πψτορχη

Mathematical Symbol Identification

——A CNN Implementation using pytorch

Proudly Presented by:
Chengxun Wu, Xingjian Gao, Yuxuan Li

TABLE OF CONTENTS

01

Introduction

Introduction: What Inspires Us?

The inspiration of this problem comes from a **mathematics student's daily life**: how to electronically store the hand-written mathematical formulas and expressions into the standardized digital form? Typing on Latex would be clearer while writing it on the paper would be much faster but with less clarity. This trade off inspired the team with this Mathematical symbol identification idea.

From a technological view, insights from LeCun's paper in Multilayer neural networks' performance in digits and LeNet's visualization inspire the team a lot.

Images:
CNN is a good friend 🤗

Supervised Learning!

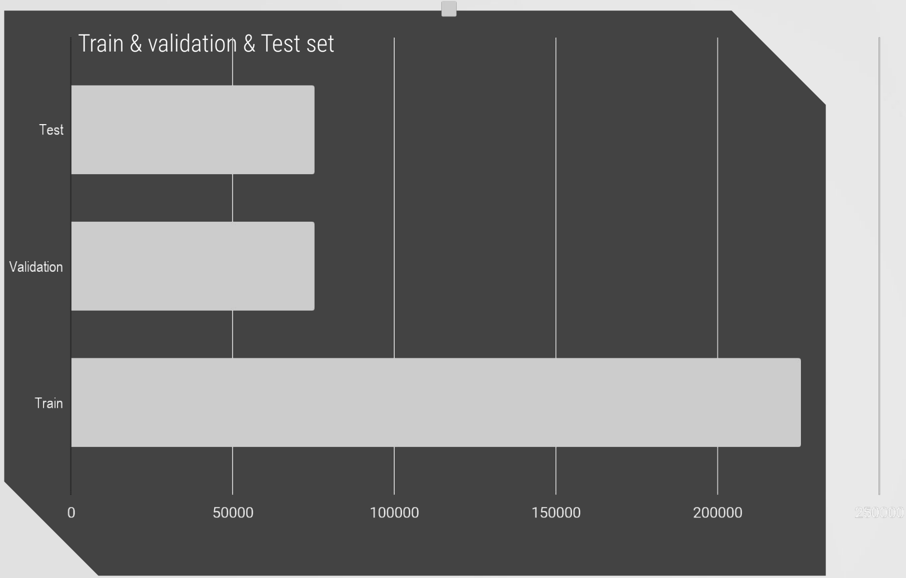
Classification 🤔

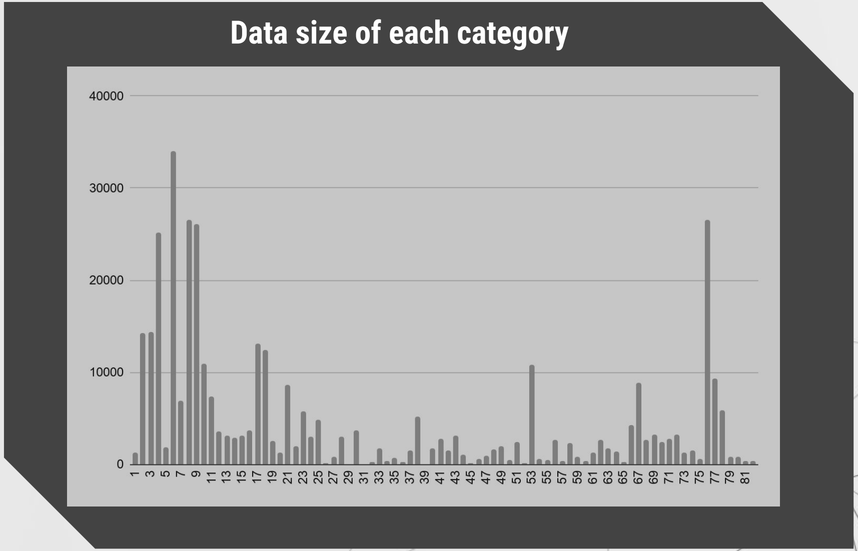
02

Data Pre-processing

What does our dataset look like?
How we process the dataset?

