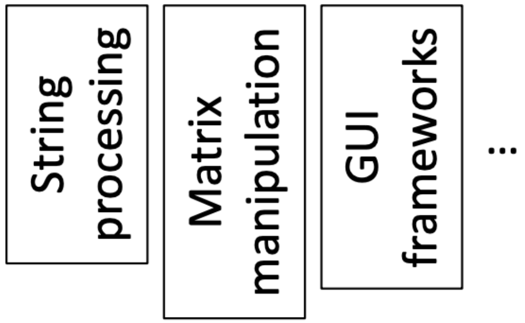
HINT

Deploying Sensitive AI Workflows Using Trial-And-Error Programming

Professor: Abhilash Jindal

Final Presentation of Vishal Singh 2018CS50426. Course COD891 Minor Project

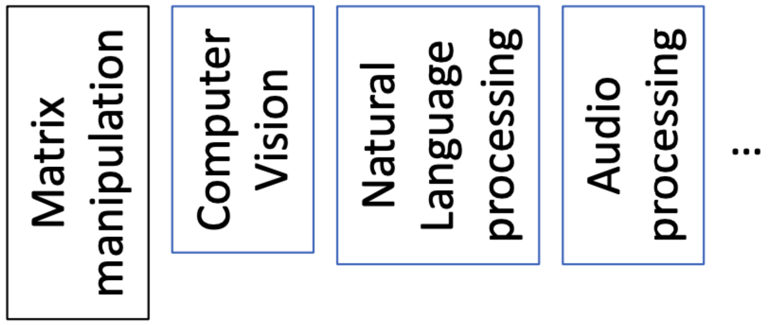
Traditional application development



Reliable building blocks

```
sort([3,4,2]) -> [2,3,4]
reverse("HINT") -> "TNIH"
set_color("#1eb4e9")
```

Flaky workflow development



Flaky building blocks:

```
ocr("doc.png") -> ["H1NT: Deploying..", 0.90]
semantic("App doesn't work. Frustrating!") -> [-0.6, 0.90]
caption("surf.jpeg") -> ["man riding a bike", 0.96]
```

Modern ML workflow development fundamentally differs from application development because the workflow building blocks are flaky.

Reasons for flow flakiness

1. Unseen input variations
   1. Rotated, skewed, blurred, torn or glued pages
   2. Poor lighting, reflection and shadows on pages
   3. 50 US states with some updating licenses every year. New format may be work well with extractor
   4. Flow may not expect renter lease agreement as a valid address proof.
   5. Handwritten text not working well

Reasons for flow flakiness (2)

2. Model errors

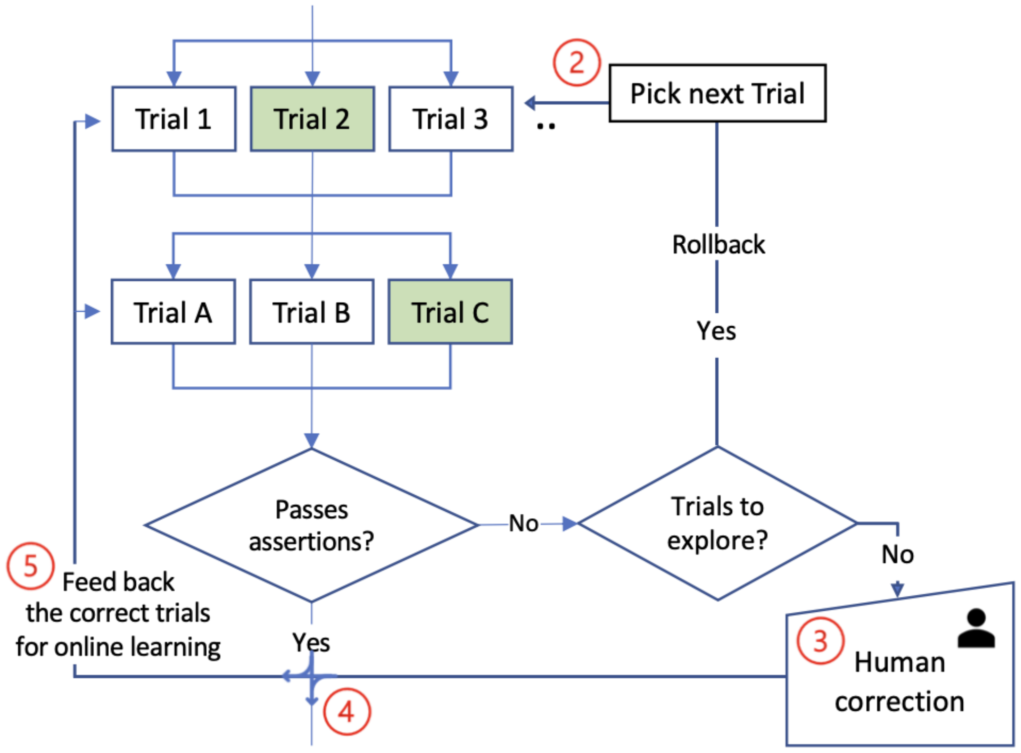
- **Model accuracy:** OCR engines making mistakes. Recognize 4 as 9 and o as 0 etc.
- **Model bias:** Overfit Problem
- **Semantic errors:** Output may be semantically incorrect. Generating name as “Tata Motors” from income proof

