

# Image Deblurring - Checkpoint 1

Group 8

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

## 1 Original Planning

The following plan is the release plan for the first sprint as we presented it in our initial *Methods and Plans* document:

Sprint	Objectives
09/10 - 20/10	<ol style="list-style-type: none"><li>1. Meet with the Huawei competition organisers and obtain the data set.</li><li>2. Research existing technology for classifying image blur types.</li><li>3. Research existing technology for image deblurring.</li><li>4. Ascertain which resources are required.</li><li>5. Experiment with some basic image (de)blurring techniques.</li><li>6. Set up our continuous integration tools and servers.</li><li>7. Build a simple website to output our results.</li></ol>

## 2 Organisation

As we stated in our Methods and Plans we started a GitHub repository. Not very active at the moment since the project is at a research state. Here is what the network looks like:

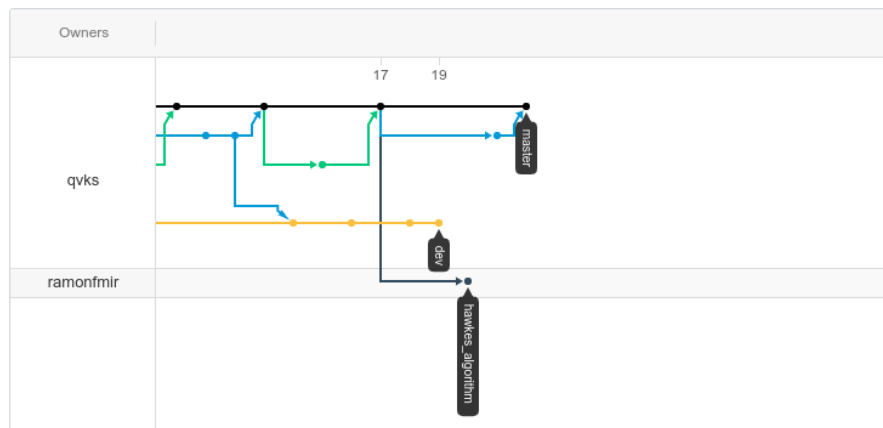


Figure 1: Github network.

And this is the current state of the repository:

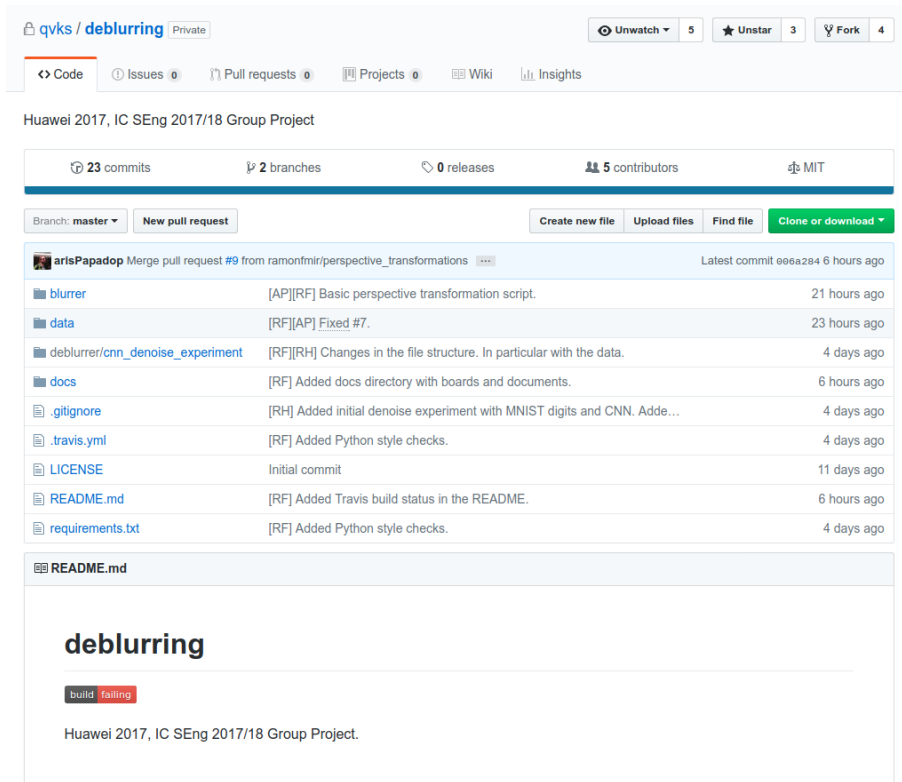


Figure 2: Github repository.