Data Overview





Data Source: kaggle

Data process

Picture → Pixel_based

PIL.Image + Torchvision

Tensor [1,3,45,45]

Compressed due to the limitation of the RAM and GPU

All data are of Black and White

Tensor [1,1,45,45]

new employees last semester

03

Algorithm

--A Convolutional Neural Network Implementation

CNN construction

Evaluation Metrics: Cross Entropy Loss and Accuracy

CNN: 3 Fully connected + 3 Convolutional

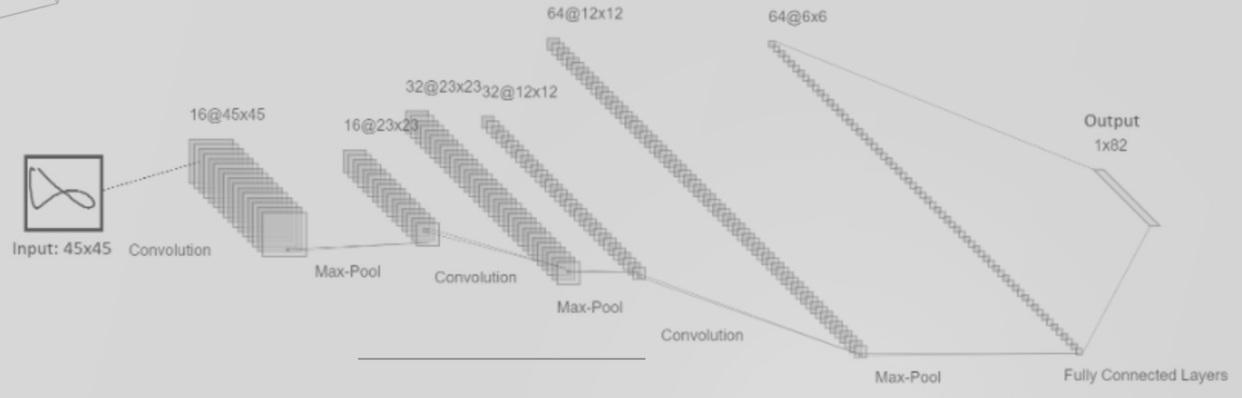


Figure 4: Simplified Visualization of CNN

In practice, we use Pytorch to implement the CNN. And this picture omit details like Batch Normalization, Drop-out mechanism.
A more detailed description of the structure can be found in the final report.
Training Epoch: 25.

Evaluation

$$= -\log\left(\frac{\exp(x[class])}{\sum_j \exp(x[j])}\right) = -x[class] + \log\left(\sum_j \exp(x[j])\right)$$

Cross Entropy Loss

Implemented by Pytorch:
A combination of soft-max function and traditional cross entropy function.
Also, it works well for data with a higher dimension, allowing batch-processing, hence the team decides to apply it directly to the output of the constructed CNN

To give an overall evaluation of the model, the project concentrates both on reducing the training loss as well as improving the accuracy respectively on validation set and test set. Also, to visualize the final report, the team decides to plot the normalized confusion matrix for the classification results on the test data. All these metrics would contribute to the evaluation of the model.

Prediction Accuracy

$$\text{Accuracy} = \frac{\#\,\text{Prediction} = \text{ActualLabel}}{\#\,\text{TotalData}}$$

04

Optimization

How do we optimize the algorithm?
What problems are we going to avoid?

