Barcode Detection

Version 0.0.1

This document describes the implementation approach used in the version identified above.

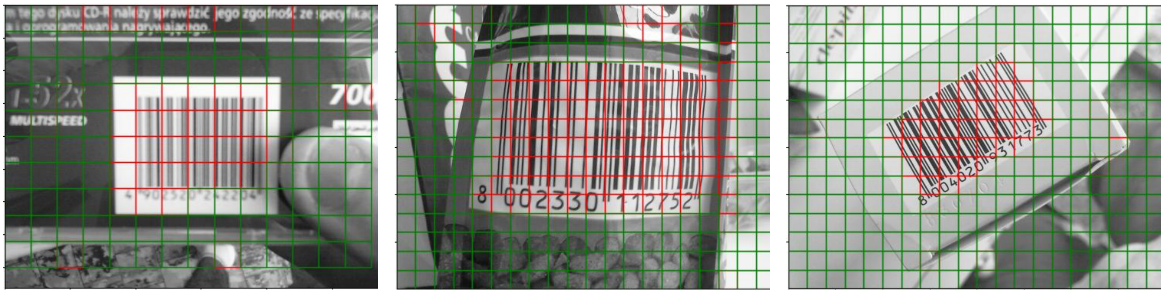
Current Approach

Step 1: Barcode Region Localization

Split image into a grid layout and classify each square patch as either part of a barcode (Class-1) or background (Class-0). See below for details on how the classifier was trained. The images below show the classifiers output. Red squares are detected as Class-1 (barcode region) and green squares are Class-0 (background).

Three different patch sizes (64, 80, 128) were tried; results were similar for patch sizes 64 and 80, which were better than the results for patch size 128

Possible alternative approaches for this step could be to use a RetinaNet like object detection models or Hough voting based approaches. These are discussed below in the Design Choices section.

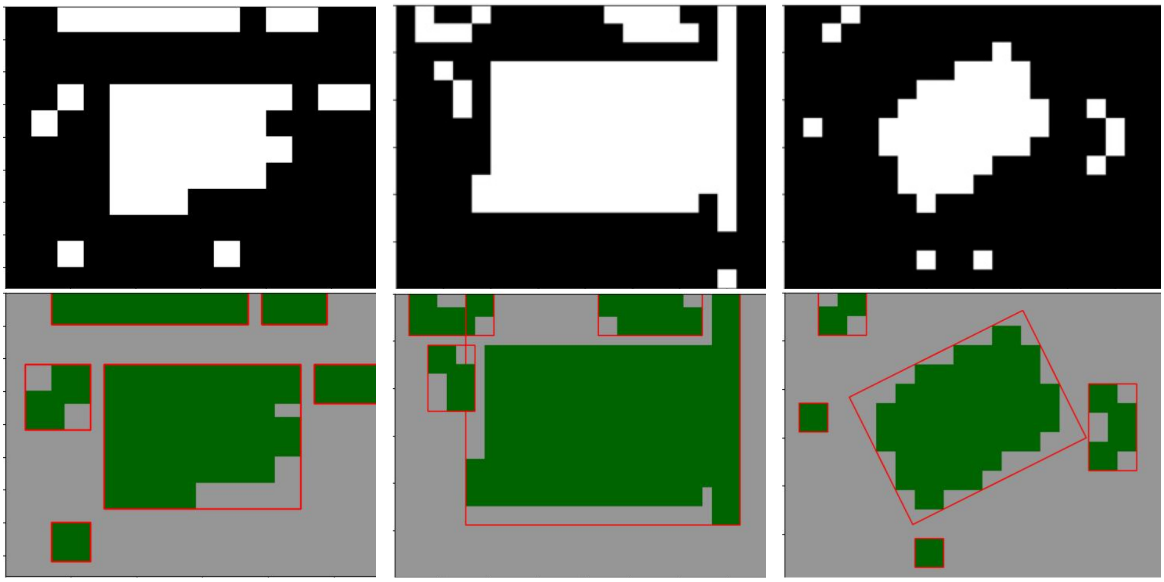


Step 2: Barcode Region Expansion

2a. A binary mask is created using Class-1 patches
2b. Morphological dilation is performed to connect patches
2c. Connected Components Analysis is performed to detect candidate barcode regions
2d. Regions are filtered based on area, to keep only those with area > 2500

The first row of images below shows the binary mask extracted from the classifier output shown above. The second row of images shows the identified connected components and the rotated bounding boxes that contain them.

