A round-trip journey in pruned artificial neural networks

Andrea Bragagnolo[+], Enzo Tartaglione[*], Gianluca Dalmasso[+,x], Marco Grangetto[x]

[+] Synesthesia s.r.l. Torino
[*] TELECOM Paris, France
[x] Università di Torino

Convegno Nazionale CINI sull'Intelligenza Artificiale
29-31 maggio 2023

Outline

Introduction

Efficient deployment: Neural Network Pruning

Efficient training: Backward Pass Pruning

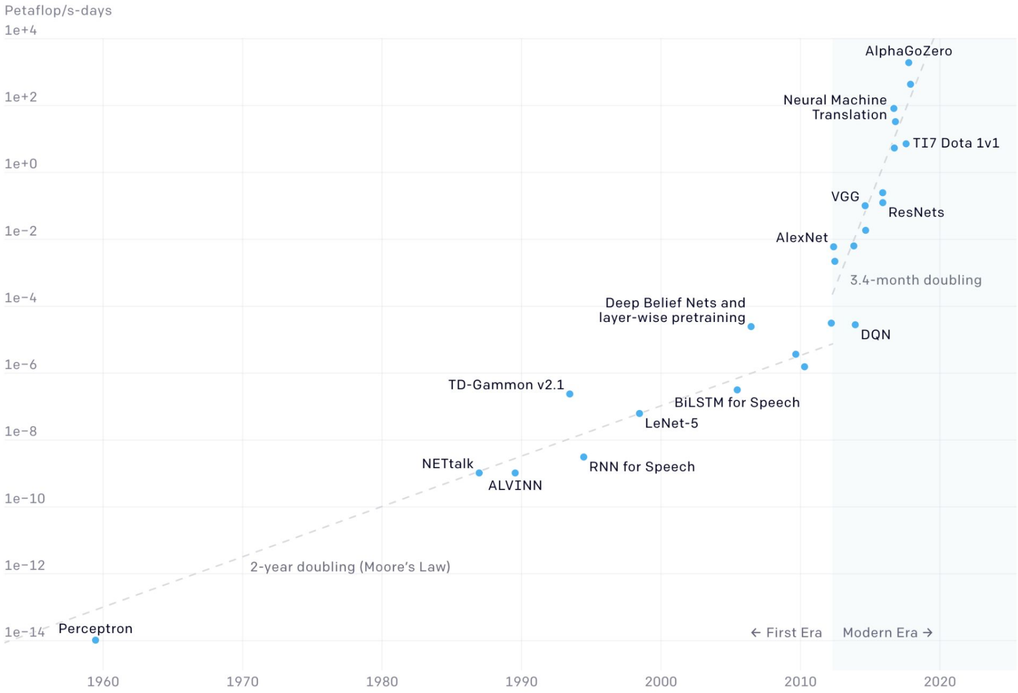
Conclusions

Deep models scale and **training cost**

State-of-the-art performance on complex tasks.

Very high number of parameters (hundreds of millions and even more…).

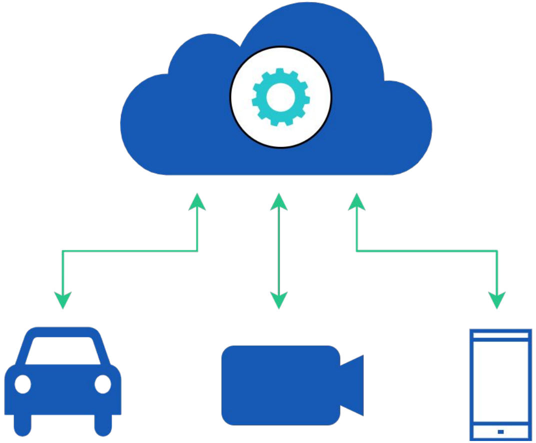
Training requires substantial computing power.



https://openai.com/blog/ai-and-compute/

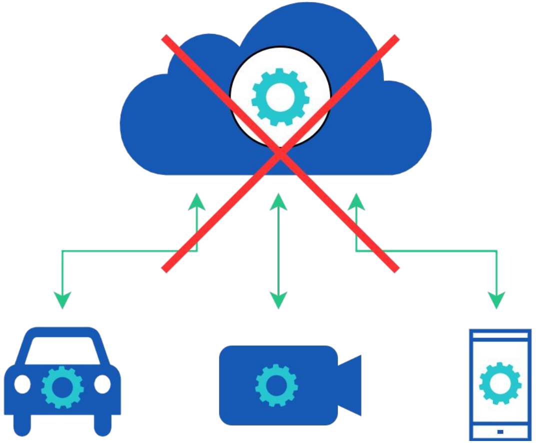
Deploying models

Cloud AI



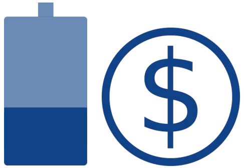
Inferences are performed remotely.

Edge AI

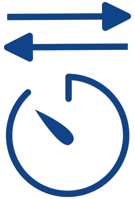


Inferences are performed locally.

