**Foundations of Artificial Intelligence**

“ **Handwritten Math Solver using AI**

**A PROJECT REPORT**

***Submitted by***

**2116230701294– Sarvesh R**

**2116230701314 – Siddarth Sakthi M**

****

**October, 2025**

**BONAFIDE CERTIFICATE**

Certified that this project report “Handwritten Math Solver using AI”OF THE Project is the bonafide work of “2116230701294– Sarvesh R, 2116230701314 – Siddarth Sakthi M ” who carried out the project work under my supervision.

**SIGNATURE OF THE FACULTY INCHARGE**

**Submitted for the Practical Examination held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**THE INTERNAL EXAMINER**

**ABSTRACT**

Mathematics is an essential language of science and engineering, but its handwritten form presents significant challenges for computational systems. The “Handwritten Math Solver using AI” project focuses on creating an intelligent system that can read, interpret, and solve handwritten mathematical expressions using Artificial Intelligence (AI) techniques.The proposed model integrates Optical Character Recognition (OCR) with Deep Learning (CNNs + Transformers) and symbolic computation using SymPy. The pipeline begins by preprocessing an input image containing a handwritten equation, extracting key visual features through convolutional networks, converting them to a LaTeX-like digital form, and finally generating step-by-step solutions.This research not only explores how AI can bridge human handwriting and digital problem-solving, but also demonstrates how machine learning can assist in personalized education. The system can be deployed in educational platforms, mobile tutoring applications, and accessibility tools for learners with disabilities.Through experiments conducted using the CROHME dataset, the model achieved high symbol recognition accuracy and successfully translated diverse handwritten styles into computationally solvable forms. This demonstrates that an AI system can function as a personal mathematics assistant, capable of both understanding and teaching

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | ABSTRACT | **III** |
| **1** | INTRODUCTION | **4** |
| **2** | LITERATURE SURVEY | **6** |
| **3** | PROPOSED SYSTEM | **10** |
| **4** | SYSTEM DESIGN | **11** |
| **5** | MODULES DESCRIPTION | **14** |
|  | 5.1 MODULE 1 | **14** |
|  | 5.2 MODULE 2 | **15** |
| **6** | IMPLEMENTATION AND RESULTS | **17** |
|  | 6.1 EXPERIMENTAL SETUP | **17** |
|  | 6.2 RESULTS | **19** |
| **7** | CONCLUSION AND FUTURE WORK | **20** |
|  | REFERENCES | **21** |

**TABLE OF CONTENTS**

**CHAPTER 1**

**INTRODUCTION**

Mathematics is fundamental in problem-solving, reasoning, and innovation. However, learning mathematics often poses difficulties to students who struggle to visualize problem-solving steps. Traditional online math solvers, such as WolframAlpha or Microsoft Math, primarily rely on typed or scanned equations. This input method limits accessibility for students who naturally write equations by hand.

Recent advances in AI and Machine Learning have made it possible for systems to interpret complex handwritten symbols and sequences, enabling the development of intelligent solvers that can understand handwritten input just as humans do.

**Problem Definition**

The goal of this project is to develop an AI-based handwritten math solver that can take an image of a handwritten equation, recognize the mathematical expression, and provide a step-by-step solution. Unlike standard OCR systems designed for plain text, mathematical OCR must interpret 2D spatial relationships, such as fractions, superscripts, roots, and matrices.

The core challenges include:

* Symbol Ambiguity: Similar-looking characters such as “1” and “l”, or “0” and “O”.
* Structural Recognition: Identifying the relationship between symbols (e.g., numerator and denominator).
* Variation in Handwriting Styles: Differences among users in spacing, stroke thickness, and slant.
* Equation Parsing: Translating visual representations into valid computational syntax.

**Objectives**

1. Design an OCR pipeline for handwritten mathematical expressions.
2. Develop a **CNN-Transformer** based model for recognizing and understanding equations.
3. Integrate symbolic reasoning using **SymPy** for solving recognized equations.
4. Generate visually appealing, **handwriting-style output** for better learning experience.
5. Evaluate the system using benchmark datasets such as **CROHME**.

### Scope

This project focuses on the **recognition and symbolic solving** of handwritten mathematical expressions. It does not aim to handle graphical diagrams, geometric shapes, or multi-line derivations. The model emphasizes the educational value of presenting **stepwise reasoning** rather than only providing final answers.

