

HANDWRITTEN MATHEMATICAL EXPRESSION RECOGNITION

Project report submitted in partial fulfilment of the requirement for
the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

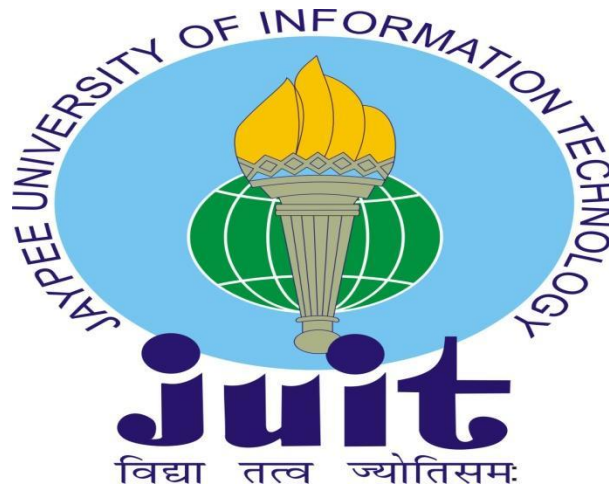
By

Samanvaya Tripathi (191284)

Under the supervision of

Dr. Himanshu Jindal

to



Department of Computer Science & Engineering and Information
Technology

**Jaypee University of Information Technology Waknaghat,
Solan-173234, Himachal Pradesh**

CERTIFICATE

This is to certify that the work which is being presented in the project report titled “**Handwritten Mathematical Expression Recognition**” in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by “Samanvaya Tripathi, 191284” during the period from January 2023 to May 2023 under the supervision of Dr. Himanshu Jindal, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Samanvaya Tripathi
(191284)

The above statement made is correct to the best of my knowledge.

Dr. Himanshu Jindal
Assistant Professor (SG)
Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date:

Type of Document (Tick): PhD Thesis M.Tech Dissertation/ Report B.Tech Project Report Paper

Name: _____ Department: _____ Enrolment No _____

Contact No. _____ E-mail. _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none"> • All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String 		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature

Librarian

.....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

191284_MajorProject_Report.pdf

ORIGINALITY REPORT

16%	8%	11%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Dmytro Zhelezniakov, Viktor Zaytsev, Olga Radyvonenko. "Online Handwritten Mathematical Expression Recognition and Applications: A Survey", IEEE Access, 2021 Publication	2%
2	Submitted to Liverpool John Moores University Student Paper	2%
3	pdfs.semanticscholar.org Internet Source	1%
4	www.researchgate.net Internet Source	1%
5	"Neural Information Processing", Springer Science and Business Media LLC, 2020 Publication	1%
6	"Pattern Recognition and Artificial Intelligence", Springer Science and Business Media LLC, 2020 Publication	1%

ACKNOWLEDGEMENT

Firstly, I express our heartiest thanks and gratefulness to almighty God for his divine blessing makes it possible to complete the project work successfully.

I am really grateful and wish our profound indebtedness to Supervisor **Dr. Himanshu Jindal, Assistant Professor (SG)**, Department of CSE/IT Jaypee University of Information Technology, Wakhnaghat. Deep Knowledge & keen interest of our supervisor in the field of “**Convolutional Neural Networks**” to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to **Dr. Himanshu Jindal**, Department of CSE/IT, for his kind help to finish my project.

I would also generously welcome each one of those individuals who has helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I also want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking. Finally, I must acknowledge with due respect the constant support and patience of our parents.

Samanvaya Tripathi
(191284)

TABLE OF CONTENT

S. No	Title	Page No.
1	Certificate	i
2	Plagiarism Certificate	ii
2	Acknowledgement	iv
3	List of figures	vi
4	List of tables	viii
5	Abstract	ix
6	Chapter - 1 Introduction	1
7	Chapter - 2 Literature survey	8
8	Chapter - 3 System Design & Development	15
9	Chapter - 4 Experiments & Results Analysis	44
10	Chapter - 5 Conclusions	57
11	References	64

LIST OF FIGURES

S. No.	Figure No.	Description	Page no.
1	1.1	Handwritten Mathematical Expressions	3
2	1.2	CNN Architecture	6
3	3.1	Convolutional Layers in CNN	16
4	3.2	Pooling Layers in CNN	17
5	3.3	Role of Activation Function	17
6	3.4	Fully Connected Layers	18
7	3.5	Handwritten Symbol Recognition	22
8	3.6	Structural Analysis of the Digits	24
9	3.7	Lenet-5 Model Architecture	26
10	3.7.1	Input Image	28
11	3.7.2	First Convolutional Operation	28
12	3.7.3	First Subsampling (Pooling) Operation	30
13	3.7.4	Fully Connected Layers	31
14	3.7.5	Working Principle of SoftMax Function	32
15	3.7.6	Prediction Process of Lenet-5 Model	35
16	3.8	YOLOv3 Model Architecture	38
17	3.8.1	YOLOv3 Workflow	41
18	3.8.2	YOLOv3 Output Layer	42

19	4.1	Symbol Loss v/s Expression Loss	50
20	4.2	Symbol Accuracy v/s Expression Accuracy	50
21	4.3	Training Loss v/s Testing Loss	51
22	4.4	Training Accuracy v/s Testing Accuracy	52
23	4.5	Classification Report	53
24	4.6	Digit and Symbol Detection using YOLOv3	55
25	4.7	SoftMax Function Formula	56
26	4.5	Predicted Probabilities of Classes 0-9	56
27	4.6.1	Image Input (1)	57
28	4.6.2	Result (1)	57
29	4.6.3	Image Input (2)	58
30	4.6.4	Result (2)	58
31	4.6.5	Image Input (3)	59
32	4.6.6	Result (3)	59

LIST OF TABLES

S.No.	Table No.	Title	Page no.
1	1	Literature Survey	9
2	2	Performance Analysis of Lenet-5	49

ABSTRACT

Handwritten mathematical expressions are a significant part of many research fields, consisting of engineering, education, and science. The prevalent availability of powerful computational touch-screen appliances, like the modern emergence of deep neural networks as high-quality sequence recognition models, result in the widespread adoption of online recognition of handwritten mathematical expressions. A deeper study and improvement of such technologies is necessary to address the current challenges posed by the extensive usage of distance learning, and remote work due to the world pandemic.

Over the past decade, significant advances in sequence recognition and computer vision models based on deep neural networks (DNN), and the ubiquitous expansion of touch and pen-enabled phones and tablets have led to an increase in interest in handwritten document processing. Handwriting is a natural part of everyday human interaction. These days, in addition to widespread smartphones and tablets, new types of devices such as interactive panels, digital pens and smart writing surfaces have become widely adopted in offices and educational institutions, opening up new opportunities for technologies for recognizing specific handwritten content such as mathematics, diagrams, charts, tables, sketches, etc.

