</

Machine Learning at Scale

/>

Speakers

Jafar Isbarov
Mirakram Aghalarov

1 0 1 1   0 1 1   0 1   1 0 1 1 0 0 1   1 0   1 1 0 1 1   0 1 1   0 1   1 1 0 1 1 0   1 1 0 1 1 1   1 1 0 1

About Us

Mirakram Aghalarov

- Lead Machine Learning Engineer @PRODATA
- Lecturer of AI&ML courses at BHOS
- PhD Candidate focusing Computer Vision at BHOS
- MSc Data Science and Engineering from Politecnico Di Torino
- Former Deep Learning Engineer at AIKO

Jafar Isbarov

- Lead Machine Learning Engineer @PRODATA
- MSc Computer Science student at George Washington University
- MSc Data Analytics student at ADA University
- Former Machine Learning Engineer at Azerbaijan AI Lab

Agenda

{01} Large Models: Training and Inference

{02} Deep Learning on Edge Devices

{03} ML on-cloud vs. on-premise

Working with large models |1

1011 011 01 1011001 10 11011 011 01 110110 110111 1101

</Issues with large models

- **Training**
  - **Doesn't fit into a single GPU**
  - **Training takes too long**

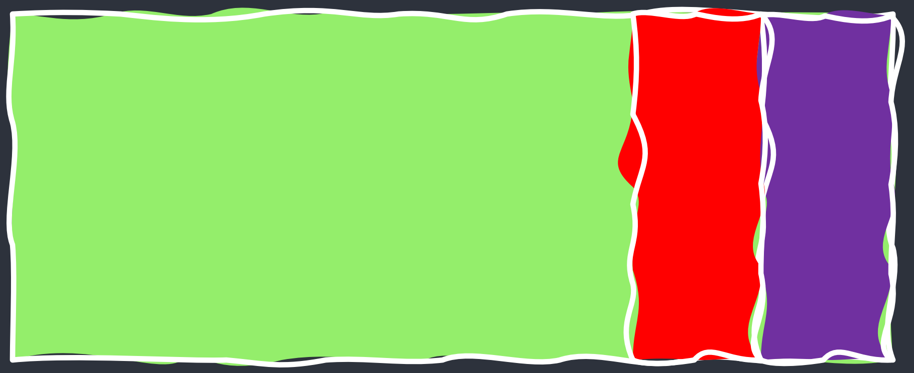
- Ops
  - Experiment tracking
  - Data lineage
  - Storage & backup

- **Deployment**
  - **Inference with CPU can be too slow**
  - **Batch size too small**

