ML-Based Image Enhancement of Low-Quality QR Code Images for Fast Data Decoding

Abstract

This work aims to construct a fast system to enhance the quality of QR Code images to improve the decoding success rate hence decreasing the time required to read the data contained in 2D bar codes. The system is an Convolutional Neural Network (CNN) with a structure of an auto-encoder, trained with TensorFlow toolkit on images self-generated. It takes as input an image with a poor resolution and quality and outputs a clear, noise-less picture of the same QR Code. The output of the CNN can then be used to decode the QR Code's data with any preferred decoding algorithm. The system has been deployed on the Raspberry Pi 3B+ and evaluated in different simulation obtaining an increase of about +100% success rate of correct decoding of low-quality QR Code images.

Keywords: QR Code, Image Enhancement, CNN, Computer Vision, Machine Learning

摘 要

此工作主要用于建立一个快速增强二维码图像的系统，从而增加二维码解码率，减少解码时间。本系统是一个由卷积神经网络构成的自动编码器，通过 TensorFlow 工具集在自生成的图片上进行训练的。它将具有较差分辨率和质量的图像作为输入，并输出相同二维码的清晰，无噪声的图像。 然后，CNN 的输出可用于利用任何优选的解码算法解码二维码的数据。 该系统已部署在树莓派 3B+上，并在不同的模拟中进行评估，使低质量二维码图像的正确解码成功率提高约+100%。

关键词：二维码，图像增强，CNN，电脑视觉，机器学习

Table of Contents

Chapter 1. Introduction

1.1 Quick Response Code

The Quick Response Code, or simply QR Code, is a two-dimensional bar code, which is nothing but a table of binary values 0 or 1 representing some kind of information. It is possible to represent this matrix as a black and white image, where the ones are white pixels and the zeros are black pixels as shown in Figure 1. Such a representation can then be used to communicate the information contained in the QR Code through the utilization of camera sensors.

Figure 1: example of a QR Code image

The QR Code is mainly used as an optical label replacing the traditional one-dimensional bar code in different applications and for different reasons. Some of the benefits and advantages of using QR Codes compared to standard 1D bar codes are listed below.

1. Storage capacity: the 2D nature of the bar code allows to store more data in a single QR Code.
2. Attractiveness: contrarily to traditional bar codes, the QR Code can be easily personalized with colors or by putting a logo in within the image representation. In advertisement or

marketing applications, this increase the probability of being noticed by people and in general it is considered more good-looking than the 1D bar codes.

3. Readability: QR Code images are in general more flexible than other representations allowing a greater success rate in data decoding.

In certain applications, as will be discussed below, it is not possible to correctly decode the information inside a QR Code as the image quality is not good enough to get a reliable decoded data [2, 4, 5, 8]. This thesis aims to increase the decoding accuracy in such applications providing a image quality enhancement system which low inference latency and low deployment cost.

1.2 Applications

Application of QR extended to many settings and fields. Nowadays it is possible to find the QR Code in manufacturing, transportation, marketing, advertisement, gaming, social networks, payment methods, receipts management etc. Two everyday-used applications in money transfer and mobile payment market, particularly in the chinese region, are the Alipay app from Alibaba Group, and WeChat, Tencent’s messaging app.

Figure 2: Alipay's payment tranfer

Figure 3: WeChat payment method using QR Code

Another interesting application is logistics of packages or in general material, products or their components. In this setting, a single camera sensor is responsible to decode the data contained in the

code printed on the object, retrieve information by querying some internal database and give insights to human operators or to automatic machines of where should the object be directed to.

One of the main challenges in this operation is that the QR Code has to appear in front of the camera for a period of time that is at least enough for the sensor to shoot a good quality picture and decode the information from it. Moving QR Code could appear blurred and not be correctly decoded by the camera, see Figure 4. Since in logistics time is crucial, to avoid stopping the material flow and to ensure that the decoding process succeeds, it might be necessary the use more expensive cameras able to take images in very high fps (frames per second).

Figure 4: Effect of the motion blur on a QR Code. On the left, the original QR Code while on the right the QR Code as seen from a camera sensor.

1.3 QR Code Decoding with Low-Quality Images

In literature one can find many decoding mechanisms and methods that can be applied to QR Codes. The decoding methods focus on the flexibility under different light conditions, or with blurred, noisy, low-quality or distorted images. The aim of all the decoding methods is to be able to increase the accuracy while decreasing the decoding time, its computational and financial costs.

The focus of this thesis however will be to make an intermediate system connecting the row image coming from the camera sensor of a device and any decoding algorithm used. The system will be in charge of enhance the image receiving from the camera and return it to the decoding algorithm. The enhanced image is supposed to achieve better detection and decoding results than the original raw image.

