NAVYA PROJECT

BERTIN Laurina / DHILLON Prashant / MANSUY Adrien /
MANSUY Alexandra / SEIFEDDINE Wassim / SHARMA Abhishek

Table of contents

- Introduction
- Metrics
- Influence of quantization
- Influence of pruning
- Summary table
- Conclusion

Introduction

Introduction

- Difficulties due to the different operating systems
  - benchmarking → affects performances
  - problems with the file transfer
  - Linux → better results than MacOS or Windows
- Sub-teams

Metrics

ESIEE PARIS

Metrics

mAP: average precision across all classes

MACs: multiplication and accumulation

Average sparsity: percentage of weights replaced by 0

Memory usage: (MB)

Model size: (MB)

Mean Inference time: (s)

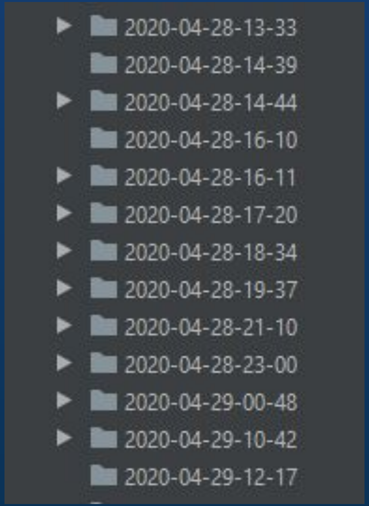
Std inference time: (s)

FPS (for video)

ESIEE PARIS

Main challenges and **solutions**

Unusable for different models (implemented where the model was created) → function taking the model as parameter + save in CSV



Influence of quantization

ESIEE PARIS

Apex experiments

Mixed Precision Training: Change most operations from F32⟶F16

- Operations that benefit from high precision remain in F32
- AMP need fine tuning for unorthodox network structures
- AMP is added on the model and optimizer
- A scaling factor is added on the backward pass to prevent underflowing gradients

Code implemented within Train_SSD variant but not debugged as of now.

- recommend the usage of linux over windows.

ESIEE PARIS

INT 8 post training static quantization

Reduce the number of bits that represent a number : F32 $\longrightarrow$ INT8

Different methods of quantization: **Linear and range-based**

Dynamic	Convert weights and activations on the run
Post training	**converting weights and activation values to floats - and then back to ints - between every operation**
Quantization-aware	fake quantization, but all the adjustments done while training

ESIEE PARIS

Main challenges and **solutions**

Google pretrained model not quantizable ⟶ architecture change (pytorch definition), fusing layers ⟶ Re-train

Type	Role
QuantStub	Quantizing input
DeQuantStub	De-quantizing output
ReLu	Bottleneck
Identify	Merging the conv2d, batchnorm, Relu
Batchnorm	Forces activations to be sd =1 and mean=0
Conv2d	Performs 2D convolution
Inverted Residual	Passing gradient through bottlenecks

ESIEE PARIS

Main challenges and **solutions**

Performance after training (not quantized) : mAP = 0.251 =**25.1%**

we focused on making quantization work rather than improving the performance

Performance after successful quantization : mAP = 0.1%

Calibration: 1-5% validation dataset used to collect statistics to quantize better

After calibration: mAP= 0.252= **25.2%**

ESIEE PARIS

Metrics un quantized model

Not quantized	MAP	Aeroplane	Bicycle	Bird	Boat	Bottle	Bus
INT8 static post training	25.1%	37.4%	33.7%	10.3%	8.3%	0.3%	44.6%
Not quantized	**Car**	**Cat**	**Chair**	**Cow**	**Dining table**	**Dog**	**Horse**
INT8 static post training	42.3%	33.3%	7%	11.2%	24.7%	26.7%	41.8%
Not quantized	**Motorbike**	**Person**	**Potted plant**	**Sheep**	**Sofa**	**Train**	**TV monitor**
INT8 static post training	40.8%	39.1%	0.4%	18.0%	18.6%	42.9%	19.5%
Not quantized		**Sparsity**	**MACs**	**Mem usage**	**Model size**	**Mean inference time**	**Std inference time**
INT8 static post training		0	0.65	367 MB	13.5 MB	0.2s	0.02s

