Homework 2 - Venice Boat Classification

Matteo Canavicci - 1548240

December 30, 2018

Contents

1 Introduction

Detection and classification of vehicles are fundamental processes in monitoring and security systems. In this report we focus on a particular catgory of vehicles : boats. Monitoring a maritme scene implies a series of challenges related for example to the water background, reflections or to the light in the images. All this features of the image make the problems of boat detection and classification an hard work.
Our work is focussed on boat images automatically extracted by the ARGOS system (Automatic and Remote GrandCanal Observation System) operating 24/7 in Venice. The system automatically extracts information on the traffic flow and density and it highlights the illegal behaviour of the boats driver. ARGOS controls a water canal of about 6 km length, with a width between 80 and 150 meters. Cameras are connected to a dedicated computer where high resolution color images are acquired and processed.
ARGOS provides the following main functionalities:

- Optical detection and tracking of moving targets present in the field of view of each survey cell;
- Computing position, speed, and heading of any moving target observed by a cell;
- Automatic detection of a set of predefined events;
- Transmission of data and video stream to the Control Center.

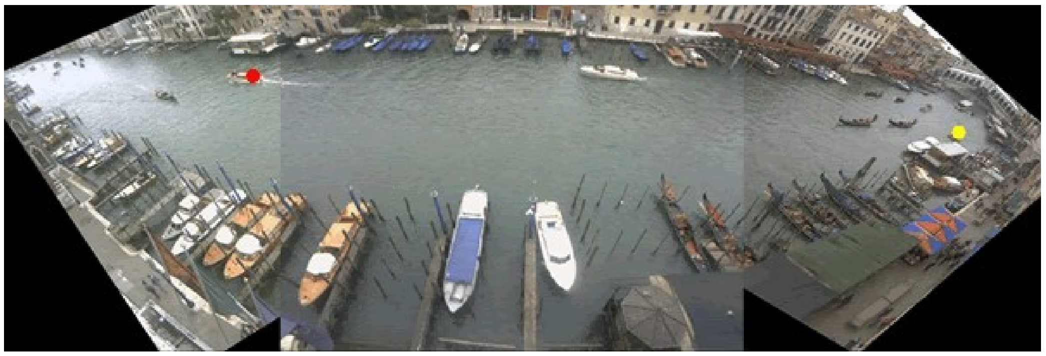

Figure 1: Example of snapshot shooted by one of the ARGOS cameras

2 The problem

Given the dataset provided by MarDCT (Maritime detection, classification and tracking dataset) of the Venice boats images, build an image classifier to classify the boats according to the type of boat. To solve the problem it has been used a convolutional neural network approach with preprocessing of the images.
After building multiple models with different parameters in the convolutional neural network, the model has been tested on a validation dataset still provided by MarDCT.

2.1 Used tools

The programming language used is Python, the main libraries and tools used are:

- Tensorflow - Documentation
- Keras - Documentation
- Numpy - Documentation
- Scikit Learn - Documentation
- Cv2 to work with images - Documentation
- Matplotlib - Documentation

The hardware used to train the network is the gpu GTX 1050 Ti.

3 The Dataset

The data sets are composed by images (800 x 240 pixels) acquired as described in the introduction. The dataset is composed of two main folders, one for training with the following structure:

```
train
├── Alilaguna
├── Ambulanza
├── Barchino
└── Gondola
```

```
│
├── Lanciafino10m
├── Lanciafino10mBianca
├── Lanciafino10mMarrone
├── LanciaMaggioredi10mBianca
├── Motobarca
├── Motopontonerettangolare
├── MotoscafoACTV
├── Mototopo
├── Patanella
├── Polizia
├── Raccoltarifiuti
├── Sandoloaremi
├── Topa
├── VaporettoACTV
└── Water
```

The other folder has been used fot testing and validation. In the train folder there are 19 folders named with the name of the class the boat image belongs to with totally 4712 images. The original datase was composed of 24 folders, but some of this classes are not contained in the test folder, so I preferred to make a 19 classes classification. In the water folder there are images with snapshots of only water without boats.
For testing has been used the test folder that contains 1969 images, not all of them has been used for testing because some images didn't belong to any classes fo the train set, moreover there was some images not labeled in the ground_truth.txt file. All this kind of images have been discarded during tests.
In the file called ground_truth.txt still provided by MarDCT there are the name of files in test folder labeled with the class that image belongs to. The file has been cleaned: labels has been cleaned removing punctuation and spaces; in the new file (new_ground_truth.txt), the structure of the file is for each line:

$$< image\_name : label >$$

where the name and the label are separated by a tab.
In the cleaning of the dataset similar labels in the ground truth has been grouped (like Mototopo and Mototopo corto have been grouped in Mototopo) and the label "Snapshot Acqua" has been translated in "Water" to match

the train folders.

