# Technical Report

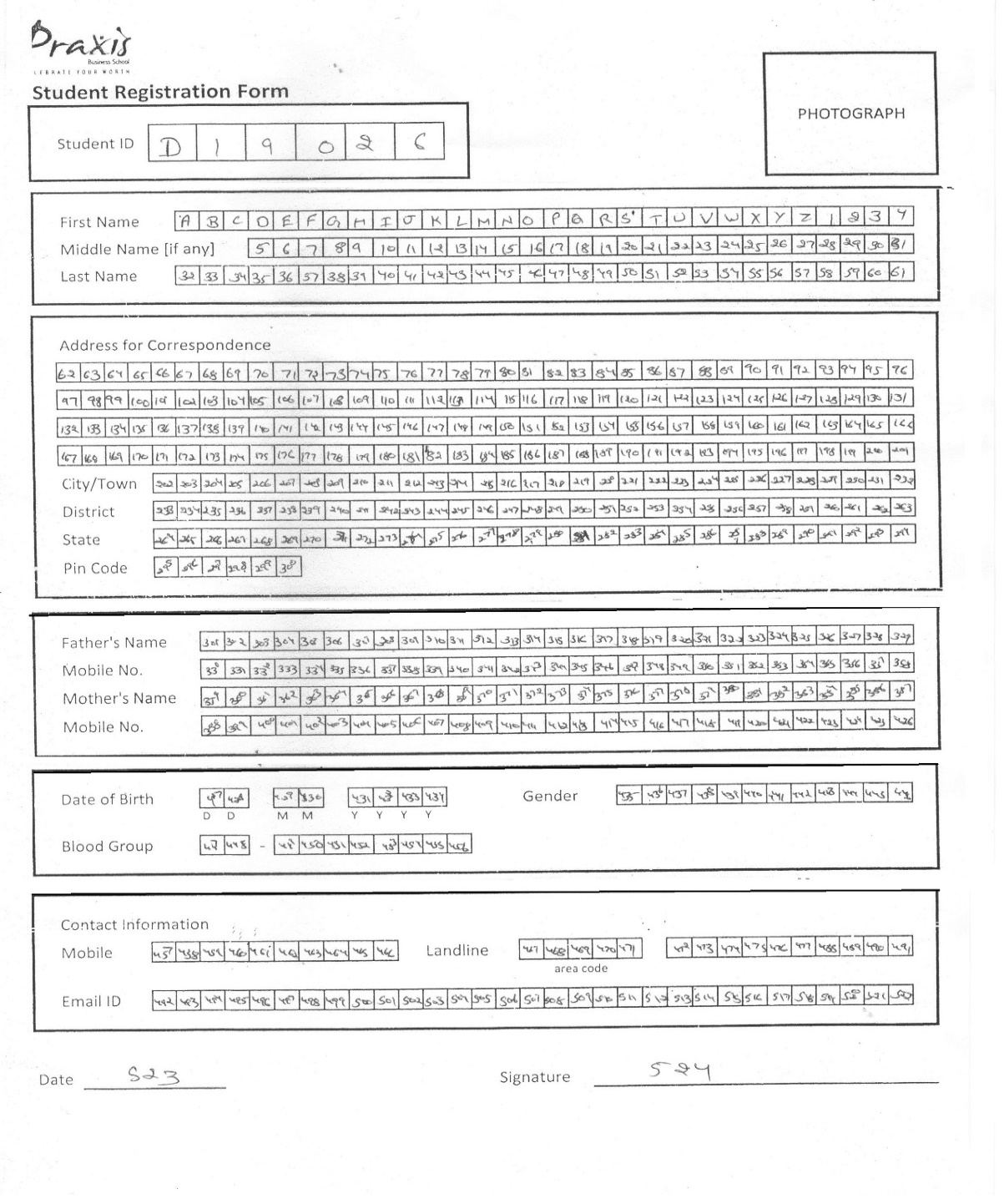
## Handwritten Intelligent Character Recognition: Isolated OCR for manually filled forms

# Introduction

In today’s time also at many places like bank, insurance, colleges paper forms are used to collect information which are later entered manually in system. Handwritten OCR detection method can be used to automate the entries for these forms. So, the purpose of our project is to make a program which can convert handwritten forms into digital format and reduce manual work involved.

