ARTIFICIAL INTELLIGENCE AND COGNITIVE COMPUTING

CHALLENGE CCAIUNICT-2023

Cognitive Computing and Artificial Intelligence

University of Catania, Italy

Project by:

- Roggio M. (m. 1000037341)
- Vinciguerra E. (m. 1000042241)

2022-2023

OUTLINE

Introduction

The DataSet

Transformations

VGG

Our CNN

Introduction

The following project concerns the realization of a CNN model for image classification.

The dataset used is CCAIUnict-2023, available on the Kaggle challenge.

THE DATASET

The DataSet

The DataSet

In order to obtain the element, target and element name for each sample available in the dataset, we used ***ImageFolderNames()*** class, which is an override of ***ImageFolder()*** class (as it does not return the element name).

```python
import random
from torchvision.datasets import ImageFolder

class ImageFolderNames(ImageFolder):
    def __init__(self, root, split, transform=None, target_transform=None):
        super().__init__(root, transform=transform, target_transform=target_transform)
        self.testSample=[]
        self.trainSample=[]

        for name in self.samples:
            if name[0].split('/')[-2]=="test":
                self.testSample.append(name)
            else:
                self.trainSample.append(name)

        if split=="test":
            self.samples=self.testSample
        else:
            self.samples=self.trainSample

        random.shuffle(self.samples)

        self.transform=transform

    def __getitem__(self, index):
        path, target = self.samples[index]
        image = self.loader(path)
        if self.transform:
            image=self.transform((image, target, path.split('/')[-1]))
        return image, target, path.split('/')[-1]
```

TRANSFORMATIONS

Transformation

Transformations used

It was necessary to use some transformations on the images constituting the dataset, on both the training and the test portion the following modifications were made:

- ***FileCrop(root, «train»)***: Class created by us, crops the image as specified in the files («train.csv» or «test.csv»);
- ***T.Resize(size)***: Resizes the image;
- ***T.ToTensor()***: Transforms to tensor.

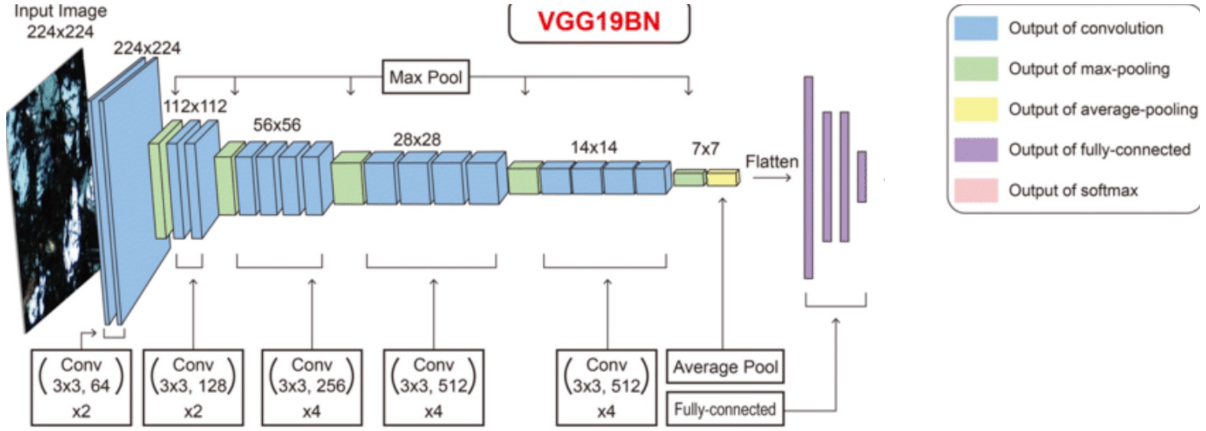
In addition, the following transformation was inserted in the training phase:

- ***T.RandomHorizontalFlip(p=0.5)*:** Applies a random horizontal flip with 50% probability.

Further transformations such as normalisation or colour perturbation were considered, however, no positive results were obtained.

VGG

VGG



VGG

VGG is a valid and widely used convolutional neural network in image classification problems.