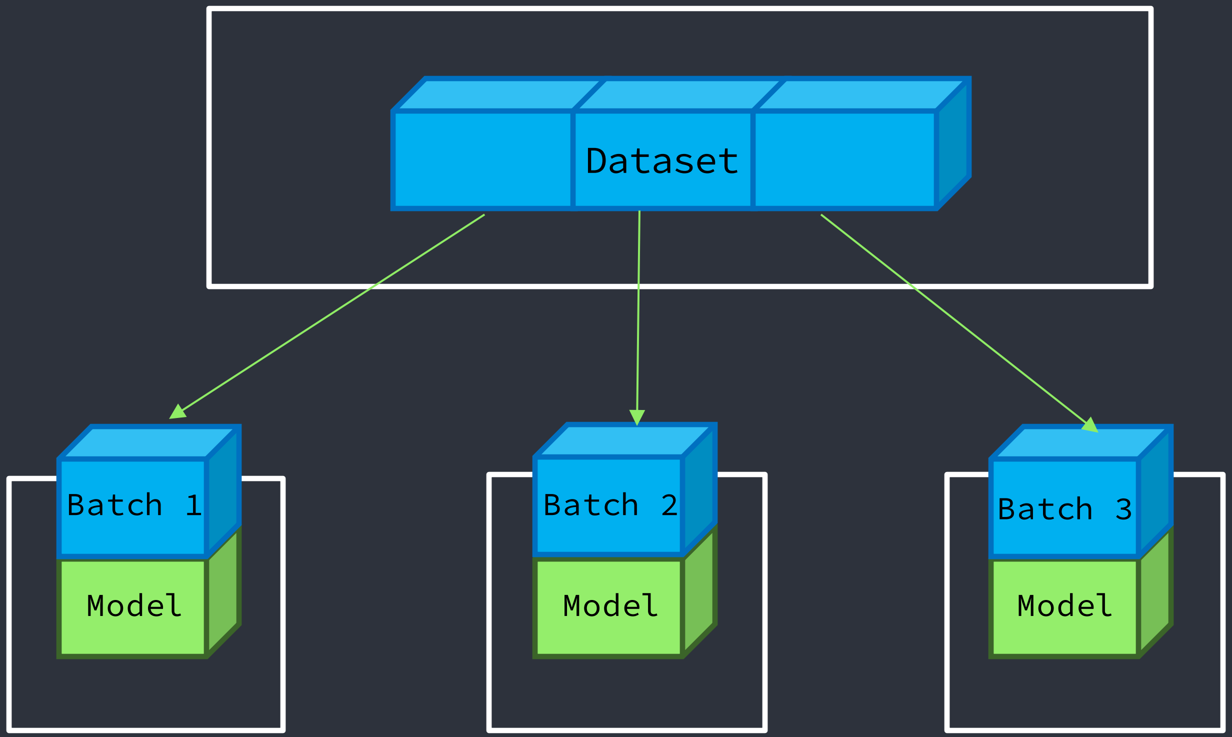
</Training large models

Memory usage:
- Weights
- Gradients
- Optimizer state

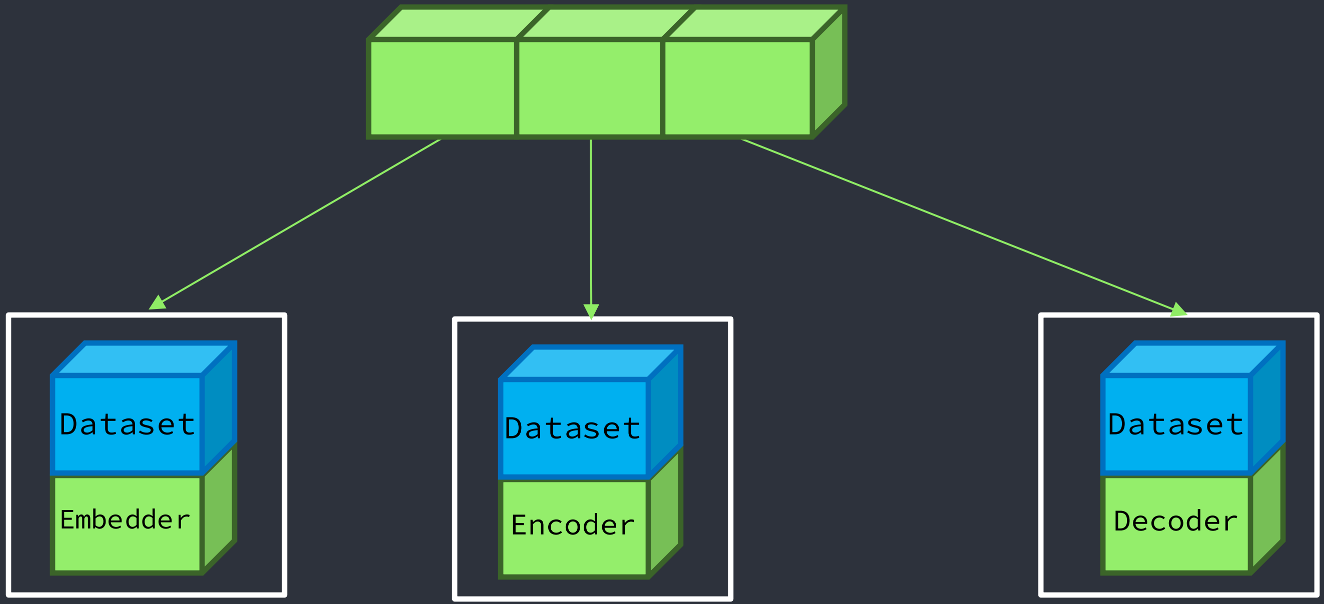
Training time:
- Batch size
- Time per step
- Dataset size