4 Convolutional Neural Network Architectures

To build the model I choose a CNN approach, building some variations of different type of CNNs and testing them with different parameters. In particular it has been used two architectures of CNNs, modified according to the requirements of our problem. To train the network has been used also the image augmentation to get a larger amount of training data. Image augmentation artificially creates training images through different ways of processing or combination of multiple processing, such as random rotation, shifts,flips, brightness etc.

4.1 Modified VggNet

The original VggNet consists of 16 convolutional layers, for our problem not all of them are needed, so the architecture has been modified.
The architecture constists of 11 layers:

1. A convolution layer with 32 filters, the activation function "relu" + maxpooling + droput to limit the overfitting.
2. Two layers with 64 filters + activation function + maxpooling + dropout
3. Two layers with 128 filters + activation function + maxpooling + dropout
4. Two layers with 256 filters + activation function + maxpooling + dropout
5. Two layers with 512 filters + activation function + maxpooling + dropout
6. Two fully connected layers with 1024 nodes
7. Output layer with Softmax activation function with 19 nodes.

First the network has been trained using images in the train set folders and the loss and accuracy of the network has been computed using the test set. Then I tried to build a model splitting the train set. The 20% has been used

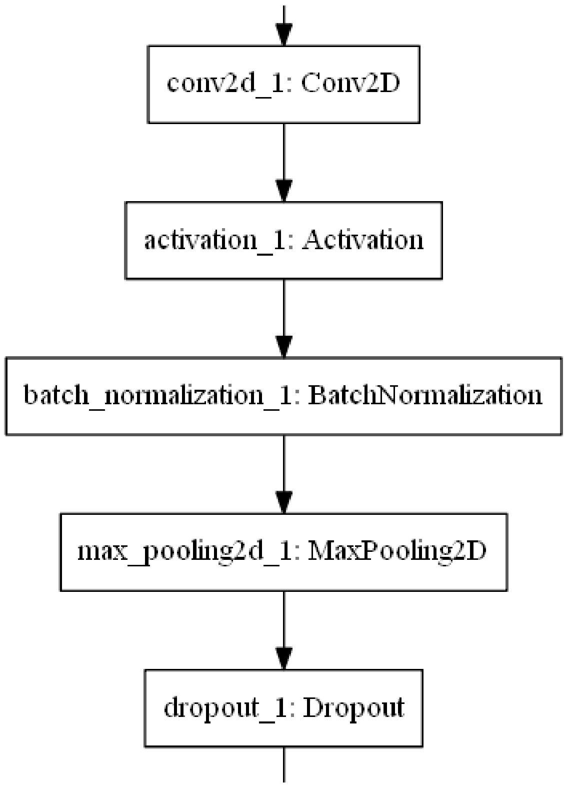
conv2d_1: Conv2D
activation_1: Activation
batch_normalization_1: BatchNormalization
max_pooling2d_1: MaxPooling2D
dropout_1: Dropout

Figure 2: First layer of modified VggNet

for test and the rest for train.

It follows the results obtained using as metrics for the evaluation accuracy, precision, recall and f1-score and using the images contained in the test folder except those images not labeled or whose class is not contained in the train set:

With the network trained with the train and test set:
$Accuracy = 0.86$
$Precision : 0.87$
$Recall : 0.86$
$F1 - score : 0.86$

With the network trained only with the splitted train set:
$Accuracy = 0.86$
$Precision : 0.86$
$Recall : 0.86$
$F1 - score : 0.85$

Notice that the results are quite similar, the second case is a bit worst probably because the train set is smaller of the one in the first case. We could reach better performances having a larger train set, this one of the way of improving performances of our classifier.

