OCR using YOLO and Py-Tesseract

Optical Character Recognition (OCR) is used to convert scanned images to text. Building a OCR requires regions of an image detection. These detections are such that the selective text from the original image can be obtained. Thus, after the text is detected, its recognition is required.

Test Detection consists of finding the bounding box and classifying it. YOLO predicts both the boundary box and the class at the same time. It is much faster but there is a tradeoff between speed and accuracy. However, it has good enough accuracy for the application of PAN Card OCR.

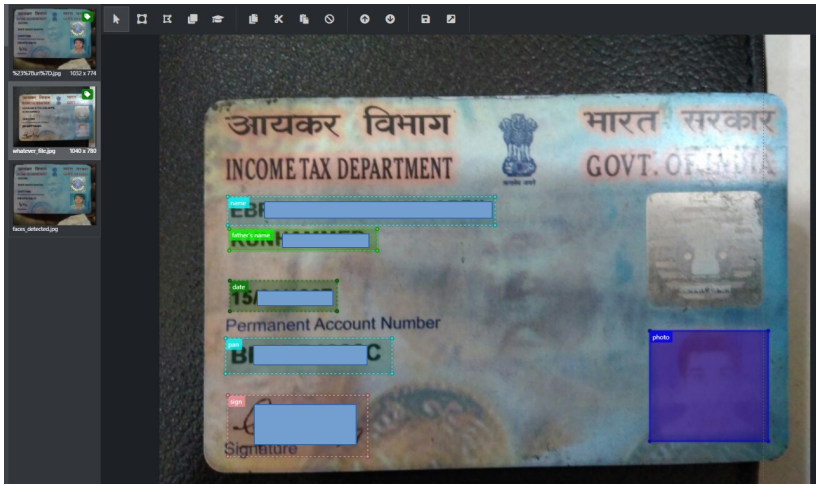
After text detection, Py-tesseract (Python-tesseract) or Tesseract OCR can be used as an open-source text recognizer. The difference between two is that the OCR wrappers for Python-tesseract is based on Googles OCR API while Tesseract OCR isn't.

The first step for building OCR will be data collection and its annotation.

Data Annotation

The data in form of PAN card images were given by the courtesy of HAPPY link.

After collecting the data its labeling was done. VoTT (Visual Object Tagging Tool 1.5) has been used because it is a simple tool and has great support and detailed documentation for the process of data annotation.



After generating the data in Pascal VOC format, it was exported to YOLO after tagging. YOLOv3 uses Darknet-53 as it’s feature extractor. It has overall 53 convolutional layers. It has successive 3 × 3 and 1 × 1 convolutional layers.

Training YOLO using Darknet framework

The Darknet neural network framework was used for training. The framework uses multi-scale training, data augmentation and batch normalization. It is an open source neural network framework written in C and CUDA.

The google notebook file for darknet training is located in

/OCR_for_PanCards/DarknetModel/YOLOv3.ipynb

```bash
!git clone https://github.com/pjreddie/darknet.git
```

Appropriate modification in config file is done. The required config file is available in ‘cfg’ folder named ‘yolov3.cfg’. The batch size, subdivision, number of classes, and filter parameters were changed according to the number of classes.
The training was started with pre-trained weights of darknet-53. This helped the model to converge early.

```bash
./darknet detector train data/obj.data yolo-obj.cfg darknet53.conv.74
```

The training done in Darkent framework was stopped using the mean average precision (mAP) as stopping criteria. The weights file with the highest mAP score was chosen.

Training Command:

```bash
$ darknet.exe detector train data/obj.data yolo-obj.cfg "last_executed" -map
```

Calculating MAP and choosing best score output weight file. Below are some screen shots-