Advantages of Edge AI



Costs &
Energy



Low latency

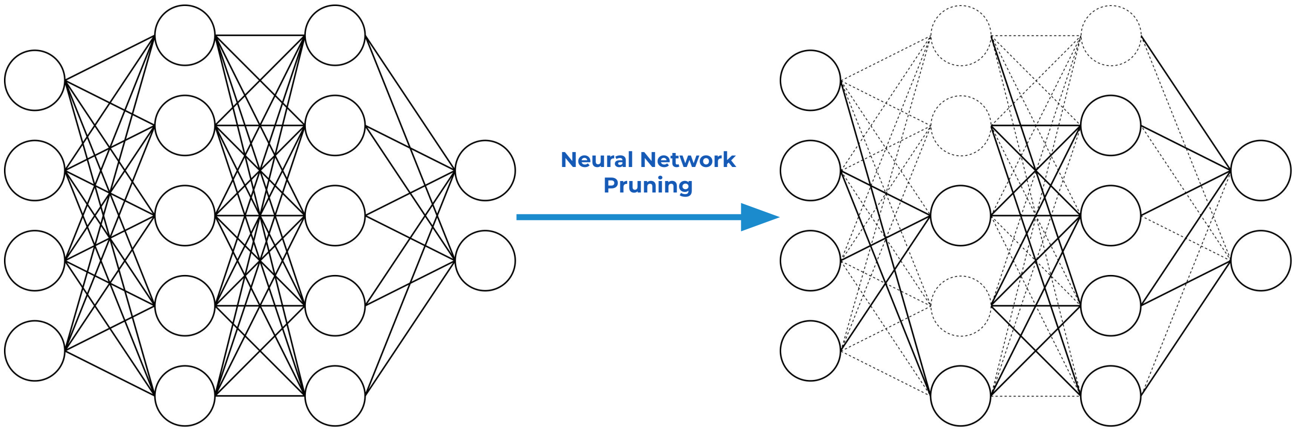
Privacy

Connectivity

Outline

Solution for efficient deployment

Neural Network Pruning



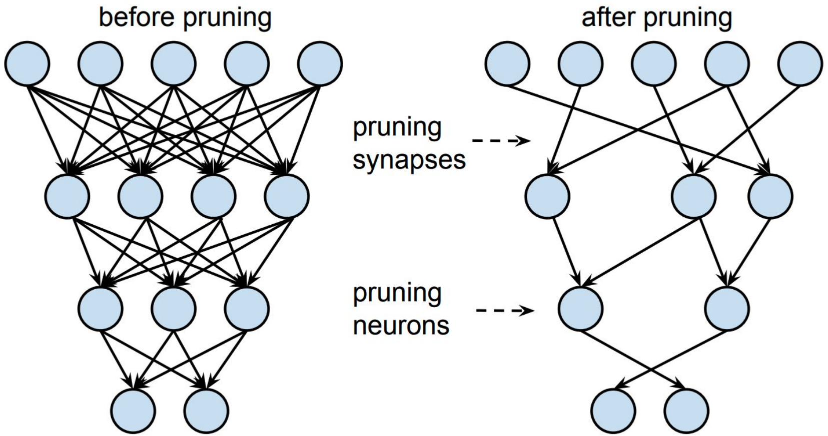
Neural Network Pruning

Intuition

Removes less influential elements while preserving the generalization capabilities.

Reduces the resources required to use the model.

Studied since the late ‘80s has seen a resurgence in 2015.



Categorization of Pruning Procedures

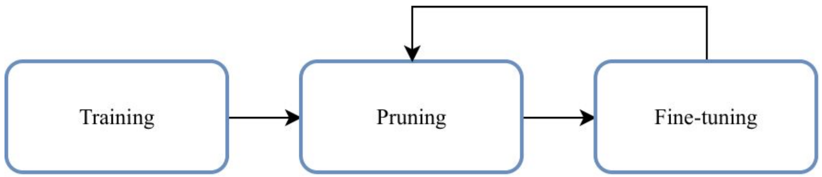
One-shot vs. Iterative

One-Shot

Performs a single pruning step.

Fine-tuning to recover performance.

Faster procedure.



Iterative

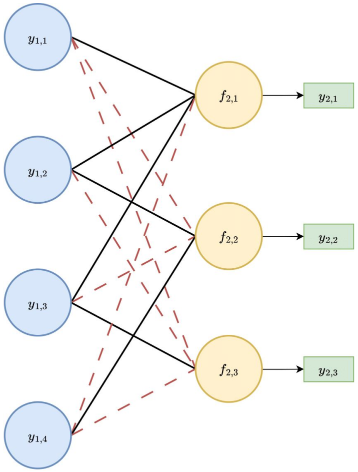
Performs multiple pruning step.

Successive training and pruning steps.

Prunes more parameters.

Categorization of Pruning Procedures

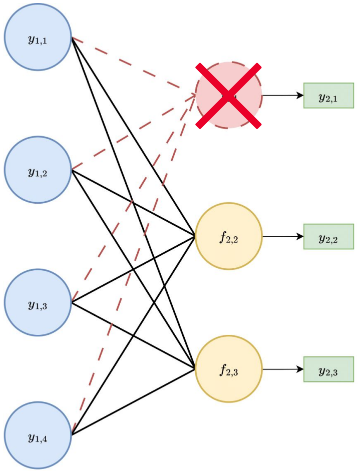
Unstructured vs. Structured



Unstructured

Removes many parameters from the network.

Highly reduces the compressed model size.



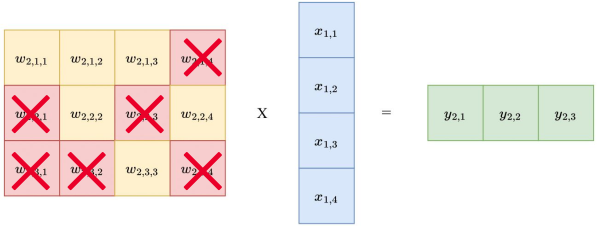
Structured

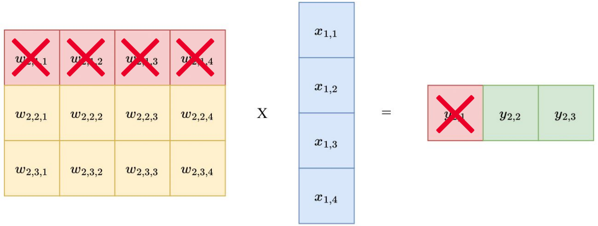
Removes entire neurons in the network.

Reduces the number of operations.

Categorization of Pruning Procedures

Unstructured vs. Structured





Unstructured

No guarantee in removing neurons.

We still consider the entire matrix to define the output

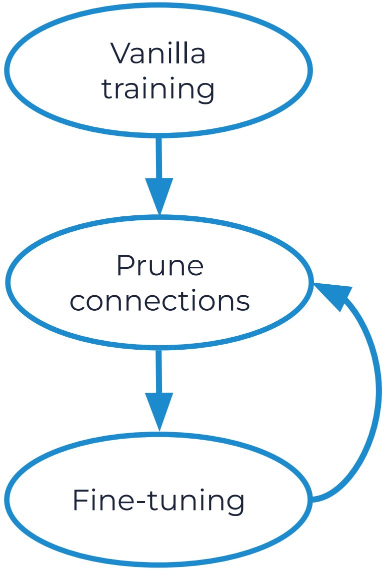
Structured

Removes entire rows from the matrix.

The rank of the final matrix is lower.

Learning Both Weights and Connections

A Template for Modern Techniques



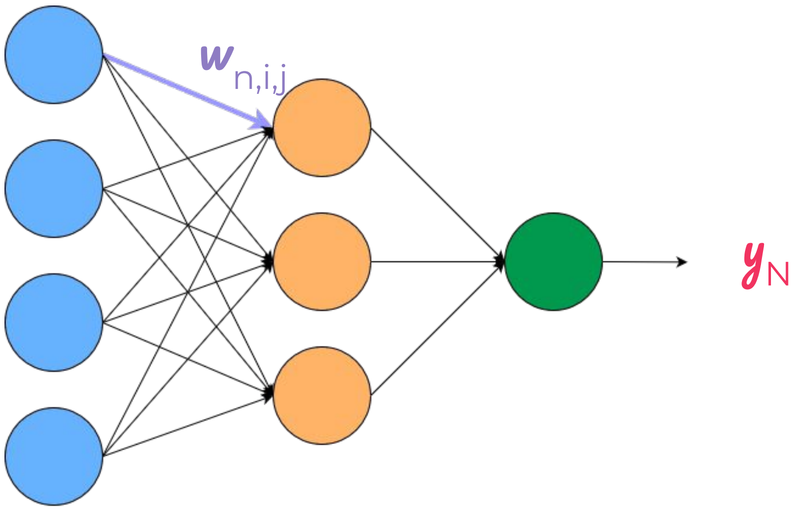
Han et al., *Learning both weights and connections for efficient neural network (2015)*.

