# Table Extraction via Eye Gaze Tracking


*Columbia Team:*

*Yibai Liu*

*Yijia Jin*

*Yeqi Zhang*

*Shihang Wang*

*Yinqiu Feng*

*Mentors from JP Morgan & Chase:*

*Nonie Thomas, Saheed Obitayo, Daniel Borrajo, Naftali Cohen*

# *Agenda*

- ❑ Project Motivation and Objective
- ❑ Methodology
- ❑ Datasets
- ❑ Data Exploration and Insights
- ❑ Implementation
  - ❑ Baseline Solution
  - ❑ Advanced Solution
- ❑ Conclusion
- ❑ Future Work

# Project Motivation and Objective

# *Project Motivation*

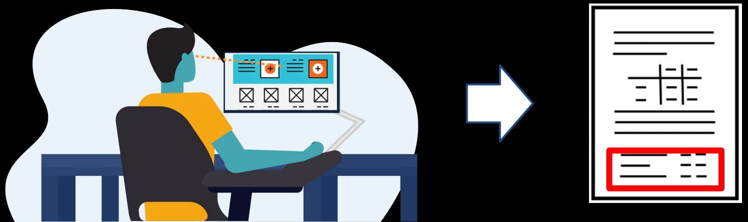
## *Why is table extraction important?*

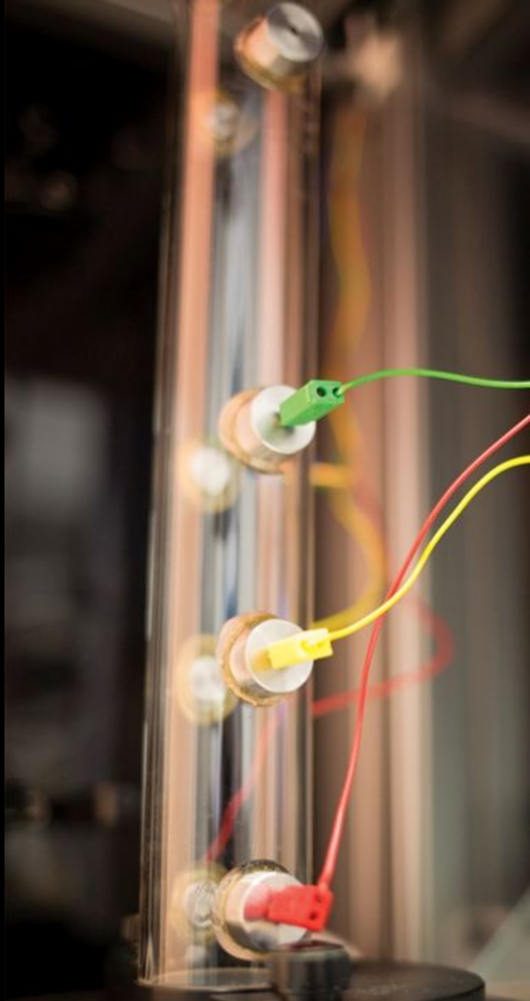
- Drastically increasing use of tabular data
- Needs for efficient workflow and smooth knowledge sharing throughout the business

Consider a use case scenario?

## *Why bring in eye gaze tracking?*

- For documents with multiple tables, current table extraction solutions cannot distinguish which table is of user’s interest
- Eye tracker knows where your attention is on





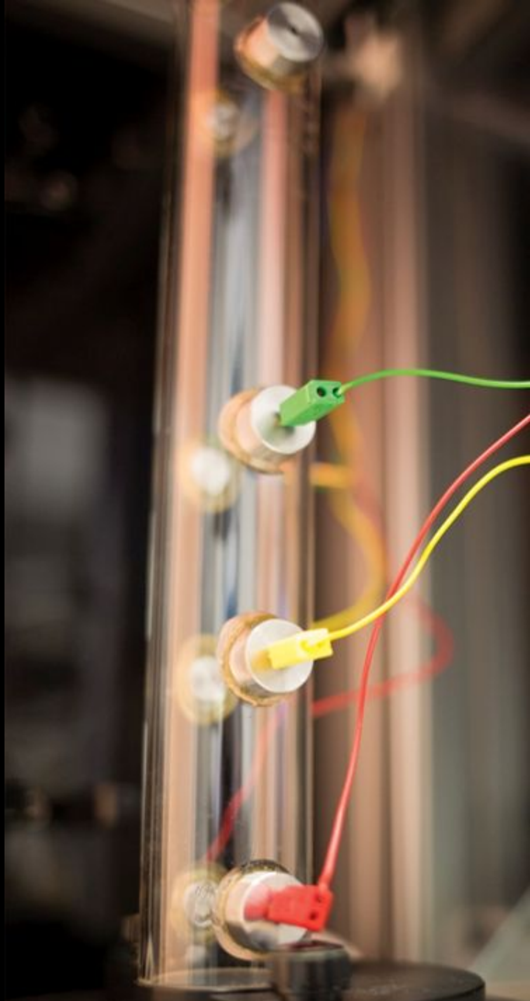
JPMorgan Chase & Co.