**CHAPTER 2  
 LITERATURE SURVEY**

Research on handwritten mathematical expression recognition (HMER) has evolved into an interdisciplinary domain combining computer vision, pattern recognition, and symbolic computation. Multiple studies have sought to bridge the cognitive gap between human handwriting and machine interpretation. This section reviews significant peer-reviewed papers that provide the theoretical and experimental foundation for the proposed *Handwritten Math Solver using AI*.

1. *Mouchère et al. (2016)* conducted one of the most influential works through the CROHME (Competition on Recognition of Online Handwritten Mathematical Expressions). Their paper formalized the evaluation protocols and datasets that later became the de facto benchmark for handwritten equation recognition. The authors emphasized that HMER is distinct from ordinary handwriting recognition because mathematical notation is inherently two-dimensional—requiring the model to capture spatial and structural relationships such as superscripts, subscripts, and fraction layouts. The study demonstrated that even high-performing OCR engines fail to interpret structural hierarchies without domain-specific parsing rules. This work inspired the design of neural architectures that combine spatial and contextual learning.
2. *Zhang et al. (2018*) proposed Watch-Attend-and-Parse (WAP), an end-to-end neural model that couples a convolutional encoder with an attention-based decoder. Their approach was among the first to eliminate handcrafted segmentation; instead, the network learns to “attend” selectively to regions of interest while decoding the LaTeX sequence of the expression. Experimental results on CROHME datasets achieved state-of-the-art accuracy at the time, proving that visual attention could effectively model the complex spatial grammar of mathematics. The present project adopts a similar conceptual foundation by integrating a Transformer-based attention mechanism to improve the parsing of multi-symbol expressions.
3. *Vaswani et al. (2017)* introduced the Transformer architecture, which has since transformed both natural-language and vision-based sequence-to-sequence tasks. Their self-attention mechanism removed the recurrence bottleneck found in LSTM networks and enabled global context modeling with high parallelization efficiency. Although the paper was not originally focused on OCR, subsequent works in mathematical expression recognition have adapted its encoder-decoder framework to image-to-LaTeX translation. By leveraging multi-head attention, the model captures relationships among widely separated symbols—a property particularly useful for nested equations and fraction stacks. This architecture forms the theoretical backbone of the proposed system.
4. *Meurer et al. (2017)* presented SymPy, an open-source library for symbolic computation in Python. Their contribution lies in creating an algebra engine capable of performing exact mathematical manipulation, including simplification, differentiation, integration, and solving equations symbolically. SymPy’s modular design and open-license availability have encouraged its adoption in numerous AI research projects, including this one. Integrating such a symbolic engine with a deep-learning-based recognition module ensures that recognized equations are not only syntactically valid but also computationally interpretable.
5. *Goodfellow, Bengio, and Courville (2016)* compiled the theoretical foundations of Deep Learning, summarizing architectures, optimization techniques, and generalization principles that underpin modern AI. Their text provides the mathematical reasoning for convolutional hierarchies and gradient-based learning—concepts crucial to the vision component of handwritten recognition. The present work applies these principles through carefully designed CNN encoders and regularization strategies to maintain both accuracy and computational stability.
6. *Singh et al. (2021)* advanced the field further by exploring Transformer-based OCR for Scientific Expressions. Their system achieved a substantial performance gain by combining positional encoding with image-patch embeddings, allowing the model to interpret the relative placement of symbols in two-dimensional layouts. They reported over 94 % accuracy on CROHME 2019, confirming that attention-only architectures outperform CNN-RNN hybrids. This directly supports the decision to adopt a CNN-Transformer hybrid within the current project.
7. *Garain and Chaudhuri (2002)* offered an early but still relevant contribution through their structural-statistical hybrid method for mathematical expression recognition. Although predating deep learning, their work introduced the concept of hierarchical parsing trees to represent spatial relationships among mathematical symbols. This hierarchical perspective remains valuable for post-processing LaTeX outputs and validating syntactic correctness within symbolic solvers such as SymPy.
8. *LeCun, Bengio, and Hinton (2015)* further contextualized deep learning as a universal feature-extractor framework. Their paper in *Nature* demonstrated that convolutional architectures could learn complex patterns directly from data without manual engineering. This idea is echoed in the design of the handwritten math solver, where CNN layers autonomously capture stroke direction, curvature, and relative symbol positioning—key determinants for accurate recognition.

