Technical Research

MLP data sets

Table of Contents

Open Source Data Sets

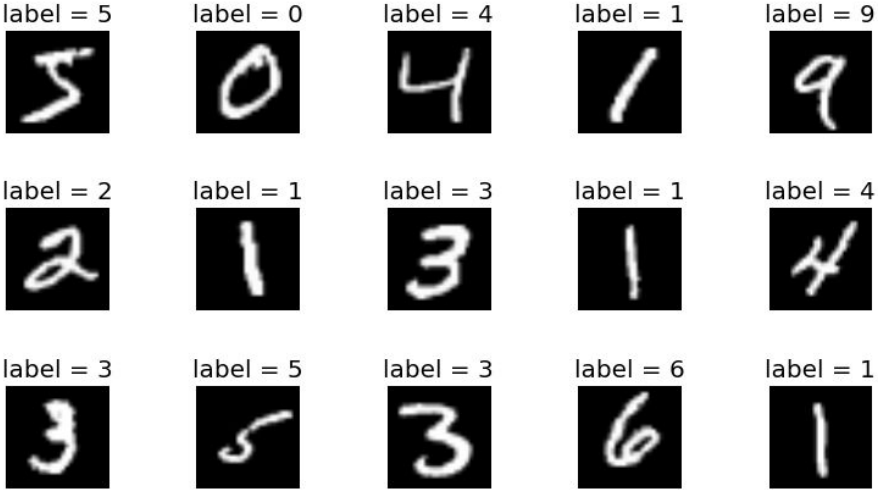
MNIST Data Set:

The MNIST Data Set is a Data Set containing over 60 000 samples of training data on handwritten digits, and a test set of over 10 000 samples. It contains images of the digits as well as the corresponding label. This can be really useful when training a model to classify handwritten digits.

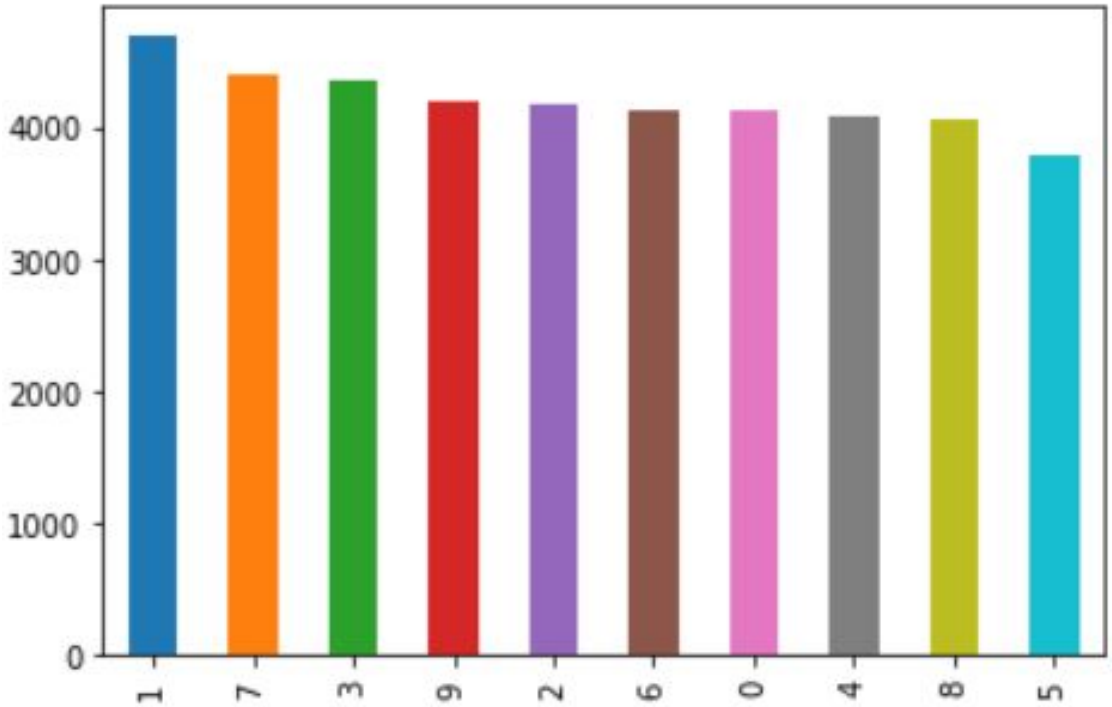
Extracting Data Set:
http://yann.lecun.com/exdb/mnist/

Article:
https://towardsdatascience.com/image-classification-in-10-minutes-with-mnist-dataset-54c35b77a38d

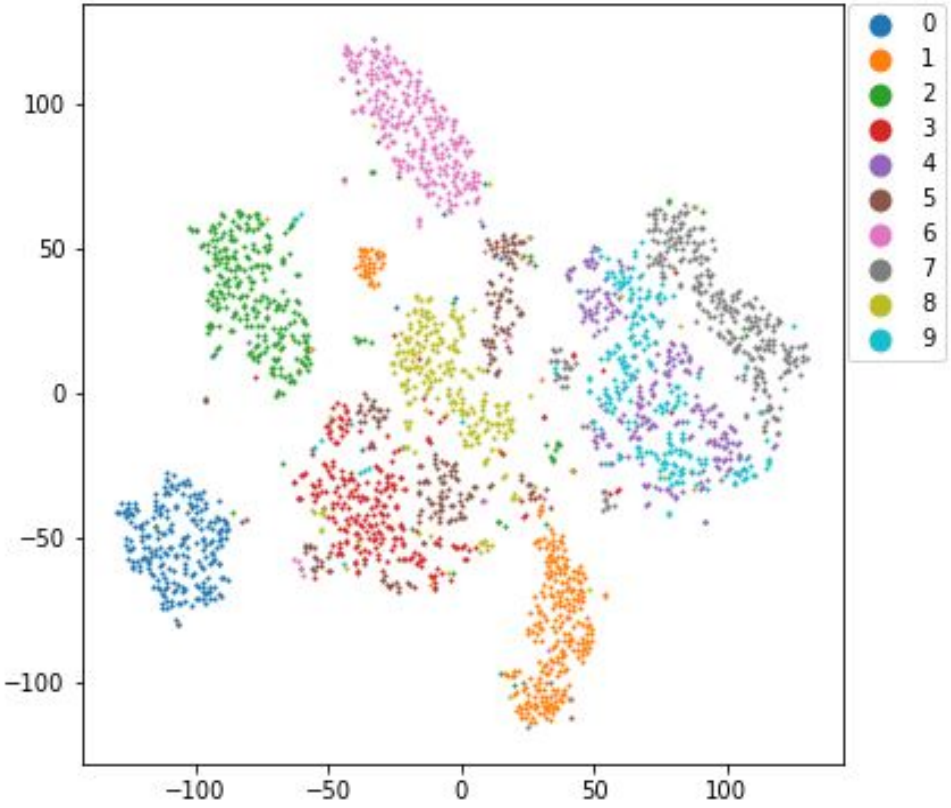
MNIST Example Data:



Distribution of Digits in the MNIST Dataset

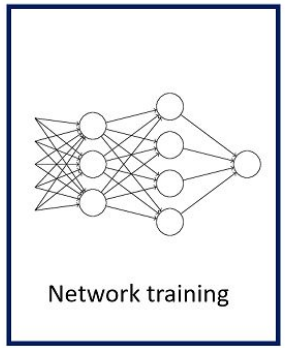


tsne representation of the MNIST dataset



MNIST Classification process

Data & Labels



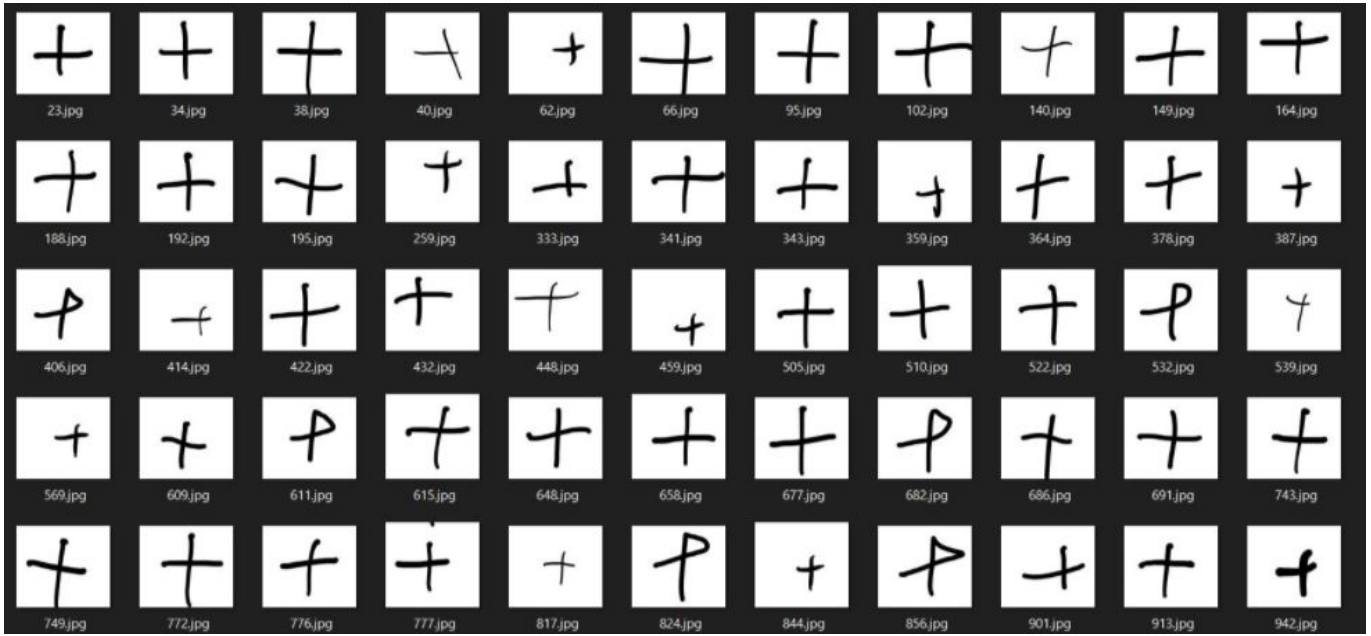
0
1
2
3
4
5
6
7
8
9

Kaggle's Mathematical Symbols:

The Kaggle Mathematical Symbols Data Set contains jpg files of: Basic Greek alphabet symbols, English alphanumeric symbols are included, All math operators, Set operators, Basic predefined math functions, Math symbols.
This Data Set can be useful for identifying Math symbols rather than just the digits as in the MNIST Data Set.

Extracting Data Set (410 MB):
https://www.kaggle.com/xainano/handwrittenmathsymbols

Kaggle's Mathematical Symbols Example Data:

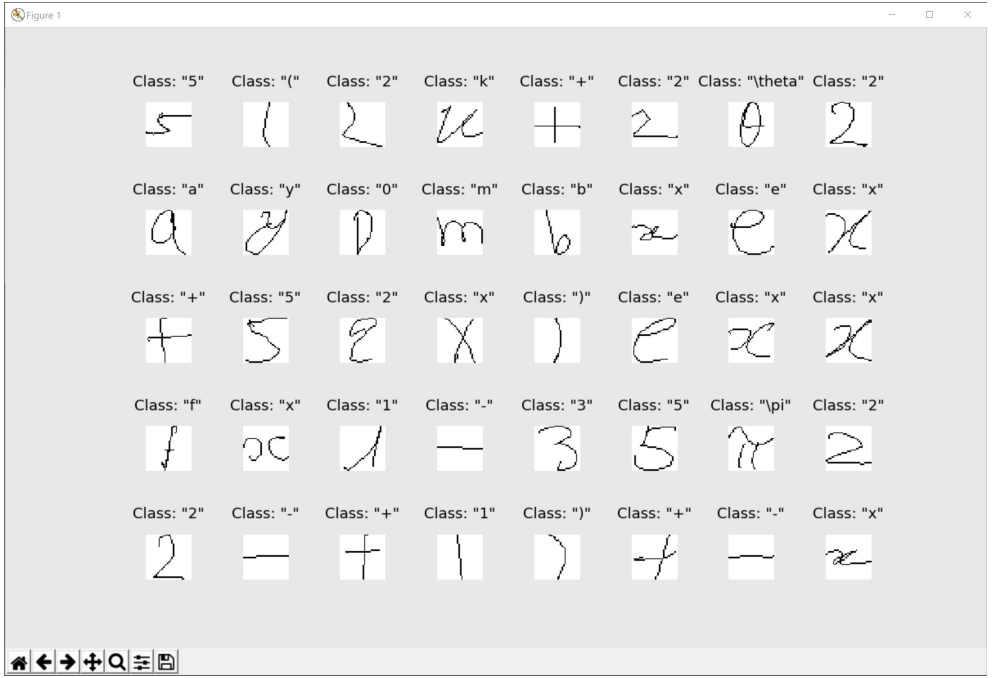


CROHME data set [data]

