# Machine Learning POC: Interpretation of Organisation Charts

[**Machine Learning POC: Interpretation of Organisation Charts**](#_neft2k6ho1d)

[Background and Objective](#_5rghjdlkrxfu) 2

[Overall Objective](#_fuzgvkfunmzp) 2

[POC Objective and Deliverable](#_j80l11wsovp7) 2

[Solution Workflow](#_meo2nu78m7pj) 3

[Code Components:](#_n9kcv0717urr) 3

[Module 1: Signature Verification](#_9nbv7y2yl56y) 3

[Module 2: Information extraction](#_v80aij1v9ev2) 4

[2.1 Detect boxes](#_eahvimj4ixy2) 4

[2.2 OCR using Google Vision API](#_6okvdy7y86el) 4

[2.3 Find paths and relationships between entities (boxes)](#_szko6y57refw) 4

[Machine Learning Plug-in](#_94aiiibfs5bb) 8

## 

## Background and Objective

* CDD review involves correlation and verification of 20+ customer documents;
* The CDD team is targeting ~850k cases per year;
* To mitigate human error, each case undergoes a Quality Control process, and a sample of 10% also go through Quality Assessment. Currently, one of these reviews is taking 2-4 hours.

This is an expensive process for the bank.

### Overall Objective

HSBC is investigating automation options to make the process faster, cheaper & more accurate – improving our customer and staff experience.

However, vendors have no current solution to interpret organisation charts. If this challenge is not resolved, when org charts are provided approximately 10% of the QC checking for that customer profile will need to remain manual.

### POC Objective and Deliverable

Demonstrate a software solution which can extract information (entities & semantics) from an org chart, complementing a Machine Learning solution and allowing the potential for 100% coverage of inputs.

To run the code, open Command Line Interface (CLI) and type:

***python F\_OSR.py***

The program will ask you to input

1. Image name (relative to the path where code is running)
2. Client name

**Note:** Input image must be of type .jpg or .png



## Tool Stack

## Python, OpenCV 3, Google Vision API, Dlib

## Solution Workflow

## Code Components and Modules

## Module 1: Signature Verification

When the Org Structure Diagram is passed to the software, it first goes through the signature verification module. The repository of valid signature’s is stored in folder named “signs”. The signature verification module finds the signature and compares it with the repository containing the valid signatures. We use SIFT features in opencv along with FLANN algorithm to do feature matching of signature in the document with the ones in the repo. The algorithm computes a match percentage between each pair of signatures by extracting out the salient features of the signature present on the org chart.

If there is a match above a certain threshold we can conclude that the org chart is valid and send it to the next component to do the information extraction. But in the case of no match, we stop the program and show an error message - “No valid signature is found in the document”.

### Module 2: Information extraction

This broad component has the following sub-modules:

### 2.1 Detect boxes

Using OpenCV library support for Python, we are finding [contours](http://docs.opencv.org/trunk/dd/d49/tutorial_py_contour_features.html) to detect rectangles, squares or any geometrical regular 4-sided shape. (Shapes with rounded edges have not be considered)

### 2.2 OCR using Google Vision API

The bounding vertices of each rectangle found from the above module is sent to the [Google Vision API](https://cloud.google.com/vision/). If OCR gives an empty result, the empty box is discarded.

The percentages (if any) present in the org chart are also found by making Google Vision API calls. The entity nearest to the location of the percentage found, is linked to that percentage .

**Note:** We also tested OCR capability using [Google Tesseract](https://github.com/tesseract-ocr/tesseract), but test results revealed better accuracy with the Google Vision API.

### 2.3 Find paths and relationships between entities (boxes)

A simple but innovative approach to finding paths has been implemented. It emulates the ‘Lead-the-mouse-to-cheese’ problem approach.

