# ML - Hw2

Eduardo Rinaldi 1797800

December 2020

## Contents

## 1 Introduction

This is my report about the second homework assigned for the course of Machine Learning. This time the topic is about **Convolutional Neural Networks** (CNN) and image classification.

### 1.1 Assignment

The assignment was about solving a classification problem on images on the following classes:

- Angle Brooms
- Chopsticks
- Oatmeal Box
- Plastic tray
- Plums
- Potato chips bag
- Rice bowl
- Soft drink bottles

so the expected output is an array of 8 elements, where each element is a probability that the image is in that class (so at the end there'll be a **softmax** function).

### 1.2 Tools used

Tools and library used for this homework are:

- **Python 3**: because of its simplicity and its support on the next libraries and technology
- **Tensorflow**: because it's one of the main libraries available for NNs (so large support on the web and good documentation) and it support Nvidia CUDA technology.
- **Numpy**: this library has been used not that much on the dataset, but it has been very useful to use it with Matplotlib.
- **Matplotlib**: used for plotting some data about training history or final results.
- **CUDA Technology**: very useful for speeding up the training process.

## 2 Dataset

The dataset provided is the **RoboCup@Home-Objects** that has been developed within the **RoboCup@Home** competition.

Each element of the dataset is an image (each of which can have different shape) for a total of 8311 images. As said before each image is labeled as one of the 8 classes already presented.

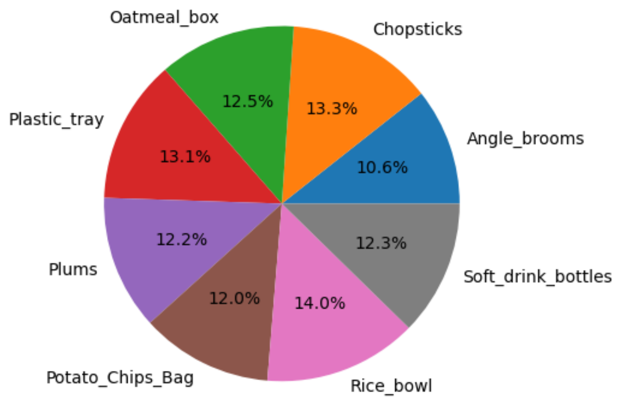


Figure 1: Pie chart that shows how the dataset it's divided

As we can see from figure [1], luckily the dataset is pretty balanced so we don't need to implement any strategy for balancing it.

The dataset has been divided in two different sets:

1. Training set (67%)
2. Validation set (33%)

### 2.1 Preprocessing

For the preprocessing phase, two main operations are carried out:

- **Normalize RGB values**: as good practice it's better to scale rgb values from a big range like $[0, 255]$ to a range like $[0, 1]$
- **Dealing with different shapes**: each image can have different shape with different aspect ratio, but we need a dataset where all the images have the same shape.

### 2.1.1 Normalize RGB values

To normalize these values we can simply map rgb from $[0, 256)$ range to $[0, 1]$ range using this function:

$$normalize(r, g, b) =< \frac{r}{255}, \frac{g}{255}, \frac{b}{255} > \tag{1}$$

### 2.1.2 Dealing with different shapes

As already mentioned, the images in the dataset are all of different sizes and therefore also have different aspect ratios, but to pass the input image to a convolutional layer we must somehow preprocess the images in such a way as to bring them all to the same resolution.

To do that I propose 2 different methods and at the end only the second one will be used for this project:

- The **first one** consists in simply reshaping the image in a common size for all the images. For example if we have an image 1280x720 (aspect ratio 16:9) and another image in 1024×768 (aspect ratio 4:3) we can reshape them in 512x512 resolution (aspect ratio 1:1). This is the simplest solution but also the worse, that because doing that we'll "stretch" the image and and we could lose proportion.
- The **second solution** consists in resizing an image to a target width and height by keeping the aspect ratio the same without distortion. If the target dimensions don't match the image dimensions, the image is resized and then padded with zeroes to match requested dimensions.

  This process can be easily done using Tensorflow function `tf.image.resize_with_pad`