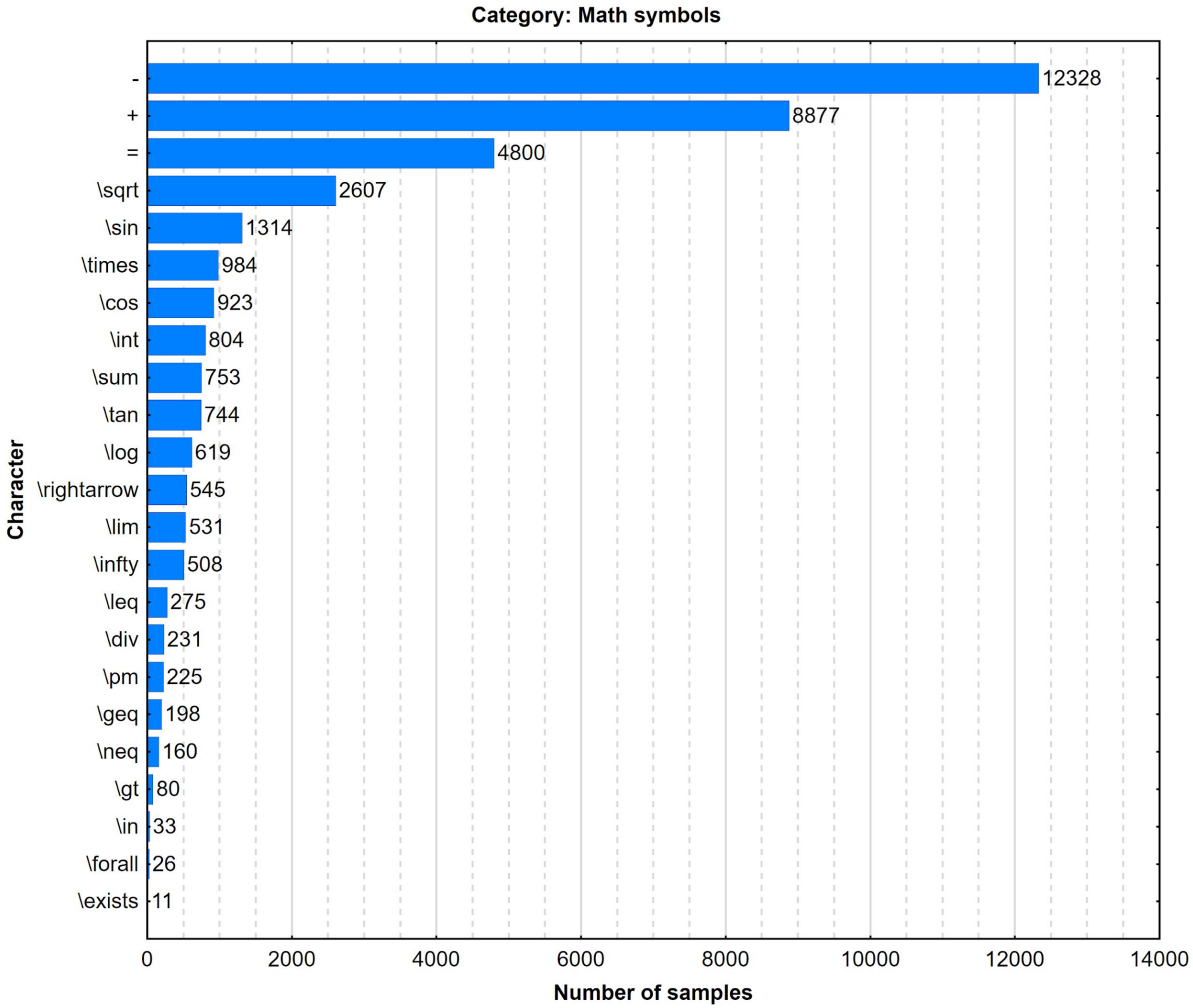
CROHME datasets originally exhibit features designed for Online-handwritting recognition task.

Apart from drawn traces being encoded, inkml files also contain trace drawing time captured.

To extract CROHME data use this github repo
https://github.com/ThomasLech/CROHME_extractor