Every time we push to our git repository, the Travis build starts. We don't have any pipeline at the moment but that is our priority number one for the next sprint. For the moment, we only run Python style checks. We are not forcing them to pass, since the code we are writing is just experimental. Here are screenshots of our .travis.yml file:

```
1 language: python
2 sudo: enabled
3
4 # Install dependencies.
5 install:
6   - pip install --upgrade pip
7   - pip install -r requirements.txt
8
9 script:
10  # Style checks.
11  - for f in $(find . -name '*.py'); do pycodestyle --first $f; done
```

Figure 3: Travis YAML.

Here is an example of one of our builds:

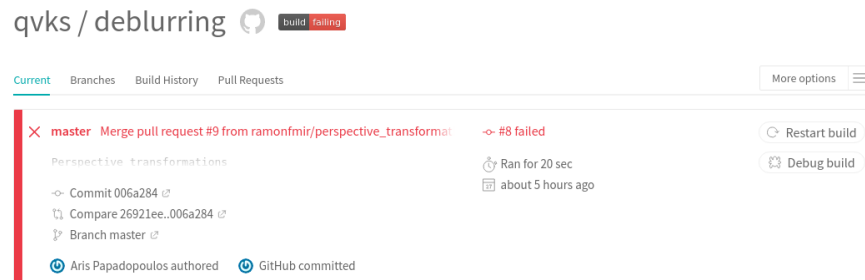


Figure 4: Travis build.

We used Trello to organise our task board. Here are three screenshots of the state of the board at different points in time in the sprint:

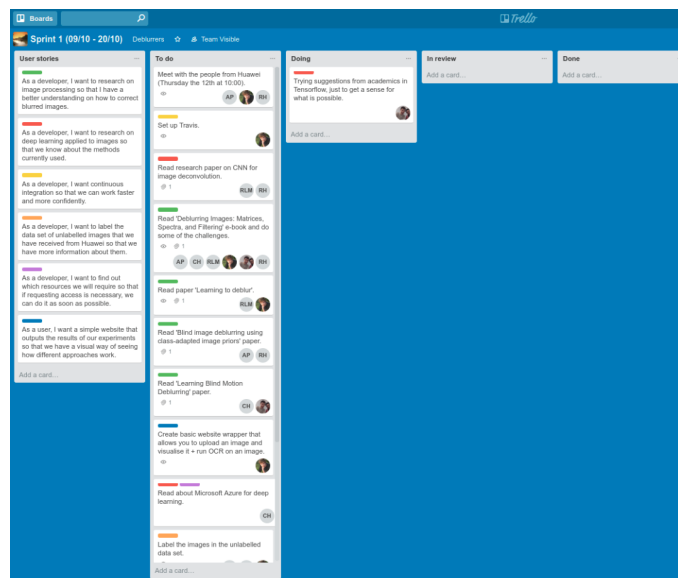


Figure 5: Trello board after the sprint planning.