Kick-started modern pruning research.

Unstructured, iterative.

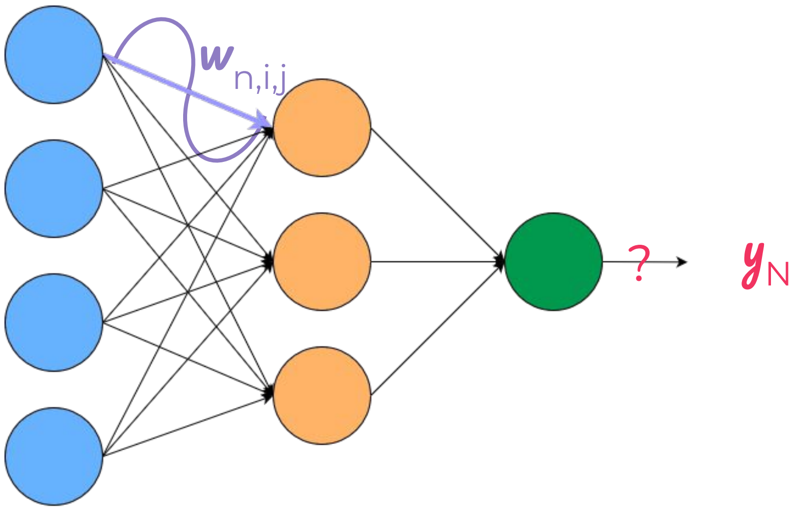
Acts as the foundation for the proposed procedures.

Sensitivity Regularization



Standard Feed Forward Neural Network.

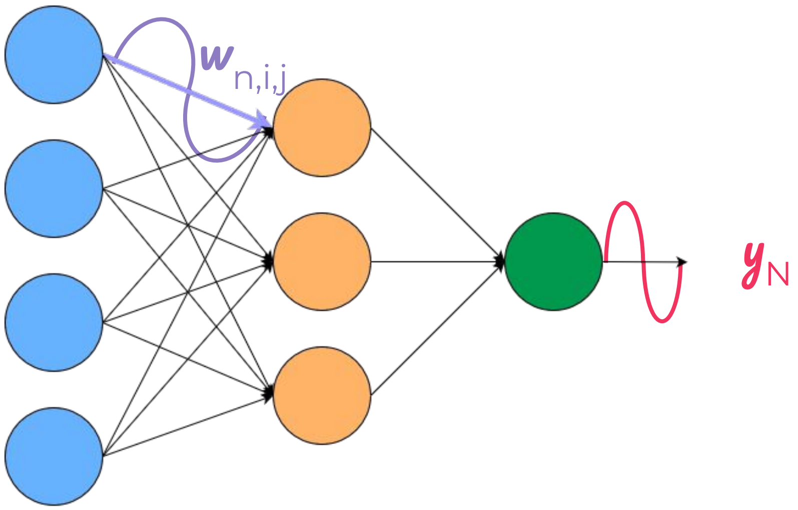
Sensitivity Regularization



Standard Feed Forward Neural Network.

Our goal: assess to which extent changes in the value of the weight $w_{n,i,j}$ would affect the output y_N.

Sensitivity Regularization



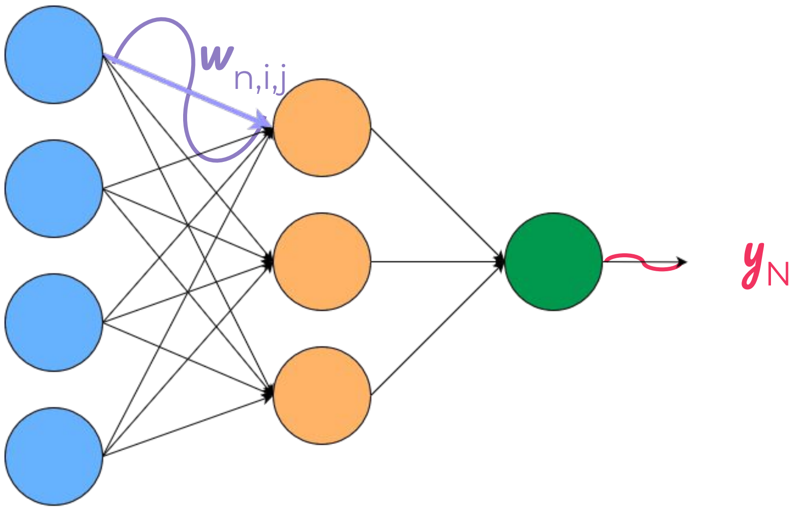
Standard Feed Forward Neural Network.

Our goal: assess to which extent changes in the value of the weight $w_{n,i,j}$ would affect the output y_N.

Intuitively:

- If the change in y_N is big, $w_{n,i,j}$ has a high Sensitivity.

Sensitivity Regularization



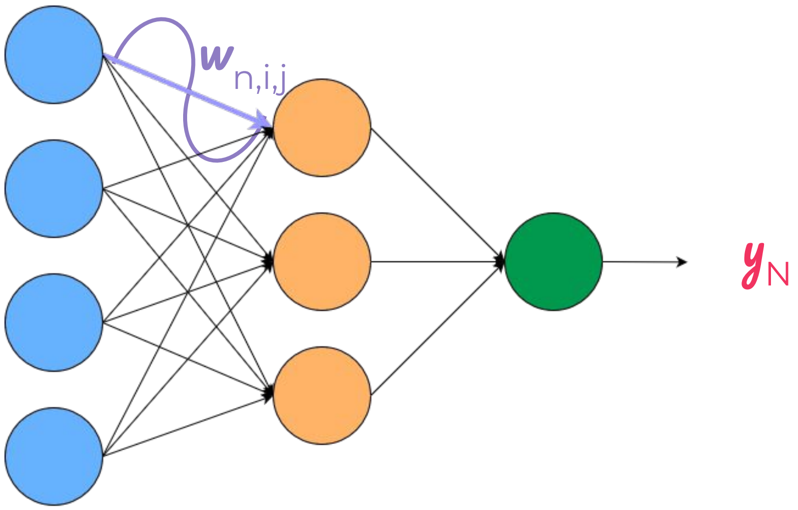
Standard Feed Forward Neural Network.

Our goal: assess to which extent changes in the value of the weight $w_{n,i,j}$ would affect the output y_N.