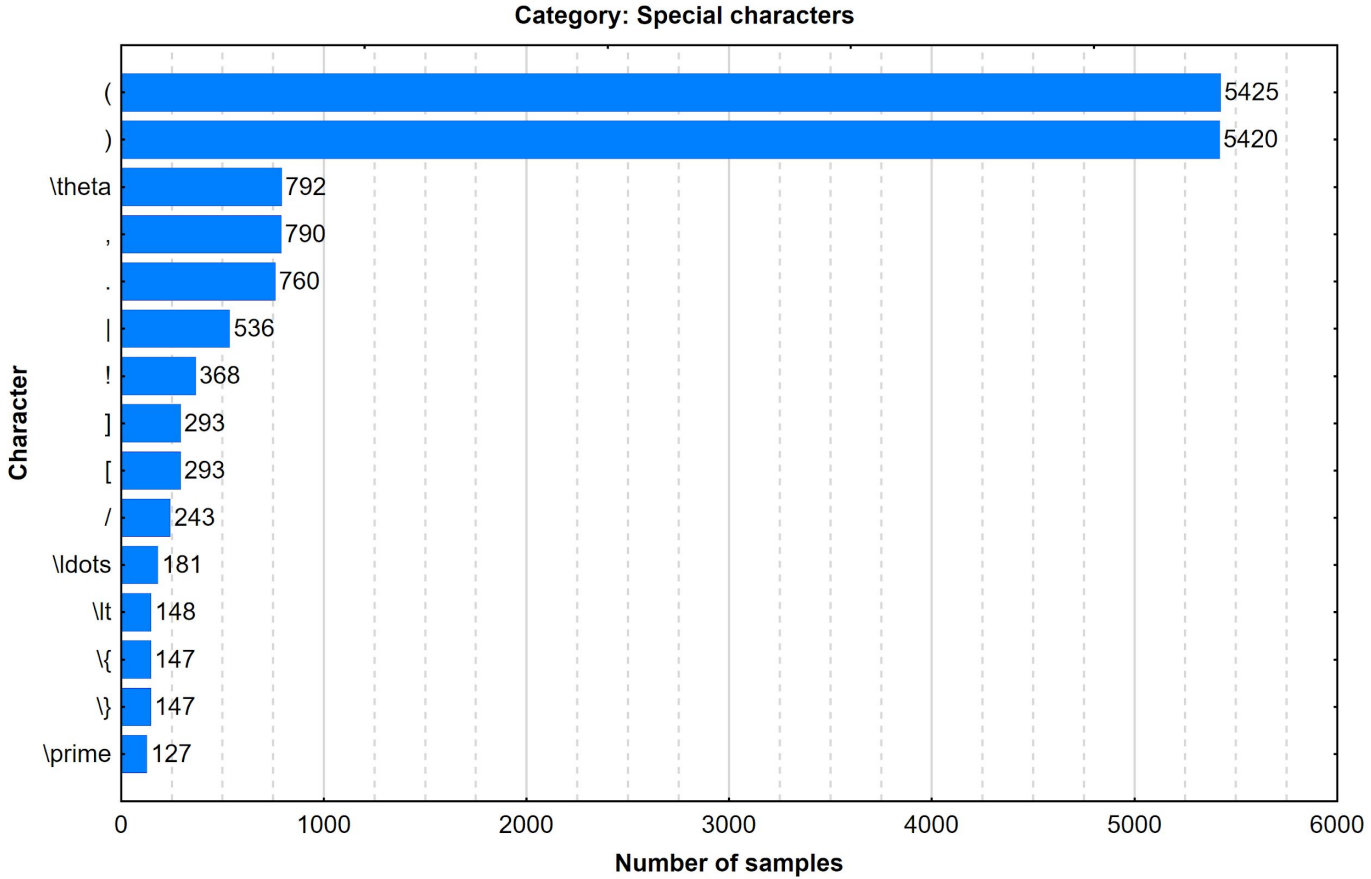


Sample from CHROHME extractor



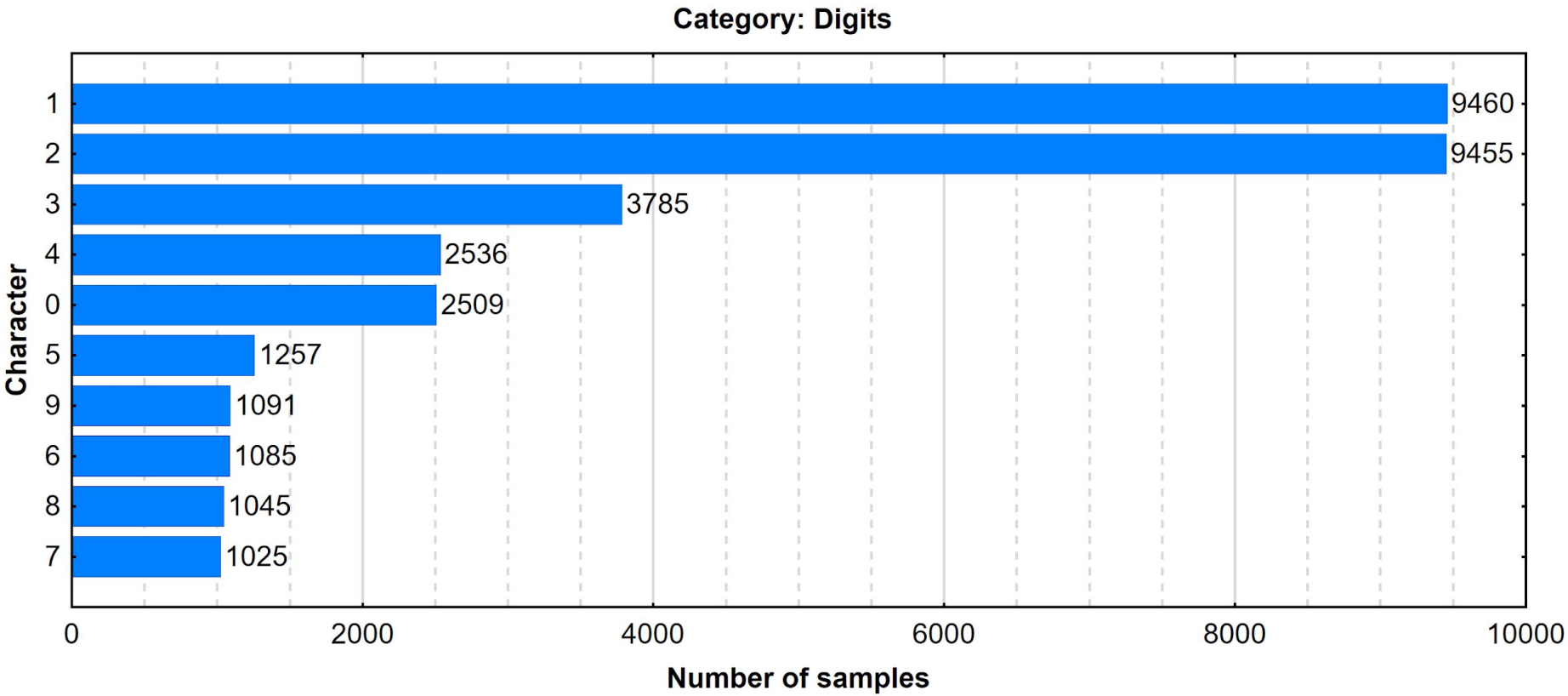
This distribution shows the occurance of Math symbols in the CROHME data set.

https://raw.githubusercontent.com/ThomasLech/CROHME_extractor/master/histograms/math_symbols_distribution.png



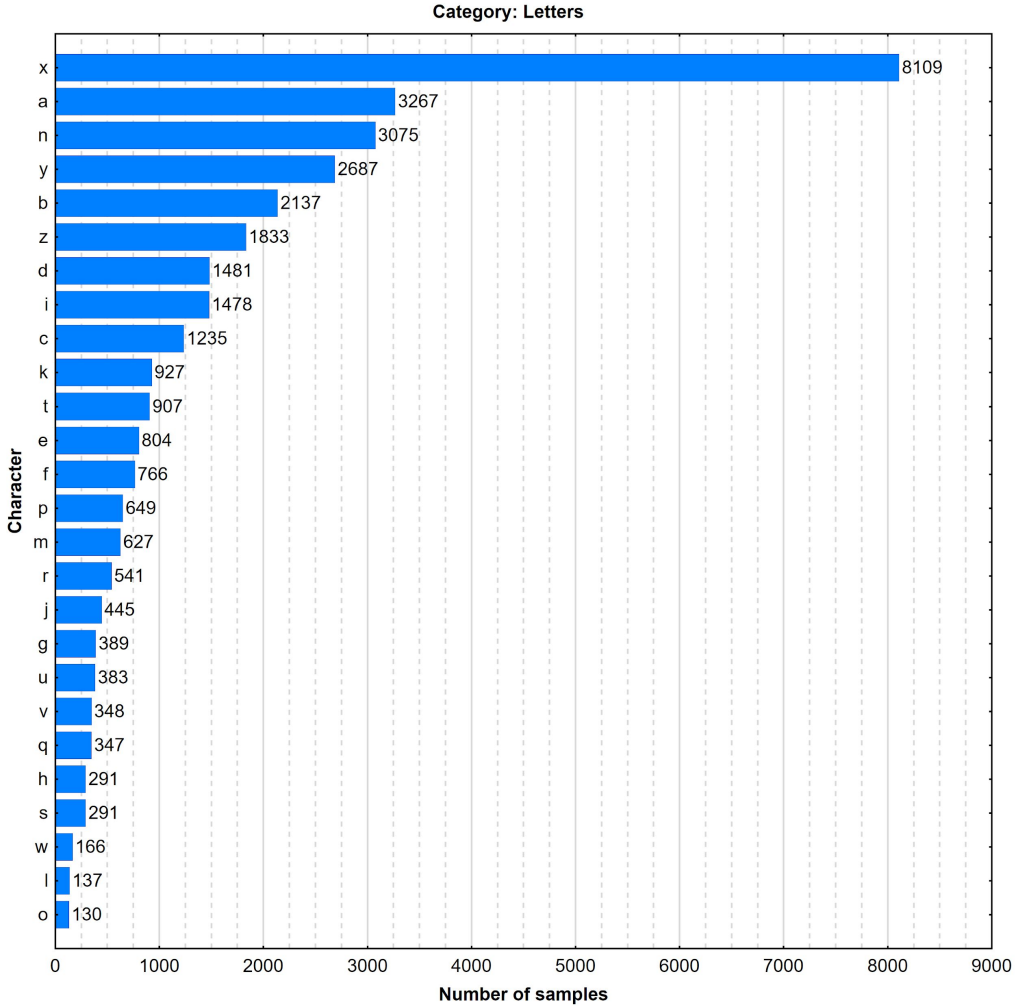
This distribution shows the occurance of Special characters in the CROHME data set.

https://raw.githubusercontent.com/ThomasLech/CROHME_extractor/master/histograms/special_characters_distribution.png