Chapter 1 : INTRODUCTION

1.1 Introduction

The inclination towards handwritten text/symbol processing technologies has increased due to notable progress in CNN-based computer vision models and the generation of pen-enabled and touch-screen devices like smartphones and tablets over the past few years. Everyday human social lives involve handwriting as a significant aspect. Technology is increasingly integrated into various educational institutions, offices, and workplaces through devices such as tablets, mobile phones, digital pens, interactive panels, and smart writing screens/surfaces. This allows for the conversion of handwritten documents, including mathematical expressions, figures, sketches, tables, diagrams, charts, and more, with new opportunities for recognition.

Notable progress in CNN-based computer vision models and the generation of pen-enabled and touch-screen devices like smartphones and tablets over the past few years has increased the inclination towards handwritten text/symbol processing technologies. Handwriting is a significant aspect of everyday human social lives. Various educational institutions, offices, and workplaces are increasingly integrating technology through devices like tablets, mobile phones, digital pens, interactive panels, and smart writing screens/surfaces. New opportunities for recognition are provided by this, which enables the conversion of handwritten documents, such as mathematical expressions, figures, sketches, tables, diagrams, charts, and more.

In many fields, such as engineering, research, finance, and education, HMEs play a critical role. Math Expressions differ from textual representations due to the presence of a 2D structure and a large codebook (over 1,500 symbols), where

characters often resemble each other in HMEs.

Users frequently prefer handwriting input over slow keyboard and mouse input while using ME. Despite recent encouraging advancements, recognition of handwritten mathematical expressions still often results in errors. Unhappy users can result from these errors. Both the system and user experience (UX) are significantly enhanced by a robust user interface (UI), whereas a weak UI may compromise UX even if HME identification accuracy is nearly perfect. To enable the user to quickly correct errors, it is recommended to combine a user interface (UI) with a recognition system for effective HME input.

Both online and offline perspectives allow for the observation of HME recognition. Online recognition uses a dynamic input representation, taking into account pen/finger movement traces, while offline recognition considers a static representation of a picture.

1.2 Problem Statement

Various professions like teaching, engineering, and science heavily rely on handwritten mathematics. There has been an increasing interest in using deep neural network (DNN) models for sequence recognition to recognize handwritten mathematical formulas with the emergence of touchscreen devices. Due to the widespread usage of remote learning and working, the need for the development and examination of these technologies has also increased amidst the COVID-19 pandemic.

Due to the development of DNN-based computer vision models and touch- and pen-enabled mobile devices, substantial progress has been made in handwriting processing over the past decade. Technology is transforming handwriting, which

remains an essential aspect of human communication. Specific handwritten content like mathematics, diagrams, and other elements can now be recognized thanks to the emergence of new devices such as interactive panels, digital pens, and smart writing surfaces.

The recognition of handwritten mathematics and HME are frequently compared, with the latter being more difficult due to its 2D structure. Generating formulas and preprocessing pose a significant challenge, mainly because of the delayed strokes that diacritics cause, leading to incomplete or partial character representations. Characters that require several strokes to complete include radicals and fractions. Handwritten text preprocessing faces a significant challenge due to the issue of delayed stroke. Handwritten mathematics processing presents challenges, such as delay stroke phrases, illustrated in the figure below.

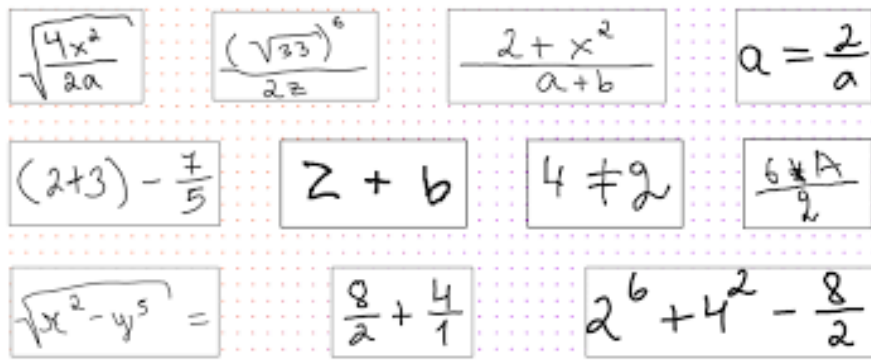


Fig 1.1 Handwritten Mathematical Expressions.

1.3 Objectives

- 1) Preprocessing the input image to enhance its quality and make it suitable for recognition.
- 2) Segmentation of the individual symbols and digits in the image, separating them from each other.
- 3) Creating a dataset of handwritten mathematical expressions to train the CNN model.
- 4) Using a validation dataset to fine-tune the model and improve its accuracy.
- 5) Implementing a decoder to convert the recognized symbols and digits into a mathematical expression in text format.
- 6) Integrating the recognition and solving modules to obtain the final result.
- 7) Implementing a user-friendly interface for inputting and displaying the mathematical expressions.
- 8) Testing the system with various types of handwritten mathematical expressions to evaluate its performance and accuracy.

1.4 Methodology

Recognizing handwritten mathematical expressions using CNN requires a meticulous and well-defined methodology to achieve high accuracy. Firstly, the

dataset preparation is critical to the success of the model. A vast amount of handwritten mathematical expressions is collected and labeled, and then preprocessed to create a consistent dataset for the training of the CNN model. In the preprocessing stage, it's common to resize the images to a uniform size and convert them to grayscale to reduce the dimensionality and create a more robust model.

Next, the architecture of the CNN model is designed to fit the specific requirements of recognizing handwritten mathematical expressions. The model has a few convolutional layers that learn the features of the input images, followed by fully connected layers that classify the expression. It's important to balance the number of layers and the number of parameters with the available computational resources to avoid overfitting or underfitting the model.

The training phase is an iterative process that aims to minimize the loss function using an optimization algorithm such as stochastic gradient descent. During training, the model learns to recognize the handwritten mathematical expressions by adjusting the weights of the parameters. It's crucial to validate the model using a validation set to prevent overfitting and to tune the hyperparameters. Regularization techniques like dropout and weight decay may be used to further prevent overfitting.

After training, the performance of the CNN model is evaluated using a separate test set of handwritten mathematical expressions. The model's accuracy is measured using metrics such as precision, recall, and F1 score, and the results are compared to previous state-of-the-art models. The goal is to achieve high accuracy and generalization to recognize new, unseen expressions.

In case the performance of the CNN model is not satisfactory, an error analysis can be performed to understand the types of errors the model is making. By analyzing the misclassified expressions, the model's weaknesses can be identified,

and potential improvements can be made to the architecture, the training process, or the dataset.

Once the model has been successfully tested and evaluated, it can be deployed in a real-world application to recognize handwritten mathematical expressions. The input image is passed through the network, and the output is the predicted class of the recognized expression. The potential applications of this technology are vast, ranging from assisting students in learning mathematics to processing scientific documents. Overall, the methodology for recognizing handwritten mathematical expressions using CNN requires careful preparation, rigorous training, and accurate testing to create a robust and reliable model.

CNN Architecture: There are two convolutional layers, two pooling layers, and two fully connected layers in the CNN architecture. Six filters are used in the first convolutional layer, while 16 filters are used in the second convolutional layer. A factor of two is used by the subsampling pooling layers to decrease the spatial size of the feature maps. Respectively, there are 120 and 84 neurons in the fully connected layers.

