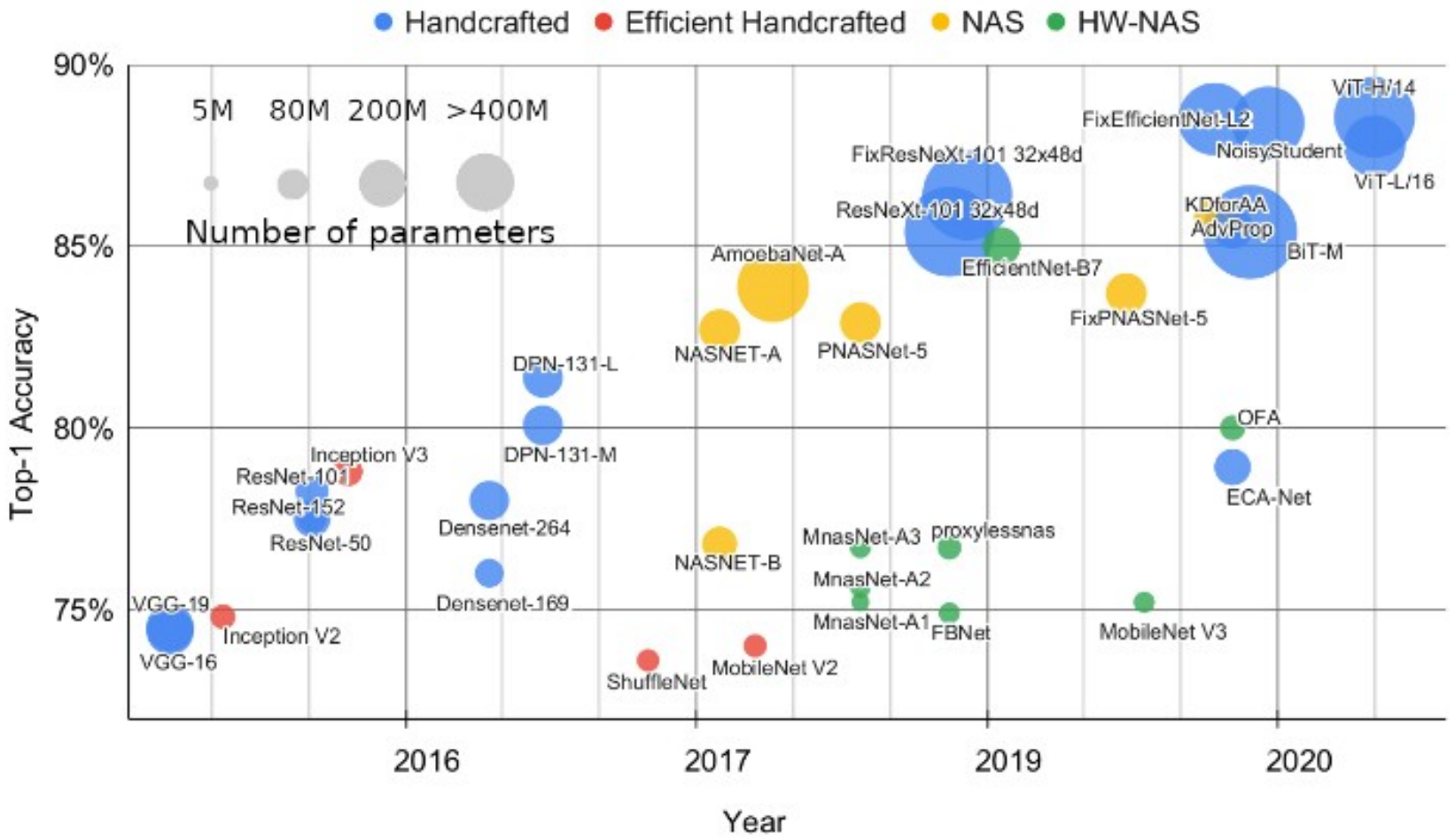
Efficiency by Pruning in Deep Learning

a LISTEN-Lab seminar
20/03/2025

Enzo Tartaglione
Maître de Conférences, équipe MM, dept.IDS, LTCI
Hi!PARIS chair holder

enzo.tartaglione@telecom-paris.fr

Less recent trends in ANNs



More recent trends in ANNs

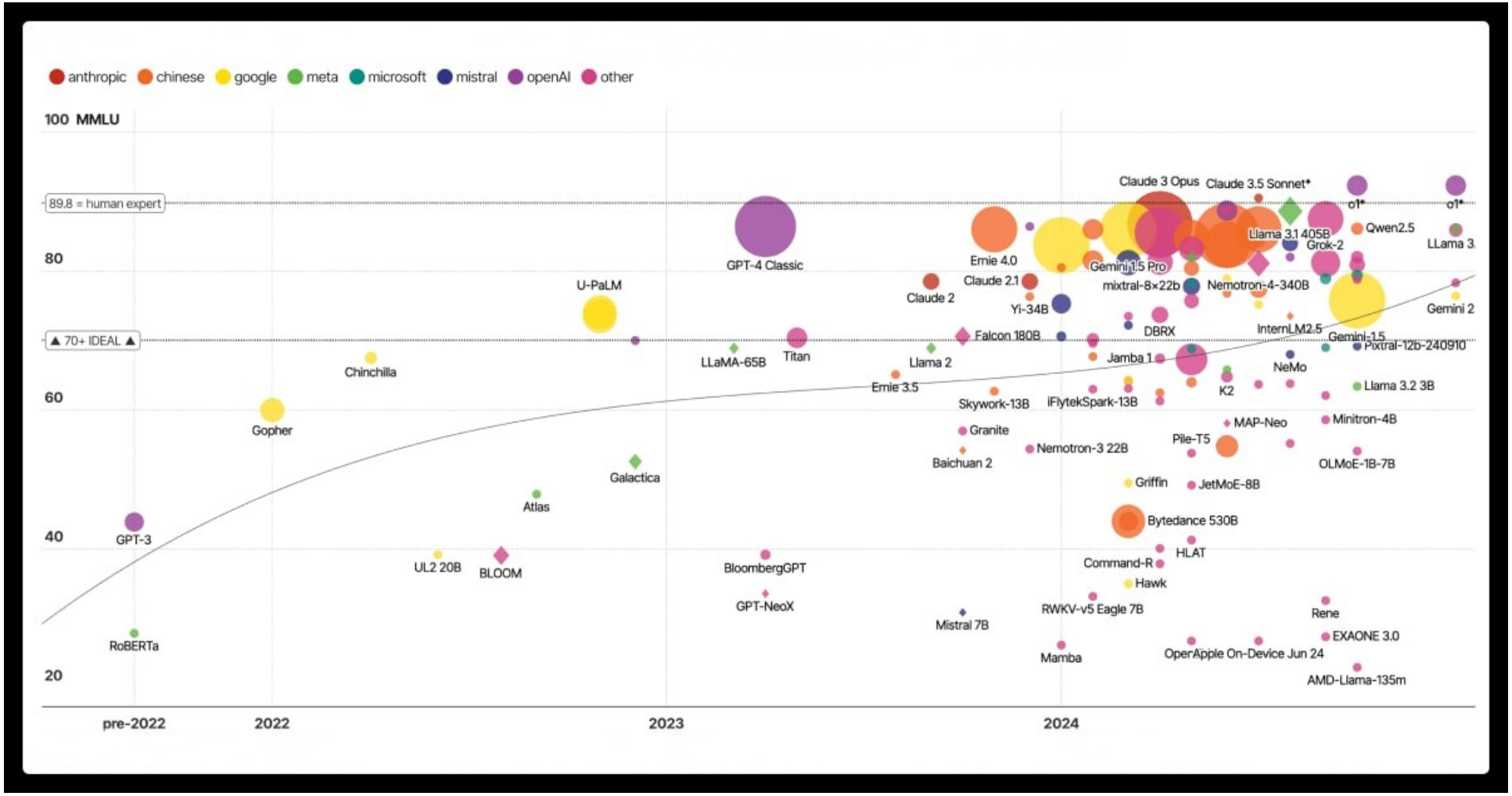


Image from https://labelyourdata.com/articles/llm-model-size

Frugality in AI

- **Perform training with little data**
  - Better optimizers
  - New loss functions
  - Ease the features extraction with priors
  - Transfer learning
  - …

Frugality in AI

- **Perform training with little data**
  - Better optimizers
  - New loss functions
  - Ease the features extraction with priors
  - Transfer learning
  - …
- **Use the trained AI models with little computational resources**
  - Pruning & Quantization
  - Knowledge distillation
  - Neural Architecture Search
  - …

Frugality in AI

- **Perform training with little data**
  - Better optimizers
  - New loss functions
  - Ease the features extraction with priors
  - Transfer learning
  - …
- **Use the trained AI models with little computational resources**
  - Pruning & Quantization
  - Knowledge distillation
  - Neural Architecture Search
  - …
- **Train with little computational resources**
  - Ensure fast convergence for models
  - Reduce the training cost
  - …

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

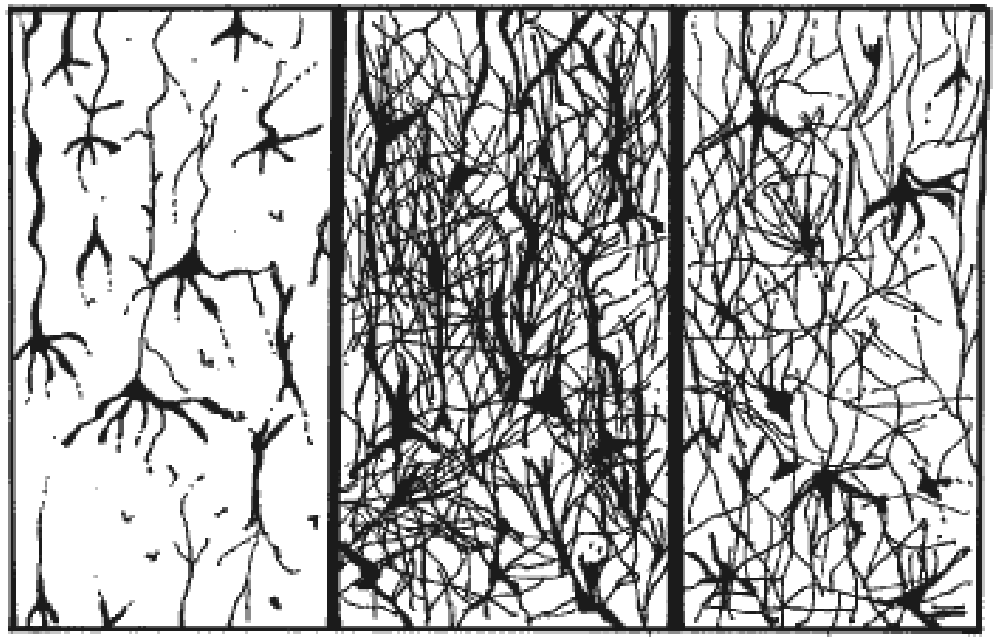
- With pruning we refer to the process of removing parameters (synapses) or entire units from the deep learning model.
- Pruning relates with **sparsification**: the weight matrix (representing a layer) becomes, indeed, sparse!
- The removed parameters (if they are removed in an “unstructured” way) still 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!

Pruning in deep learning: why?

- Reducing the storage memory (for a compressed model).
- Reducing the memory footprint.
- Reducing the FLOPs at inference time.
- Reducing energy consumption?
- Enhancing generalization?

Countless approaches to achieve sparse models…

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

Skeletonization [Mozer and Smolensky, 1988]

The idea of the work is very simple…

1) Iteratively train the network to a certain performance criterion

Skeletonization [Mozer and Smolensky, 1988]

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, 1988]

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, 1988]

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!

Optimal brain damage [Le Cun et al., 1989]

- 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., 1989]

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

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

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

Optimal brain damage [Le Cun et al., 1989]

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

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

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

- 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., 1989]

- 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., 1989]

- 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).

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

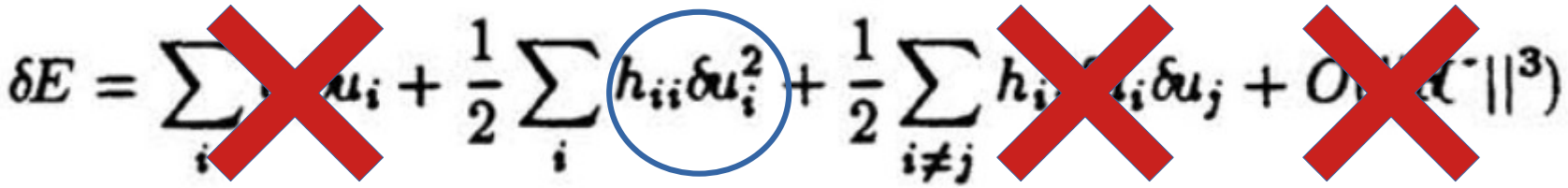
Optimal brain damage [Le Cun et al., 1989]

- 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).

