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

Final Project

Group 9
Zijie NING
Guoxiong SUN

21/03/2022

CONTENTS

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

CHAPTER 1
Results

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

MicroNet Challenge

Hosted at NeurIPS 2019

On CIFAR-10

Flops: 22675830.0, Params: 47837.0

Score flops: 0.0271

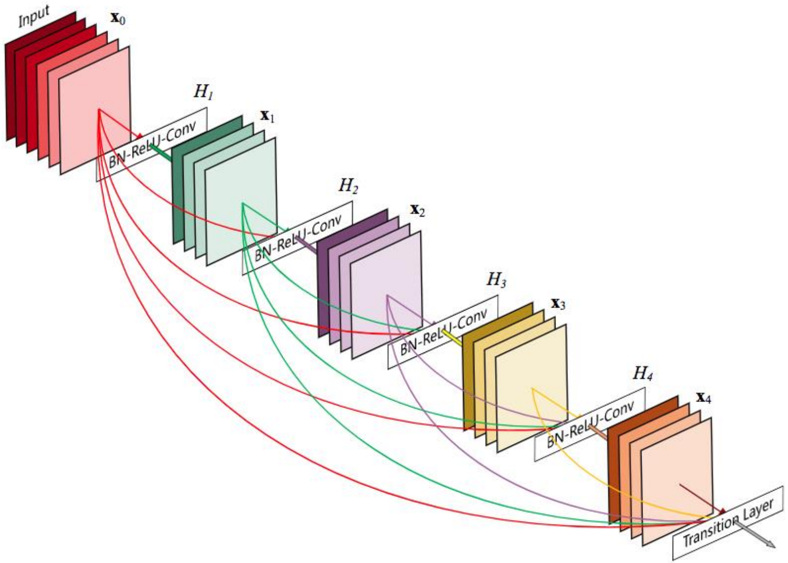
Score Params: 0.0086

Final score: 0.0357

CHAPTER 2

Method

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



DenseNet:

```
def densenet_cifar():
        return DenseNet(Bottleneck, [6,12,24,16], growth_rate=12)
```

- Learning rate

0.1 for 100 epochs, 0.01 for 100 epochs, then 0.001 for 100 epochs

- Weight decay

5e-4? 1e-4? …?

- Batch size

As big as possible

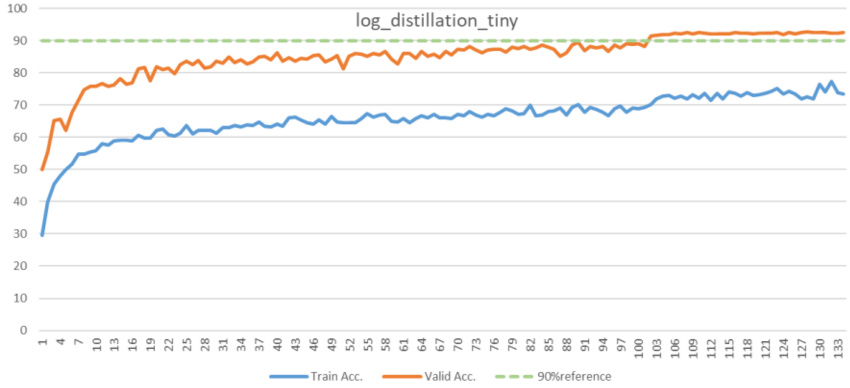
- Scheduler

CosineAnnealingLR()

- Optimizer

SGD

“SGD outperforms all other methods in most cases.”



https://medium.com/geekculture/a-2021-guide-to-improving-cnns-optimizers-adam-vs-sgd-495848ac6008#:~:text=One%20interesting%20and%20dominant%20argument,results%20in%20improved%20final%20performance