Programming Model

```
def to_date(date_str: str) -> Trial[Date]:
  return Trial[datetime.strptime(date_str , "%d.%m.%y").date(),
               datetime.strptime(date_str , "%m.%d.%y").date()]
//to_date('08.01.1990') -> Trial[1 August 1990, 8 January 1990]
```

(a) Programmers can specify uncertain Trials.

- Trial-and-error programming
- Associated runtime where the programmers are not burdened with actively correcting this flakiness



(b) Runtime system automatically tries different Trials. Successful Trials are fed back for online learning.

Programming Model (2)

- **Describing correctness**
  - Specify **constraints/checks/assertions** that intermediate states are expected to meet

  Examples:
  - Verify typing information (e.g. output is integer)
  - Verify model confidence levels
  - date of issue >= date of birth + 18 years
  - Also check for extracted data validation in a third party database

Programming Model (3)

- **Describing uncertainty**
  - **Eager Trial:** All possibilities are immediately available
    - 01.08.1990 can be understood as 1 August 1990 or as 8 January 1990
    - top-K results of ML models
  - **Lazy Trial:** Alternative possibilities may be provided lazily
    - Trying different ML Models
    - Microsoft Form Recognizer API, Google Document AI, and Amazon Textract

Assumption for Rollback Computation

- Rollback to previous state in general is hard
  - General purpose rollback mechanisms have prohibitively high overheads
  - Side-effects such as system calls (write to disk etc)
- Distributed computing frameworks such as MapReduce, Dryad and Spark assume that all tasks in execution graph are both deterministic and stateless
  - With that assumption, rollback becomes easy