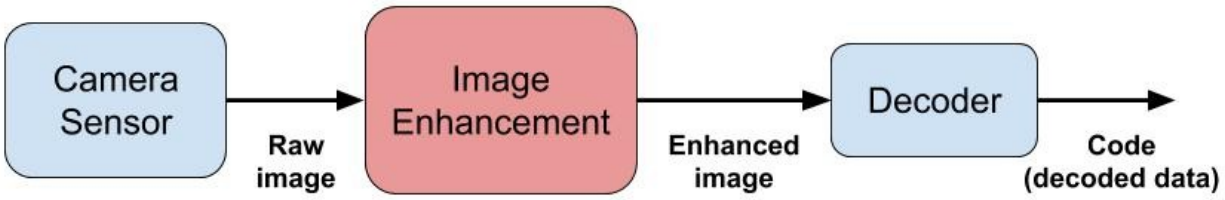

Figure 5: Framework of the overall decoding system. This thesis is focused on the central block: image enhancement

The system structure is presented in Figure 5. The main advantage of such an approach is that it does not influence the decoder's algorithm, allowing everyone to enhance their already-existing decoding system regardless the approach or application they used. In addition, the image enhancement mechanism is also independent of the standard of the QR Codes.

Chapter 2. Image Enhancement

2.1 Image De-Blur and Enhancement

The purpose of the image enhancement system is to take a low-quality image, process it to obtain a more readable image and finally input it in the decoder. The low quality of the image could come from different reasons:

- Low image resolution. [5]
- Not ideal light conditions.
- Distortion.
- Noise [5, 6]
- Motion blur, as approached in [6].

The system should be flexible enough to work under all these circumstances, without focusing on a single source. In addition, the system should run on real time, i.e. should not be time-costly. This requirement is important because image processing algorithms can require much time if not designed otherwise. The input and output of the system are presented in figure Figure 6.

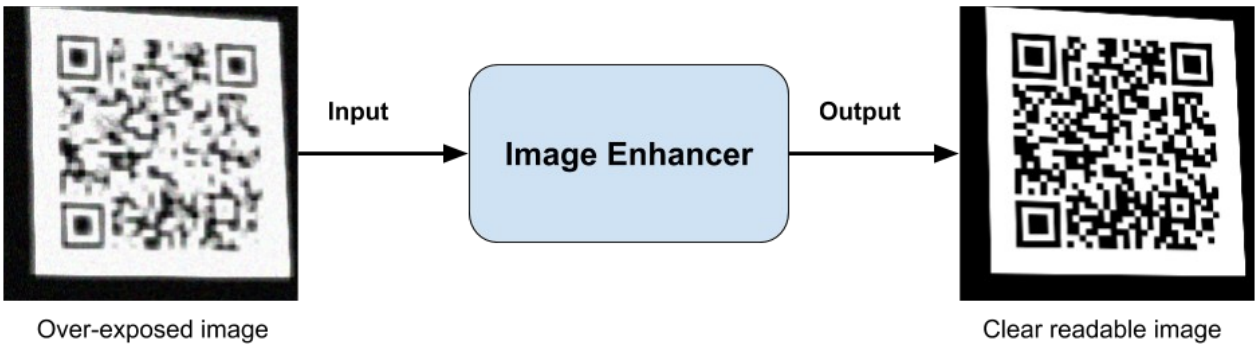

Figure 6: Enhancer input-output example

To satisfy the requirements of generality and speed, a Machine Learning (ML) approach has been utilized. This method will be described in details in the next paragraph.

2.2 CNN Model

The enhancer developed is based on a Convolutional Neural Network or in short CNN [3], trained from zero and built on a structure usually referred as auto-encoder. CNNs are particular kinds of Neural Networks (NN) which simulate the convolution operation. The convolution I*K between an input matrix I and another matrix, the *kernel* or *filter*, K is obtained by translate pixel-by-pixel the kernel over I, executing the sum of the element-wise product as shown in Figure 7. The convolution outputs a new matrix with shape depending on the size of I, K and some others parameters such as the *padding* size and the *span*.

