Image Deblurring for Chinese License Plates

Group 8

R. Fernández Mir C. Hawkes R. Holland
R. Hu R. Lee Mekhtieva A. Papadopoulos

Abstract

The aim of this report is to give an overview and a comparison of techniques that we experimented with while working on image deblurring. The specific application domain on which we focused was Chinese License Plates. Image deblurring is a hard, open ended problem in general and throughout this project we took various approaches to overcome the difficulties that were encountered. After presenting the most prominent of these we proceed to evaluate each of them in a comparative manner and conclude with some general discussion on the domain.

Contents

1 Introduction

The general problem which we were called to solve concerns image deblurring. We will get into further details on the specific application domain later in the section but we begin with a short introduction on the problem of image deblurring, in general.

Image deblurring is the 'process of making pictures sharp and useful' [1]. There are various ways in which this can be accomplished, since in some sense most of the information about the lost details of the clear image is somewhat present in the blurred one. In general, though, it is true that we can never recover the original image exactly, because of factors such as random noise and approximation errors.

The fact that image deblurring is such an ill-defined problem renders it hard to work with using exact methods and of course as any optimal solution is yet to be developed the problem is in general open and it is not sensible to try and develop general algorithms to solve it.

For this reason our project focuses on the specific problem of deblurring low quality small license plate images. This is done as part of the 2017 Huawei competition on image deblurring. The competition focuses on deblurring images of license plates, to make them readable by an OCR system.

The setting of the competition is as follows: Teams are provided with a data set consisting of 4000 clear, unlabelled images to be used as training data and a second data set consisting of 100 blurred and labelled images of license plates to be used as testing data. The competing teams are then expected to provide a tool which given an blurred image returns a deblurred version of it. The success of the deblurring is judged both subjectively based on how good the image looks and objectively based on how readable it is by Huaweii's OCR software.

As mentioned the problem at hand is in general open and to end up with our solution we experimented with various approaches, ranging from pure classical image processing techniques to different types of machine learning. In this report we will present the various experiments he carried out and compare some of their results.

2 Background

This section will cover some of the general background theory which is used in our image deblurring algorithm.

[1] http://www.siam.org/books/fa03/FA03Chapter1.pdf

2.1 Classical Image Processing

Classical image process techniques are methods of manipulating images by treating them as matrices and applying matrix operations to them in order to deduce information and to alter the in a desirable way. Very early into the project we decided that approaching the deblurring problem from a purely classical image processing standpoint would not yield satisfactory results because of the nature of the images which we had been asked to deblur. These images are very small (around 50×180 pixels) and of fairy low resolution. Therefore they do not give enough information on the latent sharp image which we would want to reconstruct. We decided instead to focus our efforts on using machine learning techniques to tackle the problem and with this approach in mind we would need to construct somehow a training set from the 4000 'clean' images that we were given. The image processing techniques described in this section were used in order to achieve this result, i.e. in order to manage to artificially blur our clean images in a way which resembles the 100 images in the testing data set.

We will start with a short introduction of how images can be artificially blurred. This gives some insight on how we try to deblur them, as we assume that natural blurs can be approximated by artificial blurs.

In general, we consider only greyscaled images, thus reducing the dimensionality of the image from the three channels of RGB to one channel specifying the intensity of the black at each pixel. This does not only simplify the process of deblurring images but it essentially does not change the quality of the results, as a greyscaled image is equally easy to read by OCR software as an RGB one.

The fundamental idea here is that images are perceived as arrays of numbers, each specifying the intensity of black in a pixel. So we can treat an image as an two dimensional matrix $Im \in \mathbb{R}^{h \times w}$. The basis of artificial blurring is the use of convolutions. Convolution by definition is the point-wise multiplication of two functions. When applied to discrete domains, the definition is as follows:

$$g(i,j) = \sum_{(m,n) \in \Omega_{ij}} \sum f(m,n) h(i-m, j-n).$$

Where Ω is the convolution domain, $f(n,m)$ is the convolution kernel, $h(n,m)$ is the function (here representing the image) on which the convolution is applied.

