UNIVERSITÀ DEGLI STUDI DI MILANO
FACULTY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

Master in
Computer Science

STATISTICAL METHODS FOR MACHINE LEARNING
FINAL REPORT ON NEURAL NETWORKS FOR THE BINARY CLASSIFICATION

Teacher: Prof. Nicolò Cesa-Bianchi

Final report written by:
Samuele Simone
Matr. Nr. 11910A

Academic year 2022-2023

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study.

Contents

Chapter 1

Introduction

The purpose of this project was to discover and work on different neural network models in order to solve an image classification problem. Specifically the recognition of chihuahuas from muffins, which because of the similarities, turns out to be a non-trivial task for a machine to perform. In the following chapters we will go into detail about the various models and through experiments and graphs figure out which model will work best to solve this task.

Chapter 2

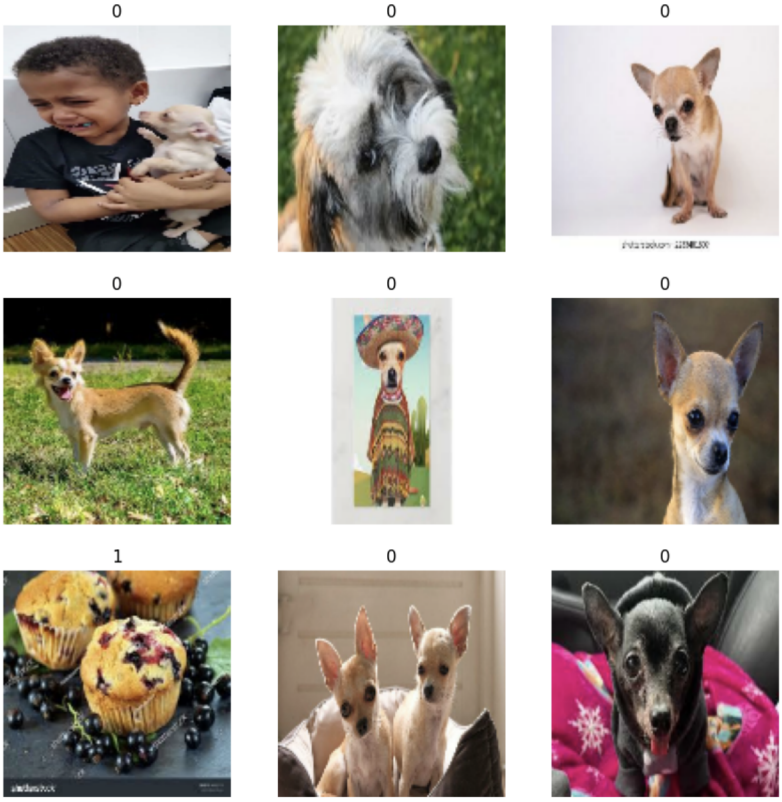
Dataset

The dataset under consideration is called "Muffin vs Chihuahua", taken from Kaggle **??**. It consists of about 6000 images taken from Google Images. Duplicate images have been removed. The first operation that was performed was to download the dataset through the Kaggle API and divide the dataset into training set and test set. After that, thanks to the Tensorflow library and specifically Keras, the images were loaded so that the neural networks could be trained. Here the code for the train_images loading using `tf.keras.utils.image_dataset_from_directory`

```
train_images = tf.keras.utils.image_dataset_from_directory(
    train_folder,
    labels="inferred",
    label_mode="int",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False,
)
```

As you can see there are different properties such as `label_mode = int`, so labels as integers. Or `color_mode ="rgb"` in which the images are working in this red blue green space. By making these parameters explicit, it is possible to obtain a more custom configuration that suits our needs. The same process was applied for the test_images. To give an idea to the reader, here a sample of images found within the dataset: As you can see muffin label is 0 and chihuahua label is 1.

Figure 1: A sample of images found within the dataset



2.1 Preprocessing

Certainly when dealing with data it is good to always make some improvement such as cleaning the data so that you have better results. In fact, the dataset was delivered without having duplicate images, which already represents a first step of preprocessing. In addition, I applied the **Data augmentation** [1] that is a technique in machine learning used to reduce overfitting when training a machine learning model, by training models on several slightly-modified copies of existing data. With the help of Keras, it is possible to achieve date augmentation in the following way:

```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
    ]
)
```

Here is graphically what the transformation looks like: In order to complete the

Figure 2: Data augmentation applied in a specific image

preprocessing I applied the data augmentation transformation to the input and I

performed normalization of pixel values in the range [0, 255] to values in the range [0, 1], indeed this helps stabilize the training of the model. To conclude, this line `train_images = train_images.prefetch(tf.data.AUTOTUNE)` optimizes the data loading process during model training, allowing the model to work more efficiently and reducing the waiting time between data batches.

Chapter 3

Neural Network Models

In this chapter we are going to examine the models that have been made in order to solve the binary classification task. All models were addrested using the principle of K-Fold cross validation. In this case the `K = 5` .

3.1 K-fold cross validation (K = 5)

3.2 Multi-Layer Perceptron (MLP)

A multilayer perceptron (MLP) is a feedforward artificial neural network, consisting of fully connected neurons with a nonlinear kind of activation function, organized in at least three layers, notable for being able to distinguish data that is not linearly separable. [2] During the project, I developed 3 different MLP models so that I could conduct different experiments and understand the performance of them.

3.2.1 First MLP Model

Here are the technical details of this setup:

- `inputs = tf.keras.Input(shape=input_shape)`: model's input with the input_shape size. In this case (128, 128, 3)
- `x = layers.Flatten()(inputs)`: Converts 2D input (an image) to a 1D vector
- `x = layers.Dense(256, activation='relu')(x)`: Fully connected layer (dense) with 256 neurons and ReLU activation

- `x = layers.Dropout(0.5)(x)`: This dropout layer introduces regularization into the network. The parameter 0.5 indicates that 50% of the neurons exiting this layer are randomly switched off during training.
- `x = layers.Dense(128, activation='relu')(x)`: Other fully connected layer with 128 neurons and ReLU activation.
- `x = layers.Dropout(0.5)(x)`: Another dropout layer for further regularization.
- `activation = 'sigmoid'`: Here the activation function for the last layer is specified. In your case, you are using 'sigmoid,' which is commonly used in binary classification problems where the output must be a probability between 0 and 1 for each class.
- `outputs = layers.Dense(num_classes, activation=activation)(x)`. num_classes = 1. This is the last layer of the model, which returns the output.

3.3 Convolutional Neural Network (CNN)

3.4 Deep Residual learning (ResNet50)

Bibliography

[1] Wikipedia. Data augmentation. https://en.wikipedia.org/wiki/Data_augmentation, 2023.

[2] Wikipedia. Multilayer perceptron. https://en.wikipedia.org/wiki/Multilayer_perceptron, 2023.