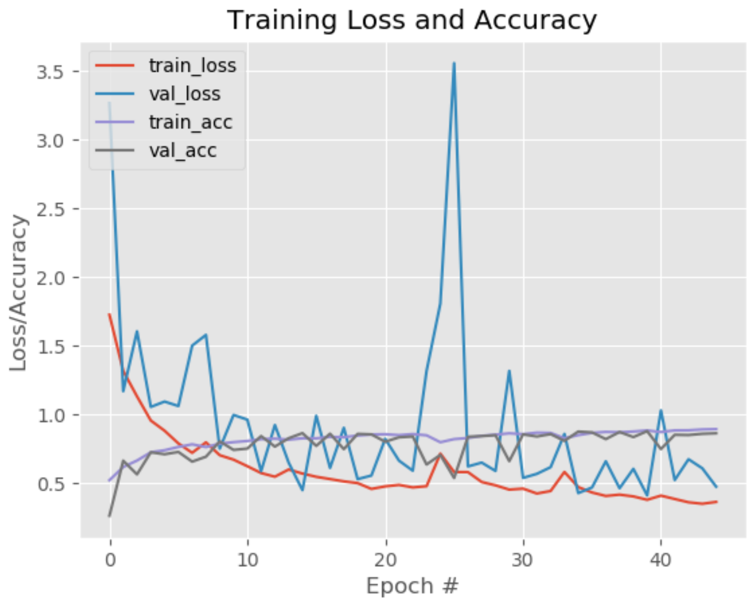

Figure 3: Training loss and accuracy of modified VggNet in the first case in 45 epochs

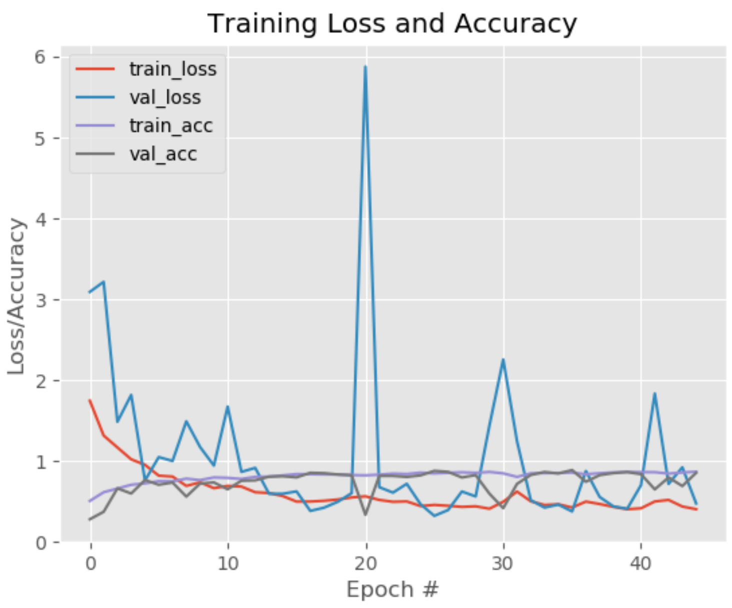

Figure 4: Training loss and accuracy of modified VggNet in the second case in 45 epochs

4.2 Modified AlexNet

AlexNet consists of 5 Convolutional Layers and 3 fully connected layers. The first two Convolutional layers are followed by the Overlapping Max Pooling layers. The third, fourth and fifth convolutional layers are connected directly. The fifth convolutional layer is followed by an Overlapping Max Pooling layer, the output of which goes into a series of two fully connected layers.

The architecture conists of:

1. One convolution layer with 96 filters having size 11x11 and a stride of 4 + the activation function ”relu” + maxpooling

2. One convolution layer with 256 filters having size 5x5 and a stride of 1 + the activation function ”relu” + maxpooling

3. Two convolution layers having size 3x3 and a stride of 2 + the activation

function "relu". The first two have 384 filters, the third uses 256 filters. The three layers are followed by a maxpooling layer.

4. Next there is two fully connected layers with 1024 filters.

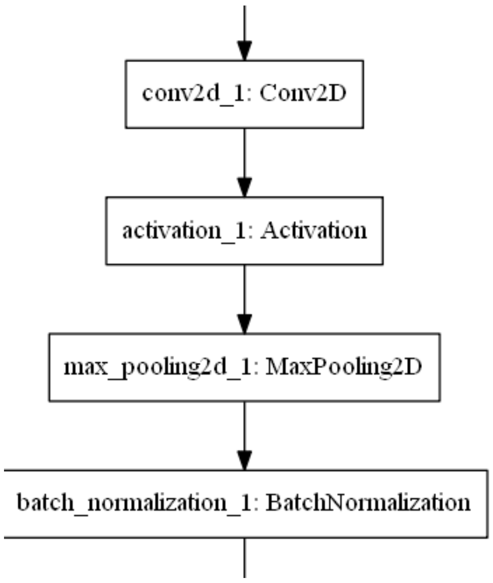
conv2d_1: Conv2D
activation_1: Activation
max_pooling2d_1: MaxPooling2D
batch_normalization_1: BatchNormalization

Figure 5: First layer of modified AlexNet

Like in the VggNet, it has been tested two cases: the first training the cnn with the whole train set and evaluating with the whole test set and the second case in which it has been splitted the train set: 20% for testing, 80% for training.

