OCR with Mathematical Equation

Agarwal Mohit, Long Yun, Ma Meng, Wang Hongyang
Georgia Institute of Technology

OVERVIEW

OCR (Optical Character Recognition)

- Electronic conversion of images of handwritten or printed text into machine-encoded text
- Current OCR softwares can go up to character-to-character accuracy of 99% for printed characters and 95% for handwritten data

Uses

- Replacement of manual work required in Data Entry and digitizing handwritten/printed manuscripts
- Automatic number-plate/pin-code recognition
- Helpful for visually impaired people

MOTIVATION

OCR performs poorly when it deals with scientific papers and mathematical equations. This is due to presence of complex mathematical symbols, specific structures and two-dimensional information with variations in fonts and positions

OBJECTIVE

We aim at building a Software Tool that takes a scanned version of mathematical equations as input and outputs the LaTeX format of the equation along with its solution.

We currently focus on

- Printed equations
- 60 Symbol Set : (0-9),(a-z), Math operators and Greek symbols
- 4 Fonts : Helvetica, Computer Modern, Verdana, Comic Sans
- Preprocessed image : No skewness/distortion ; with proper lighting

METHODOLOGY

OCR Software Development is composed of 4 blocks

- Character Segmentation (Meng Ma and Hongyang Wang)
- Symbol Recognition (Mohit Agarwal)
- Equation Recovery and Solving (Meng Ma and Hongyang Wang)
- MATLAB GUI and App (Yun Long)

1.CHARACTER SEGMENTATION

In a scanned equation, separate each character and normalize it to a 20x20 image.

- Finds connected regions, and treats each connected region as a separate character
- To identify characters like, $i, j, =, \div$ with multiple connected regions, detect all the connected regions and then use a-priori information to combine them into one character
- Displays the bounding box around each character and resizes it to 20x20 matrix

2.SYMBOL RECOGNITION

To recognize each 20x20 image matrix as one of the 60 Character Symbol,

Training Data

- Manually prepared training data set by printing symbols, scanning and running *character segmentation* on it.
- Size: 40 samples of each symbol (10 for one font) making it a total of 2400 samples

ML Algorithm

We implemented two different classification algorithms and chose one for final OCR software according to their performance and computational complexity

1. Neural Networks
   - Input Layer: 400 (20x20 Image)
   - Hidden Layer: 100 Neurons
   - Output Layer: 60 (for each symbol)
2. SVM (One vs One Classification)

Validation

Divided training data into two parts, 70% for learning model parameters and 30% for testing. Both the algorithms had similar performance in terms of accuracy. We choose Neural Networks for final OCR Software due to its reduced computational time. SVM being a binary classifier, required ${}^{60}C_2$ One vs One Classifiers making it more complex than Neural Networks.

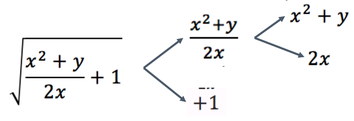
Accuracy	Training Set (70%)	Testing Set (30%)
Neural Networks	99.1667%	98.9286%
SVM Classifier	98.4722%	98.6111%

Symbol Recognition outputs the Label (ClassID) of each symbol recognized.

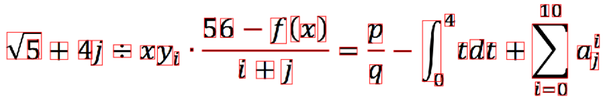
EQUATION RECOVERY & SOLVING

Recover the whole equation in machine-readable format, generates LaTeX code and solves the equation

- Handles subscript/superscript using position and size information $x^2 + y$
- Special structures like Square root, fraction, summation/product can be solved treating line under square root or summation/product symbol as separate equation $\sqrt{x^2+y}$ $\sum_{i=0}^{10} x_i^2 + y$
- Most complicated scenario is combination of three structures which can be solved using recursive way to convert into simple forms



- This equation is converted into LaTeX string by mapping each symbol with its LaTeX code
- If the equation is solvable, we parse the equation and use MATLAB equation solver to solve simple equations




```
latex =

\sqrt{5} +4j\div xy^{}_{i}\cdot \frac{56-f(x)}{i+j} =\frac{p}{q} -\int ^{4}_{0}tdt+\sum ^{10}_{i=0} a^{i}_{j}
```

GUI AND APP

We developed a GUI in MATLAB, which combines all of the steps above, takes image as an input and shows its LaTeX code with solution.

We also developed an Android Application to transfer the photos taken from smartphone to computer.