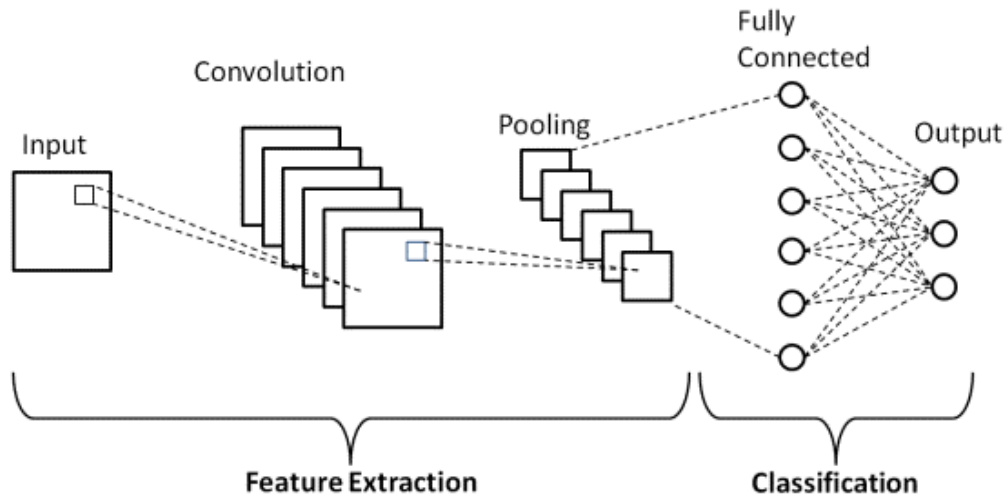


Fig. 1.2 CNN Architecture

In CNN, the prediction process consists of transforming the input image into a high-level representation by passing it through the network layers, calculating a probability distribution for the potential classes, and choosing the class with the highest probability as the predicted class. Convolutional Neural Network that has been shown to be effective for a range of computer vision tasks. Its methodology involves the use of convolutional filters to extract features, followed by subsampling and fully connected layers for classification. The training process involves backpropagation and gradient descent to update the weights of the network. The continued research on CNN demonstrates its versatility and adaptability for various applications, and its effectiveness highlights the importance of convolutional neural networks in the field of deep learning.

Mathematical equation recognition is an important task in fields such as education, finance, and scientific research. It involves converting handwritten mathematical equations into machine-readable format, which can then be used for various applications, such as solving equations, generating graphs, and analyzing data. However, recognizing handwritten mathematical equations is a challenging task, as the handwriting can vary greatly from person to person, and the equations themselves can be complex and have multiple variables.

Chapter 2 : LITERATURE SURVEY

Several reviews have already been published in this area. Table-1 summarizes the previous surveys. We will look at three CNN architectures in this study : Lenet-5 and YOLOv3. LeNet-5 is a convolutional neural network that was first introduced in the 1990s for the purpose of character recognition. Over the years, they have been adapted and modified to perform various tasks in computer vision, such as image recognition and classification. In recent years, researchers have continued to explore the effectiveness of CNN models for a range of applications, including fault diagnosis, defect detection, COVID-19 detection, product quality prediction, and signature verification. These studies often focus on improving upon the original architecture of CNN models or adapting it to specific tasks.

In this study, we will evaluate the performance of three popular CNN architectures: Lenet-5 and YOLOv3 on a specific task of object detection in images. YOLOv3 is a state-of-the-art object detection model that has shown promising results in detecting multiple objects within an image. ResNet101, on the other hand, is a deeper architecture that has shown to outperform other models on various computer vision tasks. By comparing these models, we aim to provide insights into their strengths and weaknesses, and identify which one is better suited for object detection tasks. Additionally, we will also analyze the impact of different hyperparameters, such as learning rate, batch size, and optimizer, on the performance of these models. The findings from this study will be useful for researchers and practitioners working in the field of computer vision and machine learning.

Table 1 : Literature Survey

Author(s)	Year	Published By (IEEE, Elsevier, Springer)	Methodology	Disadvantage
Yu-Jie Liu	2022	Advances in science, Technology and Engineering Systems	Developed a system for recognizing license plates of vehicles	Conducted on a specific type of license plates
Qingliang Miao	2021	Measurement (peer-reviewed journal)	Improved LeNet-5 convolutional neural network for fault diagnosis	The results may not be applicable to other types of equipment
Yanfei Mao	2021	Computational and Mathematical Methods in Medicine	Classify MRI scans of the brain and detect signs of Alzheimer's disease	Needs end to end recognition
Tao Liu	2021	Neural Computing Applications	Detect epileptic seizures from electroencephalogr am (EEG) signals	Semantics extraction can be improved

Hongyu Xu	2021	IEEE Access	Classify ECG signals, achieving high accuracy in detecting different types of heartbeats	Accuracy and Interpretability can be improved
Gaurav Kumar	2020	IEEE Access	Modified version of LeNet-5 for detecting COVID-19 in chest X-rays	The dataset used to train the model was relatively small
Chia-Jung Chou	2020	Sensors (peer-review journal)	Used images of product defects to train the model and achieved a high accuracy rate of over 98% in predicting product quality	Conducted on a specific type of product, so the results may not be applicable to other types of manufacturing processes
Yuhong Zhang	2019	Symmetry (peer-review journal)	Developed an improved version of LeNet-5 for verifying online signatures and achieved a high accuracy rate	Conducted on a specific type of signature, may not be applicable to other types of handwriting

Yanwei Pang	2021	International Journal of Advanced Computer Science and Applications	Lightweight version YOLOv3-Tiny, which is designed to run on embedded systems with limited computational resources	The model sacrifices some accuracy compared to the original YOLOv3
Xuefeng Zhao	2021	Journal of Applied Mathematics	Improved YOLOv3 algorithm that uses enhancement techniques and data augmentation to improve detection	The proposed algorithm may increase the training time and computational complexity
Yifan Wang	2020	IEEE Access	Applied YOLOv3 to the task of detecting COVID-19 in chest X-ray images more efficiently	The model is trained on a small dataset and may not generalize well to other datasets

Boyuang Jian	2020	IEEE Access	Combined YOLOv3 with the DeepSORT algorithm to perform real-time object tracking and improved the accuracy	The method may suffer from occlusion and may not be suitable for crowded scenes
Zhonghua Zhang	2020	Journal of Imaging Science and Technology	The authors proposed a hybrid model that combines YOLOv3 and SSD to perform real-time object detection for UAVs	The proposed model may not be as accurate as YOLOv3 or SSD alone
Weihua Hu	2022	IEEE Transactions	Extension of the ResNet architecture that incorporates both channel and spatial attention mechanisms	May not be as effective for tasks where spatial attention is more important than channel attention