Optimizers -- Based on Gradient Descent

Adams

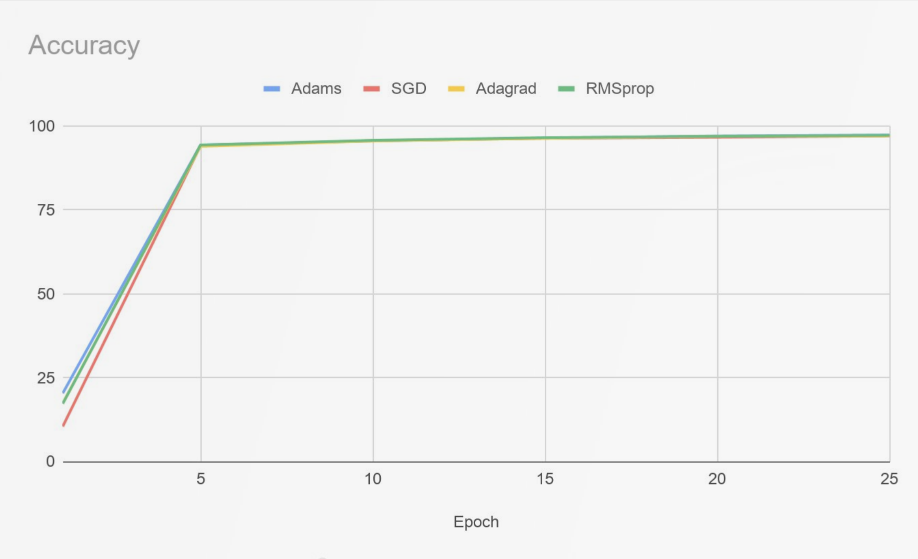
SGD

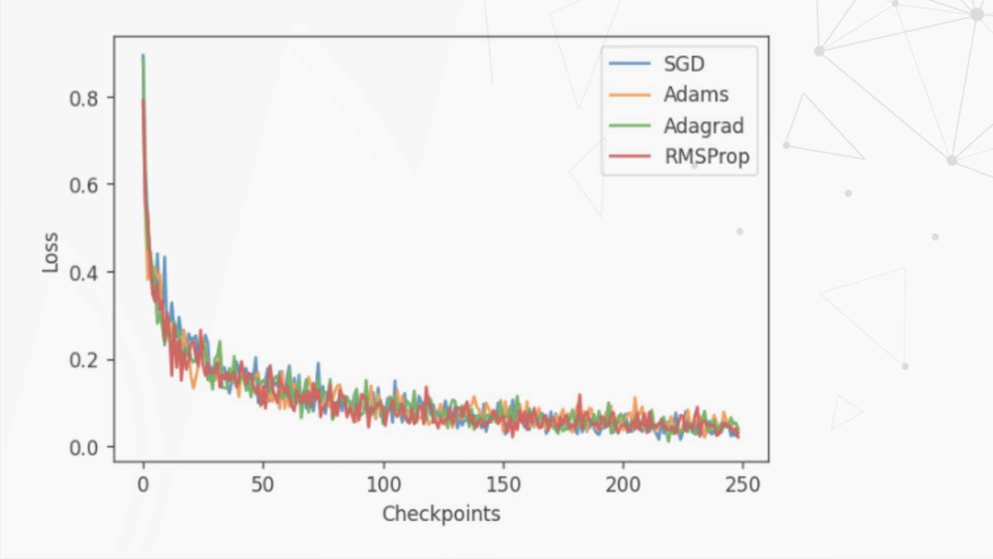
The teams discover that the convergence speed is quite fast for all of the optimizers. In the implementation of the project, the team uses ***torch.optim*** to utilize the optimizers. The fast convergence can also be found in the plot of model losses at different checkpoints for all the optimizer candidates. In the following experiments, the team chooses Adam algorithm with learning rate 0.002 for further analyses.

Adagrad

RMSProp

Optimizers -- Based on Gradient Descent





Comparisons of the different accuracies, tracked at 1, 5, 10, 15, 20, 25 epochs.

Tracking of the model losses for different optimizers.

Optimizers -- Other Optimization Methods

Apart from multiple selections of optimizers, the team also implements other methods to achieve a more efficient training of the constructed CNN.

- To accelerate the training process, the team applied **Batch Normalization** in each convolutional layer (2d Batch Normalization) and between fully connected layers (1d Batch Normalization).
- Moreover, in the initial trials, the team faced the challenges from over-fitting: the model could only recognize pictures from the dataset. While predicting, the model will perform a random guess to the newly created data which has never appeared in the existing dataset. The team then applies the technique of ***Dropout*,** adding drop-out mechanisms to each of the fully connected layers, equipping the model with better generalization performance

05

Results

A presentation of the experimental results.
Quite Exciting!