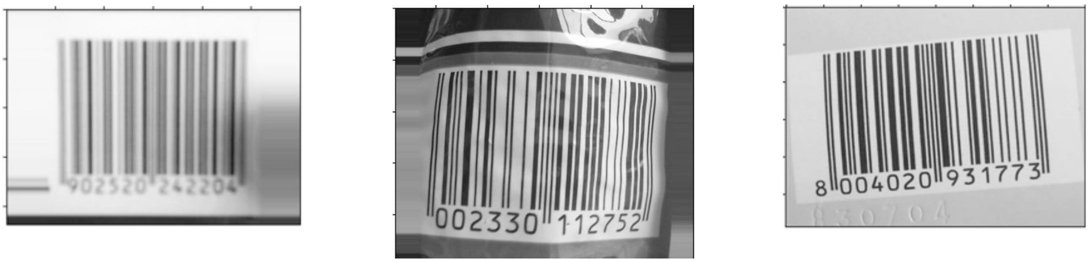
One additional step that could be introduced in future versions would be to train a binary classifier to classify these connected components as barcode or non-barcode regions. These regions being much larger, capture more contextual information.



Step 3: Barcode Region Extraction and Transformation
3a. For each candidate region compute the a tight rotated bounding box
3b. Rotate barcode region and bounding box so that bars are vertical and bounding box is axis aligned
3c. Crop axis aligned bounding box region and decode data

The images below show the extracted barcode regions for the three images. Note that for each image multiple regions are extracted, which includes these regions. The current implementation is able to extract the barcode region even for out-of-focus (left), non-linearly distorted (middle) and rotated (right) images.

The current implementation does not focus too much on the barcode decoding aspect (once the barcode region has been extracted). Defocus images and non-linear distortions require special attention. See the Challenges and Improvements section for further discussion on this.

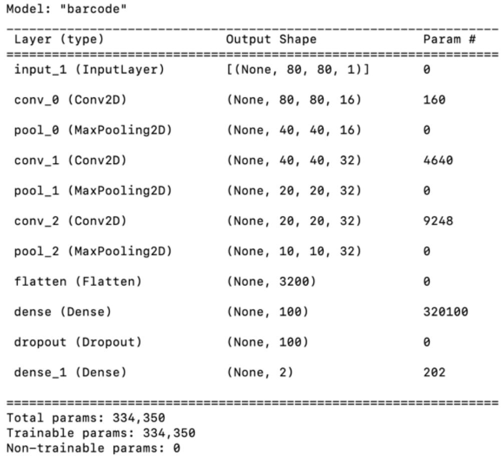


Supervised Training

- The classifier was trained in a supervised manner
- 33 randomly selected images from both datasets (with AF and without AF) were manually labelled
- These were split in 22 training and 11 validation images

