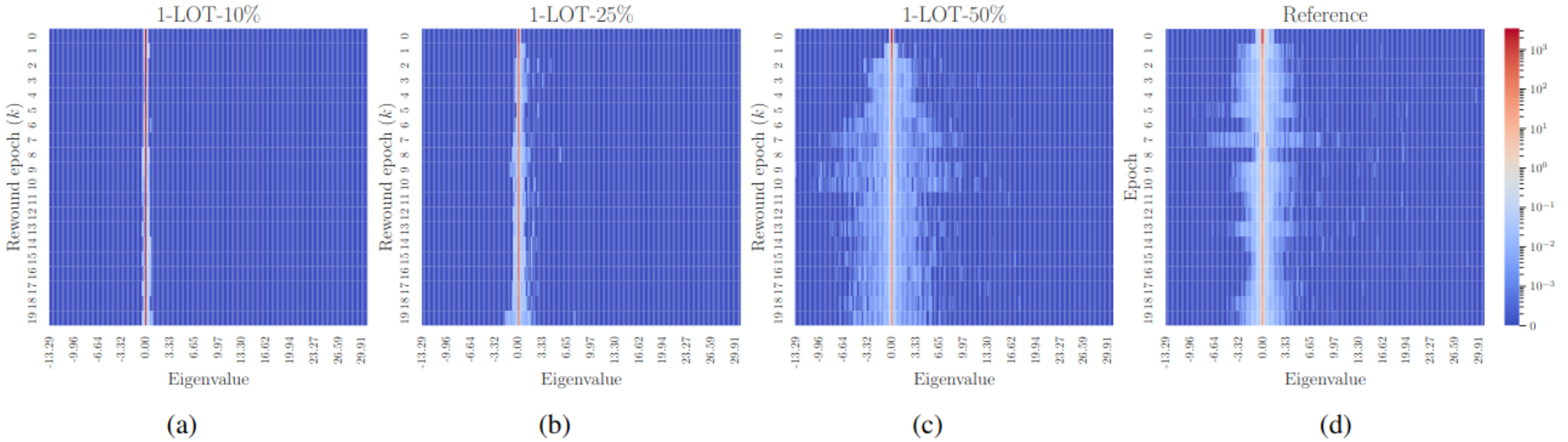
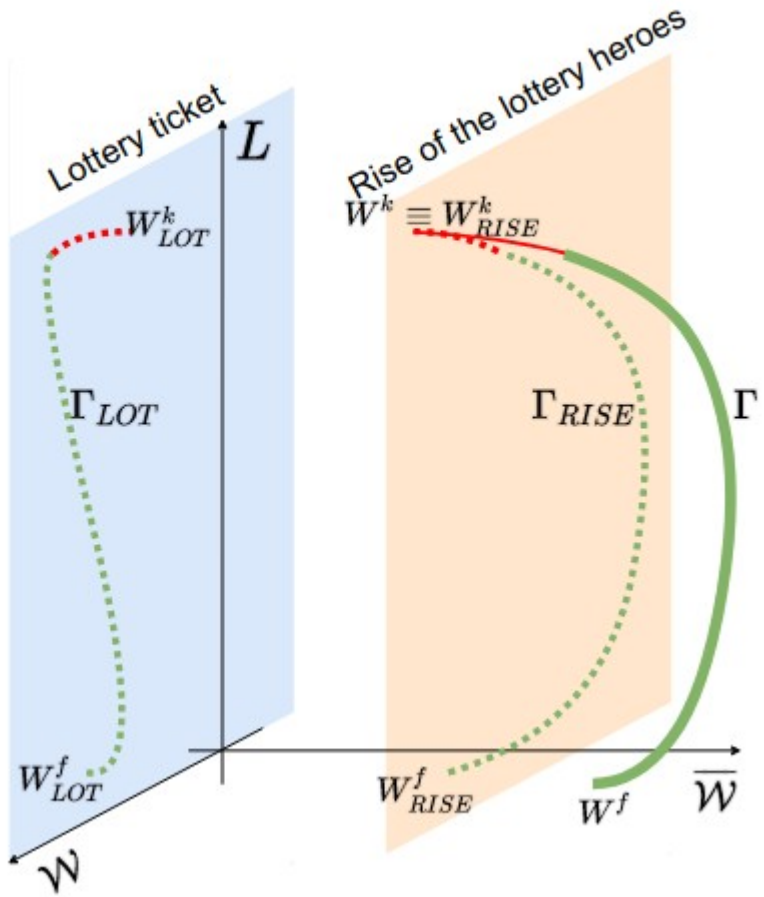


Fig. 2: Example of distribution of the eigenvalues of the Hessian matrix calculated on the CIFAR-10 training set for ResNet-32 along different rewound epochs (k) retaining the 10% of the parameters (a), the 25% (b) the 50% (c) and all the parameters (d). Here $I = 1$. The represented scenario is qualitatively matched for different initializations of the model.

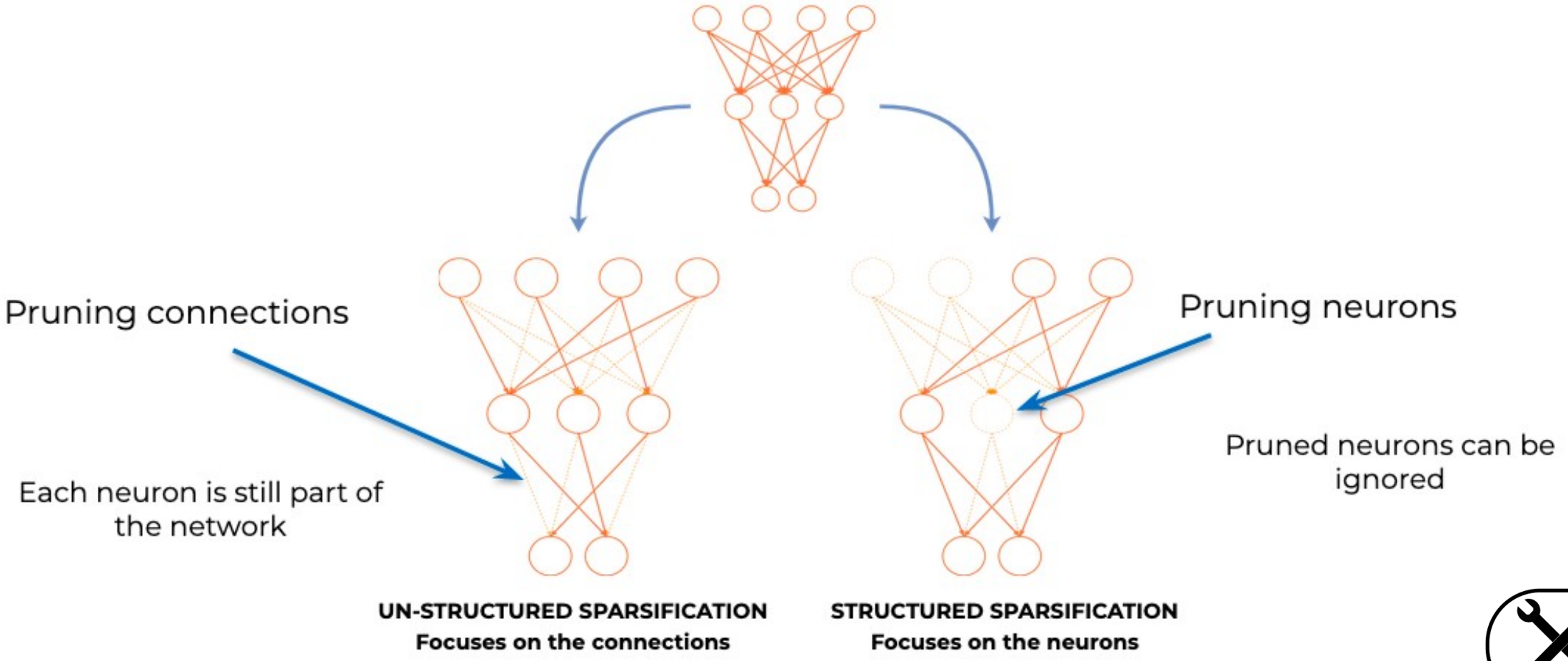
The rise of the lottery heroes: why zero-shot pruning is hard [Tartaglione, 2022]

