Robotics Software Engineer Nanodegree: Inference Project

Manuel Huertas López

Abstract—This project consists of two parts. In the first part of the project a data set is provided, and a Neural Network must be trained in order to achieve an inference time less than 10 ms and accuracy greater than 75%. In the second part of the project an original idea for a robotic inference system must be selected; them the data must be collected and a network trained. The project selected, identify and classify coins, has been taking into account that, in a robotic system we want to perceive the world, make a decision based on that perception, and them act upon this decision. The Neural Network may have several real world applications: automate the process of counting coins, help blind people in everyday situations and so on.

Index Terms—inference project, udacity, deep learning.

1 Introduction

The field of robotics has evolved during the last decades. During the industrial revolution, the robots that were manufactured could operate repetitively with high precision. The current robots can move around the scene detecting automatically obstacles and reacting in real time to changes in the environment. The flexibility needed for the new era of robots can be beneficed of the neural networks applied to image classification or semantic segmentation. By using these techniques a trained robot can rapidly identify objects in the scene and react accordantly. It is not only provided with sensors to avoid obstacles, it can detect object in the scene, classified them, and to make decision bases on this classification.

In addition, the use of neural network can be extended to days basis. Nowadays, a lot of people posses a mobile with the capabilities to carry on inference task. The range for applications that can be developed for these platforms is huge with the only limitation of the imagination.

A lot of effort to provide the robots these capabilities cames from collecting data to train the Neural Network and to define the network architecture itself. In the present work an example, classify coins, have been selected in order to evaluate the data collection process and the architecture selection and finally, to do the verification.

2 Background / Formulation

The original idea selected was identified a coin in an image and to classify the coin, three different kinds of classes were used, corresponding to the European coins of 10 cents, 20 cents and one euro. In order to classify the coin a two-step strategy have been used. In the first step a classic feature extraction algorithm has been used. Due to the coins to be classified share a circular shape, the Hough Transform was used to find the coins in the image, classified them depending on the position in the image. The second step will use the inference, after training the network, to classify new images not seen before. The idea is to extract the region of the image corresponding to the coin and to inject this image as the input of the neural network.

It must be noticed that this is not a limitation for the real system, when trying to inference new samples, since the Hough Transform will be just used in this context to extract the portion of the image relative to the coin, and the neural network will inference the exact class for this new sample.

In addition, the principal advantage for this strategy is that the background of the image is not a problem during classification, because it is extracted, making this network quite robust to different environments.

The network selected was AlexNet because the dataset was composed of colour image and the size 256x256 fits the image data set. [1] [2]

3 Data Acquisition

The data set was collected using the camera of a mobile phone. The coins were put on a white paper and a video recording was started. Every few seconds the recording was paused to turn the coin randomly. Then, a tool to extract individual images from the video, ffmpeg, was used. The image were stored in jpeg format with a size of 1920x1080 pixels. A total of 2900 images, approximately were collected.

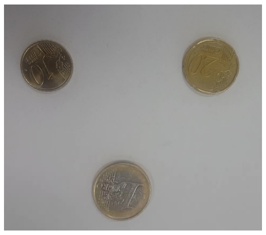
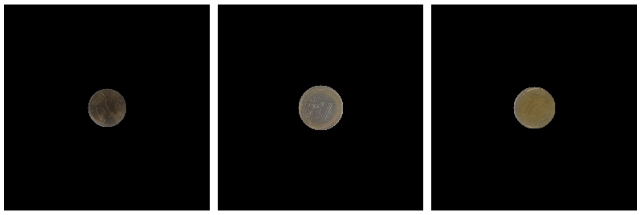


Fig. 1: Example of raw image coming from the mobile camera.

A python script was created to take every image and, using the Hough Transform, extract the coins from the images. The individual images were safe in directories with the name of the classes, in this case : coin10, coin20 and con100.

These images where down-sampled. The final image size, the ones used to train the network, was 256x256 pixels. The single images for every class was safe with the following directory structure:

```
data
 ├─coin10
 │  ├─coin-10-image-1.jpg
 │  └─coin-10-image-2.jpg
 ├─coin20
 │  ├─coin-20-image-1.jpg
 │  └─coin-20-image-2.jpg
 └─coin100
    ├─coin-100-image-1.jpg
    └─coin-100-image-2.jpg
```



(a) 10 cents (b) 20 cents (c) 1 euro

Fig. 2: Example images coin extracted using Hough Transform.

4 Neural Network Trainning

Digits tools was used to train the network. The process of training the network in digits takes two steps. The following section show the configuration for both networks: the train the dataset provided by udaciy and the original idea.