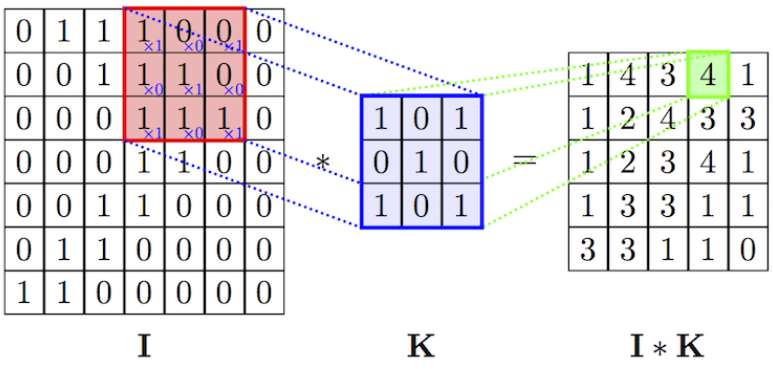

Figure 7: Example of convolution between input image (matrix) I and filter K

In the CNN designed in this thesis, convolution is executed four times. The first time from the single original image (no padding applied), with 32 different 3 by 3 filters. The output of this operation is a set of 32 images, each coming from a the convolution of the raw image with a different filter, with size slightly reduced. The size of these matrices are then reduced by a factor of 2, by keeping only the maximum values of the 2 by 2 squares composing the images. This operation is the so-called max-pooling operation. The images are then used to execute a second round of

convolution through 16 different kernels, also 3 by 3. The resulting images pass through a second max-pooling operation.

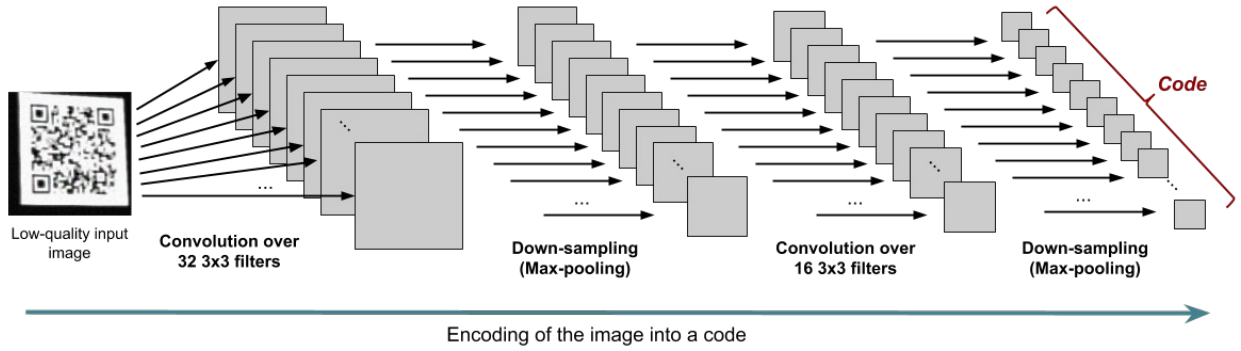

Figure 8: Image of the first part of the CNN

At this point the images have a much smaller dimension of the original image. Since they have been obtained with operations from the original image, they represent the *code* inside the auto-encoder structure. This *code* is used to recompute a non-blurred image. The matrices are input in the third convolution with 16 filter 3 by 3, but this time padding is applied, obtaining output images with bigger dimension. The dimension doubled after a up-sampling operation. The final convolution and up-sampling is finally executed, obtaining a single output image.

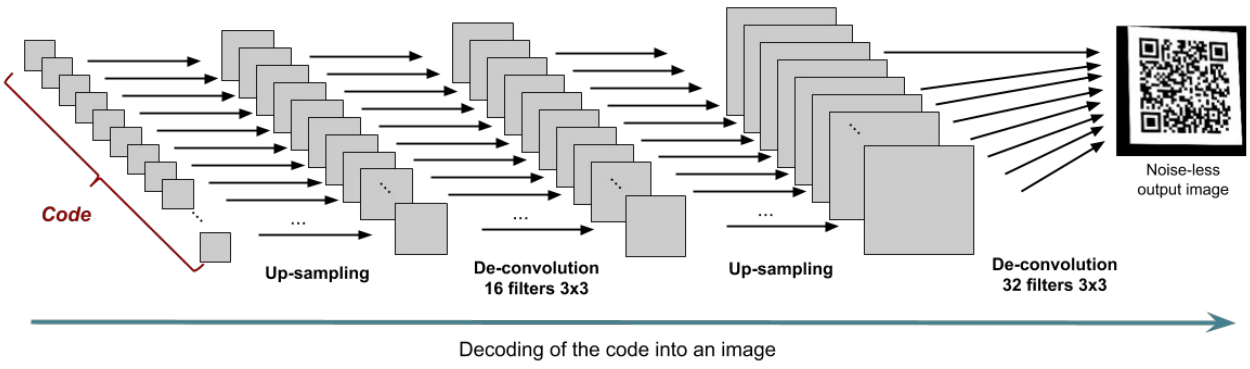

Figure 9: Representation of the second part of the CNN.

2.3 Decoding of QR Code

As decoder of the enhanced image is then input on a decoder. As the decoding part of the system is not the main focus of this thesis, a third party decoder has been used. The app utilized is *zbar* [9], an open-source library available for C++, Python, Ruby and Perl programming languages. The zbar library allows users to detect a QR Code or even a traditional one-dimensional bar code. It will be however utilized only with the former application.