Columbia Engineering
The Fu Foundation School of Engineering and Applied Science

# *Objective*

- Leveraging eye gaze technology and Computer Vision (CV) to design a system that automatically detects tables from an image

- Extracting texts from tables with the Optical Character Recognition (OCR) engine Tesseract

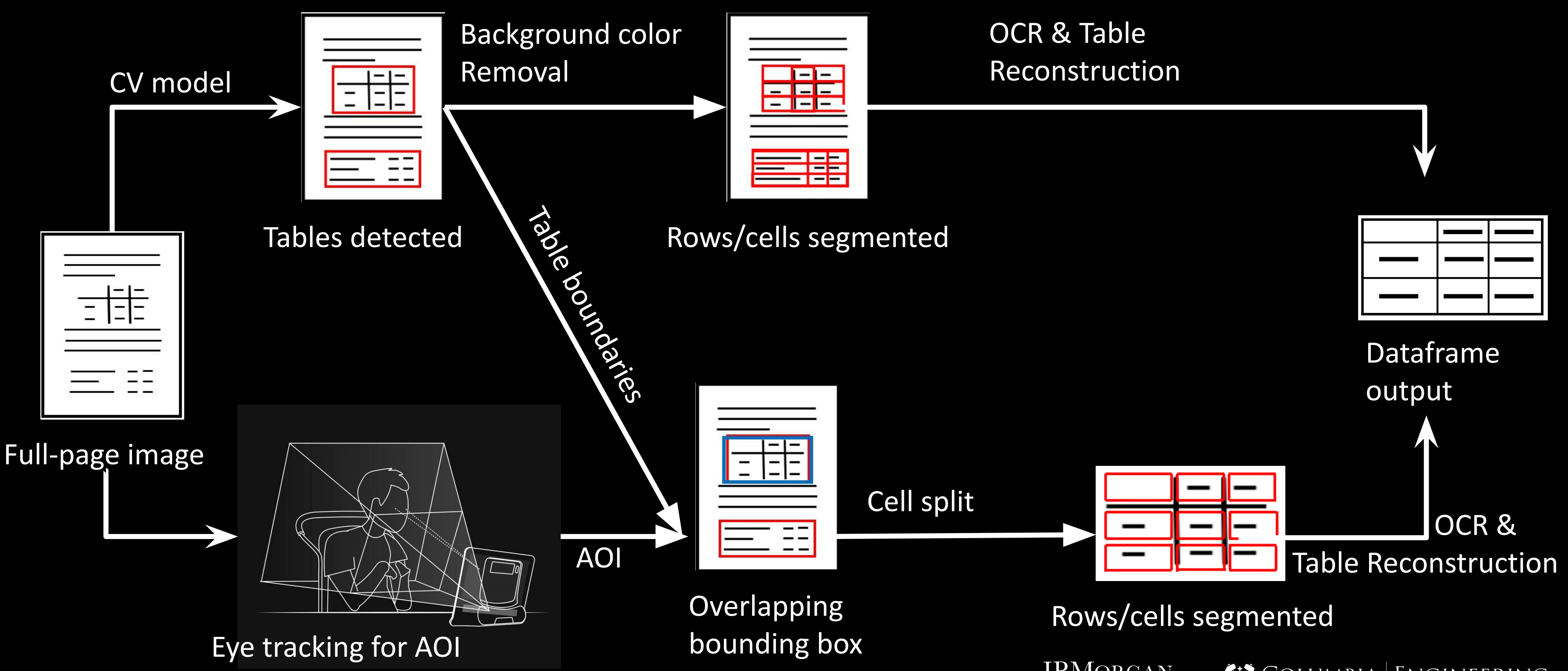


JPMorgan Chase & Co.

Columbia | Engineering
The Fu Foundation School of Engineering and Applied Science

# Methodology

# *Flow Chart of pygazeTE*




JPMorgan Chase & Co.