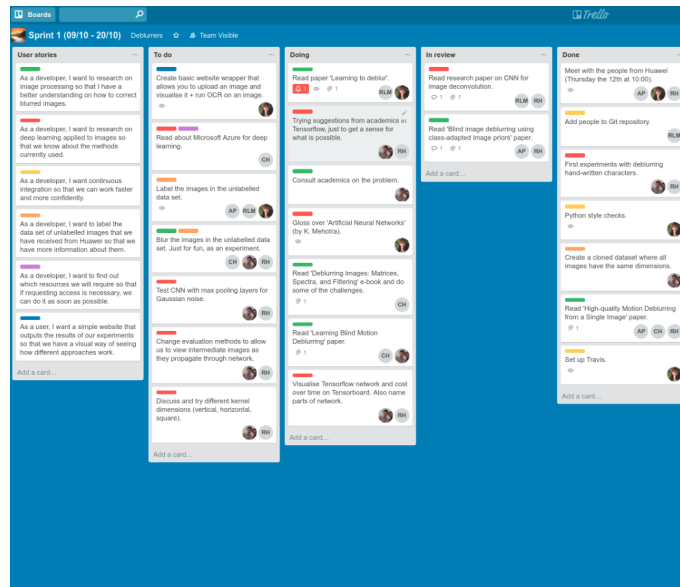


Figure 6: Trello board in the middle of the sprint.

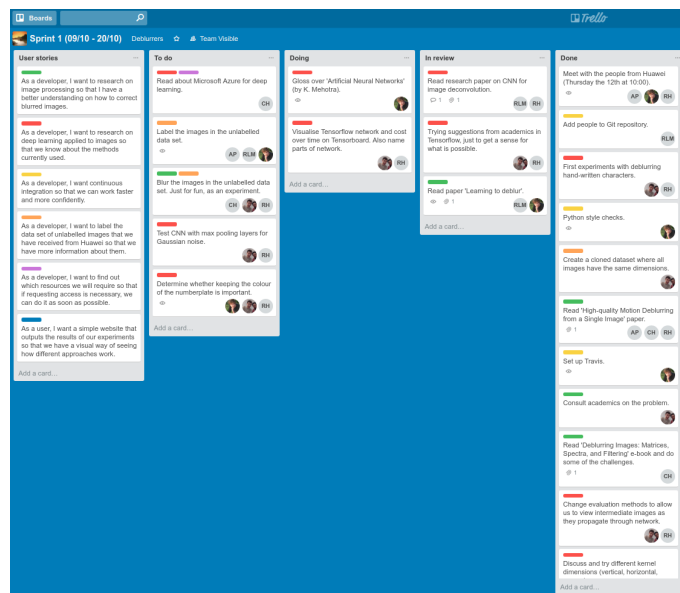


Figure 7: Trello board at the end of the sprint.

Finally, we used Slack to organise our daily stand-ups in the days that not every member of the group could be present in person for a meeting at the allocated time. We present here an example of how these daily stand-ups worked:

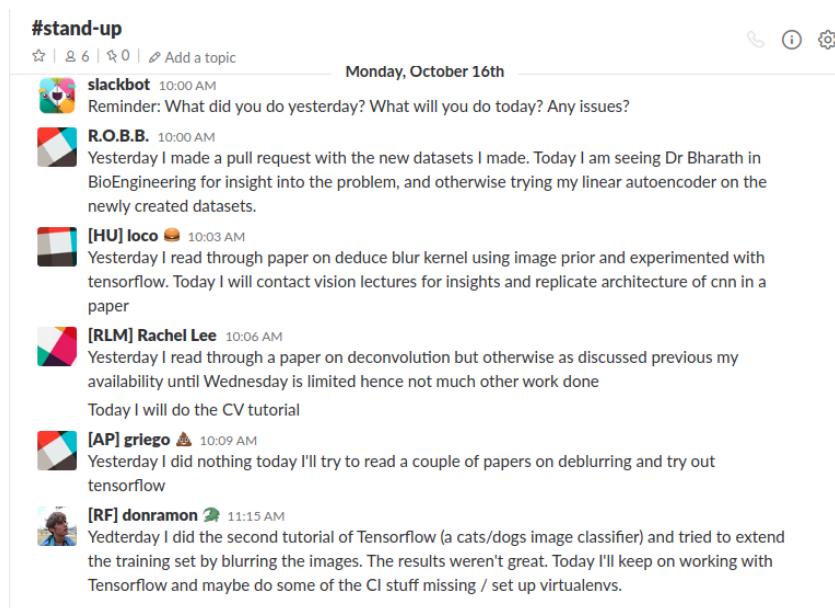


Figure 8: Virtual stand up meeting.