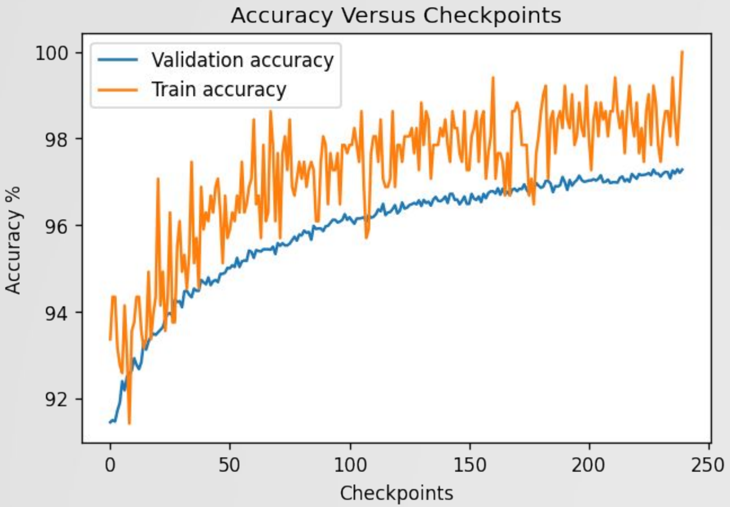
Evaluation Metrics

Numerical Metric — **Test Accuracy**

Visualization Metric — **Confusion Matrix**

Extra-data Metric — **Extra Handwritings**

Accuracy: Trends, Performance on Test Set



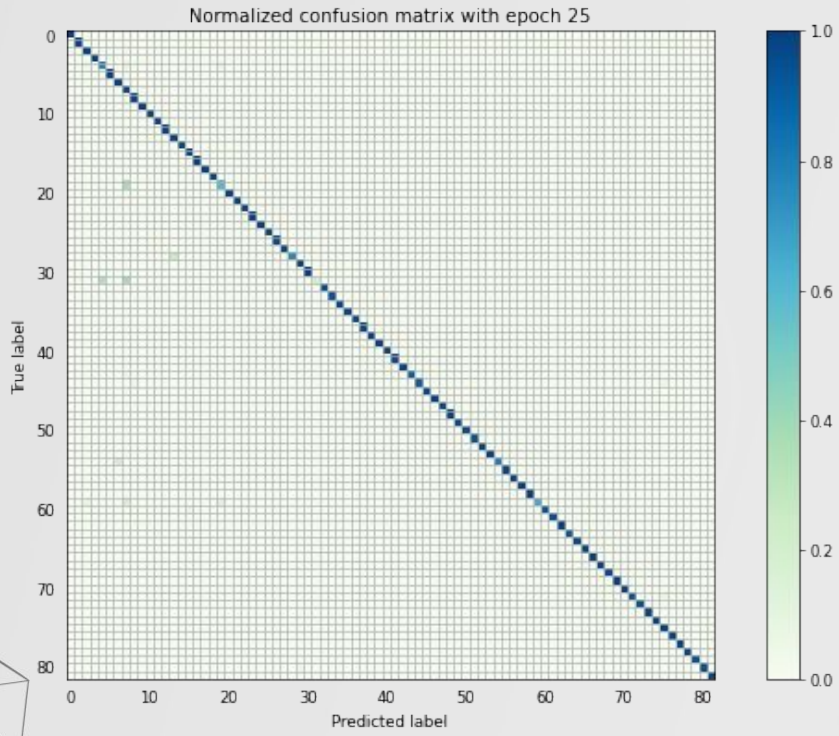
Left figure shows: the plot of accuracy versus checkpoints.

- A **drastic** speed of convergence.
- A **stably growing** trend of accuracy.
- Fluctuation due to using batches; **climax** of accuracy on training batches reaches 100%.
- Overall **outstanding** accuracy.

```
Test Accuracy:  0.9754526948816541
```

Accuracy on Test Data. Quite Impressive!

Confusion Matrix: A Visualization of the Classification Results

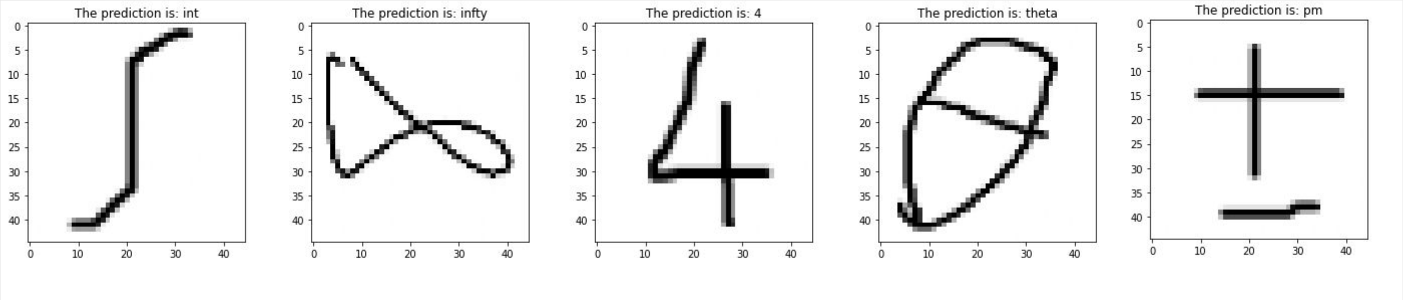


The **confusion matrix** after 25 training epochs.