Columbia | Engineering
The Fu Foundation School of Engineering and Applied Science

# Datasets

# *Eye Gaze Data*


- Designed a **webcam-based eye tracking experiment** in Python using Gazepoint's eye tracker GP3 and the open-source toolbox `PyGaze`
- Instructions would lead the participant to calibrate the device, then a number of images would be displayed on the screen for 10 seconds each.
- Eye movements are recorded and used to identify eye movements and areas of interest (AOIs).

# *Public Data: FinTabNet*

- **Source**: IBM's open source dataset
- **Format**: PDF (documents), JSON (annotations)
- **Size**: 16 GB
- **Data coverage**: about 90k pages of earnings reports from S&P 500 companies with cell annotations
- **Description**: We used the **2,000** documents images using `pdf2img` for modeling.
- **Challenges**: borderless, tight rows

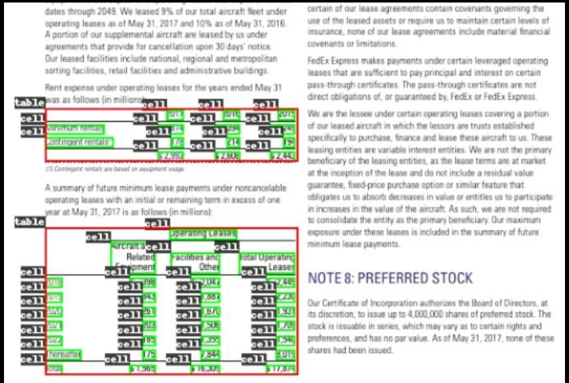
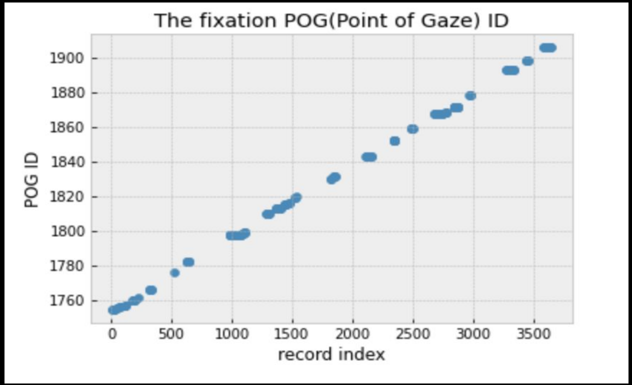



*Table structure annotations*

Source: https://developer.ibm.com/exchanges/data/all/fintabnet/

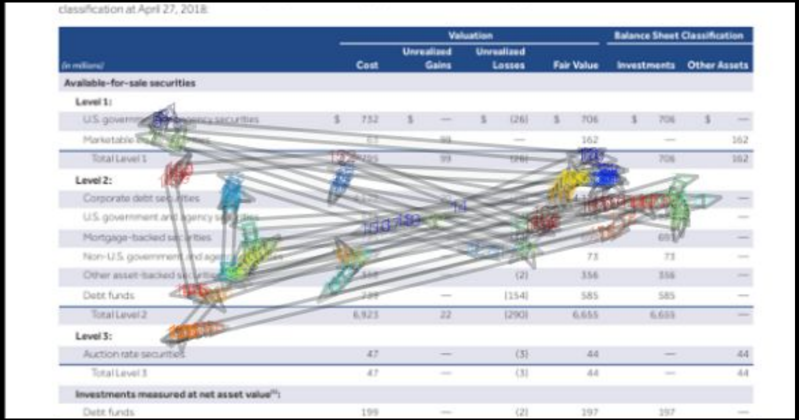
# Data Exploration and Insights

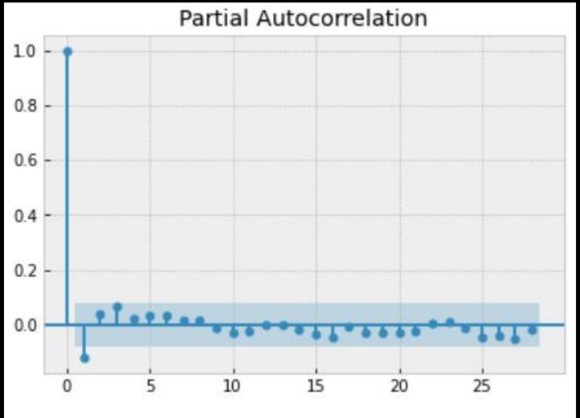
# *Data Exploration and Insights*




- **Critical impact of calibration accuracy**
- **Consistency of missing value:** Some data were missed during regular blinking. *The frequency of blinking = total time / counts of blinking*. On average, we found blinks happen every 2.5 seconds and each lasted about 0.3 seconds.