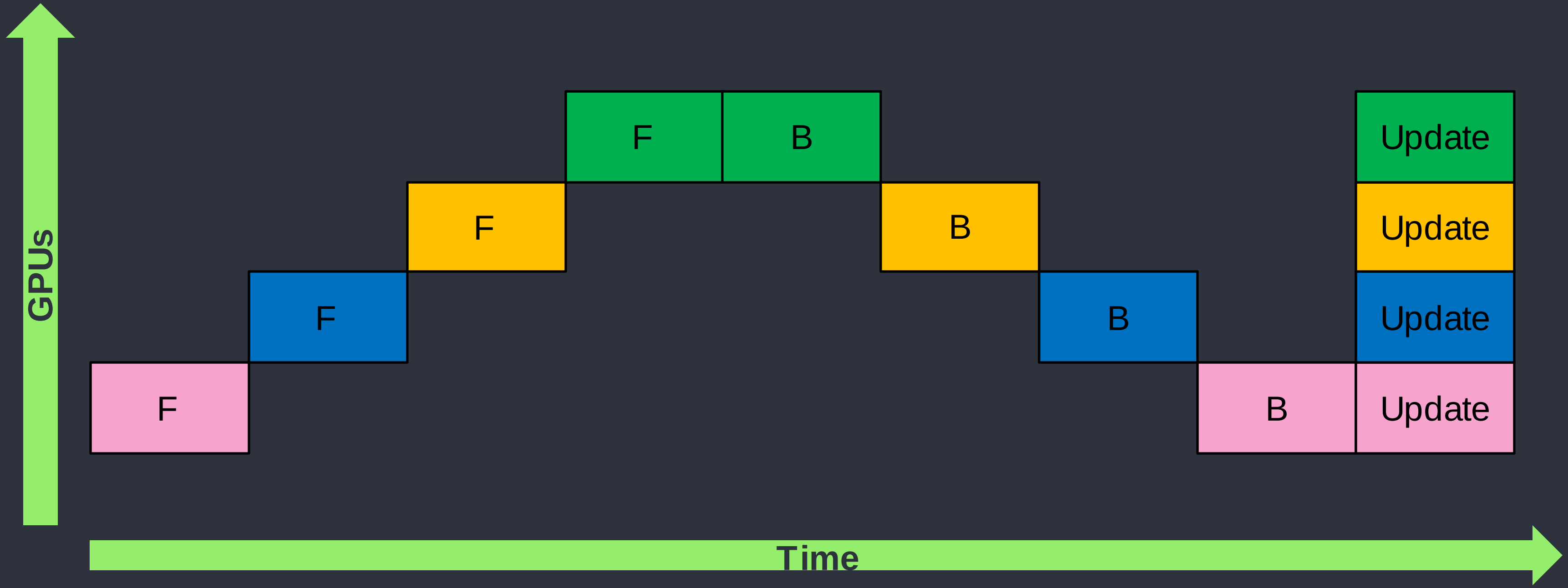
</Data parallelism



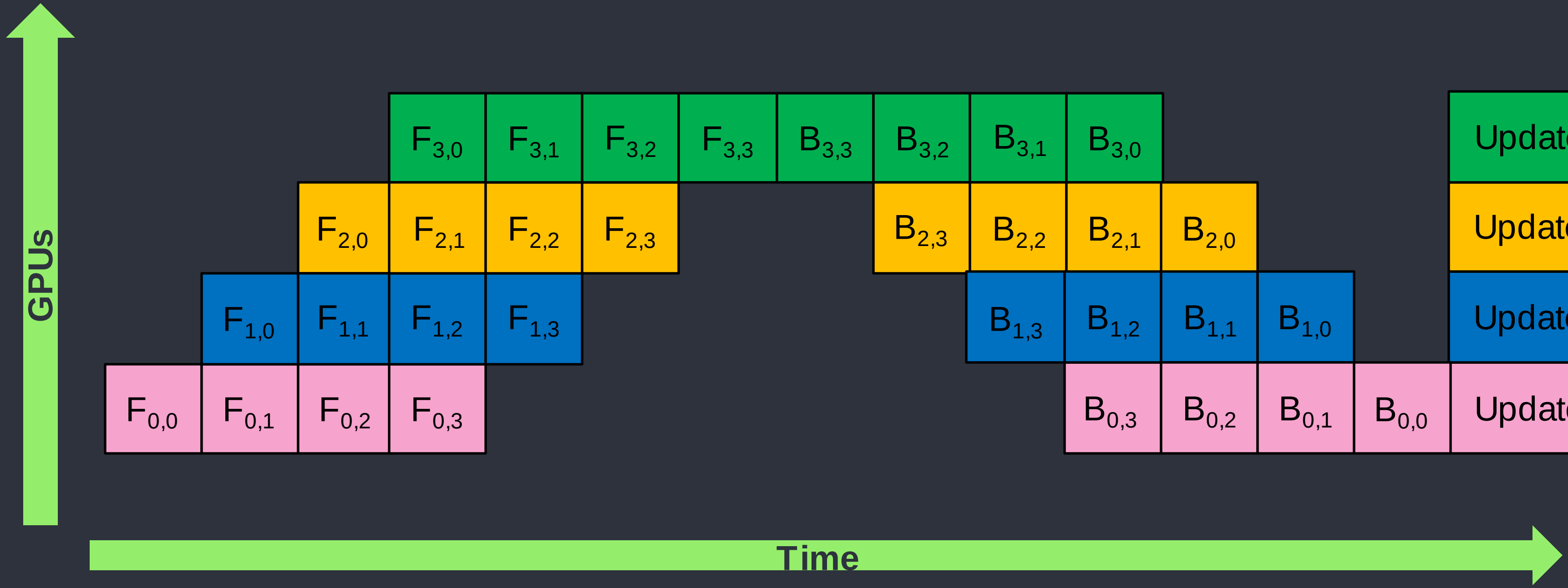
</Model parallelism



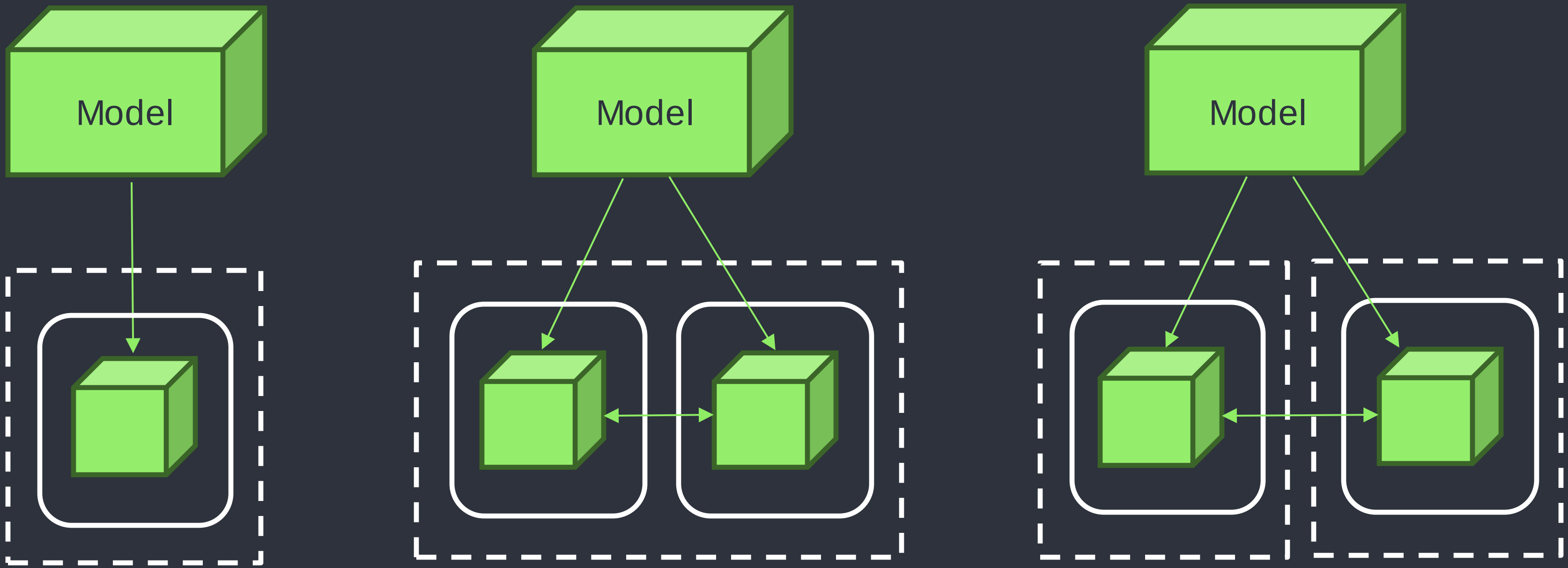
</Model parallelism



</Pipeline parallelism



</Large models in production



Single GPU, single node | Multiple GPUs, single node | Multiple GPUs, multiple nodes

Edge Computing

|2

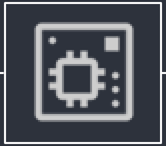
1011 011 01 1011001 10 11011 011 01 110110 110111 1101

Why Do We Need Edge Devices?

In 21st century, all of us have ben sorrounded with many gadgets

- Smartphone
- Smart Watch
- Laptops
- Sensors

Characteristics of Edge Devices



Advantages

- Privacy
- Less Power Consumption
- Cheaper
- Portability

Disadvantages

- Higher Latency
- Constraints in RAM
- Absence of GPU

Examples: Raspberry Pi, Nvidia Jetson, STM32

DL Optimization Methods for Edge

y = f(W*x + b)

x – Input Activation
y – Output Activation
W – Layer Weights
b – Bias
f() – Activation Function