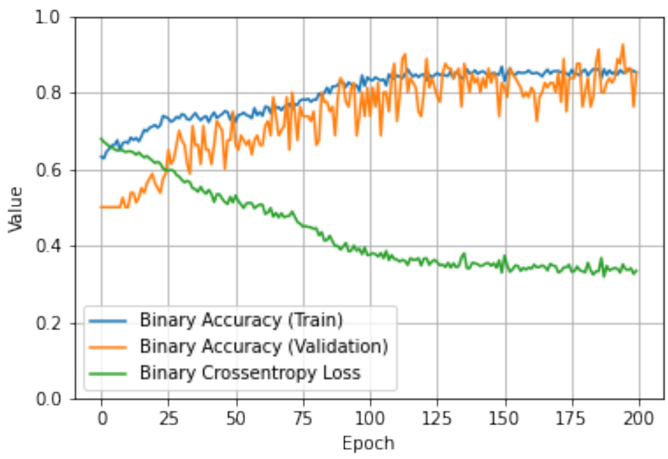
- For each training image 9 augmented (randomly transformed) versions were added, resulting in a total training set of 220 (22 x (1 original + 9 augmented)) images. Data augmentation was implemented using the following repo: https://github.com/Paperspace/DataAugmentationForObjectDetection. Each augmented image undergoes a random combination of rotation, translation, scaling, sheering, horizontal flip and colour re-mapping in the HSV space.
- Each training image is then split into a grid layout as shown above and patches that have at least 50% of their area inside a human-labelled barcode region, are assigned label 1.
- During each training step, balanced batches of 40 patches (20 of class 1 and 20 of class 0) were sampled from 4 images. That is, five class 1 and five class 0 patches were extracted from each image. Images were shuffled after each epoch. Each image participated once in an epoch and therefore in 200 epochs, ~2000 patches are sampled from each image.
- Relatively simple models with approximately 200,000 to 350,000 trainable params (depending on patch size) were used. The architecture for patch size 80 x 80 is shown below
- The model was trained for 200 epochs and training and validation accuracies were as shown below. The model achieved ~85% training and validation accuracies.

Model Architecture



```
Model: "barcode"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 80, 80, 1)]       0

 conv_0 (Conv2D)             (None, 80, 80, 16)        160

 pool_0 (MaxPooling2D)       (None, 40, 40, 16)        0

 conv_1 (Conv2D)             (None, 40, 40, 32)        4640

 pool_1 (MaxPooling2D)       (None, 20, 20, 32)        0

 conv_2 (Conv2D)             (None, 20, 20, 32)        9248

 pool_2 (MaxPooling2D)       (None, 10, 10, 32)        0

 flatten (Flatten)           (None, 3200)              0

 dense (Dense)               (None, 100)               320100

 dropout (Dropout)           (None, 100)               0

 dense_1 (Dense)             (None, 2)                 202

=================================================================
Total params: 334,350
Trainable params: 334,350
Non-trainable params: 0
```

Training Performance



Results

Patch Size = 80 x 80

Dataset1 (with AF) barcode extraction accuracy is **0.33**

Dataset2 (without AF) barcode extraction accuracy is **0.069**

Patch Size = 64 x 64

Dataset1 (with AF) barcode extraction accuracy is **0.353**

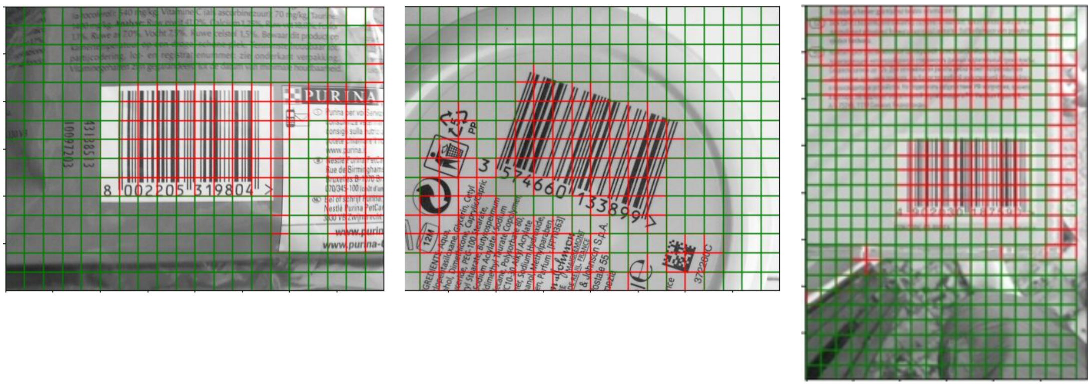
Dataset2 (without AF) barcode extraction accuracy is **0.10**

Here, barcode extraction accuracy is the proportion of images for which the correct barcode data was decoded.