4.1 Udacity

4.1.1 *Create the Model*

In the first step the model from the data set, classification type, was created. The following parameters were selected:

- Image type[Color]: color is a 3-channel RGB iamge.
- Image size(Width x Height) [256,256]: the image input will be resize to this value.
- Trainning Image: the folder with the structure as provided by udacity.
- %for validation[25%]: the percent of image to set apart for the validation process.
- %for testing[5%]: the percent of image to set apart for the test process.
- DatasetName[udacitydataset]: the name of the dataset for future references.

4.1.2 *Create the Network*

The following step the consists in training the network. A classification network was selected with the following parameters:

- Select Dataset[udacitydataset]: the previous created dataset.
- Training epochs[5]: how many passes throught the training data.
- Standard Network[GoogLeNet]: the predefined network achitecture.

4.1.3 *Train the Network*

The network was trained with the model and network architecture previously defined using the digits platform.

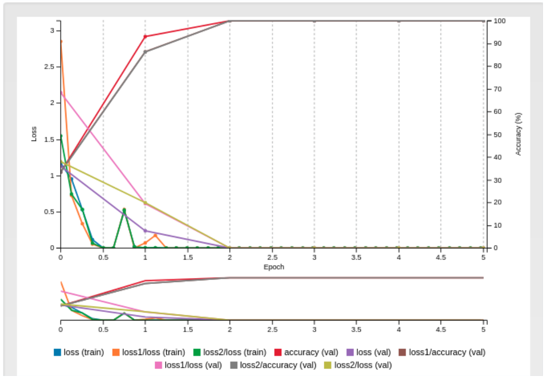


Fig. 3: Trainnig chart Udacity.

4.2 Original Idea

4.2.1 *Create the Model*

In the first step the model from the data set, classification type, was created. The following parameters were selected:

- Image type[Color]: color is a 3-channel RGB iamge.
- Image size(Width x Height) [256,256]: the image input will be resize to this value. Since the original image was already of this size, resizing will not take place.
- Trainning Image: the folder with the structure describe in the previous section.
- %for validation[25%]: the percent of image to set apart for the validation process.
- %for testing[10%]: the percent of image to set apart for the test process.
- DatasetName[coindataset]: the name of the dataset for future references.

4.2.2 *Create the Network*

The following step the consists in training the network. A classification network was selected with the following parameters:

- Select Dataset[coindataset]: the previous created dataset.
- Training epochs[5]: how many passes throught the training data.
- Standard Network[GoogLeNet]: the predefined network achitecture.

4.2.3 *Train the Network*

The network was trained with the model and network architecture previously defined using the digits platform.

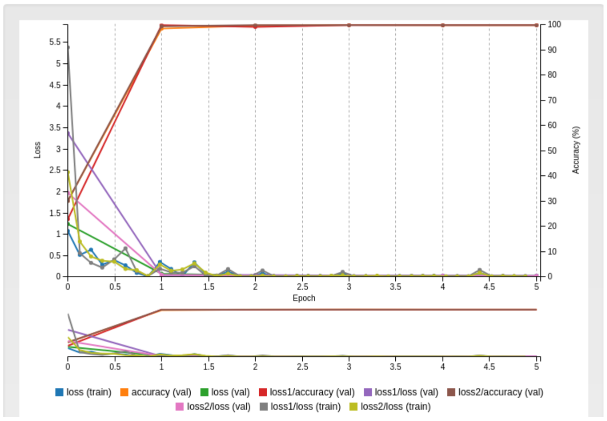
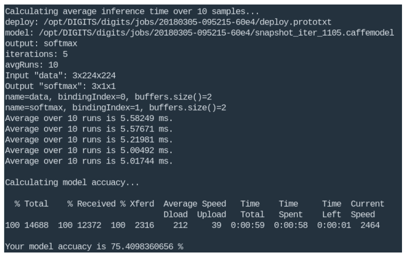


Fig. 4: Trainnig chart original idea.

5 Results

5.1 Udacity

After training the network the result was evaluated, the result is on requirements.

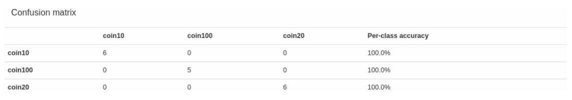


```
Calculating average inference time over 10 samples...
deploy: /opt/DIGITS/digits/jobs/20180305-095215-60e4/deploy.prototxt
model: /opt/DIGITS/digits/jobs/20180305-095215-60e4/snapshot_iter_1105.caffemodel
output: softmax
iterations: 5
avgRuns: 10
Input "data": 3x224x224
Output "softmax": 3x1x1
name=data, bindingIndex=0, buffers.size()=2
name=softmax, bindingIndex=1, buffers.size()=2
Average over 10 runs is 5.58249 ms.
Average over 10 runs is 5.57671 ms.
Average over 10 runs is 5.21981 ms.
Average over 10 runs is 5.00492 ms.
Average over 10 runs is 5.01744 ms.

Calculating model accuacy...

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 14688  100 12372  100  2316    212     39  0:00:59  0:00:58  0:00:01  2464

Your model accuacy is 75.4098360656 %
```

Fig. 5: Evaluation of the model udacity.

