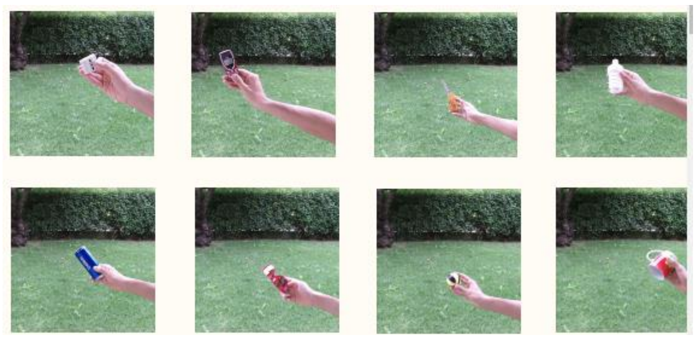
Homework 1 – Fabio Corallo

Image classification

The objective of this work is to train a deep learning model to perform image classification. The dataset consists of 8 different classes and each class consists of 200 images. The first-class images have the same background as the test images. So, I will create a neural network that can recognize the class 00 background and remove it (using GRADCAM) and later I will be able to use the new class 00 images (with a black background) for a classification using a new neural network.

Samples from train set:

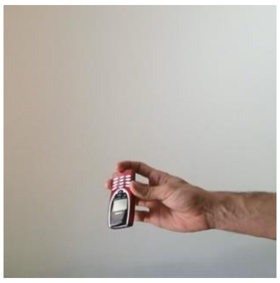
Samples from test set:



DATASET PREPARATION

We start the preparation of the dataset by cutting all the images using the coordinates of the object positions.

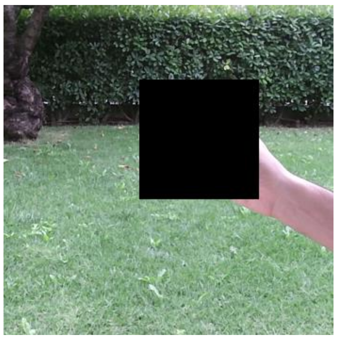
Original dimension: 350x350



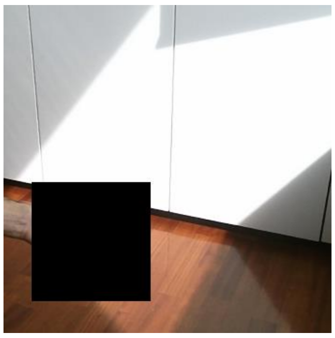
New dimension: 117x117

Later we will cut out the original images of classes 00 and 04 using the coordinates of the object positions so as to obtain only the backgrounds (the parts of the image containing the object will be colored black).

Background class 00:



Background class 04:



For all models I divided the images into train set and test set with sizes of 80% and 20% respectively. As optimizer I will use the stochastic gradient descent (SGD) with a learning rate equal to 0.01 and as loss function I will use the nn.CrossEntropyLoss() function which computes the cross-entropy loss between the predicted and actual class labels.

For the first model I use a batch size of 8 images and for the second model 32 images.

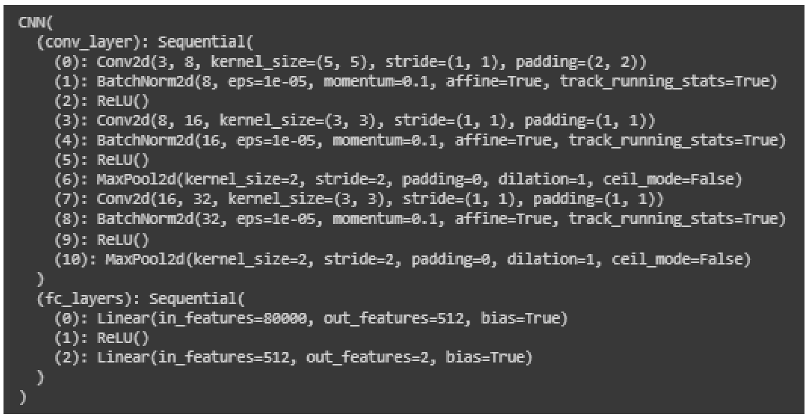
First neural network

Now that we have the backgrounds for classes 00 and 04, let's train a neural network to distinguish between the two types of background. In this model the background of class 00 will be class 0 and the background of class 04 will be class 1.