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

Optimal brain damage [Le Cun et al., 1989]

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

Neuron saliency [Le Cun et al., 1989]

- 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

Neuron saliency [Le Cun et al., 1989]

- 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 convergence)

Neuron saliency [Le Cun et al., 1989]

- 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 convergence)
3. Compute the second derivatives hu for each parameter
4. Compute the saliencies for each parameter: Sk = huu~/2

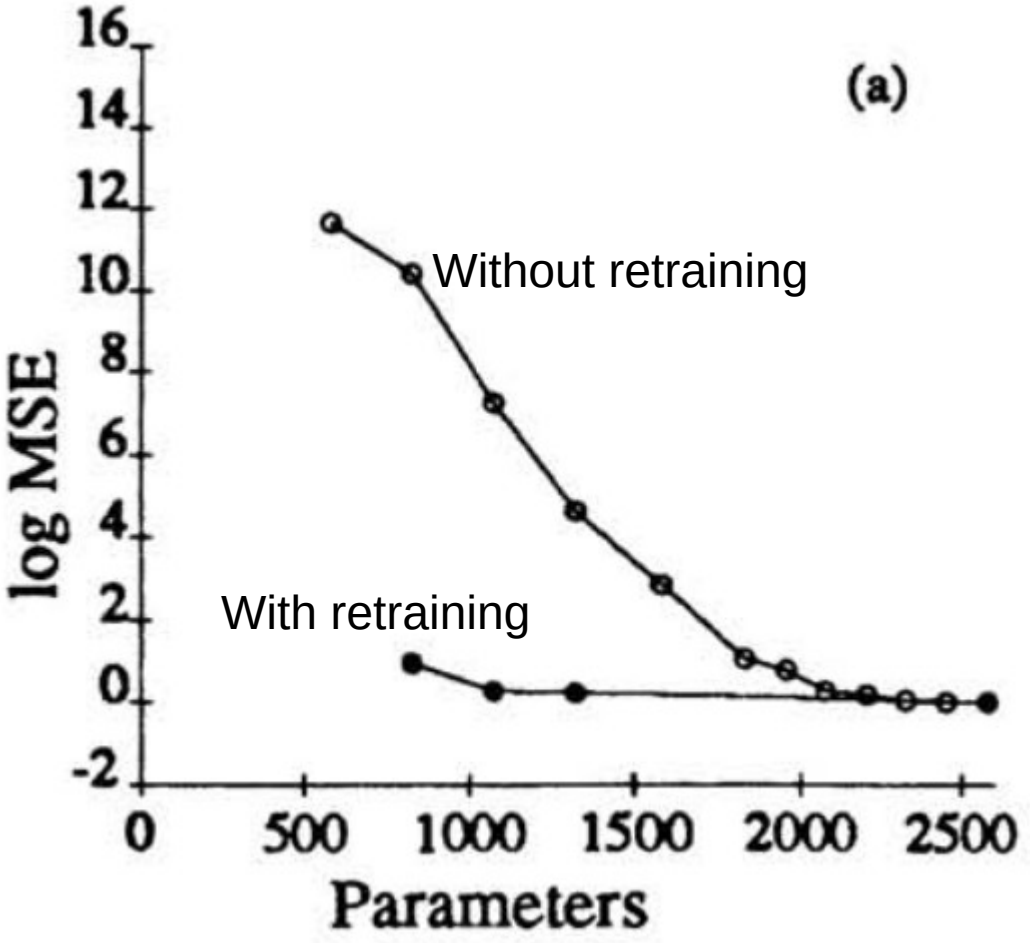
Neuron saliency [Le Cun et al., 1989]

- 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 convergence)
3. Compute the second derivatives hu for each parameter
4. Compute the saliencies for each parameter: Sk = huu~/2
5. Sort the parameters by saliency and delete some low-saliency parameters (thresholding)
6. Iterate to step 2

Results [Le Cun et al., 1989]



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!

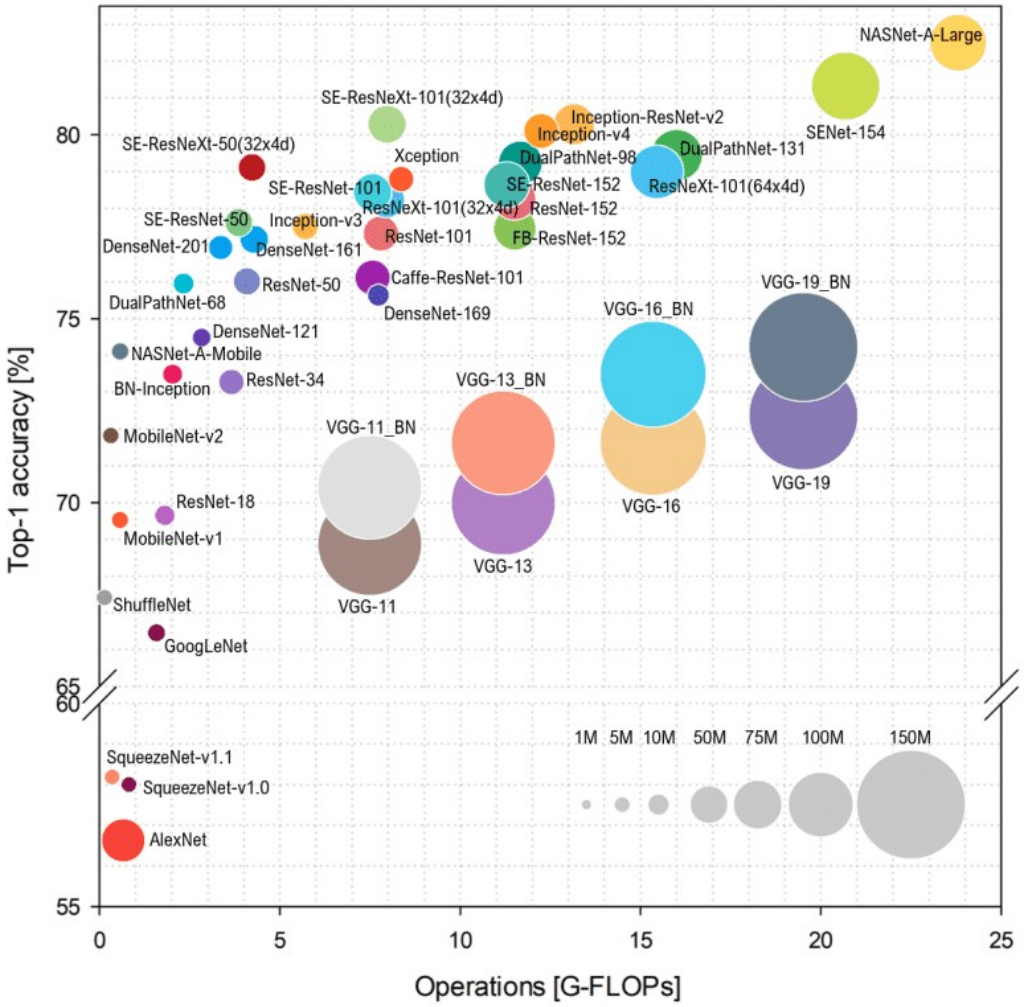


Image from https://theaisummer.com/cnn-architectures/

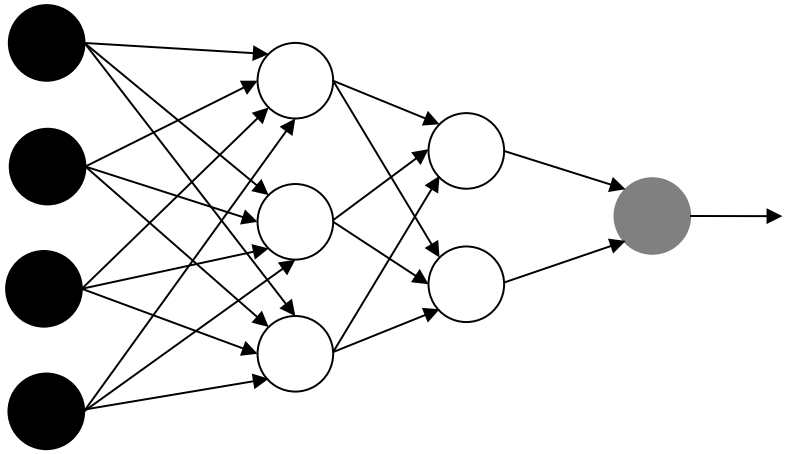
Learning both weights and connections [Han et al., 2015]

- 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., 2015]

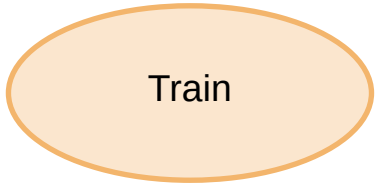
- Parameters are randomy initialized

Train

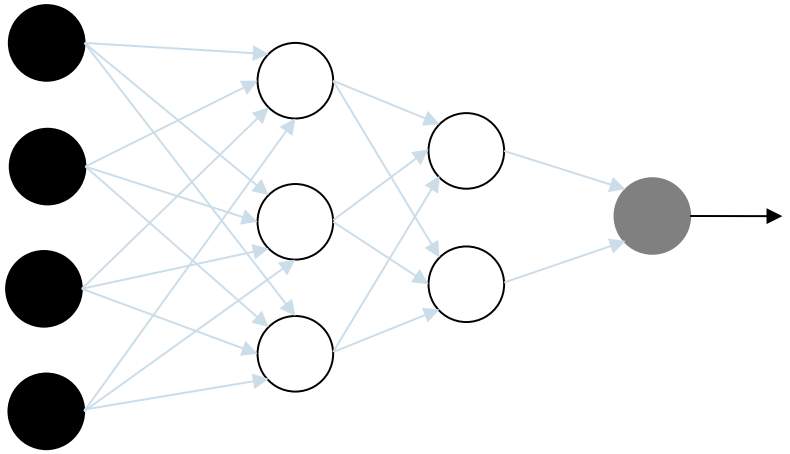


Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*.

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

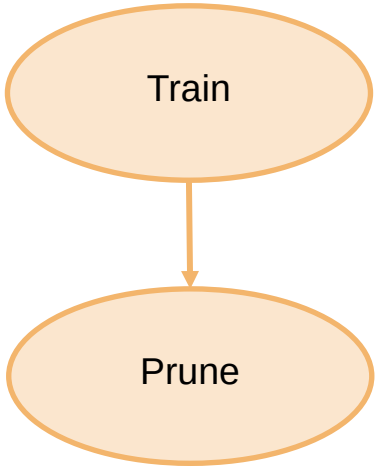


- Parameters are randomy initialized
- Parameters are updated then trained with standard gradient descent until performance is achieved (training stage)

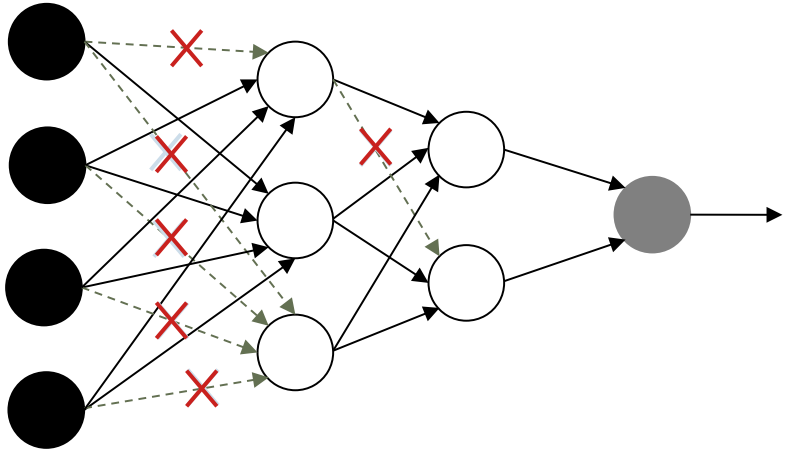


Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*.

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

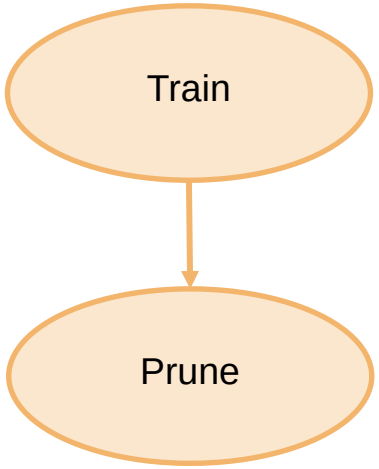


- Parameters are randomy initialized
- Parameters are updated then trained with standard gradient descent until performance is achieved (training stage)
- Parameters below threshold T are removed, pruning connections (parameter sparsification)

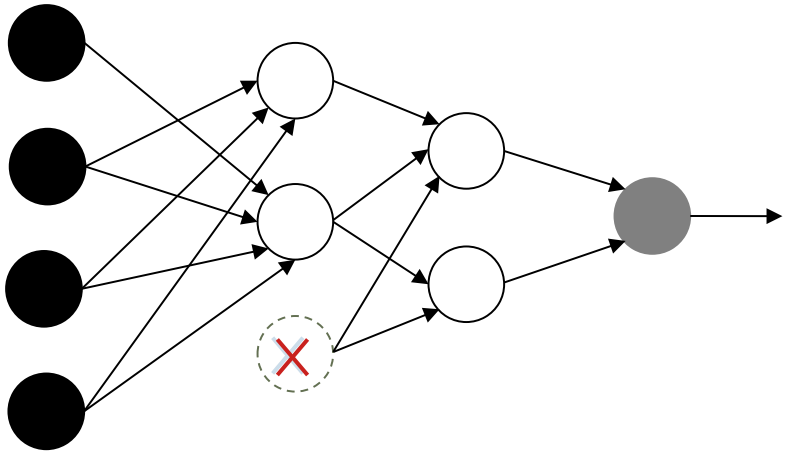


Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*.