5.2 Original Idea

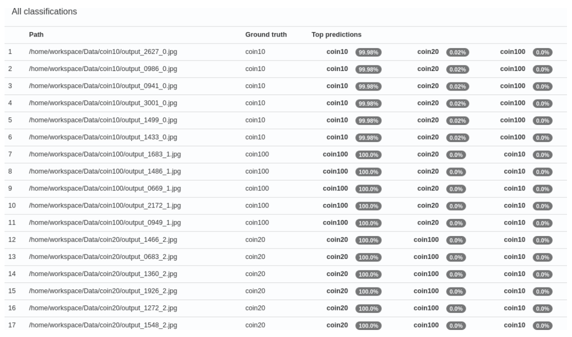
After training the network a subset of the image reserved to test the result with the following output:



Confusion matrix

	coin10	coin100	coin20	Per-class accuracy
coin10	6	0	0	100.0%
coin100	0	5	0	100.0%
coin20	0	0	6	100.0%

Fig. 6: Confusion Matrix.



All classifications

	Path	Ground truth	Top predictions					
1	/home/workspace/Data/coin10/output_2627_0.jpg	coin10	coin10	99.98%	coin20	0.02%	coin100	0.0%
2	/home/workspace/Data/coin10/output_0986_0.jpg	coin10	coin10	99.98%	coin20	0.02%	coin100	0.0%
3	/home/workspace/Data/coin10/output_0941_0.jpg	coin10	coin10	99.98%	coin20	0.02%	coin100	0.0%
4	/home/workspace/Data/coin10/output_3001_0.jpg	coin10	coin10	99.98%	coin20	0.02%	coin100	0.0%
5	/home/workspace/Data/coin10/output_1499_0.jpg	coin10	coin10	99.98%	coin20	0.02%	coin100	0.0%
6	/home/workspace/Data/coin10/output_1433_0.jpg	coin10	coin10	99.98%	coin20	0.02%	coin100	0.0%
7	/home/workspace/Data/coin100/output_1683_1.jpg	coin100	coin100	100.0%	coin20	0.0%	coin10	0.0%
8	/home/workspace/Data/coin100/output_1486_1.jpg	coin100	coin100	100.0%	coin20	0.0%	coin10	0.0%
9	/home/workspace/Data/coin100/output_0669_1.jpg	coin100	coin100	100.0%	coin20	0.0%	coin10	0.0%
10	/home/workspace/Data/coin100/output_2172_1.jpg	coin100	coin100	100.0%	coin20	0.0%	coin10	0.0%
11	/home/workspace/Data/coin100/output_0949_1.jpg	coin100	coin100	100.0%	coin20	0.0%	coin10	0.0%
12	/home/workspace/Data/coin20/output_1466_2.jpg	coin20	coin20	100.0%	coin100	0.0%	coin10	0.0%
13	/home/workspace/Data/coin20/output_0683_2.jpg	coin20	coin20	100.0%	coin100	0.0%	coin10	0.0%
14	/home/workspace/Data/coin20/output_1360_2.jpg	coin20	coin20	100.0%	coin100	0.0%	coin10	0.0%
15	/home/workspace/Data/coin20/output_1926_2.jpg	coin20	coin20	100.0%	coin100	0.0%	coin10	0.0%
16	/home/workspace/Data/coin20/output_1272_2.jpg	coin20	coin20	100.0%	coin100	0.0%	coin10	0.0%
17	/home/workspace/Data/coin20/output_1548_2.jpg	coin20	coin20	100.0%	coin100	0.0%	coin10	0.0%

Fig. 7: Classification result of the network for testing images.

The results of the classification were really good.

6 Discussion

One of the key things to obtain good result was to have a big number of samples. In addition, the image extraction, using a Hough Transform, helped to train the network. Once the network was trained it can be easily deployed in to a mobile and to be used in many different kind of applications.

7 Conclusion / Future work

The accuracy of the classification process achieved was really good. Due to time constraints the classes consists only in three coin and the same side. The final number of the classes is quite bigger than that. There are eight euros coins with two side each. One of the size is common for all the European countries but the other side is country specific. This makes the process of gathering the samples more difficult and time consuming. Once the network is trained with all the coin types and size a mobile application can be used to inference the coins in the day basis. The fact that the background is extracted will make this network quite robots again different environments: coins spread out over a pub's bar or over a counter in a book store.

References

[1] https://eu.udacity.com *Robotics Software Engineer Nanodegree program*.
[2] Adit Deshpande *The 9 Deep Learning Papers You Need To Know About*.