Chapter 3. CNN Model Training

3.1 Training data retrieval

The CNN models contains 96 different filters of size 3 by 3, which means that it contains 864 independent parameters. It wouldn't be reasonable to choose one by one all these values. Instead, as will be seen in the next paragraph, each parameters is learnt by data. The training data is composed of many sample couples input-output images. Since to train the CNN's parameters a good amount of couples is required, it is first necessary to produce realistic data.

To do that, an open-source python library has been used. First, 10,500 QR Codes containing random data of length between 50 and 150 character each. Each image has been then resized to a shape of 200 by 200 and distorted using OpenCV library, the distortion has also been set with random parameters, to ensure that no image has a completely equal transformation. This step was necessary to emulate the deformation of the image in real circumstances. The third step was to add motion blur in random directions, and at the same time modifying the exposure of the image, simulating images in different light environments. Lastly, randomly-selected salt-and-pepper noise was added to each image. Examples of this process are shown in Figure 10.

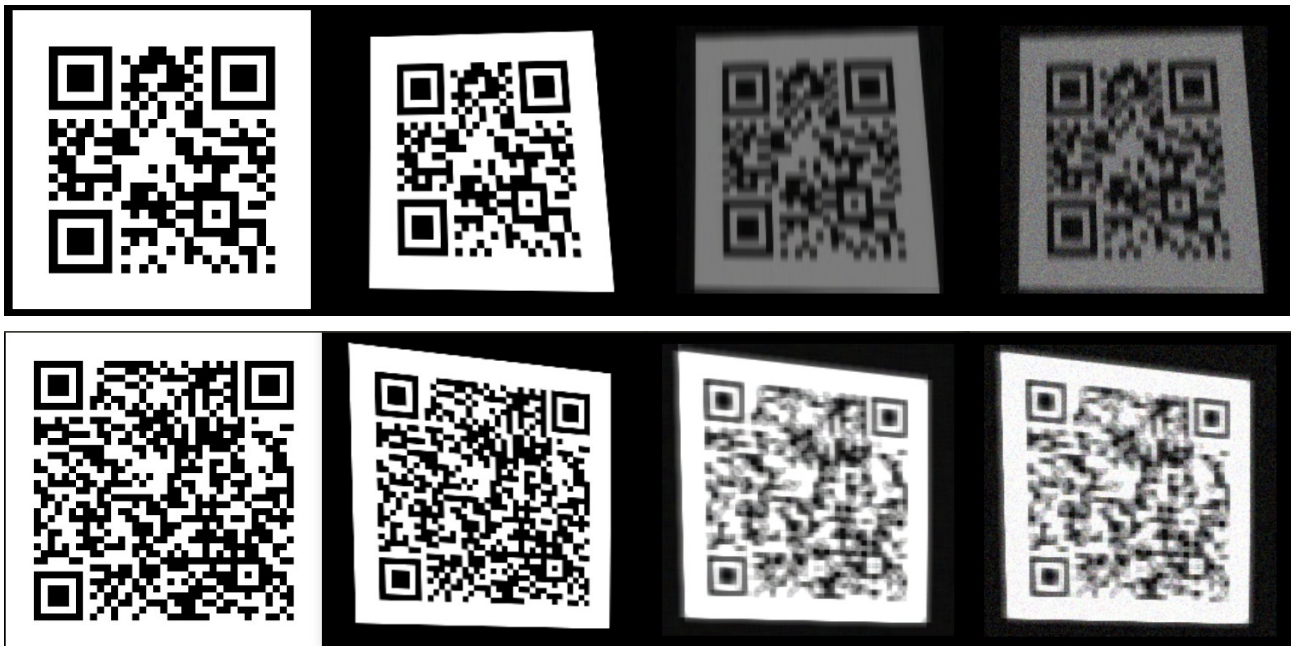

Figure 10: Process to create training data shown with two examples.

The resulting image of these transformations would then be used as input of the model, while the original distorted image will be the ground truth of the output, i.e. the image that the model should obtain with the given input, see Figure 14.