Intuitively:

- If the change in y_N is big, $w_{n,i,j}$ has a high Sensitivity.
- If the change in y_N is small, $w_{n,i,j}$ has a low Sensitivity.

Sensitivity Regularization



Standard Feed Forward Neural Network.

Our goal: assess to which extent changes in the value of the weight $w_{n,i,j}$ would affect the output y_N.

Intuitively:

- If the change in y_N is big, $w_{n,i,j}$ has a high Sensitivity.
- If the change in y_N is small, $w_{n,i,j}$ has a low Sensitivity.

$$S(y, w_j) = \sum_{k=1}^{C} \alpha_k \left| \frac{\partial y_k}{\partial w_j} \right|$$

Tartaglione et al., Learning sparse neural networks via sensitivity-driven regularization (2018).

Our pruning techniques

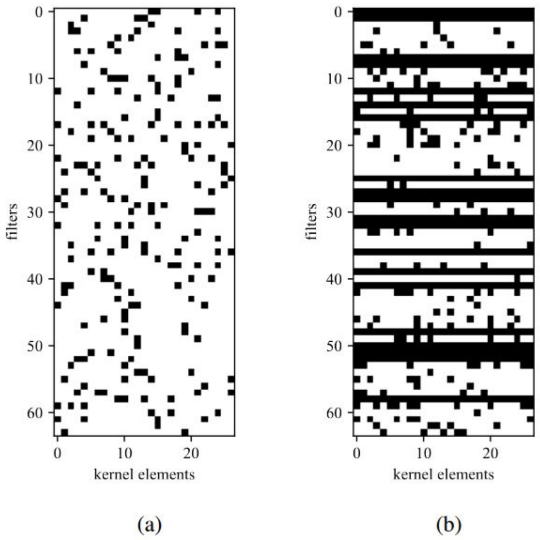
LOBSTER

Contribution of the parameters to the loss of the network.

$$S(\mathcal{L}, w_{n,i,j}) = \left| \frac{\partial \mathcal{L}}{\partial w_{n,i,j}} \right|$$

Unstructured.

"Loss-based sensitivity regularization: towards deep sparse neural networks." ***Neural Networks*** *(2022).*



SeReNe

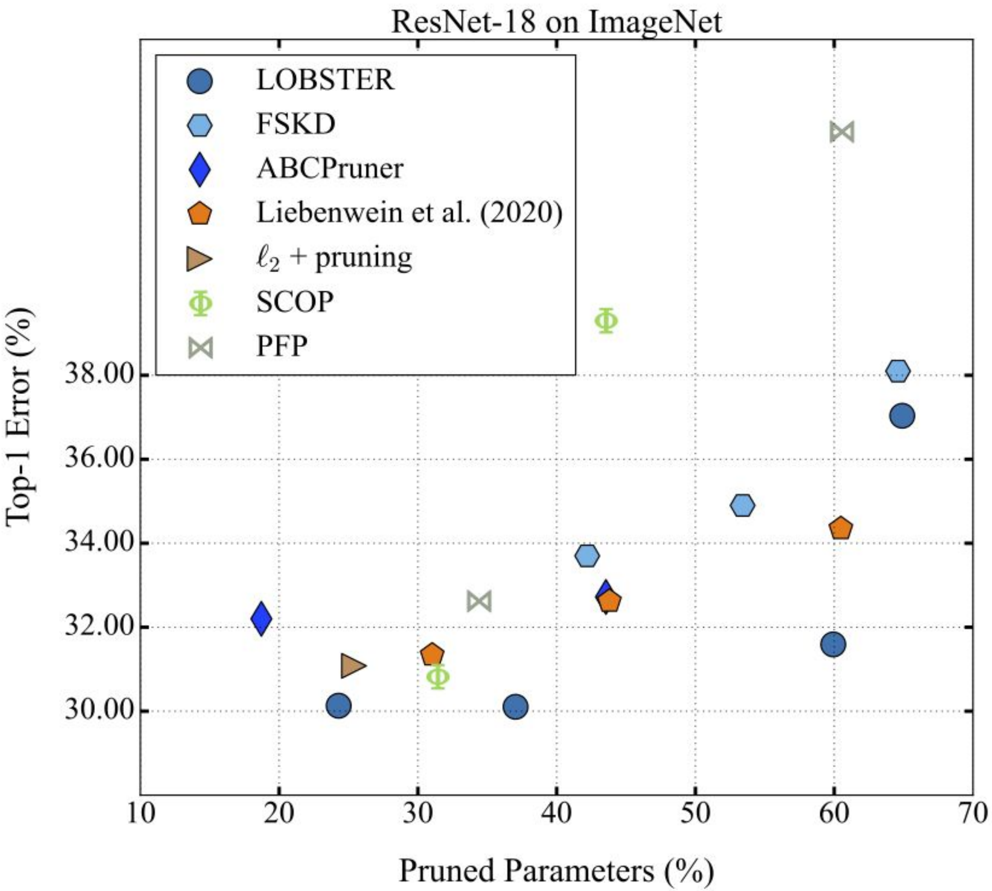
Contribution of the neuron to the output of the network.

$$S_{n,i}(\mathbf{y}_N, p_{n,i}) = \frac{1}{C} \sum_{k=1}^{C} \left| \frac{\partial y_{N,k}}{\partial p_{n,i}} \right|$$

Structured.

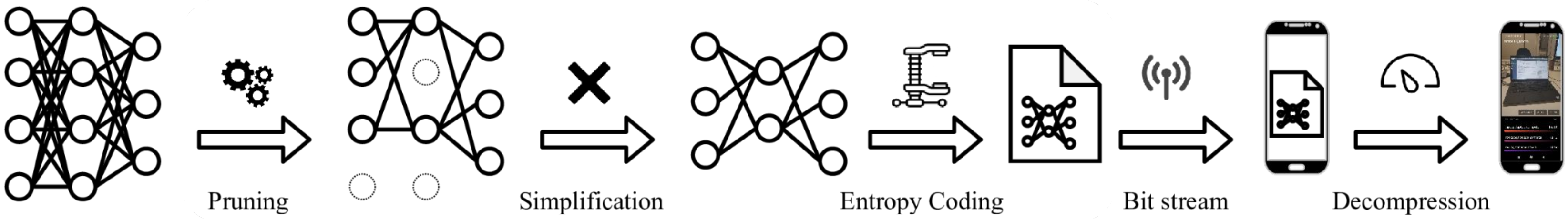
"SeReNe: Sensitivity-based regularization of neurons for structured sparsity in neural networks." ***IEEE Transactions on Neural Networks and Learning Systems*** *(2021).*

