Extracting Effective Subnetworks with Gumbel-Softmax

Robin **Dupont** — Sorbonne Université & Netatmo
Mohammed Amine **Alaoui** — Netatmo
Hichem **Sahbi** — Sorbonne Université
Alice **Lebois** — Netatmo

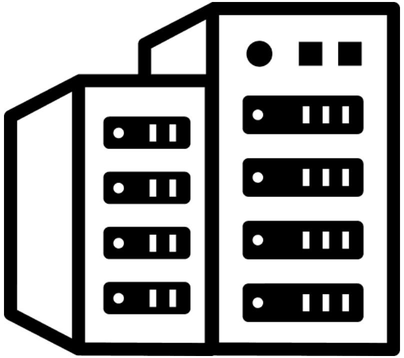
Table of Content

- Why Lightweight Neural Networks ?
- Lightweight Networks Design via Pruning
- Our Method
- Results
- Sum Up

Why Lightweight Neural Networks ?

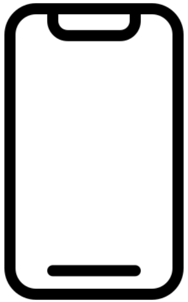
Why Lightweight Neural Networks ?

Server



- Powerful 💪
- Handle full size models 🧠

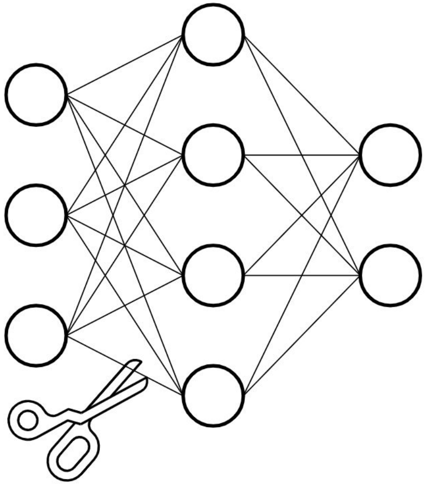
Embedded device



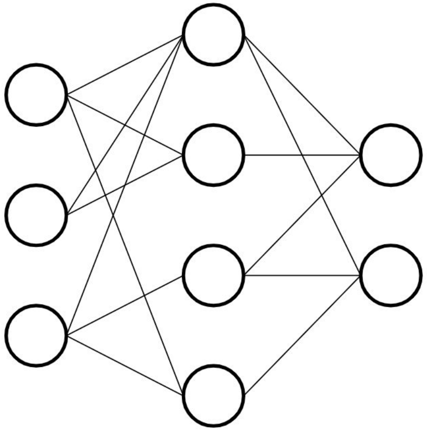
- Limited resources 🤖
- Require lightweight models 🪶

Lightweight Networks Design via Pruning

Lightweight Networks Design via Pruning



Before pruning

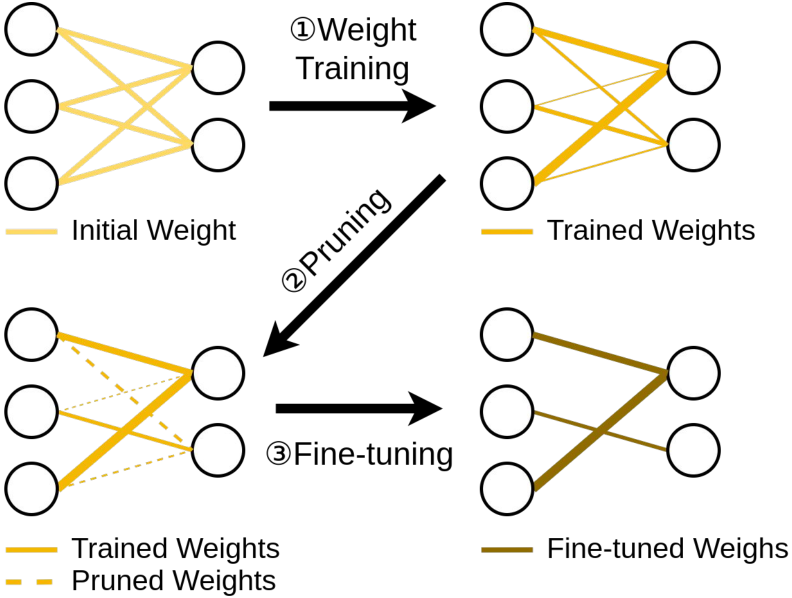


After pruning

- **Pruning** removes **weights**
- **Unstructured** pruning
- Yields **sparse** and **lightweight** models