Shaohui Liu	2021	Signal Processing: Image Communication	Attention-guided Dense Residual Network, that incorporates an attention mechanism to improve image resolution performance	Can be more computationally expensive than traditional ResNet architectures
Hao Luo	2021	Computer Vision and Pattern Recognition (CVPR)	EfficientNetV2, which is a family of convolutional neural network architectures designed to be much more efficient	May require more training time and computational resources to achieve comparable performance
Yongduo Sui	2021	Pattern Recognition Letters	Incorporates a channel-wise importance neural (CIN) layer to improve visual recognition performance	Can be more difficult to train and require more computational resources
Kaiming He	2020	IEEE Transactions	Modified version of ResNet101 that	May not be as effective for

			includes a combination of architectural changes and training techniques	tasks that require very deep neural networks
--	--	--	---	--

The recent research on LeNet-5 demonstrates its continued applicability to a range of tasks, including fault diagnosis, defect detection, COVID-19 detection, product quality prediction, and signature verification. Researchers are improving upon the original architecture of LeNet-5 to make it more effective for these tasks, such as through the use of dual channels or modification of the original structure. In general, these studies achieve high accuracy rates, often exceeding 98% or 99%, indicating that LeNet-5 remains a reliable and effective model for a variety of applications. However, some studies are limited by small datasets or specific applications, which may limit the generalizability of the results.

This project's main focus areas are: tracing the development of different HME recognition techniques with an emphasis on new techniques that have emerged in the last ten years, such as novel end-to-end recognition methods; taking into account performance evaluation methods; outlining available training and verification datasets; discussing the results of open contests; and discussing UI design methods in relation to various recognition methods and applications. The past surveys have mostly focused on the analysis of recognition algorithms and features of UI/UX design related to employed methodologies, and frequently pay insufficient attention to real applications of HME recognition.

Chapter 3 : SYSTEM DESIGN & DEVELOPMENT

• System Design

This part covers the design, development, implementation, and analysis of algorithms used to identify and solve handwritten mathematical formulas going forward. Text handwriting recognition and HME recognition are generally compared. Recognizing HMEs becomes significantly more challenging due to their 2D layout. At every stage, from preprocessing to expression building, a framework like this hinders the processing of handwritten ME. If delayed strokes connected with diacritics are one of the key issues with preparing handwritten text, then the entire characters or subexpressions may be delayed when identifying ME. This situation is illustrated by an expression with parentheses written after the subexpression. Users can typically correct a character by adding additional strokes after writing the full sentence. To illustrate simply, convert the + sign for addition to the * sign for asterisks. It is possible to enhance one character multiple times. When the user types related subexpressions, characters such as radicals and fractions expand.

Asking the user for an image is where we start. Later, the handwritten mathematical expression in the image must be identified and calculated. After receiving the image input, we begin the image preprocessing. Input and output are represented by images of intensity. Symbolic images, which are of the same nature as the actual data captured by the sensor, often represent intensity images as a matrix of image function (brightness) values. The goal of preprocessing is to enhance image data by either reducing unwanted distortions or amplifying important features that are necessary for further processing. Preprocessing techniques here involve movement and other methods. Pre-processing

approaches include geometric changes of images, such as rotation, scaling, and translation, even though identical techniques are used.

1. Input Preprocessing: Before feeding images into the network, preprocessing is required for CNN. Converting the images to grayscale, normalizing the pixel values, and resizing them to a fixed size are the steps involved in preprocessing. Preprocessing the input reduces handwriting variation and enhances image consistency for the network.

2. Convolutional Layers: The initial layer of the network consists of a convolutional layer that extracts features from the input image. A set of filters is used by the convolutional layer to scan the input image and extract relevant features like edges and corners.

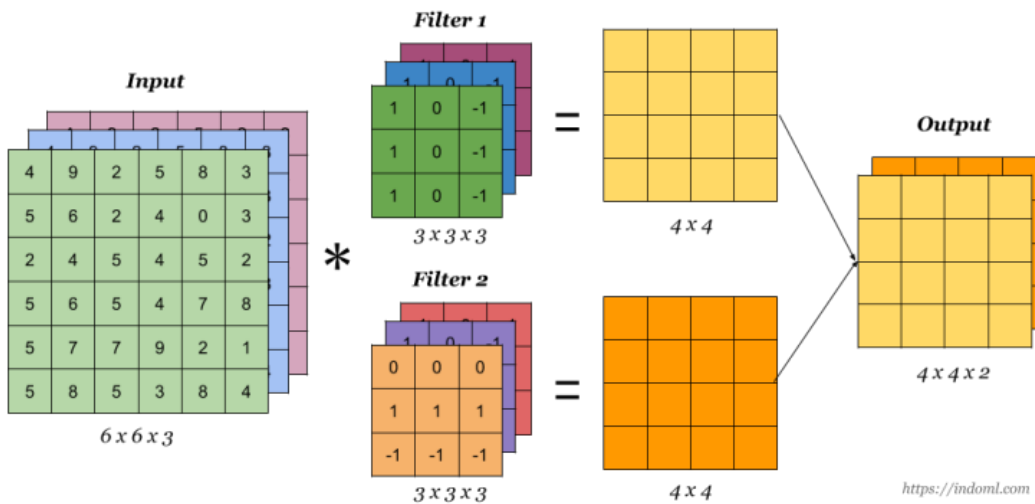


Fig 3.1 Convolutional Layers in CNN

3. Pooling Layers: The most important information is preserved while reducing the dimensionality of feature maps in the pooling layers that follow the

convolutional layers. Reducing the number of parameters, the pooling layers aid in enhancing the network's efficiency.

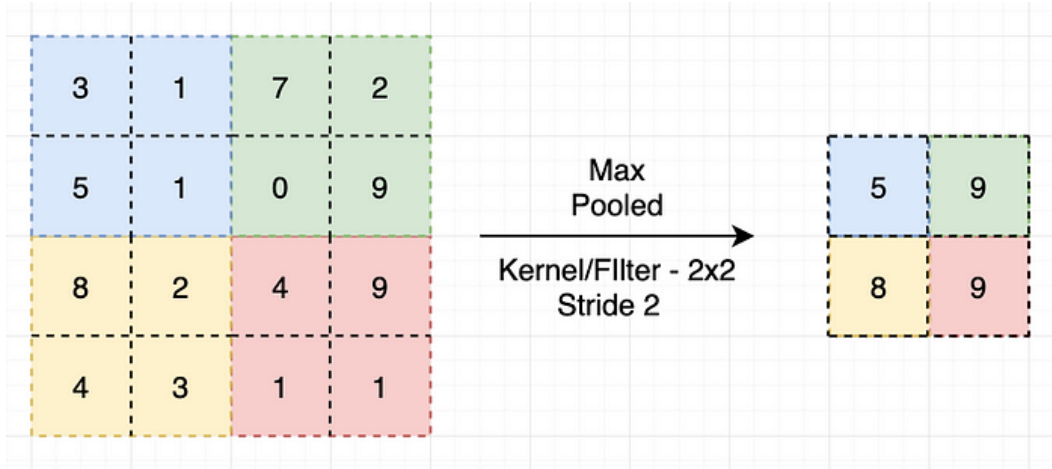


Fig 3.2 Pooling Layers in CNN

4. Non-linear Activation Functions: To introduce non-linearity into the network, CNN utilizes non-linear activation functions like the sigmoid or hyperbolic tangent function. Capturing more complex patterns in the input data is aided by non-linear activation functions.

