

A Calculator for Handwritten Math Problems

By Brighton Mica and Nathan Swedlund

Introduction

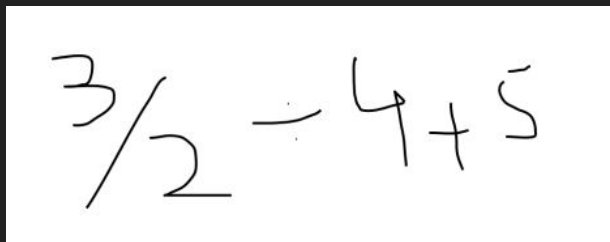
Problem Statement

Research Question

What methods and techniques can we use to identify and accurately classify handwritten numerals and mathematical notation while retaining the relationships between them so that we can reduce such input to a numerical calculation?

What We Are Trying To Do

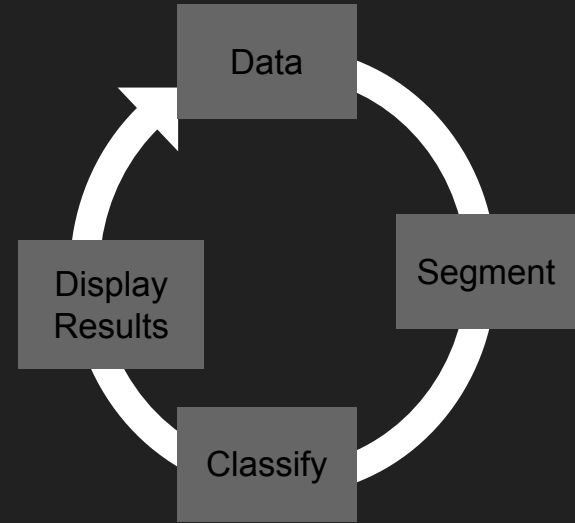
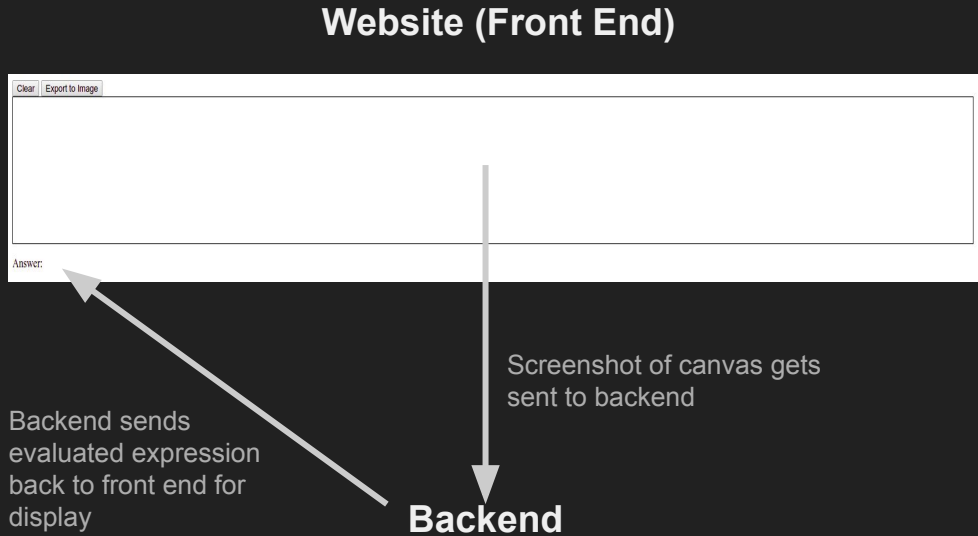
We are trying to make a classifier for handwritten digits and mathematical notation so that we can reduce handwritten math problems to their numerical answer.