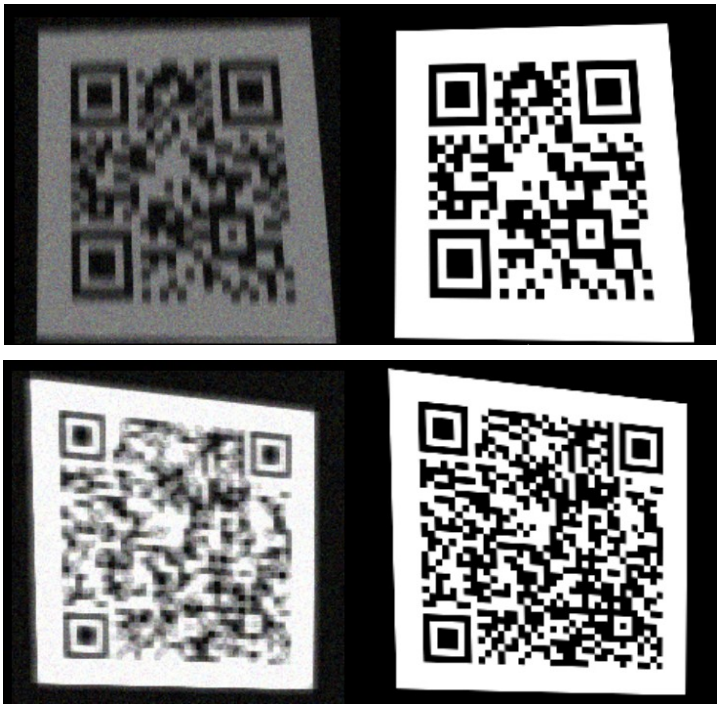

Figure 11: Example of inputs (left) and output (right) samples utilized to train the model. In total 10k+ samples

3.2 Training process

To model has been build using Google's open source ML toolkit TensorFlow v1.11. This toolkit is more and more popular in the ML development and deployment. First, the loss function has been defined: for each input, the loss of the model is the sum of the square root of the element-wise different between the reference ground truth and the actual mode output. The parameter tuning has been obtained through gradient descent, with the intent of minimize the loss function, utilizing the built-in Adam Optimizer function, which is a algorithm to automatically regulate the learning rate used after the gradient descent.Only about 90% of the total data has been used to train the model, while the remaining 10% has been used to evaluate and validate the model, ensuring that no over-fitting phenomenons happened during training. Each recursion, or epoch, the gradient of each

parameter is computed with a batch input-output samples of 16, helping the model to train faster over all the training data. The total number of epochs utilized is 50.

3.3 Results

The training process ran on a laptop with the support of a NVIDIA GT1060 graphic card. The time required for a single epoch was less than one minute. The total time required to complete the training process was 45 minutes. The trained model has been saved and is public on the author's GitHub page. The model has been evaluated by inferring new never-seen-before images and comparing the output of the model with the ground truth of the input. Some of the results are shown in Figure 12.

