Mikaël Salasca (miksa599)
Jonas Michelberger (jonmi043)

LAB 6

https://colab.research.google.com/drive/1qzfDPeBc09iiYNruoPimcYdwEbpL0bID#sandboxMode=true&scrollTo=VZ2YjlIM4Wr7

Q1)

```
# Compute the gradient (partial derivative) of the loss fonction with regard to each parameter
gradient = tape.gradient(loss, parameters)
```

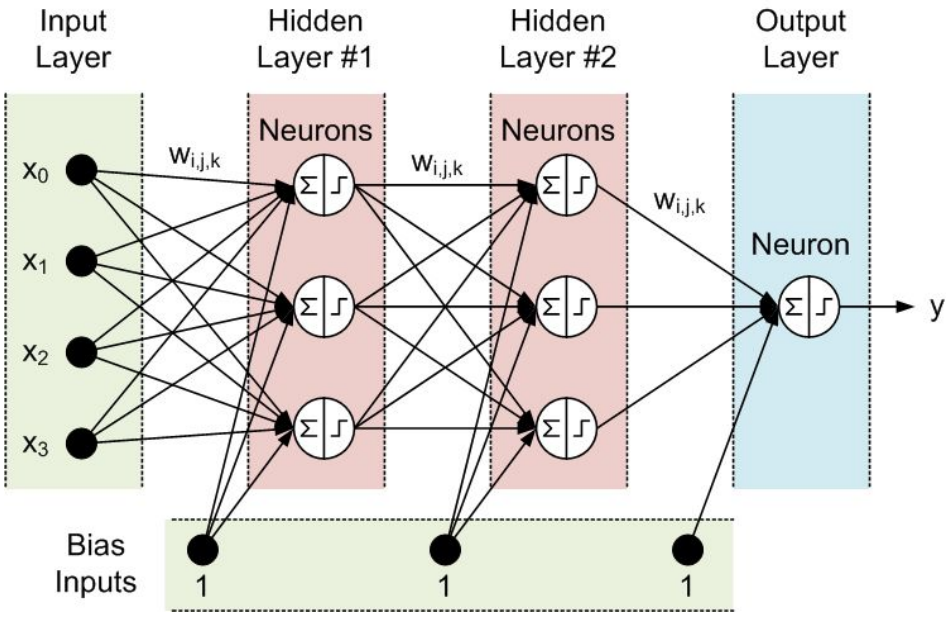
Q2)

A neuron is just a non-linear transformation (e.g. sigmoid/ReLU) of a linear model, implying one parameter for each input dimension, + 1 for the line intercept/constant (also called neuron "bias")

With width m, height n, previous layer's filters d and account for all such filters k in the current layer.

Number of parameters = ((m * n * d)+1)* k), added 1 because of the bias term for each filter.

Here :
Nb parameters = (28*28+1)*128 = **100480**

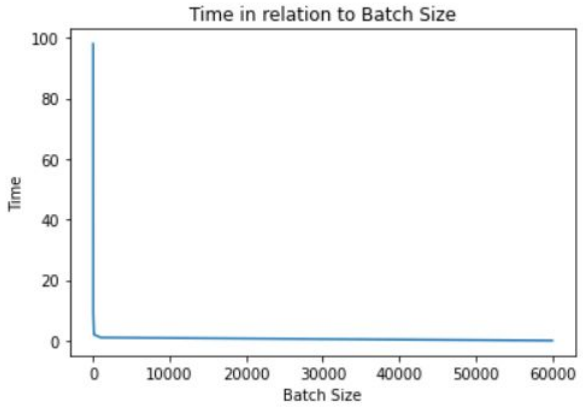


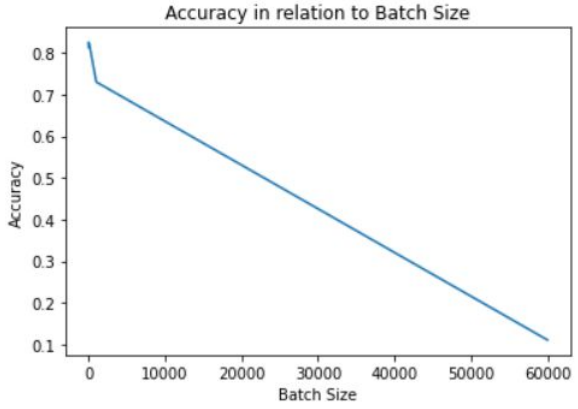
Mikaël Salasca (miksa599)
Jonas Michelberger (jonmi043)

Q3)

Batch size	*Training times & Accuracy*
1	`6000/6000 [==============================] 98s 2ms/step - loss: 0.5236 - accuracy: 0.8138`
10	`6000/6000 [==============================] - 9s 2ms/step - loss: 0.4791 - accuracy: 0.8254`
100	`600/600 [==============================] - 2s 3ms/step - loss: 0.5185 - accuracy: 0.8201`
1000	`60/60 [==============================] - 1s 14ms/step - loss: 0.8456 - accuracy: 0.7302`
60000	`1/1 [==============================] - 0s 2ms/step - loss: 2.3579 - accuracy: 0.1107`

Plots :





We can see that with a batch size of 60000, even if the training time is almost instantaneous, the accuracy is not acceptable, so the batch size is way too large. From our observations, we can see that the **batch size of 100 seems to be the best one**. Indeed, we have the best compromise accuracy/training time, compared to the other ones.