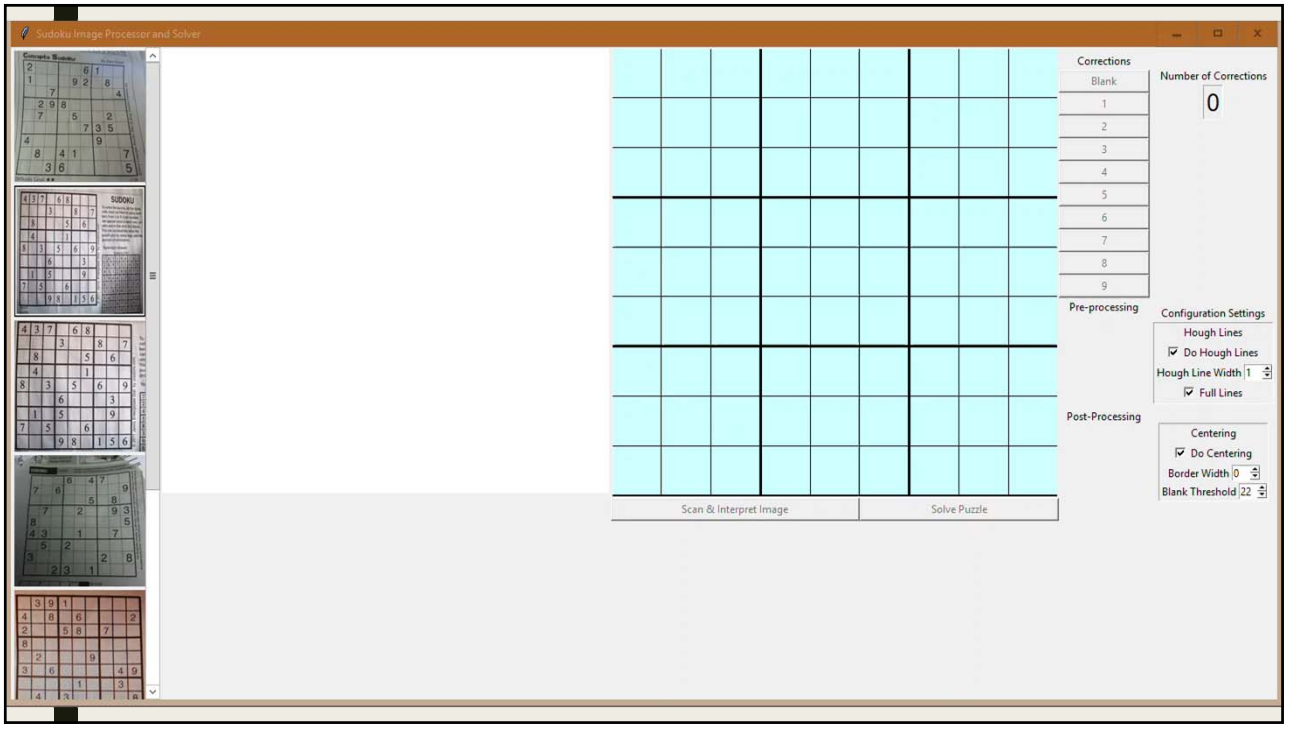
# SOLVE SUDOKU

Using Machine Learning and OpenCV



## Breakdown of the Solution

1. Import and use OpenCV to parse picture
2. Train a machine learning model
3. Apply model to recognize pictures
4. Solve the puzzle and display the results

## Learnings from OpenCV portion

Original Picture

# Learnings from OpenCV portion

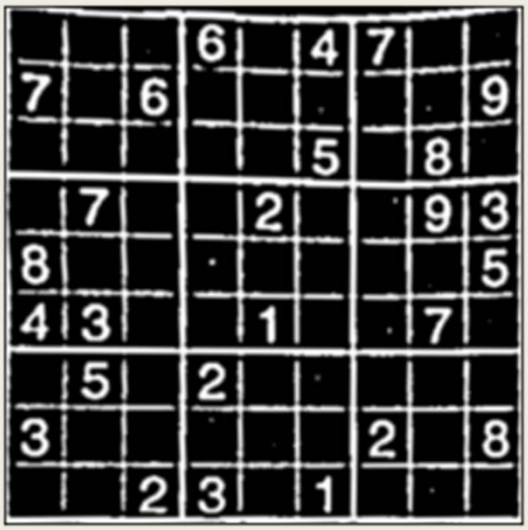
Filter, Blur, Flip

# Learnings from OpenCV portion

Corner Detection

# Learnings from OpenCV portion

Parse Board

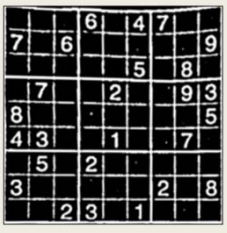


# Learnings from OpenCV portion

Original Picture

Filter and Blur

Corner Detection



Parse Board

Individual Pictures

# Machine Learning Model

- MNIST CNN model with >99% accuracy
  - Detecting our sudoku images – 23%
- Hypothesized due to
  - Printed vs Handwritten
  - Image imperfections:
    - Size
    - Scale
    - Borders