- When removing parameters, we are projected in a sub-space where to optimize (in blue).
- Despite the minimum exists, the optimization problem itself becomes harder.
- Hence, this is a limit of the current optimization strategies!



Structured VS Unstructured sparsity

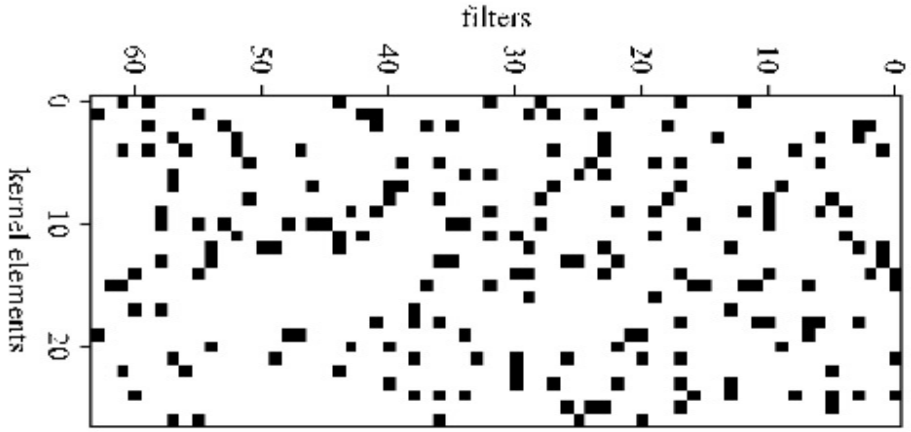
Structured vs Unstructured sparsity



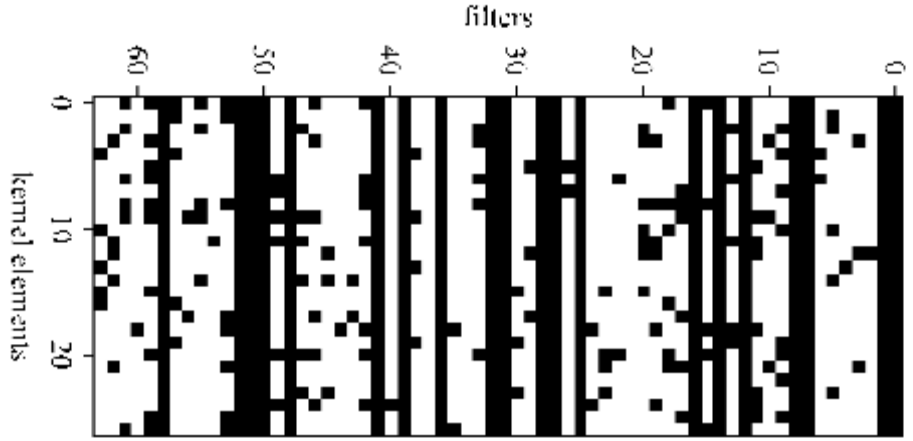
Structured vs unstructured sparsity

Looking at the internal parameters in a layer….

(black=pruned)

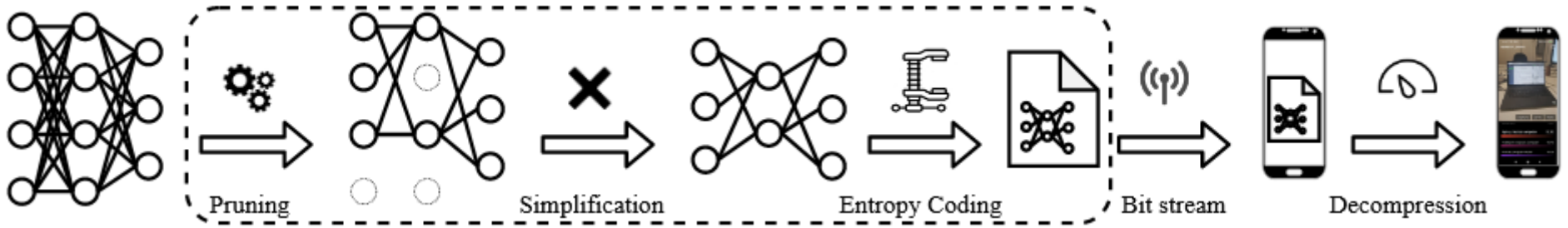


UNSTRUCTURED sparsity



STRUCTURED sparsity

Testing on embedded devices



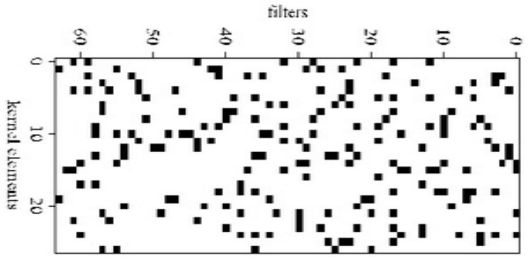
- Dashed: neural network compression pipeline in the MPEG-7 part 17 standard.
- Simplification code is open-source and available at https://github.com/EIDOSlab/simplify
- As Entropy-coding one of the possibilities is to use DeepCABAC [Wiedemann et al., 2019]