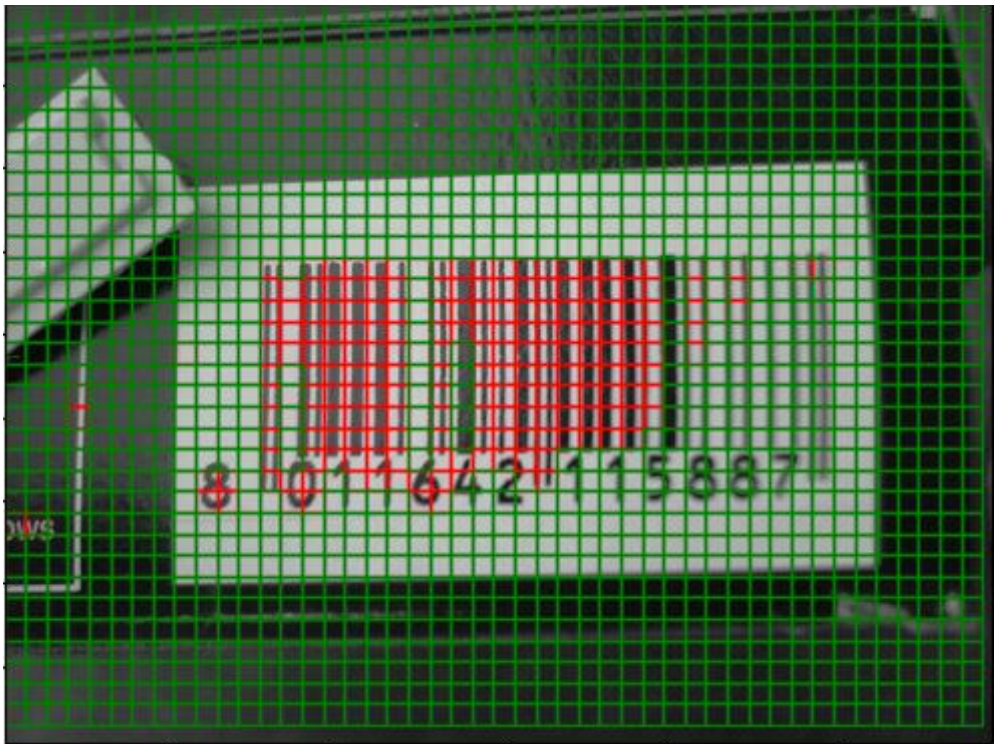
Overall, the current implementation rarely fails to detect at least some part of the barcode region. However, it detects a lot of spurious regions (particularly text) as barcode regions. Therefore, the model has low false negatives and high false positives. See the Challenges and Improvements section for further discussion on this.

Challenges and Improvements

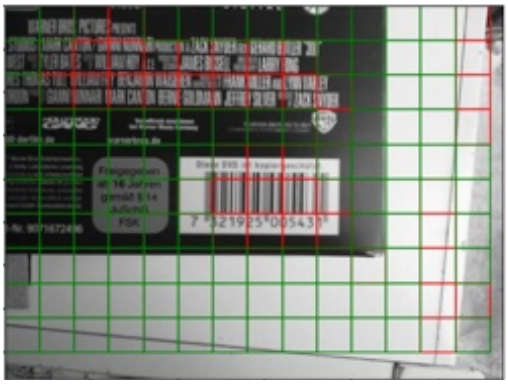
1. Misclassification with text regions
   The images below show examples where the classifier gets confused between regions containing text and barcode. The current classifier is trained on labels from only one class (barcode). One way to deal with this misclassification could be to label non-barcode regions as a negative class. Another alternative could be as mentioned above to train a classifier that classifies these larger regions. It may also be possible to use MSER and Hough voting based approaches to discriminate between text and barcode regions. One such approach is described in the Design Choices section below.