Card Processing Workflow

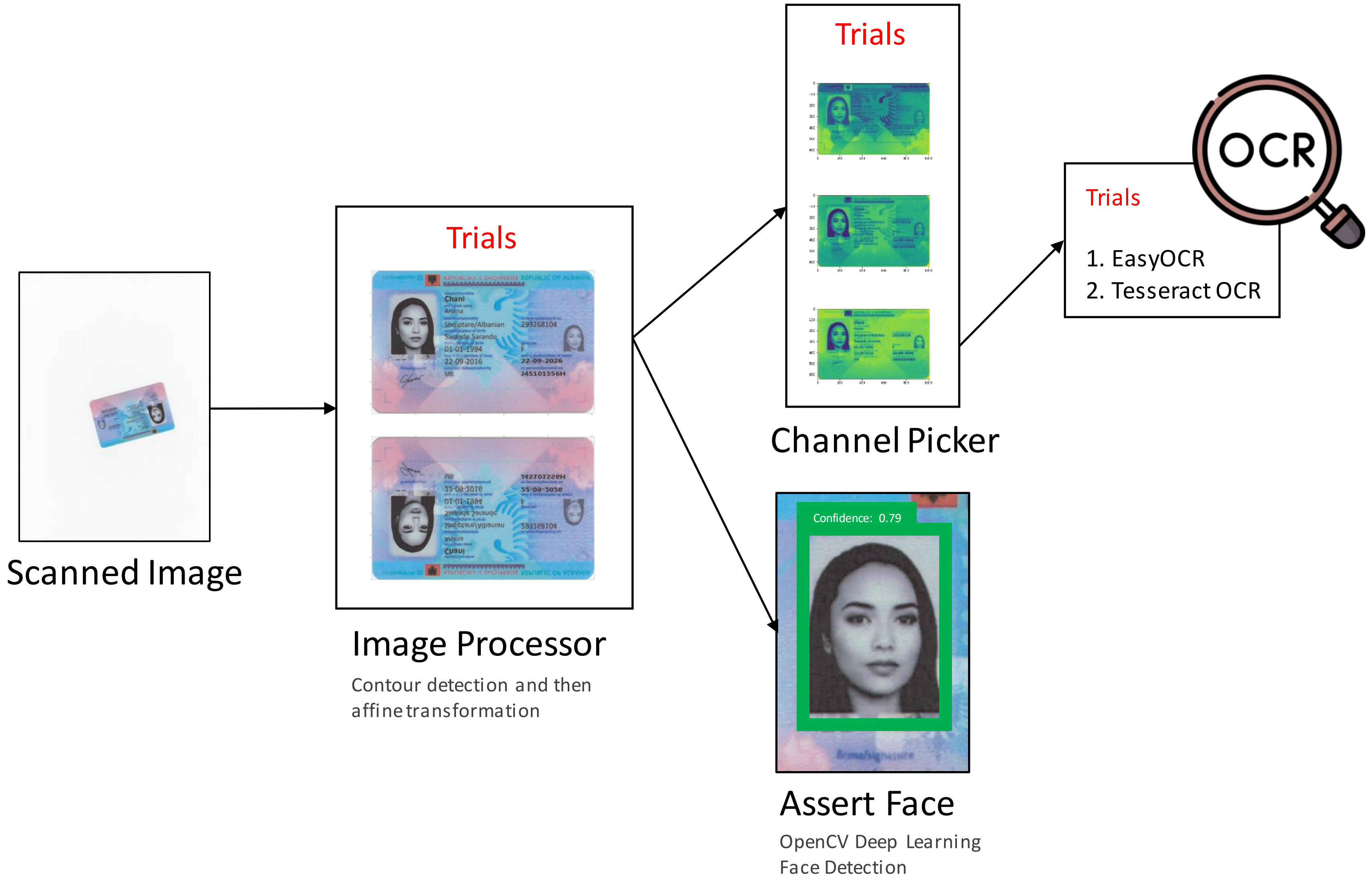
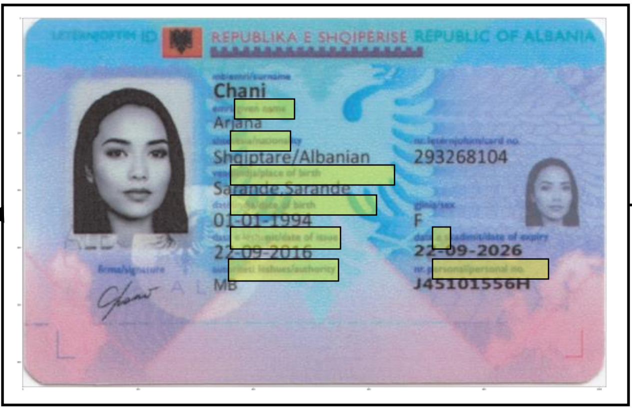


Image: https://www.flaticon.com/, MIDV Dataset

Card Processing Workflow (2)

Trials: Multiple Trained Models


REPUBLIKA E SHQIPËRISË REPUBLIC OF ALBANIA

Chani

Arjana

Shqiptare/Albanian

293268104

Sarande Sarande

01-01-1994

F

22-09-2016

22-09-2026

MB

J45101556H

Card Processor

```
Name: Arjana
Surname: Chani
Nationality: Shqiptare/Albanian
Place of Birth: Sarande
Gender: F
Date of Birth: 01-01-1994
Date of Issue: 22-09-2016
Date of Expiry: 22-09-2026
```