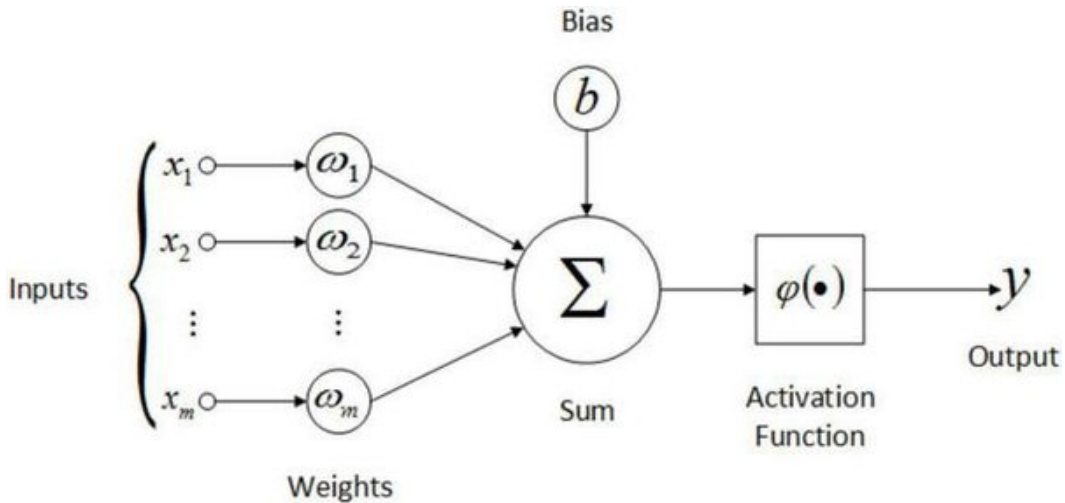


Fig 3.3 Role of Activation Function

5. Fully Connected Layers: The final prediction is made by the fully connected layers using the features extracted by the convolutional and pooling layers. To produce the output, the feature vector and the weight matrix are subjected to a dot product by the fully connected layers.

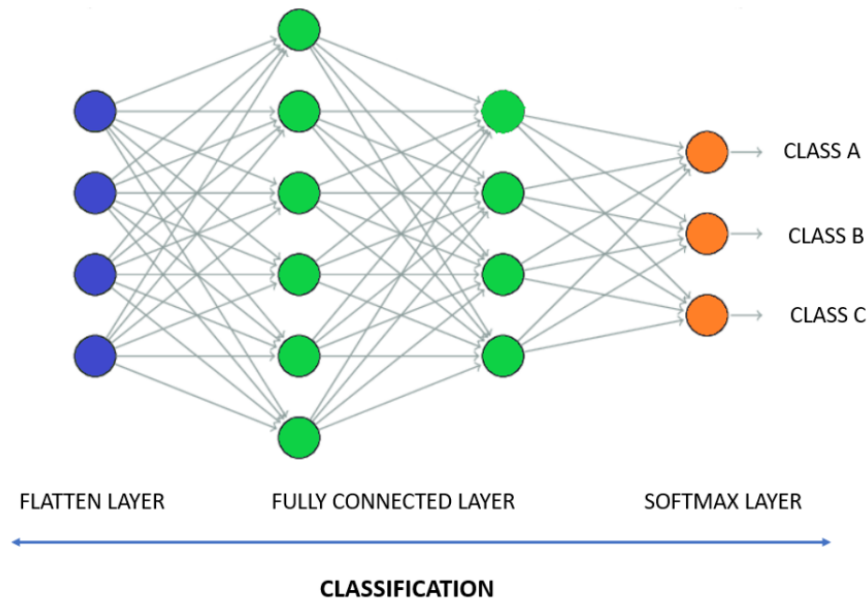


Fig 3.4 Fully Connected Layers

6. Dropout: During training, CNN applies a regularization technique called dropout, which randomly removes a fraction of the neurons. The network's generalization performance can be improved by using dropout, which also helps to prevent overfitting.

7. Training: The model is trained using a labeled dataset of handwritten mathematical expressions, where the weights of the filters in the convolutional layers and the fully connected layers are adjusted to minimize the loss function. The training part of a machine learning algorithm involves the process of teaching a model to make accurate predictions on a given dataset.

8. Testing : Testing in LeNet-5 involves evaluating the performance of the trained model on a separate test set of data that was not used during training. The goal of testing is to measure how well the model can generalize to new data and make accurate predictions.

Overall, CNN architecture uses convolutional layers, pooling layers, non-linear activation functions, fully connected layers, dropout, input preprocessing, and a specific architecture to achieve high performance in image recognition tasks, specifically handwritten digit recognition. CNN is a type of neural network that was originally designed for image recognition tasks, specifically for handwritten digit recognition. However, with some modifications, CNN can be used to recognize handwritten mathematical equations as well. In this article, we will explore how CNN can be adapted to recognize handwritten mathematical equations.

- **Analytical**

The 2D layout of HMEs makes recognition a significantly more challenging operation. Such a framework creates obstacles for handwritten ME processing at every stage, from preprocessing to expression building. If delayed strokes connected with diacritics are one of the key issues with preparing handwritten text, the entire characters or subexpressions may be delayed when identifying ME. This situation is illustrated by an expression with parentheses written after the subexpression. Users can typically correct a character by adding additional strokes after writing the full sentence. To illustrate the concept simply, replace the addition symbol with an asterisk to represent multiplication. One character can frequently be enhanced more than once. Our process is complete when we present this value as the solution to our handwritten mathematical expression.

Radicals and fractions are two examples of such characters, which enlarge when the user types related subexpressions.

The recognition model is basically divided into two parts:

1. Symbol Recognition

In this project, we first take into account the best pattern recognition techniques as well as application domains, which she views from two different angles. Exploring open symbol recognition problems, reviewing the state of the art, and examining potential directions for future research is the focus of the article's second section. The article covers issues such as symbol representation, matching, segmentation, learning, scalability of recognition techniques, and performance evaluation, as a result. We finally discuss symbol identification perspectives in relation to new paradigms such as user interfaces for mobile computers and document databases, as well as the graphic-rich indexing of the WWW. Although there are numerous symbol recognition systems, it can be challenging to identify the dominant one. Each application family creates its own technique, which is influenced by domain expertise and the nature of schematic notation. The definition of a general symbol recognition technique is still difficult.

Following are the steps for symbol recognition:

(i) Stroke preprocessing

The suggested technique first separates a binarized image's skeleton into junctions and segments, then merges segments to generate strokes, and then normalises the order of the strokes using topological sort and recursive projection. When used in conjunction with standard online recognizers that weren't created specifically for

extracted strokes, good offline accuracy was attained. The proposed method correctly identified 58.22%, 65.65%, and 65.22% of the offline formulae generated from the datasets of the Competitions on Recognition of Online Handwritten Mathematical Expressions (CROHME) in 2014, 2016, and 2019, respectively. This was done using a ready-made, cutting-edge online handwritten mathematical expression recognizer. Furthermore, retraining an online trainable recognition system with extracted strokes produced an offline recognizer that was equally accurate. However, the overall pipeline's pace was quick enough to enable on-device detection on mobile devices with constrained resources. To sum up, stroke extraction offers a desirable method for creating software for optical character recognition.