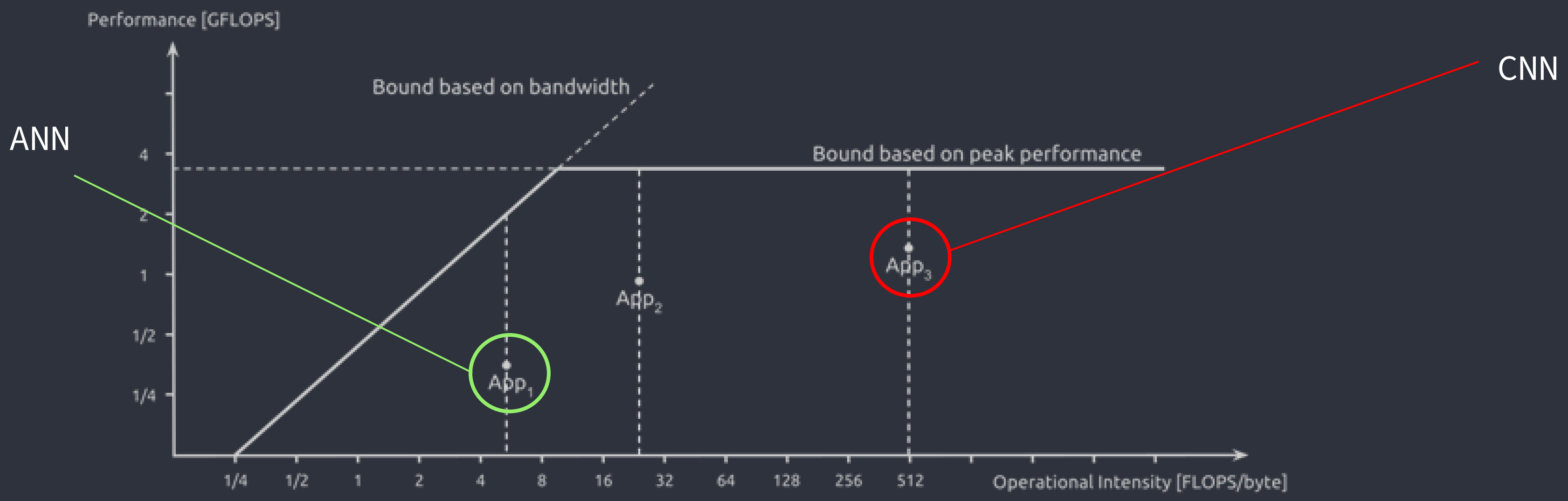
DL Optimization Methods for Edge

```
y = f(W*x + b)
```

- W * x requires O(NM) number of operations which is the major part of computation
- +b requires O(M) as same as f() activation function

Roofline Model

Roofline Model

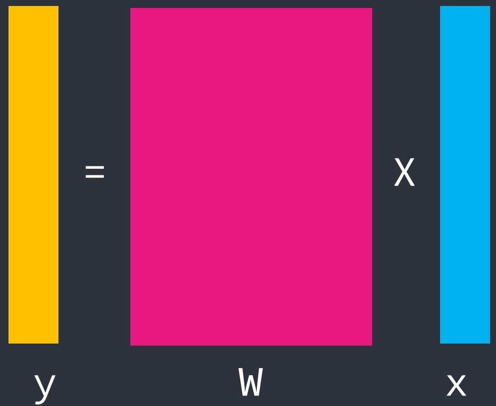


Artificial Neural Networks with Fully Connected layers are memory bound. By increasing weight re-use number of computations can be maximizable. Convolutional neural Networks are compute bound. Which means that to maximize the performance, number of FLOPs should be decreased

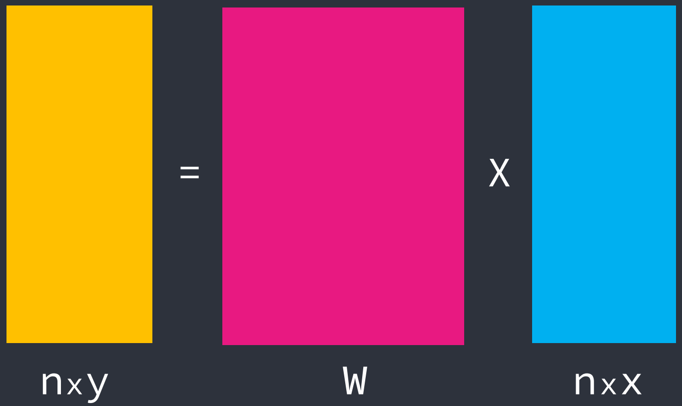
</Artificial Neural Network

Batch Size increase enables to multiply more than 1 input activation with the same input weight.

Batch Size of 1



Batch Size of n



</Convolutional Neural Networks

Re-use

Weights

Keeps the weights stationary and “slides” over the input Feature map

IFmap

Loads each input once and applies all filter weights

What about Energy?

Each access to memory requires 100x more energy than instruction using only register.

Therefore, Data and Weight Re-use increases the efficiency in power-critical applications.

Operation	nJ per operation
Register	0.45
L1	0.88
L2	7.72
Mem w Prefetch	52.14
Mem w/o Prefetch	232.62
Write to Mem	62.1

Reality

Registers can seem faster and effective while the data stored at given time is quite small and management of the data becomes hefty work. For CNN weight reuse are more common depending on the parameters of the network and underlying hardware

</Convolutional Neural Networks

As we said, Convolutional Neural networks are mainly compute-bound due to the number of FLOPs for each iteration. Therefore, it is not enough to maximize the batch size and weight re-use, but we need to decrease the number of FLOPs.

Standard Convolution: $K*K*C_{in}*C_{out}$ per location.

If we have 3x3 Kernel with 32 input channels to be extracted to 64 output channels:

$$3*3*32*64 = 18432$$

Which means we will have 18k operations for Standard Convolution!