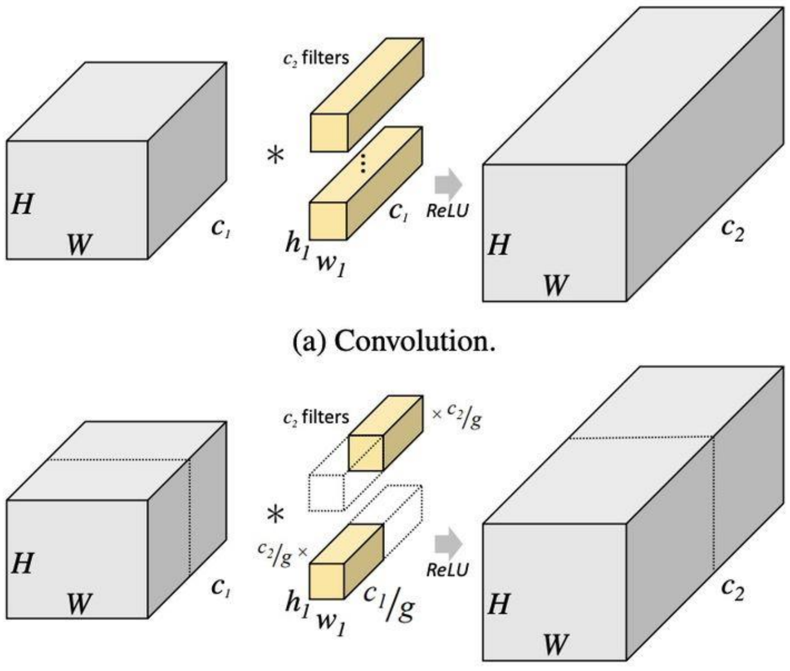
	ResNet-50	Mixup [47]	Cutout [3]	CutMix
Image				
Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4
ImageNet Cls (%)	76.3 (+0.0)	77.4 (+1.1)	77.1 (+0.8)	**78.6** **(+2.3)**
ImageNet Loc (%)	46.3 (+0.0)	45.8 (-0.5)	46.7 (+0.4)	**47.3** **(+1.0)**
Pascal VOC Det (mAP)	75.6 (+0.0)	73.9 (-1.7)	75.1 (-0.5)	**76.7** **(+1.1)**

Table 1: Overview of the results of Mixup, Cutout, and our CutMix on ImageNet classification, ImageNet localization, and Pascal VOC 07 detection (transfer learning with SSD [23] finetuning) tasks. Note that CutMix significantly improves the performance on various tasks.

CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features

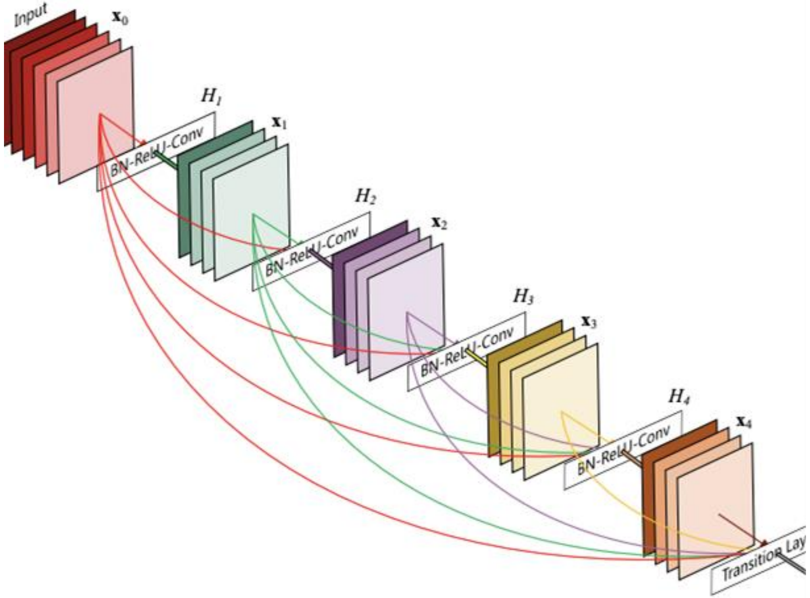
P(cutmix)=1
Acc. ↑~3%

https://arxiv.org/abs/1905.04899



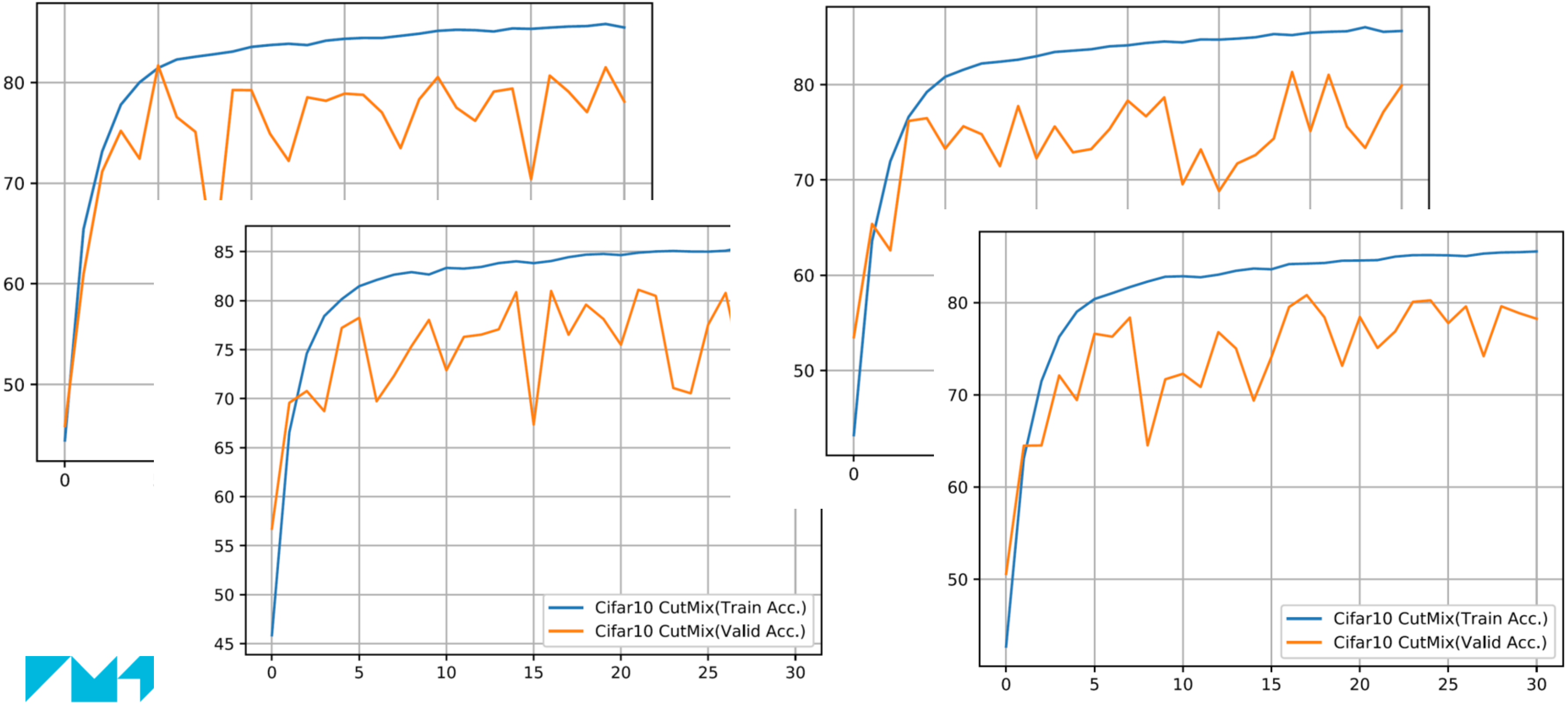
Methods for reducing Nops :
Group Convolution