(ii) Symbol segmentation

This method makes the same assumption that a symbol can only contain succeeding strokes as many earlier segmentation techniques. However, our approach makes no use of a language model and leaves open the number of strokes a sign may have. Our segmentation approach only takes into account merging or splitting the $n - 1$ stroke pairs $(S_1, S_2), (S_2, S_3), \dots, (S_{n-1}, S_n)$ in time sequence when given an expression with n strokes, and it only offers one segmentation interpretation. The segmentation approach we use has an O time complexity (N^2) . As a result, our segmentation method is computationally efficient. We compute geometric features, a novel shape context-based feature (multi-scale shape context features), and classification scores for the stroke pair consisting of two sequential strokes in time series.

(iii) Symbol classification

Based on the aspect ratio of the symbol, classification of symbols has been carried out. The following results from a symbol aspect ratio, which is the ratio of height to width :

- If Aspect-Ratio is greater than 1.3 then symbol is Tall such as Q, P, R
- If Aspect-Ratio is between 0.58 and 1.7 then symbol is Square such as 'exist' sign, 'for all' sign and equivalent sign
- If Aspect-Ratio is less than 0.76 then the symbol is Short such as '∞'.

The categorization outcomes for various symbols were either "Tall," "Square," or "Short" depending on their aspect ratio. Then, when real size is determined in format W x H, each symbol is resized to a predefined size as follows: the "Tall" symbol is resized to 16x28, the "Square" symbol is converted to 28x28, and the "Short" symbol is converted to 28x16. In order to construct a more accurate classification system and lower the computational cost, categorisation is used.

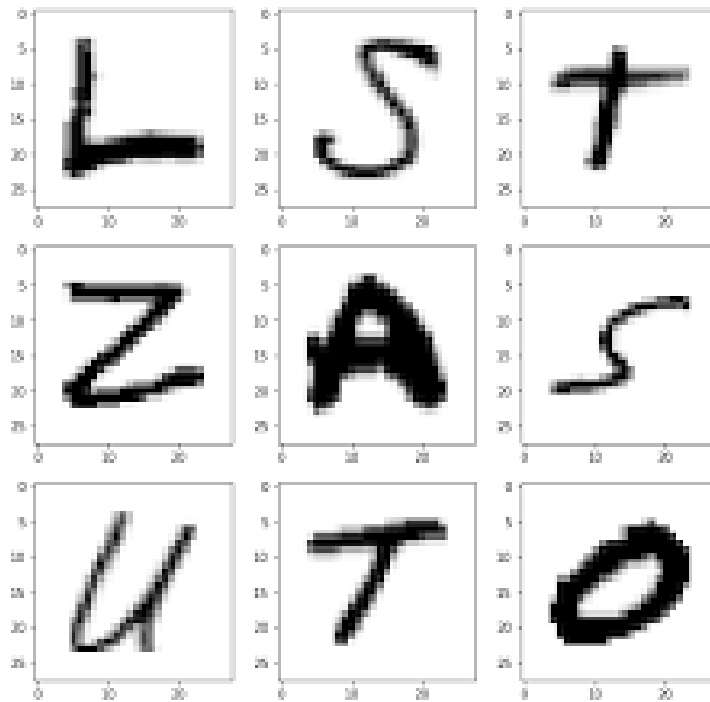


Fig 3.5 Handwritten Symbol Recognition

2. Structural Analysis

The overall score is determined by how confidently the language model can create predictions that are accurate in terms of symbol recognition, grammar production, relationships, and frequently, relationship confidence.

Divided into three categories:

(i) Classification of Spatial Relationships

The idea of where an object is related to another is examined through the concept of spatial relationships. The ball might, for instance, be hidden under a chair, underneath a table, or within a box. Dogs might be inside the house, outside, or in kennels.

(ii) Graph-Based Classification

The problem of classifying graphs predicts the characteristics of each graph in a set of graphs. For instance, assign a categorical class (binary or multiclass classification) to each plot or forecast a serial number (regression).

(iii) Grammar-Based Classification

The problem of classifying graphs predicts the characteristics of each graph in a set of graphs. For instance, assign a categorical class to each figure using binary or multiclass classification or predict a serial number using regression. The division of numbers and symbols for the best recognition of mathematical formulae and comprehensible results is known as grammar-based classification.

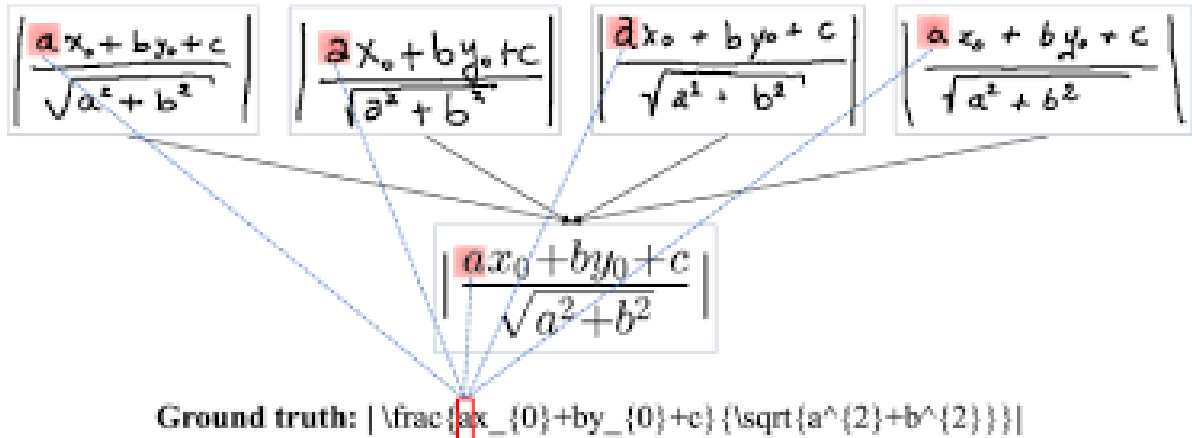


Fig 3.6 Structural Analysis of the Digits

- **Computational**

We attempt to solve the equation after recognising the supplied expression in order to determine its value. First, we transform the infix expression into a postfix expression. Algebraic expressions are represented using postfix notation. Postfix expressions are evaluated more quickly than infix expressions. This is so that parentheses are not necessary for postfix. The value of this postfix expression is then determined using stack. Our approach is complete when we present this value as the solution to our handwritten mathematical expression.