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

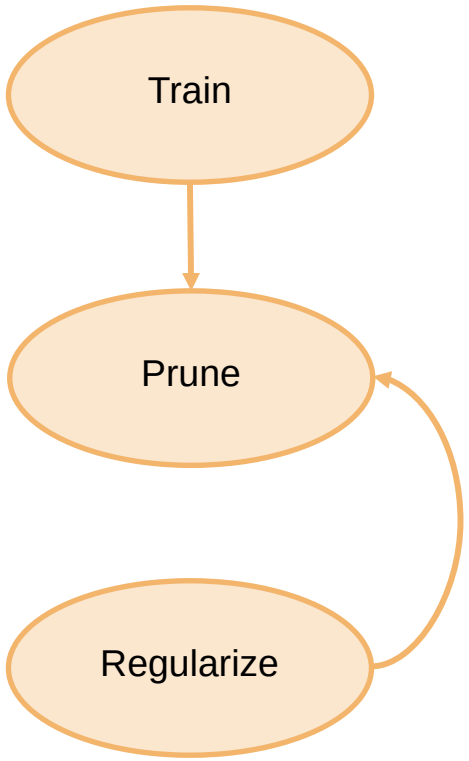


- Parameters are randomy initialized
- Parameters are updated then trained with standard gradient descent until performance is achieved (training stage)
- Parameters below threshold T are removed, pruning connections (parameter sparsification)
- Neurons without input arcs input are pruned from the network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

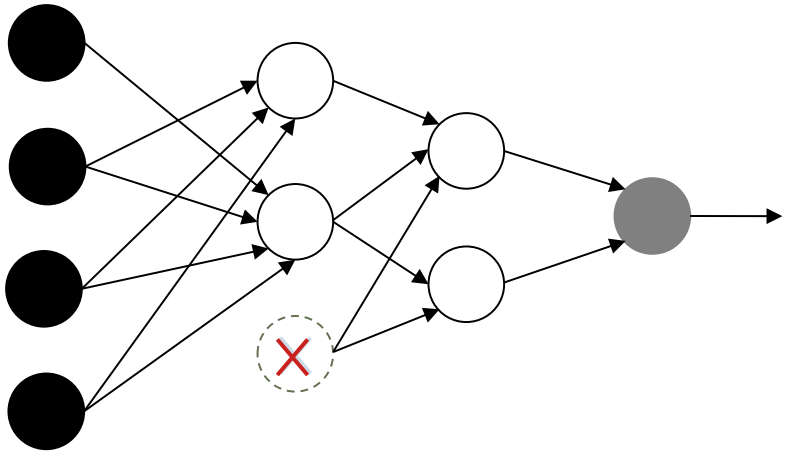


Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*.

Learning both weights and connections [Han et al., 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

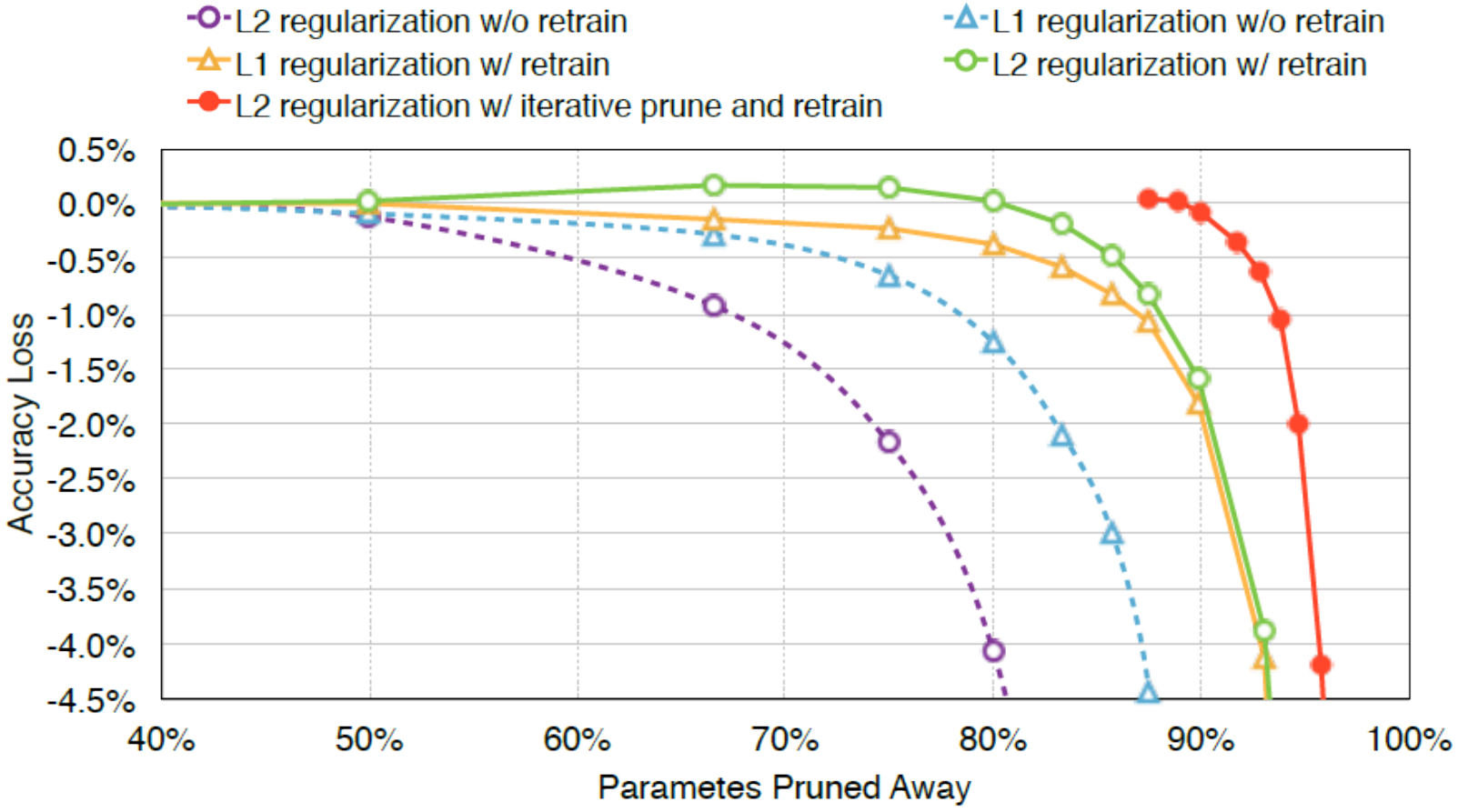
- Add extra constraints to the objective function minimized.

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

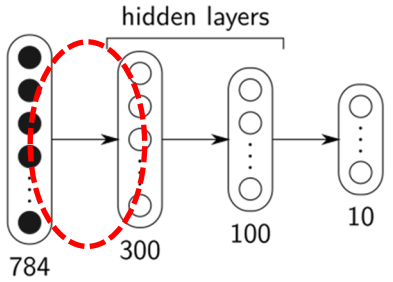
- Some very common regularization are dropout, L2 (weight-decay)…

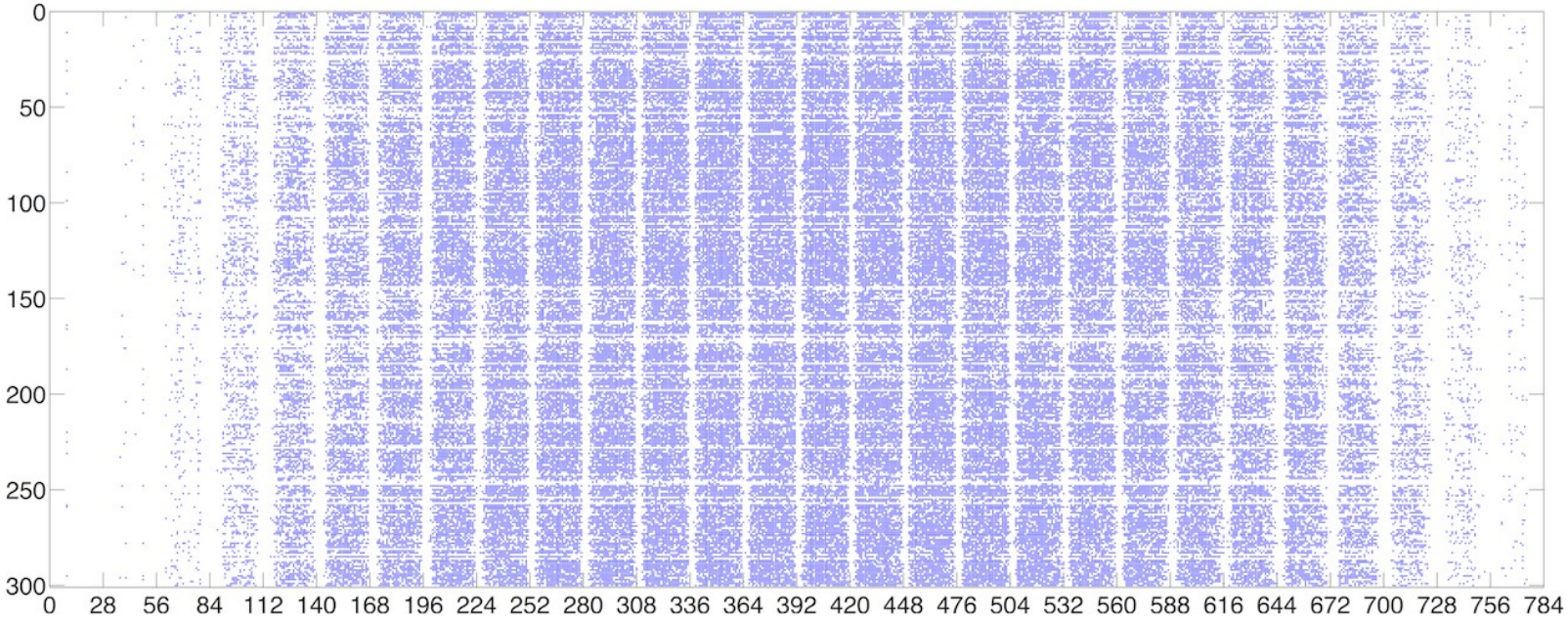
Learning both weights and connections [Han et al., 2015]

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



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

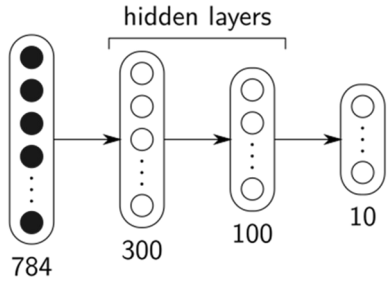
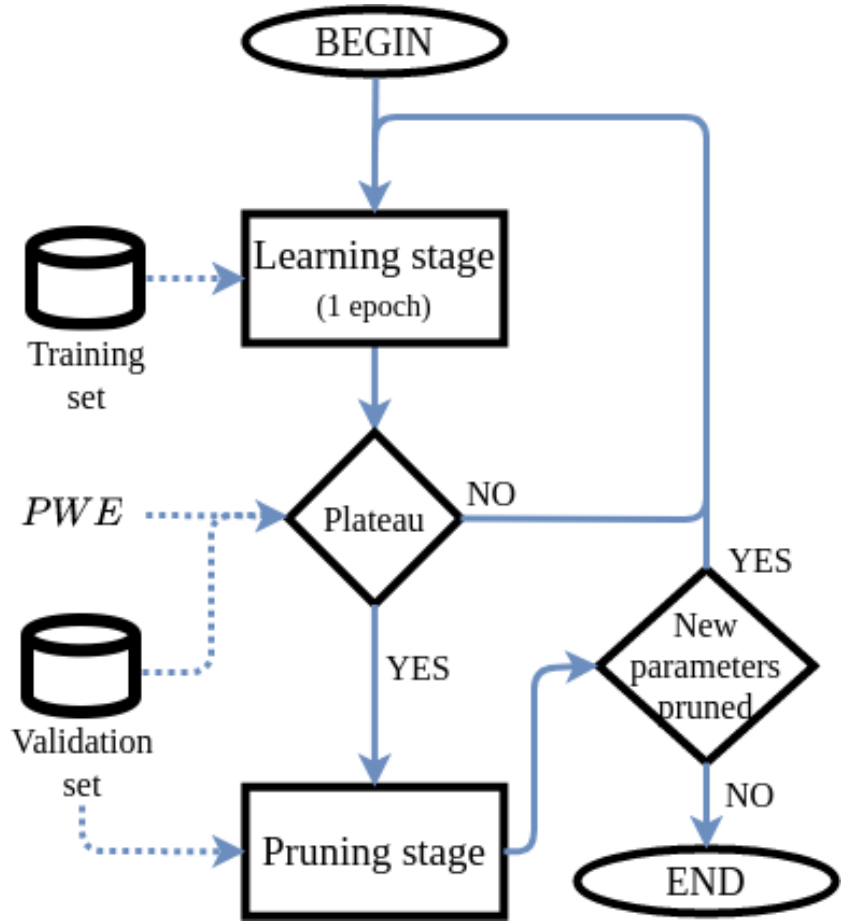


Table 1: Network pruning can save 9× to 13× parameters with no drop in predictive performance.

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.

Pruning 101


BEGIN
Learning stage
(1 epoch)
Training
set
PWE
Plateau
NO
YES
New
parameters
pruned
YES
Validation
set
Pruning stage
NO
END

The elephant in the room

- Objective: reach the highest sparsity with no task-related performance degradation.
- The most effective approaches are also the ones **more computationally intensive**
  - multiple trainings are required for training one sparse model.
  - One-shot (or few-shot) pruning approaches are in general worse.

- New challenge: achieve sparsity with less computation at training time.