- Conv2d -> Groups

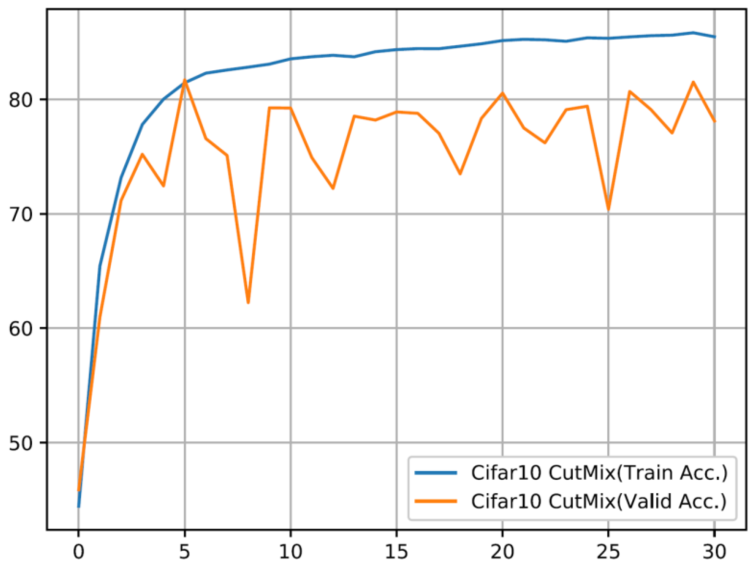


Methods for reducing Nops :
Group Convolution

- Conv2d
- Bottleneck 2
- Transition 1
- DenseNet 1

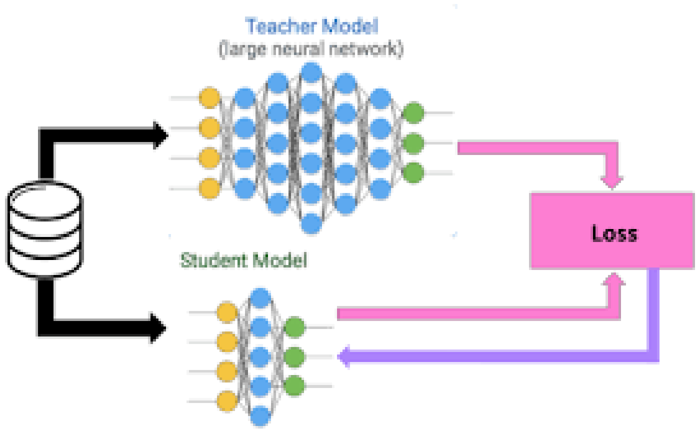


Cifar10 CutMix(Train Acc.)
Cifar10 CutMix(Valid Acc.)
Cifar10 CutMix(Train Acc.)
Cifar10 CutMix(Valid Acc.)



Sensitivity of different places

- The second place of Bottleneck is the least sensitive
- Direction of grouped convolution

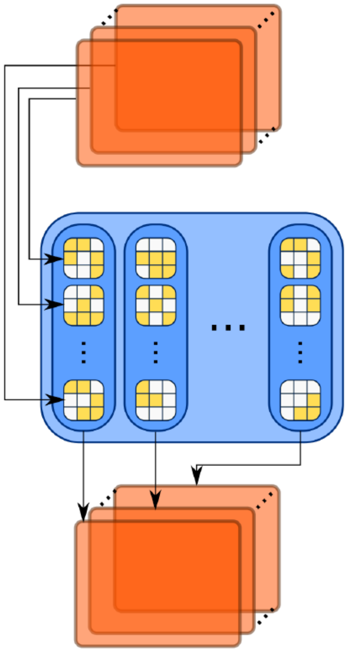


"Distilling the Knowledge in a Neural Network"
— Geoffrey Hinton, Oriol Vinyals, Jeff Dean

Temperature?

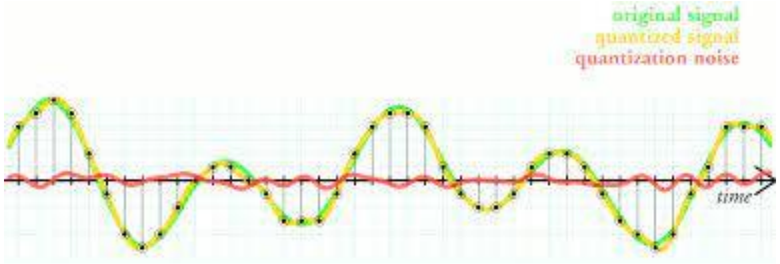
Even better teacher?

https://arxiv.org/abs/1503.02531



"Unstructured" : weight pruning

- Set weights to prune to 0 definitely
- Same but iterative with a growing pruning rate (cf. Learning both Weights and Connections for Ecient Neural Networks, by Han et. al. 2015)
- 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)