| Actual | CNN MNIST |
|---|---|
| 6 | FALSE |
| 4 | FALSE |
| 7 | TRUE |
| 7 | FALSE |
| 6 | FALSE |
| 9 | FALSE |
| 5 | TRUE |
| 8 | FALSE |
| 7 | FALSE |
| 2 | FALSE |
| 9 | FALSE |
| 3 | TRUE |
| 8 | FALSE |
| 5 | FALSE |
| 4 | FALSE |
| 3 | FALSE |
| 1 | FALSE |
| 7 | FALSE |
| 5 | TRUE |
| 2 | TRUE |
| 3 | FALSE |
| 2 | TRUE |
| 8 | FALSE |
| 2 | FALSE |
| 3 | FALSE |
| 1 | FALSE |

23.1%

# Created printed digit data set

Suduko MNIST New data set

## Characteristics of new data set:

- Created 540 images of printed digits using Adobe InDesign
- Includes 60 different fonts of digits 1-9
- Applied 70 transformations: added borders, shifts, skew, and noise
- Contains 4,200 images for each digit 1-9
- **Contains 37,800 images & corresponding labels**

# Examples of printed digit data set

Suduko

Packages used: OpenCV, numpy, matplotlib, and itertools

# Machine Learning Model

- MNIST CNN model with >99% accuracy
  - Detecting our sudoku images – 23%
- Hypothesized due to
  - Printed vs Handwritten
  - Image imperfections:
    - Size
    - Scale
    - Borders
- New Data CNN model with >99% accuracy
  - Detecting our sudoku images – 85%

| Actual | CNN MNIST | CNN DigitData |
|---|---|---|
| 6 | FALSE | TRUE |
| 4 | FALSE | FALSE |
| 7 | TRUE | FALSE |
| 7 | FALSE | TRUE |
| 6 | FALSE | TRUE |
| 9 | FALSE | TRUE |
| 5 | TRUE | TRUE |
| 8 | FALSE | FALSE |
| 7 | FALSE | TRUE |
| 2 | FALSE | TRUE |
| 9 | FALSE | FALSE |
| 3 | TRUE | TRUE |
| 8 | FALSE | TRUE |
| 5 | FALSE | TRUE |
| 4 | FALSE | TRUE |
| 3 | FALSE | TRUE |
| 1 | FALSE | TRUE |
| 7 | FALSE | TRUE |
| 5 | TRUE | TRUE |
| 2 | TRUE | TRUE |
| 3 | FALSE | TRUE |
| 2 | TRUE | TRUE |
| 8 | FALSE | TRUE |
| 2 | FALSE | TRUE |
| 3 | FALSE | TRUE |
| 1 | FALSE | TRUE |
| | 23.1% | 84.6% |

# Machine Learning Model

- MNIST CNN model with >99% accuracy
  - Detecting our sudoku images – 23%
- Hypothesized due to
  - Printed vs Handwritten
  - Image imperfections:
    - Size
    - Scale
    - Borders
- New Data CNN model with >99% accuracy
  - Detecting our sudoku images – 85%
- Additional cleaning and centering input images
  - Detecting our sudoku images – 96%

| Actual | CNN MNIST | CNN DigitData | CNN DigitData (Cleaned) |
|---|---|---|---|
| 6 | FALSE | TRUE | TRUE |
| 4 | FALSE | FALSE | FALSE |
| 7 | TRUE | FALSE | TRUE |
| 7 | FALSE | TRUE | TRUE |
| 6 | FALSE | TRUE | TRUE |
| 9 | FALSE | TRUE | TRUE |
| 5 | TRUE | TRUE | TRUE |
| 8 | FALSE | FALSE | TRUE |
| 7 | FALSE | TRUE | TRUE |
| 2 | FALSE | TRUE | TRUE |
| 9 | FALSE | FALSE | TRUE |
| 3 | TRUE | TRUE | TRUE |
| 8 | FALSE | TRUE | TRUE |
| 5 | FALSE | TRUE | TRUE |
| 4 | FALSE | TRUE | TRUE |
| 3 | FALSE | TRUE | TRUE |
| 1 | FALSE | TRUE | TRUE |
| 7 | FALSE | TRUE | TRUE |
| 5 | TRUE | TRUE | TRUE |
| 2 | TRUE | TRUE | TRUE |
| 3 | FALSE | TRUE | TRUE |
| 2 | TRUE | TRUE | TRUE |
| 8 | FALSE | TRUE | TRUE |
| 2 | FALSE | TRUE | TRUE |
| 3 | FALSE | TRUE | TRUE |
| 1 | FALSE | TRUE | TRUE |
| | 23.1% | 84.6% | 96.2% |

# Lessons Learned

- Start with the right training data
- Improving detection accuracy can be done by:
  - Cleaning images
  - “Dirty-ing training data”
- Best results come from doing both