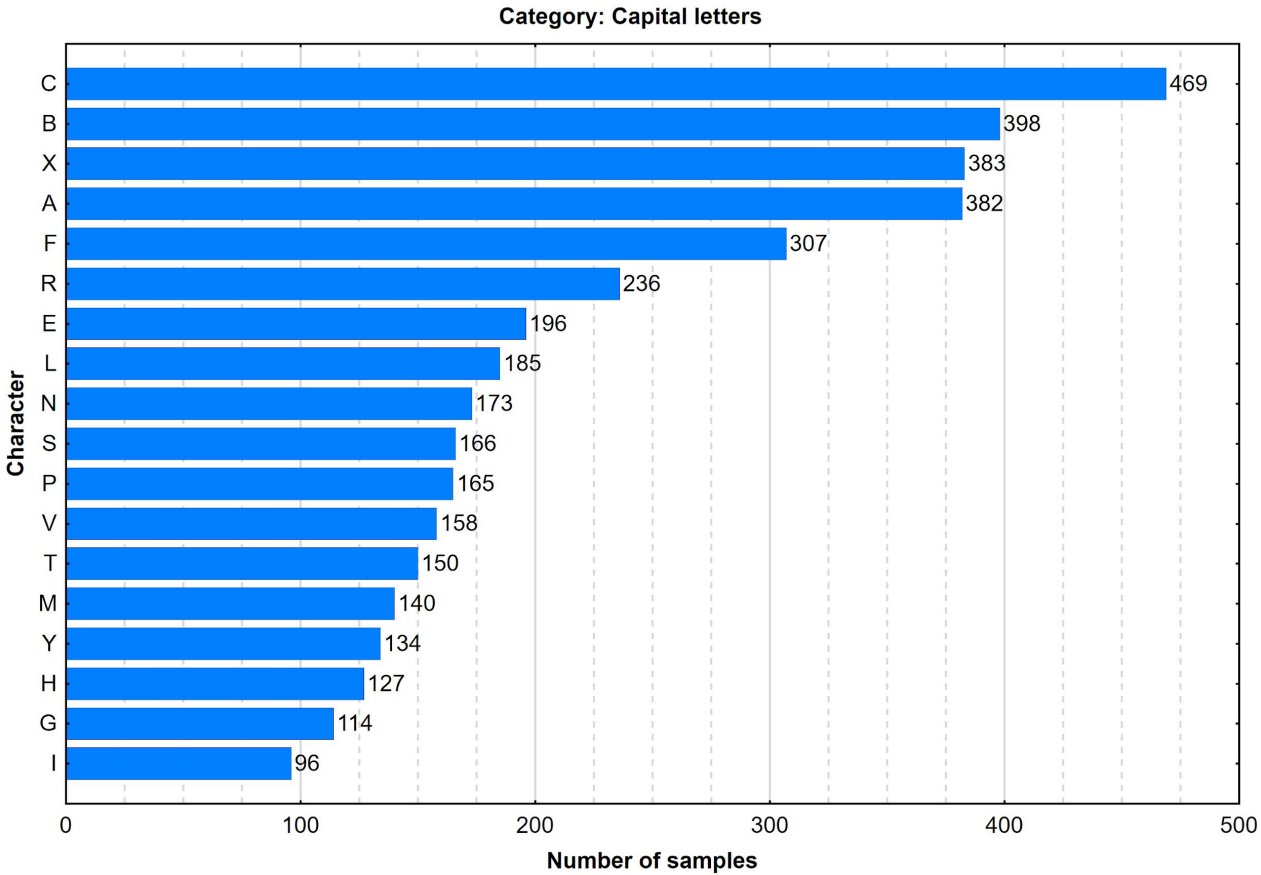
Histogram showing the distribution of digits in the CROHME dataset

https://raw.githubusercontent.com/ThomasLech/CROHME_extractor/master/histograms/digits_distribution.png



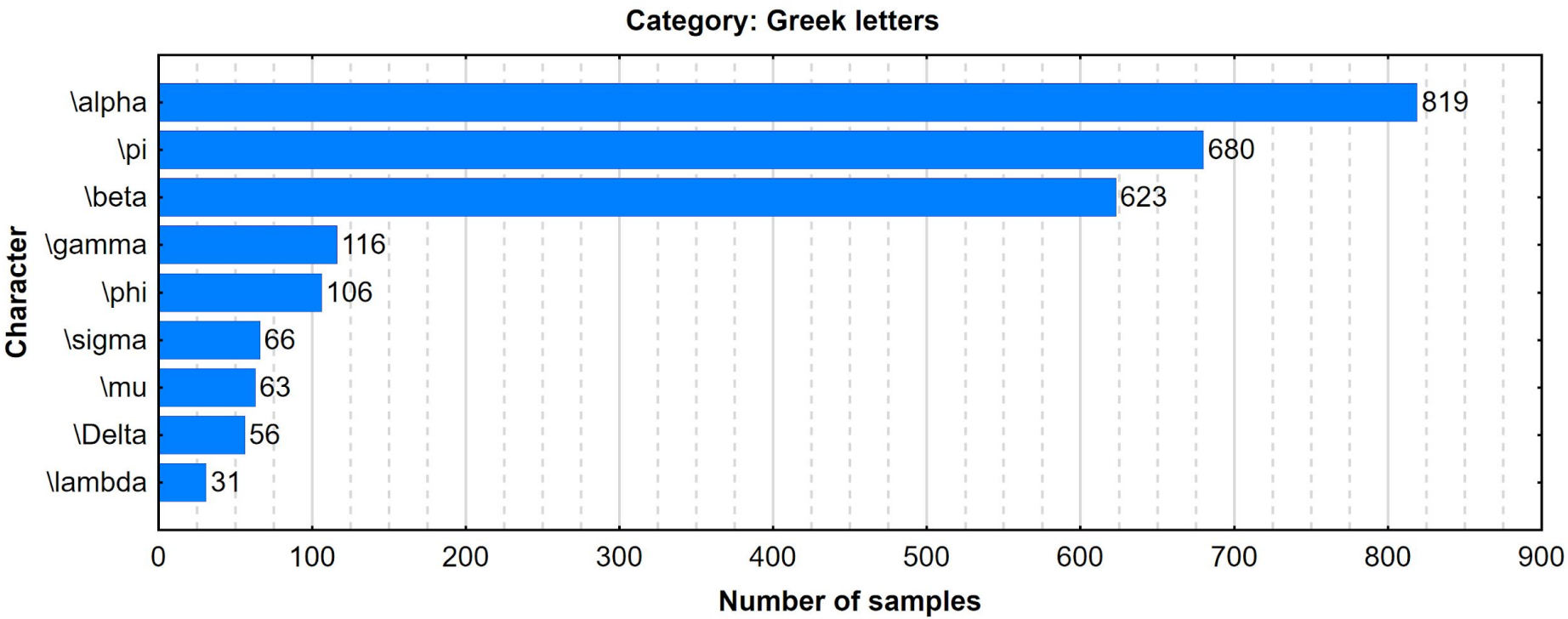
This plots shows the distribution of lower case letters in the CROHME dataset

https://raw.githubusercontent.com/ThomasLech/CROHME_extractor/master/histograms/lowercase_letters_distribution.png