Comparing structured and unstructured pruning

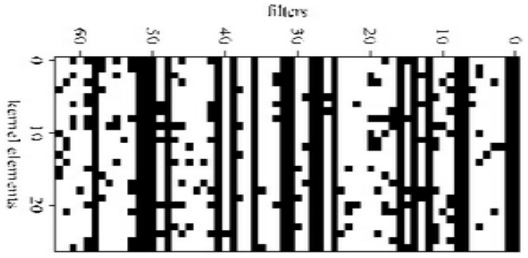
Dataset	Architecture	Pruning	Pruning ratio [%]	Simplified topology [MB]	Compressed bitstream [MB]	Inference time [ms] RPi 3B	P20	MI9	S6L	
CIFAR-10	VGG-16	No pruning	-	60.0	3.6	647	204	153	251	
		LOBSTER	**92.44**	58.61	1.61	610	191	146	242	Unstructured
		SeReNe	47.16	**31.02**	**0.34**	**594**	**99**	**85**	**106**	Structured
	ResNet-32	No pruning	-	2.0	0.30	580	32	30	31	
		LOBSTER	**81.19**	1.96	0.12	545	32	26	30	Unstructured
		SeReNe	52.80	**1.0**	**0.09**	**536**	**25**	**17**	**25**	Structured
CIFAR-100	AlexNet	No pruning	-	94.6	10.1	246	131	84	168	
		LOBSTER	**98.90**	48.84	0.40	224	95	67	120	Unstructured
		SeReNe	59.87	**37.07**	**0.20**	**186**	**75**	**53**	**96**	Structured

Bragagnolo, A., Tartaglione, E., Fiandrotti, A., & Grangetto, M. (2021, September). On the Role of Structured Pruning for Neural Network Compression. In *2021 IEEE International Conference on Image Processing (ICIP)* (pp. 3527-3531). IEEE.

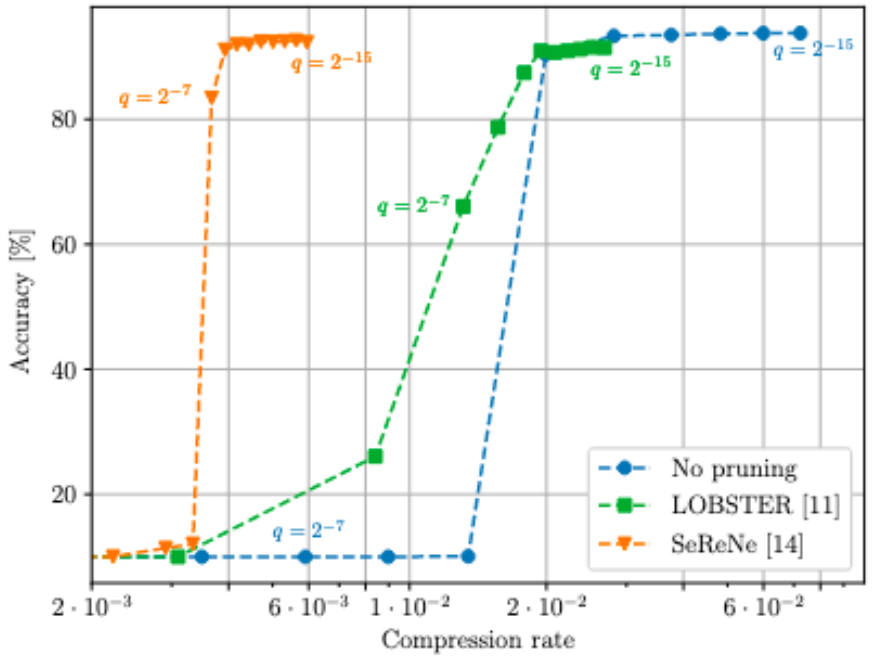
Comparing structured and unstructured pruning



Unstructured sparsity



Structured sparsity



VGG-16 trained on CIFAR-10. Compression using DeepCABAC with different quantization levels.

Bragagnolo, A., Tartaglione, E., Fiandrotti, A., & Grangetto, M. (2021, September). On the Role of Structured Pruning for Neural Network Compression. In *2021 IEEE International Conference on Image Processing (ICIP)* (pp. 3527-3531). IEEE.

What are the current challenges?

- Pruning in order to achieve some **REAL ADVANTAGE** on the final device (hence, structured sparsity)
- Currently, iterative algorithms achieve better results in terms of accuracy/compression than few-shot pruning… but at higher training cost.
- A lot of research is being done on pruning at initialization right now (towards saving of computation at training time – **frugalAI**).
- The new learning approach is more oriented towards fine-tuning + pruning, starting from pre-trained models. New paradigms involve pruning from pre-trained architectures.

Let’s move to the practical session...

Login to use the Jupyter notebook at

https://jupyter.opendeephealth.di.unito.it

But before...