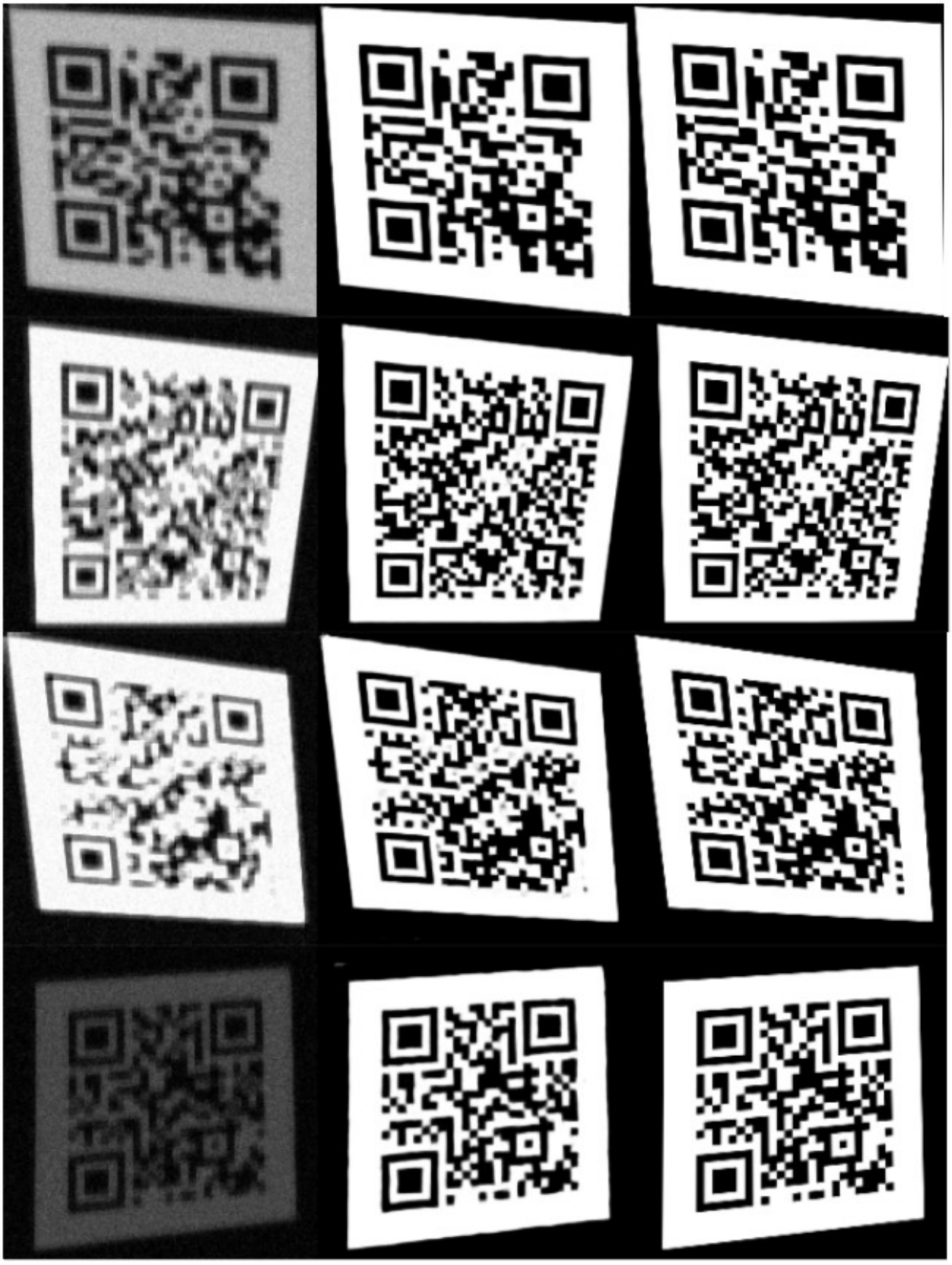

Figure 12: Evaluation over never-seen testing data. Left: the input images. On the center: the CNN's output. On the right: the ground truth.

As can be seen, although some of images still seems to be too noisy for the original image to be reconstructed, generally, the output images of the model appear to be much clearer than the inputs. The good performance of this model is more noticeable when looking at the detail of the three images (input, output and ground truth) in comparison, Figure 16.

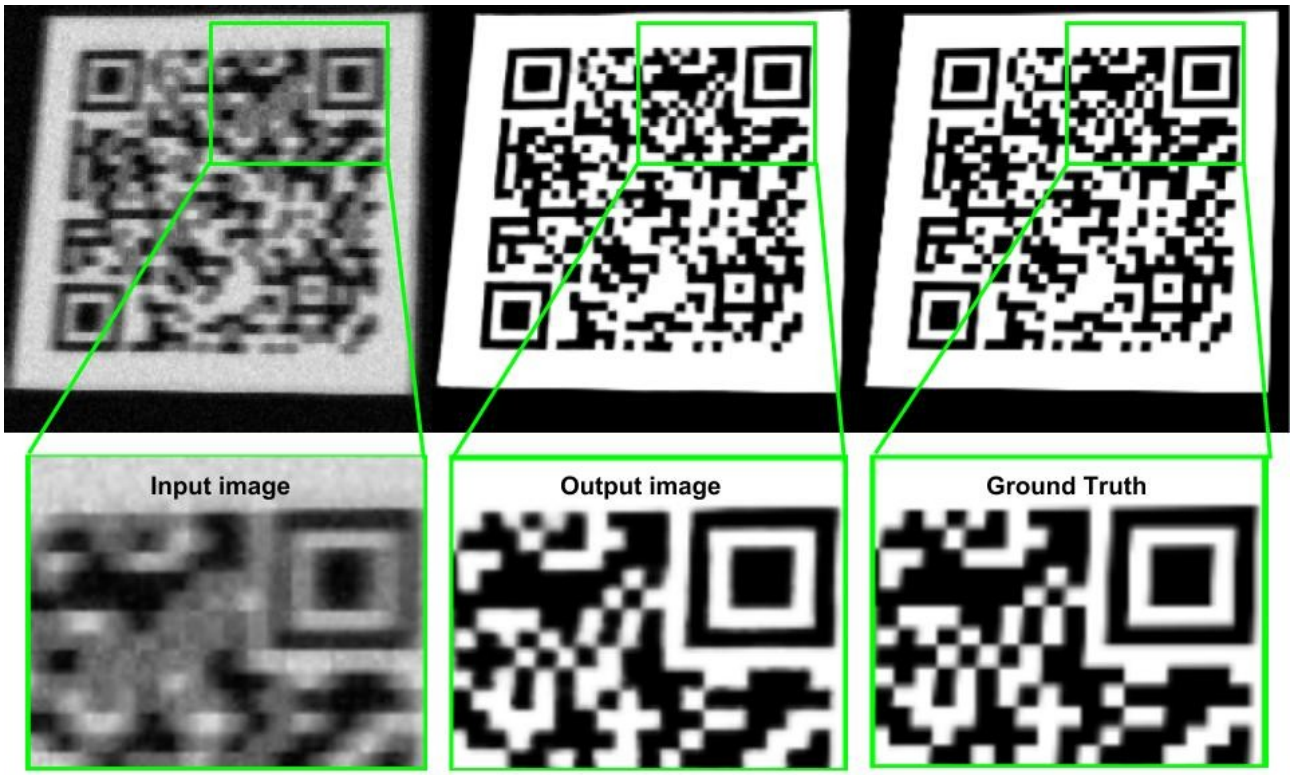

Figure 13: Details of input-output-ground truth images

Which such results, the system, as will be seen in the next chapter, is expected to improve considerably the detection success rate of the decoder (zbar library [9]). The inference time on the laptop for a single image is 0.005 seconds if running on the GPU and 0.08 if running on the CPU (quad-core Intel i7 7000).