</Convolutional Neural Networks

In order to decrease amount of FLOPs in CNNs, Efficient Convolutional Neural Networks have been proposed with Depthwise Separable Convolutions. This concept has been firstly introduced in MobileNets which were focused to be used in edge device

DW: $K*K*C_{in}$ per location.
PW: $1*1*C_{in}*C_{out}$ per location
Depthwise Seperable Convolutions: PW+DW

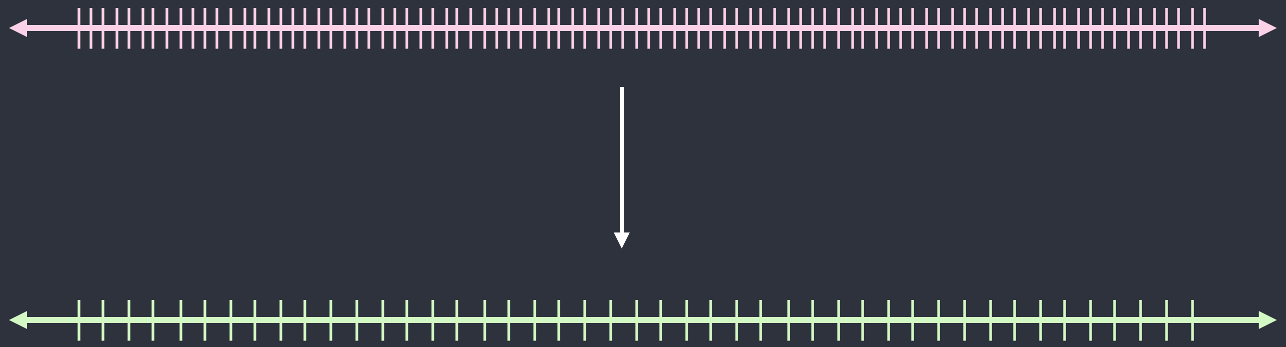
$$3*3*32 + 1*1*32*64 = 2336$$

Which is 7x less than Standard Convolutions

Quantization

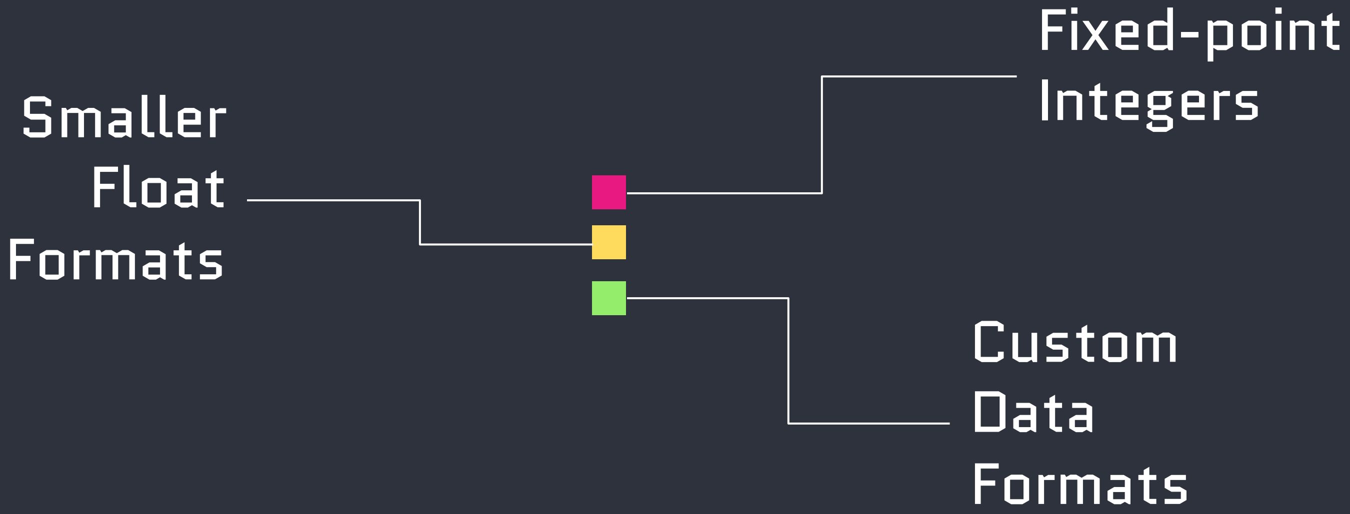
1011 011 01 1011001 10 11011 011 01 110110 110111 1101

Quantization



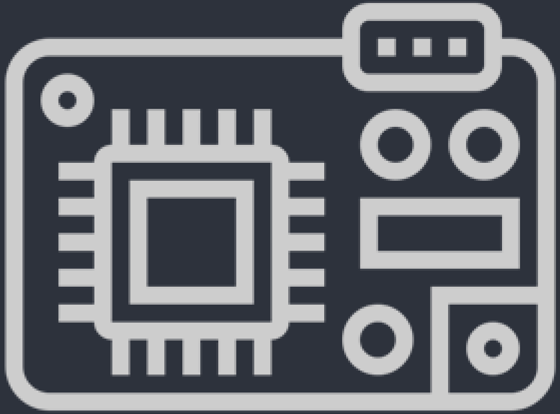
Quantization is a process of decrease in storage of variable by simply decreasing the precision. This compromise results with quantization error which has significancy depending on application.