## Assumptions

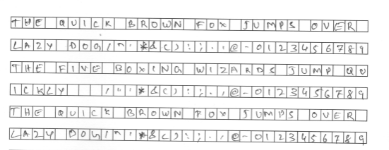
1. The characters considered are English letters in Capital, number and special symbol.
2. All characters are in rectangular box as shown below



1. The boundary of all rectangles are properly printed i.e. there is negligible missing gap in the lines.
2. The document is scan with min 300 DPI quality

## Data Collection

Data is collected by giving students studying in Praxis Business School, Bangalore a form and asked to filled the form in a particular format. Snippet is shown below.



The form is made to filled by 80 students 2 times at different interval of time to avoid bias as much as possible. Later data augmentation technique is used to increase the data size.

For Experiment purpose, EMINST dataset has been also used available online.

## Experiments

Date: 03-02-2020

### Problem - To Detect Edges

**Experiment No. 1** - Used Canny Edge library to detect edges of the box with lot of tweaking in the hyperparameter.

**Findings** - All 4 edges and their coordinates were not being detected, even with padding, Gaussian blur and making boundaries bold.

**Experiment No. 2** – Used OpenCv cv2.getStructuringElement Separated horizontal and vertical lines and then tried to get all box by combining them.

**Findings** - Some Lines not able to detect causing many box remain undetected.

**Experiment No. 3** –Used OpenCv cv2.getContour to detect almost all box but unwanted small and parent box also getting detect.

**Findings** – Able to get almost all box but unwanted box need to be filtered and original image de-noising required to get undetected some of undetected box.

Date: 06-02-2020

### Problem – Not able to extract all the boxes in the image due to noise.

**Experiment No. 1** - Used Threshold and Gaussian, binary filter

Findings – Number of contours detecting increased but some still not able to detect. Tweaked using various size of filters varying from 9 pixels to 25 pixels

**Findings –** In each filter different box remaining undetected. So, need a program to concatenate all the filters results.

**Experiment No. 2** – Loop of various size and type of filters concatenated like Gaussian, Binary and dilate. Unique set of contours taken and saved only if all contours detected.

**Findings**- Able to detect mostly in all scan images and program failed in some if outer boundary is not proper or scan images has less than 300 dpi quality. So, collected data or forms should have proper boundary for better accuracy and dpi should be > 300.

Date: 08-02-2020

### Problem – Filter of Unwanted Images

**Experiment No. 1** - Used conditional filtering of images. Removed too much small images of size < 30 or greater than 100 pixels so that specific height and width images only will be extracted.

**Findings**: We are able to filter only images that are required

Date: 10-02-2020

### Problem – To Sort all contours and tilt issue

Experiment No. 1 - We wrote a python function using minAreaRect function in CV2 library which gives the minimum angle made by the largest rectangle of information with the vertical axis. We rotated back the angle given by the minAreaRect function to make tilted image normal.  
Findings: We are able to realign the images which are tilted up to 5 degrees.

Git-Hub:  
<https://github.com/kedarkumargolla/OCR_PROJECT/blob/master/DE-TILTE.ipynb>

**Experiment No. 2** – Took last contour coordinate and stored each line in sorted order varying y pixels’ margins line by line.

Findings – Able to sort all contours in the required order and able to handle tilt upto 10 degree.

Date: 13-02-2020

### Problem – Labelling images so that we can give as input to the Neural network.

**Experiment No. 1** – We sorted the images from top to bottom and then passed to python program which we wrote to create directories of specific Alphabets, Numbers, Special characters and store the respective images in them.

**Dependency -**

**Findings** – Able to save all images in respective directories.

Date: 17-02-2020 , 18-02-2020

### Problem – To Extract outer sections from original form

**Experiment No. 1 –**

* In findContours function of CV2 library we used cv2.RETR\_LIST option which extracts all the images and gives us their co-ordinates. Then we used conditional filtering to extract the contours which are above specific width and height.
* Used adaptive threshold and dilate to de-noise the form and cv2.boundingRect to extract outer sections coordinates and saved them in a separate list in sorted order from top to bottom.

**Experiment No. 2** – Used number of different filters by changing size of dilate pixels and adptivethersholding value and boundingrect for each image and saved each filter result in list. Unique set of contours taken and saved only if all contours detected.

**Findings** – We are able to extract all 6 outer sections but in some cases due to folding of paper or scanning issue exact 6 sections not detected.