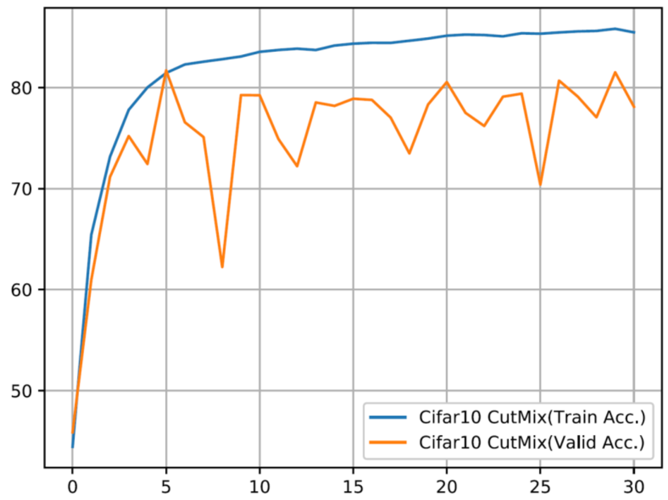
- model.half() (16bit)
- FX graph mode quantization (8bit)
- BinaryConnect (1bit)

https://arxiv.org/abs/1511.00363
https://pytorch.org/tutorials/prototype/fx_graph_mode_ptq_static.html

CHAPITRE 2 : Method

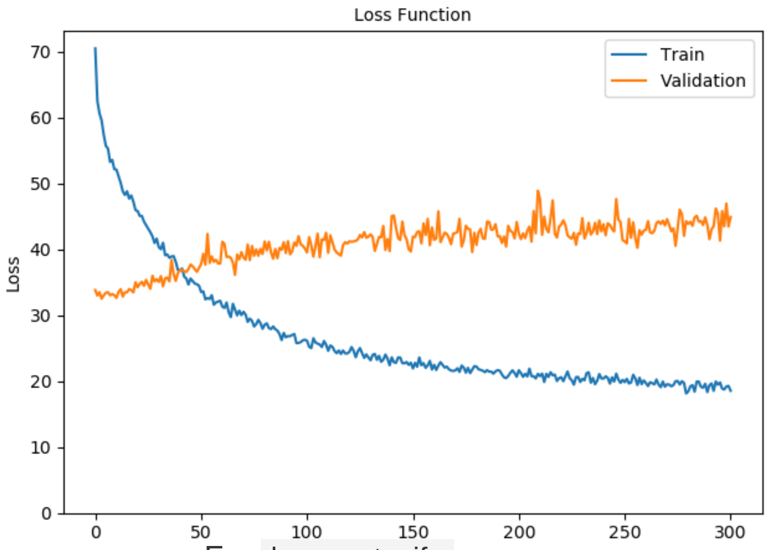
2.8 Logger

Train Acc.	Valid Acc.
26.396000	44.420000
37.164000	52.590000
42.034000	60.260000
46.234000	63.410000
46.142000	64.840000
50.502000	67.140000
51.582000	68.970000
52.082000	70.070000
51.788000	67.730000
53.536000	70.610000
54.864000	70.620000
54.166000	74.250000
55.266000	73.090000
55.528000	73.570000
55.932000	75.970000
56.040000	77.520000
56.658000	78.180000
56.490000	77.040000
58.262000	75.020000
55.886000	78.990000
58.236000	77.390000
57.948000	77.780000
59.928000	78.290000
60.588000	78.840000
58.588000	79.930000
60.858000	79.400000
59.268000	80.630000
59.918000	78.980000
60.922000	80.310000
59.520000	79.170000



CHAPTER 3
Training Process

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



For densenet_cifar
Flops: 192701552.0, Params: 500309.0
Score flops: 0.2309
Score Params: 0.0895
Final score: 0.3205