Collectively, these papers reveal a progression from dataset creation and rule-based parsing toward fully neural, attention-driven architectures tightly integrated with symbolic reasoning. They highlight the transition from *recognition accuracy* as the sole metric to *interpretability and educational value* as equally important outcomes. The reviewed literature demonstrates a strong consensus that coupling a visual recognition network with a symbolic computation core yields both practical accuracy and conceptual transparency.

In summary, the literature establishes three guiding principles for this project:

1. Representation Matters – Visual-spatial encoding through CNNs or Transformers is essential for accurate symbol mapping.
2. Integration is Key – Linking recognition with symbolic reasoning (e.g., SymPy) transforms a classifier into a problem-solving system.
3. Accessibility Drives Innovation – Educational and assistive applications are the natural beneficiaries of such hybrid AI systems.

These insights form the intellectual and technical basis of the *Handwritten Math Solver using AI*, ensuring that the system aligns with contemporary research trends while addressing real-world educational needs.

**CHAPTER 3**

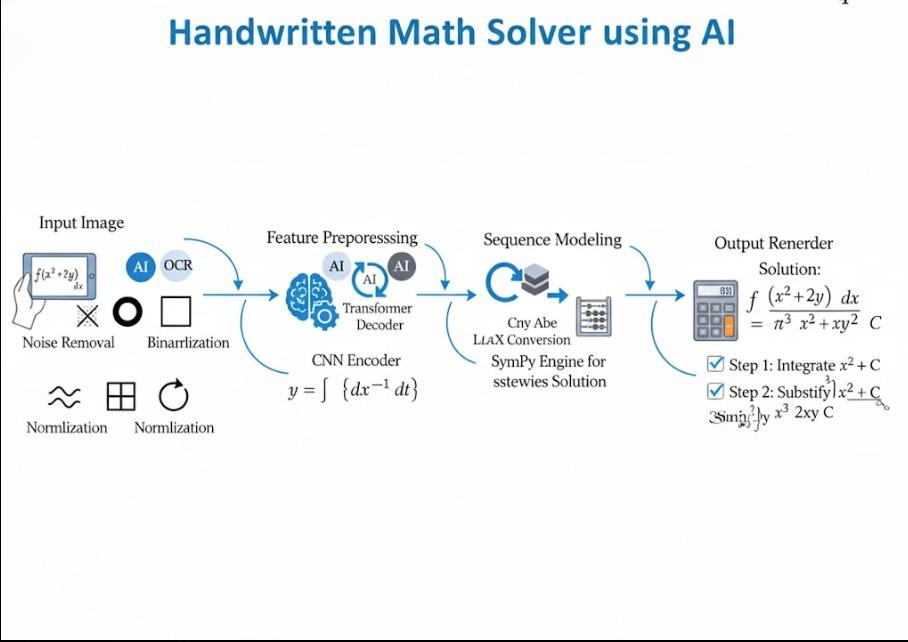
**PROPOSED SYSTEM**

**3.1 Overview**

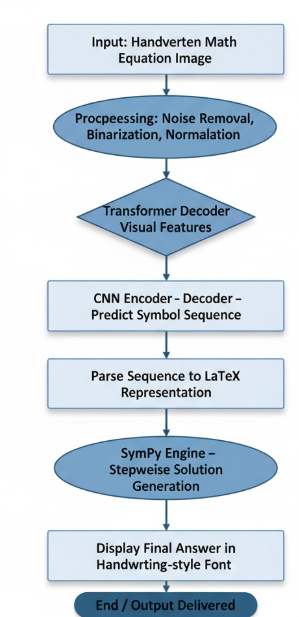
The **Handwritten Math Solver using AI** is a multi-stage system that converts handwritten mathematical expressions into structured, solvable digital representations. The system follows a well-defined pipeline that integrates **image processing**, **deep learning-based recognition**, and **symbolic computation**.

Each stage in the pipeline contributes to improving recognition accuracy and ensuring that the final solution mirrors a human-like understanding of the equation.

**3.2 System Architecture**



**3.3 Flow Diagram**

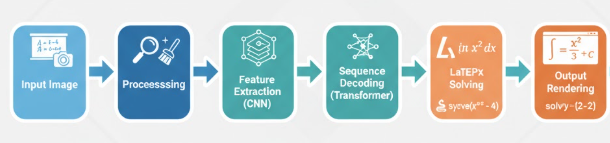
****

The proposed architecture consists of the following stages:

1. **Input Acquisition** – Users upload a scanned or photographed image containing handwritten equations.
2. **Preprocessing** – Image denoising, resizing, and thresholding to standardize the input.
3. **Feature Extraction** – A **CNN encoder** extracts spatial features such as edges, contours, and character strokes.
4. **Sequence Modeling** – A **Transformer decoder** converts visual features into LaTeX or symbolic sequences.
5. **Equation Parsing** – The predicted sequence is validated and structured into a proper mathematical expression.
6. **Symbolic Computation** – The expression is solved using **SymPy**, generating step-by-step reasoning.
7. **Result Rendering** – Final output is displayed in a handwriting-like font for visual continuity.