# Data Exploration and Insights

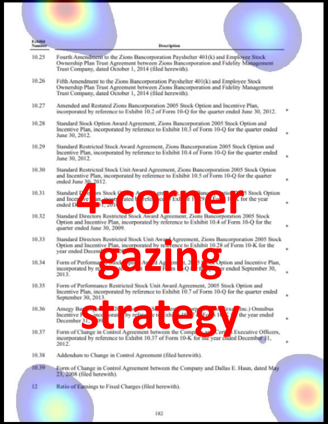


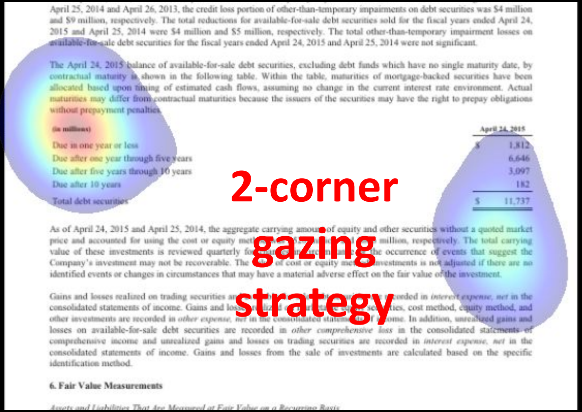




- **Difference of reading patterns:** reading habits differ from person to person.
- **Time Series analysis:** eye movements can be potentially predicted with ARIMA(0,1,1) model

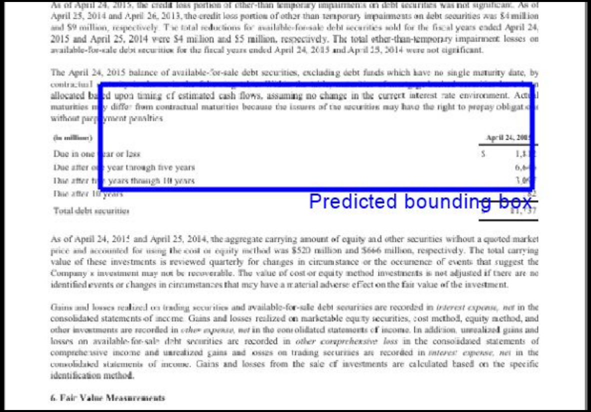
# Implementation

# *Baseline solution: Eye gaze only*




*Fixation density maps*



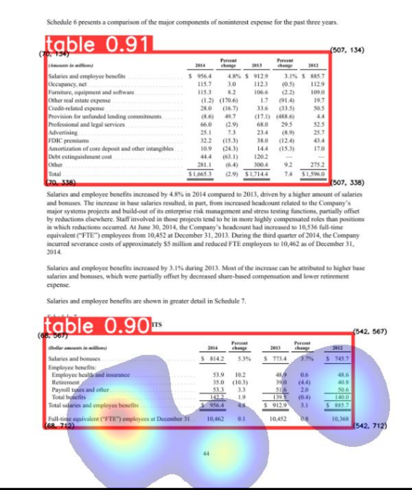

*Predicted bounding box*

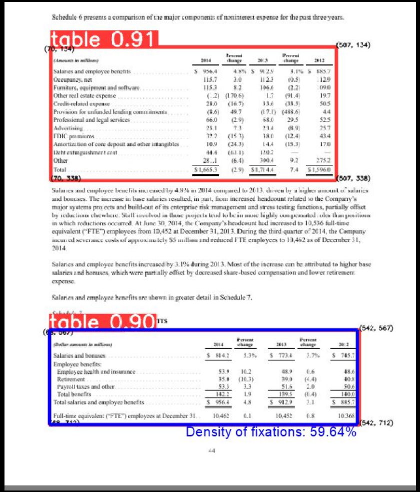
**Idea:** Crop a bounding box by capturing corners of a table via DBSCAN clustering

**Gaze strategy attempts**:

1. *Four-corner*: all corners of the table, 2 seconds each
2. *Two-corner*: only the *upper left* and the *bottom right corners, 5 seconds each*