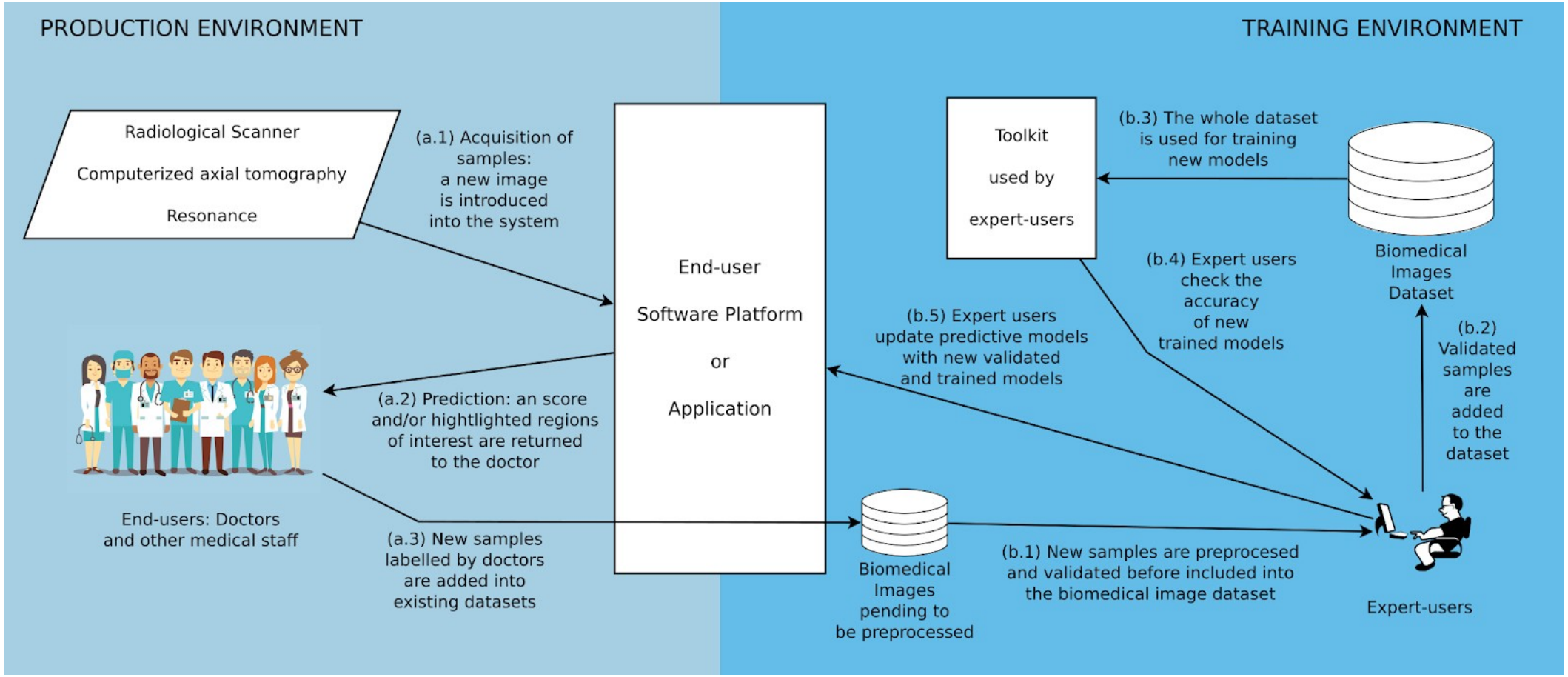
DEEPHEALTH

The DeepHealth project

22 partners from **9** **countries**: Research centers, Health organizations, large industries and SMEs

Duration: 36 months
Starting date: Jan 2019

The DeepHealth project




HPC4AI

The Reason

Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

CLOUD Act

From Wikipedia, the free encyclopedia

The **Clarifying Lawful Overseas Use of Data Act** or **CLOUD Act** (H.R. 4943) is a United States federal law enacted in 2018 by the passing of the Consolidated Appropriations Act, 2018, PL 115-141, section 105 executive agreements on access to data by foreign governments. Primarily the CLOUD Act amends the Stored Communications Act (SCA) of 1986 to allow federal law enforcement to compel U.S.-based technology companies via warrant or subpoena to provide requested data stored on servers regardless of whether the data are stored in the U.S. or on foreign soil.

Contents [hide]

1 Background
2 Support
3 Passing
4 References

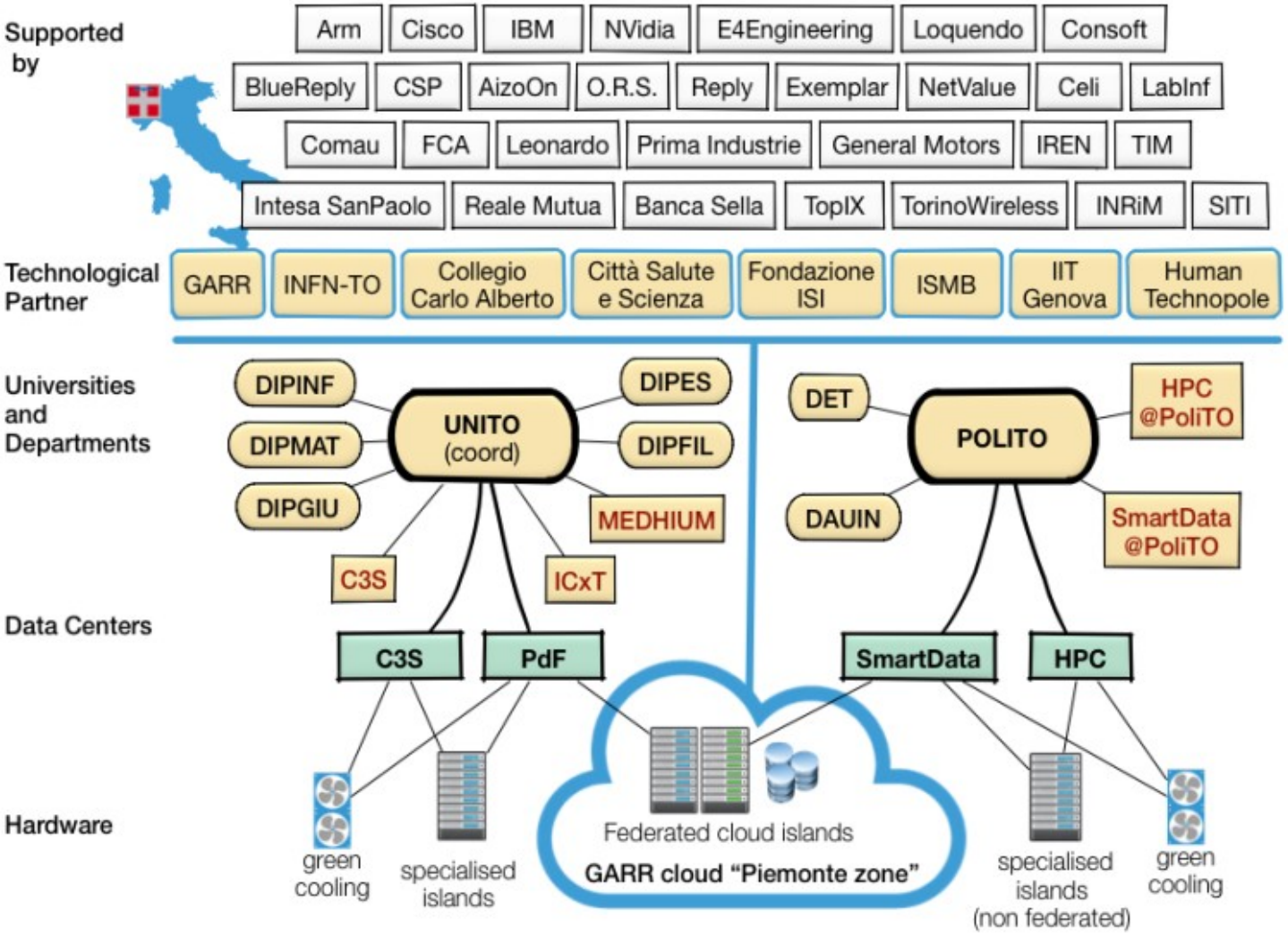
Background [edit]