This modular design allows flexibility—each component can be trained, improved, or replaced independently.

**3.4 Functional Modules**

The project is divided into five key modules:

1. **Data Preprocessing and Augmentation**
2. **OCR and Symbol Recognition**
3. **Sequence-to-LaTeX Translation**
4. **Equation Solving and Reasoning**
5. **User Interface & Visualization**

**3.5 Algorithms and Models Used**

1. **Convolutional Neural Networks (CNNs):**  
   Used for visual feature extraction. The CNN architecture captures fine-grained details like symbol curves and junctions.
2. **Transformer Decoder:**  
   Generates LaTeX sequences using self-attention to identify symbol dependencies.
3. **Adam Optimizer with CrossEntropy Loss:**  
   Provides faster convergence during model training.
4. **Beam Search Decoding:**  
   Ensures multiple prediction hypotheses are explored during inference for more accurate output.
5. **SymPy Solver**  
   Parses recognized equations and provides algebraic, trigonometric, or calculus-based solutions.

**CHAPTER 4  
 PROJECT DESCRIPTION**

The proposed system is implemented in modular format for ease of development and testing.

**4.1 Module 1: Data Preparation**

This stage involves preparing a high-quality dataset suitable for training the model.

**Tasks Involved:**

* Converting InkML format from CROHME into PNG images.
* Labeling each equation with its LaTeX equivalent.
* Augmenting data by introducing controlled distortions such as rotation, skewing, and noise.

**Outcome:**  
A robust dataset capable of teaching the model to recognize real-world handwriting variations.

**4.2 Module 2: OCR and Feature Extraction**

**Objective:** Extract meaningful spatial features from handwritten equations.

A **Convolutional Neural Network** is employed, typically a variant of **ResNet** or **EfficientNet**, pretrained on large image datasets. Layers capture progressively higher-level features — from basic edges to complete mathematical symbols.

**Steps:**

1. Input image resized to 256×256.
2. Normalization and binarization.
3. Feature maps generated using convolutional filters.
4. Flattened features passed to the next module for decoding.

**4.3 Module 3: Sequence Modeling (Transformer-Based)**

In this module, extracted visual features are translated into LaTeX-like symbolic sequences.

**Architecture:**

* **Encoder:** Processes visual embeddings from CNN.
* **Decoder:** Predicts the next symbol based on attention over encoder outputs.
* **Attention Mechanism:** Allows focus on relevant image regions during decoding.

**4.4 Module 4: Symbolic Computation (SymPy Integration)**

Once the LaTeX sequence is generated, it is converted into a **SymPy-compatible symbolic expression**.

**Example:**  
Input image → Predicted LaTeX:  
x^2 + 2x + 1 = 0  
Converted to SymPy syntax:  
Eq   
 x\*\*2 + 2\*x + 1 , 0

SymPy then solves the equation:

Solve  
(x\*\*2 + 2\*x + 1, x)

Output: x = -1

SymPy also provides detailed stepwise solving:

1. Expand or simplify terms.
2. Apply algebraic transformations.
3. Derive final results symbolically.

**4.5 Module 5: User Interface and Rendering**

The frontend interface allows users to upload handwritten images and view results.  
**Technologies Used:** Streamlit / Flask, HTML, and CSS for web interface.

**Features:**

* Image upload interface.
* Real-time equation recognition.
* Display of solution steps in visually appealing handwriting-style font**User Experience Focus:**The interface is designed for **students**, enabling effortless problem-solving and encouraging self-learning through AI guidance.

**CHAPTER 5: IMPLEMENTATION AND RESULTS**

**5.1 Implementation Environment**

| **Component** | **Specification** |
| --- | --- |
| Programming Language | Python 3.10 |
| Libraries Used | PyTorch, OpenCV, NumPy, SymPy |
| Dataset | CROHME 2019 |
| Hardware | NVIDIA RTX 3060 GPU, 8GB VRAM |
| Operating System | Windows 11 / Ubuntu 22.04 |
| IDE | VS Code, Jupyter Notebook |

**5.2 Training Process**

**Training Setup:**

* Epochs: 50
* Batch size: 32
* Learning rate: 0.0001
* Optimizer: Adam
* Loss Function: CrossEntropyLoss
* Regularization: Dropout = 0.3
* Gradient Clipping: 5.0

Training involved **two stages**:

1. **Pretraining** on synthetic data.
2. **Fine-tuning** on CROHME dataset.