A white rectangular box containing a handwritten mathematical expression in black ink. The expression is $\frac{3}{2} - 4 + 5$. The handwriting is casual, with the fraction $\frac{3}{2}$ written as '3' over '2', and the minus and plus signs being simple horizontal strokes.



$$\frac{3}{2} - 4 + 5 = 2.5$$

Application Lifecycle



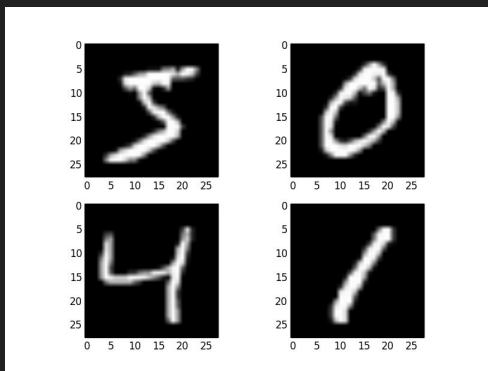
Training Data

Digits and Math Symbols

MNIST Digits

- Images: ~50,000
- Features: 784 (28x28 pixels)
- Digits 0–9

URL: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html



Handwritten Math Symbols

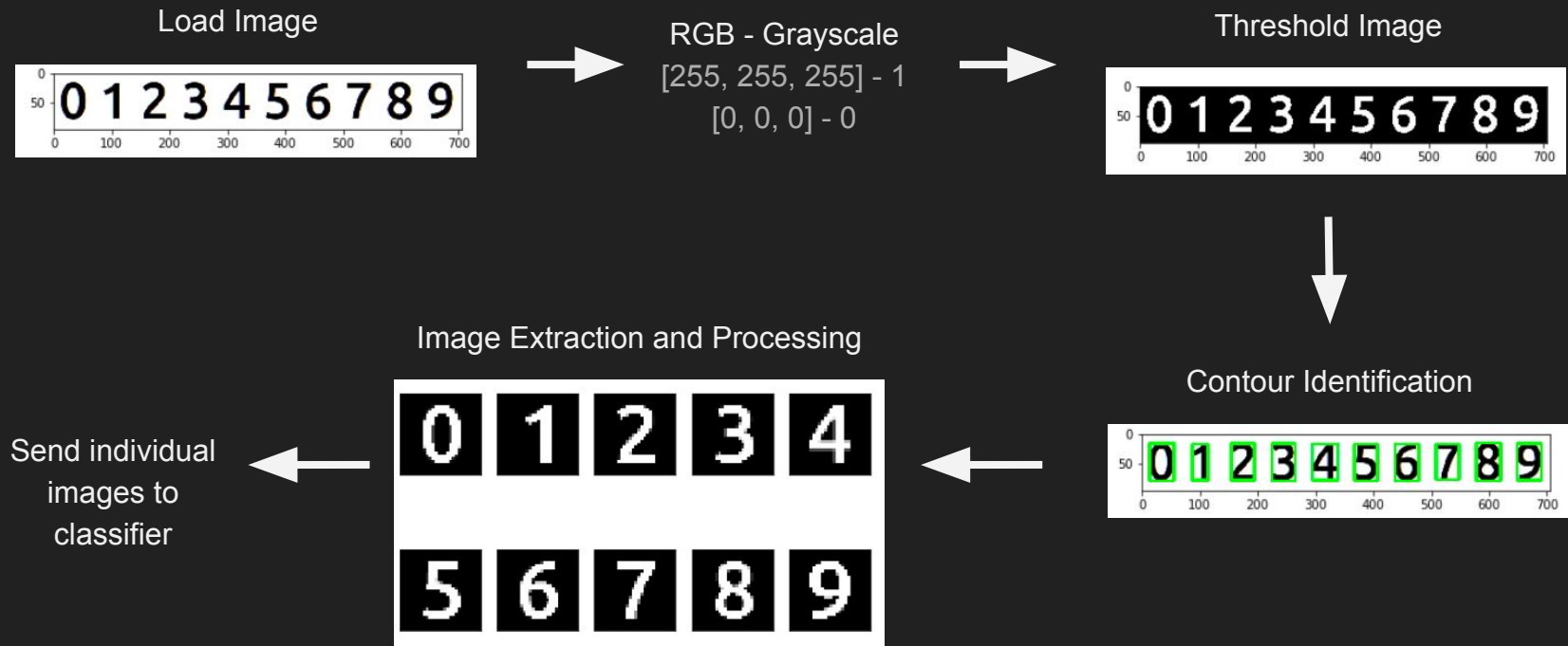
- Images: 100,000+
- Features: 2,025 (45x45 pixels)
- +, -, /,), (, *