Quantization



Going from 32-bit format into 8-bit format gives us benefit from both performance and energy consumption in memory transfers. If underlying hardware supports, smaller data formats are possible to be used.

Quantization



Considering the underlying hardware in microcontrollers, it is common to see integer (fixed/dynamic precision) quantization types.

The main reason is that most of the microcontrollers does not have floating point unit for float computation.

8-bit Fixed Point Integers

This is the method to represent the float number with the help of shared exponent Δ:

Δ = 0.05

X1 (int representation) = 35
X2 (int representation) = 13

X1 (Actual value) = 35 * 0.05 = 1.75
X2 (Actual value) = 13 * 0.05 = 0.65

1	0	0	1	0

1	1	0	1	1	0	0	1	0
1	1	0	1	1	0	0	1	0

Sign

Shared Exponent

Integer Component (Mantissa)

16 bit mini-float [half-precision]

- They are generally used in servers
- It keeps the floating-point architecture but decreases the size
- Some hardware like modern GPUs can natively use this format.

Pruning

1011 011 01 1011001 10 11011 011 01 110110 110111 1101

Pruning

Pruning

Weights Pruning

Exploits the redundancy in Neural Networks

Activation Pruning

Small activations set to zero

When is it Efficient?

{01} Making the variable 0 (zero) can decrease the storage size if file is stored in sparse format.

{02} Sparse format does not decrease the amount of computation because it is decompressed during inference.

{03} Node based structured pruning helps to decrease the model occupation in GPU and number of computation

</CSR and CSC format

Compressed Sparse Row

Matrices are scanned through rows, while non-zero values and corresponding indexes are saved

1	1	0	1	1	0	0	1	0
1	1	0	1	1	0	0	1	0
1	1	0	1	1	0	0	1	0
1	1	0	1	1	0	0	1	0
1	1	0	1	1	0	0	1	0
1	1	0	1	1	0	0	1	0

Compressed Sparse Column

Inverted version of CSR, Matrices are scanned through the columns.

1	1	0	1	1	0	0	1	0
1	1	0	1	1	0	0	1	0
1	1	0	1	1	0	0	1	0
1	1	0	1	1	0	0	1	0
1	1	0	1	1	0	0	1	0
1	1	0	1	1	0	0	1	0

</Structured Pruning

Structured Pruning considers underlying hardware, removes groups of weight to speed up the access to the registers and save the space of operating memory.

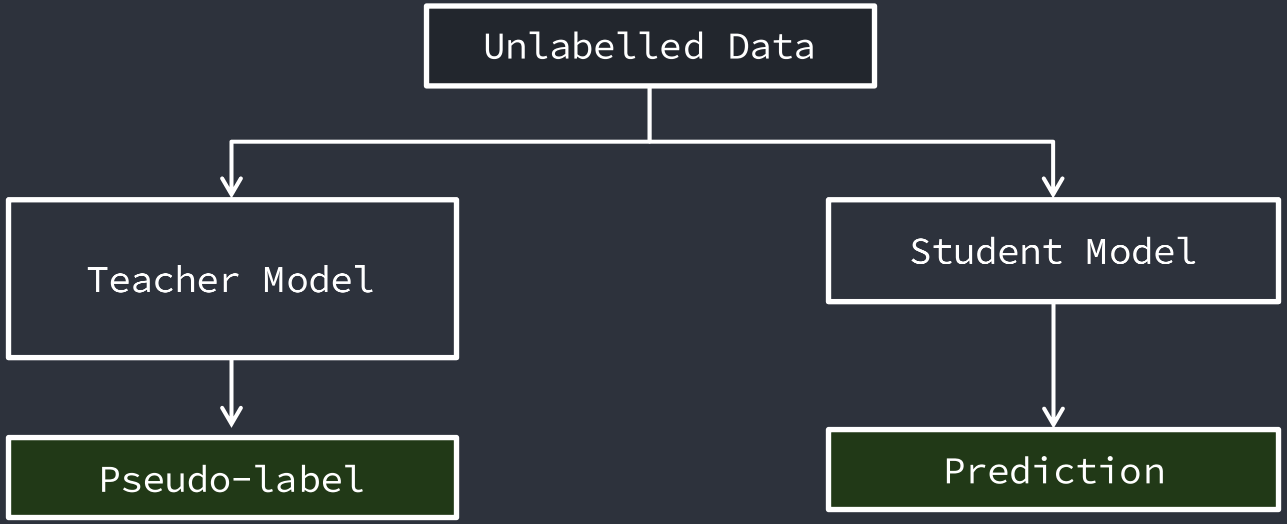
Another approach is node-based pruning in which larger group of weights are removed.

