Localization of Visual Codes in the DCT Domain Using Deep Rectifier Neural Networks

Péter Bodnár[1], Tamás Grósz[2], László Tóth[2] and László G. Nyúl[1]

[1]University of Szeged, Department of Image Processing and Computer Graphics, Szeged, Hungary
[2]MTA-SZTE Research Group on Artificial Intelligence, Hungarian Academy of Sciences and University of Szeged, Szeged, Hungary

Abstract. The reading process of visual codes consists of two steps, localization and data decoding. This paper presents a novel method for QR code localization using deep rectifier neural networks, trained directly in the JPEG DCT domain, thus making image decompression unnecessary. This approach is efficient with respect to both storage and computation cost, being convenient, since camera hardware can provide JPEG stream as their output in many cases. The structure of the neural networks, regularization, and training data parameters, like input vector length and compression level, are evaluated and discussed. The proposed approach is not exclusively for QR codes, but can be adapted to Data Matrix codes or other two-dimensional code types as well.

1 Introduction

Two-dimensional visual codes are widely used at industrial setups and private projects as well. QR codes had a decent increase of usage in the last couple of years, more than other patented code types, like Aztec codes or Maxicodes. This is due to its well-constructed error correction scheme that allows recovery of damaged codes up to cca. 30 % of damage. Image acquisition techniques and computer hardware have also improved significantly, that made automatic reading of QR codes available. State of the art algorithms [8, **?**] do not require human presence and assumptions on code orientation, position and coverage rate in the image any longer. However, image quality and acquisition techniques vary considerably and each application has its own requirements for detection speed and accuracy, making the task more complex.

Speed is a general challenge in image processing, and in many cases, on-line execution of the algorithms is required. Usage of mathematical morphology or convolutions for edge or corner detection can greatly decrease performance. Machine learning techniques can overcome this issue and produce efficient solutions with respect to speed and accuracy.

In the last few years, there has been a renewed interest in applying neural networks, especially deep neural networks to various tasks. As the name suggests, deep neural networks (DNN) differ from conventional ones in that they consist of several hidden layers. However, to properly train these deep networks, the training method requires modifications, as the conventional backpropagation algorithm encounters difficulties

Bodnár P., Grósz T., Tóth L. and G. Nyúl L..
Localization of Visual Codes in the DCT Domain Using Deep Rectifier Neural Networks.
DOI: 10.5220/0005125700370044
In *Proceedings of the International Workshop on Artificial Neural Networks and Intelligent Information Processing* (ANNIIP-2014), pages 37-44
ISBN: 978-989-758-041-3

("vanishing gradient" and "explaining away" effects). In this case the "vanishing gradient" effect means that the error might vanish as it gets propagated back through the hidden layers [1]. In this way some hidden layers, in particular those that are close to the input layer, may fail to learn during training. At the same time, in fully connected deep networks, the "explaining away" effects make inference extremely difficult in practice [3]. Several solutions have been proposed to counter these problems. These solutions modify either the training algorithm by extending it with a pre-training phase [3, 7], or the architecture of the neural networks [2].

JPEG [11] is one of the most common still image formats, and provides efficient data storage and transfer. Most cameras can acquire images directly to JPEG format, some devices even can output a stream of JPEG images, which motivates research of image processing methods using this format. In this paper, we propose a neural network that works directly with the DCT coefficients of the JPEG image. Using this approach, only the first steps of the decoding have to be performed, while the most complex step, inverse DCT can be skipped.

2 The Proposed Method

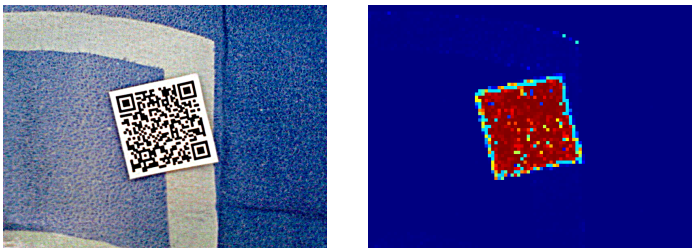
The JPEG decoding process can be stopped at the point where quantized DCT coefficients are restored from the file, right after the execution of the inverse of RLE and Huffman-coding. The matrix of coefficients that represent a 8×8 px block in the image, serves as the input vector of the DNN. For the order of the coefficients, the zigzag pattern is appropriate as it is defined by the JPEG standard, since learning efficiency does not differ when using differently ordered vectors of the same training set, but this order represent the "importance level" of the coefficients, and therefore it is recommended for evaluation of DNN performance using only the first n elements.