Pruning at initialization??
The lottery ticket hypothesis

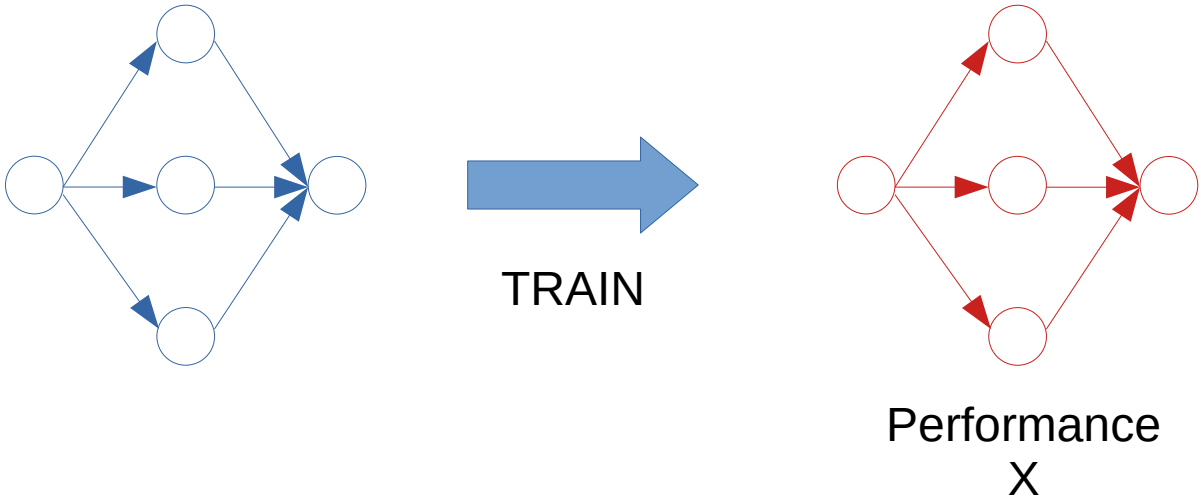
The lottery ticket hypothesis

"A randomly-initialized, dense neural network contains a sub-network that is initialized such that, when trained in isolation, it can match the test accuracy of the original network after training for at most the same number of iterations." [Frankle and Carbin, 2019]

- The sub-network exists already at initialization: if it can be found, a lot of computation can be saved.
  - Computation for training the model (less parameters to train)
  - Computation for pruning (no iterative pruning anymore, just pruning at initialization, "zero-shot" pruning)

How to identify the lottery winners?

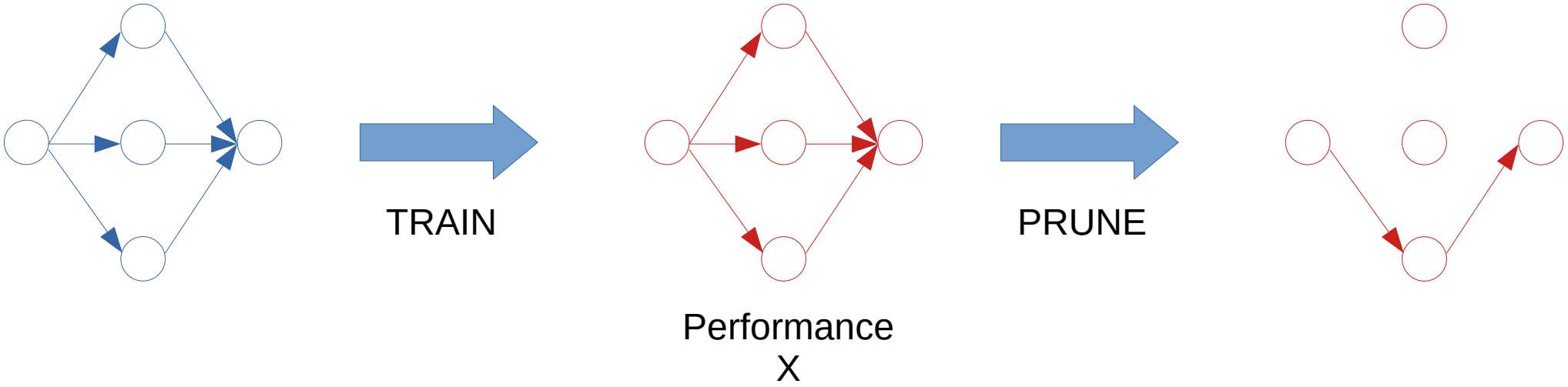
How to identify the lottery winners?



BEFORE TRANING

AFTER TRAINING

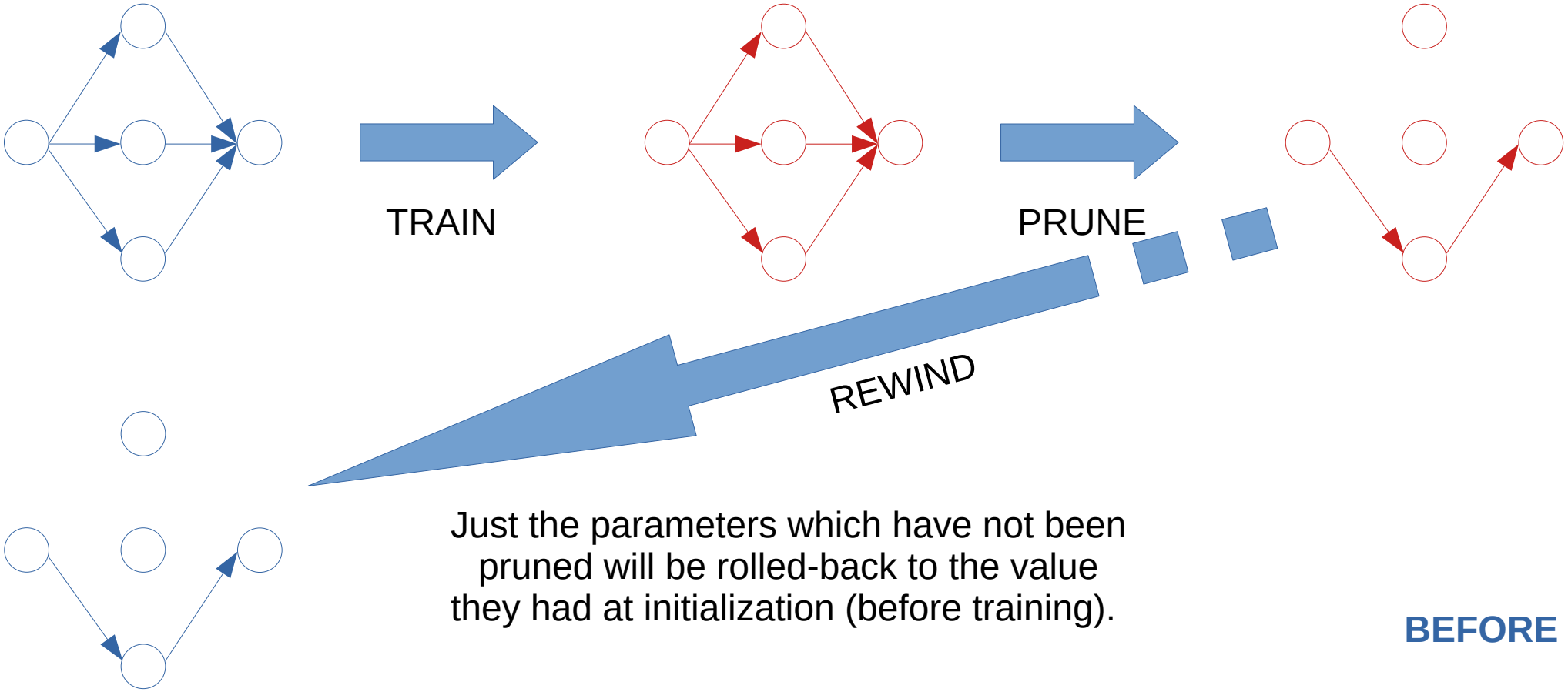
How to identify the lottery winners?



BEFORE TRANING

AFTER TRAINING

How to identify the lottery winners?

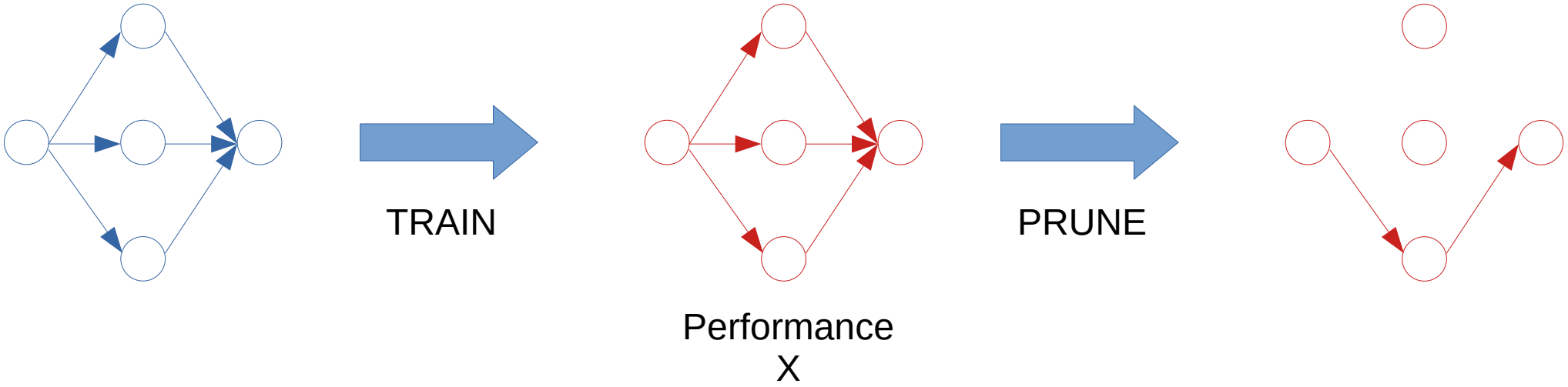


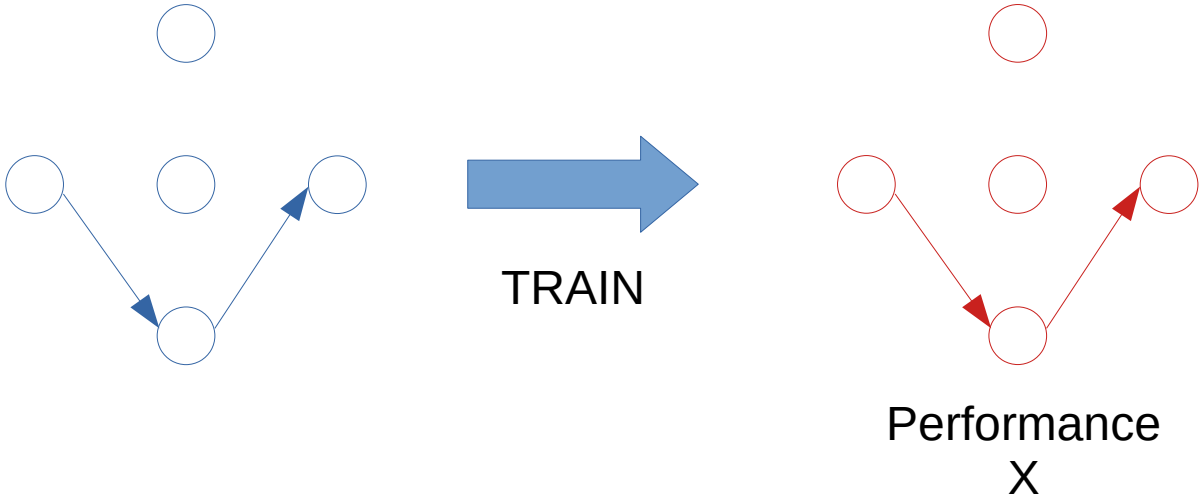
Just the parameters which have not been pruned will be rolled-back to the value they had at initialization (before training).

BEFORE TRANING

AFTER TRAINING

How to identify the lottery winners?



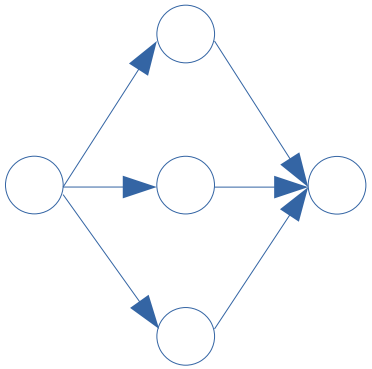


The **same** performance is obtained training the model with **less** parameters.

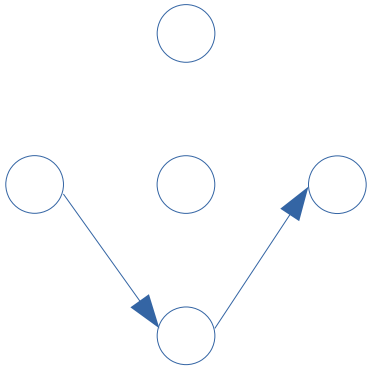
BEFORE TRANING

AFTER TRAINING

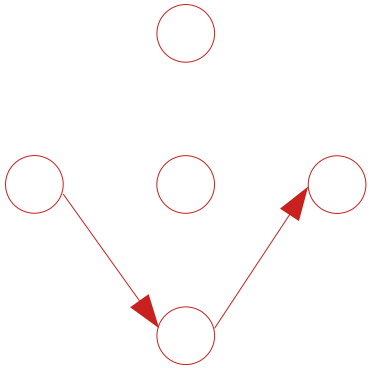
What we would like!



