RoboND Project: Robotic Inference

Milton Wong

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, Deep Neural Network, Nvidia TX2.

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 comes 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

2.1 Udacity

The Udacity provided data set, to be classified by the Neural Network, consists of a collection of objects of three types: candy boxes, bottles and nothing.

GoogleNet network was selected for several reasons:

1) The image size fits well with one required by the network.
2) The network gives enough accuracy to pass requirements.
3) The number of operations, that is related to the inference time, is around 3 G-ops.

A classification network was selected with the following parameters:

- Training epochs: how many passes throughout the training data.
- Standard Network[GoogLeNet] [1]: the predefined network architecture.

2.2 Original Idea

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.

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.

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.

A python script was created to take every image and, using the Hough Transform, to extract the coins from the

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

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:

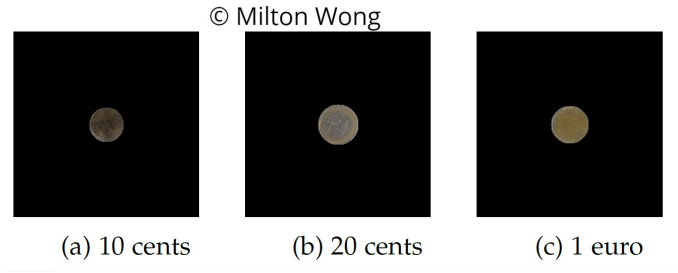

Fig. 2. .

4 Neural Network Training

Digits tools [2] 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 udacity 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 image.
- Image size(Width x Height) [256,256]: the image input will be resized 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 Train the Network

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

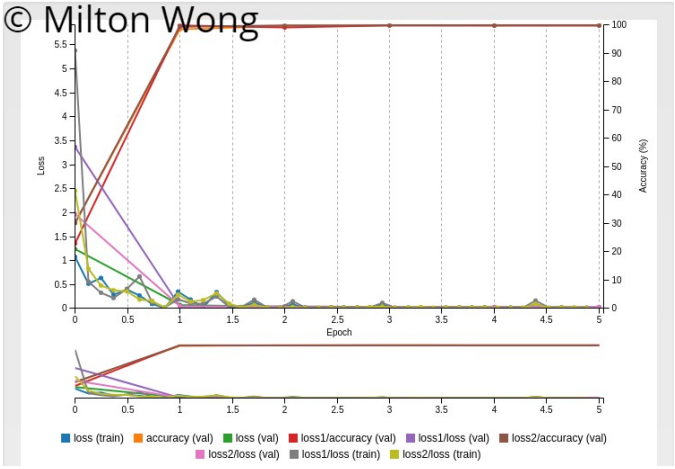

Fig. 3. Training chart from Udacity.

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 image.
- 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.
- Training 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 Train the Network

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

5 Results

5.1 Udacity

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

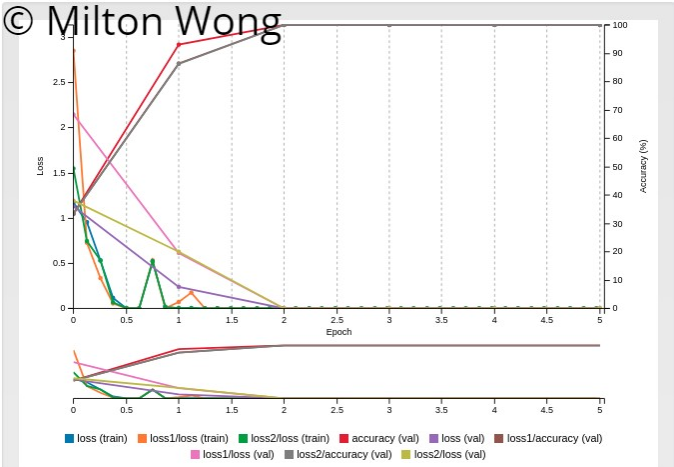

Fig. 4. Training chart from original idea case.

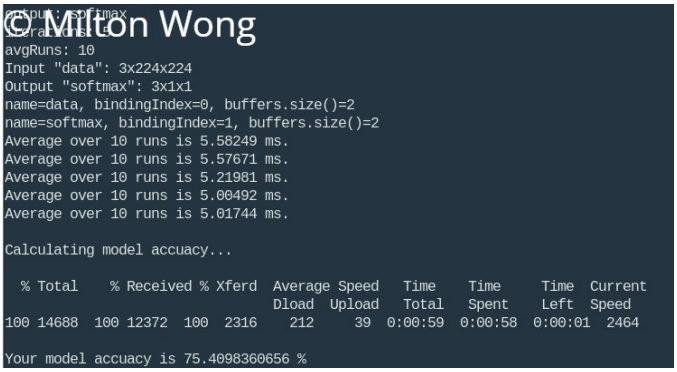

Fig. 5. Evaluation of the model from Udacity.

Fig. 6. Confusion Matrix.

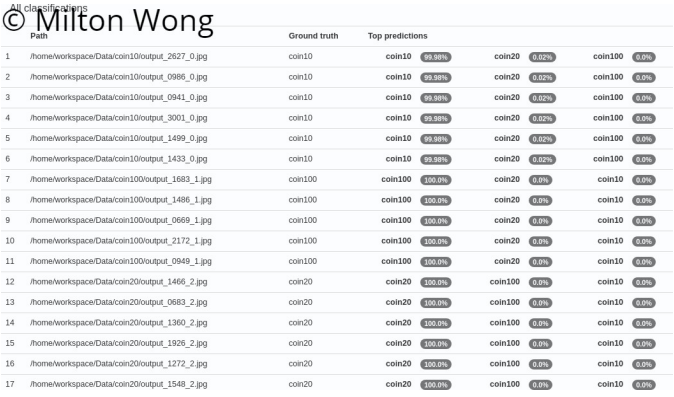

Fig. 7. Classification result of the network for testing images.

5.2 Original Idea

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

The results of the classification were really good.

6 Discussion

It has been notice during the development of this project the importance of the data collection process. In order to have a good accuracy is quite important to have a big number of samples of every class. The process of acquiring these images can be a complex and time-consuming task. The background where the object were collected is also part of the acquisition problem. In this project a mix solution, using traditional image processing techniques, Hough Transform, was used to overcame this problem. The final accuracy obtained was quite good.

There are some parameters that must be taken in account depending on the final deployed hardware and the desire performance:accuracy, inference time, power consumption, memory use.

There are relation between these parameters. For example,there are a hyperbolic relationship between the accuracy and inference time, in order to have a small increase in the accuracy the inference time will increase following a hyperbolic curve.

For that reason the network architecture must be selected taking in consideration the specific requirements for the concrete application. Furthermore, due to the possible limitations of the hardware where the solution will be deployed, and taking in consideration the frame per seconds to be inference and the accuracy desire, the correct network architecture must be selected.

In this project GoogleNet was selected due that the solution for both Udacity model and own model was on requirements. Furthermore, due to the possible limitations

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] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[2] N. DIGITS, "Interactive deep learning gpu training system," *NVIDIA Developer [2017]. URL: https://developer. nvidia. com/digits*, 2015.