# *pygazeTE: Eye gaze + CV + OCR*



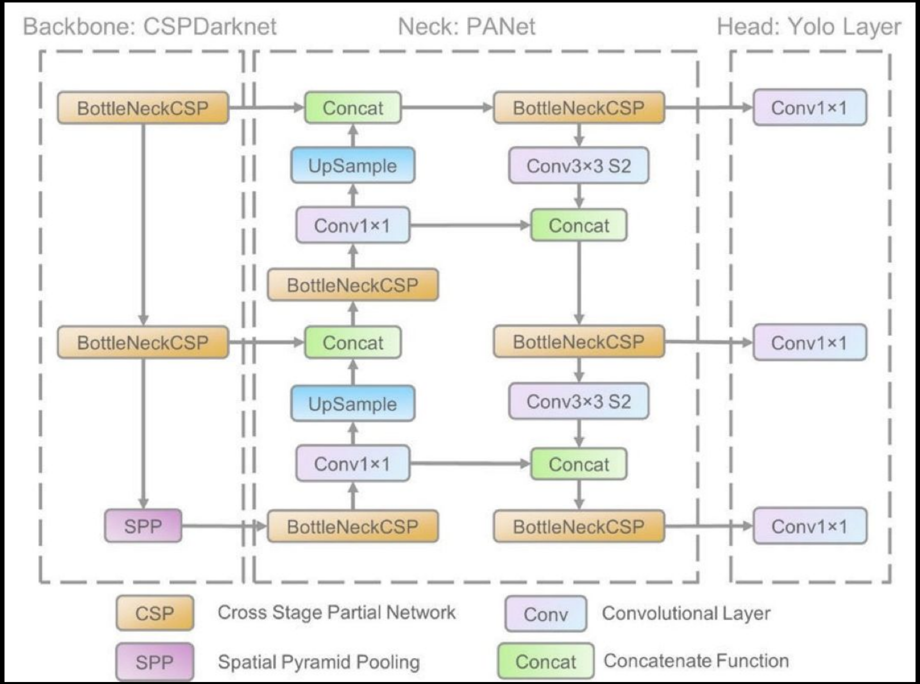




**Idea:** Use more precise CV and OCR models to first identify table boundaries and texts, and then use eye fixations to determine the table of interest.

- ***YOLO-v5***: table boundaries prediction
- ***Tesseract***: text recognition
- **Pygaze**: AOI prediction

**Gaze strategy:** *Center-focused*

# Table Detection by Transfer Learning (CV Model)




- **Backbone**: CSP Darknet
- **Neck**: PANet
- **Head**: Yolo layer

- Data are first input to CSP Darknet for feature extraction.
- Then fed to PANet for feature fusion.
- Finally, Yolo Layer outputs detection results (class, score, location, size).

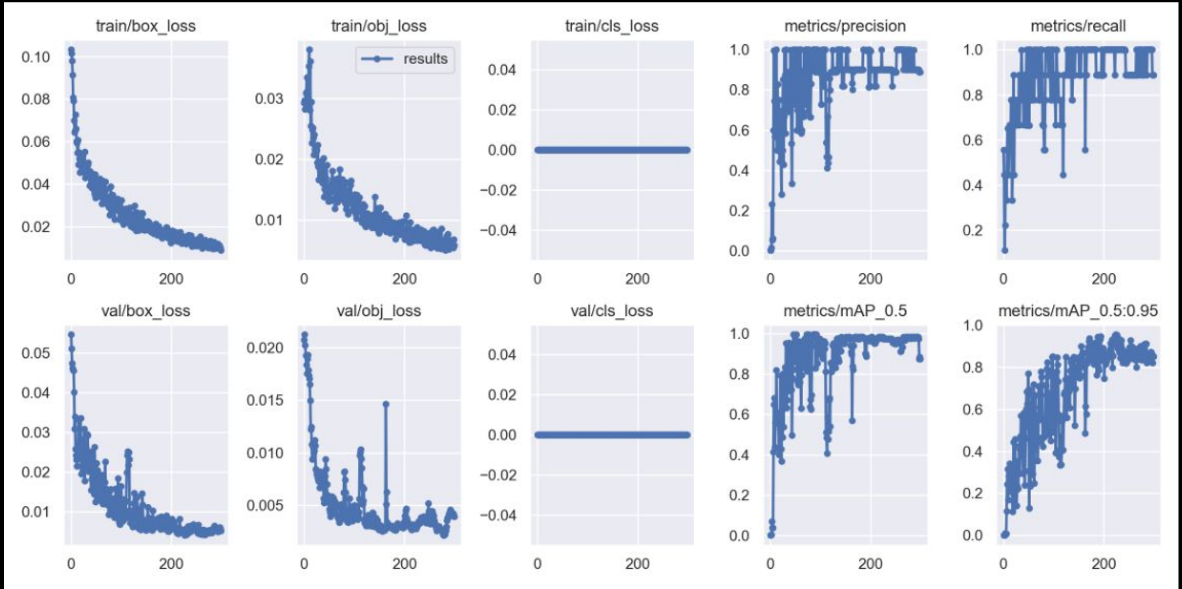
Source:
*https://www.researchgate.net/figure/The-network-architecture-of-Yolov5-It-consists-of-three-parts-1-Backbone-CSPDarknet_fig1_349299852*

