# Project #2 : CNN architecture implementation

In this project we explore CNN architectures implementations using the PyTorch framework. For the whole project we utilize the CIFAR-10 dataset which consists of 6000 pictures for each of the 10 classes. First, we are going to train a VGG-16 model using an implementation that was already made. Secondly, we are going to fill in a skeleton code of an implementation of ResNet-50 and train a model for one epoch.

**VGG-16 implementation**

Using a pre-trained model (for 250 epochs) and an implementation of VGG-16, let’s train it one more time to observe the results. The training was made using the cross-entropy loss with the Adam optimizer.

A black screen with white text

Description automatically generatedThe training for 1 epoch took around 10 minutes (using a Macbook CPU). We observe that the accuracy of the model is 86% which is pretty good.

**ResNet-50 implementation**

Question 1 : Residual block

First, to implement the residual block we had to distinguish whether the residual block was doing down sampling or not. If it does, we must set the stride amount to 2 in the first 1x1 convolution layer. For the other layers we can just set strides and padding so the spatial size of the feature map doesn’t change. For the other type of residual block (without down sampling), we simply set stride and padding so the spatial size doesn’t change: for the 1x1 convolution let’s put stride 1 padding 0 and for the 3x3 convolution, stride 1 padding 1.

Question 2 : ResNet-50 architecture implementation

Finally, to implement the entire architecture of ResNet-50 we will refer to the table on the page 14 of the instruction sheet. Using the formula we can find the right padding and stride values to respect the layer architecture and the output size values at the end of each layer described on the table. For the average pooling layer, we know that the output of the last convolution layer is a 2x2x1024, since we want to put the feature map in a fully connected network, we need a vector. Therefore, we must make an average pooling with a kernel size of 2 to get a 1x1x1024 tensor. At the end, the FC layer must give us 1x1x10 tensor (corresponding to the number of classes) to achieve the classification.

A black screen with white text

Description automatically generated We can now train a pre-trained model (which was already trained for 285 epochs) for one more epoch. The training took approximately an hour (including the evaluation step). We get an accuracy on the test set of 82%, which was predicted from the beginning.