Each vector representing an image block is processed by the neural network individually and a measure is assigned which reflects the probability of presence of a QR code part in that block. After the evaluation of all image blocks, a matrix is formed with the probability values (Fig. 1). This can be considered as a feature image, downsampled by a factor of 8 from the original resolution. The next step of the process is to find clusters in this matrix that has sufficient size, compactness and high values of probability to form a QR code.

2.1 The Neural Network

Vectors are passed to the core of this approach, which is the neural network. We propose using deep rectifier networks instead of conventional ones, since they perform better at MNIST and other various image classification tasks, according to the findings of Glorot et al. [2].

Deep Rectifier Neural Networks (DRN) alter the hidden neurons in the network and not the training algorithm by using rectified linear units. These rectified units differ from standard neurons only in their activation function, as they apply the rectifier function $\max(0, x)$ instead of the sigmoid or hyperbolic tangent activation. Owing to their properties, the DRNs do not require any pre-training to achieve good results [2].



(a) real-life image example (b) feature image using JET color palette

Fig. 1. Image captured by a phone camera (a) and visualized feature image according to the output of the neural network (b).

The rectifier function has two important properties, namely its hard saturation at 0 and its linear behaviour for positive input. The first property means that only a subset of neurons will be active in each hidden layer. For example, when we initialize the weights uniformly, around half of the hidden units output are zeros. In theory, this hard saturation at 0 could harm optimization by blocking gradient backpropagation. Fortunately, experimental results do not support this, showing that the hard non-linearities do no harm as long as the gradient can propagate along some path [2]. Owing to the other property of the rectified units, namely the linear behaviour of the active units, there is no "vanishing gradient" effect [2]. This linear behaviour also means that the computational cost will be smaller, as there is no need to compute the exponential function during the activation calculation, and the sparsity can also be exploited. Unfortunately, there is a disadvantage because of this linearity, the "exploding gradient" effect, which means that the gradients can grow without limit. To prevent this, we applied L1 normalization by scaling the weights such that the L1 norm of each layer's weights remained the same as it was after initialization. What makes this possible is that for a given input the subset of active neurons behaves linearly, so a scaling of the weights is equivalent to a scaling of the activations.

Our deep networks consisted of three hidden layers and each hidden layer had 1000 rectified neurons, as DRN with this structure yielded the best results on the development sets.

The output layer of the neural networks consisted of two softmax neurons, one for the positive and one for the negative label, allowing the networks to output not only classification decisions but also posterior probabilities. As error function we applied the cross entropy function.

In our study were utilized two regularization methods to prevent overfitting, namely early stopping and weight decay. Early stopping was achieved by stopping the training when there was no improvement in two subsequent iterations on the validation set. As weight decay regularization the weights were scaled back after each iteration, forcing them to converge to smaller absolute values than they otherwise would.

The neural networks were trained using semi-batch backpropagation, the batch size being 100. The initial learn rate was set to 0.001 and held fixed while the error on the development set kept decreasing. Afterwards, if the error rate did not decrease in the given iteration, then the learn rate was subsequently halved.

We used our custom implementation for neural networks, which was implemented to run on GPU. The training of a DRN on the synthetic dataset took less than 8 minutes using an NVIDIA GTX-770 graphics card.

2.2 Input Data

The training vectors are labeled positive if the QR code coverage ratio is higher than a selected T_c threshold for that block. Typically, F-score peaks at $T_c \approx 0.5$, while $T_c \approx 0.1$ leads to better recall (hit rate). However, the number of these partially covered blocks is one scale smaller than the one of empty and fully covered blocks, therefore T_c is not a determinative parameter of the training. Furthermore, even if the DRN misses partially covered blocks, that only means it misses the perimeter of the code object, and expansion of the positively classified cell groups of the feature matrix overcomes this issue.

The input vectors for the DRN are one-dimensional vectors, formed by the quantized DCT coefficients of a 8×8 px block. During the decoding, the multiplication with the quantization table can be omitted for two reasons. On one hand, due to the non-linear nature of neural networks, they are capable to learn on a vector set and on the same set multiplied element-wise with another fixed vector, with similar efficiency. The other reason is that, the components of the input vectors were normalized as described in [4] to have zero mean and unit variance. This normalization improved the numerical condition of the optimization problem during training, ensuring faster convergence.

According to this setup, the DRN has to be trained using images of the same compression level as images of the end-user application, since we are not using the original DCT matrix, which would require a multiplication by the quantization table. Without de-quantization, different levels of compression applied to the same image content lead to different vectors. These vectors can be far away from the training samples of a specific compression level. To overcome this, DRNs can be trained using the de-quantized coefficient vectors, which are roughly the same on similar compression levels. Detailed evaluation of this concept is in the Results section (Fig. 4).