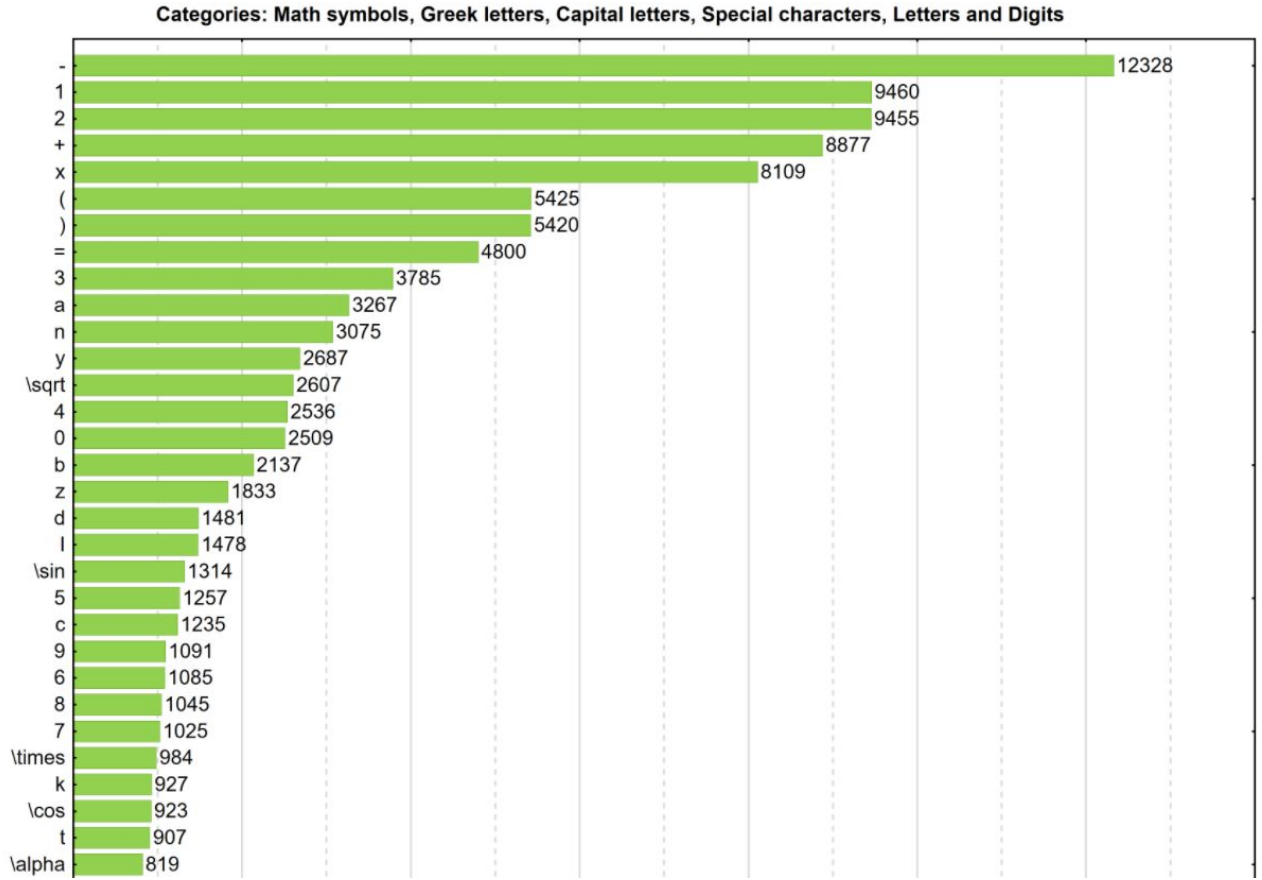
Distribution of capital letters in the CHROME dataset

https://raw.githubusercontent.com/ThomasLech/CROHME_extractor/master/histograms/capital_letters_distribution.png



Distribution of greek letters found in the CROHME dataset

https://raw.githubusercontent.com/ThomasLech/CROHME_extractor/master/histograms/greek_letters_distribution.png



Most of the dataset contains:

- Numbers
- Variable x
- - + x =
- Brackets ()
- √
- Sin, Cos

This plot shows the distribution of symbols in the CROHME data set.
https://cloud.githubusercontent.com/assets/22115481/26694312/413fb646-4707-11e7-943c-b8ecebd0c986.png

HASY data set

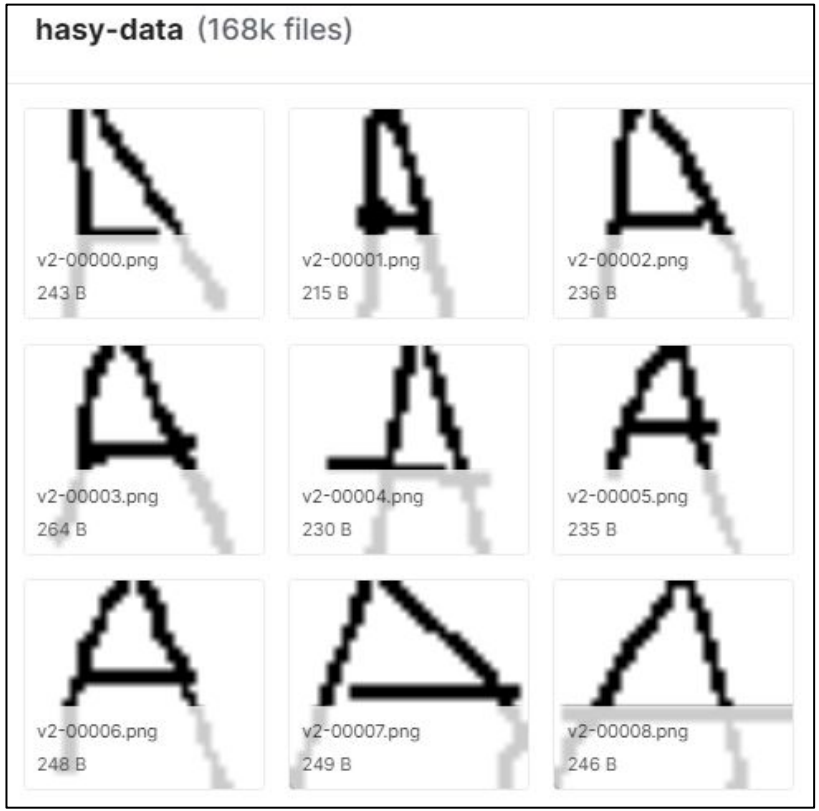
- HASY is derived from the HWRT dataset.
- HWRT is an on-line recognition dataset, meaning it does not contain the handwritten symbols as images, but as point-sequences.
- HWRT contains strictly more information than HASY.
- The smaller dimension of each recordings bounding box was scaled to be 32 px.
- The image was then centered within the 32 px × 32 px bounding box

Link to a paper on the HASY data set [paper]

- 32px x 32px images
- 369 symbol classes
- over 150,000 instances of handwritten symbols.

“A weakness of HASYv2 is the amount of available data per class.
For some classes, there are only 51 elements in the test set”

MNIST has grayscale images while HASY has black and white images.



Sample of HASY data found here
https://www.kaggle.com/kerneler/starter-hasyv2-df54c04f-b?

There are 168233 items stored as png files.

Figure 1: 100 recordings of the HASYv2 data set.

The ten classes with most recordings are:

$$\int, \sum, \infty, \alpha, \xi, =, \partial, \mathbb{R}, \in, \square$$

Same algorithms applied to MNIST and HASY datasets.