Lightweight Networks Design via Pruning

Standard pruning pipelines

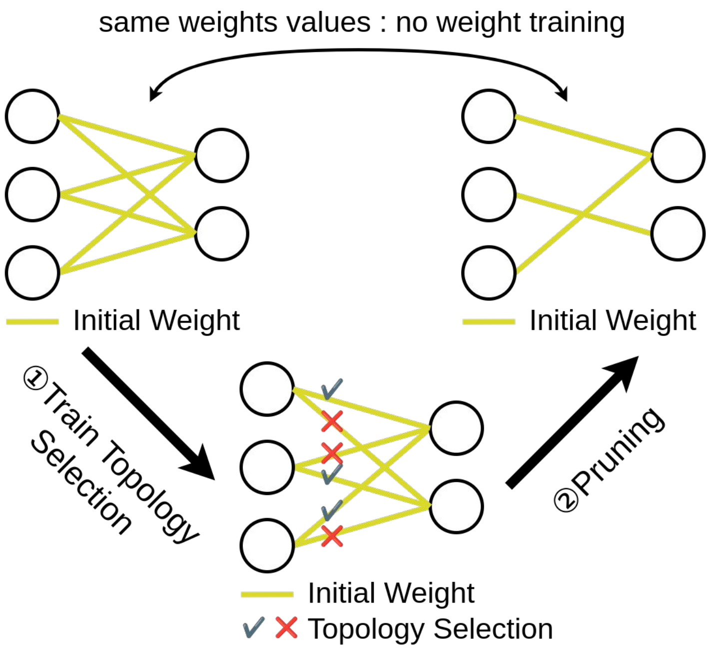


- 3 steps procedure
  train - prune - fine-tune
- Pruning **criterion** depends on the **method**
- **Fine-tuning** needed

Our Method

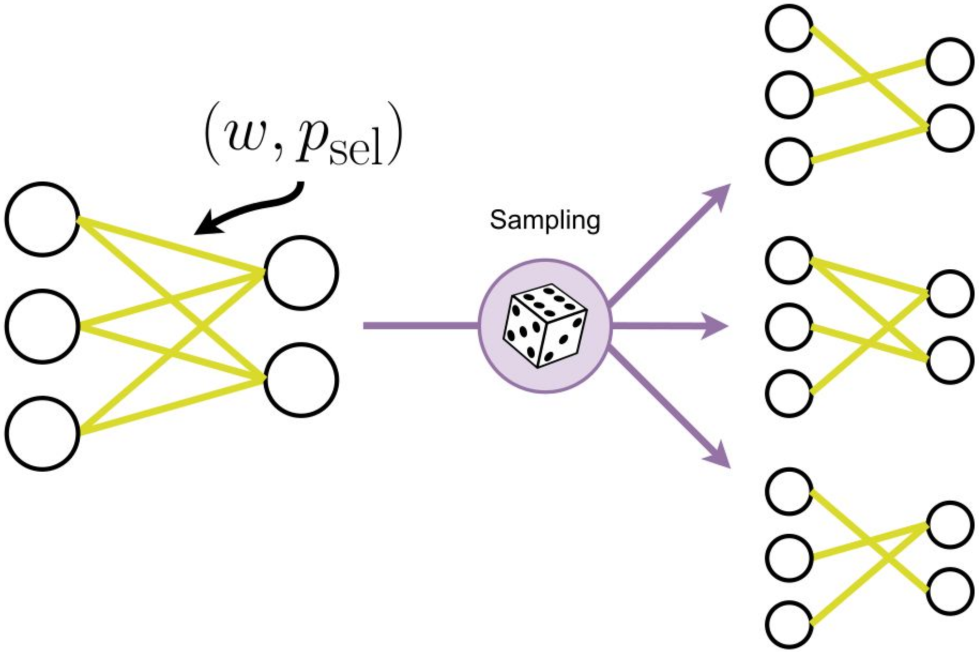
Our Method

Our pruning pipeline



- **No weight training** ⚠️
- **Topology selection** only

Our Method



Our Method

Layer equation :