ESIEE PARIS

Metrics Quantized Model

Quantization	MAP	Aeroplane	Bicycle	Bird	Boat	Bottle	Bus
INT8 static post training	25.2%	43.7%	28.3%	14.6%	5.8%	0.4%	38.4%
Quantization	Car	Cat	Chair	Cow	Dining table	Dog	Horse
INT8 static post training	37.0%	41.4%	6%	15.8%	24.7%	23.4%	42.3%
Quantization	Motorbike	Person	Potted plant	Sheep	Sofa	Train	TV monitor
INT8 static post training	43.2%	36.0%	0.6%	6.6%	23.1%	49.9%	21.3%
Quantization		Sparsity	MACs	Mem usage	Model size	Mean inference time	Std inference time
INT8 static post training		0	0	353 MB	3.85 MB	0.08s	0.01s

Wassim
Laptop

ESIEE PARIS

Influence of quantization

Size of the model: decreased by 9.65 MB **(÷ 3.5)**
Mean inference time: decreases ⟶ FPS increases **(x 2.5)**
Memory consumption: cannot conclude

Influence of pruning

Pruning

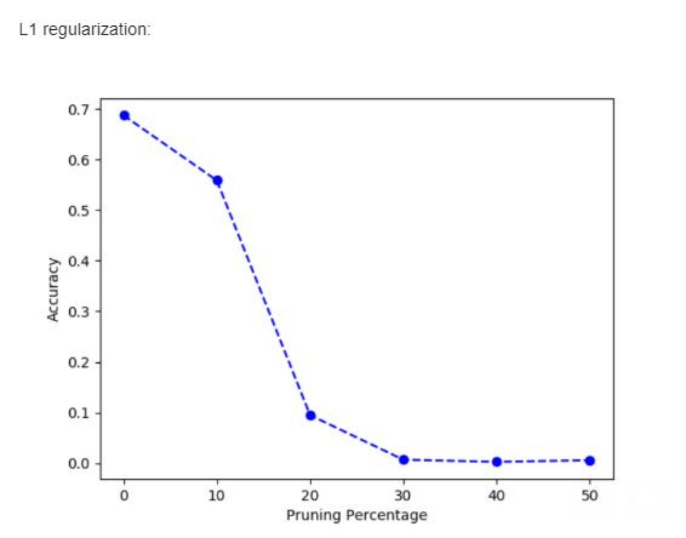
Definition: Reduce the network size through compression, by determining the importance of connections.

Type	Description
From scratch	Pruning pipeline that can be learned from randomly initialized weights
L1-norm	Layers with filters with smaller L1-norm will be pruned
Transformable architecture search	Width and depth of the pruned network are obtained through knowledge transfer from the original
Self adaptative	A pruning module is embedded in each layer, convolution skipped if the pruning decision is 0
Random	Randomly chooses weights to be set to 0

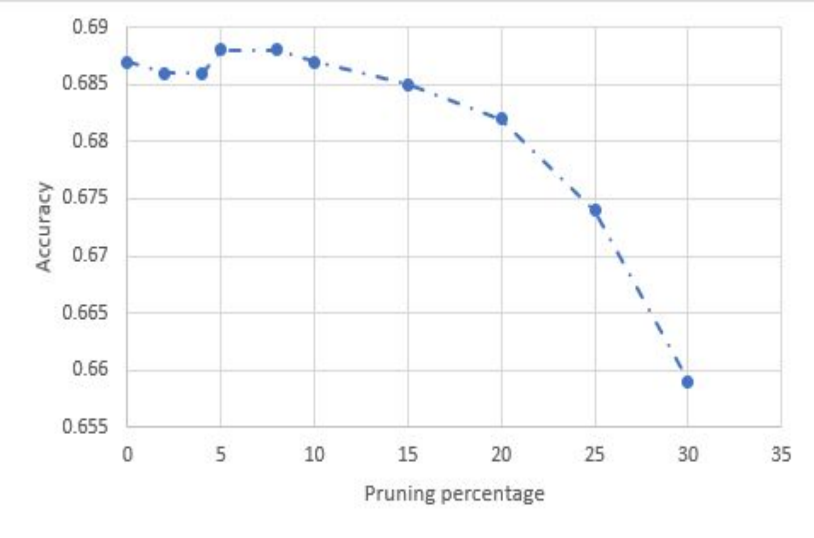
ESIEE PARIS

Main challenges and **solutions**

Pruning all possible layers hurt the accuracy ⟶ we pruned only the convolution layers