Out putter

OCR

Trials
1. EasyOCR
2. Tesseract OCR

OCR To Text

Use fuzzy matching to check if name, date present in OCR Text

Assert Card Text

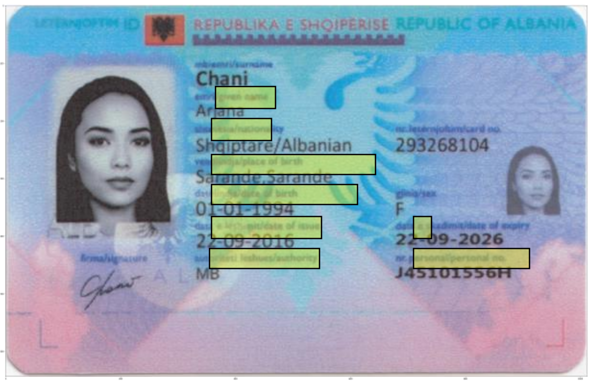
Image: https://www.flaticon.com/, MIDV Dataset

Card Processing Workflow (3)

OCR

Trials

1. EasyOCR
2. Tesseract OCR



REPUBLIKA E SHQIPERISE REPUBLIC OF ALBANIA

Chani

Arjana

Shqiptare/Albanian

293268104

Sarande Sarande

01-01-1994

F

22-09-2016

22-09-2026

MB

J45101556H

Card Processor

```
Name: Arjana
Surname: Chani
Nationality: Shqiptare/Albanian
Place of Birth: Sarande
Gender: F
Date of Birth: 01-01-1994
Date of Issue: 22-09-2016
Date of Expiry: 22-09-2026
```

Out putter

1. Use **Layout LM** (Pre-training of Text and Layout for Document Image Understanding) for data extraction.
2. Will add multiple trained models for different categories of expected cards.

Image: https://www.flaticon.com/, MIDV Dataset

Status

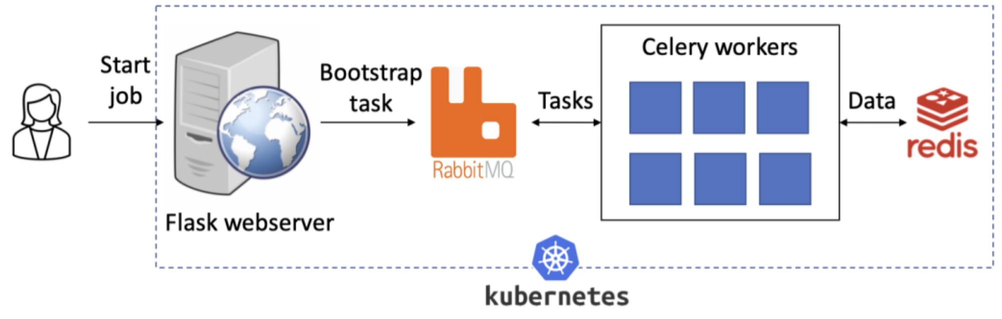


Figure 4. HINT architecture.

- HINT has a scalable architecture (can scale to 50+ nodes) and has fault tolerance
- Allows developers to write assertions
- Right now, if checks fail then users can correct flakiness

Current Goals

- Automating exploration of Trials and only ask users if no trials combination is found
- Make real world workflows with large real dataset
- Perform evaluation metrics

Future Goals for this Project

- Add support for continual learning
- Improve scalability

Status (2)

- Currently implemented eager trials support in HINT with simple rollback computation that works well for a tree
- Added Lazy Trial support
- Formalized rollback to next trial problem
- Added some unit tests for code.
- Working on making a real-world workflow using large real-world dataset
  - Worked on Card Processing Workflow
  - Goal is to now work for Industrial quality control workflow and add evaluation metrics

Thank You

Comparison with Constrained Programming

```
dob = to_date ('08.01.1990')
if dob.month == 'August':
        discount=True
doi = datetime.strptime('13 July 2008',..).date()
assert doi >= dob.replace(year = dob.year + 18)
```

- Constraint programming is typically run-in isolation
- First try 1 August 1990. It fails assertion
- Then try 8 January 1990. It passes assertion.
- But you also have to rollback and change discount to False and all other transitive side-effects caused by the failed Trial of dob.
- Conventional constraint programming does not handle this