# CV Model

## 1. Train

Trained using ***Yolov5x.pt*** pretrained weights for 300 epochs, on 80 annotated images and tested on 20 images with the mAP of 0.824.

- As the object category in this experiment is only table, the classification loss is 0.



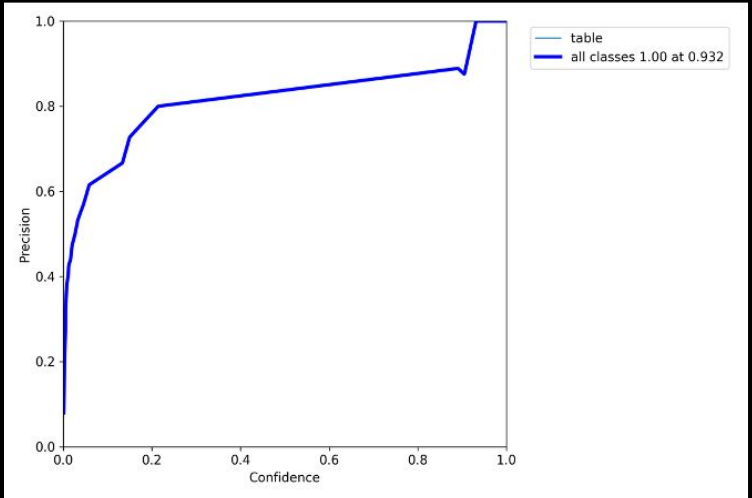
# *CV Model*

## 2. Evaluate

Total Loss = Classification Loss + Localization Loss + Confidence Loss

- Classification Loss : 0.0
- Localization Loss : 0.005
- Confidence loss : 0.004

The P-Curve shows that the model has a high confidence on detecting a table with high precision in the test set.

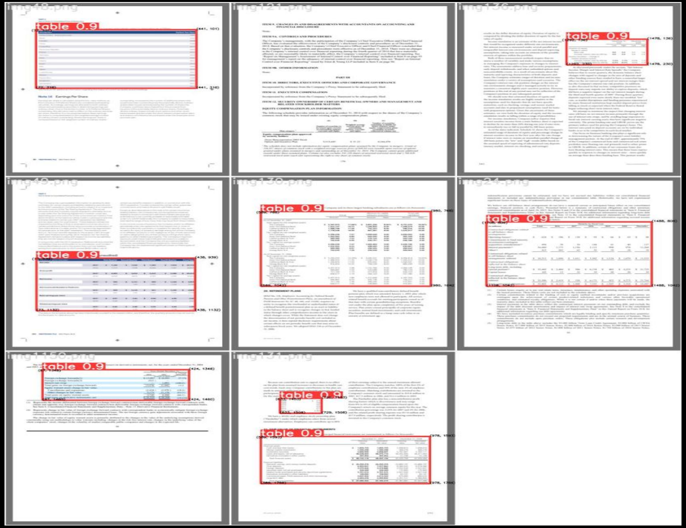


# CV Model

## 3. Output

Given document images as input, the model will :

- detect all tables with confidence scores
- crop each table with its coordinates