The CLOUD Act was introduced following difficulties that the Federal Bureau of

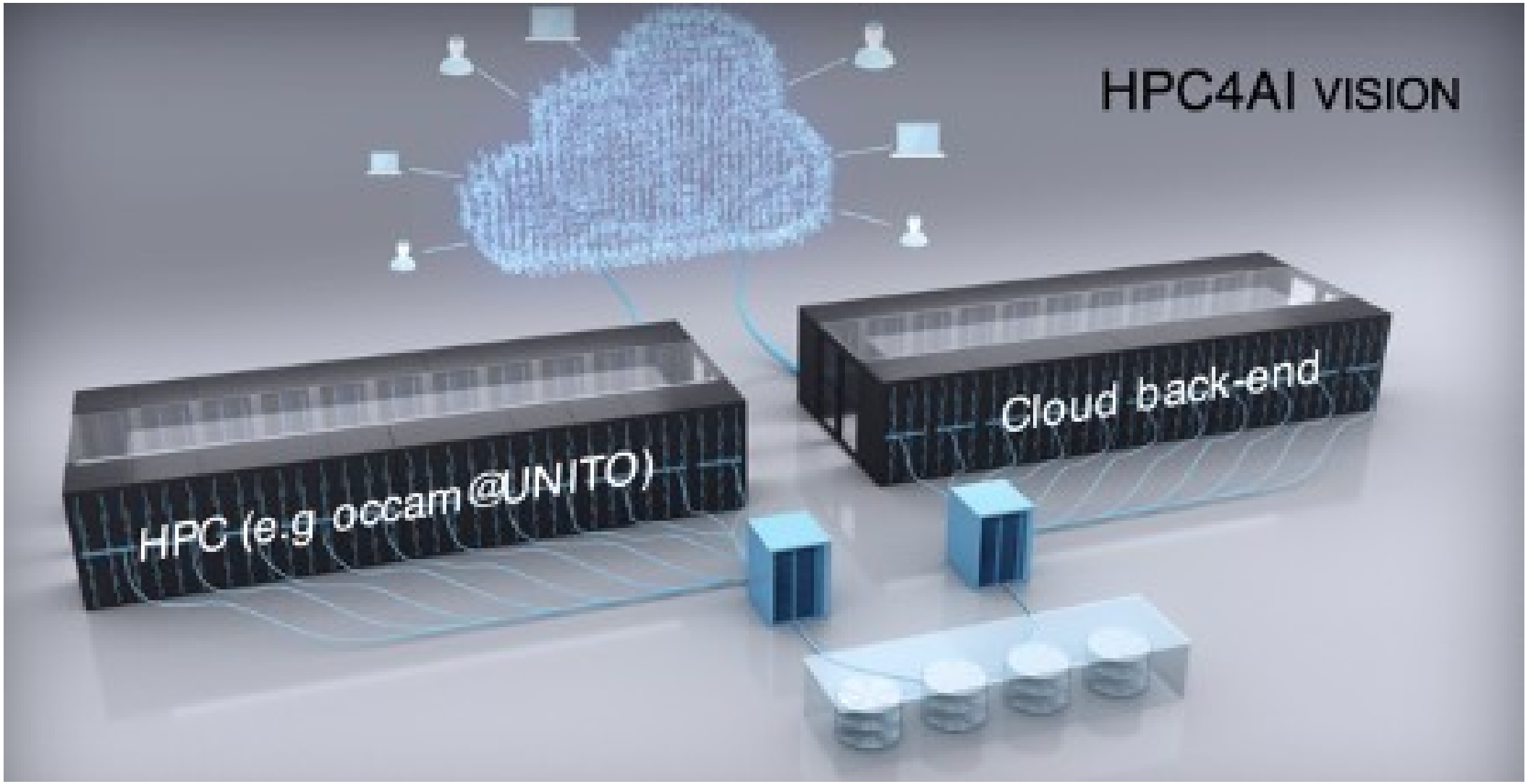
Clarifying Lawful Overseas Use of Data Act

Acronyms (colloquial)	CLOUD Act
Enacted by	the 115th United States Congress
Effective	March 23, 2018
Citations	
Public law	Pub.L. 115–141
Codification	
Acts amended	Stored Communications Act, Electronic Communications Privacy Act
Titles amended	18

The Project


Supported by
Arm
Cisco
IBM
NVidia
E4Engineering
Loquendo
Consoft
BlueReply
CSP
AizoOn
O.R.S.
Reply
Exemplar
NetValue
Celi
LabInf
Comau
FCA
Leonardo
Prima Industrie
General Motors
IREN
TIM
Intesa SanPaolo
Reale Mutua
Banca Sella
TopIX
TorinoWireless
INRiM
SITI
Technological Partner
GARR
INFN-TO
Collegio Carlo Alberto
Città Salute e Scienza
Fondazione ISI
ISMB
IIT Genova
Human Technopole
Universities and Departments
DIPINF
DIPMAT
DIPGIU
UNITO (coord)
DIPES
DIPFIL
MEDHIUM
C3S
ICxT
DET
DAUIN
POLITO
HPC @PoliTO
SmartData @PoliTO
Data Centers
C3S
PdF
SmartData
HPC
Hardware
green cooling
specialised islands
Federated cloud islands
GARR cloud "Piemonte zone"
specialised islands (non federated)
green cooling