Let us consider how this applies to our case. Perceiving an image as a matrix we also need have that the convolution kernel is also a matrix. In this case the convolution operation, which we denote by $*$ is a variant of matrix multiplication which in an intuitive way can be explained as replacing each pixel by a weighted sum of its neighbouring pixels, with weights given by the convolution kernel. So for example if we consider the following convolution kernel:

$$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

we see that the result of applying this operation on an image is that we replace each pixel with the average of all of its neighbours, excluding though the first two pixels in each of the border of the image (We will discuss this dimensionality reduction and how we dealt with it later). This is an example of *averaging blur*.

In general we only consider kernels of odd sizes, because the way the operation of convolution is preformed on a matrix involves replacing the pixel which 'falls in the centre' of the kernel with the result of the convolution. In this sense a matrix of an even size does not have a well defined centre so it is not useful for our purposes.

The result of any such convolution is, of course, also a matrix. By varying the convolution kernel $f(n, m)$, the resulting matrix have artificial blur effects.

To replicate blurred images seen in the testing data we were given, we consider only two types of blurs and consider each natural blur as a specific combination of these two blurs. More specifically the blurs we considered are:

1. Gaussian blur and
2. Motion blur.

Gaussian blur is the simplest of the two. Essentially the values in the kernel matrix are given by a Gaussian distribution. The values in the kernel are produced, informally, in the following way:

1. The value in the centre is the probability of the mean, so the highest probability in the distribution.
2. Values in the other cells are given by considering their distance from the centre and taking the probability of a value that many standard deviations away from the mean.

Motion blur is slightly more complicated. In general there are two parameters which we need to consider when dealing with such a blur, the magnitude of the blur and the direction of the blur. To replicate motion blur using a blur kernel we start by constructing a matrix of zeroes. We then replace the entries of a specific row with 1's, to imitate the effect of the blur. Finally we rotate the matrix by a specified angle.

Besides blurs we used additional classical image processing techniques to make the clean images resemble more the blurred ones. Some of the techniques we used were *image pixelation*, to achieve a lower definition effect on the images.

We did this by resizing the image removing random pixels from the resized image and then restoring the image to its original size. Further we preformed some *perspective transformations* which involved changing the relative sizes of parts of the images while maintaining straight lines. Also, we applied variants of *histogram equalisation*, which are transformations used to reduce the high contrast in our goal images we also varied the brightness of the images to deal with darker images. Additionally we needed to also consider *resizing*, since in the testing images we were given the relative size of the license plate varies.

2.2 Convolutional Neural Networks

Convolutional Neural networks is a flavour of feed forward neural network which uses convolution to generate the input for subsequent layers. It is more suited to the problem of image deblurring as each perceptrons utilises visual field information in generating the output. In comparison to general linear neural network, it places more emphasis on spacial homogeneity and variation. It differ to linear models as at each mode, the output is dependent on the convolution kernel which is updated in the same way as connection weights. Hence due the versatility of kernel structure as explained before, it excels in "learning" spacial structures for purposes such as image processing.

2.3 DCGAN

DCGAN stands for Deep Convolutional Generative Adversarial Network. DCGAN is a powerful tool which performs the best on image completion and generation of new images from an existing dataset. A DCGAN can be used for deblurring, although for the purpose of our project we have examined it primarily as a tool to extending our dataset by generating new images.

A DCGAN consists of a Generator (G) - a deep network that produces images from vectors of data and a Discriminator (D) - a deep network classifier that judges whether an input image is genuine or fake. Initially both G and D are extremely inaccurate, but they improve with iterations by playing a one step minimax game against each other. The goal of G is to produce genuine images that can trick D into believing that the images are from the original dataset and the goal of D is to accurately distinguish real images from fake.

3 Approaches Taken

From our initial experiments and after reading some papers on the topic, we quickly realised that the images we had to deal with were too small and had too much noise to apply classical image processing techniques. However, this gave us a good foundation to understand how images are manipulated.

We also decided not to use DCGAN's. The output we observed was far from resembling a license plate and, even though we believe that with enough training time we could have potentially achieved relatively good results, we decided not to use them. Moreover, one of the factors that influenced our decision was that we found a new and more promising way of multiplying the training set, described below.

In summary, we decided to build a convolutional neural network the evolution of which will be discussed in the following section.