$$\mathbf{z}_\ell = g_\ell((\boldsymbol{m}_\ell \odot \boldsymbol{w}_\ell) \otimes \mathbf{z}_{\ell-1})$$

binary masks tensor

weights tensor

coefficient follows bernoulli distribution :

$$m \sim \mathcal{B}(p_{\text{sel}})$$

Probability for a **weight** of **being selected**

Sampling is **not differentiable** ⚠️

Our Method

Probability reparametrization : $$p_{\mathrm{sel}} = \sigma(\hat{m})$$

Learnt variable

Sigmoid ensures $0 \leq p_{\mathrm{sel}} \leq 1$

Our Method

Probability reparametrization : $$p_{\text{sel}} = \sigma(\hat{m})$$

Naive Gumbel-Softmax formulation :

$$m = \text{STGS}\left(\begin{bmatrix} \log(\sigma(\hat{m})) \\ \log(1-\sigma(\hat{m})) \end{bmatrix}\right)$$

Combination of log and exponential functions :

❌ Numerical **instabilities**
❌ Computationally **intensive**

Our Method - ASLP

Probability reparametrization : $$p_{\text{sel}} = \sigma(\hat{m})$$

Our formulation **Arbitrarily Shifted Log Parameterization** (ASLP)

$$m = \mathrm{STGS}\left(\begin{bmatrix} \hat{m} \\ 0 \end{bmatrix}\right)$$

✅ Numerically **stable**
✅ **Less** computationally intensive

Our Method - ASLP

Our formulation :

$$m = \mathrm{STGS}\left(\begin{bmatrix}\hat{m}\\0\end{bmatrix}\right)$$

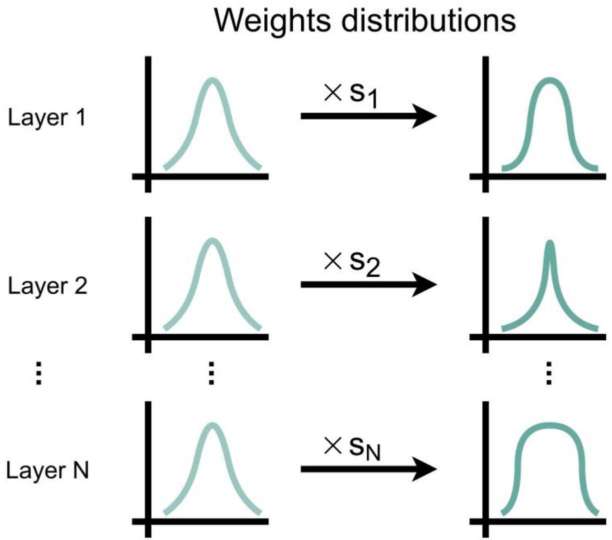
Arbitrary unknown constant that shifts log probabilities

$$\begin{bmatrix}\hat{m}\\0\end{bmatrix} = \begin{bmatrix}\log(\sigma(\hat{m})) + c\\ \log(1-\sigma(\hat{m})) + c\end{bmatrix} \implies p_{\mathrm{sel}} = \sigma(\hat{m})$$