Pruning potential



Pruning in the MPEG-7 Part 17 pipeline

Experimental Setup

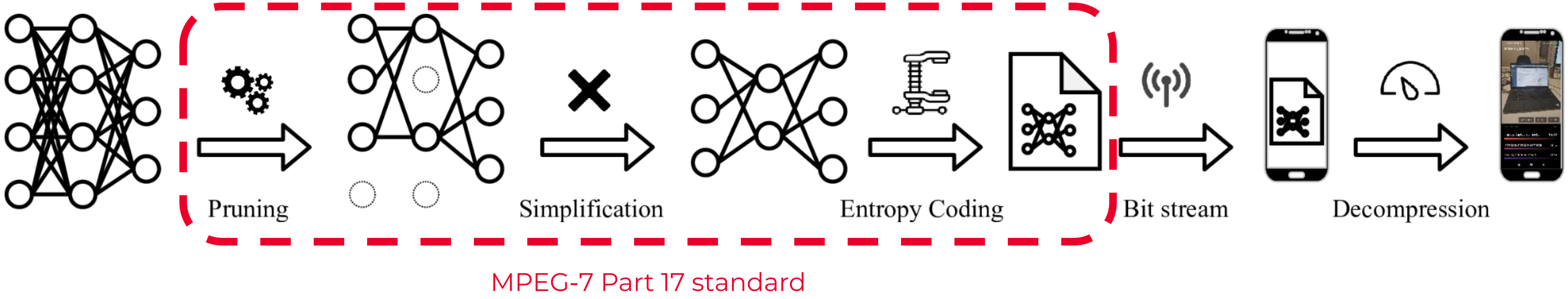


Evaluate the benefits of structured pruning approaches
within the MPEG-7 Part 17 neural network compression pipeline.

"On the role of structured pruning for neural network compression." 2021 IEEE International Conference on Image Processing (2021).

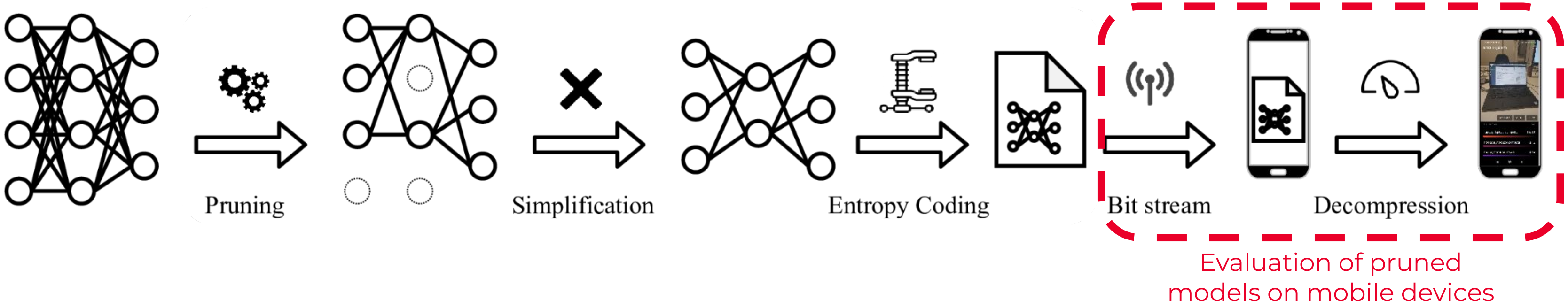
Pruning in the MPEG-7 Part 17 pipeline

Experimental Setup



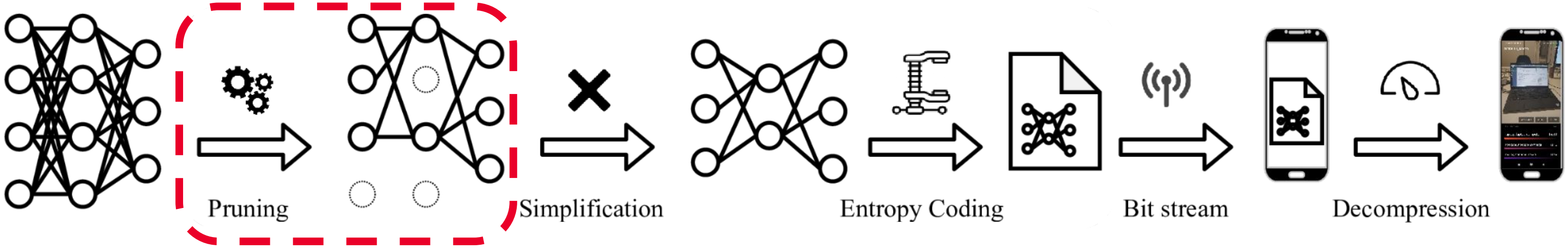
Pruning in the MPEG-7 Part 17 pipeline

Experimental Setup



Pruning in the MPEG-7 Part 17 pipeline

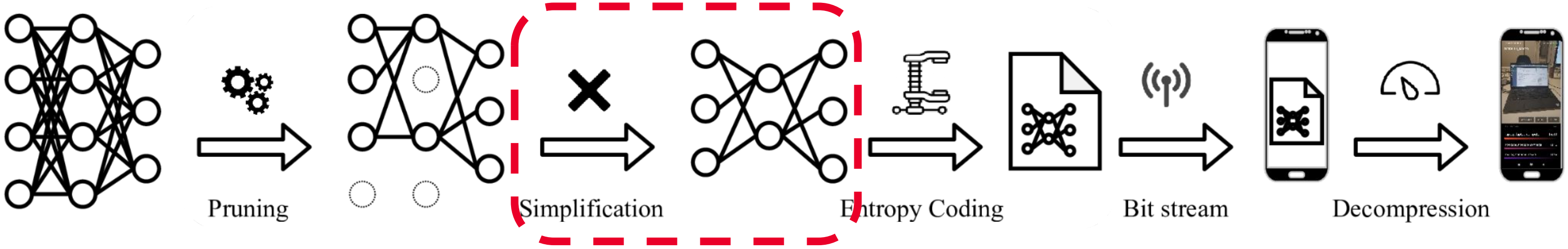
Experimental Setup



Parameter pruning:
SeReNe vs LOBSTER

Pruning in the MPEG-7 Part 17 pipeline

Experimental Setup



Removal of pruned neurons: Simplify

Pruning in the MPEG-7 Part 17 pipeline

Evaluation Results

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
		SeReNe	47.16	**31.02**	**0.34**	**594**	**99**	**85**	**106**
	ResNet-32	No pruning	-	2.0	0.30	580	32	30	31
		LOBSTER	**81.19**	1.96	0.12	545	32	26	30
		SeReNe	52.80	**1.0**	**0.09**	**536**	**25**	**17**	**25**
CIFAR-100	AlexNet	No pruning	-	94.6	10.1	246	131	84	168
		LOBSTER	**98.90**	48.84	0.40	224	95	67	120
		SeReNe	59.87	**37.07**	**0.20**	**186**	**75**	**53**	**96**
ImageNet	ResNet-101	No pruning	-	178.4	26.24	11919	958	416	1008
		LOBSTER	**87.39**	173.87	9.24	11879	956	403	985
		SeReNe	1.09	**172.53**	**7.51**	**11699**	**929**	**371**	**974**

Even removing less parameters, SeReNe produces smaller and faster models.