URL: <https://www.kaggle.com/xainano/handwritten-mathsymbols>



Segmentation

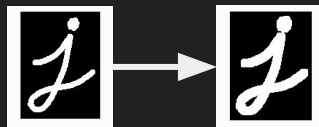
Segmentation - Process



Segmentation - Further Image Processing

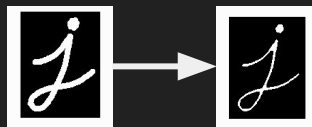
Dilation

- Increasing pen thickness



Erosion

- Decreasing pen thickness

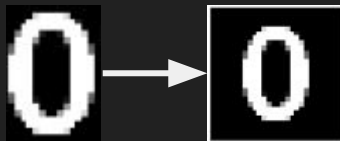


Blur

- Gaussian blur to make images smoother

Centering and Resizing (28x28)

- Adding padding (solid black background)



Classifiers

Training

Classifier Results on Training Data:

- GuassainNB - 77.6%
- Decision Tree - 92.3%
- K-neighbors - 78.4%
- Neural Network - 94.5%

This may seem good, but due to the differences in data, these numbers are much higher than the performance on the actual data from our site.

Results and Response

Results

Accuracy: 35%

$$7 + 2 - 1$$

$$3 + 3 - 1$$

$$1 + 2 + 3 - 4$$

$$/ + 0 + 3 - +$$

$$8(9 - 1)$$

$$864 - 13$$



What Went Wrong?

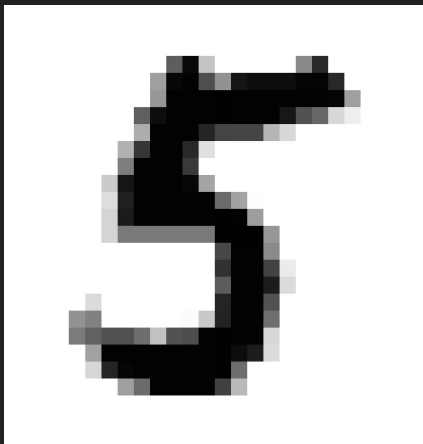
Differences in Training Datasets

MNIST - approximately 3px padding around images (fairly uniform) (28x28)

Math - approximately 0px of padding (not uniform) (45x45)

(There is also an clear difference in stroke size)

This is a “)” not a “1”



What Went Wrong Cont.

Differences in Challenge Data vs Training Datasets

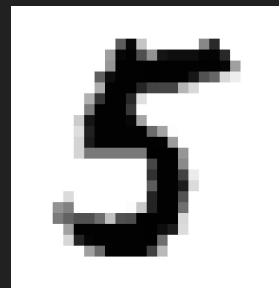
The pen size on our website was perfect for the math dataset, but too thin for the MNIST dataset.

Writing with a mouse is different than writing with one's hand. The structure of the symbols being classified from our website were different than those in our training set.

Website "5"



MNIST "5"



Creating Custom Datasets and Retraining

Custom Data

Rather than trying to get our segmenter to match MNIST and the Math datasets, we built our own; thus, resolving issues regarding data integrity and the differences in datasets.

Using the segmenter, we were able to create a dataset with 1500+ images in under an hour.

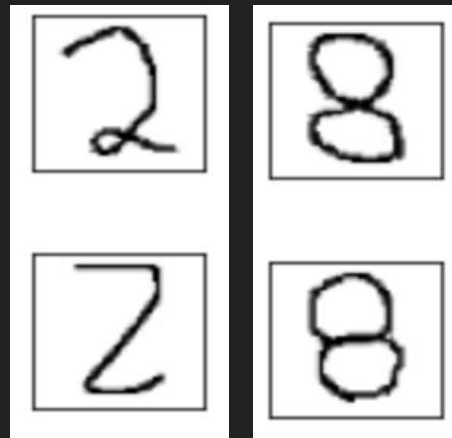


A Note on Handwriting:

Something probably worth mentioning is the fact that in creating this dataset, we only have samples from the way we write these symbol. That said, given the unstable nature of drawing with a mouse, we don't think that that will impact the performance on other people's handwriting.

Even with this, we still tried to mix up the way we wrote just to minimize any potential issues.