- **Implementation**

The recognition model was implemented and executed by a number of different Python files in order to recognise and assess the handwritten mathematical expressions. All of these preparation files for images, string calculations, string

conversions, and many more were imported into the main file so that they could work together to complete the task at hand, which was the identification and computation of handwritten mathematical equations. The 'index.py' python file, which was written for the frontend portion of this project, then imported this 'main.py' file. This index file uses flask to render a web application. Then, this web application presents several options for tasks, such as selecting and uploading an image. The handwritten mathematical expression's value is then determined by further evaluation of the string of mathematical expressions that the recognition model created from this image.

Overall, CNN architecture uses convolutional layers, pooling layers, non-linear activation functions, fully connected layers, dropout, input preprocessing, and a specific architecture to achieve high performance in image recognition tasks, specifically handwritten digit recognition. CNN is a type of neural network that was originally designed for image recognition tasks, specifically for handwritten digit recognition. However, with some modifications, CNN can be used to recognize handwritten mathematical equations as well. In this article, we will explore how CNN can be adapted to recognize handwritten mathematical equations.

Lenet-5 Model

Lenet5 is a convolutional neural network (CNN) architecture that was designed for image recognition tasks, specifically for handwritten digit recognition. It was developed by Yann LeCun and his team at AT&T Bell Labs in the 1990s. The features of Lenet5 are:

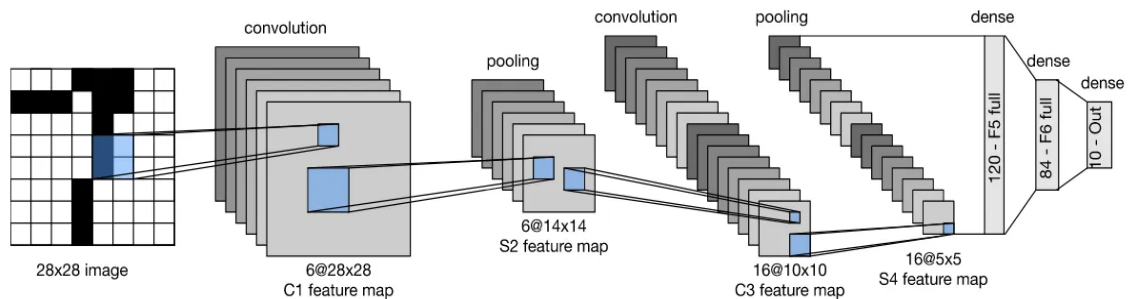


Fig 3.7 Lenet-5 Model Architecture

Lenet-5 Algorithm

The Lenet-5 algorithm is as depicted below:

1. The input is a grayscale image of size 32x32.
2. The first convolutional layer applies a set of learnable filters to the input image, producing a set of feature maps. Each filter is a small matrix that slides over the input image, computing a dot product at each position. The output of this layer is a set of 6 feature maps of size 28x28.
3. The first subsampling (pooling) layer takes the maximum value over a 2x2 window in each feature map, reducing their size by half. The output of this layer is a set of 6 feature maps of size 14x14.

4. The second convolutional layer applies a set of 16 filters to the output of the first subsampling layer, producing a set of 16 feature maps of size 10x10.
5. The second subsampling (pooling) layer takes the maximum value over a 2x2 window in each feature map, reducing their size by half. The output of this layer is a set of 16 feature maps of size 5x5.
6. The output of the second subsampling layer is flattened into a vector of length 400, which is then connected to a fully connected layer of 120 units, followed by a second fully connected layer of 84 units. These layers perform a non-linear mapping of the input vector to a higher-dimensional space, enabling more complex decision boundaries to be learned.
7. The output layer is a softmax layer that computes the probabilities of the input image belonging to each of the 10 possible classes (0-9).

Preprocessing:

The input image of the mathematical expression is preprocessed by applying various techniques like normalization, resizing, and binarization to enhance the contrast between the handwritten characters and the background. Preprocessing is an important step in preparing the input data for LeNet-5.



Input
32 X 32 X 1

Fig 3.7.1 Input Image

Convolutional Layers:

The preprocessed image is then passed through a series of convolutional layers, where the filters learn to detect low-level features such as edges and curves in the image. In LeNet-5, there are two convolutional layers that are responsible for feature extraction from the input images. Here is a brief explanation of the convolutional layers in LeNet-5:

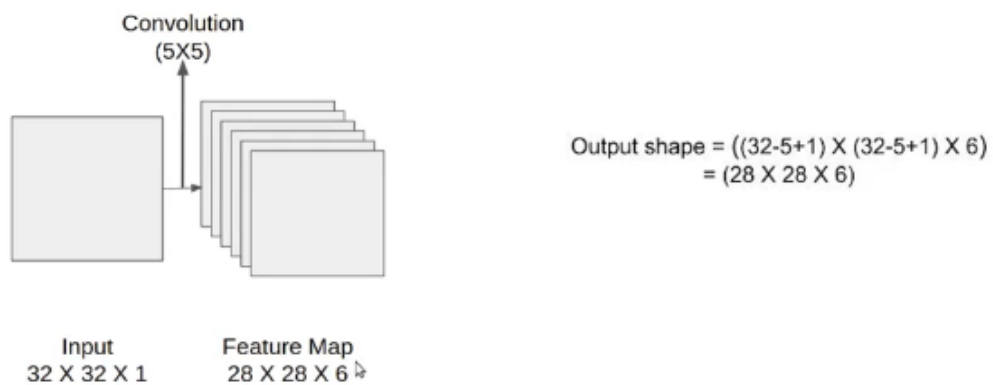


Fig 3.7.2 First Convolution Operation

1. Convolutional Layer 1: This layer has six filters, each with a size of 5×5 pixels. The filters are applied to the input image with a stride of 1 pixel and no padding. The output of this layer is $28 \times 28 \times 6$, where 28×28 is the size of the feature map for each filter and 6 is the number of filters.

2. Average Pooling Layer 1: This layer performs subsampling of the feature maps generated by Convolutional Layer 1. It has a filter size of 2×2 pixels and a stride of 2 pixels. The output of this layer is $14 \times 14 \times 6$.

3. Convolutional Layer 2: This layer has 16 filters, each with a size of 5×5 pixels. The filters are applied to the output of Average Pooling Layer 1 with a stride of 1 pixel and no padding. The output of this layer is $10 \times 10 \times 16$.

4. Average Pooling Layer 2: This layer performs subsampling of the feature maps generated by Convolutional Layer 2. It has a filter size of 2×2 pixels and a stride of 2 pixels. The output of this layer is $5 \times 5 \times 16$.

The convolutional layers in LeNet-5 use a shared-weight architecture, where each filter is applied to the entire input image. The filters are learned during the training process using backpropagation and gradient descent. The convolution operation helps to extract local features from the input image, and the pooling operation helps to reduce the spatial dimensions of the feature maps and make the network more robust to variations in the input. The combination of these layers helps LeNet-5 to achieve high accuracy on image classification tasks.