3.1 Core Network

Initially, we had already experimented with a simple CNN and achieved good results. The core structure is the following:

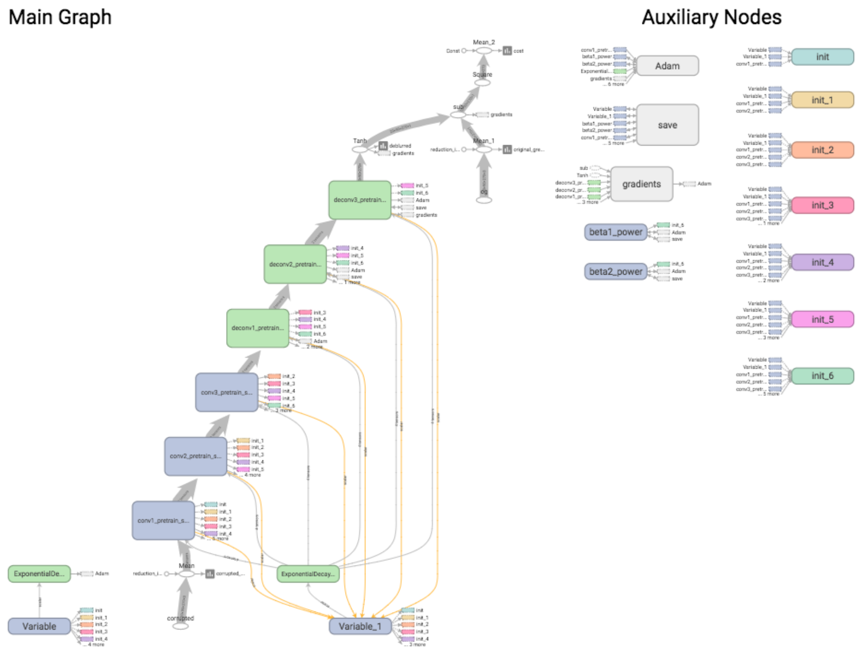

Figure 1: Network graph.

We apply three convolutions to the image, which reduce its size by a factor of 6 and then three deconvolutions which brings it back to the original size. The outer layers have 256 filters, the intermediate ones 128 and the central ones 64. This is an example of the filters that each of the convolutional layers learns:

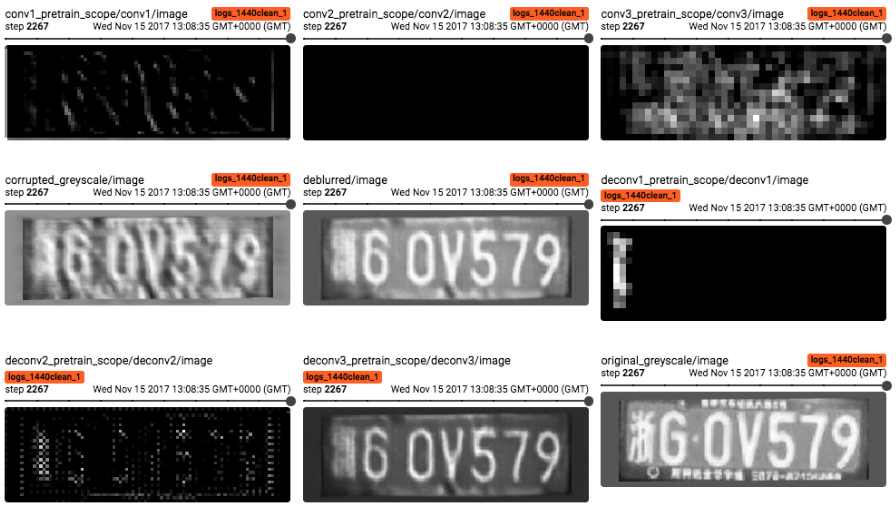
conv1_pretrain_scope/conv1/image
logs_1440clean_1
step 2267
Wed Nov 15 2017 13:08:35 GMT+0000 (GMT)
conv2_pretrain_scope/conv2/image
conv3_pretrain_scope/conv3/image
corrupted_greyscale/image
deblurred/image
deconv1_pretrain_scope/deconv1/image
deconv2_pretrain_scope/deconv2/image
deconv3_pretrain_scope/deconv3/image
original_greyscale/image

Figure 2: Convolutional layers output.

3.2 Data set augmentation and refinement

Since the generative part of the DCGAN was not a practical approach for efficient data generation we needed to find a different method to achieve generation of data. This was the case because we were not satisfied by the 4000 images in the training data set, as they were not general enough (e.g. most of them had the same Chinese character) further their quality was fairly low.