Some key points distinguish this network from others, including:

- ***Simplicity***: Unlike other networks (such as ResNet), VGG's architecture is not complex; in fact, it is relatively simple to understand. This simplicity does not affect the overall performance; on the contrary, VGG's results in terms of accuracy are comparable to those of much more complex networks;
- ***Reduction of input size***: The use of 3x3 kernels and pooling layers gradually reduces the size of the data, which is advantageous in terms of computational resources;
- ***Batch Normalisation (BN)***: This technique normalizes the data and accelerates model convergence during training; it can also help mitigate vanishing gradient problems. It introduces a normalization of the data within each training batch, reducing the dependence of the network on the specific values of individual examples in the batch.

In this project, we took eight layers from the VGG19BN model and used them in combination with a model of our own creation.

OUR CNN

Our CNN

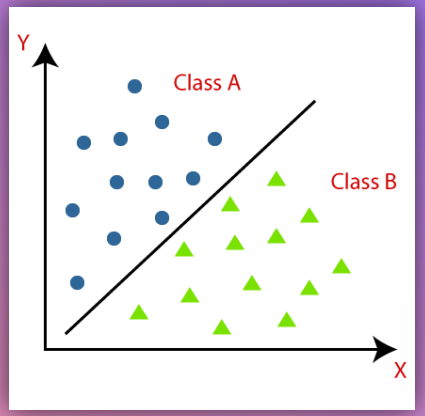
Our CNN

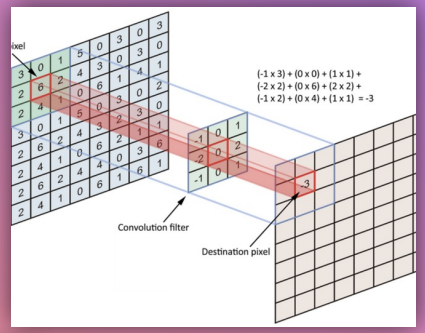
As described previously, eight layers of the VGG19BN model were taken and included in a model of our own creation, which performs convolution and pooling operations.

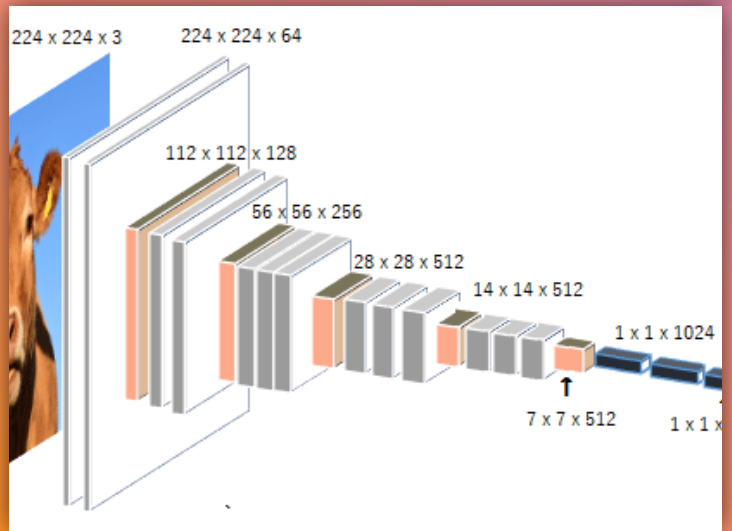
An accuracy of 72.725% was found during training.

```
torch.Size([1, 3, 150, 150])
Model output size: torch.Size([1, 8])

Model(
  (vgg): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (7): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  )
  (conv_layers): Sequential(
    (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(3, 3))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1))
    (4): ReLU()
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(256, 512, kernel_size=(2, 2), stride=(1, 1))
    (7): ReLU()
    (8): Conv2d(512, 1024, kernel_size=(2, 2), stride=(1, 1))
    (9): ReLU()
    (10): MaxPool2d(kernel_size=2, stride=1, padding=0, dilation=1, ceil_mode=False)
  )
  (fc_layers): Sequential(
    (0): Linear(in_features=4096, out_features=1024, bias=True)
    (1): ReLU()
    (2): Linear(in_features=1024, out_features=8, bias=True)
  )
)
```







THANK YOU

Roggio M. (m. 1000037341)

Vinciguerra E. (m. 1000042241)