Google pre trained: 02/03



Google pre trained: 24/04

Main challenges and **solutions**

Model size increasing

26.5MB → 13.5MB

With pruning masks

Without pruning masks

ESIEE PARIS

Main challenges and **solutions**

Memory consumption

235 MB → 101 MB

In project

Outside of project

ESIEE PARIS

Google Model

Pruning %	Overall Accuracy	Car	Bicycle	Train	Person	Accuracy drop
0%	68.7%	74.2%	77.8%	83.5%	71.6%	0
2%	68.6%	74.2%	77.7%	83.3%	71.6%	-0.1%
4%	68.6%	74.3%	77.7%	83.3%	71.5%	-0.1%
5%	68.8%	74.3%	77.8%	83.5%	71.6%	-0.1%
8%	68.8%	74.4%	78.0%	84.2%	71.6%	+0.1%
10%	68.7%	74.4%	78.3%	83.7%	71.6%	0
15%	68.5%	82.5%	78.1%	83.5%	71.6%	-0.2%
20%	68.2%	74.0%	77.1%	83.7%	71.3%	-0.5%
25%	67.4%	72.5%	77.1%	80.7%	70.6%	-1.3%
30%	65.9%	72.3%	75.7%	79.8%	69.8%	-2.8%
50%	6.3%	21.0%	19.1%	1.5%	12.0%	-62.4%

!

Alex
Laptop

ESIEE PARIS

Navya Model

Pruning %	Overall Accuracy	Car	Bicycle	Train	Person	Accuracy drop
0%	25.1%	42.3%	33.7%	42.9%	39.1%	0
2%	25.0%	42.3%	33.6%	42.6%	39.0%	-0.1%
4%	25.0%	42.3%	33.6%	42.6%	39.0%	-0.1%
5%	25.0%	42.3%	33.8%	43.4%	39.4%	-0.1%
8%	24.9%	42.2%	34.0%	43.9%	38.9%	-0.2%
10%	24.8%	41.7%	33.3%	43.1%	38.6%	-0.3%
15%	23.8%	39.4%	32.3%	42.0%	37.4%	-1.3%
20%	22.1%	37.6%	29.3%	38.1%	35.2%	-3%
25%	18.5%	33.1%	23.0%	34.7%	30.1%	-6.6%
30%	11.9%	24.1%	16.5%	12.3%	24.6%	-13.2%

Alex
Laptop

ESIEE PARIS

Navya Model metrics

Pruning (L1) %	MAP	Model size MB	Mean inference time (s)	Std inference time (s)	Memory consumption MB
0%	25.1%	13.5	0.7	0.09	207
2%	25.0%	13.5	0.7	0.07	237
4%	25.0%	13.5	0.8	0.08	235
5%	25.0%	13.5	0.7	0.09	235
8%	24.9%	13.5	0.8	0.08	238
10%	24.8%	13.5	0.8	0.08	240
15%	23.8%	13.5	0.7	0.09	245
20%	22.1%	13.5	0.7	0.08	249
25%	18.5%	13.5	0.8	0..09	254
30%	11.9%	13.5	0.7	0.08	261

Alex
Laptop

ESIEE PARIS

Navya Model metrics

Pruning (L1) %	MAP	Model Size MB	Mean inference time (s)	Std Inference Time (s)	Memory Consumption MB	MACs
8% Wassim	24.9%	26.5	0.2	0.04	408	0.65
8% Alexandra	24.9%	26.5	0.8	0.08	238	Not working

ESIEE PARIS

Influence of pruning

Size of the model: same

Mean inference time: decreases → FPS increases (x 3.5)

Memory consumption: ÷ 2 on outside script

Summary table

ESIEE PARIS

Google pre-trained model: 68.7% accuracy
Navya trained model: 25% accuracy

	Status	Main result
Mixed precision training Apex	Fail	Apex not compatible with MBV2
Post training static quantization INT8	Success	0% accuracy drop Size of model ÷ 3.5 FPS x 2.5 Navya model
Pruning L1 regularization	Success	<u>At 8% pruning</u> -0.2% accuracy Size of model: same in the project, ÷ 2 on outside script FPS: x 3.5 Navya model

Conclusion

Ways to improve

MBV3

Apex compatibilities with MBV3

Improve architecture of Navya model

Implement on Raspberry Pi

Basic performance test on embedded hardware

Unified work environment

THANK YOU FOR YOUR ATTENTION !