**Experiment No. 3 –** Created a function to check if sections found satisfying particular condition of width and height compare to original scan image an if not then sections are extracted using reference of 1st section coordinates and gap between subsequent section.

**Findings** – We were able to detect 6 sections perfectly in all the scanned images successfully.

Date: 22-02-2020

### Problem – To Extract inner images in from each section obtained above.

**Experiment No. 1** - Loop of various size and type of filters concatenated like Gaussian, Binary and dilate. Unique set of images taken and saved only if all images coordinates are different.

**Findings-** Able to detect mostly all images and failed to detect only some our one assumption not satisfied i.e. outer boundary is not proper. Some overlapping images also obtained using this method which need to be removed to avoid duplicity.

Date: 26-02-2020

### Problem – Filter of Unwanted Images

**Experiment No. 1** - Used conditional filtering of images. Removed too much small images of size < 30 or greater than 100 pixels so that specific height and width images only will be extracted.

**Findings** - We are able to filter images that are required and removed others.

Date: 26-02-2020

### Problem – To Sort all images in sorted order.

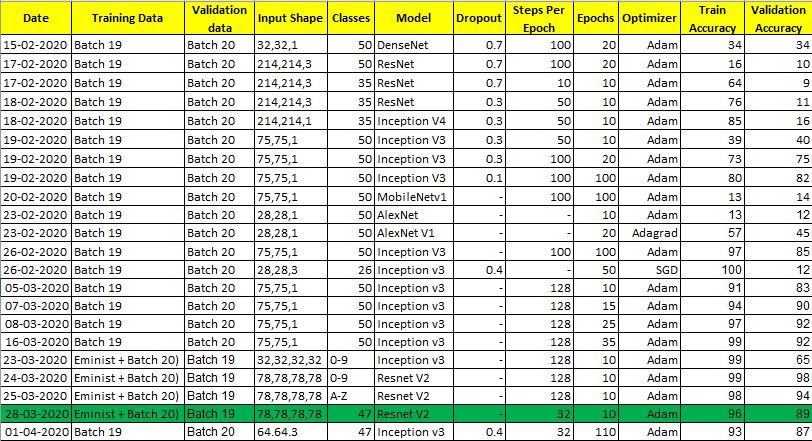
**Experiment No. 1** – Took last image coordinate and stored each row in sorted order varying y pixels’ margins line by line.

**Findings** – Able to sort all images in the required order.

Date: 15-02-2020: 01-04-2020

### Problem – Model Building

**Experiments -** Train and Test data is collected from students as described in Data Collection section.

****

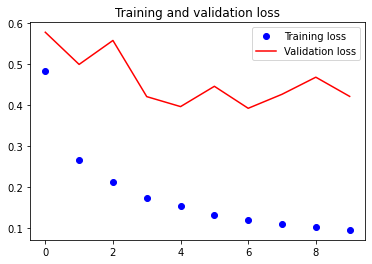
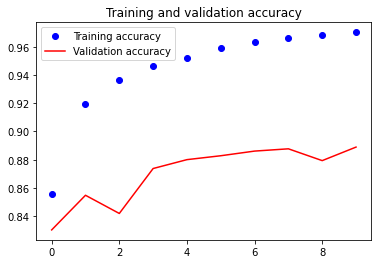
* We started with small sequential models which are not shown above to, due to low accuracy and under fitting issue. So, moved to pre-built models with different input shape and hyperparameter.
* DenseNet and ResNet are deep models and our dataset was small. So, these were overfitting and we moved to other models.
* Using Inception V3 and same data we were able to get accuracy upto 85% and on the model 08-03-2020 we got 92%. During inferencing it was not giving good result as inference was on actual form. So, we realised mismatch in train-validate and actual inference data of boundary as below.

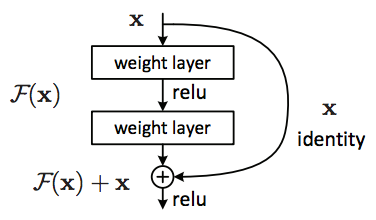
 

(Train Image Sample) (Inference Image Sample)

* So, we used smart crop by taking mode of pixel values of corner rows. If mode is 0 i.e. white then cropping and replacing the row by black pixel otherwise continue.