Start with densenet_cifar

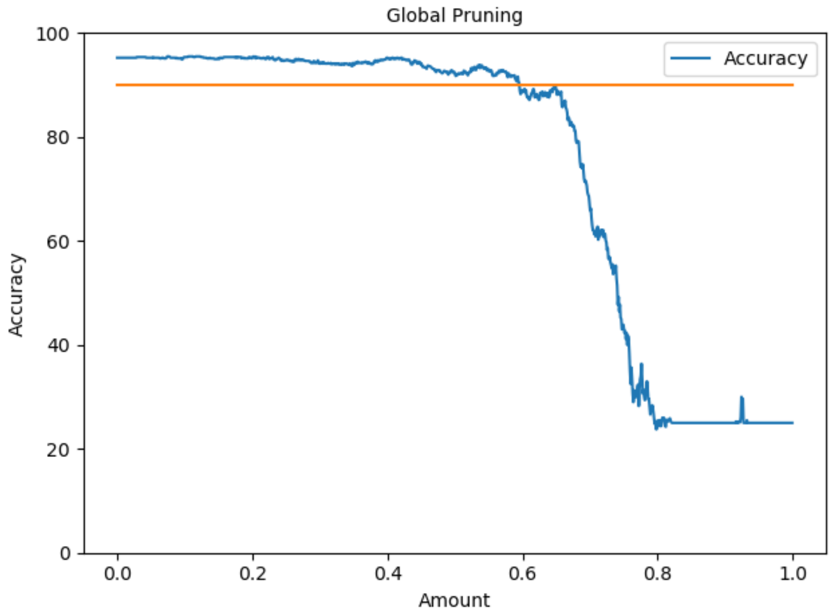
```
def DenseNet121():
    return DenseNet(Bottleneck, [6,12,24,16], growth_rate=32)

def DenseNet169():
    return DenseNet(Bottleneck, [6,12,32,32], growth_rate=32)

def DenseNet201():
    return DenseNet(Bottleneck, [6,12,48,32], growth_rate=32)

def DenseNet161():
    return DenseNet(Bottleneck, [6,12,36,24], growth_rate=48)

def densenet_cifar():
    return DenseNet(Bottleneck, [6,12,24,16], growth_rate=12)
```



Change growth rate?

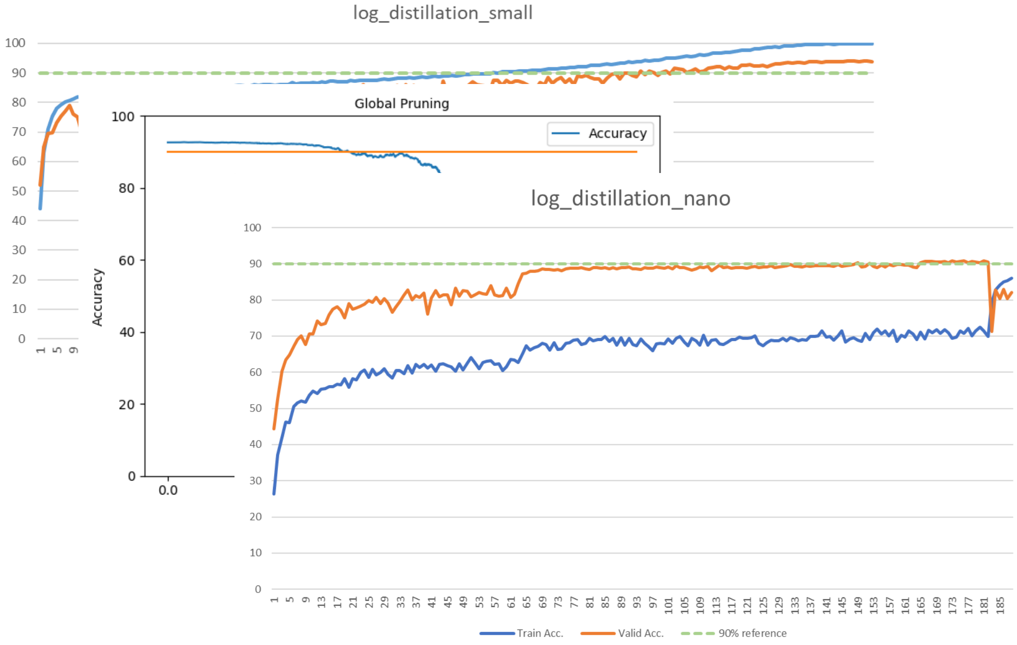
```python
def densenet_cifar():
    return DenseNet(Bottleneck, [6,12,24,16], growth_rate=12)

def densenet_small():
    return DenseNet(Bottleneck, [6,12,24,16], growth_rate=9)

def densenet_tiny():
    return DenseNet(Bottleneck, [6,12,24,16], growth_rate=6)

def densenet_nano():
    return DenseNet(Bottleneck, [6,12,24,16], growth_rate=4)
```