Classifier	Test Accuracy		
	MNIST	HASY	min – max
TF-CNN	99.20 %	81.0 %	80.6 % – 81.5 %
Random Forest	96.41 %	62.4 %	62.1 % – 62.8 %
MLP (1 Layer)	89.09 %	62.2 %	61.7 % – 62.9 %
LDA	86.42 %	46.8 %	46.3 % – 47.7 %
QDA	55.61 %	25.4 %	24.9 % – 26.2 %
Decision Tree	65.40 %	11.0 %	10.4 % – 11.6 %
Naive Bayes	56.15 %	8.3 %	7.9 % – 8.7 %
AdaBoost	73.67 %	3.3 %	2.1 % – 3.9 %

The following observations are noteworthy:

- All algorithms achieve much higher accuracy on MNIST than on HASYv2.
- While a single Decision Tree performs much better on MNIST than QDA, it is exactly the other way around for HASY. One possible explanation is that MNIST has grayscale images while HASY has black and whiteimages.

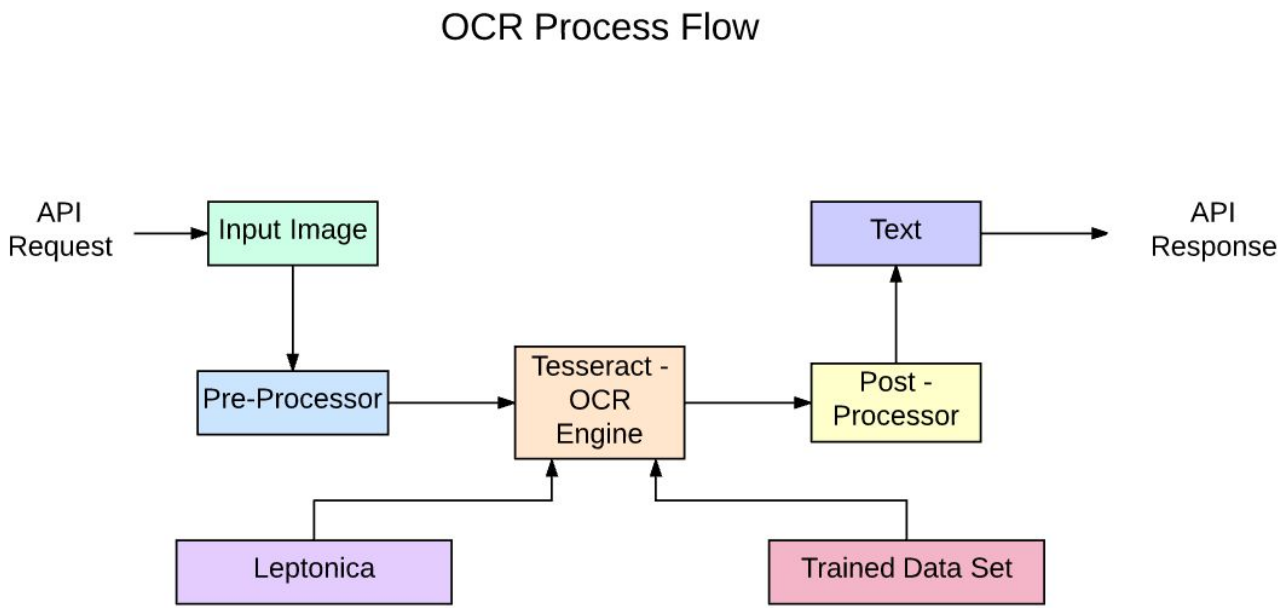
LaTeX	Rendered	Total	Confused with	
\mid	|	34	|	|
\triangle	△	32	\Delta	Δ
\mathds{1}	𝟙	32	\mathbb{1}	𝟙
\checked	✓	28	\checkmark	✓
\shortrightarrow	→	28	\rightarrow	→
\Longrightarrow	⟹	27	\Rightarrow	⇒
\backslash	\	26	\setminus	\
\O	Ø	24	\emptyset	∅
\with	&	21	\&	&
\diameter	⌀	20	\emptyset	∅
\triangledown	▽	20	\nabla	∇
\longmapsto	⟼	19	\mapsto	↦
\dotsc	…	15	\dots	…
\fullmoon	○	15	\circ	∘
\varpropto	∝	14	\propto	∝
\mathsection	§	13	\S	§
\vartriangle	∧	12	\Delta	Δ
O	*O*	9	\circ	∘
o	*o*	7	\circ	∘
c	*c*	7	\subset	⊂
v	*v*	7	\vee	∨
x	*x*	7	\times	×
\mathbb{Z}	ℤ	7	\mathds{Z}	ℤ
T	*T*	6	\top	⊤
V	*V*	6	\vee	∨
g	*g*	6	9	9
l	*l*	6	|	|
s	*s*	6	\mathcal{S}	𝒮
z	*z*	6	\mathcal{Z}	𝒵
\mathbb{R}	ℝ	6	\mathds{R}	ℝ
\mathbb{Q}	ℚ	6	\mathds{Q}	ℚ
\mathbb{N}	ℕ	6	\mathds{N}	ℕ

Table IV: 32 classes which were not a single time classified correctly by the best CNN.

Tesseract OCR:

- Tesseract is an open source text recognition (OCR) Engine.
- It can be used directly, or using an API.
- It supports multiple languages
- Web Page: https://nanonets.com/blog/ocr-with-tesseract/
- Installation with Python: `pip install pytesseract`

Tesseract OCR Process Flow



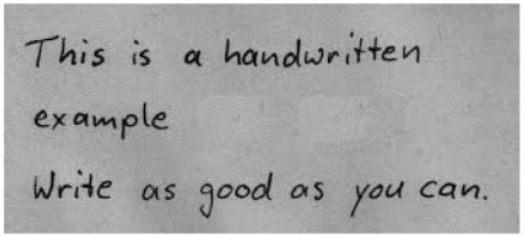
PyTesseract

This is simple python optical character recognition.

- Uses PyTesseract Libary
- Webpage: https://stackabuse.com/pytesseract-simple-python-optical-character-recognition/
- Webpage contains python code that can be implemented.
- More so focused on Strings than math, but the concept is still there and could be modified.
- Webpage uses flask to deploy there code.

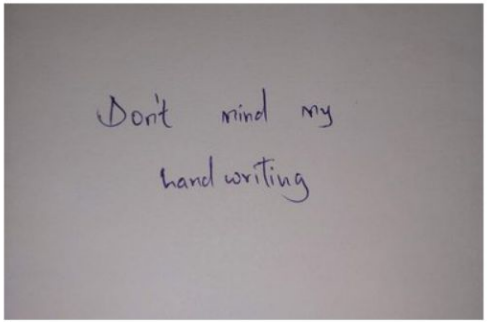
Example Outputs:

Result:



The extracted text from the image above is: **This is a handwritten example Write as geoal as you can.**

Result:



The extracted text from the image above is: **ad oviling**

The Difference between the two outputs is really quite something, showing that handwriting does in fact play a crucial role, we could try to improve these by training our model on the datasets we discussed earlier.

