Neural Networks Speed-up and Compression

Lecture 1: Introduction

The Team

Julia Gusak, PhD
Senior research scientist, Skoltech
(main collaborations: Prof. Oseledets, Prof. Cichocki, Prof. Phan)

Stanislav Abukhovich
PhD student, Skoltech
(supervisor: Prof.Cichocki)

Dmitry Ermilov
PhD student, Skoltech
(supervisor: Prof.Cichocki)

Konstantin Sobolev
PhD student, Skoltech
(supervisor: Prof.Phan)

Outline

- Motivation for neural networks speed-up and compression
- Estimation of neural networks effectiveness
- Overview of main compression techniques
- Course schedule, outcomes, and completion criteria

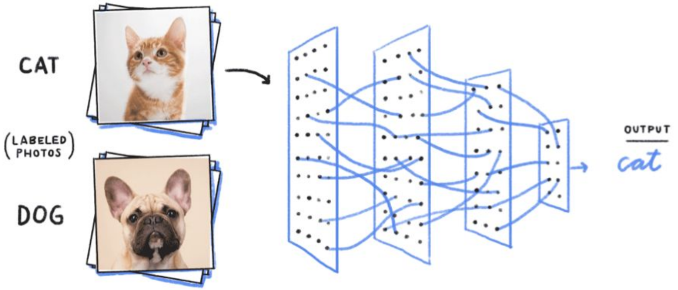
Motivation

What is a neural network?

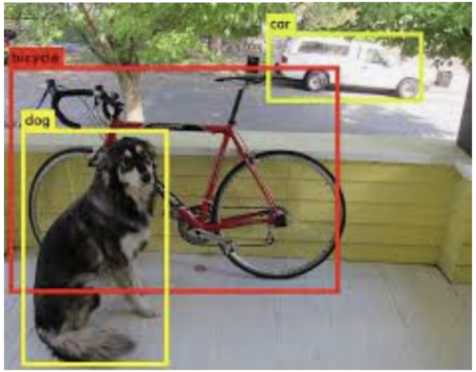
- Function that is defined by its parameters.
- Inference phase: given an input sample neural network provides the output.
- Training phase: neural networks' parameters are tuned using a set of input samples, loss function, and iterative gradient-based parameter updates.

Example: Computer Vision

- Image recognition
  - image -> NN -> class label



- Object detection
  - image -> NN -> image with frames around objects



- Action recognition
  - video frame -> NN -> action description

Example: Speech Recognition

- Speaker recognition
  - audio frame-> NN -> speaker ID
- Speech recognition
  - audio frame -> NN -> text corresponding to the audio frame
- Speech generation
  - audio frame -> NN -> next audio frame

Example: Natural Language Processing

- Sentiment analysis
  - sentence -> NN -> positive/negative
- Question & Answer
  - sentence-question -> NN -> sentence-answer
- Translation
  - sentence -> NN -> sentence translated into foreign language
- Text generation
  - sentence -> NN -> next sentence

What's the problem?

- Most state of the art deep neural networks are overparameterized and exhibit a high computational cost

- The size of neural networks affects
  - power and memory consumption
  - running time
  - CO_2 emission
  - money

What's the problem?