To this end we found a website in with a small number of very high definition Chinese license plates. From these we compiled a short set of around 60 images which we used as training data. As the data was of very high resolution the training not only deblurred the images but also increased their resolution.

That being said, 40 images are not of course a large enough training set so we needed to increase their number. For this we devised an algorithm which splits up clean license plates into their characters and produces all possible permutations of the characters, but keeping the Chinese character first. Using this method we managed to construct 720 license plates from each license plate. The results of using blurred license plates generated in this way instead of data from the 'clean' set were surprisingly satisfying.

Additionally we decided that we would not completely abandon the training set we were provided, but in order for it to be useful we had to somehow select the images in this set which were desirable goal images. Quite a few images in the set were blurry or very dark and having them as goal images for the training would result in a badly trained model. To do this we manually selected the cleanest images from the set, throwing away more than half of the images.

Using the even cleaner data set for training significantly improved our results.

3.3 Dropout layers

One of the first improvements to our naive network was adding dropout layers attached to every convolutional and deconvolutional layers. These will randomly select half of the weights (which we decided judging from examples we saw online) and not change them for that iteration. The advantage of doing this is avoiding overfitting. they have not affected our training time or performance significantly and, even though we don't have strong evidence to support this claim, we believe that ever since we added them, the model performed better on the unseen data.

3.4 Generative pre-training using stacked autoencoders

This is one of the most successful alterations to our initial model. The idea behind it is that, instead of starting the training session with no context, we pretrain each layer separately (with special focus on the first layer). By doing this, the network is initialised with some initial weights. Before it would take roughly a couple of dozens of epochs for the network to be at the state we currently start with. This pretraining is done with 40 images that we have hand picked because of they high quality. This makes the pretraining incredibly fast whilst at the same time ensures that the initial weights lead to sharp images. In the next subsection we show the results of this approach combined with dynamically decreasing the learning rate.

3.5 Decreasing the learning rate

Before, the learning rate was constant and it had a value of 0.001. Now, we dynamically reduce it every 1000 iterations. We use a stepped decay function, since that is a standard way of handling the learning rate in neural networks. Here we show the results obtained in the first long training session:

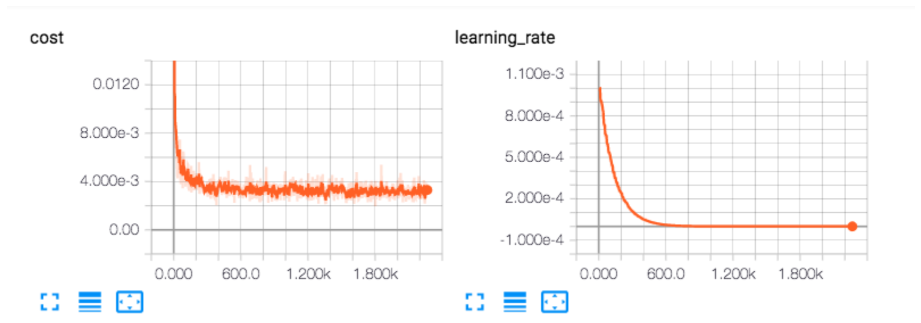

Figure 3: Cost and learning rate in a network with dynamically decreasing learning rate and pretraining.

The rest of the graphs we will present, all include the learning rate since we realised that having the right learning rate is crucial. This was our first experiment and we also noticed that we decreased it way too fast, which is not necessarily a bad thing but makes the last stage of the training much slower. This illustrated the idea behind it:

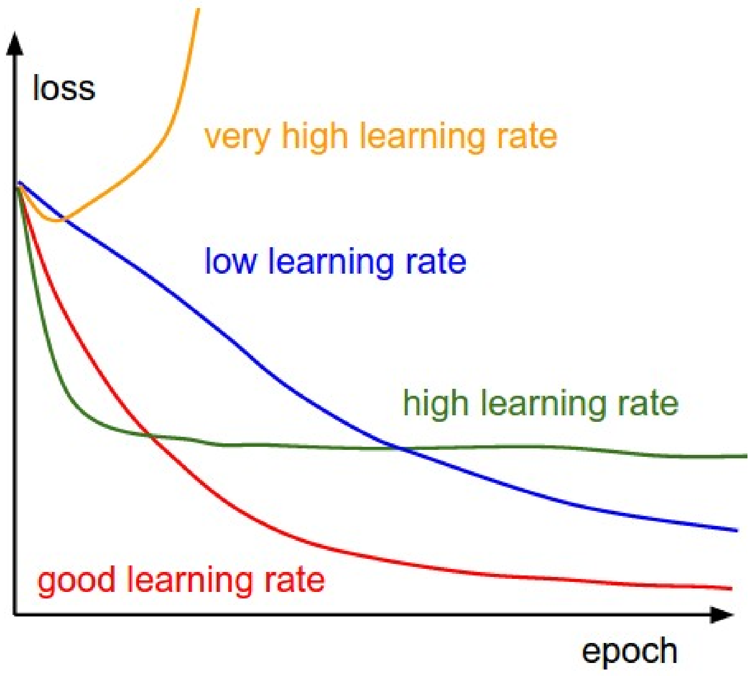

Figure 4: Best learning rate.

3.6 Fully connected layers

Looking a tour neural network structure, we thought it would be a good idea to add a fully connected layer after the last convolution and before the first deconvolution since those are the smaller code layers. These are the results of our experiment:

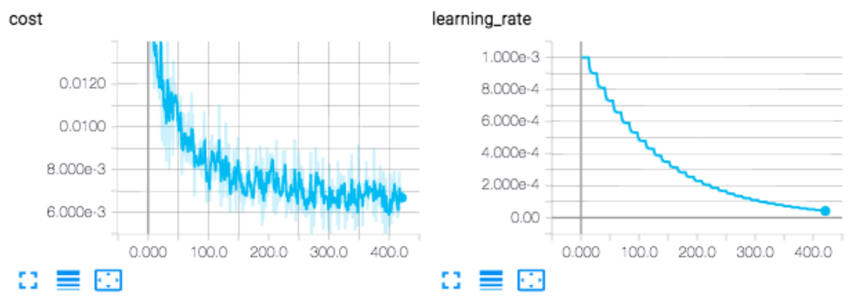

Figure 5: Cost and learning rate in a network with a fully connected layer in the middle.

As we can see, it makes the cost quite irregular and does not show any immediate advantage compared to not having this layer.

3.7 Number of channels increase

We also tried drastically increasing the number of filters in each convolutional layer. We thought that this way the algorithm would be able to learn more functions and be able to deblur a wider range of images. These are the results:

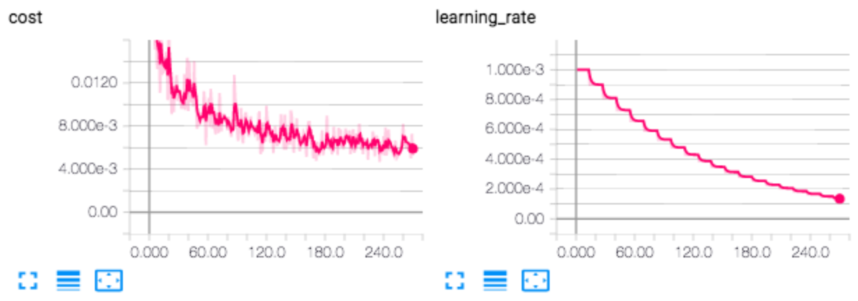

Figure 6: Cost and learning rate in a network with more channels per layer.

As we can see, the model finds it harder to learn. This might not be necessarily bad, however, we believe that our blurrer might not be that complex so that a large number of channels per layer makes sense.

3.8 Aggressive blurs

We also realised that the training data was quite corrupted, so we decided to add more adventurous blurs, in particular pixelation combined with random pixel removal. After training the model, we observed:

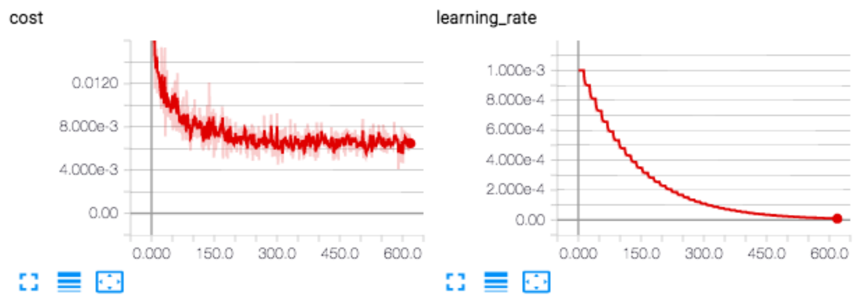

Figure 7: Cost and learning rate in a network with a more aggressive blur.