mAP- 74.12% Iteration Count- 2722

```
 (next mAP calculation at 2742 iterations)
 Last accuracy mAP@0.5 = 74.12 %, best = 74.12 %
 2722: 1.703769, 1.627247 avg loss, 0.001000 rate, 5.164064 seconds, 174208 images, 20.014403 hours left
Loaded: 0.000074 seconds
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.000000, GIOU: 0.000000), Class: 0.000000, Obj: 0.000000, No (
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.807712, GIOU: 0.801193), Class: 0.858393, Obj: 0.718702, No (
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.713008, GIOU: 0.691722), Class: 0.682916, Obj: 0.806141, No
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.608551, GIOU: 0.575755), Class: 0.960694, Obj: 0.054708, No (
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.759461, GIOU: 0.739582), Class: 0.837306, Obj: 0.562739, No (
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.628862, GIOU: 0.594195), Class: 0.598805, Obj: 0.336547, No
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.789868, GIOU: 0.783171), Class: 0.500715, Obj: 0.966732, No (
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.814928, GIOU: 0.811696), Class: 0.861250, Obj: 0.680489, No (
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.689477, GIOU: 0.660968), Class: 0.552798, Obj: 0.502943, No
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.786706, GIOU: 0.781545), Class: 0.994883, Obj: 0.922124, No (
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.794348, GIOU: 0.787166), Class: 0.745998, Obj: 0.800728, No (
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.763158, GIOU: 0.759571), Class: 0.955924, Obj: 0.395865, No
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.000000, GIOU: 0.000000), Class: 0.000000, Obj: 0.000000, No (
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.741534, GIOU: 0.734382), Class: 0.555824, Obj: 0.586926, No (
```

mAP- 78.34% Iteration Count- 3134

```
 (next mAP calculation at 3224 iterations)
 Last accuracy mAP@0.5 = 78.34 %, best = 78.34 %
 3134: 1.578159, 1.578921 avg loss, 0.001000 rate, 4.627341 seconds, 200576 images, 20.921223 hours le
Loaded: 0.000085 seconds
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.000000, GIOU: 0.000000), Class:
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.772992, GIOU: 0.762907), Class:
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.713569, GIOU: 0.708765), Class
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.000000, GIOU: 0.000000), Class:
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.791599, GIOU: 0.787758), Class:
```

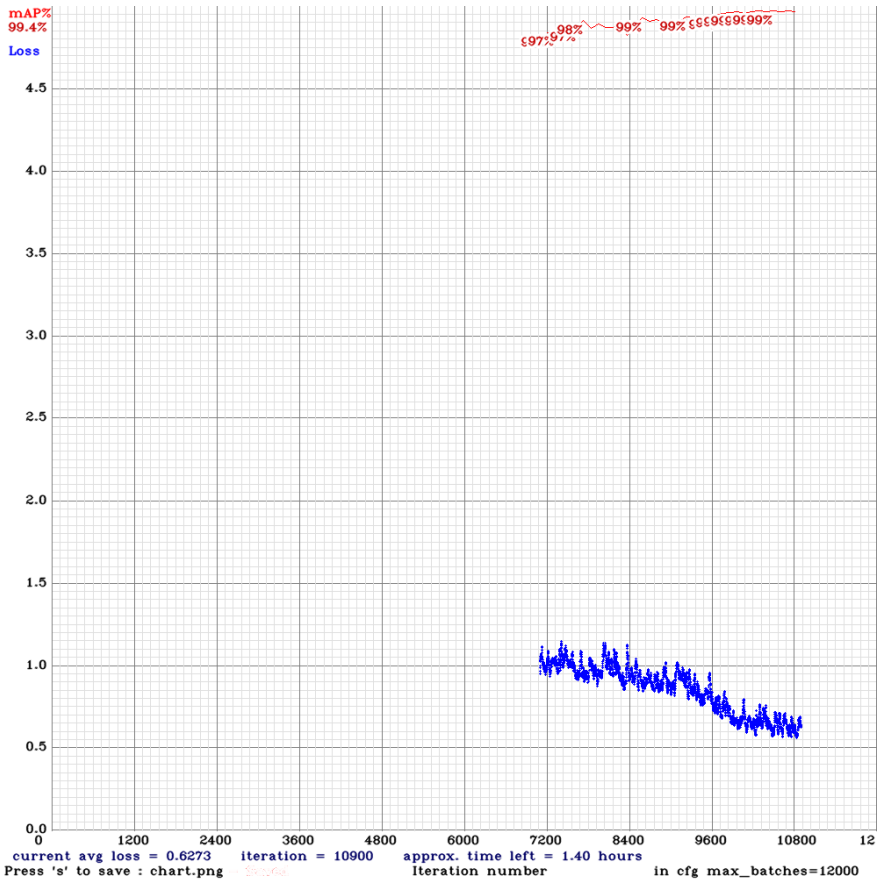
mAP- 80.97% Iteration Count- 3513

```
 (next mAP calculation at 3524 iterations)
 Last accuracy mAP@0.5 = 80.97 %, best = 80.97 %
 3513: 1.650075, 1.649649 avg loss, 0.001000 rate, 4.678863 seconds, 224832 images, 18.299336 hour
Loaded: 0.000083 seconds
```

mAP-96.78% Iteration Count- 7210

```
 (next mAP calculation at 7312 iterations)
 Last accuracy mAP@0.5 = 96.78 %, best = 96.78 %
 7210: 1.070790, 1.020686 avg loss, 0.001000 rate, 3.587335 seconds, 461440 images, 13.004496 hour
Resizing, random_coef = 1.40

 576 x 576
 try to allocate additional workspace_size = 95.55 MB
 CUDA allocate done!
Loaded: 0.000060 seconds
```