3 Evaluation and Results

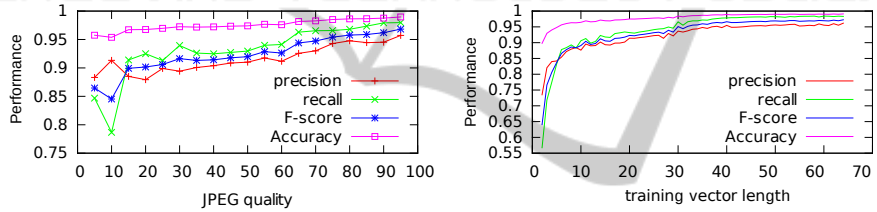
The test database consists of 10 000 synthetic and 100 arbitrarily acquired images containing QR code. The synthetic examples are built with a computer-generated QR code containing all of the lower- and uppercase letters of the alphabet in random order. This QR code was placed on a random negative image, with perspective transformation. After that, Gaussian smoothing and noise have been gradually added to the images, ranging [0,3] for the σ of the Gaussian kernel. For noise, a noise image (I_n) was generated with intensities ranging from [-127, 127] following normal distribution, and added gradually to the original 8-bit image (I_o) as $I = \alpha I_n + (1-\alpha) I_o$, with α ranging [0, 0.5]. Some samples with parameters being in the discussed ranges are present in (Fig. 2). A total number of 42 million vectors were extracted from those images, about 10 millons of them are labeled as positive. Real images were taken with a 3.2 Mpx Huawei hand-held phone camera. Significant smoothing is present on those images due to the

absence of flash and precise focusing capability. These images contain a printed QR code that mostly suffer from minor bending, placed on various textured backgrounds, like creased cloth, carpet, marble or cover page of a book (Fig. 1(a)), in order to create high background variability and to make the classification task more complex. The arbitrarily acquired image set roughly had QR codes of the same size as the artificial set, and its extracted vector set consisted about 750 000 vectors, about 180 000 of them being positive.

(a) $\sigma = 0.07$, $\alpha = 0.39$ (b) $\sigma = 0.65$, $\alpha = 0.77$ (c) $\sigma = 1.32$, $\alpha = 0.05$ (d) $\sigma = 1.88$, $\alpha = 0.95$ (e) $\sigma = 2.86$, $\alpha = 0.65$ (f) $\sigma = 2.99$, $\alpha = 0.69$

Fig. 2. Samples of the training database with different amount of smoothing and noise.

We examined the effect of compression to DRN performance. As expected, better image quality yield to better training results, as shown on Fig. 3(a).



(a) (b)

Fig. 3. DRN performance with respect to (b): the quality of the data set and (b): the length of the training vectors

Since the zigzag pattern represents the order of coefficients with respect to visual importance, it raises the question of how many elements are needed from this vector for efficient classification. Fig. 3(b) shows performance measures of DRNs trained on the first n elements of the vectors. It is shown that roughly the first 10 elements of the coefficient vector are sufficient for training a DRN that has F-score above 0.9, and performance only slightly increases when using more than half of the vector.

All networks trained to a test set of specific quality also have been evaluated on all sets, thus showing robustness of the networks with respect to difference of compression levels between the training and test sets (Fig. 4). Results show that DRNs are somewhat specific to the quality they were trained on, but still perform well up to a cca. 10–15 % tolerance. According to these results, training of a DRN to a specific image quality is not required, however they perform the best when both images are of the exact same quality. An analogous rule applies when training the DRN using all vectors extracted from images of various JPEG qualities. The DRN trained to those vectors would perform inferior to the one trained with the same JPEG quality as the end-user scenario.

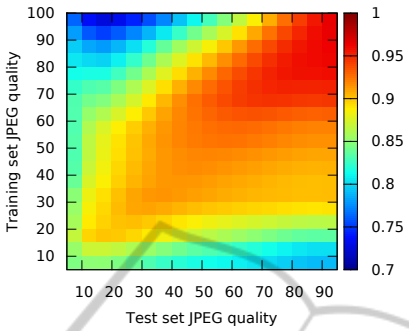


Fig. 4. Performance map of the trained DRNs used on images having the same content but different JPEG quality.

After all these experiments, a DRN has been trained with the best experimental training setup, in order to compare it with other localization methods of the literature. According to Table 1, NN training is proven to be a viable option for real-time efficient QR code localization, compared to the works of Ohbuchi et al. [6] and Lin et al. [5]. However, precision of the DRN on the real set is inferior, which is probably due to the smaller number of training vectors. Furthermore, the QR codes were slightly larger compared to the training set, which resulted in less prominent patterns to learn within an 8×8 px block. High-resolution images having QR code structure elements larger than the JPEG block size are required to have been downsampled before processing. This also has a positive effect to processing speed, since it greatly reduces the number of blocks that have to be evaluated. The downsampled image computation can be avoided by setting the camera to acquire lower resolution images in case we have assumption on the expected QR code size.