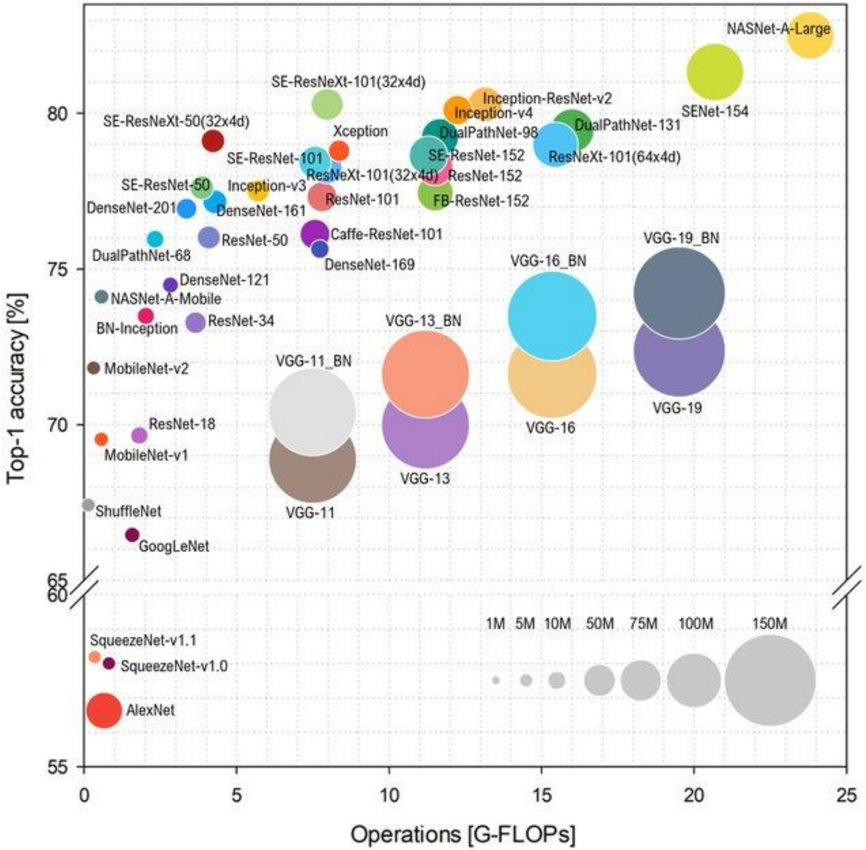
DL model limitations:

- High memory consumption
- Huge computational requirements
- Great power consumption

Difficult to deploy on portable devices

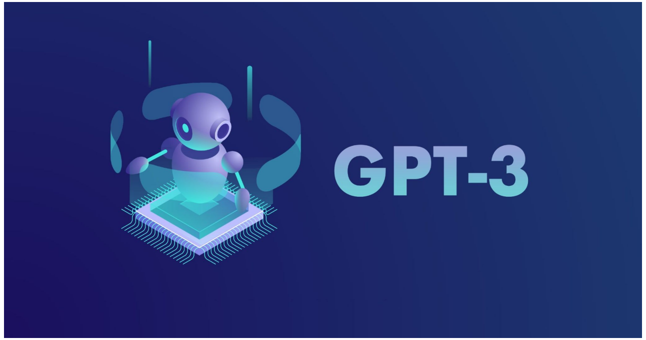
(e.g. laptops and smartphones)

Efficient architecture design is required



What's the problem?

Energy: 190 MW*hours → Air emissions : 85 tonnes of CO_2



GPT-3 training

Heating for 126 houses in Denmark

Car drive to the Moon

Different modalities, Similar Techniques

- Image
- Video
- Audio
- Text
- Medical signals (EEG, ECG)
- Physical measurements

-> NN -> prediction

! Similar Deep Learning techniques are used to solve a vast variety of tasks

What we want from Neural Networks?

- High predictive quality
- Robustness to the shifts in input data
- Efficient training (fast convergence to optimal parameters)
- **Efficient inference** (fast execution, low memory usage)

Estimation of effectiveness

Representation of Neural Network parameters

- Bits & Bytes
  - bit: 0 or 1
  - 1 B(=byte) = 8 bits
  - 1 KB = 1024(=2^10) bytes
  - 1 MB = 1024 KB = 1024*1024 bytes
  - 1 GB = 1024 MB = 1024*1024*1024 bytes
- Types of representations
  - int (sign|absolute value)
    - int8: 8 bits = 1 byte
    - int32: 32 bits = 4 bytes
  - float (sign|exponent|mantissa)
    - float32: 32 bits = 4 bytes
    - float64: 64 bits = 8 bytes

Key Factors

- Speed
  - Theoretical: FLOP, MAC, FLOP/second, Bytes/second
  - Empirical: wall-clock inference time, throughput
- Memory
  - Theoretical: (# of model parameters) * (# of bytes in one parameter)
  - Empirical: allocated memory