With the network trained with the train and test set:
$Accuracy = 0.86$
$Precision : 0.86$
$Recall : 0.86$
$F1 - score : 0.85$

With the network trained only with the splitted train set:
$Accuracy = 0.86$
$Precision : 0.86$
$Recall : 0.86$
$F1 - score : 0.85$

In this case the results are identical, also the performances of both the network's architecture are quite similar.

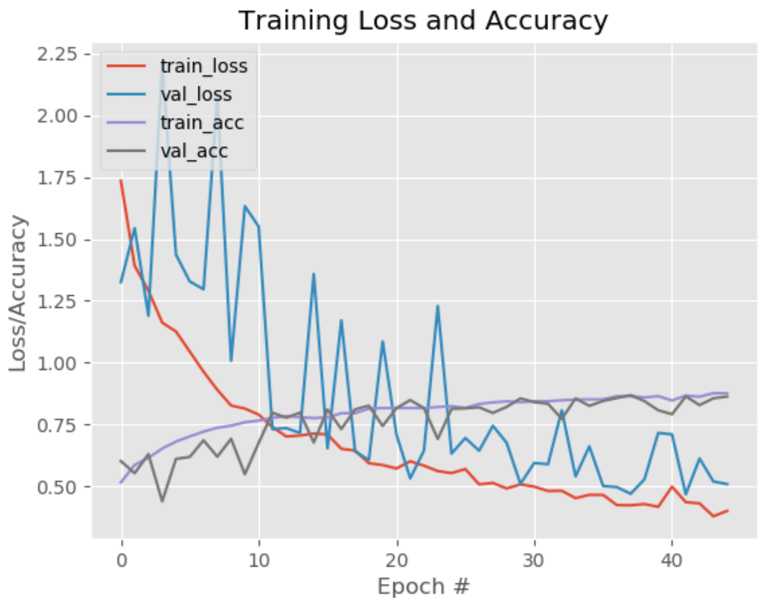

Figure 6: Training loss and accuracy of modified AlexNet in the first case in 45 epochs

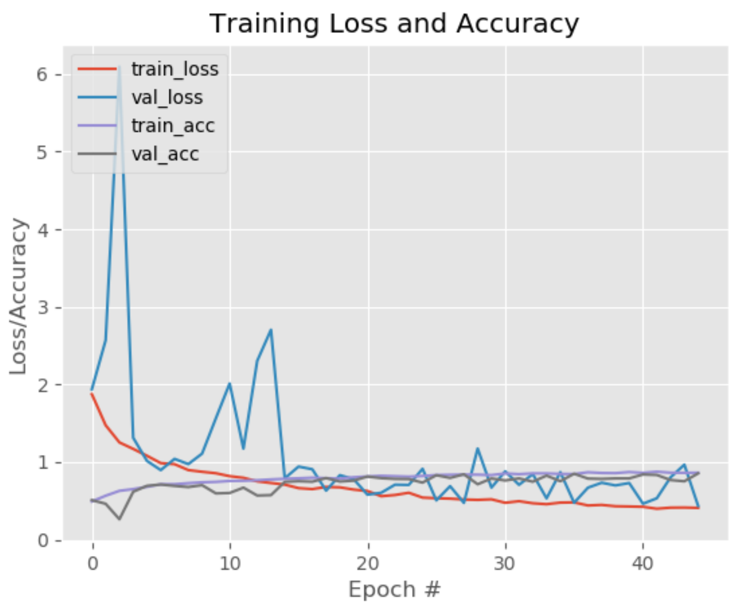

Figure 7: Training loss and accuracy of modified AlexNet in the second case in 45 epochs

5 Conclusion

It has been reached good performances with both network architectures VggNet and AlexNet, we get better performances on larger datasets that we can use to train the cnn. The performances could be improved increasing the number of epochs. It has been used 45 epochs in our tests, the network is trained in about five minutes reaching 85% of accuracy that is a quite good result.

6 References

- VggNet Architecture
- AlexNet Architecture
- ARGOS Venice Boat Classification
- CNN slides