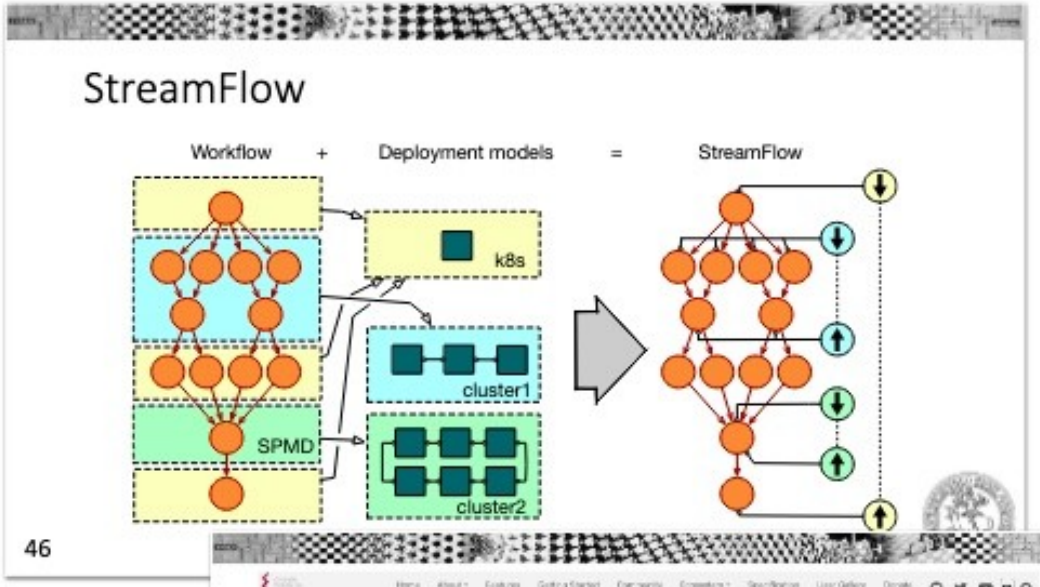
The Vision

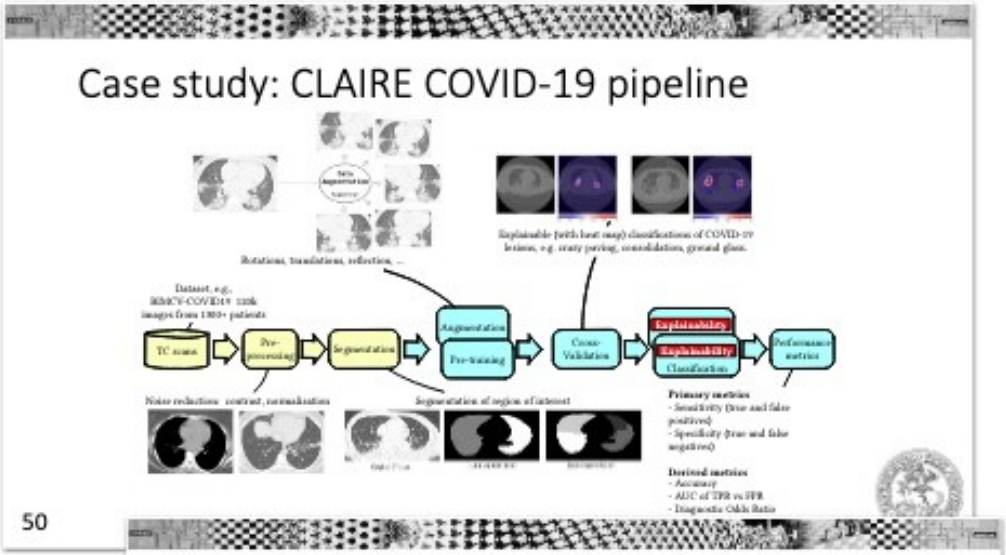


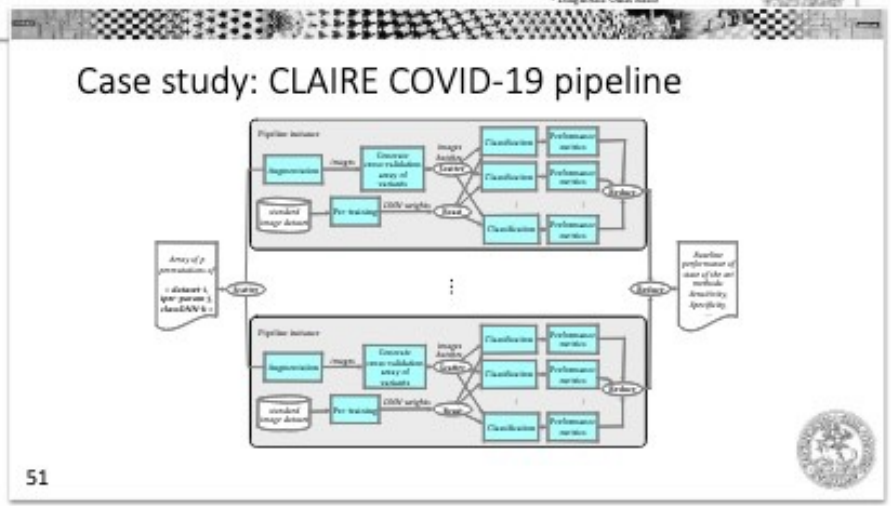
The Hardware

- +3500 cores, +100 GPUs
- Mujltiple architectures (Intel Xeon, ARM v9, SiFive RISC-V boards)
- Multiple accelerators (NVIDIA V100, T4, A100, A40)
- Mujltiple storage technologies (HDD, SSD, NVMe, Intel Optane)

The Software - StreamFlow





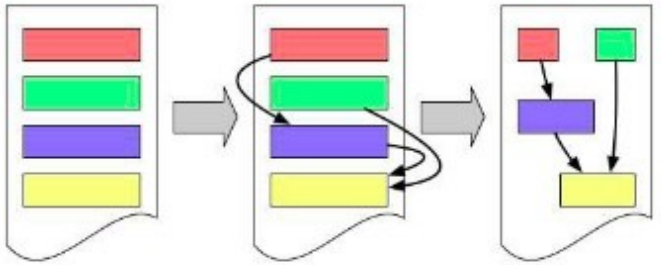


https://streamflow.di.unito.it/

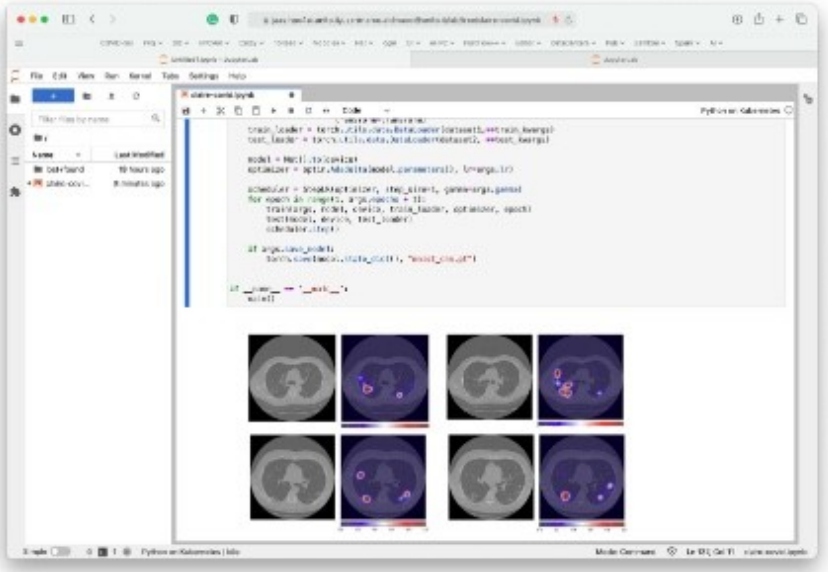
The Software - Jupyter Workflow

Hybrid literate workflows

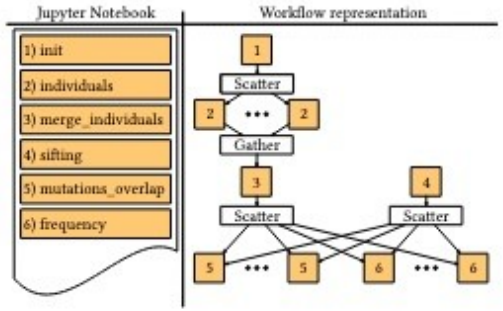
Use Notebook metadata as a **coordination language** to express distributed workflows