The first model is a Convolutional Neural Network (CNN) which takes 3-channel images (RGB) as input and produces a binary classification as output.

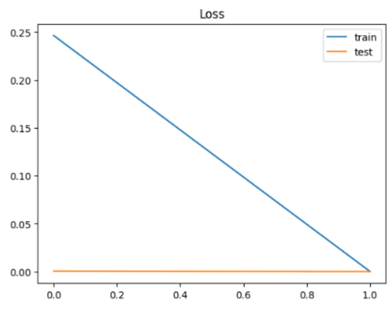
The network consists of two main parts: the first part consists of a sequence of convolutional, batch normalization and ReLU activation layers. This part is used to extract the main features of the image and reduce its dimensionality. In particular, the first convolution uses a 5x5 kernel with 8 activation maps and 2 pixels padding to keep the same size as the input image. The second convolution uses a 3x3 kernel with 16 activation maps and 1-pixel padding, while the third convolution uses a 3x3 kernel with 32 activation maps and 1 pixel padding. After each convolution, a batch normalization is applied to improve model stability, followed by a ReLU activation function to introduce non-linearities. All batch normalizations use an epsilon of 1e-05 and a momentum of 0.1. Finally, there are two MaxPooling layers with kernels of size 2x2 and strides of 2, which further reduce the dimensionality of the image.

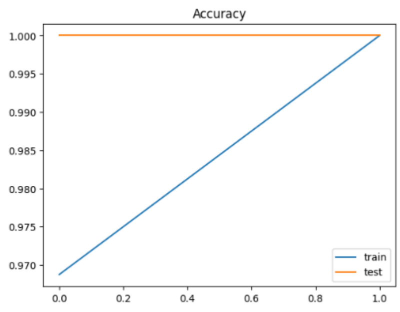
The second part of the network consists of two fully connected layers (FC), i.e., a ReLU activation layer and an output layer with two neurons (one for each class). The FC layers receive input feature data extracted from the first part of the network, which is flattened into a vector with 80000 elements before being passed to the initial fully connected layer. The output layer uses a linear activation function to produce the probability of belonging to one of the two classes.



```
CNN(
  (conv_layer): Sequential(
    (0): Conv2d(3, 8, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Conv2d(8, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (7): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (9): ReLU()
    (10): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc_layers): Sequential(
    (0): Linear(in_features=80000, out_features=512, bias=True)
    (1): ReLU()
    (2): Linear(in_features=512, out_features=2, bias=True)
  )
)
```

The model had to perform a simple classification in two classes, and it only takes 1 or 2 epochs to get perfect results.





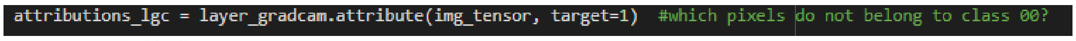
Background elimination in class 00

Now that we have a model that can recognize a background of class 00 and a background of class 04, I'm going to use it to classify which pixels in the original images of class 00 don't belong to background 00.

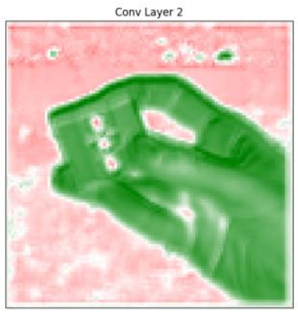
So i will ask the model which pixels belong to class 1 (class 1 is background of class 04 of original images)

Original image:

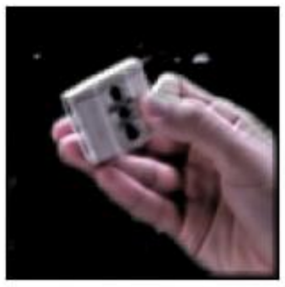
Which pixels do not belong to class 00?



```
attributions_lgc = layer_gradcam.attribute(img_tensor, target=1)  #which pixels do not belong to class 00?
```



And therefore, I will get the following excellent result:



So, for each image of class 00 I remove the background like this.