FLOP & MAC

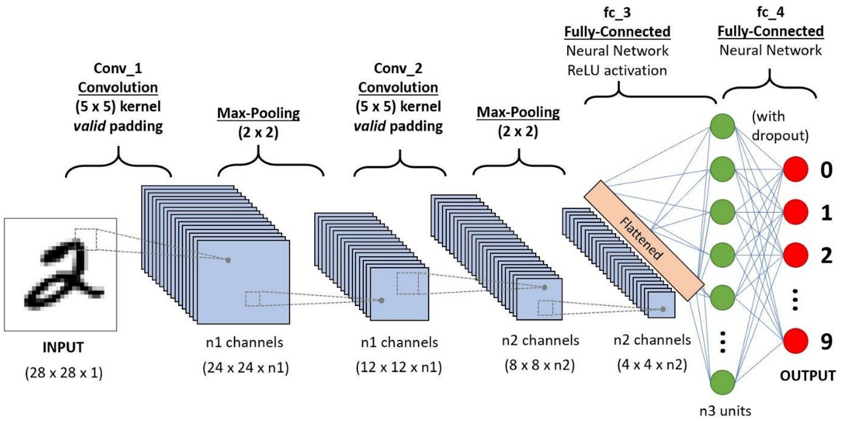
- FLOP - floating point operations
- MAC - multiply-accumulate operations

Example: c = c + (a * b), where a, b, c - scalars

- 2 FLOP
- 1 MAC

Recap: Neural Networks

- Main operations
  - Linear
  - Non-linear
- Main neural networks types
  - Feed-forward
  - Residual
  - Attention-based



Linear (Fully-connected) Layer

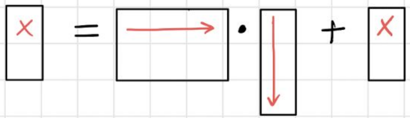
X – input, $N_{in} \times 1$

Y – output, $N_{out} \times 1$

W – weight matrix, $N_{in} \times N_{out}$

b – bias vector, $N_{out} \times 1$

$$Y = W^T X + b$$



$$Params_{Linear} = N_{in} N_{out} + N_{out}$$

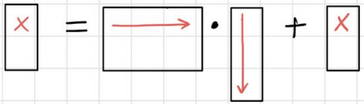
Liner (Fully-connected) Layer: FLOP

Example

$N_{in} = 3, \quad N_{out} = 2 \quad \Rightarrow \quad X: \; 3\times1 \qquad W: \; 3\times2$

$\qquad\qquad\qquad\qquad\qquad\quad Y: \; 2\times1 \qquad b: \; 2\times1$

$$Y = W^T X + b$$



$$Params_{Linear} = N_{in} N_{out} + N_{out} = 3\cdot2 + 2 = 8$$

$FLOP_{Linear}$ – ?

$$y_i = \underbrace{(w_{i1}\cdot x_1 + w_{i2}\cdot x_2 + w_{i3}\cdot x_3)}_{\substack{\text{mult: } 3 \\ \text{add: } 2}} \underbrace{+\, b_i}_{\text{add: } 1}$$

$$FLOP_{y_i} = N_{in} + (N_{in} - 1) + 1 = 2N_{in}$$

$$FLOP_{Linear} = 2\, N_{in} N_{out}$$

Linear (Fully-connected) Layer: MAC

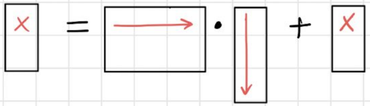
Example

$N_{in} = 3, \quad N_{out} = 2 \quad \Rightarrow$

X: 3×1 W: 3×2

Y: 2×1 b: 2×1

$$Y = W^T X + b$$



MAC_{Linear} – ?