Same reparametrization

💡 Adding a constant **does not change** the result of STGS

Our Method - Smart rescale

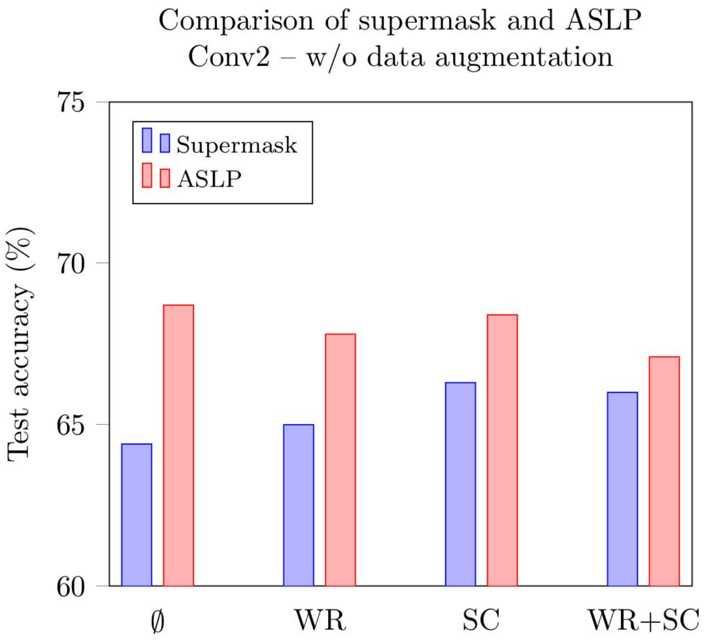


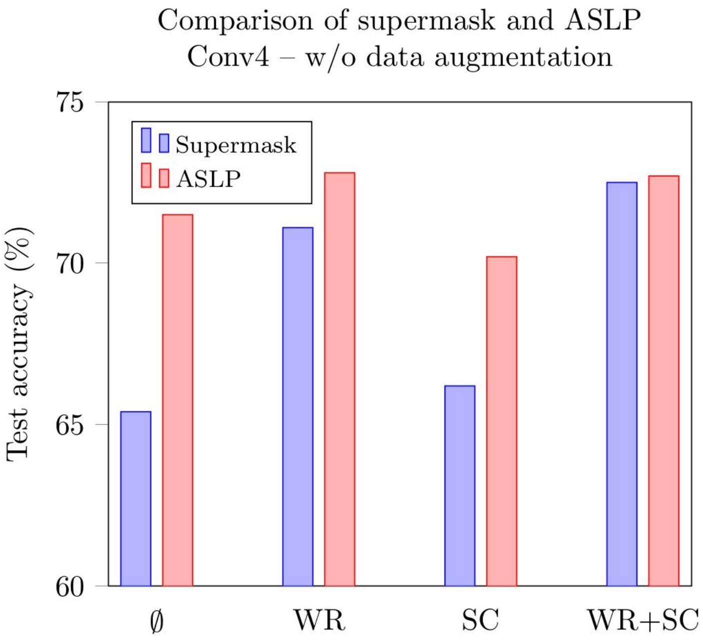
- Scaling learnt per layer
- **Mitigates** the change of **variance** due to **pruning**
- Improves **performances**
- **Reduces** number of epochs needed for **convergence**

Results

Results

CIFAR 10





WR = Weight Rescale, SC = Signed Constant

H. Zhou, et al. “Deconstructing lottery tickets: Zeros, signs, and the supermask,” in NeurIPS, 2019

Results

CIFAR 100

	Conv2	Conv4	Conv6
EP	40.9	51.1	**53.2**
ASLP	**43.4**	**51.7**	52.8

Table 1: Edge Popup and ASLP on CIFAR100

Results for WR+SC

V. Ramanujan, et al. "What's hidden in a randomly weighted neural network?," in CVPR, 2020

Sum Up

Sum Up

- Lightweight networks are **useful** for **embedded devices**
- Our method prunes **untrained** networks - **topology selection** only
- We use **Gumbel Softmax** for differentiable sampling
- **ASLP** : simpler formulation, **less** computationally intensive, numerically **stable**
- **Smart Rescale** : improves performances, reduces number of epochs
- Our method yields **lightweight** networks **without weight training.**

Thanks you!

Robin **Dupont** — Sorbonne Université & Netatmo
Mohammed Amine **Alaoui** — Netatmo
Hichem **Sahbi** — Sorbonne Université
Alice **Lebois** — Netatmo