0	5	2	5	0	0
0	0	1	7	0	0
2	3	0	0	4	2
8	4	0	0	0	0
0	0	1	1	8	3
3	2	0	0	0	0

Knowledge Distillation

1011 011 01 1011001 10 11011 011 01 110110 110111 1101

</Knowledge Distillation



Transferring the knowledge to the smaller model
Adding natural regularization on overfitting
Generalizing the errors well on unseen data

Cloud vs. on-premise

|3

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</Why [not] cloud solutions?

- Readily scalable
- Out-of-box tools
- Lower maintenance cost
- (Usually) higher reliability

- Unmanaged cost increase
- Less customizable
- Higher latency
- Potential data privacy issues

</Why [not] on-premise solutions?

- More customizable
- Faster connections
- Easier to handle data privacy issues

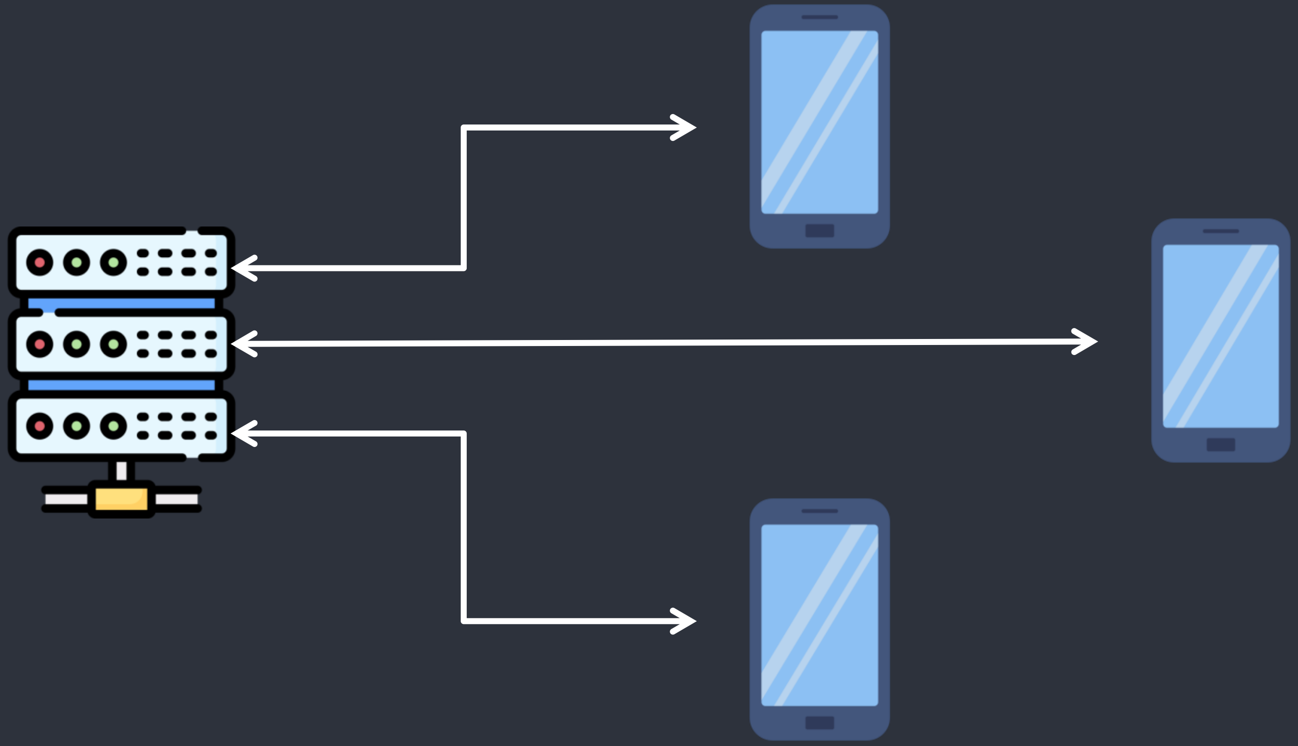
- Initial cost
  - Starting cost
  - Scaling cost
- Higher maintenance cost
- Reliability issues
- Not every service is available on-premise
  - OpenAI API
  - GCD, Azure ML, etc.

</Best of two worlds

- Hybrid of cloud and on-premise solutions
- Can be both a transitionary or a long-term strategy

- Outsourcing ML infrastructure while maintaining more traditional components in-house
- Edge devices + cloud backend

</Federated learning



Thank you

DevFest 2023
Baku, Azerbaijan

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1