

# Pruning neural networks

## Optimizing AI - Session 3



**IMT Atlantique**  
Bretagne-Pays de la Loire  
École Mines-Télécom

## Sessions

- 1 Introduction/Refresher on Deep Learning
- 2 Quantization,
- 3 Pruning,
- 4 Data Augmentation,
- 5 Factorization,
- 6 Distillation,
- 7 Embedded Software and Hardware for DL,
- 8 Final session.

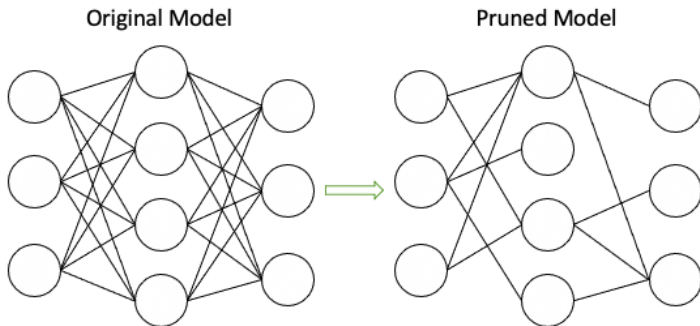
## Sessions

- 1 Introduction/Refresher on Deep Learning
- 2 Quantization,
- 3 **Pruning,**
- 4 Data Augmentation,
- 5 Factorization,
- 6 Distillation,
- 7 Embedded Software and Hardware for DL,
- 8 Final session.

# What is pruning?

## Intuitive principle

Removing parts of the network to reduce its cost (in memory, computation power, etc.).



# What does it involve?

## The three big questions of pruning

- 1 What kind of part should I prune?
- 2 How to tell which parts can be pruned?
- 3 How to prune parts without harming the network?

- ⇒ **Pruning structure:** the kind of part you will remove, how it will affect the network's cost.
- ⇒ **Pruning criterion:** the metric to use to identify parts to prune.
- ⇒ **Pruning method:** the removal strategy and schedule, how to reduce the loss in performance, how to avoid some known pitfalls...

# What does it involve?

## The three big questions of pruning

- 1 What kind of part should I prune?
- 2 How to tell which parts can be pruned?
- 3 How to prune parts without harming the network?

- ⇒ **Pruning structure:** the kind of part you will remove, how it will affect the network's cost.
- ⇒ **Pruning criterion:** the metric to use to identify parts to prune.
- ⇒ **Pruning method:** the removal strategy and schedule, how to reduce the loss in performance, how to avoid some known pitfalls...

# Pruning structure (1/2)

Mainly two kinds of pruning:

- **"Unstructured" (weight) pruning:** removing individual weights
  - More straightforward to implement
  - More fine-grained
  - Allows better accuracy/parameters ratio
  - **But**, very hard to optimize: you not always get any speedup out of it!  
(cf. *Non-Structured DNN Weight Pruning – Is It Beneficial in Any Platform?*, by Ma et. al. 2019)
  - However good as a proof of concept
- **"Structured" (filter) pruning:** removing convolution filters (or neurons)
  - Watch out for dimensional consistency between layers!
  - More coarse-grained (watch out not to prune entire layers by accident!)
  - Produces a smaller network that allows real speedup on any framework
  - Lighter to run because of fewer feature maps

# Pruning structure (1/2)

Mainly two kinds of pruning:

- **"Unstructured" (weight) pruning:** removing individual weights
  - More straightforward to implement
  - More fine-grained
  - Allows better accuracy/parameters ratio
  - **But**, very hard to optimize: you not always get any speedup out of it!  
(cf. *Non-Structured DNN Weight Pruning – Is It Beneficial in Any Platform?*, by Ma et. al. 2019)
  - However good as a proof of concept
- **"Structured" (filter) pruning:** removing convolution filters (or neurons)
  - Watch out for dimensional consistency between layers!
  - More coarse-grained (watch out not to prune entire layers by accident!)
  - Produces a smaller network that allows real speedup on any framework
  - Lighter to run because of fewer feature maps