Thanks to Aziz & Alexandre…



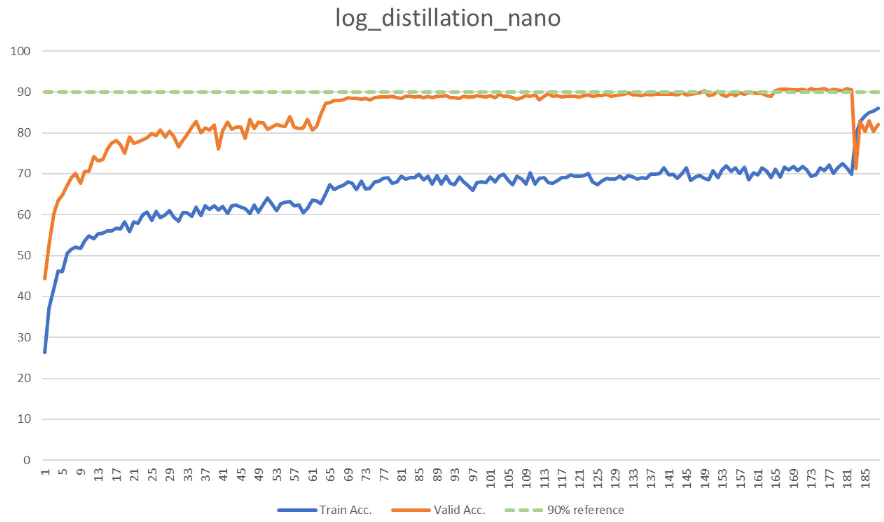
We got… nano!

