PROJECT ABSTRACT

Handwritten Equation Solver using CNN

A CNN symbol recognizer trained on EMNIST and CROHME competition symbols that segments, parses and solves handwritten math equations through SymPy.

01. Title

Handwritten Equation Solver using CNN

02. Introduction / Background

Reading a handwritten equation from a photo and returning the answer is a compact tour of applied computer vision and symbolic computing. The task decomposes into four hard subproblems: splitting the ink into symbols, recognizing each symbol, recovering the two-dimensional layout — fractions, superscripts, roots — and evaluating the resulting mathematics. Convolutional networks settled the recognition subproblem on datasets like MNIST decades ago, and extended symbol inventories such as EMNIST (Cohen et al., 2017) and the CROHME competition datasets (ICDAR 2011-2019) cover the digits, letters, operators and Greek letters a school-level equation needs. Structural parsing then turns a flat list of classified symbols into an expression tree using recursive baseline-structure analysis, and SymPy performs exact symbolic computation on that tree. This project assembles the full chain: CNN symbol classification, connected-component segmentation, structural parsing, and a SymPy solving backend, with an evaluation notebook that runs a symbol-accuracy procedure on a held-out symbol set and logs solve-rate on a curated equation sheet during the buyer's own build.

03. Problem Statement

Students and tutors routinely photograph handwritten work, but no off-the-shelf pipeline turns an equation photo into a verified solved answer with an inspectable intermediate parse. The need is a complete, explainable system that segments handwritten equations, classifies every symbol with a trained CNN, recovers two-dimensional structure such as fractions and exponents, and solves the expression symbolically — with every reported number produced by the buyer's own evaluation run rather than asserted in advance.

04. Objectives

The objectives of this project are to train a CNN symbol classifier on EMNIST and CROHME competition symbols covering around 100 classes; to build a preprocessing

and connected-component segmentation stage with deskewing and normalization; to implement recursive baseline-structure parsing that recovers fractions, superscripts and roots into an expression tree; to connect the parsed tree to SymPy for exact evaluation, simplification and solving; to provide annotated equation outputs and an equation-sheet batch mode; and to evaluate with a held-out symbol-accuracy procedure plus a solve-rate log via the included notebook.

- Train a CNN symbol classifier on EMNIST and CROHME symbols (around 100 classes).
- Build a binarization, deskewing and connected-component segmentation stage.
- Implement recursive baseline-structure parsing into an expression tree.
- Connect the parsed tree to SymPy for exact symbolic computation.
- Provide annotated equation images and equation-sheet batch solving.
- Evaluate with a held-out symbol-accuracy procedure and solve-rate log during the build.

05. Proposed System / Working

The equation photo is binarized and deskewed with OpenCV, then split into connected components, each normalized to the classifier's input size. The CNN assigns every component a symbol class with a confidence score. Recursive baseline-structure parsing analyzes spatial relationships — what sits above a fraction bar, what is raised as an exponent, what nests inside a root — and assembles the symbols into an expression tree. The tree is converted to a SymPy expression and evaluated, simplified or solved symbolically. The demo app lets the buyer upload equation photos and step through segmentation, classification and parsing. The evaluation notebook runs the held-out symbol-accuracy procedure, builds a confusion analysis naming the confused symbol pairs, and logs solve-rate over a curated equation sheet.

- OpenCV binarization, deskewing and connected-component segmentation.
- CNN symbol classification with per-symbol confidence.
- Recursive baseline-structure parsing into an expression tree.
- SymPy exact evaluation, simplification and symbolic solving.
- Annotated outputs and equation-sheet batch mode.
- Notebook evaluation: symbol accuracy, confusion analysis, solve-rate log.

06. Technologies / Components Used

The system uses Python 3.10 and PyTorch for CNN training and inference. EMNIST (Cohen et al., 2017) supplies digits and letters; the CROHME competition datasets (ICDAR 2011-2019) add math operators and Greek letters. OpenCV handles binarization, deskewing and connected components; SymPy performs the symbolic computation; NumPy and matplotlib support confusion analysis and annotated outputs; and a Jupyter notebook performs the evaluation.