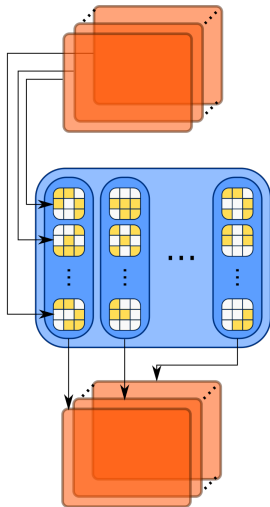


# Pruning structure (1/2)

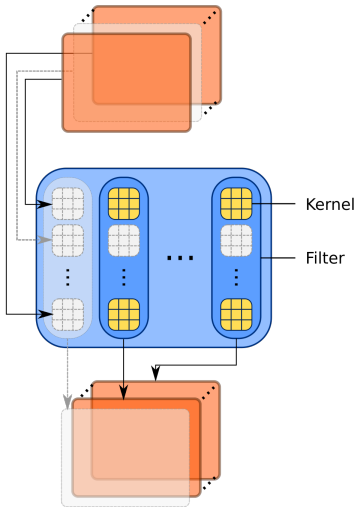
Mainly two kinds of pruning:

- **"Unstructured" (weight) pruning:** removing individual weights
  - More straightforward to implement
  - More fine-grained
  - Allows better accuracy/parameters ratio
  - **But**, very hard to optimize: you not always get any speedup out of it!  
(cf. *Non-Structured DNN Weight Pruning – Is It Beneficial in Any Platform?*, by Ma et. al. 2019)
  - However good as a proof of concept
- **"Structured" (filter) pruning:** removing convolution filters (or neurons)
  - Watch out for dimensional consistency between layers!
  - More coarse-grained (watch out not to prune entire layers by accident!)
  - Produces a smaller network that allows real speedup on any framework
  - Lighter to run because of fewer feature maps

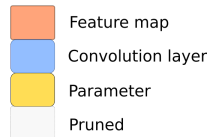
# Pruning structure (1/2)



"Unstructured" : weight pruning



"Structured" : filter pruning



# Pruning structure (2/2)

## How to distribute pruning:

- **Local pruning:** remove the same proportion to each layer
  - A lot simpler to implement
  - Sub-optimal
- **Global pruning:** remove different proportions to each layer to reach a global target
  - More subtle to implement
  - More optimal
  - Warning: can lead to dimensional discrepancies in the case of structured pruning

# Pruning structure (2/2)

How to distribute pruning:

- **Local pruning:** remove the same proportion to each layer
  - A lot simpler to implement
  - Sub-optimal
- **Global pruning:** remove different proportions to each layer to reach a global target
  - More subtle to implement
  - More optimal
  - Warning: can lead to dimensional discrepancies in the case of structured pruning

# Pruning structure (2/2)

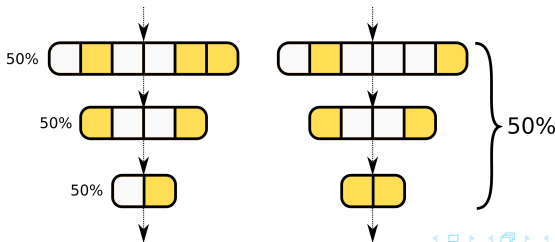
How to distribute pruning:

- **Local pruning:** remove the same proportion to each layer
  - A lot simpler to implement
  - Sub-optimal
- **Global pruning:** remove different proportions to each layer to reach a global target
  - More subtle to implement
  - More optimal
  - Warning: can lead to dimensional discrepancies in the case of structured pruning

# Pruning structure (2/2)

How to distribute pruning:

- **Local pruning:** remove the same proportion to each layer
  - A lot simpler to implement
  - Sub-optimal
- **Global pruning:** remove different proportions to each layer to reach a global target
  - More subtle to implement
  - More optimal
  - Warning: can lead to dimensional discrepancies in the case of structured pruning



# Pruning criteria (1/2)

Two widespread examples for individual weights:

- **Weight magnitude:** prune weights of least  $\mathcal{L}_1$  norm
- **Weight gradient:** do a back-propagation over a minibatch and prune weights whose gradients are of least  $\mathcal{L}_1$  norm

Things get more tricky in the case of structured pruning:

- $\mathcal{L}_1$ ,  $\mathcal{L}_2$  of the filter or its gradient... (cf. *Pruning Filters for Efficient ConvNets*, by Li et. al. 2016)
- Magnitude of the batch-normalization layer's multiplicative parameter (cf. *Learning Efficient Convolutional Networks through Network Slimming*, by Liu et. al. 2017)
- Magnitude of an inserted "gate" (added multiplicative parameter) (cf. *Importance Estimation for Neural Network Pruning*, by Molchanov et. al. 2019)

# Pruning criteria (1/2)

Two widespread examples for individual weights:

- **Weight magnitude:** prune weights of least  $\mathcal{L}_1$  norm
- **Weight gradient:** do a back-propagation over a minibatch and prune weights whose gradients are of least  $\mathcal{L}_1$  norm

Things get more tricky in the case of structured pruning:

- $\mathcal{L}_1, \mathcal{L}_2$  of the filter or its gradient... (cf. *Pruning Filters for Efficient ConvNets*, by Li et. al. 2016)
- Magnitude of the batch-normalization layer's multiplicative parameter (cf. *Learning Efficient Convolutional Networks through Network Slimming*, by Liu et. al. 2017)
- Magnitude of an inserted "gate" (added multiplicative parameter) (cf. *Importance Estimation for Neural Network Pruning*, by Molchanov et. al. 2019)



# Pruning criteria (1/2)

Two widespread examples for individual weights:

- **Weight magnitude:** prune weights of least  $\mathcal{L}_1$  norm
- **Weight gradient:** do a back-propagation over a minibatch and prune weights whose gradients are of least  $\mathcal{L}_1$  norm

Things get more tricky in the case of structured pruning:

- $\mathcal{L}_1, \mathcal{L}_2$  of the filter or its gradient... (cf. *Pruning Filters for Efficient ConvNets*, by Li et. al. 2016)
- Magnitude of the batch-normalization layer's multiplicative parameter (cf. *Learning Efficient Convolutional Networks through Network Slimming*, by Liu et. al. 2017)
- Magnitude of an inserted "gate" (added multiplicative parameter) (cf. *Importance Estimation for Neural Network Pruning*, by Molchanov et. al. 2019)

# Pruning criteria (1/2)

Two widespread examples for individual weights:

- **Weight magnitude:** prune weights of least  $\mathcal{L}_1$  norm
- **Weight gradient:** do a back-propagation over a minibatch and prune weights whose gradients are of least  $\mathcal{L}_1$  norm

Things get more tricky in the case of structured pruning:

- $\mathcal{L}_1, \mathcal{L}_2$  of the filter or its gradient... (cf. *Pruning Filters for Efficient ConvNets*, by Li et. al. 2016)
- Magnitude of the batch-normalization layer's multiplicative parameter (cf. *Learning Efficient Convolutional Networks through Network Slimming*, by Liu et. al. 2017)
- Magnitude of an inserted "gate" (added multiplicative parameter) (cf. *Importance Estimation for Neural Network Pruning*, by Molchanov et. al. 2019)

# Pruning criteria (1/2)

Two widespread examples for individual weights:

- **Weight magnitude:** prune weights of least  $\mathcal{L}_1$  norm
- **Weight gradient:** do a back-propagation over a minibatch and prune weights whose gradients are of least  $\mathcal{L}_1$  norm

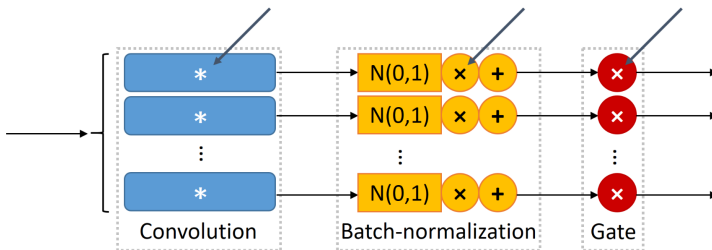
Things get more tricky in the case of structured pruning:

- $\mathcal{L}_1, \mathcal{L}_2$  of the filter or its gradient... (cf. *Pruning Filters for Efficient ConvNets*, by Li et. al. 2016)
- Magnitude of the batch-normalization layer's multiplicative parameter (cf. *Learning Efficient Convolutional Networks through Network Slimming*, by Liu et. al. 2017)
- Magnitude of an inserted "gate" (added multiplicative parameter) (cf. *Importance Estimation for Neural Network Pruning*, by Molchanov et. al. 2019)

# Pruning criteria (1/2)

Things get more tricky in the case of structured pruning:

- $\mathcal{L}_1$ ,  $\mathcal{L}_2$  of the filter or its gradient... (cf. *Pruning Filters for Efficient ConvNets*, by Li et. al. 2016)
- Magnitude of the batch-normalization layer's multiplicative parameter (cf. *Learning Efficient Convolutional Networks through Network Slimming*, by Liu et. al. 2017)
- Magnitude of an inserted "gate" (added multiplicative parameter) (cf. *Importance Estimation for Neural Network Pruning*, by Molchanov et. al. 2019)



# Pruning criteria (2/2)

## Remarks:

- 1 Criteria reported previously are among the simplest: the literature has been very prolific on the topic of pruning criteria and include many complex ones. For example:
  - Filter identification through reinforcement learning (cf. *AMC: AutoML for Model Compression and Acceleration on Mobile Devices*, by He et. al. 2018)
  - Optimization through variational inference (cf. *Variational Dropout Sparsifies Deep Neural Networks*, by Molchanov et. al. 2017)
  - Computation of the network's second derivative (Hessian)... (cf. *Optimal Brain Damage*, by Le Cun et. al. 1989)
- 2 **Warning**, in the case of global pruning:
  - Some criteria may be unbalanced: for example, magnitude-based criteria tend to prune mostly the last layers.
  - There are risks to prune entire layers by accident, a.k.a. "**layer collapse**". (cf. *Pruning neural networks without any data by iteratively conserving synaptic flow*, by Tanaka et. al. 2020)

# Pruning criteria (2/2)

## Remarks:

- 1 Criteria reported previously are among the simplest: the literature has been very prolific on the topic of pruning criteria and include many complex ones. For example:
  - Filter identification through reinforcement learning (cf. *AMC: AutoML for Model Compression and Acceleration on Mobile Devices*, by He et. al. 2018)
  - Optimization through variational inference (cf. *Variational Dropout Sparsifies Deep Neural Networks*, by Molchanov et. al. 2017)
  - Computation of the network's second derivative (Hessian)... (cf. *Optimal Brain Damage*, by Le Cun et. al. 1989)
- 2 **Warning**, in the case of global pruning:
  - Some criteria may be unbalanced: for example, magnitude-based criteria tend to prune mostly the last layers.
  - There are risks to prune entire layers by accident, a.k.a. "**layer collapse**". (cf. *Pruning neural networks without any data by iteratively conserving synaptic flow*, by Tanaka et. al. 2020)

# Pruning criteria (2/2)

## Remarks:

- 1 Criteria reported previously are among the simplest: the literature has been very prolific on the topic of pruning criteria and include many complex ones. For example:
  - Filter identification through reinforcement learning (cf. *AMC: AutoML for Model Compression and Acceleration on Mobile Devices*, by He et. al. 2018)
  - Optimization through variational inference (cf. *Variational Dropout Sparsifies Deep Neural Networks*, by Molchanov et. al. 2017)
  - Computation of the network's second derivative (Hessian)... (cf. *Optimal Brain Damage*, by Le Cun et. al. 1989)
- 2 **Warning**, in the case of global pruning:
  - Some criteria may be unbalanced: for example, magnitude-based criteria tend to prune mostly the last layers.
  - There are risks to prune entire layers by accident, a.k.a. **"layer collapse"**. (cf. *Pruning neural networks without any data by iteratively conserving synaptic flow*, by Tanaka et. al. 2020)