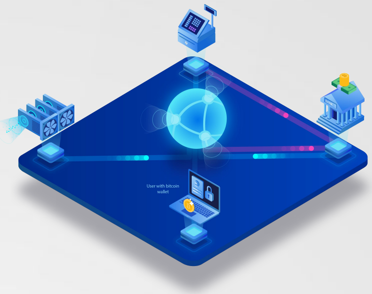
The deep colors on the diagonal demonstrates the high successful rate in model predictions.

Implemented by **sklearn**

Extra-data: Manifestation of Model Generalization



We created extra data to test whether the model is getting **overly familiar** with the dataset we used for training, testing and validation.
Above shows some additional data we write by hands as input to the model.
The model has successfully identified those inputs, which is an **auspicious** sign of the model generalization robustness.



06

Limitations and Future Work

The whole project, though decent in many dimensions, is still not perfect.
How can we improve?

Limitations

Monotone Features

Datas from the datasets all have the similar features:

1) Located at the center of the picture
2) No interference of other figures

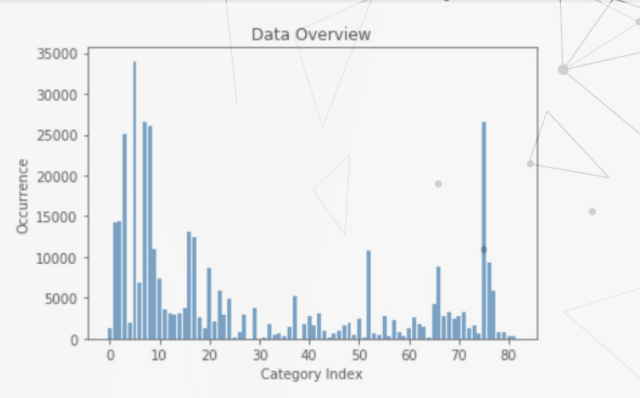
Shape of the Figures

The dataset doesn't contain figures with various thickness

Imbalanced Dataset

The dataset has an uneven distribution of the symbols, some with thousands of samples but some with only less than one hundred samples.
Consequences could be that the overall performance is satisfying, but performance for one specific symbol might be disappointing.

Future works and improvement

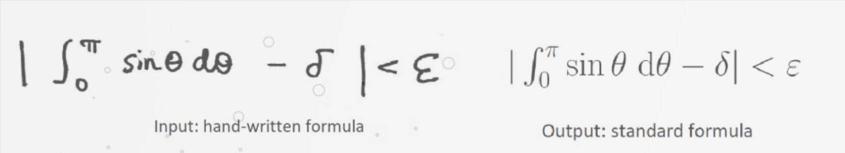


Balance the data set

Create more training data (via handwriting or Generative Adversarial Network)

Increasing variety

Include a a dataset with greater variety for the further training and optimization.



Multiple object detection

Combine the model with the multiple model detection to make the model more practical.

References

[1] Yann LeCun, Leon Bottou, Yoshua Bengio and Patrick Haffner. *Gradient-Based Learning Applied to Document Recognition*. IEEE, 1998.

[2] Xai Nano (at Kaggle). *Handwritten math symbols dataset*. Link: https://www.kaggle.com/xainano/handwrittenmathsymbols.

[3] Alex Nail. *Publication-ready NN-architecture schematics.* Link: http://alexlenail.me/NN-SVG/index.html.

[4] Pytorch Official Documents. *CROSS ENTROPY LOSS.* Link: https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html.

THANKS FOR LISTENING!

If you are interested in cooperating with the future improvement of the project, or if you have any questions, please contact the team members via email:

Chengxun Wu, cw2961@nyu.edu
Yuxuan Li, yl5582@nyu.edu
Xingjian Gao, xg790@nyu.edu