$$y_i := b_i$$
$$y_i = y_i + w_{i1} \cdot x_1$$
$$y_i = y_i + w_{i2} \cdot x_2$$
$$y_i = y_i + w_{i3} \cdot x_3$$

$$MAC_{y_i} = N_{in}$$

$$MAC_{Linear} = N_{in} N_{out}$$

Convolutional Layer

X – input, $C_{in} \times H_{in} \times W_{in}$

Y – output, $C_{out} \times H_{out} \times W_{out}$

W – weight (kernel), $C_{out} \times C_{in} \times k_h \times k_w$

b – bias, C_{out}

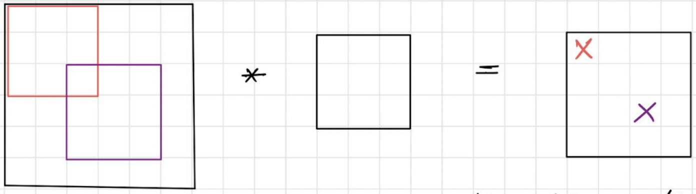
$$\boxed{Y = X * W + b}$$

Example

$C_{in} = C_{out} = 1$

$H_{in} = W_{in} = 6$

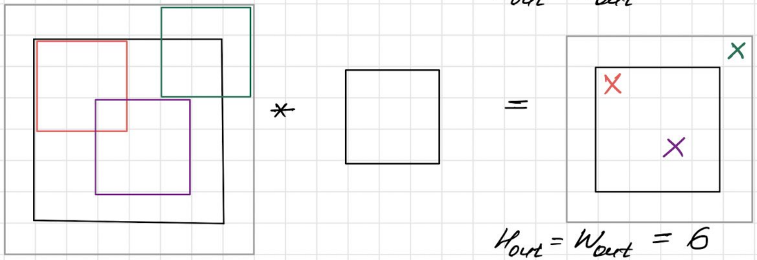
$k_h = k_w = 3$



$H_{out} = W_{out} = 4$

$Padding_h = 1$

$Padding_w = 1$



$H_{out} = W_{out} = 6$

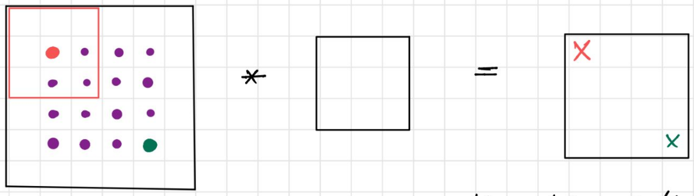
Convolutional Layer

Example

$C_{in} = C_{out} = 1$

$H_{in} = W_{in} = 6$

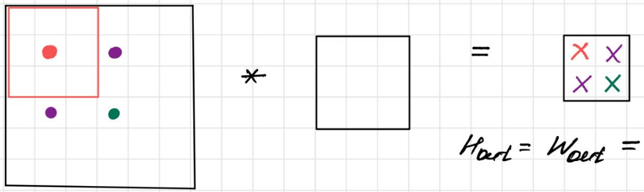
$k_h = k_w = 3$



$H_{out} = W_{out} = 4$

$Stride_h = 2$

$Stride_w = 2$



$H_{out} = W_{out} = 2$

$$MAC_{y_{ij}} = k_h k_w$$

$C_{in} = C_{out} = 1 \Longrightarrow MAC_{conv} = H_{out} W_{out} k_h k_w$

$C_{in} = 1,\ C_{out} > 1 \Longrightarrow MAC_{conv} = H_{out} W_{out} k_h k_w C_{out}$