# Pruning methods

- Many existing and very different methods in the literature...
- Two aspects that encompasses most simple ones:
  - 1 Removal strategy
  - 2 Training schedule



# Pruning methods

- Many existing and very different methods in the literature...
- Two aspects that encompasses most simple ones:
  - 1 **Removal strategy**
  - 2 **Training schedule**

- Many existing and very different methods in the literature...
- Two aspects that encompasses most simple ones:

## 1 Removal strategy

- Set weights to prune to 0 definitely
- Same but iterative with a growing pruning rate (cf. *Learning both Weights and Connections for Efficient Neural Networks*, by Han et. al. 2015)
- Progressively all throughout pruning, until the target is reached (cf. *To prune, or not to prune: exploring the efficacy of pruning for model compression*, by Zhu et. al. 2017)
- At each epoch, set targeted weights to 0 but let them train again until the end, a.k.a. "Soft pruning" (cf. *Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks*, by He et. al. 2018)

## 2 Training schedule

# Pruning methods

- Many existing and very different methods in the literature...
- Two aspects that encompasses most simple ones:

- 1 **Removal strategy**

- 2 **Training schedule**

- Train, prune and fine-tune once (or prune while training) (cf. *Learning Efficient Convolutional Networks through Network Slimming*, by Liu et. al. 2017)
- Fine-tune after each pruning step (cf. *Learning both Weights and Connections for Efficient Neural Networks*, by Han et. al. 2015)
- Train, prune, and wholly retrain, a.k.a. LR-Rewinding (cf. *Comparing Rewinding and Fine-tuning in Neural Network Pruning*, by Renda et. al. 2020)

# Pruning methods

- Many existing and very different methods in the literature...
- Two aspects that encompasses most simple ones:
  - 1 **Removal strategy**
  - 2 **Training schedule**

## Remark 1

For the sake of convenience, many methods/paper count as pruned weights that are simply set to 0. Even though it does not produce any speedup, it allows measuring a parameters/performance ratio without worrying about dimensional discrepancies or efficiency issues.

# Pruning methods

- Many existing and very different methods in the literature...
- Two aspects that encompasses most simple ones:
  - 1 **Removal strategy**
  - 2 **Training schedule**

## Remark 1

For the sake of convenience, many methods/paper count as pruned weights that are simply set to 0. Even though it does not produce any speedup, it allows measuring a parameters/performance ratio without worrying about dimensional discrepancies or efficiency issues.

## Remark 2

Iterative or progressive methods are more robust to layer-collapse, because values of remaining weights values may be updated so that they are less likely to be targeted afterward.

# Some unstructured examples

- *Learning both Weights and Connections for Efficient Neural Networks*, by Han et. al. 2015
  - Structure: global individual weights
  - Criterion: weights magnitude
  - Method: train, then iterate between pruning and fine-tuning
- *To prune, or not to prune: exploring the efficacy of pruning for model compression*, by Zhu et. al. 2017
- *Comparing Rewinding and Fine-tuning in Neural Network Pruning*, by Renda et. al. 2020

# Some unstructured examples

- *Learning both Weights and Connections for Efficient Neural Networks*, by Han et. al. 2015
- *To prune, or not to prune: exploring the efficacy of pruning for model compression*, by Zhu et. al. 2017
  - Structure: local individual weights
  - Criterion: weights magnitude
  - Method: train while pruning progressively the smallest weights, no fine-tuning
- *Comparing Rewinding and Fine-tuning in Neural Network Pruning*, by Renda et. al. 2020

# Some unstructured examples

- *Learning both Weights and Connections for Efficient Neural Networks*, by Han et. al. 2015
- *To prune, or not to prune: exploring the efficacy of pruning for model compression*, by Zhu et. al. 2017
- *Comparing Rewinding and Fine-tuning in Neural Network Pruning*, by Renda et. al. 2020
  - Structure: global individual weights
  - Criterion: weights magnitude
  - Method: train, prune the lowest 20%, re-train and repeat until pruned enough
  - Remark: the principle of retraining, with its learning-rate schedule, instead of fine-tuning with the lowest learning-rate is the heart of *Learning Rate Rewinding*.