We added NIST dataset of 128\*128 size images. We took 2000 each class of letters and 1000 each class of digits with crop of 33 from each side to make it similar to our requirement.

* Used Resnet\_v2 with different parameters and able to reach 89% accuracy with below loss and accuracy plots.  
   
* Resnet with even deepening of network can handle the issue of gradient dispersion and to improve network performance using shortcut connections. So, it performed well on our dataset.



* To further enhance the output, we used custom rules which forces model to predict characters as per the form characteristics on row level. e.g. – In mobile number model will can predict only digits and in name column model can predict alphabets only. So, we able to reduce errors in between classes where the model was getting confuse like ‘0’ and ‘O’, ‘1, and ‘I’ etc.

Acknowledgement

**Gourab Nath** **Ankit Sekseria** **Kedarkumar Golla**

Department of Data Science Student Student  
Praxis Business School Praxis Business School Praxis Business School  
Bangalore, India Bangalore, India Bangalore, India  
[gourab@praxis.ac.in](mailto:gourab@praxis.ac.in) [sekseriaankit0657@gmail.com](mailto:sekseriaankit0657@gmail.com) [kedarkumar.golla@praxis.ac.in](mailto:kedarkumar.golla@praxis.ac.in)

**Manish Kumar** **Rahul Aggarwal** **Varishu Pant**

Student Student Student  
Praxis Business School Praxis Business School Praxis Business School  
Bangalore, India Bangalore, India Bangalore, India  
[manishkumar.ms64@gmail.com](mailto:manishkumar.ms64@gmail.com) [aggarwalrahul31@gmail.com](mailto:aggarwalrahul31@gmail.com) [varishupant@gmail.com](mailto:varishupant@gmail.com)

## Appendix

### Github Link to all codes –

<https://github.com/aggarwalrahul31/Handwritten-OCR>

### Code - To filter the required contours

Let h be is image of height

Let raw\_rects = [(x1, y1, w1, h1 ),…..(xn , yn , wn , hn )] is list of all contours

and rects = [ ] is list of all required contour.

for contour in raw\_rects:

if (abs (contour [2] - contour [3]) < 10):

if (contour [2] > h/75 and contour [2] < h/40 and contour [3] > h/75 and contour [3] < h/40 and contour not in rects):

rects.append(contour)

return(rects)

### Code - To sort the contours

Let h be the image of height and n be number of rows.

Let b is list of list of all sorted contours row wise.

rects = [(x1, y1, w1, h1 ),…..(xn , yn , wn , hn )] is list of all contours to be sorted

and rects = [ ] is list of all required contour.

for count in range (1, n+1):

for contour in rects:

x= contour[0]

y= contour[1]

w= contour[2]

h= contour[3]

if ((y > bottom\_y - h\*j - h/2.5) and (i not in temp)):

b[j].append(contour)

temp.append(contour)

b[count]=sorted(b[count], key=lambda x:(-x[0]))

### Libraries Used

Cv2 - https://pypi.org/project/opencv-python/

Numpy - https://pypi.org/project/numpy/

Matplotlib - https://matplotlib.org/

Argparse - https://docs.python.org/3/library/argparse.html

Imutils - https://pypi.org/project/imutils/

Os - https://docs.python.org/3/library/os.html

Glob - https://docs.python.org/3/library/glob.html

Shutil - https://docs.python.org/3/library/shutil.html

PIL - https://pypi.org/project/Pillow/

Pytesseract - https://pypi.org/project/pytesseract/

Urllib - https://docs.python.org/3/library/urllib.html

Re - https://docs.python.org/3/library/re.html

Time - https://docs.python.org/3/library/time.html

Statistics - https://docs.python.org/3/library/statistics.html

scipy.stats - https://docs.scipy.org/doc/scipy/reference/stats.html

operator - https://docs.python.org/3/library/operator.html

pandas - https://pandas.pydata.org/

math - https://docs.python.org/3/library/math.html

csv - https://docs.python.org/3/library/csv.html

sys - https://docs.python.org/3/library/sys.html

gzip - https://docs.python.org/3/library/gzip.html

seaborn - https://seaborn.pydata.org/

pickle - https://docs.python.org/3/library/pickle.html

keras - https://keras.io/

tensorflow - https://www.tensorflow.org/

tqdm - https://pypi.org/project/tqdm/

drive - https://pypi.org/project/google-colab/