REMOVE UN-NECESSARY PARAMETERS



TRAIN



Performance X

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

BEFORE TRANING

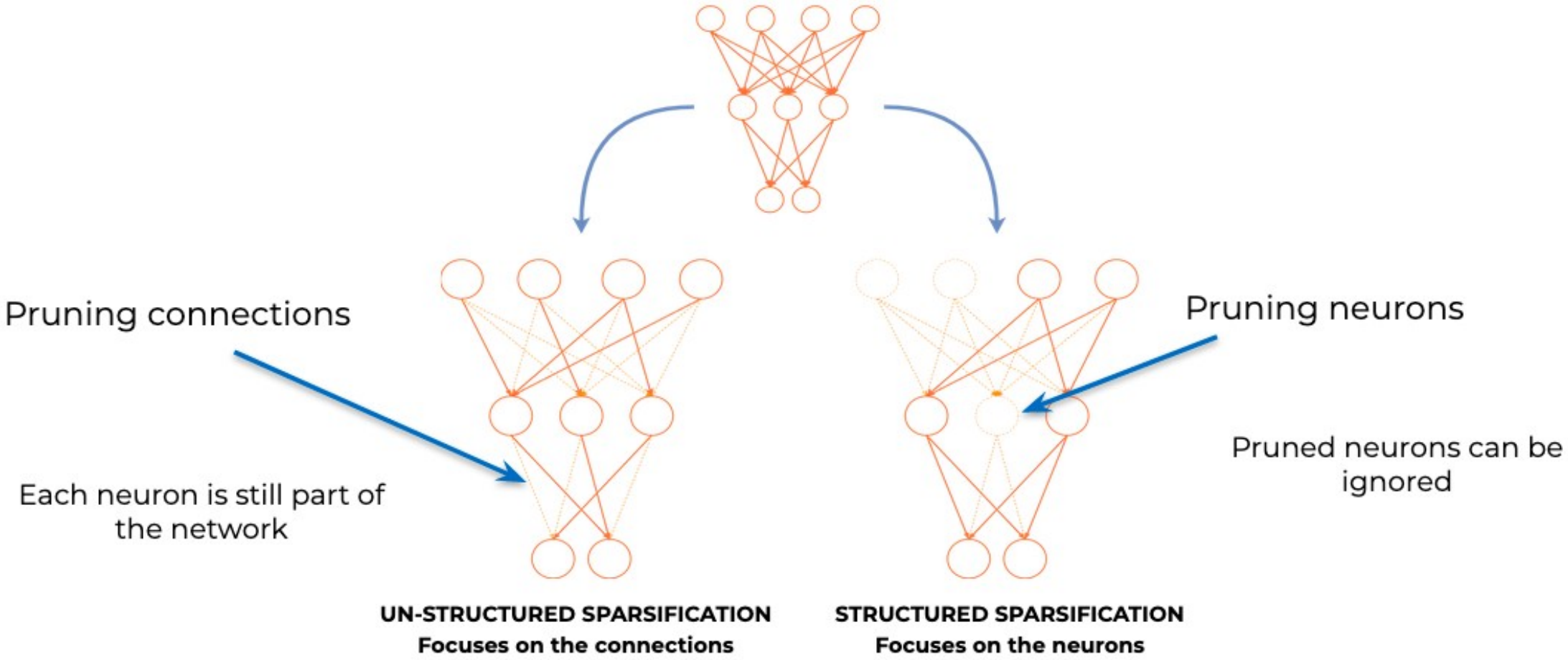
AFTER TRAINING

The lottery ticket hypothesis: finding sparse, trainable neural networks

[Frankle and Carbin, ICLR 2019]

- 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…

Structured vs Unstructured sparsity



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.

Can we prune entire layers?

Layer collapse

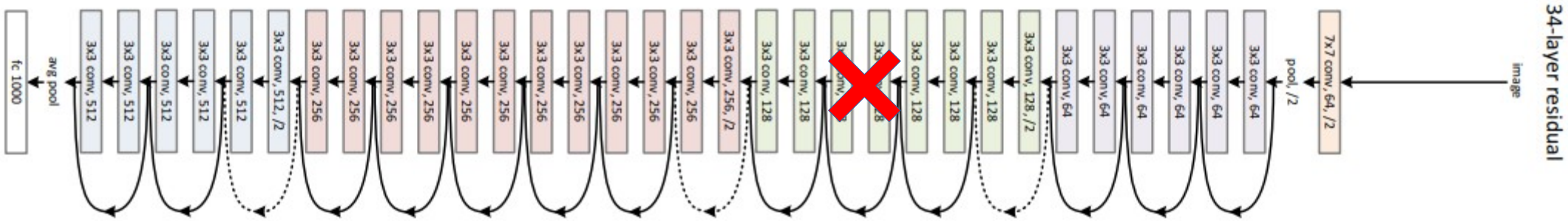
In deep learning, **layer collapse** refers to a phenomenon where a specific layer in a neural network fails to effectively differentiate the input features, causing all outputs from the layer to converge to similar or identical values regardless of input.

Layer collapse

In deep learning, **layer collapse** refers to a phenomenon where a specific layer in a neural network fails to effectively differentiate the input features, causing all outputs from the layer to converge to similar or identical values regardless of input.

Although this effect is traditionally seen as a downside of poor initialization or hyper-parameter fine-tuning (ie. the model does not learn), what if *the model learns, while some layers are collapsed?*

- If that happens, we can completely skip computation of such layer!



Layer collapse- without skip connections?

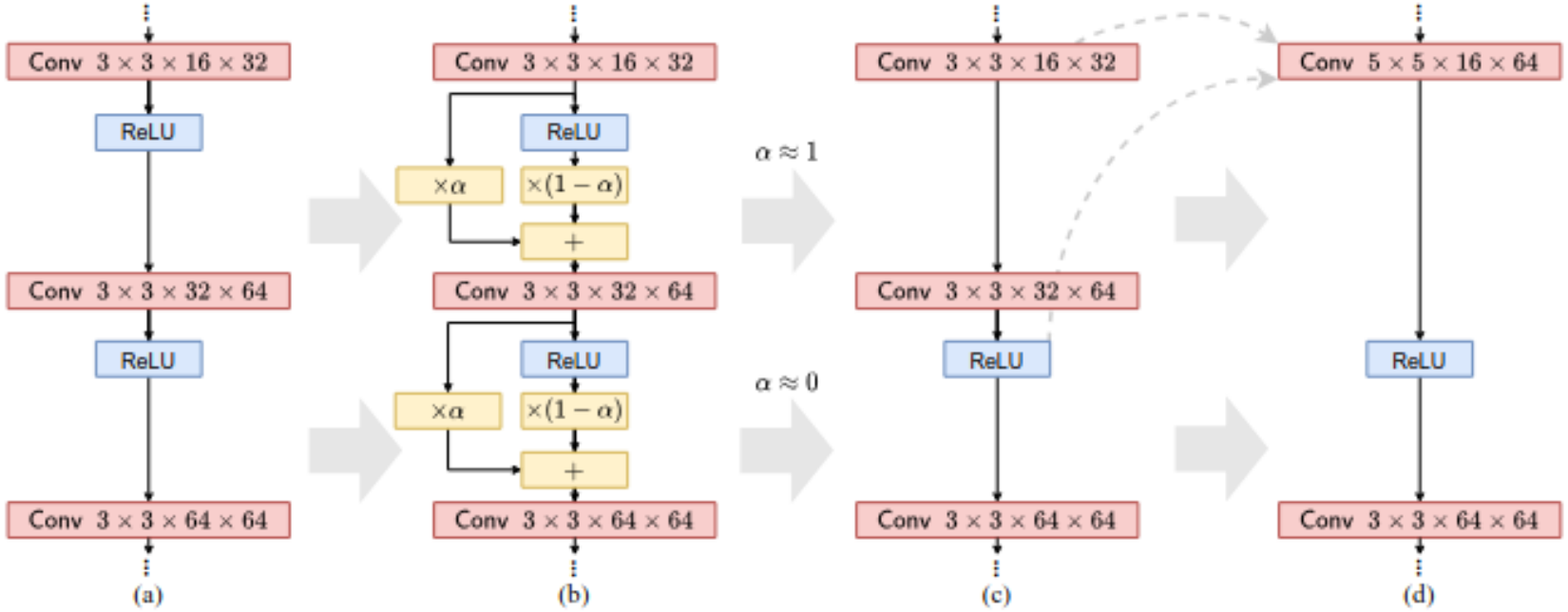
Layer collapse- without skip connections?

Is it really possible?

Layer collapse- without skip connections? [Dror et al. 2021]

Is it really possible?

We can attack the problem differently. One way to reduce the model's depth is to *remove non-linearities!* Layer fold is one approach parametrizing the negative slope of a PReLU.



Is it straightforward? [Pilo et al. 2025]

Case of study: fully-connected layers

$$\boldsymbol{y} = (\boldsymbol{x}W_1^T + b_1)W_2^T + b_2,$$

Pilo, G., Hezbri, N., e Ferreira, A. P., Quétu, V., & Tartaglione, E. (2025). LayerFold: A Python library to reduce the depth of neural networks. SoftwareX, 29, 102030.

Is it straightforward? [Pilo et al. 2025]

Case of study: fully-connected layers

$$\boldsymbol{y} = (\boldsymbol{x}W_1^T + b_1)W_2^T + b_2,$$

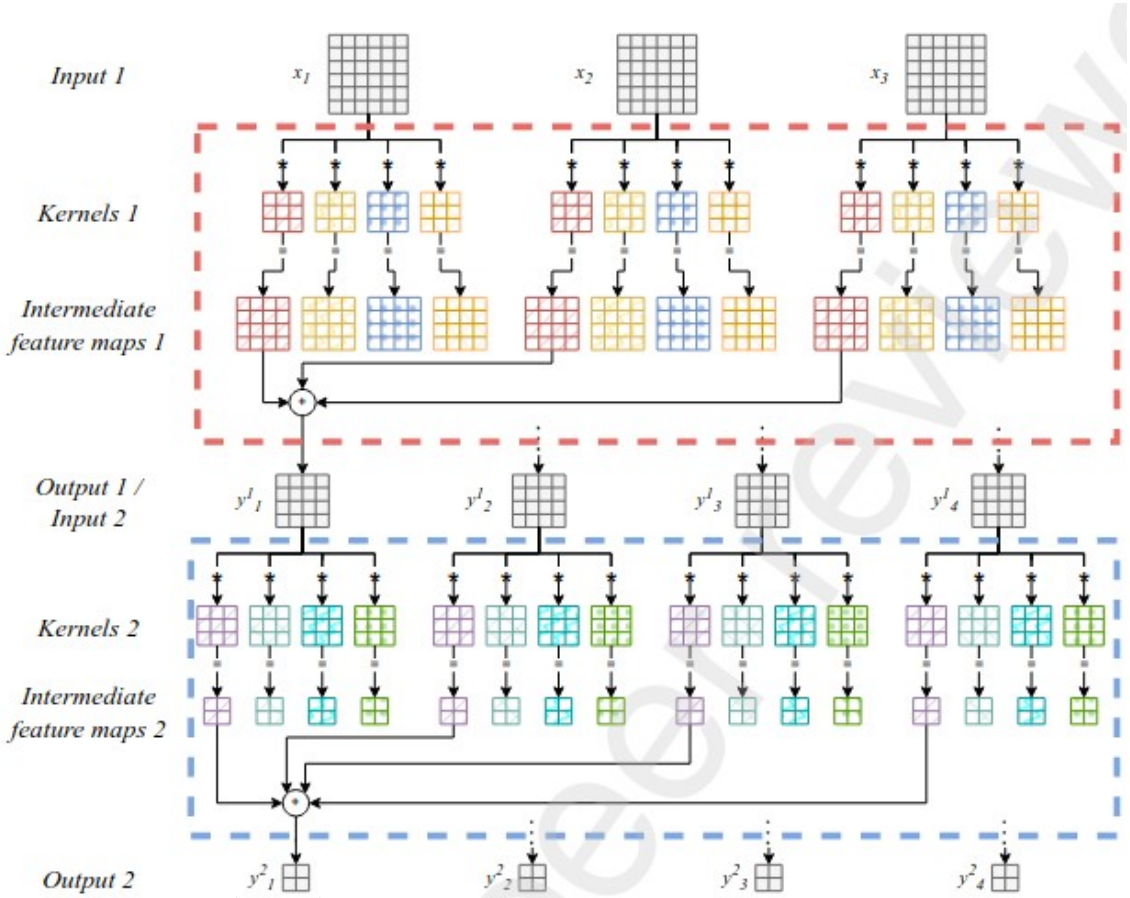
$$W_{eq} = W_2 W_1 \qquad b_{eq} = b_1 W_2^T + b_2.$$

The solution is straightforward.

Pilo, G., Hezbri, N., e Ferreira, A. P., Quétu, V., & Tartaglione, E. (2025). LayerFold: A Python library to reduce the depth of neural networks. SoftwareX, 29, 102030.

