



Rendering Handwritten Equations



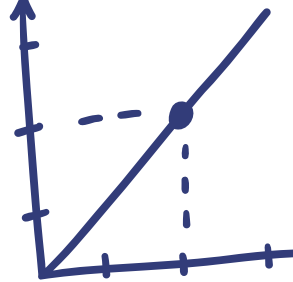
Martijn de Vries

Capstone Project Presentation
June 12th, 2023


$$\sqrt{\frac{3}{4}} = (a^2)$$



Problem Statement



Develop a tool that can render handwritten equations in digital format.

Input: an image containing an equation

Output: the equation in digital format, using LaTeX math notation





For example..

$$\log_2 8 + \log_3 9 + \log_4 16$$

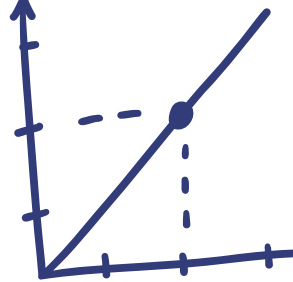


$$\text{\textbackslash}\log_2 8 + \text{\textbackslash}\log_3 9 + \text{\textbackslash}\log_4 16\text{\textbackslash}$$

How we evaluate performance



$$\log_2 8 + \log_3 9 + \log_4 16$$



Why is this complicated?

**Symbols are not
just read
left-to-right**

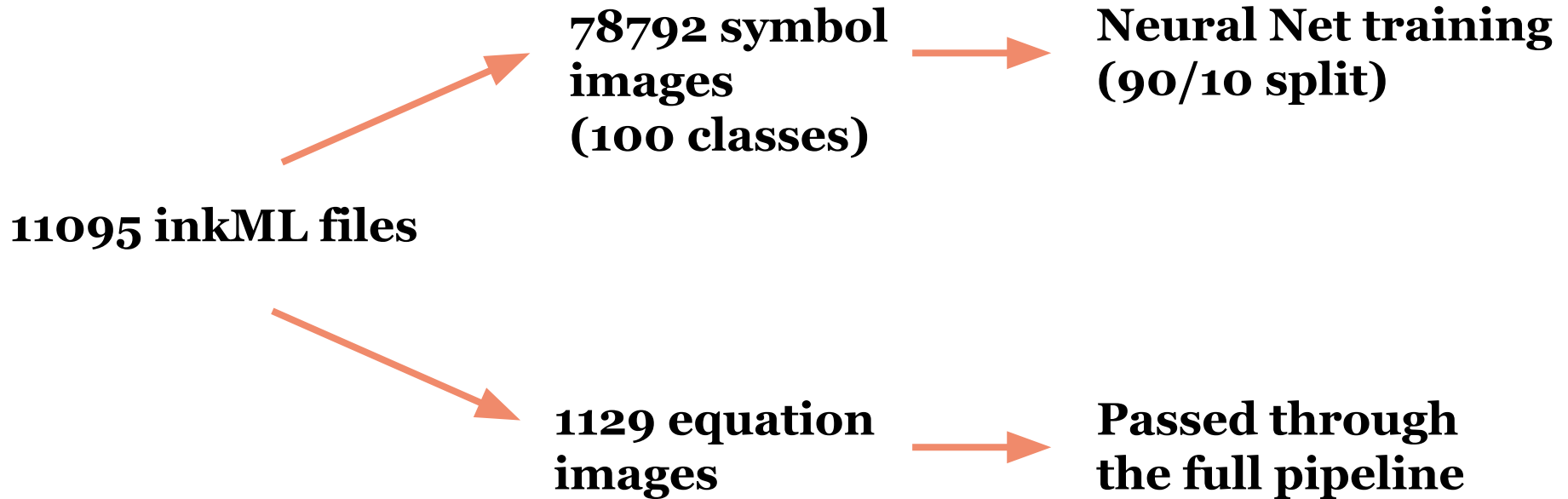
$$\int_{\log 3}^0 \frac{1}{e^t + 1} dt$$

Data

- CROHME
- 'inkML' format
- Full equation label
- Labels for each symbol
- 100 different symbols

The image shows a handwritten mathematical equation: $A = \sqrt{a + \frac{1}{\sqrt{a + \frac{1}{\sqrt{a}}}}} + \sqrt{b}$. The equation is rendered in a stylized, handwritten font with various colors (blue, green, red, orange, yellow, pink, brown) used for different parts of the symbols. Dashed lines of corresponding colors are overlaid on the equation, representing the 'inkML' format where each symbol is labeled. For example, the 'A' is labeled with a blue dashed line, the '=' with a green dashed line, and the 'a' under the first square root with a brown dashed line. The entire equation is labeled with a long purple dashed line at the top.