Outline

Introduction

Efficient deployment: Neural Network Pruning

Efficient training: Backward Pass Pruning

Conclusions

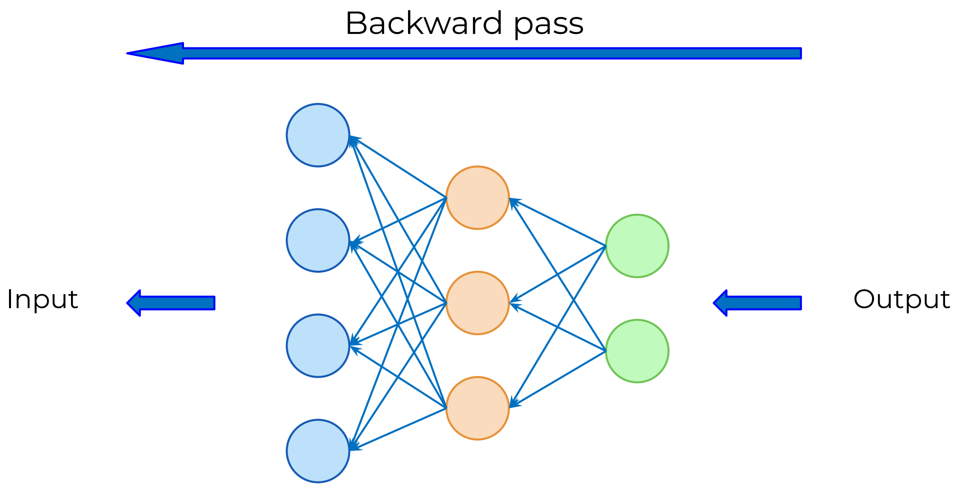
Reducing the Training Cost

We have seen how pruning can reduce the resources required by a deployed model, but what about the training process?

It is true that a model will perform thousands of inferences but the training is still very expensive.

Reducing the Training Cost

The Idea



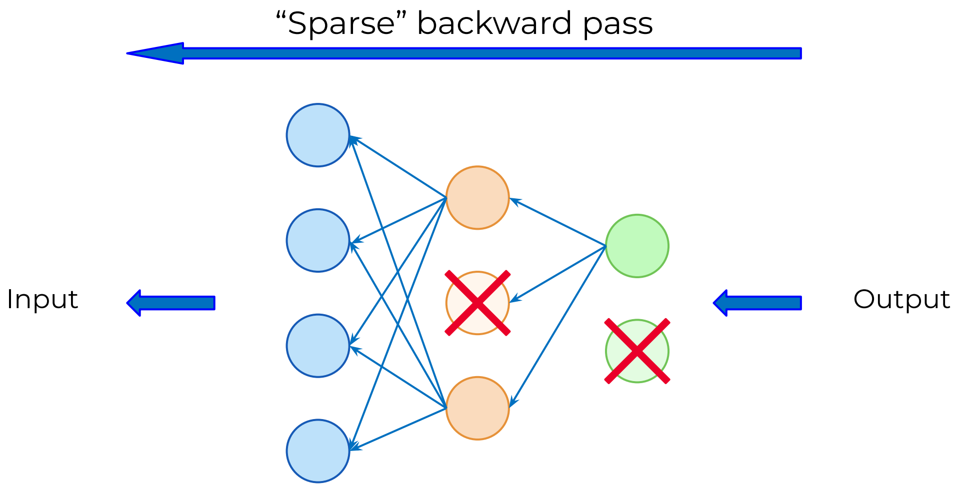
Backpropagation is the more computational-heavy part of the training.

We can reduce the training cost by slimming the backpropagation.

How?

Reducing the Training Cost

The Idea

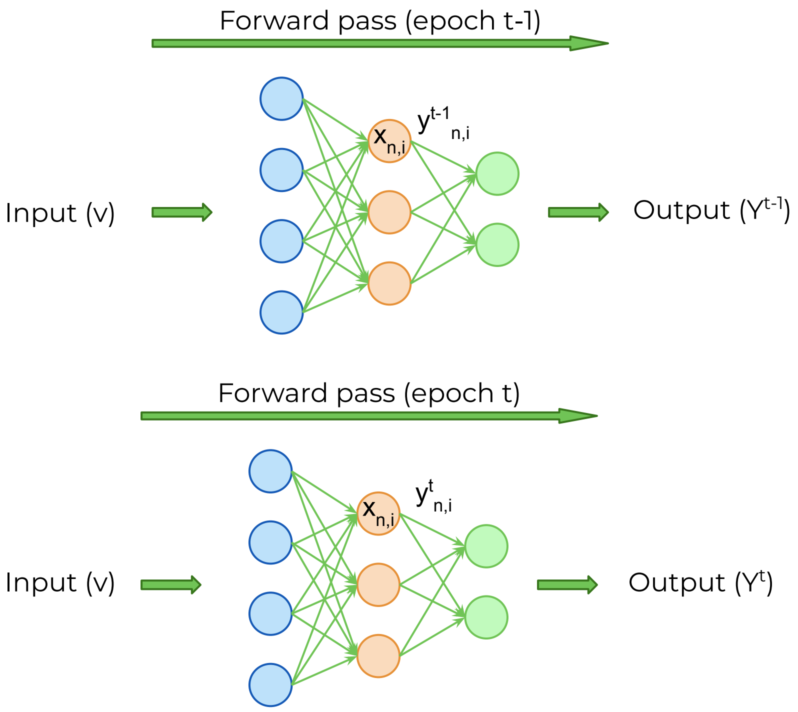


Are there neurons that “converge” before the end of the training? If so, we could disable their backpropagation.

How can we find these neurons?