Is it straightforward? [Pilo et al. 2025]

Case of study: convolutional layers, stride=1, no padding



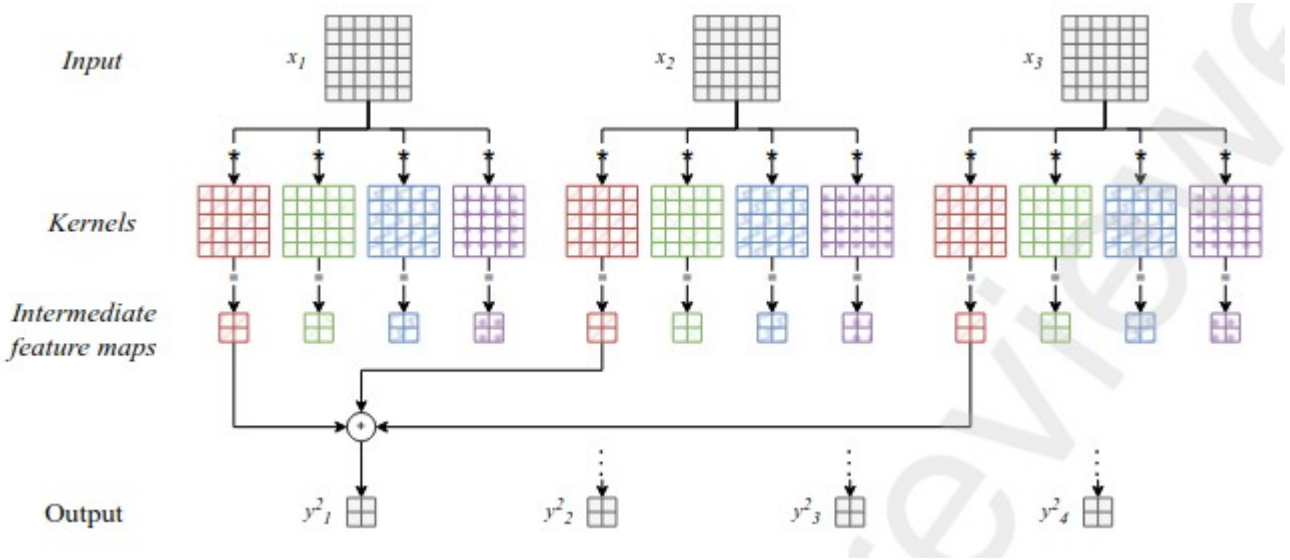
Pilo, G., Hezbri, N., e Ferreira, A. P., Quétu, V., & Tartaglione, E. (2025). LayerFold: A Python library to reduce the depth of neural networks. SoftwareX, 29, 102030.

Is it straightforward? [Pilo et al. 2025]

Case of study: convolutional layers, stride=1, no padding

$$W_{eq} = W_1 * W_2$$

$$b_{eq,j} = b_{2,j} + \sum_{i=1}^{c_1} b_{1,i} \sum_{l=1}^{k_2} \sum_{m=1}^{k_2} w_{2,i,j}.$$



The solution is straightforward.

Pilo, G., Hezbri, N., e Ferreira, A. P., Quétu, V., & Tartaglione, E. (2025). LayerFold: A Python library to reduce the depth of neural networks. SoftwareX, 29, 102030.

Is it straightforward? [Pilo et al. 2025]

Case of study: convolutional layers, stride=1, padding=1

Pilo, G., Hezbri, N., e Ferreira, A. P., Quétu, V., & Tartaglione, E. (2025). LayerFold: A Python library to reduce the depth of neural networks. SoftwareX, 29, 102030.

Is it straightforward? [Pilo et al. 2025]

Case of study: convolutional layers, stride=1, padding=1

To the moment, no closed-form solutions!

→ (psst!) You can bypass it with knowledge distillation

Is this case that much frequent?

Pilo, G., Hezbri, N., e Ferreira, A. P., Quétu, V., & Tartaglione, E. (2025). LayerFold: A Python library to reduce the depth of neural networks. SoftwareX, 29, 102030.

Is it straightforward? [Pilo et al. 2025]

Case of study: convolutional layers, stride=1, padding=1

To the moment, no closed-form solutions!

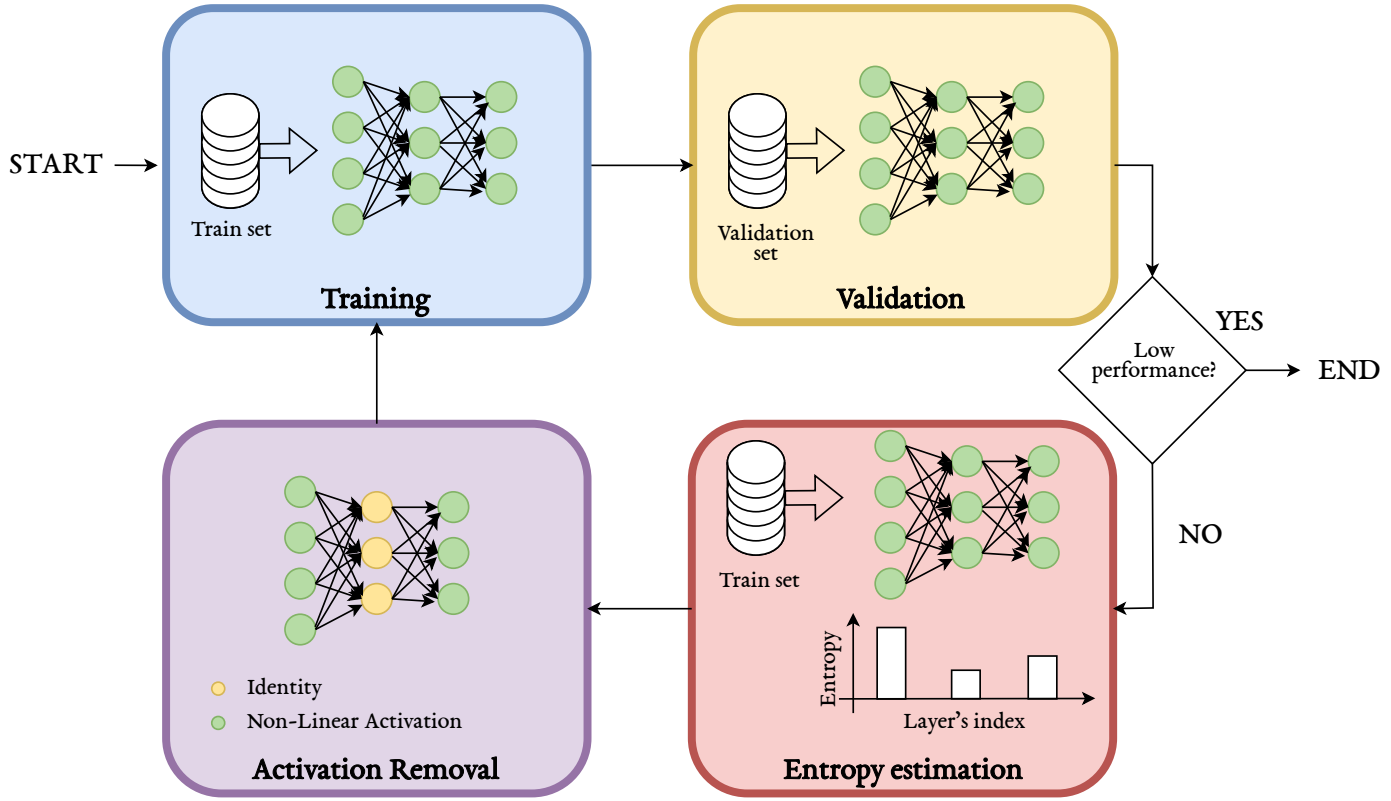
Is this case that much frequent?

Most of the architectures using skip connections make use of padding to maintain same dimensionality of the output!

Pilo, G., Hezbri, N., e Ferreira, A. P., Quétu, V., & Tartaglione, E. (2025). LayerFold: A Python library to reduce the depth of neural networks. SoftwareX, 29, 102030.

OK, how can we remove layers?

The naive way: EASIER [Quetu et al. 2024]



Quétu, V., Liao, Z., & Tartaglione, E. (2024, August). The simpler the better: An entropy-based importance metric to reduce neural networks' depth. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 92-108). Cham: Springer Nature Switzerland.

The naive way: EASIER [Quetu et al. 2024]

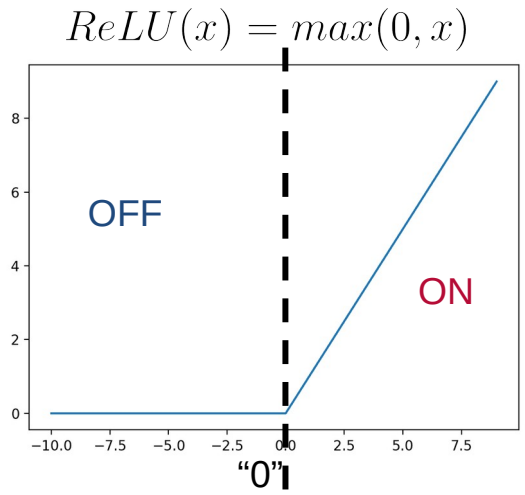
Three "states" for the neuron can be identified:

$$s_{l,i}^{\boldsymbol{x}} = sign(z_{l,i}^{\boldsymbol{x}}) = \begin{cases} +1 & \text{if } y_{l,i}^{\boldsymbol{x}} > 0 \\ -1 & \text{if } y_{l,i}^{\boldsymbol{x}} < 0 \\ 0 & \text{if } y_{l,i}^{\boldsymbol{x}} = 0 \end{cases}$$

The probability of the i-th neuron belonging to ON state can be calculated from the average over a batch of outputs for this neuron, and from this we define an estimator of neuron's degeneration as

$$\mathcal{H}_{l,i} = -\sum_{s_{l,i}=\pm 1} p(s_{l,i}) \log_2 \left[p(s_{l,i})\right]$$

We can estimate layer's degeneration averaging over all the neurons.

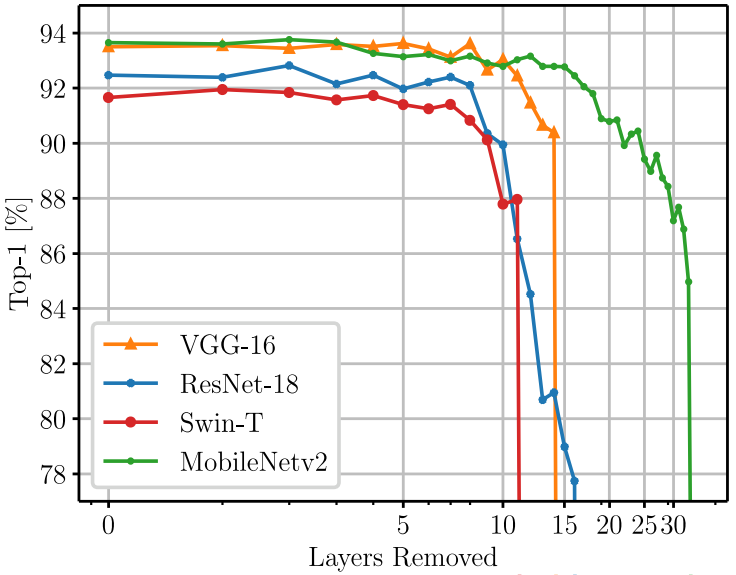


Quétu, V., Liao, Z., & Tartaglione, E. (2024, August). The simpler the better: An entropy-based importance metric to reduce neural networks' depth. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 92-108). Cham: Springer Nature Switzerland.

The naive way: EASIER [Quetu et al. 2024]

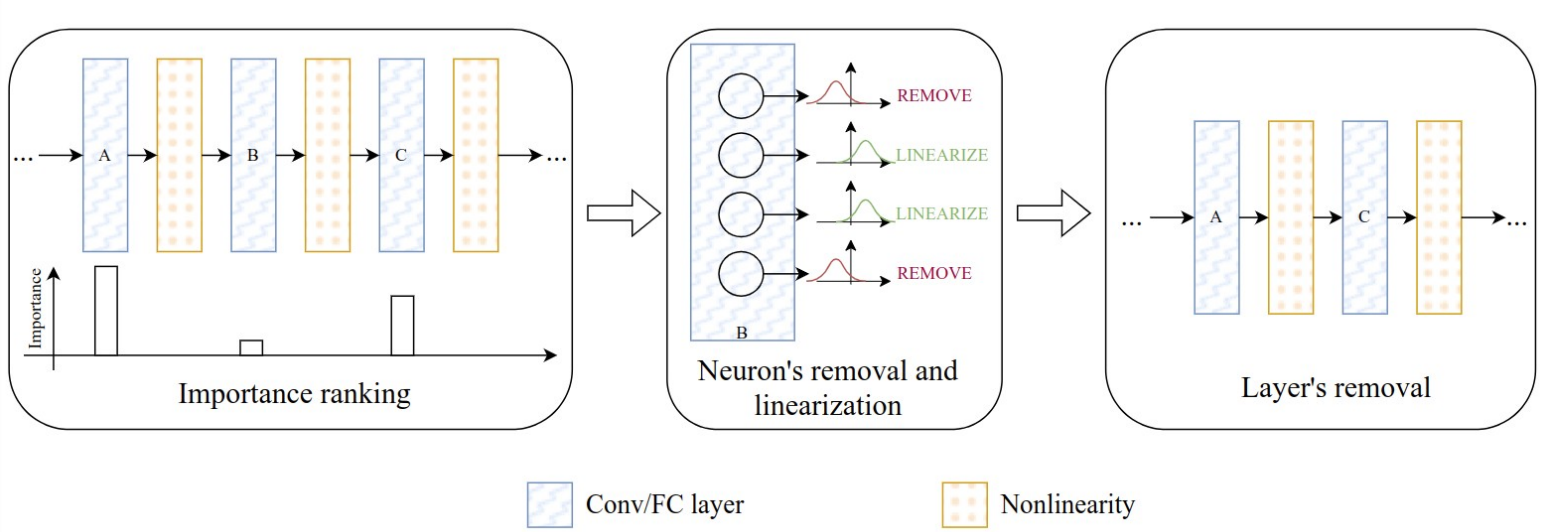
EASIER applied on ResNet-18, VGG-16, Swin-T and MobileNetv2 on CIFAR-10. For each model, we gradually remove non-linear layers.