Data Processing



Pipeline Overview



1. Image thresholding

Change image into black/white values

2. Resolving Symbols

Figure out the symbol on the image and their order

3. Model Prediction

Use an EfficientNetBo CNN to predict labels

4. Rendering the equation

Stitch the predictions together into an equation

1) Image Tresholding

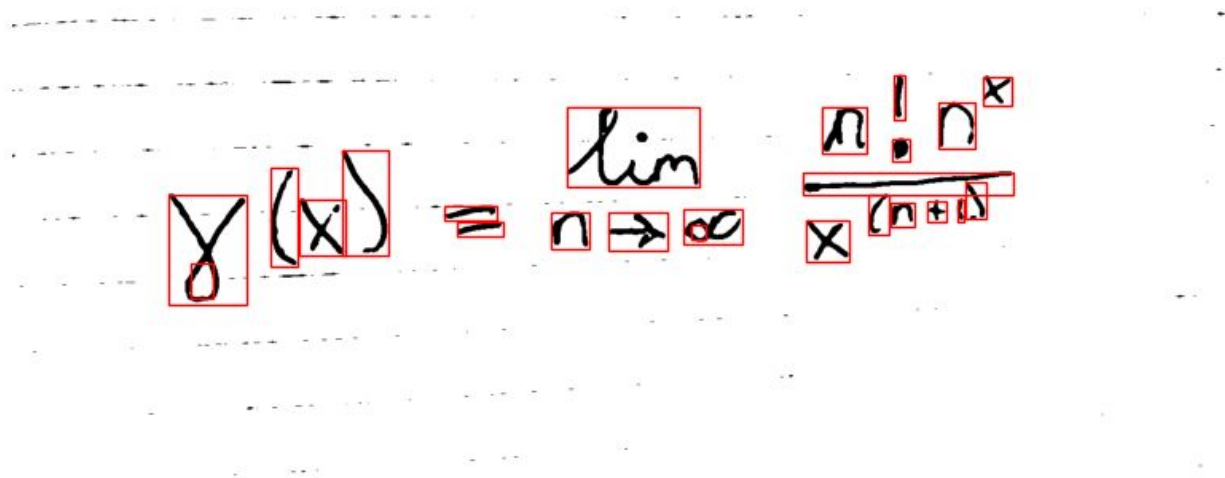
$$\gamma(x) = \lim_{n \rightarrow \infty} \frac{n! n^x}{x^{(n+1)}}$$