Convolutional Layer

Standard

$$\text{Params} = k_h k_w C_{in} C_{out}$$

$$\text{MAC} = H_{out} W_{out} \cdot k_h k_w C_{in} \cdot C_{out}$$

Depth-wise $(C_{out} = C_{in} = C)$

$$\text{Params} = k_h k_w 1\, C$$

$$\text{MAC} = H_{out} W_{out} \cdot k_h k_w 1 \cdot C$$

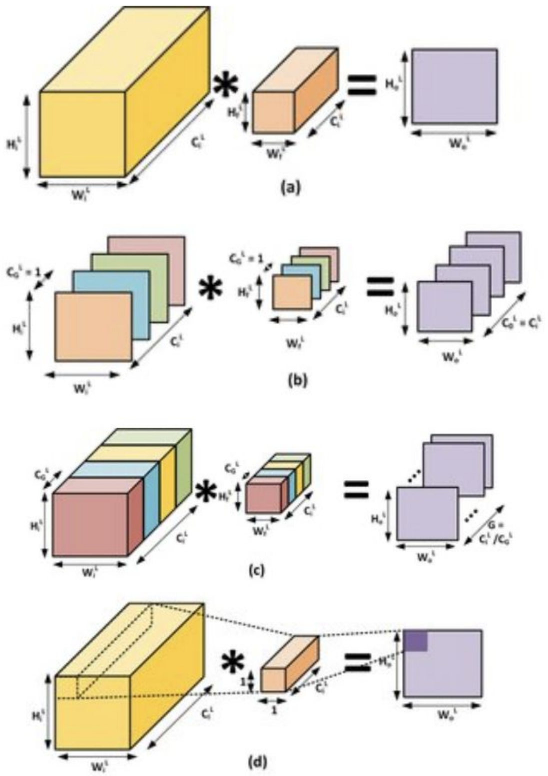
Group (G – # of groups)

$$\text{Params} = k_h k_w \frac{C_{in}}{G} \cdot C_{out}$$

$$\text{MAC} = H_{out} W_{out} \cdot k_h k_w \frac{C_{in}}{G} \cdot C_{out}$$

Point-wise

Standard with $k_h = k_w = 1$



Practical exercises

- Compute Parameters/ FLOP/ MAC for different Linear layers
  - theoretically
  - empirically, using Python packages
    - FlopCo
    - time

Practical exercises

- Non-linear layers
    - ReLU
    - GeLU
    - etc.
- Normalization layers
    - Layer Norm
    - Batch Norm
    - etc.

Normalization Layers

$$\hat{x}_i = \frac{1}{\sigma_i}(x_i - \mu_i), \quad \mu_i = \frac{1}{m}\sum_{k \in S_i} x_k$$

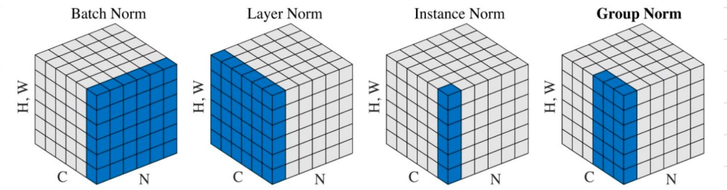
$$\sigma_i = \sqrt{\frac{1}{m}\sum_{k \in S_i}(x_k - \mu_i)^2 + \varepsilon}$$

S_i – set of pixels in which mean & std are computed

$$i = (i_N, i_C, i_H, i_W)$$

$y_i = \gamma \hat{x}_i + \beta$, γ, β – trainable scale & shift

		Same statistics for:
BN	$S_i = \{k \mid k_C = i_C\}$	all spacial coords (h, w) in one channel (across batch)
LN	$S_i = \{k \mid k_N = i_N\}$	all channel-spacial coords (c, h, w)
IN	$S_i = \{k \mid k_N = i_N,\ k_C = i_C\}$	all spacial coords (h, w) in one channel
GN	$S_i = \left\{k \mid k_N = i_N,\ \left\lfloor \frac{k_C}{C/G} \right\rfloor = \left\lfloor \frac{i_C}{C/G} \right\rfloor \right\}$	all spacial coords (C/G, h, w) in one group of channels



Intensity of Operations

- Real speed is limited by processor characteristics
  - time of math operation (T_math)
  - the time of memory access operation (T_mem)

- p = T_math/T_mem = BW_math/BW_mem - intensity of operation,
  - BW_math - speed of floating point operations (measured in FLOP/second)
  - BW_mem - speed of memory access (measured in bytes/second)
  - Modern GPUs: BW_math >> BW_mem

- p_algo = #FLOP/#Mem
  - p_algo = p, - memory and processor are fully utilized during computations
  - p_algo < p, - algo is limited by memory access speed
  - p_algo > p, - algo is limited by math operation speed