DRNs have to be trained to vectors that come from QR codes of comparable size to the expected test images. This limitation can be overcome using multiple DRNs trained on vectors of various QR code sizes, or one single DRN trained on all expected sizes, which will have inferior overall performance than the specific neural networks.

Performance measures were computed considering each block individually. However, groups of blocks with appropriate size and shape have to be extracted from this feature map by clustering or connected component labeling, and bounding boxes have to be defined in order to send cropped image regions to decoding algorithms [10, **?**].

We used our custom implementation for neural networks, which was implemented to run on GPU. The training of a DRN on the synthetic dataset took less than 8 minutes using an NVIDIA GTX-770 grapics card. The computation power of this setup allowed to process about 450 000 vectors per second, which means real-time processing of 800×600 px images with cca. 60 FPS, and 1600×1200 px images with cca. 15 FPS.

Table 1. Performance comparison of the proposed DRN and other REF-OHBUCHI [6] and REF-LIN [5].

Data set	Algorithm	Precision	Recall	Accuracy
Synthetic	Proposed	0.9607	0.9770	0.9899
Synthetic	REF-OHBUCHI	1.0000	0.8370	0.8730
Synthetic	REF-LIN	0.9340	0.9490	0.8890
Real	Proposed	0.8379	0.8935	0.9715
Real	REF-OHBUCHI	0.9500	0.8750	0.8360
Real	REF-LIN	0.9400	0.8930	0.8450

4 Concluding Remarks

Usage of neural networks is a viable approach for real-time localization of visual codes. We have examined performance of deep rectifier neural networks trained on the DCT coefficients of JPEG image blocks, without accessing actual pixel data. Using this approach, the most complex step of the JPEG decoding process, the inverse DCT can be omitted. We have proven efficiency of NNs on a large amount of input blocks of both synthetic and real input images.

NNs can also be trained on vectors of a specific compression level, and using these networks on images of the same level, the multiplication of the coefficients with the quantization table can also be omitted, thus further speeding up the process.

Our future work includes examination of the JPEG coefficients considering data of adjacent blocks, and the construction of post-processing steps aiming to increase compactness of block groups.

Acknowledgements

This publication is supported by the European Union and co-funded by the European Social Fund. Project title: Telemedicine-oriented research activities in the fields of mathematics, informatics and medical sciences. Project number: TÁMOP-4.2.2.A-11/1/KONV-2012-0073.

References

1. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proc. AISTATS. pp. 249–256 (2010)
2. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier networks. In: Proc. AISTATS. pp. 315–323 (2011)
3. Hinton, G. E., Osindero, S., Teh, Y. W.: A fast learning algorithm for deep belief nets. Neural Computation 18(7), 1527–1554 (2006)
4. LeCun, Y., Bottou, L., Orr, G., Muller, K. R.: Efficient backprop. In: Orr, G., K., M. (eds.) Neural Networks: Tricks of the trade. Springer (1998)
5. Lin, D. T., Lin, C. L.: Automatic location for multi-symbology and multiple 1D and 2D barcodes. Journal of Marine Science and Technology 21(6), 663–668 (2013)

6. Ohbuchi, E., Hanaizumi, H., Hock, L. A.: Barcode readers using the camera device in mobile phones. In: Cyberworlds, 2004 International Conference on. pp. 260–265 (2004)
7. Seide, F., Li, G., Chen, X., Yu, D.: Feature engineering in context-dependent deep neural networks for conversational speech transcription. In: Proc. ASRU. pp. 24–29 (2011)
8. Sörös, G., Flörkemeier, C.: Blur-resistant joint 1D and 2D barcode localization for smartphones. In: Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia. pp. 11:1–11:8. MUM '13 (2013)
9. Szentandrási, I., Herout, A., Dubská, M.: Fast detection and recognition of qr codes in high-resolution images. In: Proceedings of the 28th Spring Conference on Computer Graphics. pp. 129–136. SCCG '12 (2013)
10. Tekin, E., Coughlan, J.: A bayesian algorithm for reading 1D barcodes. In: Proceedings of the 2009 Canadian Conference on Computer and Robot Vision. pp. 61–67. CRV '09, IEEE Computer Society, Washington, DC, USA (2009)
11. Wallace, G. K.: The JPEG still picture compression standard. Consumer Electronics, IEEE Transactions on 38(1), xviii–xxxiv (Feb 1992)
12. Wang, K., Zou, Y., Wang, H.: Bar code reading from images captured by camera phones. In: Mobile Technology, Applications and Systems, 2005 2nd International Conference on. p. 6 (2005)