Rem.	MFLOPs	Inference on CPU [ms]		Inference on GPU [ms]			
		Xeon E5-2640	Raspi 4	Jetson Orin	P2000	RTX 2080	A4500
0/17	725,47	13,50	135	8,52	4,45	4,43	3,32
1/17	258,24	9,33	111	8,31	4,53	4,43	3,27
2/17	243,46	9,69	106	7,83	4,28	4,21	3,10
3/17	231,79	9,43	139	7,38	4,02	3,93	2,96
4/17	197,85	10,10	117	6,91	3,79	3,68	2,78
5/17	159,05	11,30	144	6,44	3,60	3,46	2,60
6/17	159,99	8,39	225	6,13	4,11	3,18	1,79
7/17	152,36	9,18	144	6,06	4,16	3,10	1,71
8/17	149,84	9,14	149	6,14	3,67	3,21	1,55



Quétu, V., Liao, Z., & Tartaglione, E. (2024, August). The simpler the better: An entropy-based importance metric to reduce neural networks' depth. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 92-108). Cham: Springer Nature Switzerland.

A better way: TLC [Liao et al. 2025]



Works by simply observing distributions in batch norms and in layer norms.

$$\hat{x}_{l,i} = \frac{x_{l,i} - \mu^B_{l,i}}{\sqrt{(\sigma^B_{l,i})^2 + \epsilon}}; \qquad z_{l,i} = \gamma_{l,i}\hat{x}_{l,i} + \beta_{l,i};$$

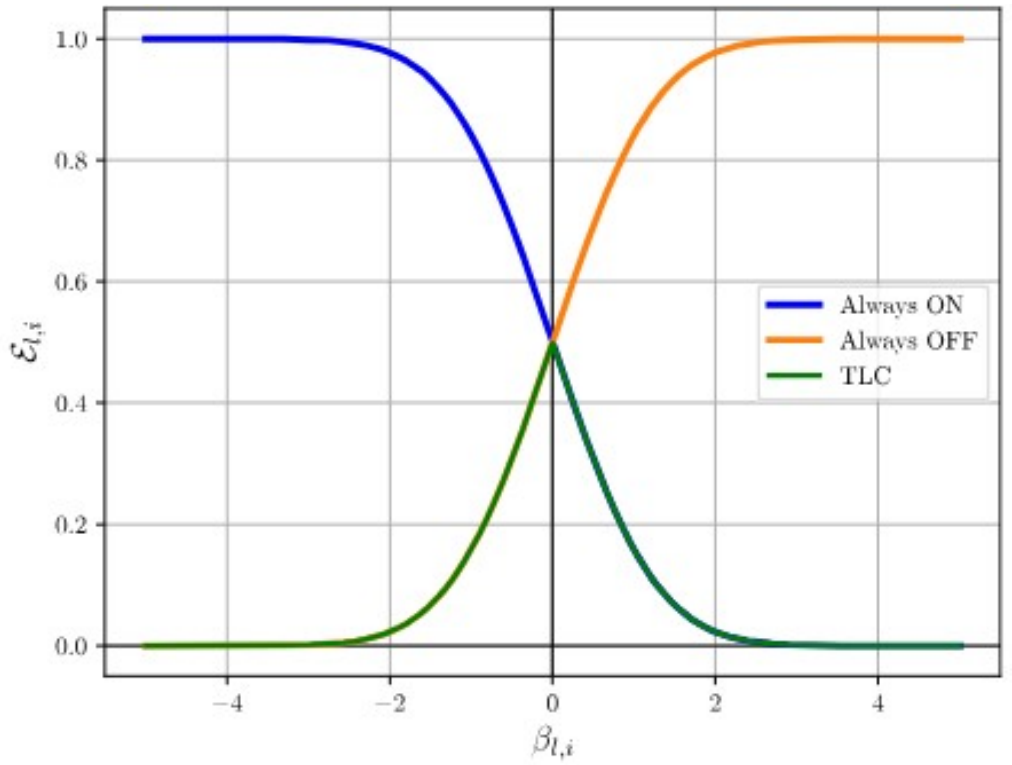
Still, sharing iterative fine-tuning nature, so computationally expensive….

Liao, Z., Hezbri, N., Quétu, V., Nguyen, V. T., & Tartaglione, E. (2025). Till the Layers Collapse: Compressing a Deep Neural Network through the Lenses of Batch Normalization Layers. AAAI 2025.

A better way: TLC [Liao et al. 2025]

$$\hat{x}_{l,i} = \frac{x_{l,i} - \mu^B_{l,i}}{\sqrt{(\sigma^B_{l,i})^2 + \epsilon}}; \qquad z_{l,i} = \gamma_{l,i}\hat{x}_{l,i} + \beta_{l,i};$$

$$\mathcal{E}_{l,i} = \Phi\left(-\frac{|\beta_{l,i}|}{\gamma_{l,i}}\right)$$



Liao, Z., Hezbri, N., Quétu, V., Nguyen, V. T., & Tartaglione, E. (2025). Till the Layers Collapse: Compressing a Deep Neural Network through the Lenses of Batch Normalization Layers. AAAI 2025.

Browsing the literature…

"Deeper layers are typically quite similar to each other and can be more easily dropped."

The Unreasonable Ineffectiveness of the Deeper Layers, ICLR 2024

Browsing the literature…

"Deeper layers are typically quite similar to each other and can be more easily dropped."

The Unreasonable Ineffectiveness of the Deeper Layers, ICLR 2024

"The changes in parameters and output representations between adjacent layers within the LLM are not particularly significant."

LaCo: Large Language Model Pruning via Layer Collapse, ICLR 2024

Browsing the literature…

"Deeper layers are typically quite similar to each other and can be more easily dropped."

The Unreasonable Ineffectiveness of the Deeper Layers, ICLR 2024

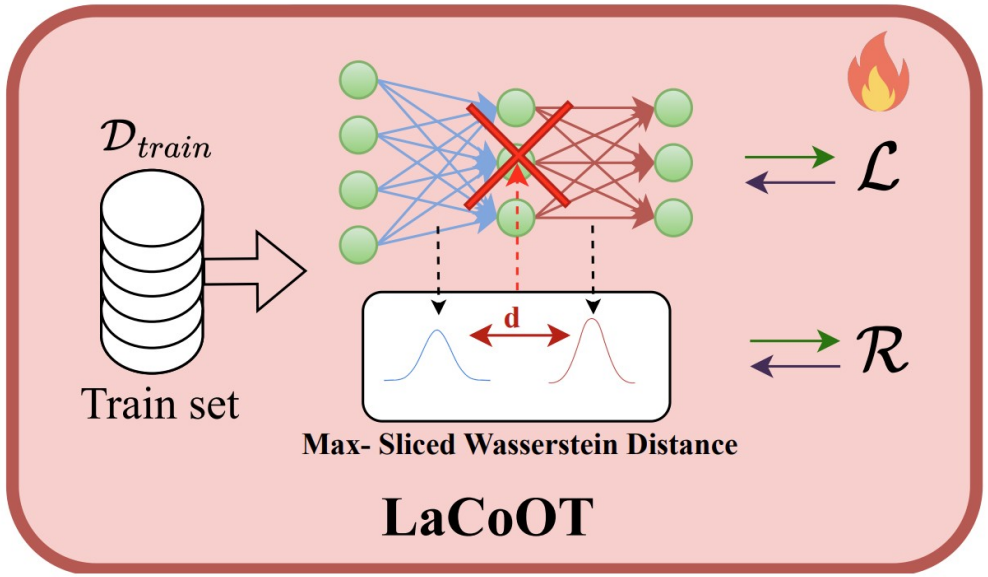
"The changes in parameters and output representations between adjacent layers within the LLM are not particularly significant."

LaCo: Large Language Model Pruning via Layer Collapse, ICLR 2024

"LLMs exhibit significant redundancy at the layer level, enabling the simple removal of entire layers without substantially impacting downstream task performance."

ShortGPT: Layers in Large Language Models are More Redundant Than You Expect, ICLR 2024

Layer Collapse by Regularization [Quetu et al. 2024]

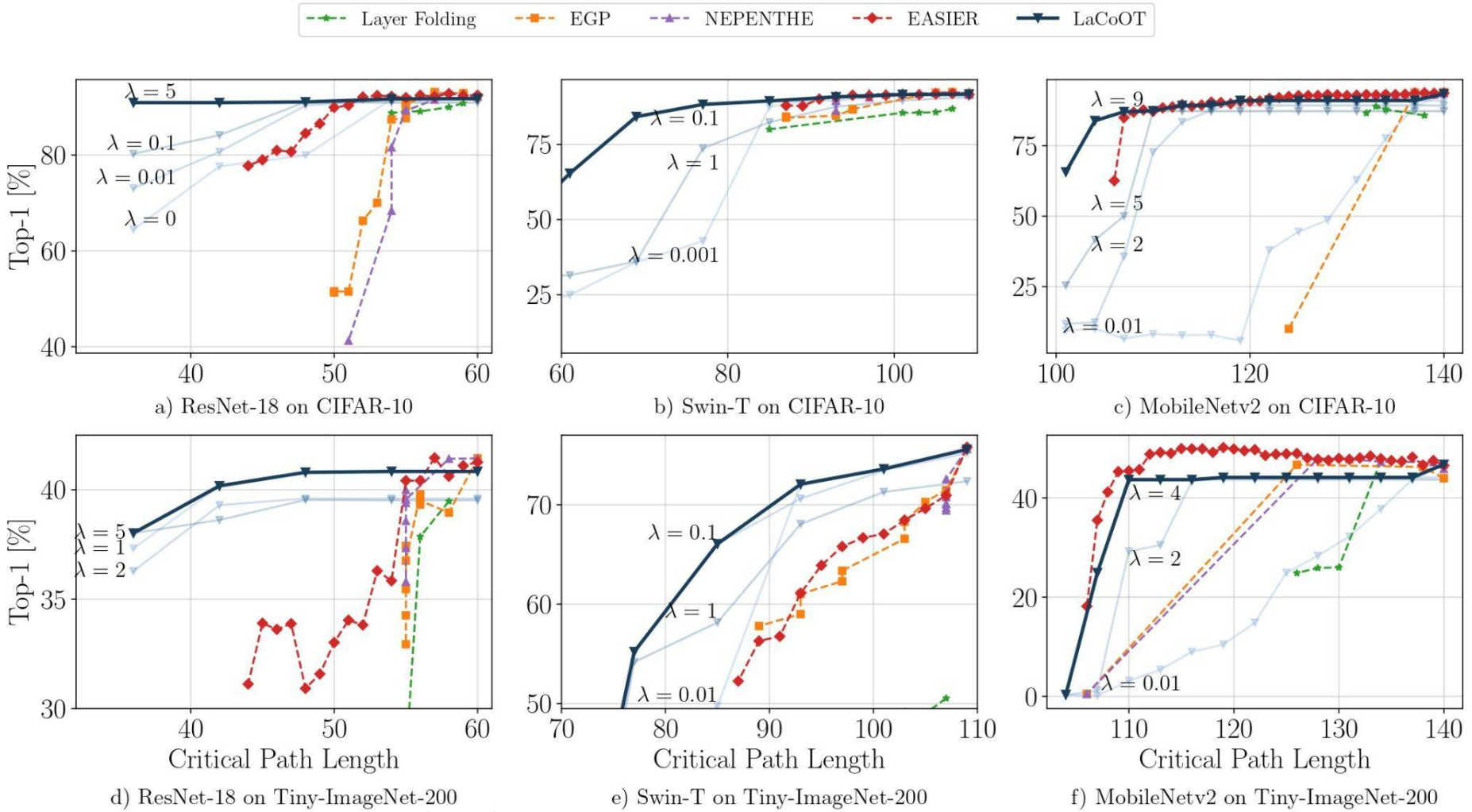


Training performed by minimizing distribution distance between adjacent layers (or group of) through a regularization term.

Layer pruning is performed in one-shot without retraining.

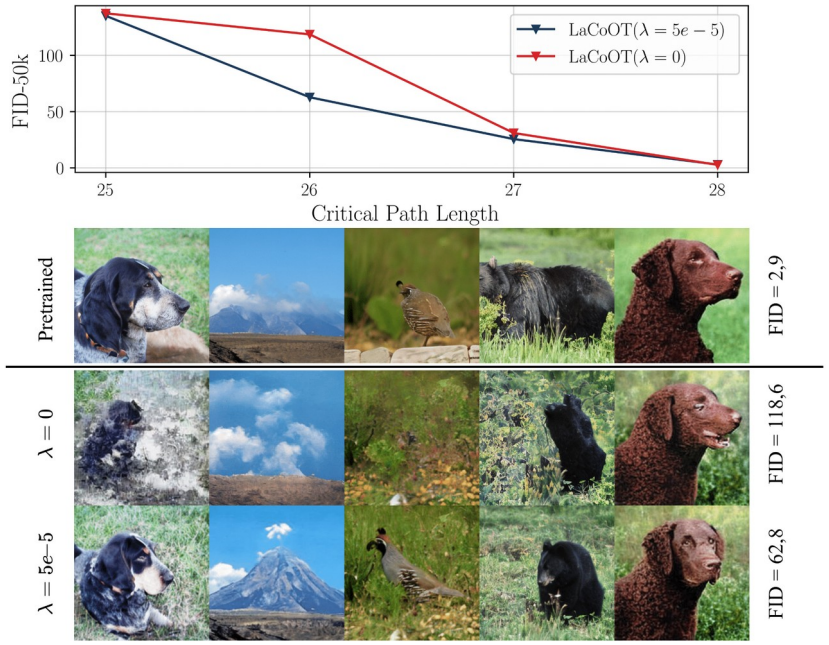
Quétu, V., Hezbri, N., & Tartaglione, E. (2024). LaCoOT: Layer Collapse through Optimal Transport. arXiv preprint arXiv:2406.08933.