Clearly, the model finds it harder to learn these blurs. However, it ends up achieving quite good results as we can see here:

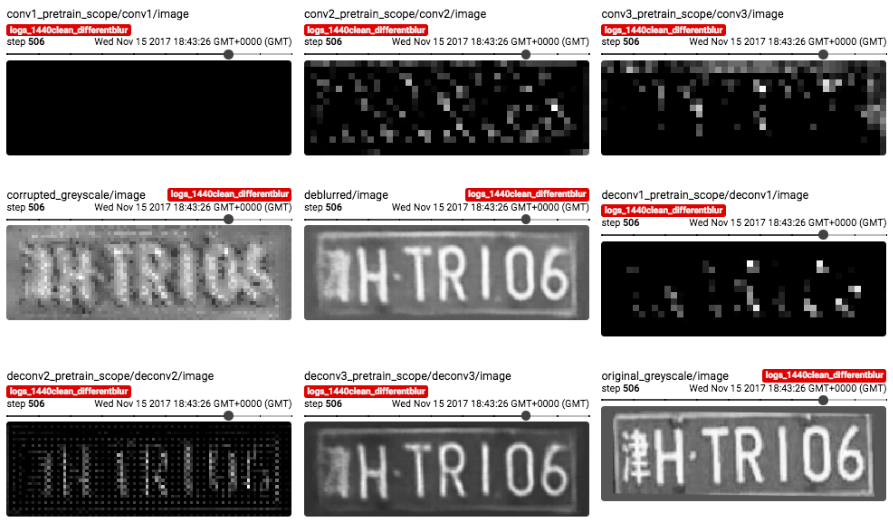
conv1_pretrain_scope/conv1/image
conv2_pretrain_scope/conv2/image
conv3_pretrain_scope/conv3/image
corrupted_greyscale/image
deblurred/image
deconv1_pretrain_scope/deconv1/image
deconv2_pretrain_scope/deconv2/image
deconv3_pretrain_scope/deconv3/image
original_greyscale/image
logs_1440clean_differentblur
step 506
Wed Nov 15 2017 18:43:26 GMT+0000 (GMT)

Figure 8: Layers with an aggressive blur.

4 Evaluation

After evaluating separate metrics for each technique, we can conclude that, for the purpose of image deblurring the 3 approaches, namely: data set augmentation, generative pre-training and decreasing the learning rate have improved the model, whereas fully connected layers did not lead to a noticeable improvement. Dropout layers have led to a significant improvement.

As we mentioned previously augmenting and refining the data set did yield better results, but the results are purely subjective, in the sense that now our deblurred images look better and are more readable. Below is an example of the output from the blurrer with the old dataset and the new dataset:

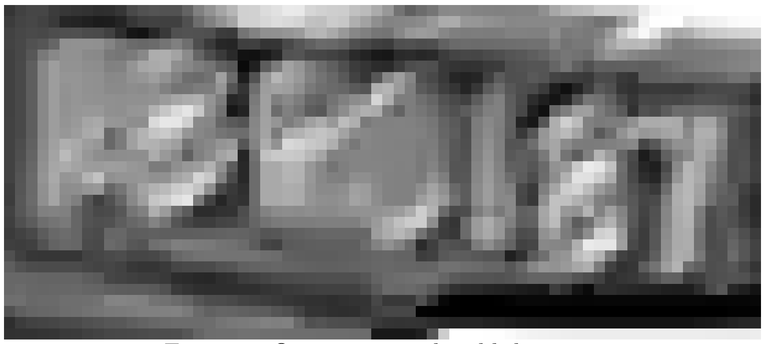
Figure 9: Output using the old data set.