(See example on right side)



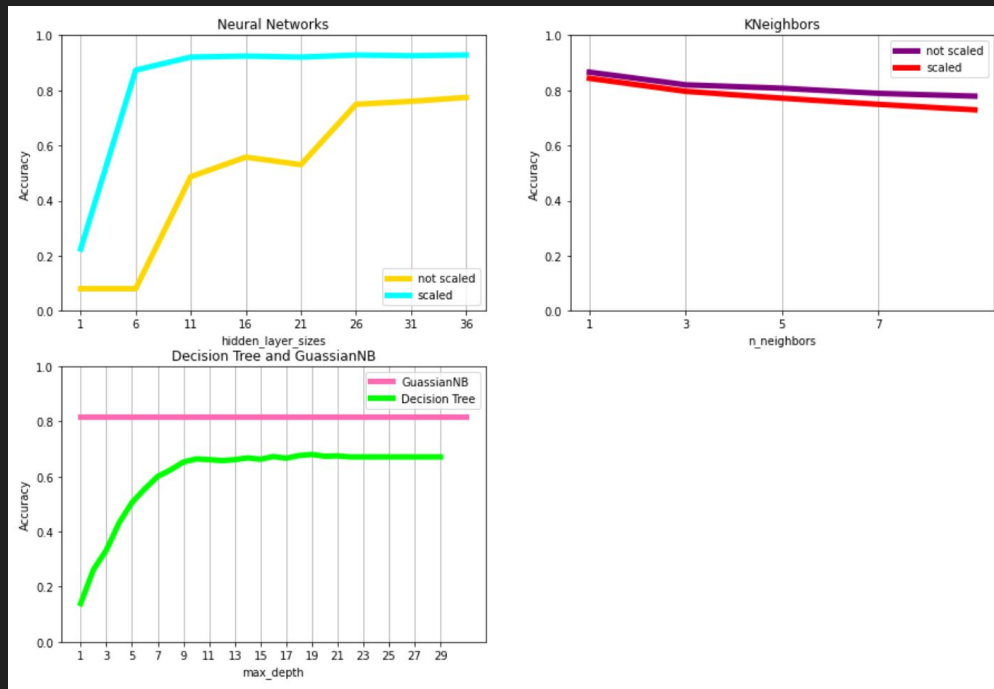
Retraining Classifiers on New Dataset

Classifier Results on New Training Data:

- **GuassianNB** - 80%
- **Decision Tree** - 64%
- **K-neighbors** - 86%
- **Neural Network** - 90%

These classifiers seem worse than the previous iterations of our classifiers, but since they were trained on actual data from the website, their performance is far better than the last batch of classifiers when used on live data.

Even though the neural net had the best performance on the test/train split, we decided to use the K-neighbors classifier on the website due to better live performance.



Performance

As you can see from the images on the right, even when our classifier fails in guessing the character it's still somewhat close. We think much of this could be solved with a larger dataset, but of course it will always have errors.

Overall, we are quite pleased with the performance of the classifier.

Successes

Answer: $9-63 = -54$

Failures

Answer: $(9)13 = \text{Could not parse}$

Answer: $59/0 = \text{Could not parse}$

Answer: $(90)(3) = 270$

Demo

Group/Team Management

Time Spent:

Both group members have spent approximately 35 hours on the project.

Communication:

Group members meet at least 3 times a week over Discord.

Nathan

- Created Server
- Made Preliminary Classifiers
- Made Raw-to-CSV notebook
- Tested Alternate Classification Avenues
- Bug Fixes
- Created data
- Created new classifiers

Brighton

- Image Manipulation Functionality
- Segmentation
- Classifier/Segmenter/Website Integration
- Bug Fixes
- Recreated server
- Created data

Conclusion

Overview

We set out to create a program which evaluates handwritten math expressions; we did just that, but we had to narrow our scope due to our inability to reconcile data issues at an early stage in the project. As a result of creating our own dataset, we created a performant classifier at the cost of the classifier being bias toward our handwriting.

What We Learned

Data integrity is of the utmost importance. Prior to even working on classification, we should have ensured that our training data was uniform and that our challenge data had a structure similar to the training data.

Future Work

We left much to the future due to us practically restarting our project 4 weeks in. Below is a list of features we would like implement

- Create more training data
- Find optimal combination of classifiers
- Dynamic Training (the more the website is used the better the classifier gets)
- Publish and update the look of the website

Sources

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html

Bradski, G., 2000. The OpenCV Library. *Dr. Dobbs's Journal of Software Tools*.

Pedregosa, F. et al., 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct), pp.2825–2830.

Link to Github

<https://github.com/NathanSwedlund/HandwritingCalculator>