Layer Collapse by Regularization [Quetu et al. 2024]



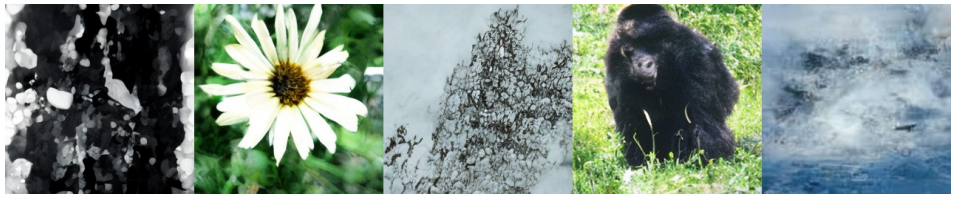
Quétu, V., Hezbri, N., & Tartaglione, E. (2024). LaCoOT: Layer Collapse through Optimal Transport. arXiv preprint arXiv:2406.08933.

Layer Collapse by Regularization [Quetu et al. 2024]



FID-50k as a function of the critical path length achieved by a DiT-XL/2 finetuned on ImageNet.

Pretrained DiT-XL/2



DiT-XL/2 finetuned with λ=0 – 2 blocks removed

DiT-XL/2 finetuned with λ=5e-5 – 2 blocks removed

Quétu, V., Hezbri, N., & Tartaglione, E. (2024). LaCoOT: Layer Collapse through Optimal Transport. arXiv preprint arXiv:2406.08933.

Another dimension: token pruning

Why token pruning?

- Tokens process information in parallel in transformers.
- During forward propagation, it can happen that some encode exactly the same information.
- Removing redundant token saves memory and computation.

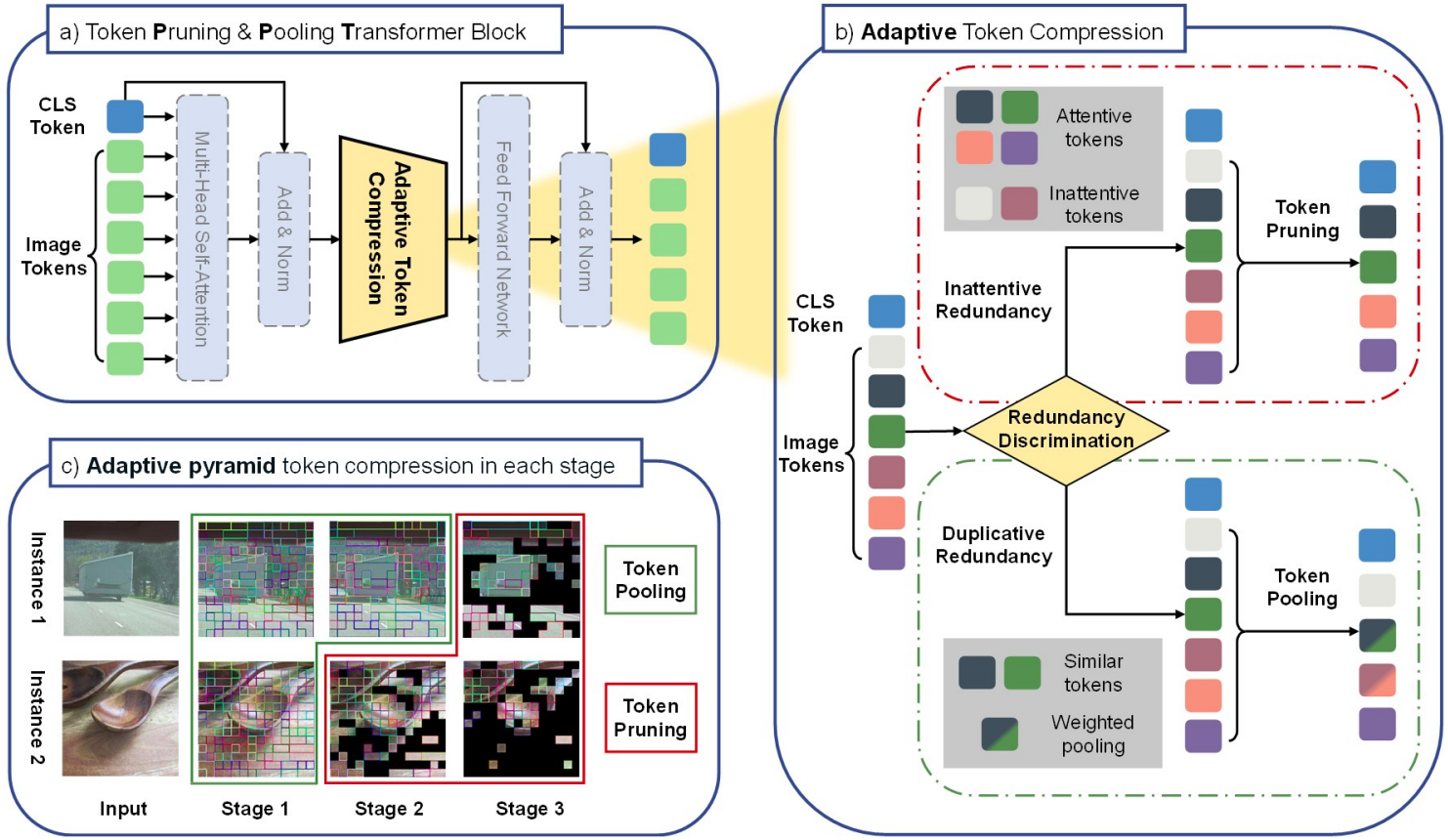
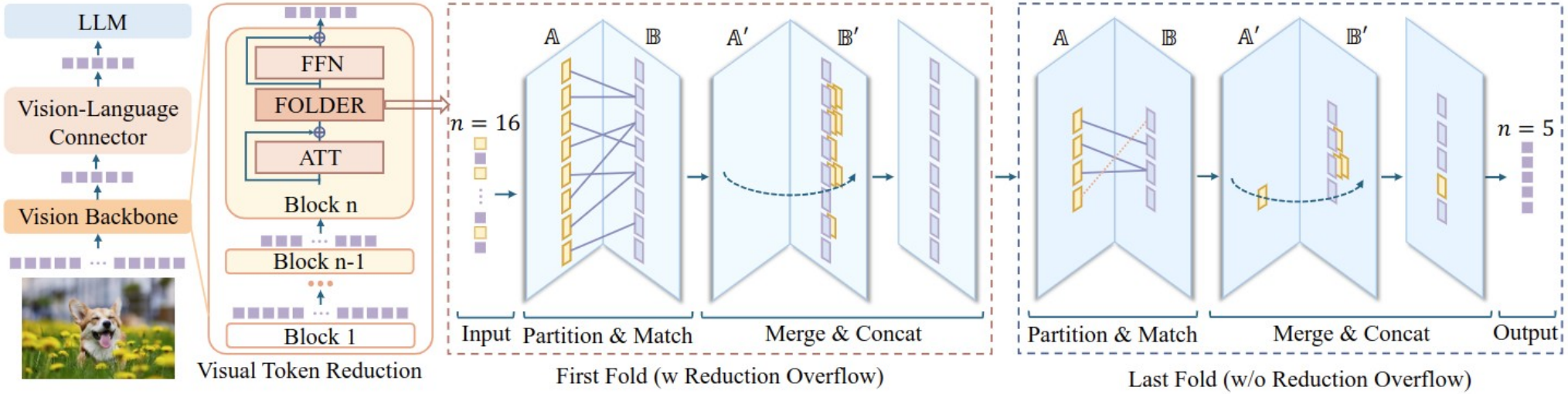


Image from https://paperswithcode.com/paper/ppt-token-pruning-and-pooling-for-efficient

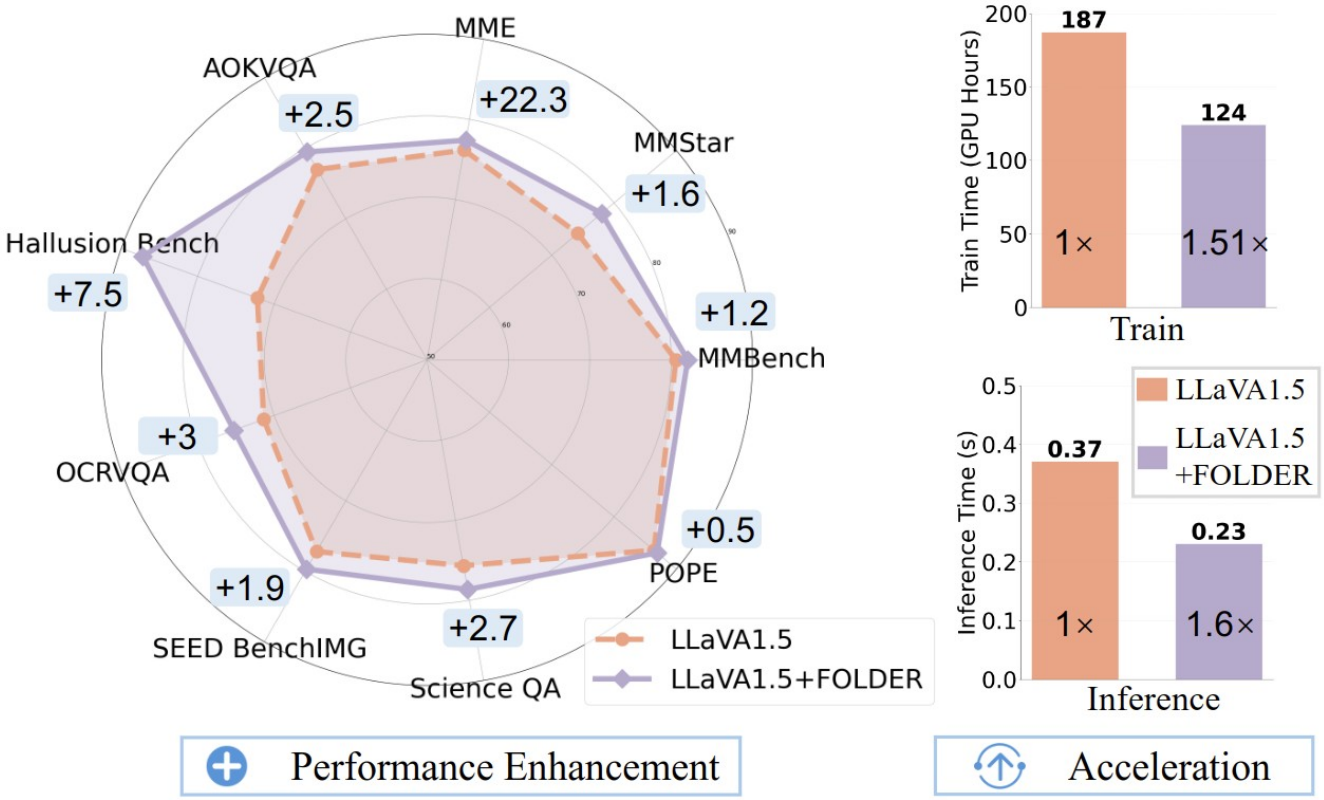
FOLDER [Wang et al. 2025]



As a plug-and-play module for speeding-up training and inference, FOLDER is integrated into the final blocks of the vision backbone (last two here). To deal with reduction overflow, FOLDER automatically executes another FOLD operation when the expected reduction is more than half. The last FOLD, which escapes from reduction overflow, merges tokens according to the remaining reduction numbers.

Wang, H., Yu, Z., Spadaro, G., Ju, C., Quétu, V., & Tartaglione, E. (2025). FOLDER: Accelerating Multi-modal Large Language Models with Enhanced Performance. arXiv preprint arXiv:2501.02430.

FOLDER [Wang et al. 2025]



Wang, H., Yu, Z., Spadaro, G., Ju, C., Quétu, V., & Tartaglione, E. (2025). FOLDER: Accelerating Multi-modal Large Language Models with Enhanced Performance. arXiv preprint arXiv:2501.02430.

Other current challenges…

- Early Exit strategies: a sort of “layer pruning on-demand”, where execution on some layers is conditioned to the current input.

Other current challenges…

- Early Exit strategies: a sort of “layer pruning on-demand”, where execution on some layers is conditioned to the current input.
- Joining pruning with knowledge distillation: already implicitly happening…

Other current challenges…

- Early Exit strategies: a sort of “layer pruning on-demand”, where execution on some layers is conditioned to the current input.
- Joining pruning with knowledge distillation: already implicitly happening…
- Pruning the back-propagation graph: some neurons are not updated, once they reach “convergence”.

Other current challenges…

- Early Exit strategies: a sort of “layer pruning on-demand”, where execution on some layers is conditioned to the current input.
- Joining pruning with knowledge distillation: already implicitly happening…
- Pruning the back-propagation graph: some neurons are not updated, once they reach “convergence”.
- Better pruning or quantization?

Other current challenges…

- Early Exit strategies: a sort of “layer pruning on-demand”, where execution on some layers is conditioned to the current input.
- Joining pruning with knowledge distillation: already implicitly happening…
- Pruning the back-propagation graph: some neurons are not updated, once they reach “convergence”.
- Better pruning or quantization?
- Lottery ticket at initialization…?

Other current challenges…

- Early Exit strategies: a sort of “layer pruning on-demand”, where execution on some layers is conditioned to the current input.
- Joining pruning with knowledge distillation: already implicitly happening…
- Pruning the back-propagation graph: some neurons are not updated, once they reach “convergence”.
- Better pruning or quantization?
- Lottery ticket at initialization…?
- How is now pruning different from Neural Architecture Search?


Q&A