Chapter 4. Evaluation of the System

4.1 Evaluation Scenario

To evaluate the system, a real-life scenario was simulated. In this scenario, a QR Code is positioned in front of a cheap, low-resolution camera. The system, structured as seen in Figure 5, is compared to the straight detection agent who takes the raw image from the camera and decodes it with the exact same decoder method. The outputs of the two approaches (see Figure 14) is then compared.

The camera sensor provide the image of a real QR Code through a video streaming of 1000 frames. The ground truth information of the QR Code has been previously saved on the software. For each frame, the output of each overall system is compared with the ground truth. At the end of the 1000 frames, it is possible to say which of the two systems performs better by comparing the number of successful detection obtained by the two methods.

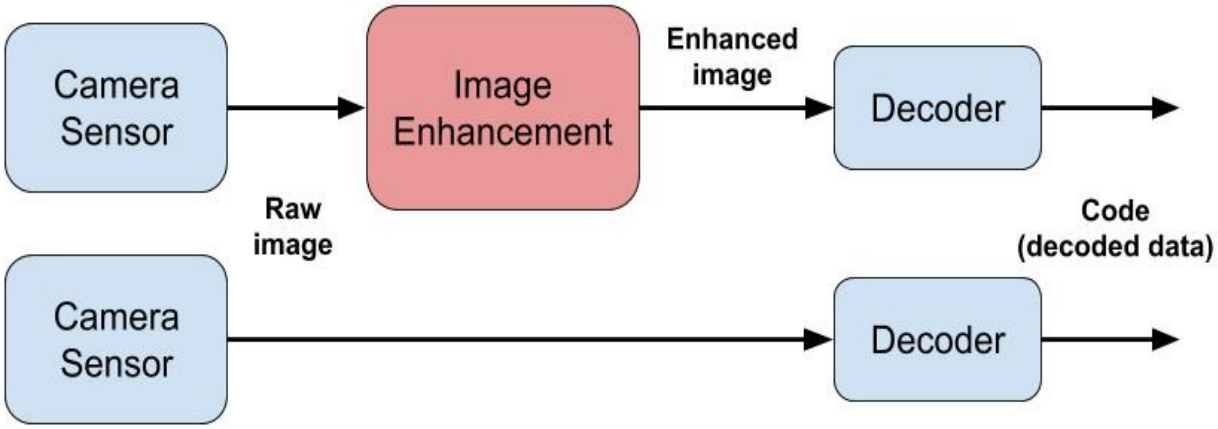

Figure 14: the two systems to be compared. On the top, the images from the camera are enhanced before being decoded.

4.2 Hardware Platform

Since, as introduced in the first chapter, one of the requirements of the system is its low-cost nature, the hardware platform chosen for the evaluation experiment has been chosen to be the Raspberry Pi 3B+, cheap but powerful board running Linux and support for TensorFlow ML framework [1]. The Raspberry, shown in , has a small dimension and contains numerous open source libraries, that make it suitable for many applications.

Figure 15: A picture of the Raspberry Pi utilized for the comparison

4.3 Results

The scenario described above has been run with different QR Codes containing URL links to common websites. The plot in Figure 16 presents the number of successful decodings vs the total number of frames taken by the camera for both the image-enhancing system (in green) and the normal straight-forward one (in blue).