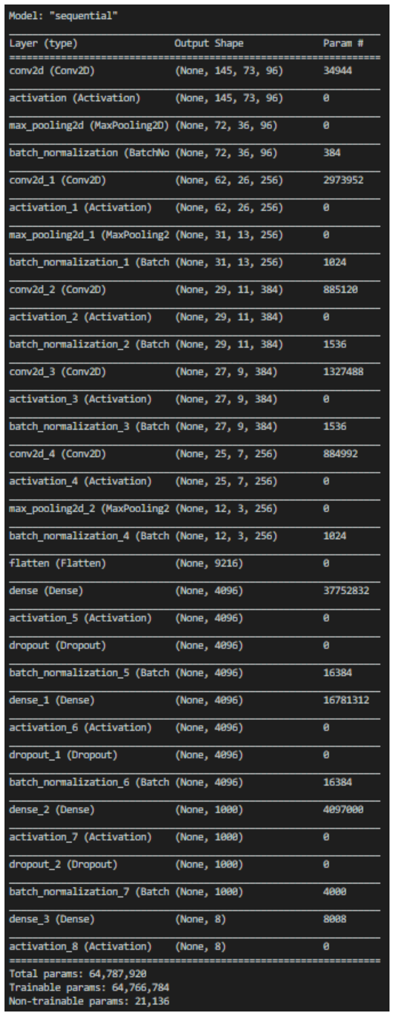
## 2.2 Data Augmentation

In order to improve the performance and ability of the model to generalize, a process of data augmentation has been done applying the following geometric transformation to each image:

- Horizontal flip
- Vertical flip
- Random rotation
- Zoom
- Width or height shift

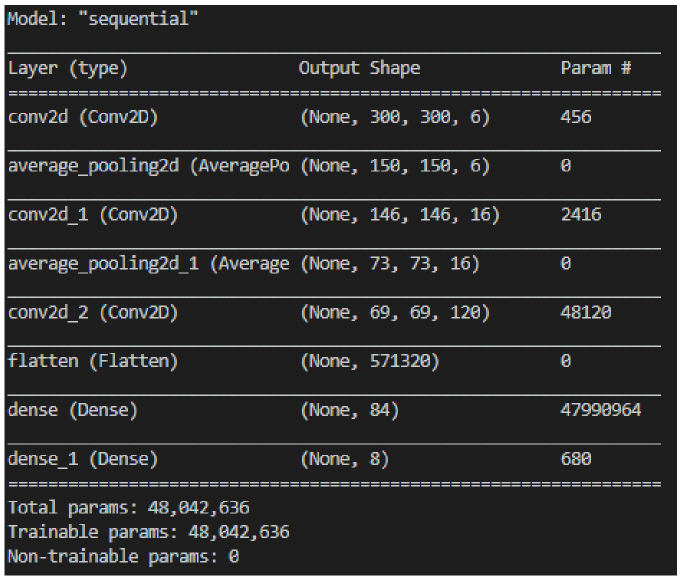
## 3 Network selection

As for the first homework also in this I decided to implement, test and compare different approaches. In particular in this case I decided to use two neural network models which are inspired by **LeNet** and **AlexNet** models.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 145, 73, 96)       34944
_________________________________________________________________
activation (Activation)      (None, 145, 73, 96)       0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 72, 36, 96)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 72, 36, 96)        384
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 62, 26, 256)       2973952
_________________________________________________________________
activation_1 (Activation)    (None, 62, 26, 256)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 31, 13, 256)       0
_________________________________________________________________
batch_normalization_1 (Batch (None, 31, 13, 256)       1024
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 29, 11, 384)       885120
_________________________________________________________________
activation_2 (Activation)    (None, 29, 11, 384)       0
_________________________________________________________________
batch_normalization_2 (Batch (None, 29, 11, 384)       1536
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 27, 9, 384)        1327488
_________________________________________________________________
activation_3 (Activation)    (None, 27, 9, 384)        0
_________________________________________________________________
batch_normalization_3 (Batch (None, 27, 9, 384)        1536
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 25, 7, 256)        884992
_________________________________________________________________
activation_4 (Activation)    (None, 25, 7, 256)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 3, 256)        0
_________________________________________________________________
batch_normalization_4 (Batch (None, 12, 3, 256)        1024
_________________________________________________________________
flatten (Flatten)            (None, 9216)              0
_________________________________________________________________
dense (Dense)                (None, 4096)              37752832
_________________________________________________________________
activation_5 (Activation)    (None, 4096)              0
_________________________________________________________________
dropout (Dropout)            (None, 4096)              0
_________________________________________________________________
batch_normalization_5 (Batch (None, 4096)              16384
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
activation_6 (Activation)    (None, 4096)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
batch_normalization_6 (Batch (None, 4096)              16384
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              4097000
_________________________________________________________________
activation_7 (Activation)    (None, 1000)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 1000)              0
_________________________________________________________________
batch_normalization_7 (Batch (None, 1000)              4000
_________________________________________________________________
dense_3 (Dense)              (None, 8)                 8008
_________________________________________________________________
activation_8 (Activation)    (None, 8)                 0
=================================================================
Total params: 64,787,920
Trainable params: 64,766,784
Non-trainable params: 21,136
```

Figure 2: AlexNet



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 300, 300, 6)       456
_________________________________________________________________
average_pooling2d (AveragePo (None, 150, 150, 6)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 146, 146, 16)      2416
_________________________________________________________________
average_pooling2d_1 (Average (None, 73, 73, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 69, 69, 120)       48120
_________________________________________________________________
flatten (Flatten)            (None, 571320)            0
_________________________________________________________________
dense (Dense)                (None, 84)                47990964
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 680
=================================================================
Total params: 48,042,636
Trainable params: 48,042,636
Non-trainable params: 0
```