## 3 Tasks Accomplished

### 3.1 Sprint Overview

Keeping the numbering of the tasks planned for the first sprint from the table in page 1, in this sprint we did the following:

For 1, a meeting between the head of the Machine Learning Department of Huawei and the two teams which are implementing the same project, namely group 8 and group 16, was arranged on the 12th of October. In the meeting various aspects of the competition were discussed and we have kept in touch with Huawei for additional information on the project.

For 2 we read a list of papers and books on the subject matter, these include the following:

- Deblurring Images: Matrices, Spectra, and Filtering
- High-quality Motion Deblurring from a Single Image
- Learning to Deblur

This list is incomplete, but these are the main papers on Image Processing that have influenced the approach we will take in the next sprint. We have also read some papers about neural networks for deblurring, from which we would highlight Deep Convolutional Neural Network for Image Deconvolution.

For 3 we tried out various existing tools to get a sense of the overall quality of existing implementations and a better understanding of what our results should be like. These tools include:

- MATLAB. It was useful for the initial experiments with image processing.
- OpenCV. We learned about this very powerful Computer Vision library, in particular about the image processing part of it.
- TensorFlow. We did some of the tutorials and played with it by building our own naive CNN to deblur images.

The research described above lead us to the following conclusions, regarding the tools we would require to use for the project, as stated in 4:

- Image processing is not enough to tackle this problem. The images are very small and noise plays an important role so we cannot use techniques based on a single image.
- We will use image processing techniques to augment the data set and also as an aid in the training stage. All of these will be done using OpenCV.
- A convolutional neural network will be the core of our algorithm for deblurring. And we will use TensorFlow to build it. It is essential to train the network using different kinds of blur, detailed in 5, due to the differences in noise distribution in varying types of blur. In section 3.2 we explain the current state of our neural network.
- Even though this is our initial approach we want to try out other more adventurous ideas and see if they lead to better results. Some of these are splitting the alphanumerical sequences and training with characters or auto-generating high quality license plates as training data.

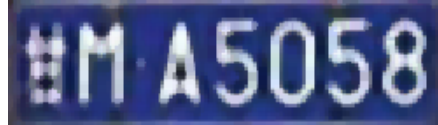
For 5 we experimented with various image blurring techniques.



Figure 9: Original Image.



(a) Averaging blur.



(b) Median blur.



(c) Bilateral blur.



(d) Motion blur.

Figure 10: Different Blurs

For 6 we set up the TravisCI continuous integration server which for now only runs Python style checks every time someone pushes to master.

As far as 7 is concerned we decided to shift our approach from a website to an Android application. We present here some of the mock-ups:

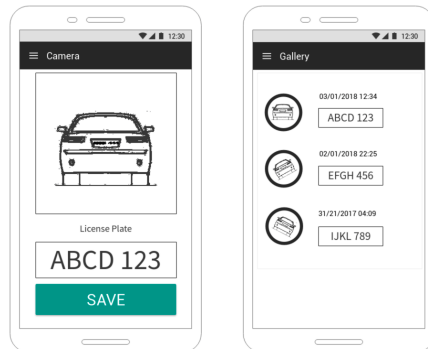
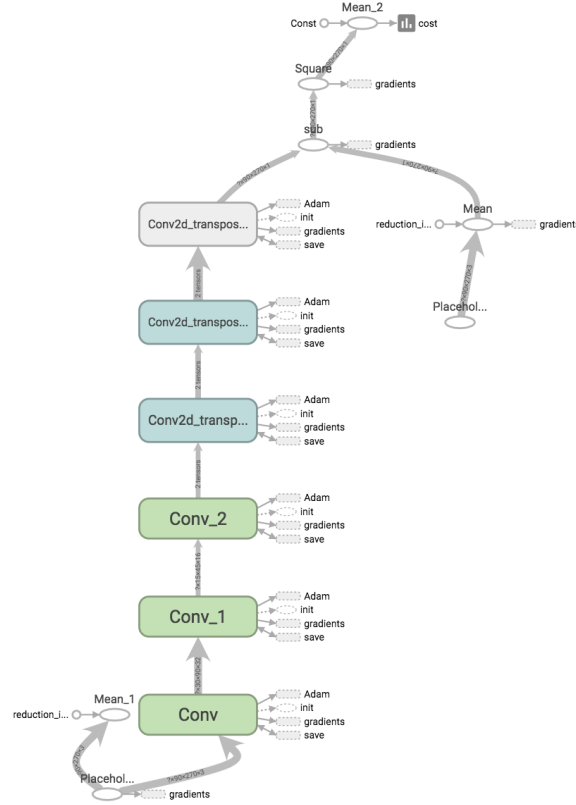