- Python 3.10, PyTorch (CNN training and inference).
- EMNIST dataset (Cohen et al. 2017, digits and letters).
- CROHME competition symbols (ICDAR 2011-2019, math operators).
- OpenCV (binarization, deskewing, connected components).
- SymPy (symbolic evaluation and solving).
- Jupyter notebook (evaluation with symbol-accuracy procedure).
- NumPy, matplotlib (confusion analysis, annotated outputs).

07. Key Features

Key features include a CNN symbol classifier trained on EMNIST plus CROHME math symbols, connected-component segmentation with deskewing and size normalization, a two-dimensional structural parser recovering superscripts, fractions and roots, a SymPy backend for exact symbolic solving, annotated equation images showing segments with labels and confidence, equation-sheet batch solving, an evaluation notebook with held-out symbol accuracy and a solve-rate log, and a demo app for uploading equation photos.

- CNN symbol classifier (EMNIST plus CROHME, around 100 classes).
- Segmentation with deskewing and size normalization.
- Structural parser for fractions, superscripts and roots.
- SymPy exact solving backend.
- Annotated equation images with labels and confidence.
- Equation-sheet batch-solve mode.
- Evaluation notebook: symbol accuracy and solve-rate log.
- Demo app for uploading equation photos.

08. Applications / Use Cases

Handwritten equation solving applies to homework-checking assistants, tutoring tools that show the recognized parse before the answer, digitization of handwritten worksheets, and accessibility tooling that reads photographed math aloud. It also demonstrates segmentation, CNN classification, structural parsing and symbolic computing as a complete viva-ready pipeline.

- Homework-checking and tutoring assistants.
- Digitization of handwritten worksheets.
- Accessibility tooling for photographed mathematics.
- Demonstrating segmentation plus CNN classification in a viva.
- Teaching structural parsing and symbolic computation.

09. Results & Scope

The expected outcome is a working equation solver whose quality is measured, not asserted: the included notebook runs a symbol-accuracy procedure on a held-out symbol set and logs solve-rate on a curated equation sheet during the buyer's build, with a confusion analysis naming the confused symbol pairs as viva error-analysis material. The design target is strong symbol accuracy on the held-out set with a high solve-rate on clearly written school-level equations; the actual numbers come from the buyer's training run. Scope covers legible, clearly separated handwriting in the trained symbol inventory; documented limits include cursive and overlapping symbols, poor lighting, and handwriting styles far outside the EMNIST and CROHME mix. This is an educational prototype, not a certified math-assessment tool — results are advisory. Future scope includes matrix and multi-line layout support, on-device inference, and grammar extensions for calculus notation.

- Measured outcome: held-out symbol accuracy and equation-sheet solve-rate computed by the notebook during the buyer's build.
- Confusion analysis naming confused symbol pairs.
- Design target: strong symbol accuracy and high solve-rate on clear school-level equations.
- Scope: legible, separated handwriting within the trained symbol inventory.
- Documented limits: cursive and overlapping symbols, poor lighting, out-of-coverage styles.
- Future scope: matrices, multi-line layouts, on-device inference, calculus notation.

10. Conclusion

CNN symbol recognition plus structural parsing plus SymPy turns an equation photograph into a verified solved answer with an inspectable parse at every stage, and this project delivers that as a complete auditable system: EMNIST and CROHME-trained classifier, segmentation, baseline-structure parser, symbolic solver, demo app, and a notebook that measures symbol accuracy and solve-rate on the buyer's own build. Built to order, it ships with source code, the evaluation notebook, report, presentation, and viva preparation material.

11. References

The project cites the EMNIST dataset paper, the CROHME competition overview, the foundational CNN paper, and the SymPy library paper.

- G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, 'EMNIST: Extending MNIST to Handwritten Letters,' Proc. IJCNN, 2017.
- H. Mouchere, R. Zanibbi, U. Garain, and C. Viard-Gaudin, 'Advancing the State of the Art for Handwritten Math Recognition: The CROHME Competitions, 2011-2014,' Int. J. Document Analysis and Recognition, 2016.

- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, 'Gradient-Based Learning Applied to Document Recognition,' Proc. IEEE, 1998.
- A. Meurer et al., 'SymPy: Symbolic Computing in Python,' PeerJ Computer Science, 2017.