Figure 3: LeNet

### 3.1 Input shape

Two different image ”target” resolutions were chosen for the two models:

- LeNet: 256x256
- AlexNet: 118x224

### 3.2 Metrics and optimizers

For both models I decided to use the classic metrics:

- Accuracy (precisely categorical accuracy)
- Precision
- Recall

The optimizer that initially was used is SGD, but the models obtained better results using **Adam**, so I decided to switch to it.

## 4 Results

Once the models were defined, I started several training phases on 12 epochs with different "settings".
For example, the first training phases were carried out with very low resolutions (64x64), and it was possible to see how doubling or even tripling the number of pixels greatly increases the accuracy of the models (and in fact I decided to increase the resolution accordingly).
Finally, the two models were trained for 50 epochs with the best settings identified, so that we could identify which was also the best number of epochs to train the model. Final results on validation set are reported in the next two sections.

### 4.1 LeNet

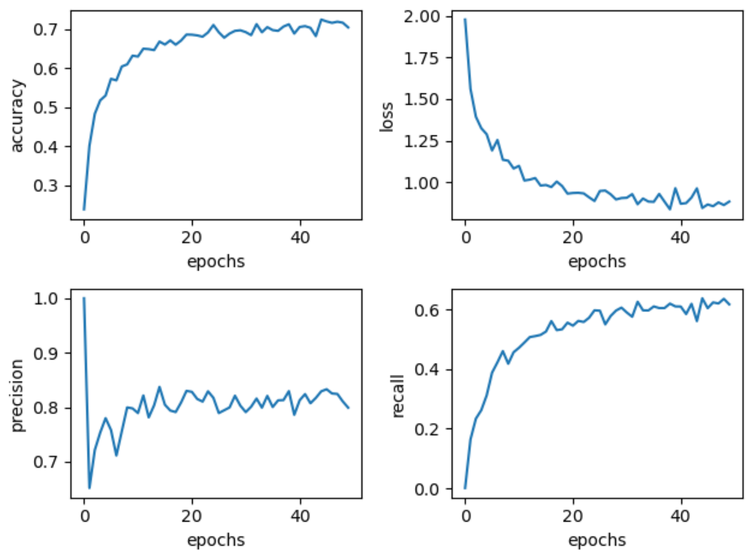


Figure 4: Training history, metrics calculated on validation set at each epoch

- Accuracy: 0.7099
- Precision: 0.8007
- Recall: 0.62

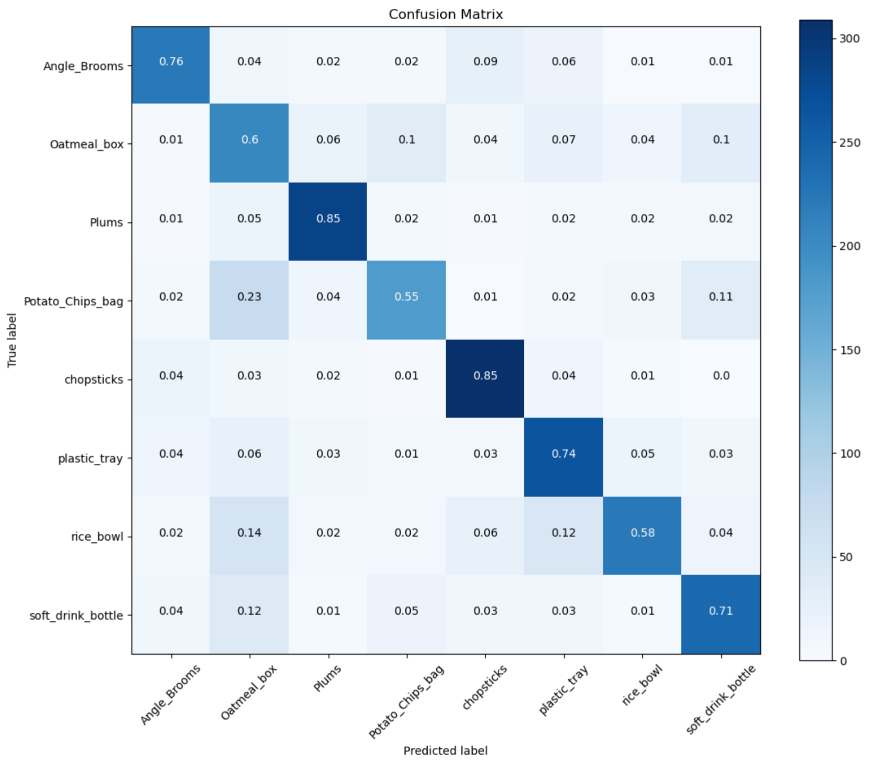
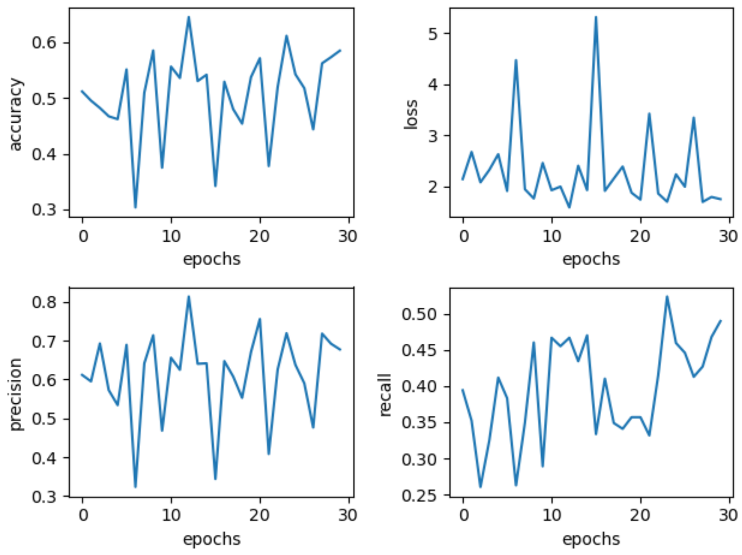