Reducing the Training Cost

Equilibrium evaluation



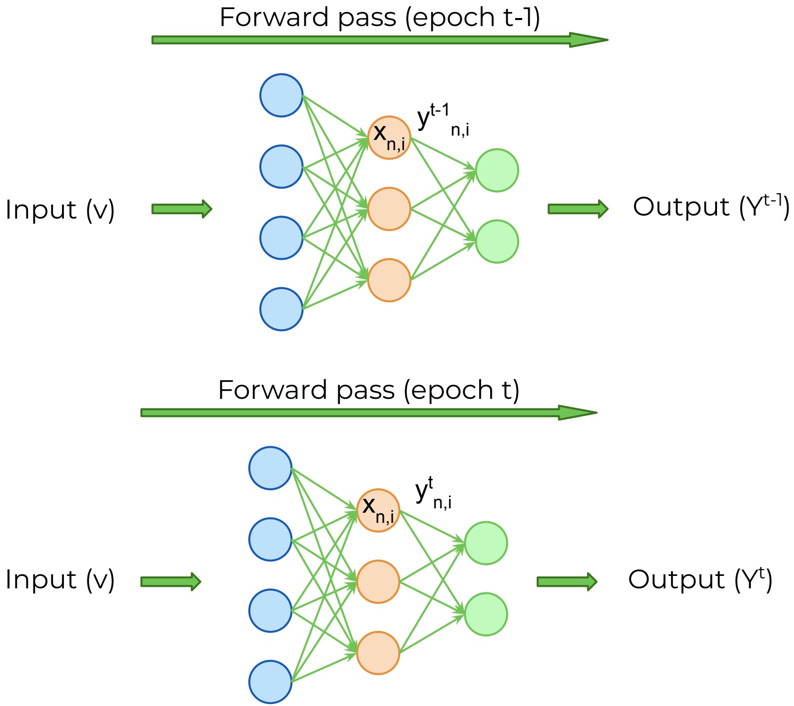
We evaluate a neuron's state and disable the update for neurons that reached equilibrium.

We consider the neuron's output in 2 adjacent epochs and evaluate the similarity.

$$\phi^t_{n,i} = \sum_{v}^{V} \sum_{m=1}^{M_i} y^t_{n,i,m,v} \cdot y^{t-1}_{n,i,m,v}$$

Reducing the Training Cost

Equilibrium evaluation



To asses the convergence to equilibrium, we evaluate the variation of similarities.

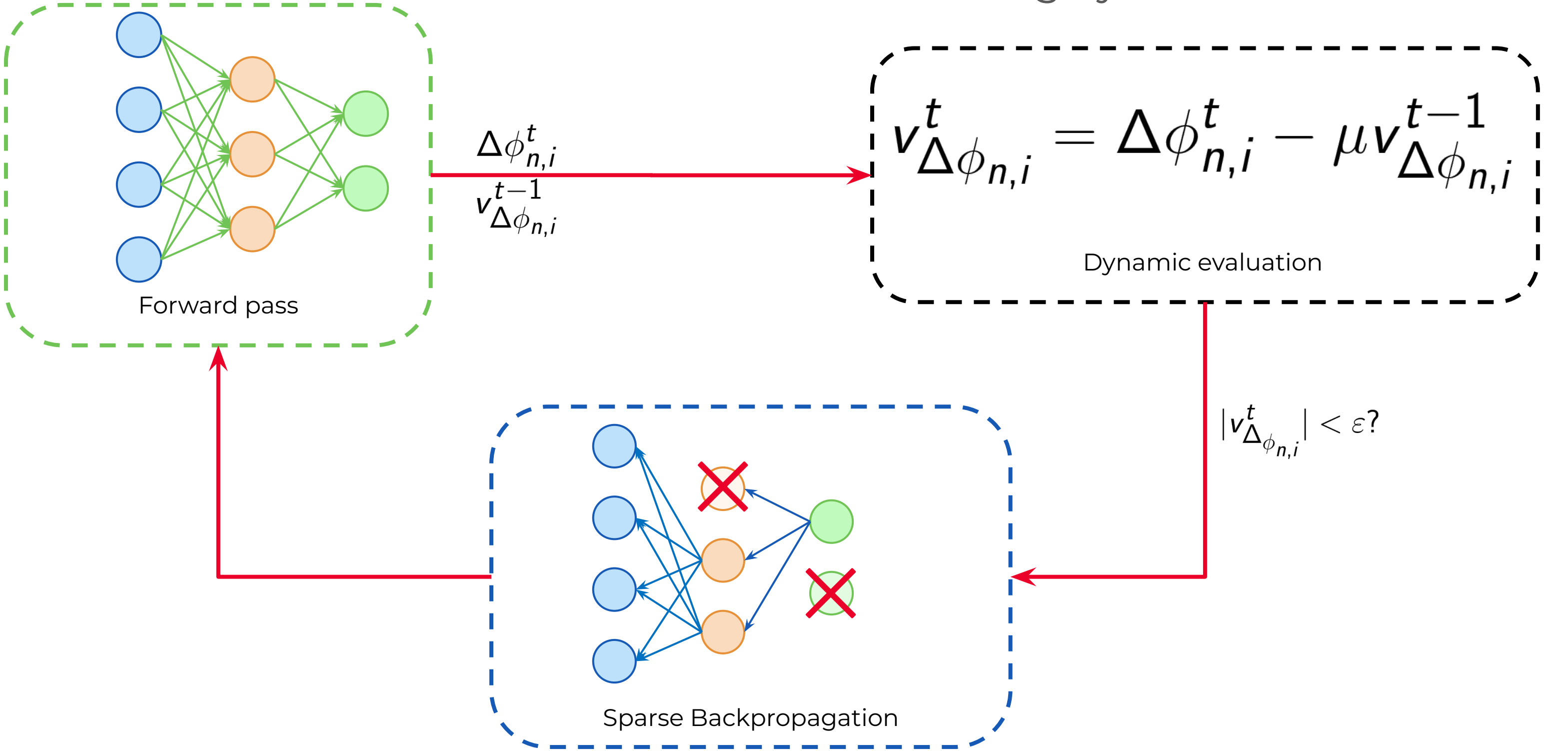
$$\Delta\phi^t_{n,i} = \phi^t_{n,i} - \phi^{t-1}_{n,i}$$

We say that we reach equilibrium when

$$\Delta\phi^t_{n,i} \rightarrow 0$$

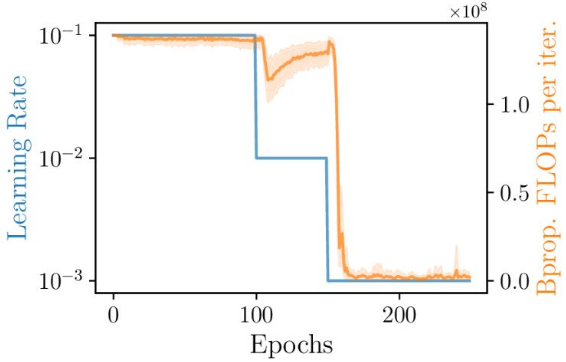
To update or not to update? Neuron at equilibrium

Advances in Neural Information Processing Systems 2022

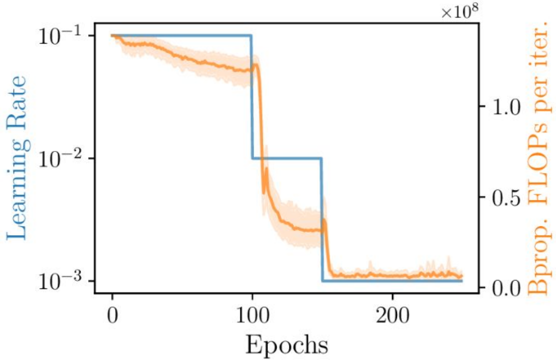


NEq

Some Neurons may Unfreeze



SGD



Adam

In the first phase of the train (high learning rate and stochastic noise) the amount of the trained neurons is higher.

