

APAI LAB03

DNN shrinking & quantization

Credits: *Davide Nadalini, Lorenzo Lamberti, Luca Bompani, Alberto Dequino, Francesco Conti.*
(University of Bologna)

Contacts: *d.nadalini@unibo.it, lorenzo.lamberti@unibo.it, alberto.dequino@unibo.it,*
luca.bompani5@unibo.it

How to deliver the assignment:

DEADLINE: 25/10/2024 at 14:30

Instructions:

- Use Virtuale platform to load your file
- update only the .ipynb file, named as follows: LAB#_APAI_yourname.ipynb

Important: the notebook must be pre-run by you. Outputs must be correct and visible when you download it

Links to COLAB exercise:

[GitHub](#)

Solution is coming after the deadline

Lab session summary:

1. Load model's trained weights of LAB1;
2. Reduce network's size under 5MMAC;
3. Re-train the reduced network and verify network's accuracy;
4. Quantize with QuantLab;
5. Export Onnx and analyze the float32 and quantized models with Netron.

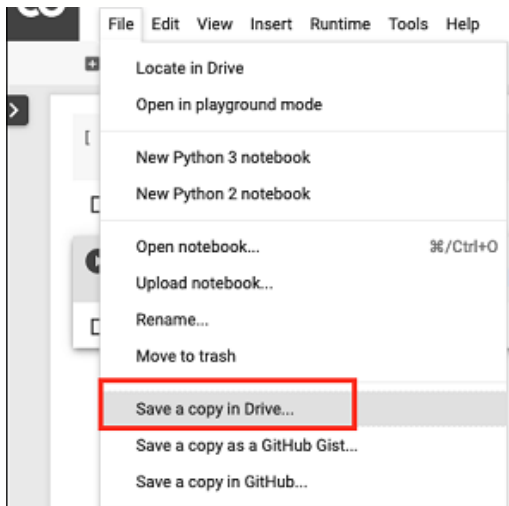
Setup:

0. Open the link to launch [COLAB](#) from the [GitHub repository](#)



1. Create your own [COLAB](#) copy of the notebook!

- In Google Colab, use the menu: *File > Save a copy in Drive*

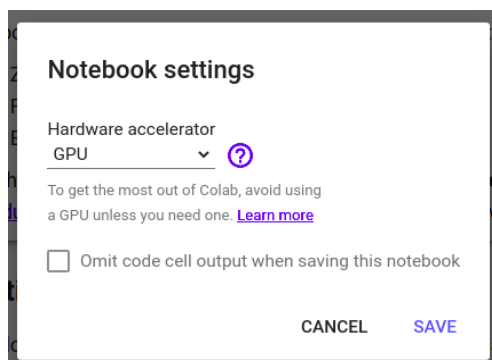


Then, log in with your own Google account.

Note: in case you don't complete the assignment, Colab keeps your file in your Google Drive!

2. Activate/deactivate GPU: *Runtime -> Change runtime type*

- **Note:** *If you use for too much time the GPU, your account will be limited to CPU for 24h.*



Task 1: Load Pre-Trained Model from LAB5

Resources: The pre-trained model is available at [this GitHub link](#)

Description: Define the network topology first (I already coded it for you!) and then 1) download the pre-trained weights from APAI-LAB05 github, and 2) load these weights into the network.

Output: Verify the correct loading by testing our model on the validation set. You must achieve a ~90% accuracy!

Task 2: Reduce the model size below 5MMAC

Resources: 1) [torchinfo](#) (to check the network MACs), and 2) the MAC operations formula for a single convolutional layer:

Say you have these parameters:

K is your kernel width and height

C_{in} is number of input channels

C_{out} is number of output channels

H_{out} and W_{out} height and width of the output matrix respectively

Then, you need $(K^2) * C_{in}$ MAC operations to compute each output feature map and you will have $H_{out} * W_{out} * C_{out}$ of these output feature maps. Then total MACs would be:

$$(K^2) * C_{in} * H_{out} * W_{out} * C_{out}$$

Figure 3. the MAC operations formula

Description: We want to decrease the model size below 5MMAC operations, while keeping a minimal accuracy of 90%. (All the elements in the MAC formula will affect the total MAC count. Therefore, you can shrink the network by reducing any of them. Use the previously implemented tools (torchinfo) to count parameters and MAC operations. For decreasing the model size, you can change the topology in three ways:

1. Modify the number of channels in the convolutional layers.
2. Add or remove layers
3. Play with the "stride" of the convolutional layers.

Suggestion: work on the number of channels of each conv layer! (easiest solution)

Output: a new NN model trained on Fashion MNIST, with less than 5MMACs and at least 90% Accuracy.