| (in millions, except per share data) | Fiscal Year 2016 | 2015 | 2014 |
|---|---|---|---|
| **Net sales** | $ 28,833 | $ 20,261 | $ 17,005 |
| **Costs and expenses:** | | | |
| Cost of products sold | 9,142 | 6,309 | 4,333 |
| Research and development expense | 2,224 | 1,640 | 1,477 |
| Selling, general, and administrative expense | 9,469 | 6,904 | 5,847 |
| Special charges (gains), net | 70 | (38) | 40 |
| Restructuring charges, net | 290 | 237 | 78 |
| Certain litigation charges, net | 26 | 42 | 770 |
| Acquisition-related items | 283 | 550 | 117 |
| Amortization of intangible assets | 1,931 | 733 | 349 |
| Other expense, net | 107 | 118 | 181 |
| **Operating profit** | 5,291 | 3,766 | 3,813 |
| Interest income | (431) | (386) | (271) |
| Interest expense | 1,386 | 666 | 379 |
| **Interest expense, net** | 955 | 280 | 108 |
| **Income from operations before income taxes** | 4,336 | 3,486 | 3,705 |
| **Provision for income taxes** | 798 | 811 | 640 |
| **Net income** | $ 3,538 | $ 2,675 | $ 3,065 |
| **Basic earnings per share** | $ 2.51 | $ 2.44 | $ 3.06 |
| **Diluted earnings per share** | $ 2.48 | $ 2.41 | $ 3.02 |
| **Basic weighted average shares outstanding** | 1,409.6 | 1,095.5 | 1,002.1 |
| **Diluted weighted average shares outstanding** | 1,425.9 | 1,109.0 | 1,013.6 |
| Cash dividends declared per ordinary share | $ 1.52 | $ 1.22 | $ 1.12 |

```
0 0.507949 0.336025 0.799682 0.458385
```

JPMorgan Chase & Co.

Columbia | Engineering
The Fu Foundation School of Engineering and Applied Science

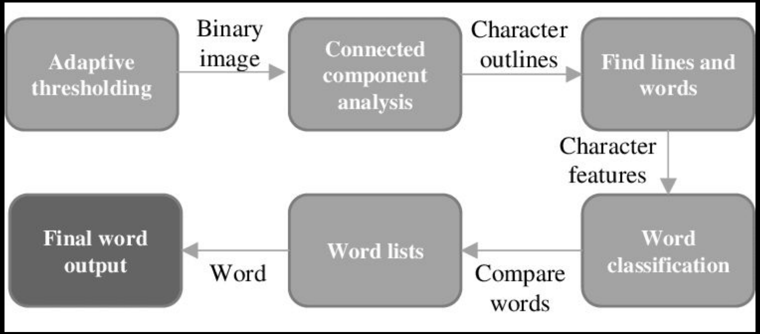
# *OCR Model: Tesseract*

1. Train

Trained using Tesseract with annotated images of 2000 tables, based on English-language pretrained model

- method: single-line page segmentation(psm=7)
- datasize: 12.8k
- input: cropped single-cell images of each table
- output: retrained language model *fintabnet_full.traineddata*

Tesseract OCR



# OCR Model

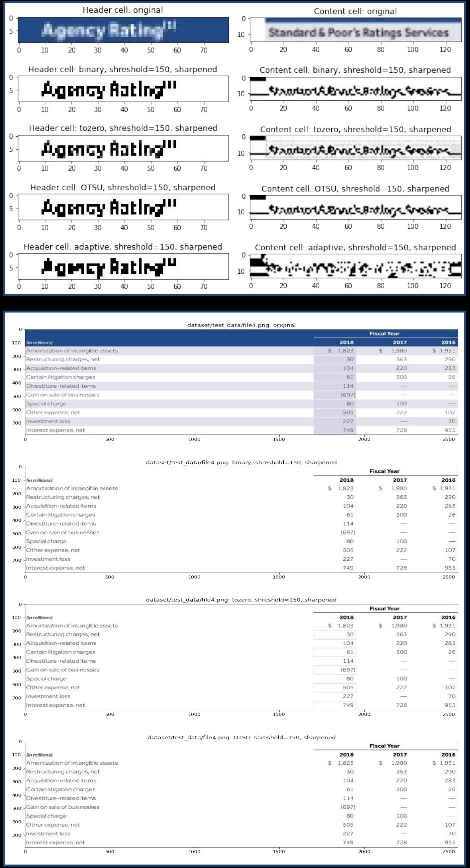
## 2. Preprocessing

**Problems for prediction:**

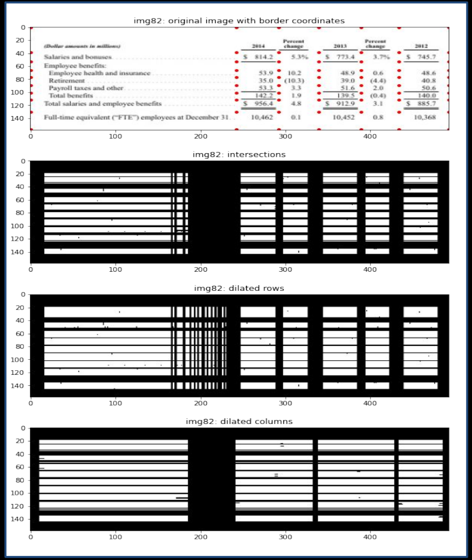
- headers with darker background colors are hard to recognize
- recognizing the whole table is harder than recognizing each cell