63

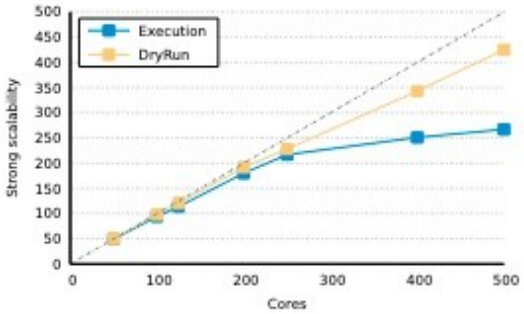


Case study: 1000-genome Notebook



102

Case study: 1000-genome Notebook

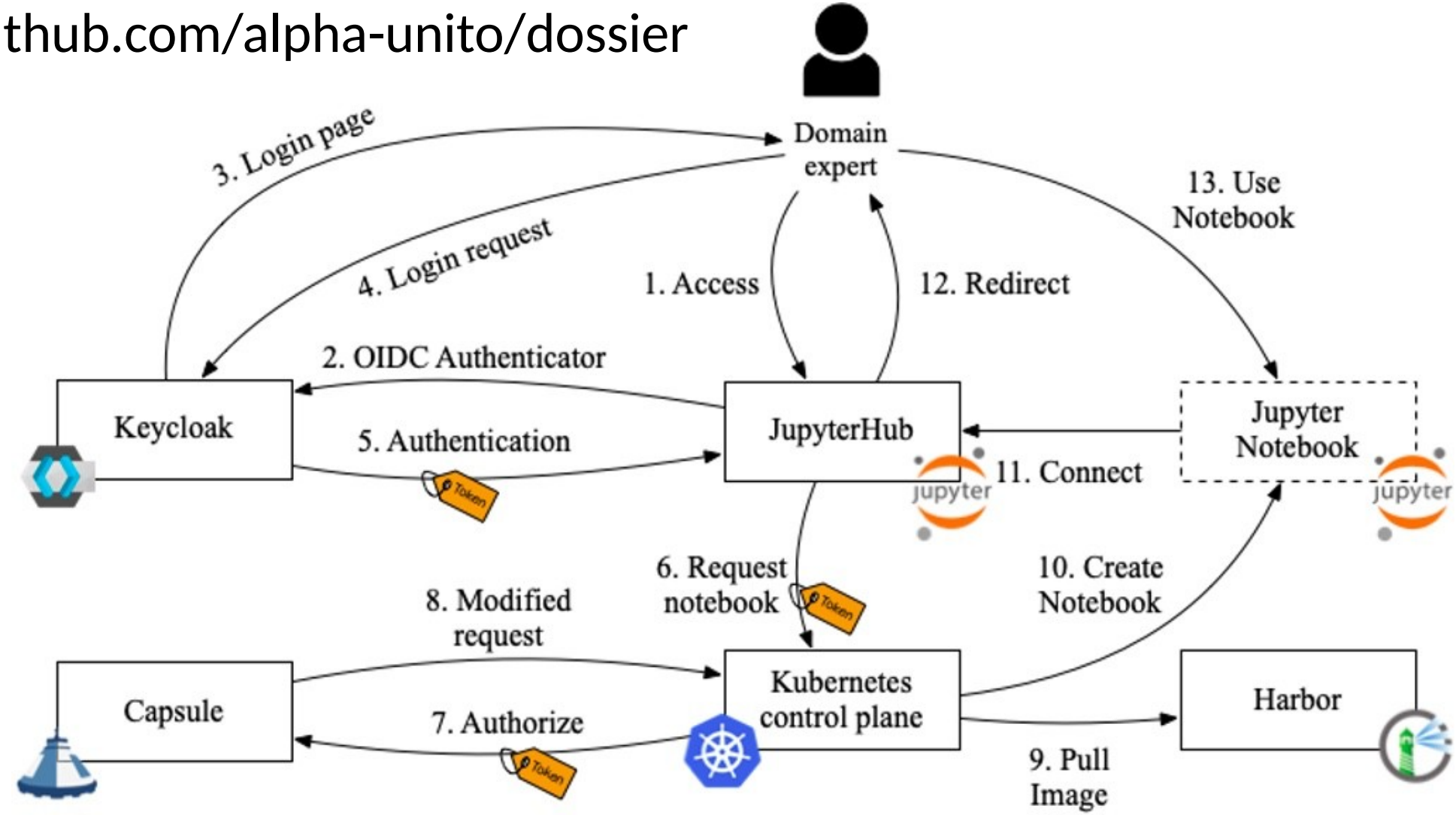


104

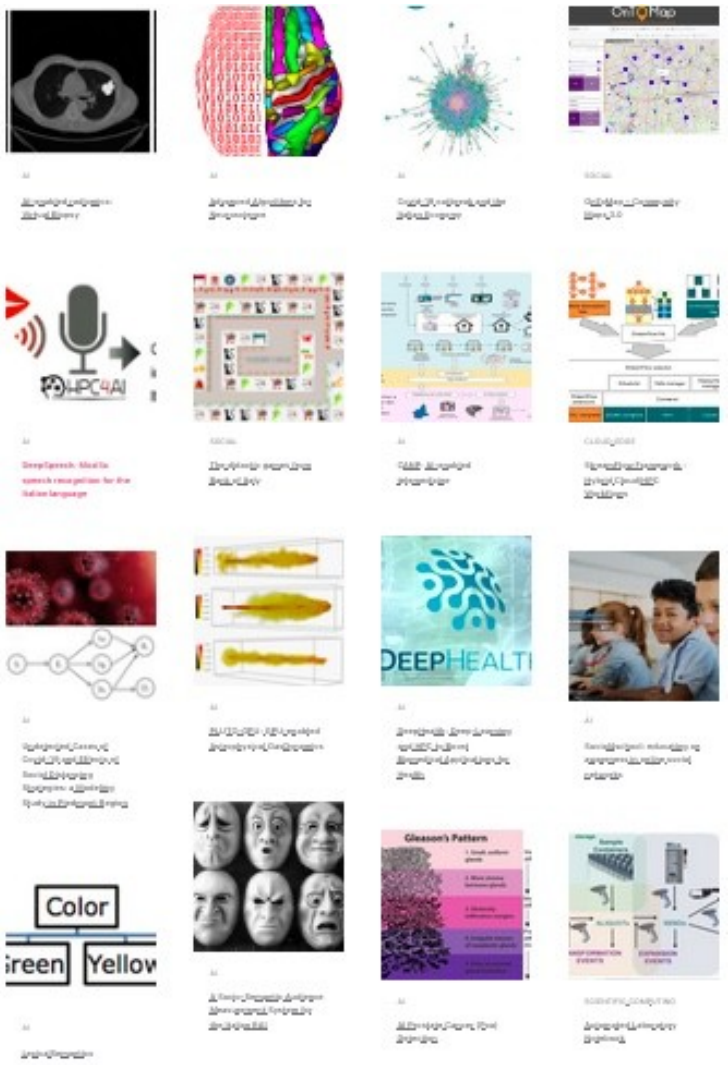
https://jupyter-workflow.di.unito.it

The Software - Dossier

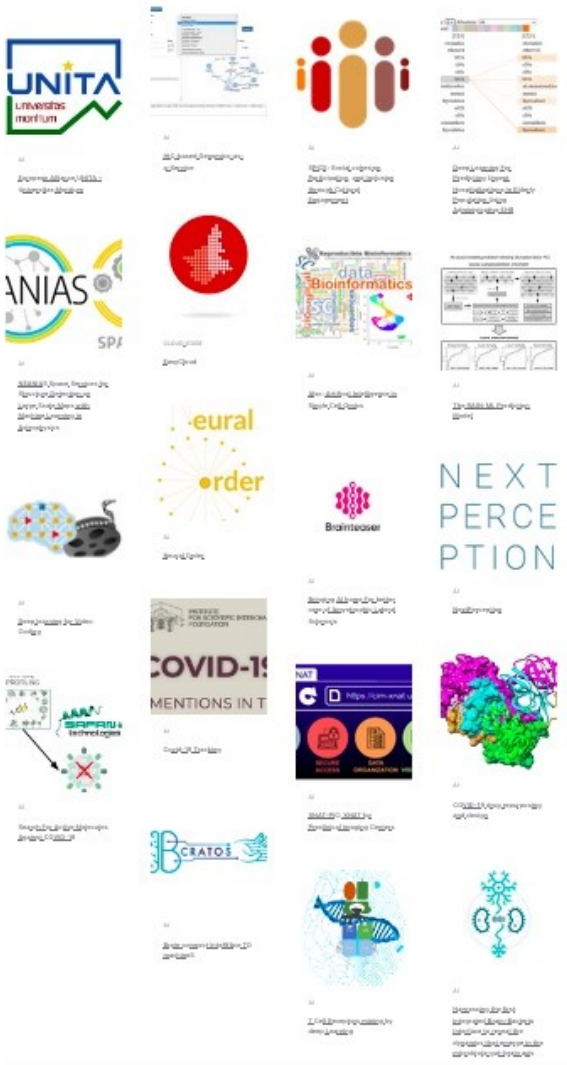
https://github.com/alpha-unito/dossier



The Results



- +35 hosted research projects (mostly AI)
- +100M of research projects' total cost
- +10 contributions to the Cloud stack software



The Results

Articles

THE LANCET

Machine learning-based prediction of adverse events following an acute coronary syndrome (PRAISE): a modelling study of pooled datasets

Fabrizio D'Ascenzo, Ovidio De Filippo, Gugl
Christoph Liebetrau, Sergio Manzano-Fern
Alberta Dominguez-Rodriguez, Marco Aldi
Gaetano Maria De Ferrari, on behalf of the I

Summary
Background The accuracy of curren
(ACS) remains insufficient for indi
risk stratification model to predict

Medical Image Analysis 36 (2017) 41–51

Contents lists available at ScienceDirect

Medical Image Analysis

journal homepage: www.elsevier.com/locate/media

Deep learning for automated skeletal bone age assessment in X-ray images☆