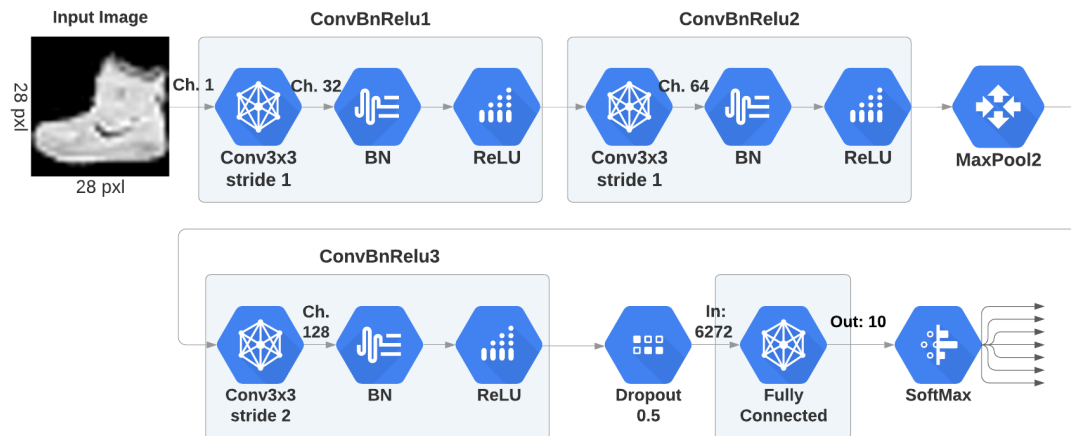


Figure 4. The original Network topology defined in LAB5. Now you must shrink it down!

Task 3: Retrain the network and verify validation accuracy

Resources: same exercise of LAB5 Task 5, but on a different network !

Description: after verifying that your reduced network fits inside the 5MMAC constraint that we gave you, it is time to train it! In fact, we can not load the older weights from lab5, since the topology of the network has changed since. Therefore, we will apply 2 epochs of training. and we want to verify that the new network still achieves an accuracy of 90%, even if ~3 times smaller

Output: a network with less than 5MMAC that achieves a ~90% accuracy on the fashion-mnist dataset.

Task4: Quantize with QuantLab

Resources:

- 2021 Class: [Theoretical Lesson Slides: Quantization](#)
- 2021 slides: [Quantlab lesson slides](#)
- 2021 colab: [Quantlab Hands-on: COLAB](#)

Description: We want to quantize the network from a float32 representation to a fixed-point representation (integer) using Quantlab, a quantization tool introduced in LAB03. As we studied in class, quantization is a technique that allow us to compress the neural network (i.e., reduce its size) and to accelerate the inference time (float operations vs int operations). We want to quantize the network as following:

- First convolutional layer and last FullyConnected layer quantized to 8 bits (x4 compression);
- all the other layers to 2 bits (note: from 32 bits to only 2 bits is 16x compression !).


Output:

1. Quantize the network defined in LAB5 to 2 bits, except for the first convolutional layer and last FullyConnected layer that must be quantized to 8 bits.
2. After the float-to-fake transformation, some layers are replaced. Write down the names of the blocks replacing: `Conv2d()`, `Linear()`, `ReLU()`

Task5: Export to ONNX format

Resources:

1. Onnx Website: [ONNX](#)
2. Onnx GitHub repository: [GitHub - onnx/onnx: Open standard for machine learning interoperability](#)
3. Netron (tool for visualizing ONNX models) [Netron APP](#)

Quickstart:  [Netron的操作](#) how to use netron to visualize a ONNX model

Description: The trained networks can be exported in ONNX (Open Neural Network eXchange) format for deployment tools. This format is recognized as the official format to export a deep neural network. This format stores all the necessary elements to reload a network:

- topology: number of layers, how they are linked and their type;
- layer structure: stride, filter dimension, channels, etc..
- parameters: for each layer, all the weights and numeric parameters are stored.

To see how the model looks like, the cell will trigger a download to your local machine and then open a frame into a tool called Netron that can be used to visualize it: just drag the ONNX file and drop it on top of the Netron frame!

Output: Use Netron to analyze the ONNX file of the floating point model and of the TrueQuantized one. What are the differences between them? Answer to the following questions:

Float32 model:

- What's the data format of the convolutional layers' weights?
- Is the decimal part of the convolutional weights !=0 ? Paste here the value of a single weight as an example.
- What's the kernel size of all the convolutions?

Quantized Model:

- What's the size of the input image?
- What's the data format of all the convolutional weights?
- Is the decimal part of the convolutional weights !=0 ? . Paste here the value of a single weight as an example.
- What's the quantization bitwidth of the `ConvBnRelu1` convolutional weights? (look for `weight_bits`)
- What's the quantization bitwidth of the `ConvBnRelu2` convolutional weights? (look for `weight_bits`)
- What's the kernel size of all the convolutions?

Hint: search the information that i deleted in the images below

INPUTS

weight

name: **ConvBnRelu1.bn1.weight**

-

kind: **Tensor**

type: ~~float32[512]~~