Adam drives the neurons towards equilibrium faster.

At the first learning rate decay, for SGD, the number of updated neurons decreases and then increase, as SGD looks for large minima, preventing equilibrium in high learning rate regimes.

NEq

Experiments

We evaluate our approach on different combinations of architecture and dataset:

- ResNet-32 on CIFAR-10
- ResNet-18 on ImageNet
- Swin-B on Imagenet
- DeepLabV3 on COCO

All the learning policies used are borrowed from other works and are un-optimized to test the adaptability of NEq.

The pruning performance is evaluated according to multiple metrics:

- Average FLOPs per iteration at backpropagation.
- Final performance of the model evaluated on the test set (classification accuracy or IoU).

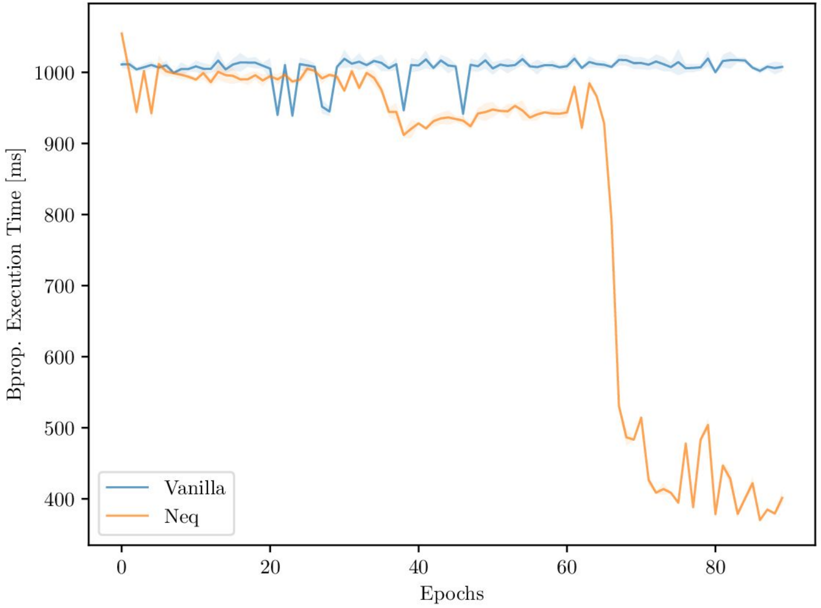
NEq

Experiments

Dataset	Model	Approach	Bprop. FLOPs per iteration	Performance
CIFAR-10	ResNet-32	Baseline	138.94M $\pm$ 0.0M	92.85% $\pm$ 0.23%†
		Stochastic ($p = 0.2$)	112.99M $\pm$ 0.00M (-18.68%)	92.78% $\pm$ 0.19% (-0.07%)†
		Stochastic ($p = 0.5$)	69.75M $\pm$ 0.00M (-49.8%)	91.88% $\pm$ 0.27% (-0.97%)†
		Stochastic*	86.34M $\pm$ 0.00M (-37.85%)	92.23% $\pm$ 0.25% (-0.62%)†
		Neq	84.81M $\pm$ 0.63M (-38.96%)	92.96% $\pm$ 0.21% (+0.11%)†
ImageNet-1K	ResNet-18	Baseline	3.64G $\pm$ 0.0G	69.90% $\pm$ 0.04%†
		Stochastic ($p = 0.2$)	2.94G $\pm$ 0.00G (-19.26%)	69.42% $\pm$ 0.16% (-0.48%)†
		Stochastic ($p = 0.5$)	1.85G $\pm$ 0.00G (-49.11%)	69.18% $\pm$ 0.03% (-0.72%)†
		Stochastic*	2.82G $\pm$ 0.00G (-22.58%)	69.45% $\pm$ 0.06% (-0.45%)†
		Neq	2.80G $\pm$ 0.03G (-23.08%)	69.62% $\pm$ 0.06% (-0.28%)†
	Swin-B	Baseline	30.28G $\pm$ 0.00G	84.71% $\pm$ 0.04% †
		Stochastic ($p = 0.2$)	24.65G $\pm$ 0.00G (-18.6%)	84.54% $\pm$ 0.04% (-0.83%)†
		Stochastic ($p = 0.5$)	16.15G $\pm$ 0.00G (-46.67%)	84.40% $\pm$ 0.02% (-0.31%)†
		Stochastic*	11.02G $\pm$ 0.00G (-63.67%)	84.27% $\pm$ 0.04% (-0.44%)†
		Neq	10.78G $\pm$ 0.02G (-64.39%)	84.35%$\pm$0.02% (-0.36%)†
COCO	DeepLabv3	Baseline	305.06G $\pm$ 0.0G	67.71% $\pm$ 0.02%‡
		Stochastic ($p = 0.2$)	248.69G $\pm$ 0.00G (-18.48%)	67.11% $\pm$ 0.02% (-0.60%)‡
		Stochastic ($p = 0.5$)	163.42G $\pm$ 0.00G (-46.43%)	66.91% $\pm$ 0.04% (-0.80%)‡
		Stochastic*	229.00G $\pm$ 0.00G (-24.93%)	67.02% $\pm$ 0.03% (-0.69%)‡
		Neq	217.29G $\pm$ 0.04G (-28.77%)	67.22% $\pm$ 0.04% (-0.49%)‡

NEq

Faster Backpropagation

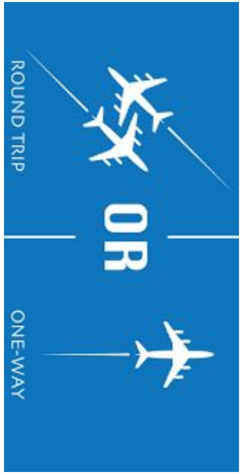


Backpropagation execution time for vanilla and NEq ResNet-18.

We observe a reduction in the wall-clock time of around -17.52%.

Outline

Conclusions



- We shared our recent experiences in **frugal deep learning**
  - **one way**: simplify the model via pruning for faster/lower memory footprint when deploying nets
  - **return**: prune the backward pass to reduce the training cost

- Future research
  - joint approaches including quantization and target device constraints
  - dataset pruning
  - efficient automatic hyper-parameter tuning