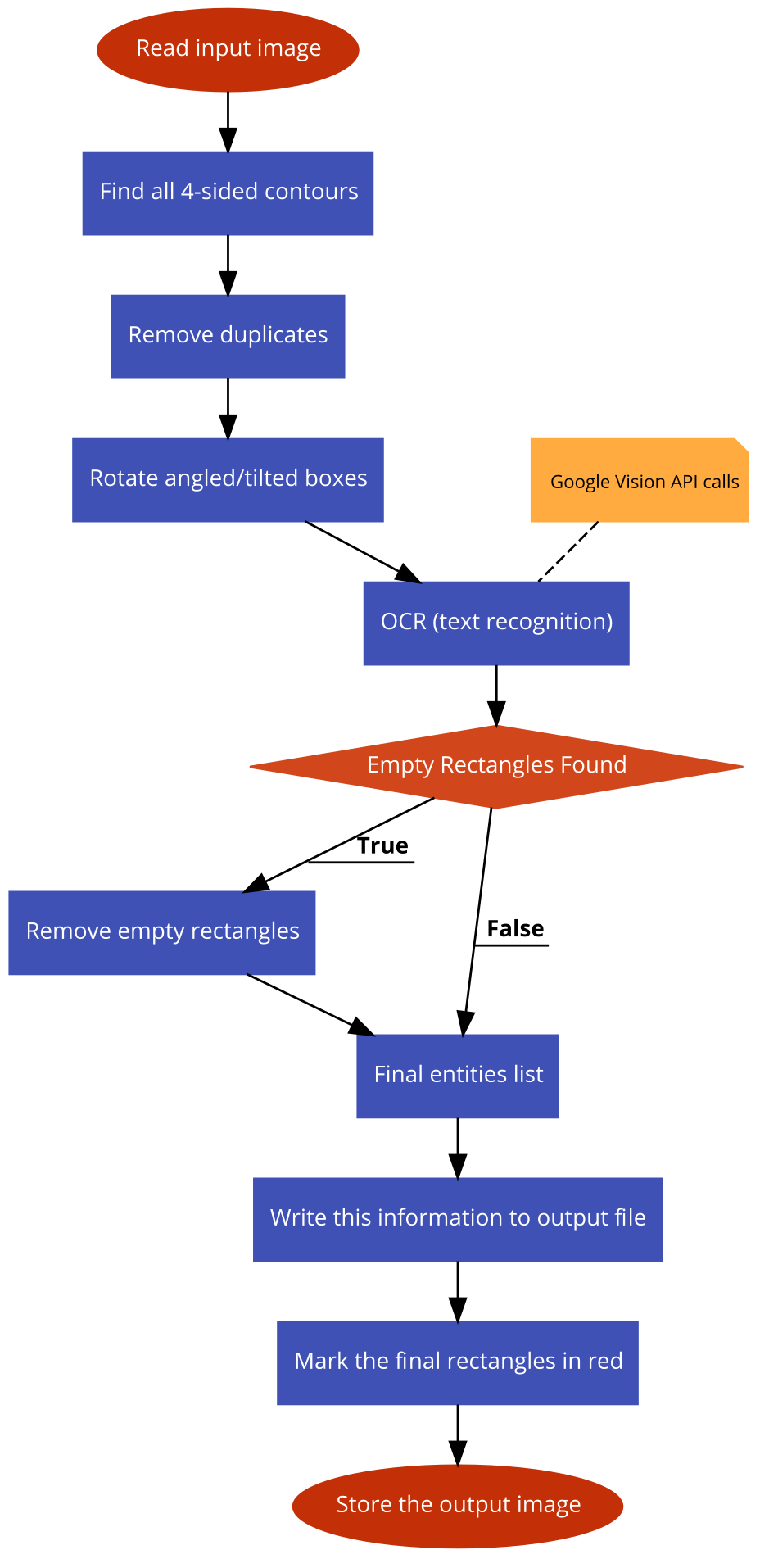
Step 1: Convert input image to its negative

Step 2: Form all nC2 pairs of entities.