**Solution:**

- background removal
- reconstruct table structure, then separate each cell
- implemented using OpenCV




comparisons among background removal methods, for single cell and the entire table




reconstructing table structure, drawing borders and separating cells

# *OCR Model: Tesseract*

**3. Predict**

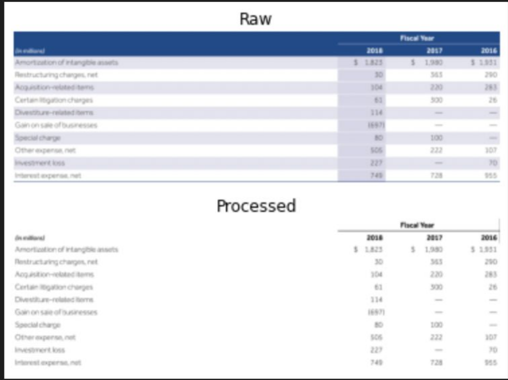
**Our OCR predictor supports two modes:**

**(1) Table mode:** recognize the entire table, output all text with built-in table structure ([psm]=6 or 12)

- faster, preserves row-column table structures

**(2) Cell mode:** recognize text per cell, then reconstruct table structure ([psm]=7)

- more accurate, preserves row structures




```
Fiscal Year
(in millions) 2018 2017 2016
Amortization of intangible assets $1,823 $1,980 $1,931
Restructuring charges, net 30 363 290
Acquisition-related items 104 220 283
Certain litigation charges 61 300 26
Divestiture-related items 114 — —
Gain on sale of businesses (697) — —
Special charge 80 100 —
Other expense, net 505 222 107
Investment loss 227 — 70
Interest expense, net 749 728 955
```

# OCR Model: Tesseract

## 4. Evaluate

- Evaluated using 1.2k single-line images of table cells (10% of the entire dataset)
- Metrics:
  - Character Error Rate (CER)
  - Word Error Rate (WER)
  - Accuracy

$$CER = (S+D+I)/N$$

$$WER = (S_w+D_w+I_w)/N_w$$

*where:*

*S = Number of Substitutions*
*D = Number of Deletions*
*I = Number of Insertions*
*N = Number of characters in reference text (aka ground truth)*

$$Accuracy = (1-WER) * 100\%$$

| Mode | Structured text (psm 6) | Unstructured text (psm 12) | Single-line (psm 7) |
|---|---|---|---|
| Train Acc. | - | - | 98.4% |
| Test Acc. | 97.0% | 97.7% | 98.9% |

# *Time Comparison*

- Average time spent based on 4 images per trial (includes response wait time) :

| | | Eye gaze | Eye gaze +CV+OCR |
|---|---|---|---|
| Mean time | | **4.5 min** | **3.5 min** |
| Time decomposition | Model prediction | / | **13 sec** |
| | Initiation | 70 sec | 70 sec |
| | Calibration | 70 sec | 30 sec |
| | Experiment | 130 sec | 90 sec |
| | Preprocessing | 0.67 sec | 0.06 sec |
| | AOI prediction | 0.06 sec | 0.01 sec |

*Note: Initiation time doesn't vary with # images; each calibration attempt takes about 30s; experiment time fluctuates with fixation data validity; other time components are linear varying with # images*

# Conclusion

# *Conclusion*

- Implemented transfer learning to develop a YOLOv5 object detection model and a Tesseract OCR model to identify tables in a document and recognize texts
- Designed an end-to-end table extraction pipeline to automatically extract the table in AOI based on eye fixations, and returns it in tabular format.
- Built a Python package `pygazeTE`, wrapping up all functions used in the project for preprocessing, visualizing, and predicting. More information can be found in the repo: https://github.com/ybliu9/pygazeTE

# Future Work

# *Future Work*

- **Eye gaze data quality improvements**: The accuracy of eye gaze data was unstable caused by calibration inaccuracy and trivial posture changes. Improving eye gaze validation would lead to more consistent results.
- **Generalization to more diverse tables**: Our current solution was not tested on unstructured tables or tables with multi-level headers.
- **Deep learning methods for table structure**: We used OpenCV's image processing techniques to recognize table structures, but DL solutions would allow training and tuning to provide more flexible and accurate results.

***Thanks for listening!***