Loss vs Iterations and mAP vs Iterations:



The weight file is provided as [yolov3_custom_best_10000.weights]. The best file from 10,000 interations was chosen based on mAP score.

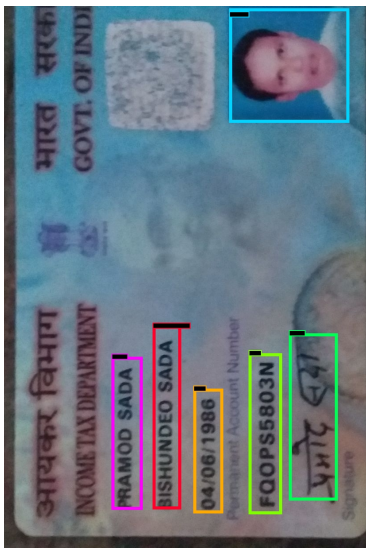
Output after training

Following are the three different types of orientation.

1. Horizontal or 0° Skew

2. -90° Skew

3. +90° Skew



Creating a pipeline

1. Detect the required region from the image and crop the region to save cropped images.
2. Pass that detected regions to Py-Tesseract OCR with anglular skew and blured improved filters for recognition.
3. Store the results in required csv format.

Pytesseract OCR on the fact is maintained better. Both the OCRs were tried and Pytesseract OCR worked better this project.
The pipelined system source code can be located at

/OCR_for_PanCards/PAN_OCR

Both OCR wrapers i.e. Tesseract OCR and Py-tesseract are used. The python code files for both can be located at

/OCR_for_PanCards/PAN_OCR/MainWithPytesseract.py & at

/OCR_for_PanCards/PAN_OCR/MainWithTesseract.py respectively.

Source Code

Python3 and py-tesseract for python3 are used for this project. To run the above main.py files, follow the instructions:

Clone my github repo, then download the weight file from this google drive link and place this file in below directory.

/OCR_for_PanCards/PAN_OCR/

Now, run the following commands in same directory.

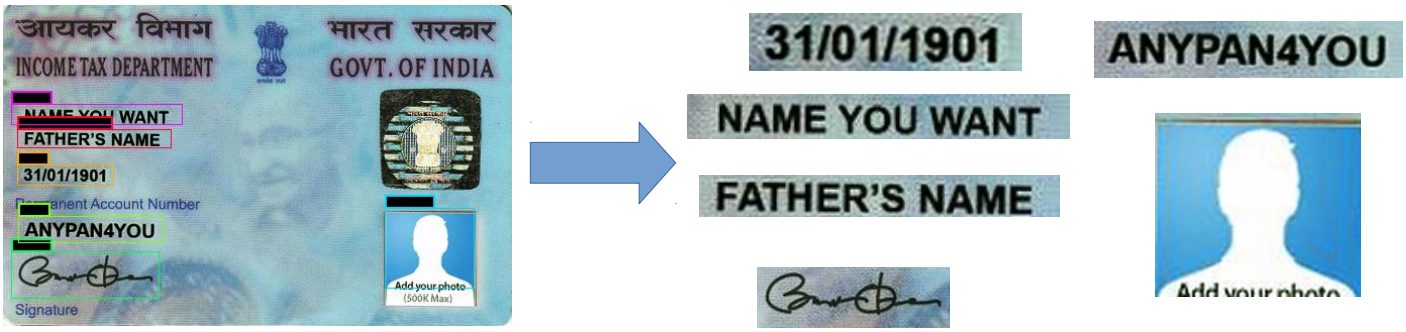
```bash
$ bash ./darknet.sh

$ python3 MainWithPytesseract.py -d -t --weights
yolov3_custom_best_10000.weights
```

Keep your images at */OCR_for_PanCards/PAN_OCR/pancards* directory and see the csv file at /OCR_for_PanCards/PAN_OCR/output directory.