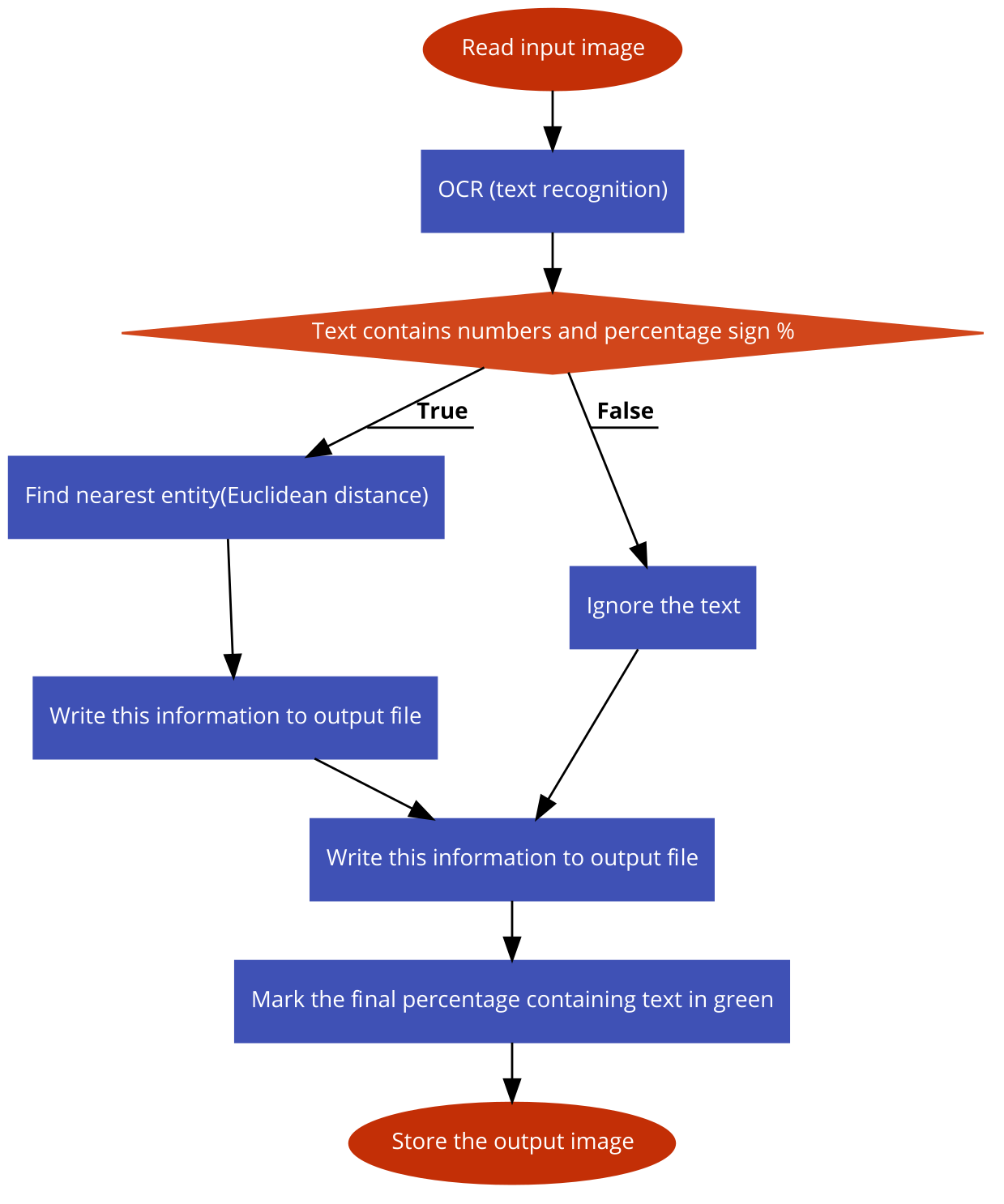
Step 3: For each pair of entity at a time, we see if there exists a connection between the 2 entities.

Finding a connection/path constitutes of a series of operations:

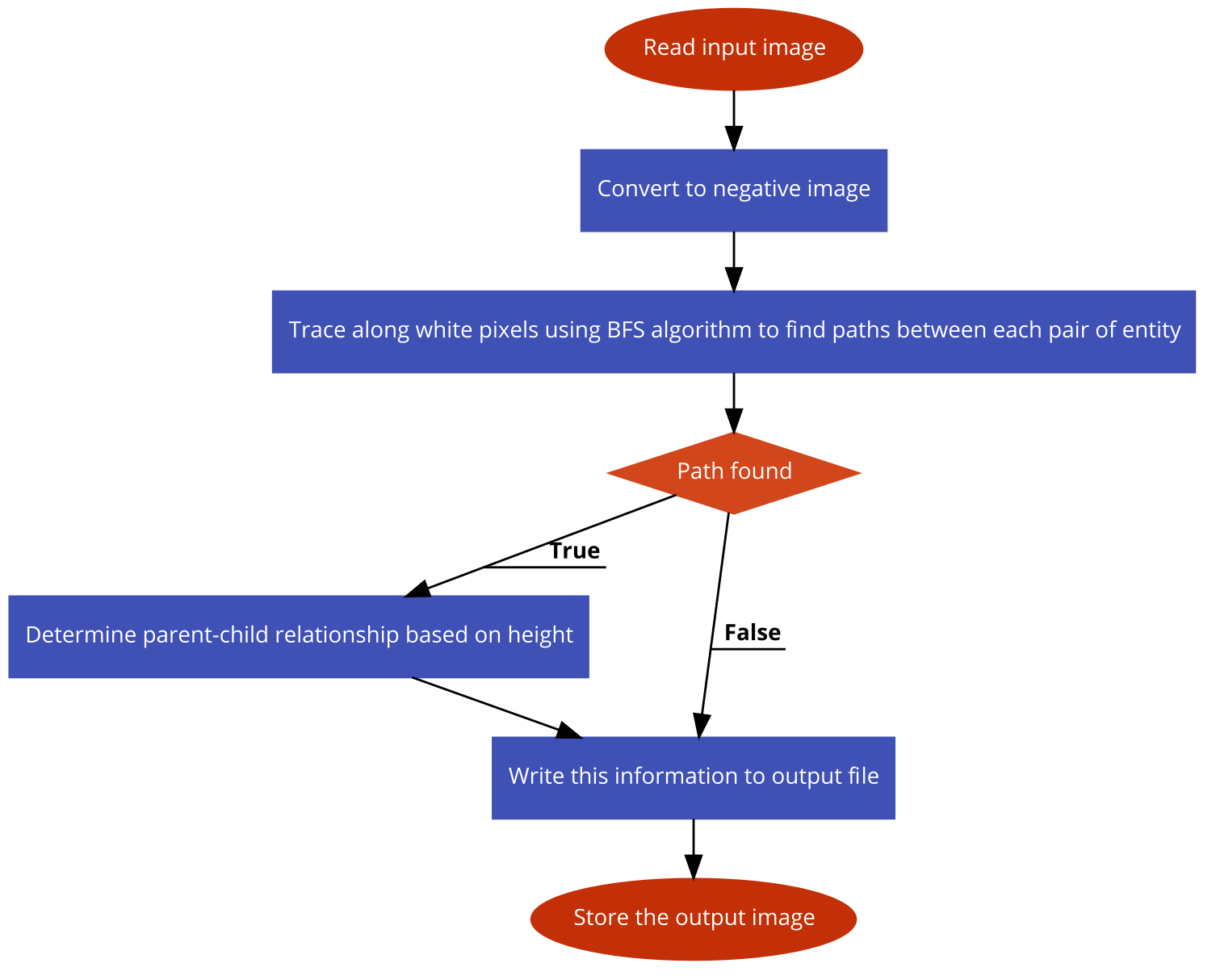
* In a (temporary) copy of the input image, make all entities/rectangles completely black except the two being considered as candidates for path finding.
* Starting from the centre of one entity, trace through the pixels in all 4 directions trying to find a pixel which is not black.
* Continue this in a BFS (Breadth First Search) fashion.
* If you are successfully able to reach from centre of entity to the centre of the other entity, then you can conclude that a path exists.



Module 2.1



Module 2.2



Module 2.3

## Machine Learning Plug-in

The use of the Machine Learning component is to demonstrate it capabilities. For now we are using it to identify the boxes (i.e, entities and subsidiaries) in the Organization chart. Various machine learning libraries like TensorFlow, dlib, Scikit-Learn, Theono etc can be used to achieve the above problem. Tensorflow, Theono gives us granular controls but on the expense of time to implement. So the best solution was to use dlib.

To demonstrate use of ML in box detection we first train the machine with limited data and see the accuracy, and then we keep on increasing the training data to show how increasing the training data benefits the prediction accuracy of the model.

***python F\_ML.py <argument1> <argument2>***

where argument1:

Here you provide option whether you want to train the model or predict using the model

1. train: Using this option will removing any overwrite on any pre existing model and make a new model in its place.
2. trainAppend: This option preserves the learning done before by writing on top of the earlier model and appending instead of overwriting.
3. predict: This will do the prediction on all the images provided as the second argument in the script.

argument2:

Here you have to specify the path of the image folder which you want to use for training or prediction.