$$\gamma(x) = \lim_{n \rightarrow \infty} \frac{n! n^x}{x^{(n+1)}}$$

Adaptive Gaussian Tresholding

2) Resolving Symbols

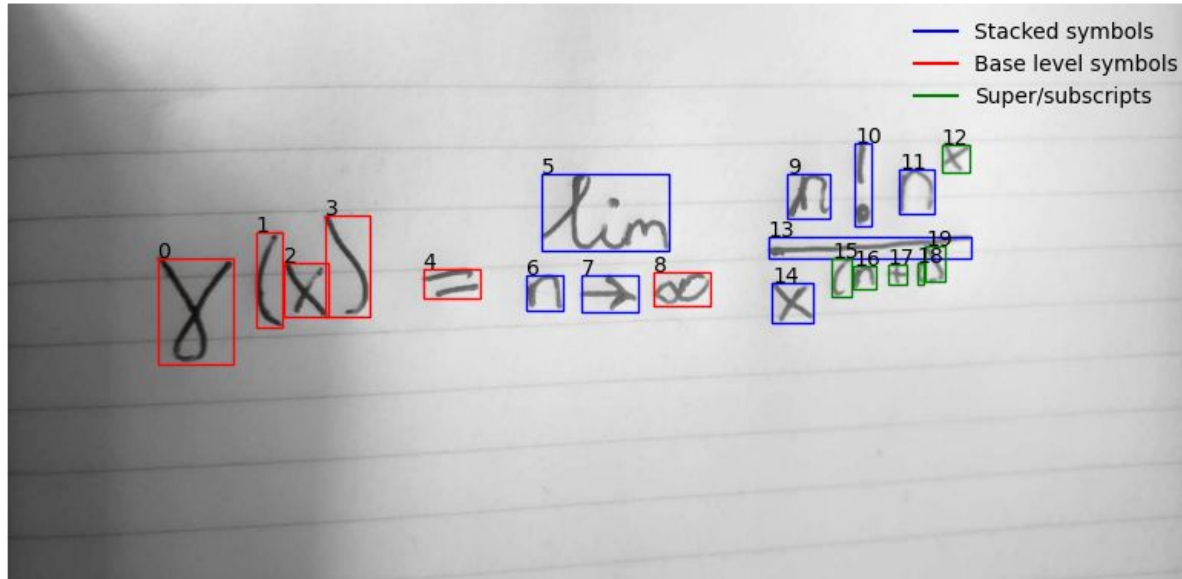
Find Contours and Bounding Boxes



- i) Remove Inner Contours
- ii) Merge Boxes

- iii) Determine Symbol order
- iv) Determine Script level

2) Resolving Symbols



- i) Remove Inner Contours
- ii) Merge Boxes

- iii) Determine Symbol order
- iv) Determine Script level

3) Model Predictions

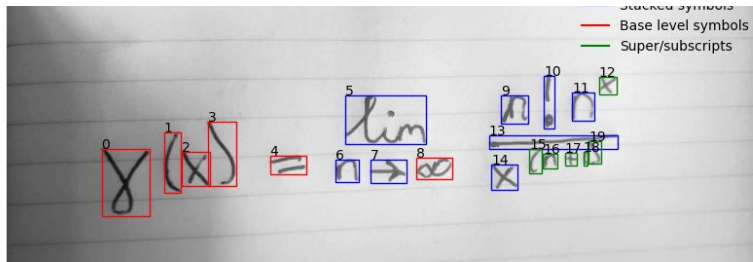
Model: EfficientNetBo CNN

95% accuracy on validation data
(*individual symbol images*)

Prediction Step:

Symbol Images ->
List of class labels

4) Render the Equation



$\gamma(X) = \lim_{n \rightarrow \infty} \frac{n! n^k}{X^{(n+1)'}}$



$$\gamma(X) = \lim_{n \rightarrow \infty} \frac{n! n^k}{X^{(n+1)'}}$$

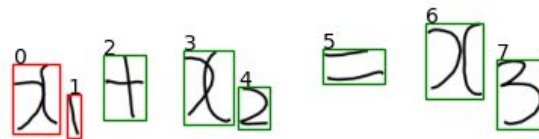
Two more examples

- Stacked symbols
- Base level symbols
- Super/subscripts

- Stacked symbols
- Base level symbols
- Super/subscripts



Prediction: $\sin^2 \theta + \cos^2 \theta = 1$



Prediction: $x_1 + x_2 = x_3$

Overall Pipeline Performance

Damerau-Levenshtein distance

Actual: ' $x_{\{i\}}-x_{\{i+1\}}+x_{\{i+2\}}$ '

Predicted: '**1** $x_{\{1\}}-x_{\{1+1\}}+x_{\{1+2\}}$ '

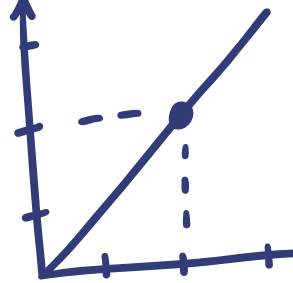
1 deletion, 3 substitutions

Normalize by actual string length:

Distance = $4/21 \approx 0.477$

Normalized DL distance	% of equations
0	15.7%
< 0.10	20.9%
< 0.50	65.9%

Final thoughts..



This is a difficult problem to solve!

Errors compound through the 4 pipeline steps

Different linewidths seem to make the predictions less accurate than the 95% accuracy in NN

Multiple Object Detection algorithms