C. Spampinato[a,*], S. Palazzo[a], D. Giordano[a], M. Aldinucci[b], R. Leonardi[c]

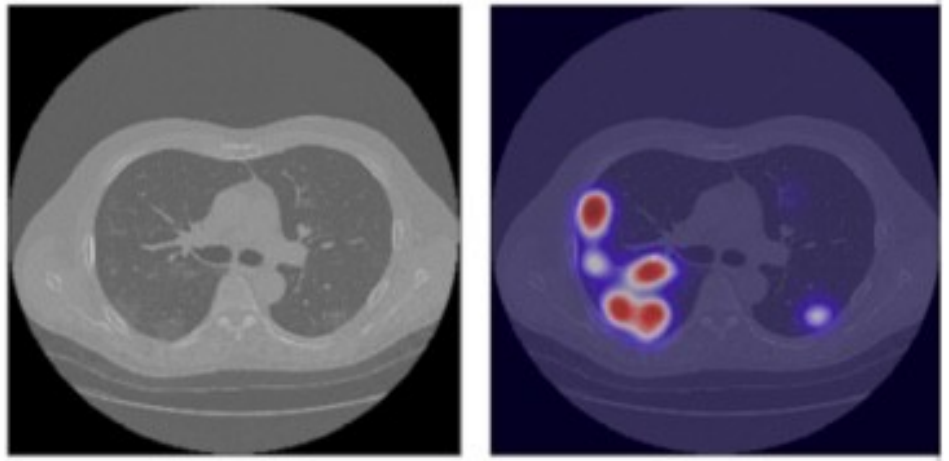
[a] Pattern Recognition and Computer Vision (PeRCeiVe) Lab, Department of Electrical, Electronics and Computer Engineering, University of Catania, Viale Andrea Doria, 6 - 95125 - Catania, Italy
[b] Computer Science Department, University of Torino, Corso Svizzera, 185 - 10149 - Torino, Italy
[c] Department of Orthodontics, University of Catania, Via Santa Sofia, 78 - 95125 - Catania, Italy

High-performance computing and AI team up for COVID-19 diagnostic imaging

by Marco Aldinucci

articles

January 12, 20



The Confederation of Laboratories for Artificial Intelligence Research in Europe (CLAIRE taskforce on AI & COVID-19 supported the creation of a research group focused on AI

The End

HPC4AI website - https://hpc4ai.unito.it/

StreamFlow:

- Website - https://streamflow.di.unito.it/
- Code - https://github.com/alpha-unito/streamflow
- Article - https://doi.org/10.1109/TETC.2020.3019202

Jupyter Workflow:

- Website - https://jupyter-workflow.di.unito.it/
- Code - https://github.com/alpha-unito/jupyter-workflow
- Article - https://doi.org/10.1016/j.future.2021.10.007

Dossier - https://github.com/alpha-unito/dossier