Results

1. Horizontal or 0° Skew

Cropped Image Extraction:



OCR Conversion:

Converted to two sets of OCR extractions because the cropped image for some test cases was mirror rotated. Two API hits to NSLD for validation of PAN card shall solve the purpose.

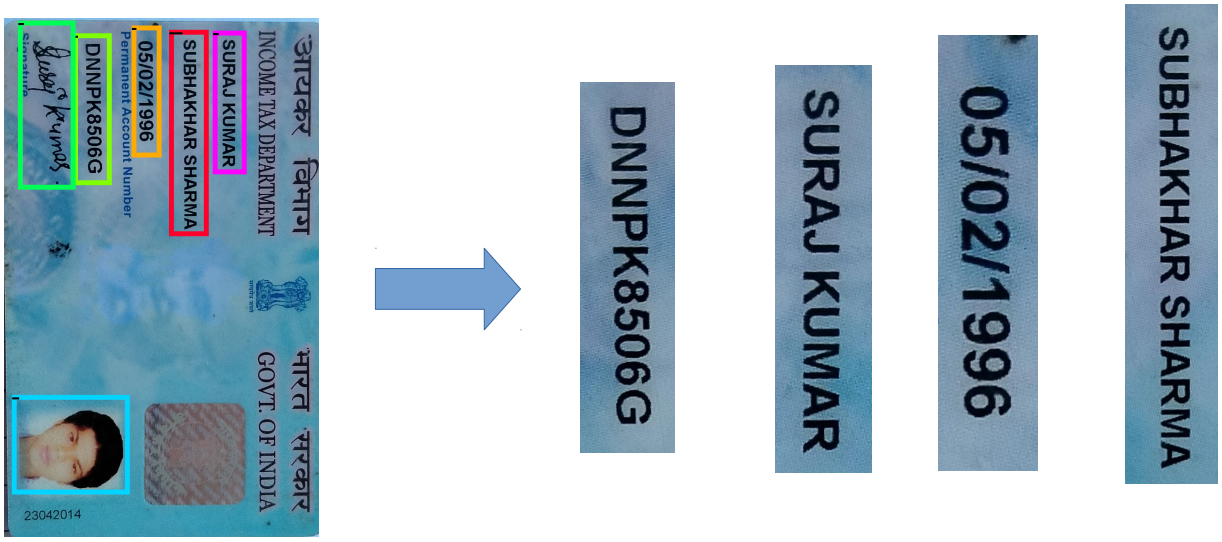
CSV Output:

name	fathername	dob	pan	name2	fathername2	dob2	pan2
NAME YOU WANT	FATHER'S NAME	31/01/1901	ANYPAN4YOU	4ANVM NOA SWYN	SWYN S.YSHLV4	LOGL/LO/LE	NOAPNVdANY.

As shown above, first the image of pan card was passed into YOLO. Then, YOLO detected the required text regions and crops them out from the image.

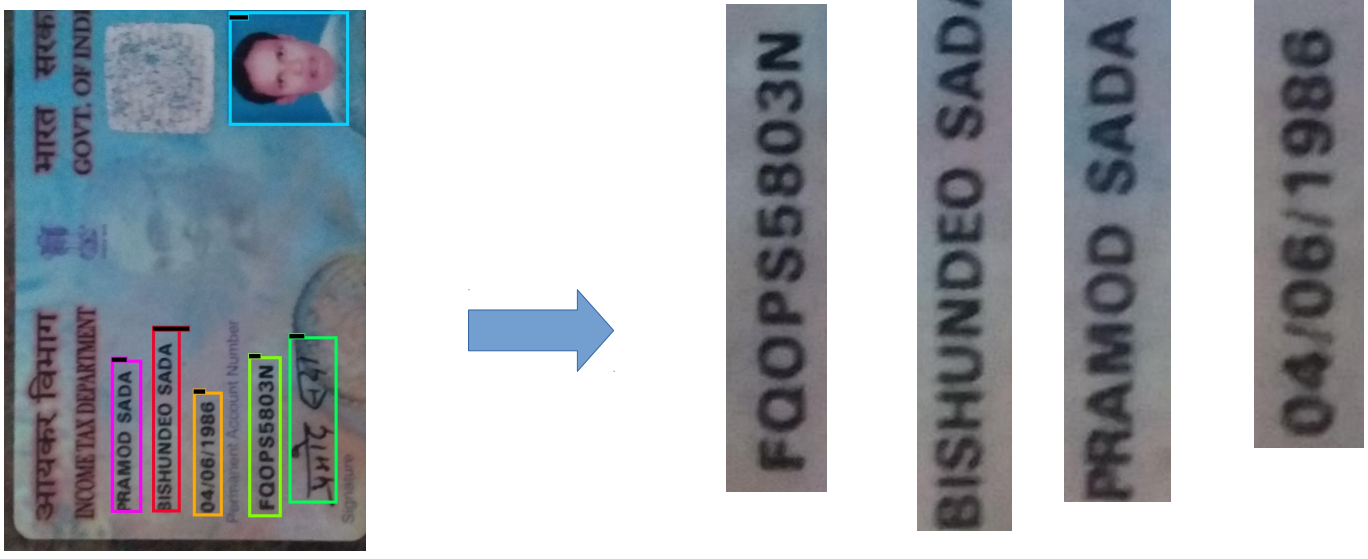
The cropped regions were passed one by one to pytesseract. Pytesseract read them, and the information was stored in csv format.