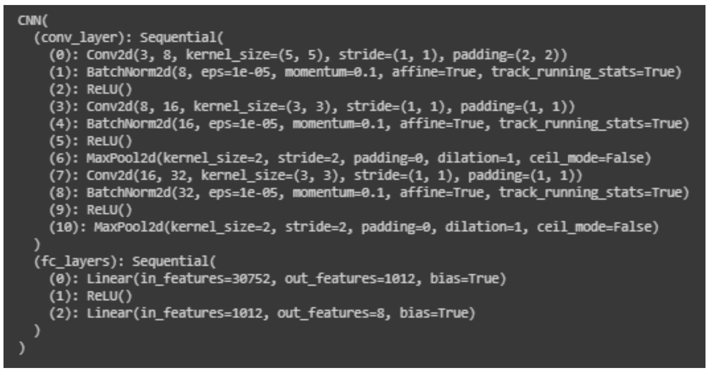
To make sure that every background of class 00 is not the same color (black) I also tried using different colors, but the results weren't any better, so I didn't use the images below.

Second neural network

The second neural network is then created to classify the original images into their 8 classes.

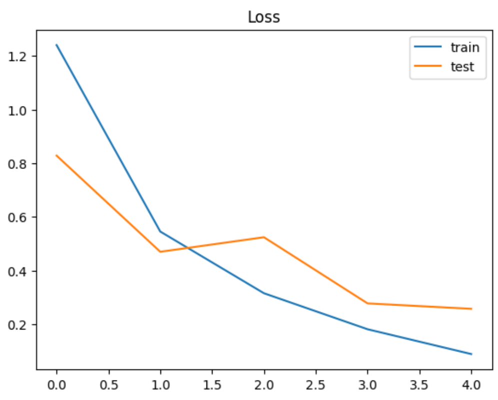
The network consists of two main parts: the first part consists of a sequence of convolutional, batch normalization and ReLU activation layers. This part is used to extract the main features of the image and reduce its dimensionality. In particular, the first convolution uses a 5x5 kernel with 8 activation maps and 2 pixels padding to keep the same size as the input image. The second convolution uses a 3x3 kernel with 16 activation maps and 1-pixel padding, while the third convolution uses a 3x3 kernel with 32 activation maps and 1 pixel padding. After each convolution, a batch normalization is applied to improve model stability, followed by a ReLU activation function to introduce non-linearities. All batch normalizations use an epsilon of 1e-05 and a momentum of 0.1. Finally, there are two MaxPooling layers with kernels of size 2x2 and strides of 2, which further reduce the dimensionality of the image.

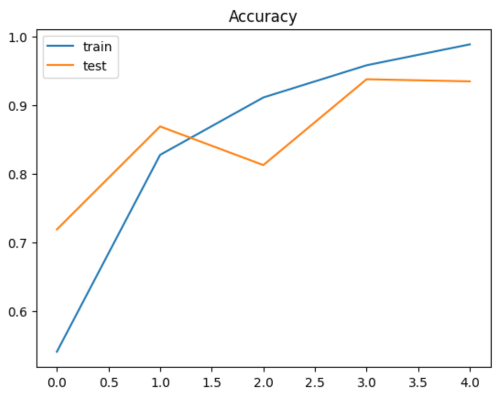
The second part of the network consists of two fully connected layers (FC), i.e., a ReLU activation layer and an output layer with eight neurons (one for each class). The FC layers receive input feature data extracted from the first part of the network, which is flattened into a vector with 30752 elements before being passed to the initial fully connected layer. The output layer uses a linear activation function to produce the probability of belonging to one of the eight classes.



```
CNN(
  (conv_layer): Sequential(
    (0): Conv2d(3, 8, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Conv2d(8, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (7): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (9): ReLU()
    (10): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc_layers): Sequential(
    (0): Linear(in_features=30752, out_features=1012, bias=True)
    (1): ReLU()
    (2): Linear(in_features=1012, out_features=8, bias=True)
  )
)
```

In 5 epochs I obtained a Test loss equal to 0.2552, and a Test accuracy equal to 0.9313.





Finally in classifying the images for the competition I got an accuracy of 0.6825.