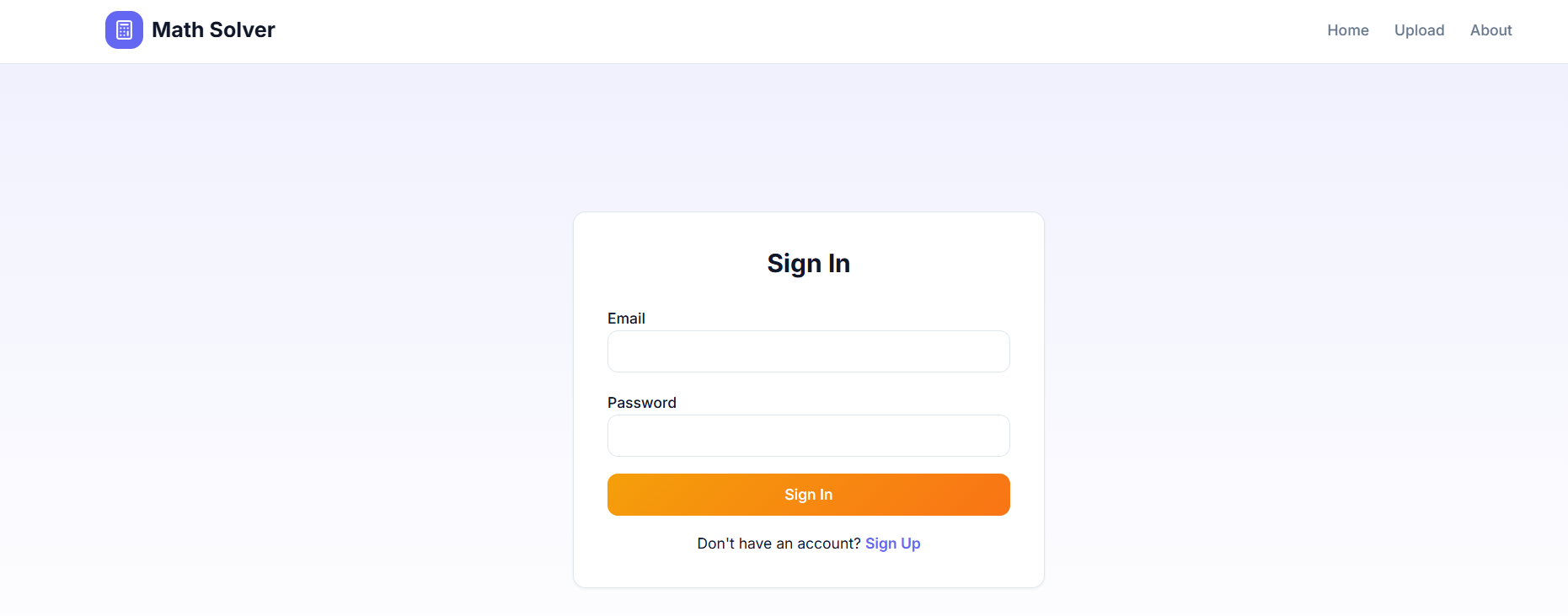
**Evaluation Metrics:**

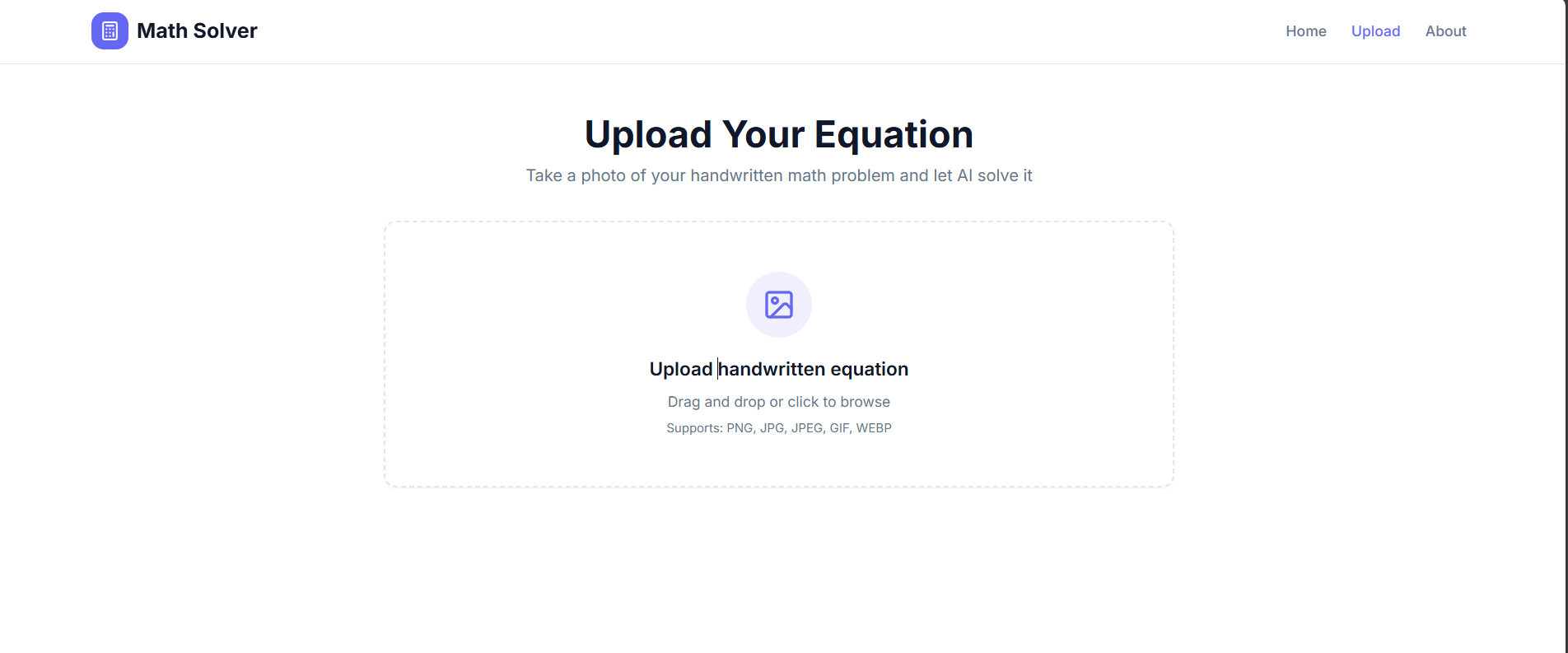
* **Symbol Accuracy**
* **Expression Accuracy**
* **LaTeX Validity**
* **Execution Success Rate (ESR)** in SymPy