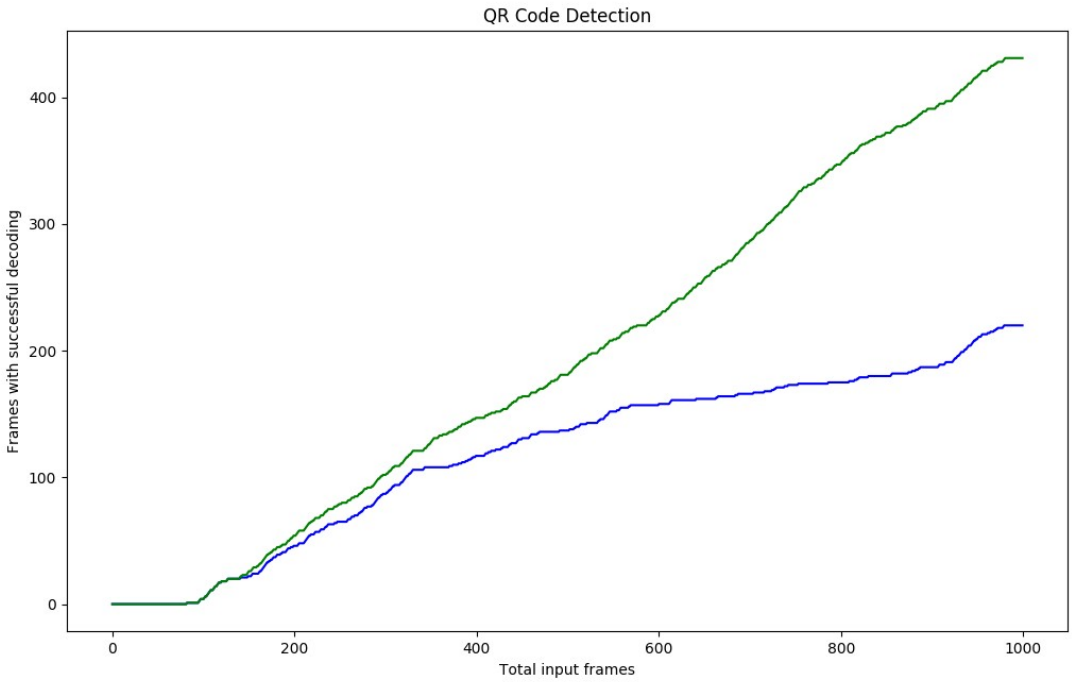

Figure 16: Comparison of number of successful decoding of this thesis' system (in green) vs traditional straight-forward detection (in blue).

The system achieves a number of correct decodings that is twice as high as the normal decoding system. The average accuracy over 10 different runs, with different lights conditions and different QR Codes, has been presented in the table below.

	Traditional system	Our system
Success decodings per 1000 frames	194.2	**386.4**
Percentage	19.42%	**38.64%**

Chapter 5. Conclusion

5.1 Summary and Conclusion

This work aimed to construct a system to enhance QR Code images to increase the success rate of decoding algorithms, i.e. to decrease the time required for decoding of the same QR Codes. A CNN autoencoder has been built and trained from scratch using self-produced data. The CNN model has been shown to be quite effective clearing the images from motion blur, general noise and not favorable light conditions. In addition, experiments on the field have shown that it help increasing QR Codes decoding's success rate of about +100% when utilizing low-resolution, cheap camera sensors.

5.2 Future work

Although the system has been successfully tested, more h noise, partially unfair light conditionsas to be done to generalize the model to other kinds of low-quality images, such us glass reflection etc. Also, it might be possible to integrate and construct a general model that works for both 1D and 2D bar codes, since in many real applications, they both appear on the image.

It would be also interesting to verify whether the auto-encoder structure can be used as decoding mechanism instead of image enhancement method. By adding an additional convolution and possibly increasing the filter numbers and the training data, it might be possible to train an auto-encoder that on is able to decode the information inside the bar code.

References

[1.] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. and Kudlur, M., 2016, November. Tensorflow: a system for large-scale machine learning. In OSDI (Vol. 16, pp. 265-283).

[2.] Hradiš, M., Kotera, J., Zemcík, P. and Šroubek, F., 2015. Convolutional neural networks for direct text deblurring. In Proceedings of BMVC (Vol. 10, p. 2).

[3.] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.

[4.] Liu, N., Zheng, X., Sun, H. and Tan, X., 2013. Two-dimensional bar code out-of-focus deblurring via the increment constrained least squares filter. Pattern Recognition Letters, 34(2), pp.124-130.

[5.] Munoz-Mejias, D., Gonzalez-Diaz, I. and Diaz-de-Maria, F., 2011. A low-complexity pre-processing system for restoring low-quality QR code images. IEEE Transactions on Consumer Electronics, 57(3).

[6.] Svoboda, P., Hradiš, M., Maršík, L. and Zemcík, P., 2016, September. CNN for license plate motion deblurring. In Image Processing (ICIP), 2016 IEEE International Conference on (pp. 3832-3836). IEEE.

[7.] Van Gennip, Y., Athavale, P., Gilles, J. and Choksi, R., 2015. A regularization approach to blind deblurring and denoising of qr barcodes. IEEE Transactions on Image Processing, 24(9), pp.2864-2873.

[8.] Xu, W. and McCloskey, S., 2011, January. 2D Barcode localization and motion deblurring using a flutter shutter camera. In Applications of Computer Vision (WACV), 2011 IEEE Workshop on (pp. 159-165). IEEE.

[9.] Zbar Website. URL: http://zbar.sourceforge.net/. Accessed in December 2018.