Figure 5: Confusion matrix (normalized) for LeNet model

## 4.2 AlexNet



inin

Figure 6: Training history, metrics calculated on validation set at each epoch

- Accuracy: 0.5832
- Precision: 0.6665
- Recall: 0.4858

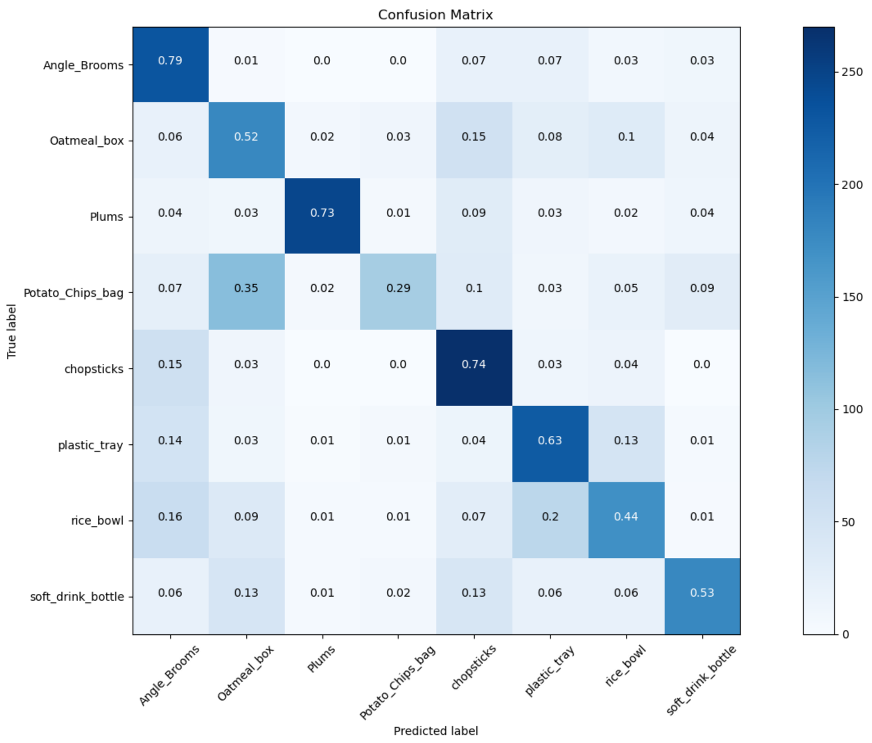


Figure 7: Confusion matrix (normalized) for AlexNet model

## 4.3 Final thoughts

As we can see from the results obtained, **LeNet is the one who performs better**. AlexNet had a strange training process with some high scores followed by lower scores (I also tried using 1:1 aspect ratio on AlexNet but it not improved results).
LeNet instead achieved good scores and its learning process was very constant.

I also tested the two models on a validation set without image editing (h/v flip, rotation, etc..) and LeNet's results were very high.

I noticed that there are a lot of ”noise” images in the dataset as they are not labeled correctly. A possible improvement of these two models could involve filtering these images (also through another already trained classifier with high confidence) so as to further improve the training of the two networks.

## 5 Professor notes

If needed, the professor can write it's note here on this page.