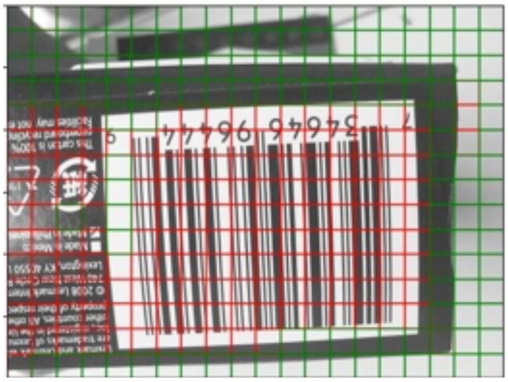


2. Low Coverage
   One major challenge with the patch approaches is that the detected regions may not completely cover the barcode region, as shown in the image below. Possible remedies for this could be to iteratively expand the detection region by adding neighbouring patches that have a higher likelihood of being a barcode region (based on classifier score and/or features of MSERs within these patches).



3. Accounting for Scale
   In the current implementation only a single, heuristically optimised, grid resolution is used at any time. The grid resolution however should be based on image scale. The figure below shows two images with very different scales. Such scale variations require multiple grid resolutions. At higher grid resolutions, a patch may not capture sufficient context and within a very local region, barcode and text can appear similar. Therefore, it can be useful to consider a hierarchy of grid resolutions and judge whether a region contains barcode based on the consensus across the hierarchy. An alternative would be to over sample patches ranging in sizes and aspect ratios, as is done in most region proposal based object detection models.





4. Out-of-focus images
   Deblurring defocused images is a widely studied problem in Computer Vision and various generic approaches have been developed for the same. The most simple such solution would be to train deblurring autoencoders or UNet like networks that can be trained in an semi-supervised manner by simulating defocus on sharp images.

Barcode detection itself being a widely studied problem, several approaches specific to noisy barcodes have been developed. These often rely on the fact that a section of a barcode that encodes a single numeric value has a specific pattern, and the fact that there are a finite number of such patterns can be used to reconstruct the blurred image.

5. Sampling to improve decoding accuracy
   In the current implementation, the entire barcode image is passed to the zbar library to extract the code. However, it may be useful to sample multiple strips from the barcode region and get multiple codes. The consensus of these can be used as the final result. This step can make the approach more occlusion invariant.

6. Non-linear distortions
   Perhaps the most challenging scenario is to decode barcode regions that have undergone significant non-linear distortion. One possible approach to correct for this distortion could be to use the fact that barcode regions have many corners, which when not distorted are in straight horizontal lines (top and bottom). Align corners detected in the barcode to ideal grid lines can be used to correct the non-linear distortion. Point pattern matching algorithms like the Iterative Closest Point (ICP) algorithm that compute piecewise affine transformations can be used to achieve this.

Design Choices

1. An alternative approach that uses traditional CV methods was also studied in this investigation. The steps involved are:
   a. Extract MSERs from the entire image
   b. For each MSER extract features such as mean pixel intensity, area, and aspect ratio.
   c. Filter MSERs based on these features to keep only those that have long bars
   d. Dilate these regions to get larger connected regions. These are proposed barcode regions.

   This methods, with heuristically identified thresholds for the three features, correctly identified barcodes in 6% of images. To achieve more practical results two approaches can be taken:
   a. Train a supervised classifier to classify MSERs based on these feature
   b. Use a Hough voting like mechanism (which would include MSER's angle as a feature) to find regions in the Hough space that represent MSERs belonging to barcodes. This approach would require fine-tuning of bin sizes in the Hgh space.

   One advantage of deep learning based methods over traditional CV based method is that most models when running on GPUs are extremely efficient during inference. Whereas not all traditional CV algorithms have versions optimized for GPU.

2. Alternative choices for supervised barcode region detection
   The most obvious alternative supervised deep learning based approach would be to use object detection models like RetinaNet. In the current implementation, the rationale behind using a custom patch based network was to be able to detect non-axis aligned bounding regions. Axis aligned bounding regions detected by standard object detection models can be much larger than a rotated barcode. The primary advantage of region based models is that they are occlusion resistant

3. Decoding using deep learning
   Finally, one alternative approach is to use and end-to-end deep learning based solution where the model not only extracts the barcode region but also predicts the code given a barcode region.