Figure 11: Android app mockups.

### 3.2 Current CNN Iteration

Below is our first attempt to build a CNN for image deblurring in Tensorflow. This graph was generated by Tensorboard; a tool we found during our research. It was extremely useful in debugging out implementation of the network.



This defines our skeleton for the state-of-the-art autoencoding CNN. We start by applying a Gaussian blur to our original image and then converting it to grey scale (individual colour channels contain little additional information, and any reduction in dimensions reduces training time). Then the image is propagated through three sequential convolution layers, each with a set number of kernels with varying strides and dimensions. Similarly, it is then passed through three transpose convolutional, or *deconvolutional*, layers. The resulting image can then be compared with the original image, a cost (MSE) calculated and the kernel weights can be adjusted appropriately with Tensorflow's Adam optimizer.



### 3.2.1 Data preparation

To train our CNN a stream of data is required. Our current data preparation system loads the entire unblurred image data into an array and blurs them using our blur model. This is fed into the CNN in batches. This current system does not use modern techniques such as random shuffling the array which will benefit the learning rate. As our sample data sets are images of different size, we standardise the size of image to  $270 \times 90$  pixels by padding the expanded space with black. This will keep the aspect ratio of the image and prevent blurring from preprocessing.

### 3.2.2 Results

This CNN is a skeleton trained on CPU, and does not take advantage of max-pooling and other beneficial architectures. After training on the data set for a number of iterations of the dataset we obtain the following results:

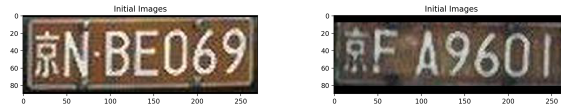


Figure 12: Original clean image from train set

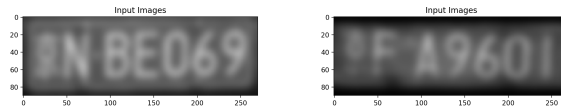


Figure 13: Blurred and grey scaled input to network

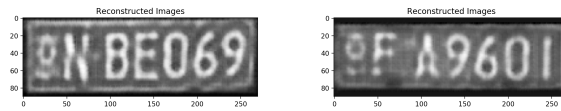


Figure 14: Network output

These are the results that we expected to reproduce in Sprint 1. The simplistic / experimental CNN has effectively learnt deblurring filters that we could have created ourselves. For sprint 2 we plan to extend this model and train it with a

collection of different noising techniques. For example, it should be noted that the applied Gaussian blur does not accurately mimic images blurred and noised in the real world. The comparison below illustrates this discrepancy:



(a) Simple blur method we used.

(b) Real Life example.

Figure 15: Simplistic / Unrealistic Blurs

## 4 Project Plan

We will have our sprint planning meeting on Monday, but these are the main objects we want to achieve:

1. Have an end-to-end pipeline that keeps track of performance metrics of every build so that we know which direction to move to.
2. Have a basic Android app to take a picture and process it.
3. Keep working on our current neural network and try out different blurring techniques on the training data to make it better.
4. As mentioned before, try riskier approaches and see if they lead to good results.