2. -90° Skew



name	fathername	dob	pan	name2	fathername2	dob2	pan2
SURAJ KUMAR	~ 05/02/1996	SUBHAKHAR SHARMA	DNNPK8506G	YVAN PVANS	9661/Z0/S0. |	VWYVHS M|90S8MdNNG	

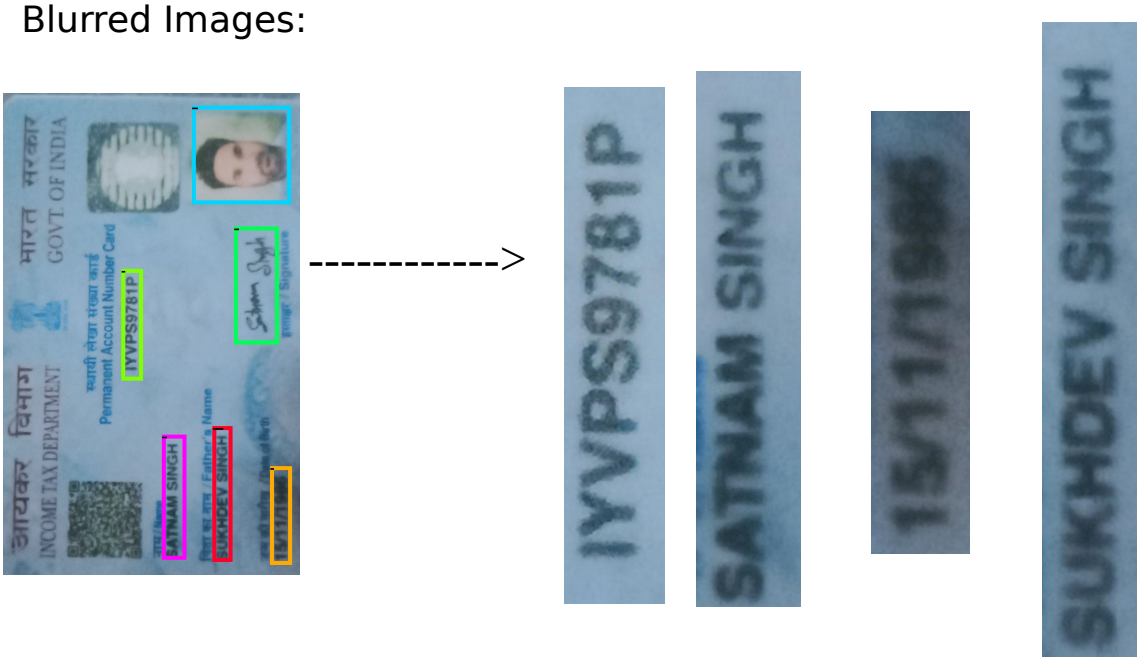
3. +90° Skew



name	fathername	dob	pan	name2	fathername2	dob2	pan2
VGOVS GOWViid	VaVS OJONNHSIS	986 1/90/%0	NeosssdO0d	PRAMOD SADA	BISHUNDEO SADA	04/06/1986	FQOPS5803N

Outlier

Blurred Images:



pan	name	fathername	dob	pan2	name2	fathername2	dob2
di 8L6SdAAl			SLA	i¥YVPS9781P			ivl1/iee6 |

Solution: Exception handling subroutine for Re-upload or Mannual Detection.