Intensity of operations: example

16-bit arithmetics, GPU Nvidia A100

Operation	Arithmetics' intensity	Time is limited by .....
Linear (4096 x 1024, batch 512)	315 FLOP/B	arithmetics
Linear (4096 x 1024, batch 1)	1 FLOP/B	memory access
MaxPooling with kernel 3x3	2.25 FLOP/B	memory access
ReLU	0.25 FLOP/B	memory access
LayerNorm	< 10 FLOP/B	memory access

Compression Techniques: Brief Overview

Compression Methods

- **Pruning methods** (structured / unstructured)
    - redundant weights / neurons are pruned, hence, the whole model is compressed.
- **Tensor-based methods**
    - use matrix or tensor decomposition to estimate the informative parameters of deep neural networks;
    - In most cases, a much lower total computational cost can be achieved by replacing a convolutional layer with a sequence of several smaller convolutional layers.
- **Quantization methods**
    - use low-bit representations for weights / activations;
    - can significantly accelerate networks, but they usually require special hardware to reach a theoretical speed-up in practice.
- **Knowledge distillation methods**
    - deal with a pre-trained network (*teacher* network), and an accelerated network (*student* network);
    - outputs (resulting and/or intermediate) of the teacher network are used to guide the student network training.

Course Info

Schedule

1. Introduction. Measures of neural networks effectiveness.
2. Pruning.
3. Tensor methods.
4. Quantization.
5. Inference on mobile devices.
6. 6.1. Knowledge distillation.

   6.2. Large neural networks training acceleration

Completion Criteria

- 5 Homework assignments (HAs) + 1 optional HA
- Acceptance rules
    - HA submitted before 11.59 am next day - max 100%
    - HA submitted before 11.59 am next next day - max 75%
    - HA submitted before 11.59 am last lecture day - max 50%
- To pass the course:
    - Get >= 50% for any 4 of required 5 HAs

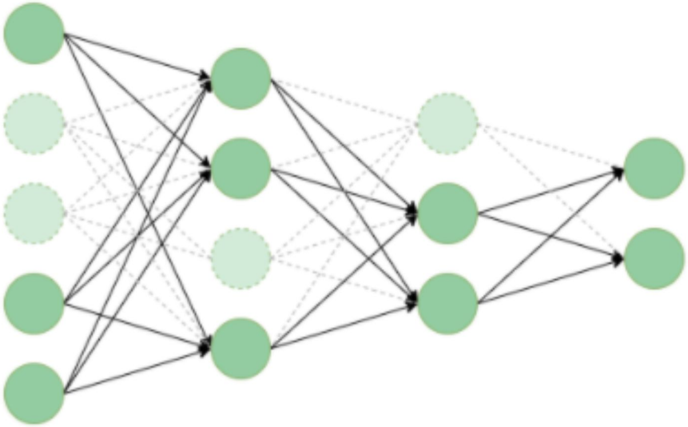
Resources

- FlopCo https://github.com/juliagusak/flopco-pytorch
    - Python library that aims to make FLOPs and MACs counting simple and accessible for PyTorch neural networks.
    - FlopCo allows to collect other useful model statistics, such as number of parameters, shapes of layer inputs/outputs, etc.

- Papers & code links

  https://github.com/juliagusak/model-compression-and-acceleration-progress

See you Next Lecture!



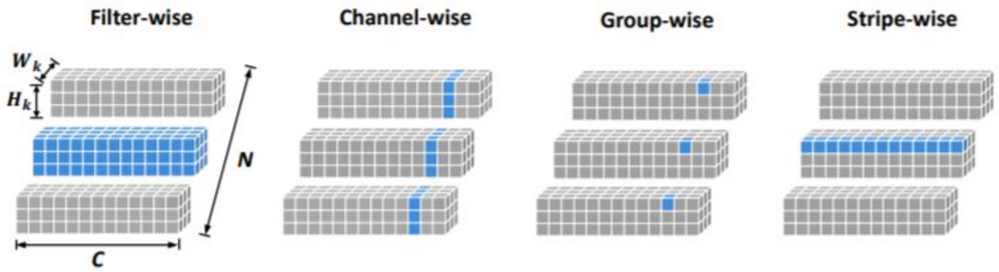


Figure 2: The visualization of different types of pruning.