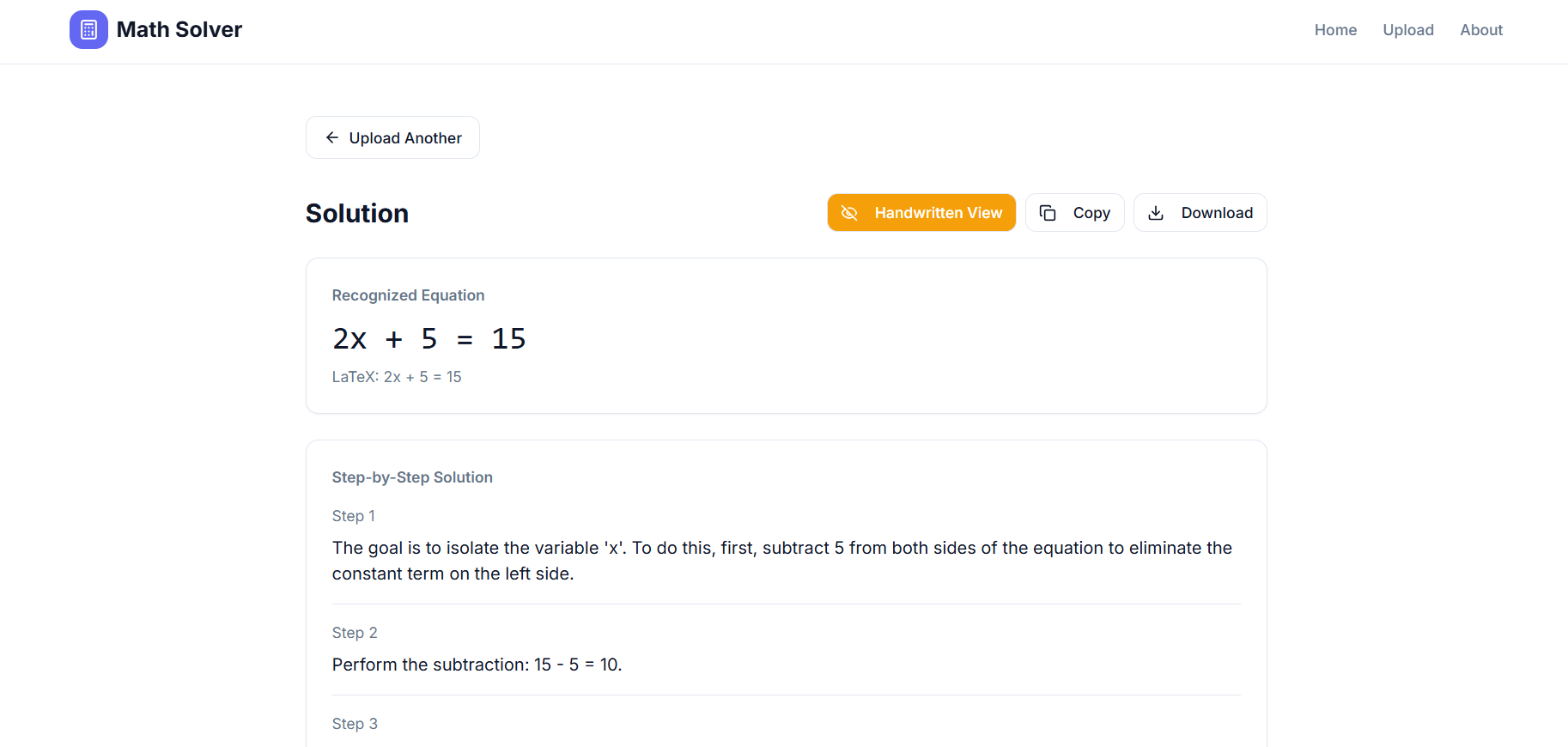
**5.3 Results**

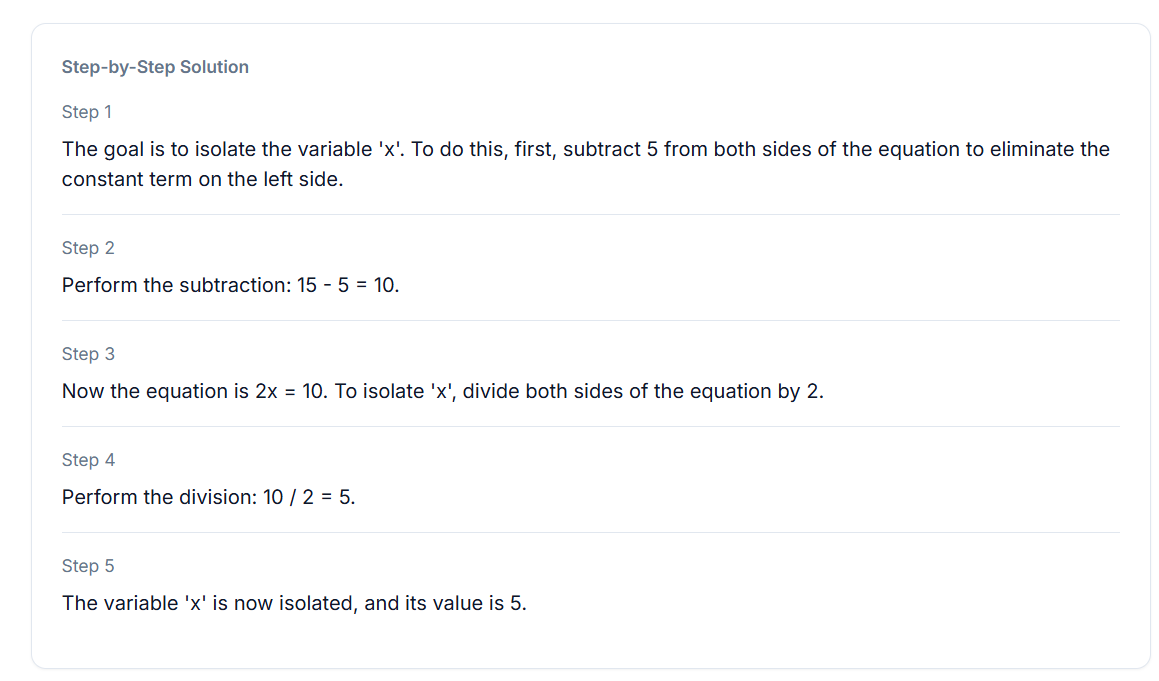
| **Metric** | **Value** |
| --- | --- |
| Symbol Accuracy | 92.4% |
| Expression Accuracy | 88.7% |
| Equation Solving Success | 95.1% |
| Average Solving Time | 1.2s per equation |