Example Outputs:

Result:

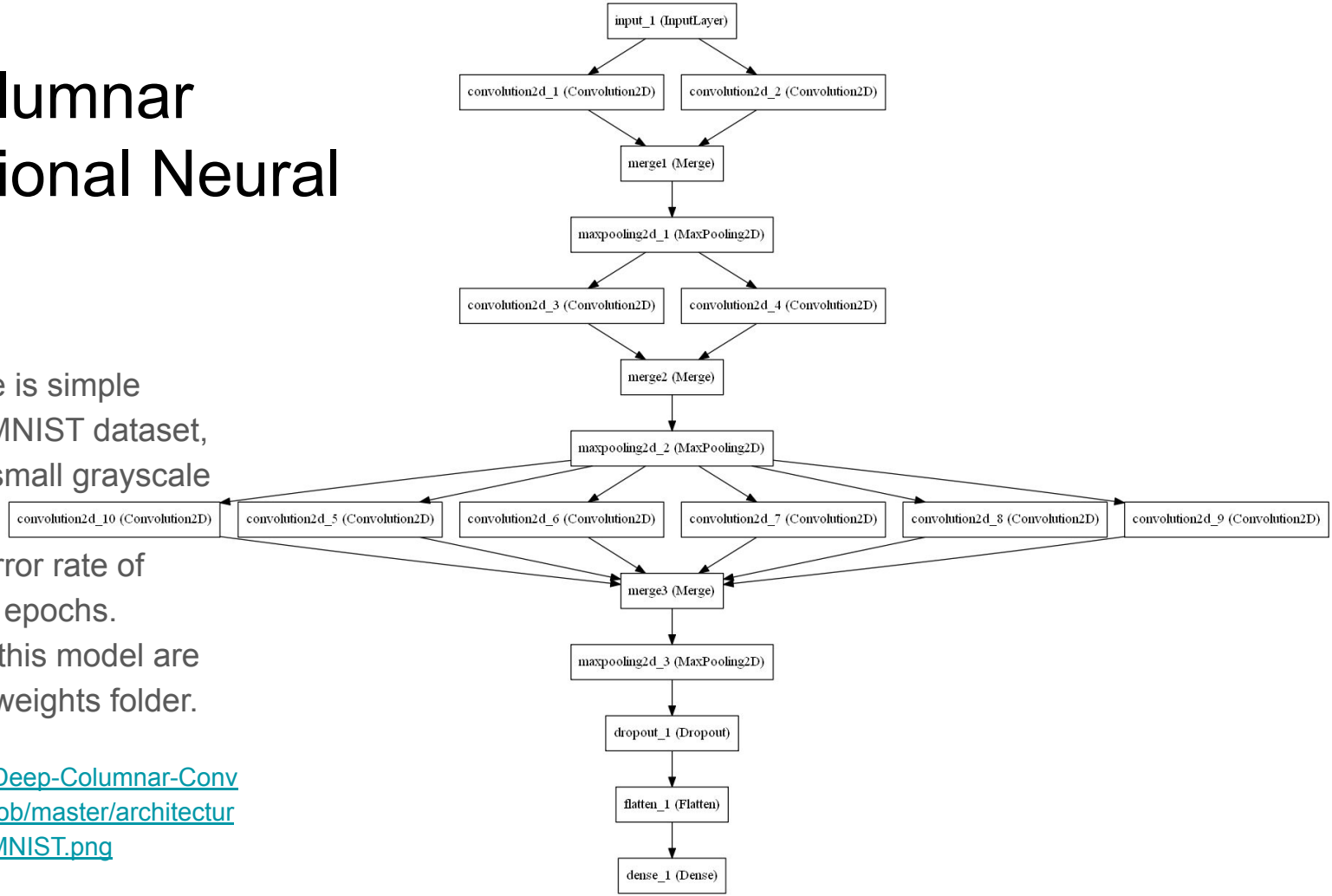
The extracted text from the image above is: **CETTE ALARME EST LOCALE SEULEMENT, EN CAS DE FEU SIGNALER 911**

Included this image to show how it did in fact extract a number being 911, however this is a lot different to a handwritten image. The difference being though that we would have trained our model on the previous datasets.

Deep Columnar Convolutional Neural Network

- This architecture is simple enough for the MNIST dataset, which contains small grayscale images.
- It achieves an error rate of 0.23% after 500 epochs.
- The weights for this model are available in the weights folder.

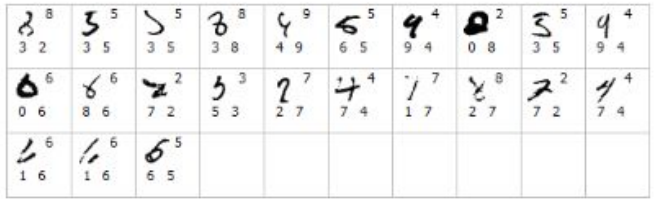
https://github.com/titu1994/Deep-Columnar-Convolutional-Neural-Network/blob/master/architectures/DCCNN%20MNIST.png



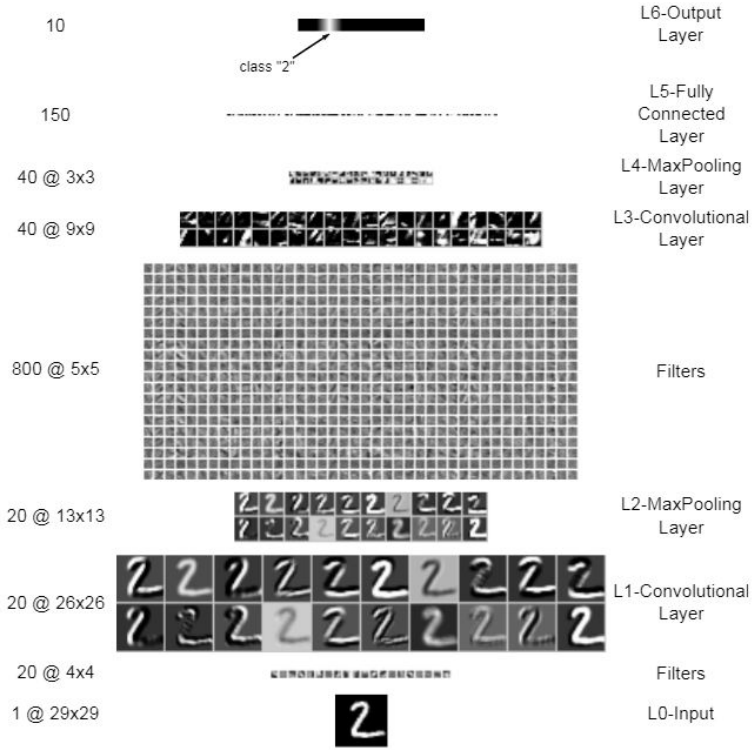
How does it work?

[notebook]

Handwritten digits from the training set (top row)and their distorted versions after each epoch (second to fifth row)



The 23 errors of the MCDNN, with correct label (up right) and first and secondbest predictions (down left and right)



DNN architecture for MNIST. Output layer not drawn to scale;weights of fully connected layers not displayed.

Results

Table 4. Average error rates of MCDNN for all experiments, plus results from the literature. * case insensitive

Data (task)	MCDNN error [%]	Published results Error[%] and paper	
all (62)	**11.63**		
digits (10)	**0.77**	3.71 [12]	1.88 [23]
letters (52)	**21.01**	30.91[16]	
letters* (26)	**7.37**	13.00 [4]	13.66[16]
merged (37)	**7.99**		
uppercase (26)	**1.83**	10.00 [4]	6.44 [9]
lowercase (26)	**7.47**	16.00 [4]	13.27 [16]

There seems to be a larger error when classifying letters compared to digits