# Some structured examples

- *Pruning Filters for Efficient ConvNets*, by Li et. al. 2016
  - Structure: local convolution filters and corresponding kernels in the following layer (with varying rates)
  - Criterion:  $\mathcal{L}_1$  norm of filters
  - Method: train, then iterate between pruning and fine-tuning
- *Learning Efficient Convolutional Networks through Network Slimming*, by Liu et. al. 2017
- *Importance Estimation for Neural Network Pruning*, by Molchanov et. al. 2019

# Some structured examples

- *Pruning Filters for Efficient ConvNets*, by Li et. al. 2016
- *Learning Efficient Convolutional Networks through Network Slimming*, by Liu et. al. 2017
  - Structure: global convolution filters
  - Criterion: magnitude of multiplicative parameter in batch-normalization layers
  - Method: train, prune and fine-tune once
  - Remark: applies a smooth- $\mathcal{L}_1$  penalty on multiplicative parameters in batch-normalization layers during training
- *Importance Estimation for Neural Network Pruning*, by Molchanov et. al. 2019

# Some structured examples

- *Pruning Filters for Efficient ConvNets*, by Li et. al. 2016
- *Learning Efficient Convolutional Networks through Network Slimming*, by Liu et. al. 2017
- *Importance Estimation for Neural Network Pruning*, by Molchanov et. al. 2019
  - Structure: global gates (or global individual weights)
  - Criterion: gradient magnitude
  - Method: train, then iterate between pruning and fine-tuning

# Some structured examples

- *Pruning Filters for Efficient ConvNets*, by Li et. al. 2016
- *Learning Efficient Convolutional Networks through Network Slimming*, by Liu et. al. 2017
- *Importance Estimation for Neural Network Pruning*, by Molchanov et. al. 2019

## Advice

Refer to the original papers (available online) for more details on methods, implementation or eventual variants.

# Some structured examples

- *Pruning Filters for Efficient ConvNets*, by Li et. al. 2016
- *Learning Efficient Convolutional Networks through Network Slimming*, by Liu et. al. 2017
- *Importance Estimation for Neural Network Pruning*, by Molchanov et. al. 2019

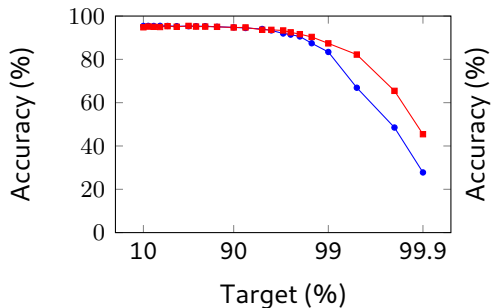
## Advice

Refer to the original papers (available online) for more details on methods, implementation or eventual variants.

To go further:

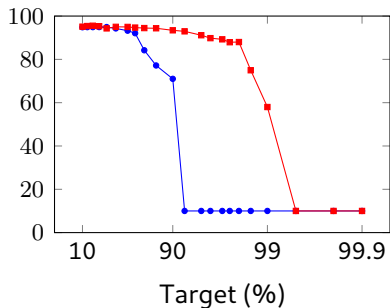
- *Neural Network Pruning 101* on [towardsdatascience.com](https://towardsdatascience.com).

# Some results



(a) Unstructured pruning

—●— Han et. al.  
—■— Han et. al. + LR-Rewinding



(b) Structured pruning

—●— Liu et. al.  
—■— Liu et. al. + LR-Rewinding

Figure: Pruning rate/Accuracy tradeoff, ResNet-20 on CIFAR-10

# Lab Session and Project

## Lab Session

- Implement one of the pruning methods from this course
- Apply it on MiniCIFAR

## Presentation at next session

Present your current explorations on MiniCIFAR, CIFAR10 and / or CIFAR100 using the methods seen so far!