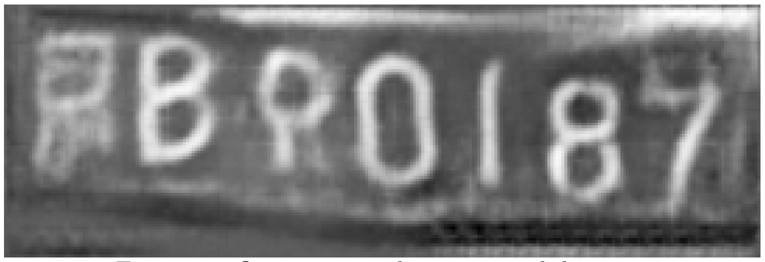
Figure 10: Output using the augmented data set.

The difference between the two is clear but since we have not yet been able to use Huaweii's OCR library, and because the size of the images is very small we have no way of quantifying the blurriness of an image.

Part of the reason why we chose to experiment with DCGAN's was to take advantage of the discriminator network they contain in order to generate license plates to be used as training data later. DCGAN's have displayed outstanding performance on generating landscapes and bedroom images. However, they perform less well on human faces and, most importantly as far as our application domain is concerned, alphanumeric characters. We have therefore decided to seek alternative approaches to extending out dataset.

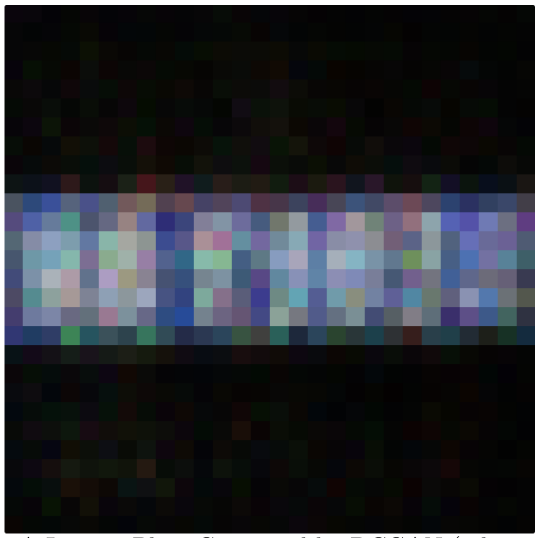
Figure 11: A License Plate Generated by DCGAN (1 hour training).

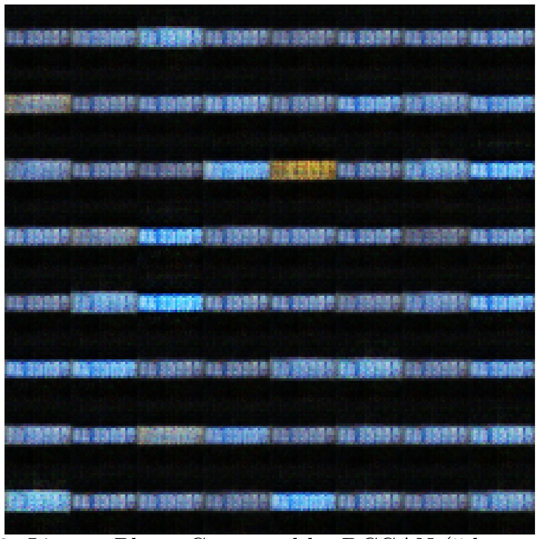
Figure 12: License Plates Generated by DCGAN (5 hour training).

Above are examples of a ‘license plate’ generated by our DCGAN after an hour of training. The obvious lack of efficiency combined with the fact that our handwritten approach to data generation was fairly satisfactory drove us away from the possibility of using a DCGAN.

5 Conclusions

Judging from all the results that we have presented, we decided to include the following in our final model: droputs, pretraining, a better data set (both bigger and with higher quality), a more aggressive and varied blur and a dynamically changing learning rate. Before the competition we will consider adding more layers and augmenting the number of channels even though the current results are not that good. In general, the results of our experiments show that in general using a CNN as a black box without any awareness of the internal workings of the network in not a valid approach and that results will very quickly improve if one tries to be more domain specific.

We also saw that ML is not the solution to everything and often handwritten approaches can provide quick solutions to problems that a machine learning approach would struggle to solve. In the end, the results of our deblurring algorithm on the testing data show that our approach of trying to simulate specific blurs is a valid approach and can lead to good results.

Before the competition we will work mainly on the final data set and the final blurring that we will use since we are quite confident about our model. We do not discard slight modifications to the model. Finally, once all of this is done, we will train the model for as long as we can before the competition. assuming the training is resulting in positive gain.