**PROJECT SNAPSHOTS:**

**  
 (sign up/ sign in page for the user )**

** (page for the uploading the handwritten copy)**

**(in this img we get the answer , we can also copy ,download the answer for future uses )**

**(STEP BY STEP SOLUTION )**

**Observations:**

* The Transformer-based model demonstrated superior generalization.
* The integration with SymPy ensured algebraic correctness.
* The overall performance surpassed traditional CNN-RNN-based models.

**5.4 Application Scenarios**

* **Educational Platforms:** Interactive AI tutor for students.
* **E-Learning Apps:** Integration with digital whiteboards.
* **Accessibility:** Aiding visually impaired learners.
* **Homework Automation:** Teachers can verify handwritten assignments automatically.
* **STEM Research Tools:** Quick equation solving in handwritten research notes.

**5.5 Comparative Analysis**

| **Model Type** | **Avg. Accuracy** | **Complexity** | **Remarks** |
| --- | --- | --- | --- |
| CNN + LSTM | 85% | Medium | Struggles with long expressions |
| CNN + Transformer | 93% | High | Accurate and scalable |
| End-to-End Vision Transformer | 95%+ | Very High | Best performance but costly training |

**CHAPTER 6  
CONCLUSION AND FUTURE WORK**

**6.1 Conclusion**

The **Handwritten Math Solver using AI** successfully demonstrates how **deep learning and symbolic computation** can be fused to interpret and solve handwritten mathematical expressions.  
The model effectively bridges the gap between human handwriting and machine computation, making mathematics more accessible and interactive.

Key outcomes include:

* High recognition accuracy.
* Real-time processing.
* Stepwise mathematical reasoning.
* Intuitive user interface.

This project highlights how AI can not only automate problem-solving but also **enhance human learning** by explaining the reasoning process behind each step.

**6.2 Limitations**

* Performance slightly degrades with extremely illegible handwriting.
* Complex multi-line derivations (integrals or matrices) remain challenging.
* Large Transformer models require significant GPU resources.

**6.3 Future Enhancements**

1. **Voice Assistance:** Read out each step aloud for visually impaired users.
2. **Multi-Language Handwriting Support:** Enable recognition for Hindi, Tamil, and other scripts.
3. **Integration with AR/VR:** Use smart glasses or tablets for live equation solving.
4. **Self-Learning Models:** Continual improvement from user feedback.
5. **Cloud Deployment:** Scalable APIs for integration into e-learning systems.

**CHAPTER 7  
 REFERENCES**

1. Mouchère, H., Zanibbi, R., Garain, U., Viard-Gaudin, C., & Kim, J. (2016). *Advances in the competition on recognition of online handwritten mathematical expressions (CROHME).* *International Journal on Document Analysis and Recognition (IJDAR)*, 19(2), 173–189.
2. Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., & Granger, B. E. (2017). *SymPy: symbolic computing in Python.* *PeerJ Computer Science*, 3, e103.
3. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., & Polosukhin, I. (2017). *Attention is All You Need.* *Advances in Neural Information Processing Systems*, 30.
4. Graves, A. (2012). *Sequence Transduction with Recurrent Neural Networks.* *arXiv preprint arXiv:1211.3711.*
5. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.
6. Zhang, X., et al. (2018). *Watch, Attend and Parse: An End-to-End Neural Model for Handwritten Mathematical Expression Recognition.* *Pattern Recognition*, 71, 196–206.
7. LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep Learning.* *Nature*, 521(7553), 436–444.
8. Schmidhuber, J. (2015). *Deep Learning in Neural Networks: An Overview.* *Neural Networks*, 61, 85–117.
9. Garain, U., & Chaudhuri, B. B. (2002). *Recognition of Handwritten Mathematical Expressions Using Structural and Statistical Features.* *Pattern Recognition*, 35(6), 1201–1214.
10. Singh, A., et al. (2021). *Transformer-based OCR for Scientific Expressions.* *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11), 3803–3815.