90.9%

growth rate = 4

Params: 59573.0

Final score: 0.0378



Transformation, Batch normalization

Lr, weight_decay, temperature

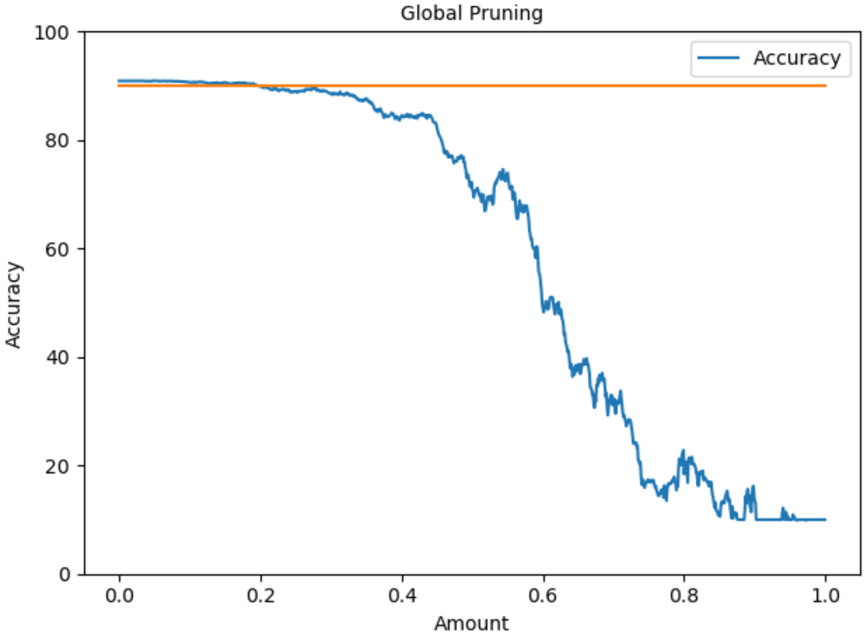
Distillation

Pruning-retrain

Groups convolution

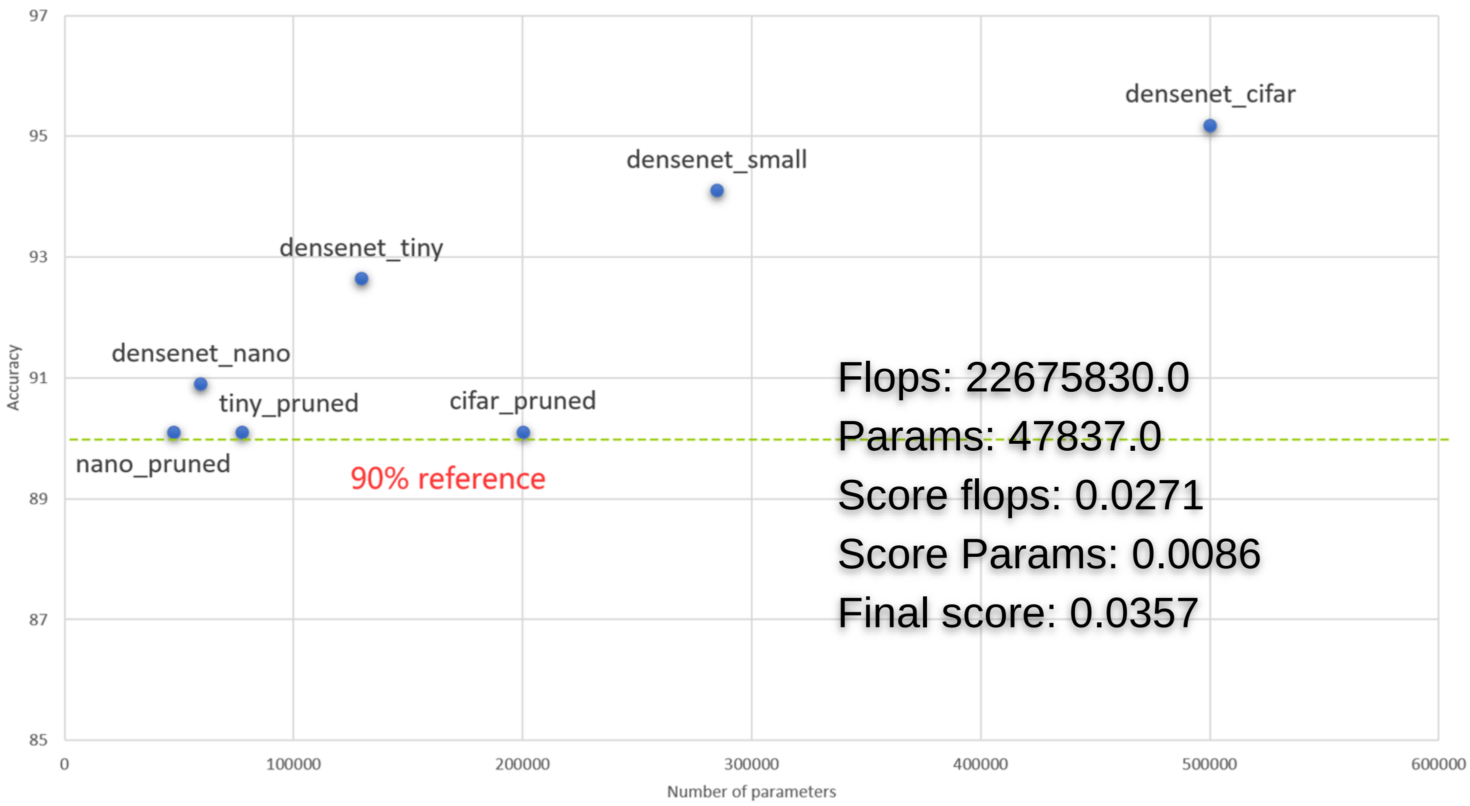
Cutmix-retrain

Quantization(half, 8bit, BC)



A simple L1norm unstructured pruning

Pruning rate = 19.7%



Flops: 22675830.0

Params: 47837.0

Score flops: 0.0271

Score Params: 0.0086

Final score: 0.0357

Thanks for listening

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