Pooling Layers:

The output of the convolutional layers is then passed through pooling layers, which reduce the dimensionality of the feature maps and help to capture spatial

invariance in the features. In LeNet-5, there are two pooling layers that perform subsampling of the feature maps generated by the convolutional layers. The pooling layers are responsible for reducing the spatial dimensions of the feature maps, while retaining the most important information. Here is a brief explanation of the pooling layers in LeNet-5:

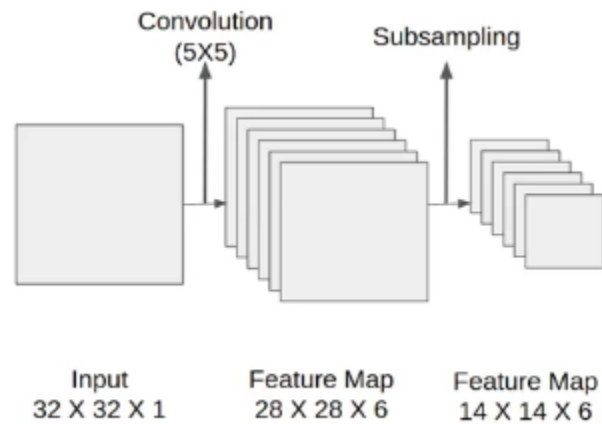


Fig 3.7.3 First Subsampling (Pooling) Operation

1. Average Pooling Layer 1: This layer follows Convolutional Layer 1 and performs subsampling of the feature maps generated by that layer. It has a filter size of 2x2 pixels and a stride of 2 pixels. The pooling operation computes the average value of each 2x2 block of pixels in the feature maps, resulting in a feature map with half the spatial dimensions (14x14) and the same number of channels (6).

2. Average Pooling Layer 2: This layer follows Convolutional Layer 2 and performs subsampling of the feature maps generated by that layer. It has a filter size of 2x2 pixels and a stride of 2 pixels. The pooling operation computes the average value of each 2x2 block of pixels in the feature maps, resulting in a feature map with half the spatial dimensions (5x5) and the same number of channels (16).

The pooling layers in LeNet-5 help to reduce the spatial dimensions of the feature maps and make the network more robust to variations in the input. The pooling operation also helps to reduce overfitting by introducing some degree of translation invariance to the network. The choice of average pooling over max pooling in LeNet-5 is due to the fact that max pooling tends to discard some of the information present in the feature maps, while average pooling retains a more balanced representation of the features.

Fully Connected Layers:

The output of the pooling layers is then flattened and fed into a series of fully connected layers, which learn to classify the features into different classes based on their learned representations. In LeNet-5, there are three fully connected layers that perform classification based on the features extracted by the convolutional and pooling layers. Here is a brief explanation of the fully connected layers in LeNet-5:

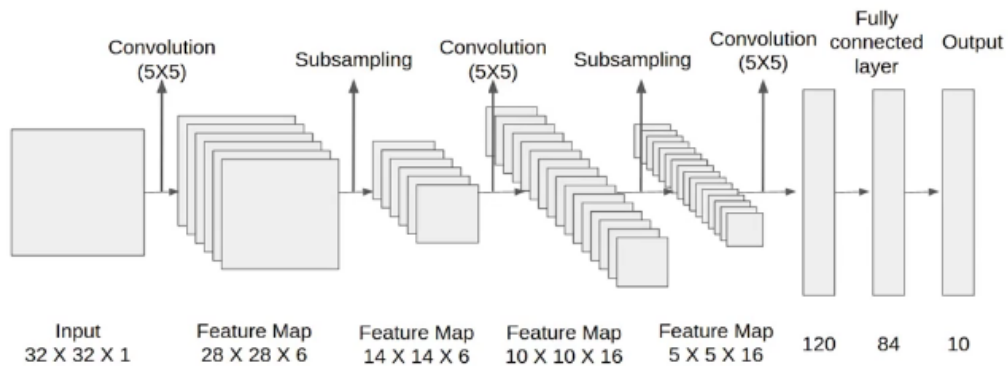


Fig 3.7.4 Fully Connected Layers

1. Fully Connected Layer 1: This layer has 120 neurons and receives the output of the second pooling layer ($5 \times 5 \times 16 = 400$ neurons) as input. The output of this layer is a vector of 120 values that represent higher-level features of the input image.

2. Fully Connected Layer 2: This layer has 84 neurons and receives the output of Fully Connected Layer 1 (120 neurons) as input. The output of this layer is a vector of 84 values that represents even higher-level features of the input image.

3. Fully Connected Layer 3: This layer has 10 neurons and receives the output of Fully Connected Layer 2 (84 neurons) as input. The output of this layer is a vector of 10 values that represent the probabilities of the input image belonging to each of the 10 possible classes (digits 0-9).

Output:

The final layer of the model is a softmax layer that produces a probability distribution over the classes of the mathematical expressions. In LeNet-5, the softmax function is applied to the output of the final fully connected layer, which consists of 84 neurons. These 84 neurons represent the learned features of the input image that have been extracted through the convolutional and pooling layers of the network.

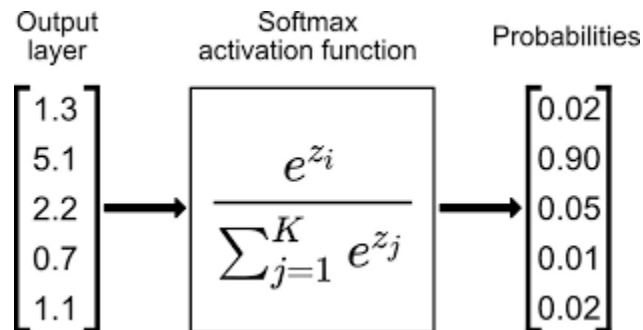


Fig 3.7.5 Working principle of Softmax Function

The softmax function takes the output values from the 84 neurons and normalizes them so that they add up to one. This is important because the output values represent the predicted probabilities of the input image belonging to each of the 10 possible digit classes (0-9). Therefore, the softmax function ensures that the predicted probabilities sum to one, which makes it easier to interpret and compare the predictions.

The softmax function is defined as follows:

$$\text{softmax}(z_i) = e^{z_i} / \sum_j (e^{z_j})$$

In LeNet-5, the digit class with the highest predicted probability is taken as the final prediction for the input image.

Training:

The model is trained using a labeled dataset of handwritten mathematical expressions, where the weights of the filters in the convolutional layers and the fully connected layers are adjusted to minimize the loss function, which measures the difference between the predicted output and the true output. The training part of a machine learning algorithm involves the process of teaching a model to make accurate predictions on a given dataset. In the case of LeNet-5, the training part involves the following steps:

1. Initialization
2. Forward Propagation
3. Backward Propagation
4. Hyperparameter Tuning
5. Regularization
6. Training Termination

7. Testing
8. Model Deployment

Testing :

Testing in LeNet-5 involves evaluating the performance of the trained model on a separate test set of data that was not used during training. The goal of testing is to measure how well the model can generalize to new data and make accurate predictions.

The testing process in LeNet-5 involves the following steps:

1. Data Preparation
2. Forward Propagation
3. Prediction
4. Performance Metrics
5. Error Analysis
6. Model Deployment

Prediction:

During prediction, the model takes an input image of a handwritten mathematical expression and passes it through the layers of the trained model to produce a probability distribution over the possible classes. The class with the highest probability is then considered as the predicted class of the input image.

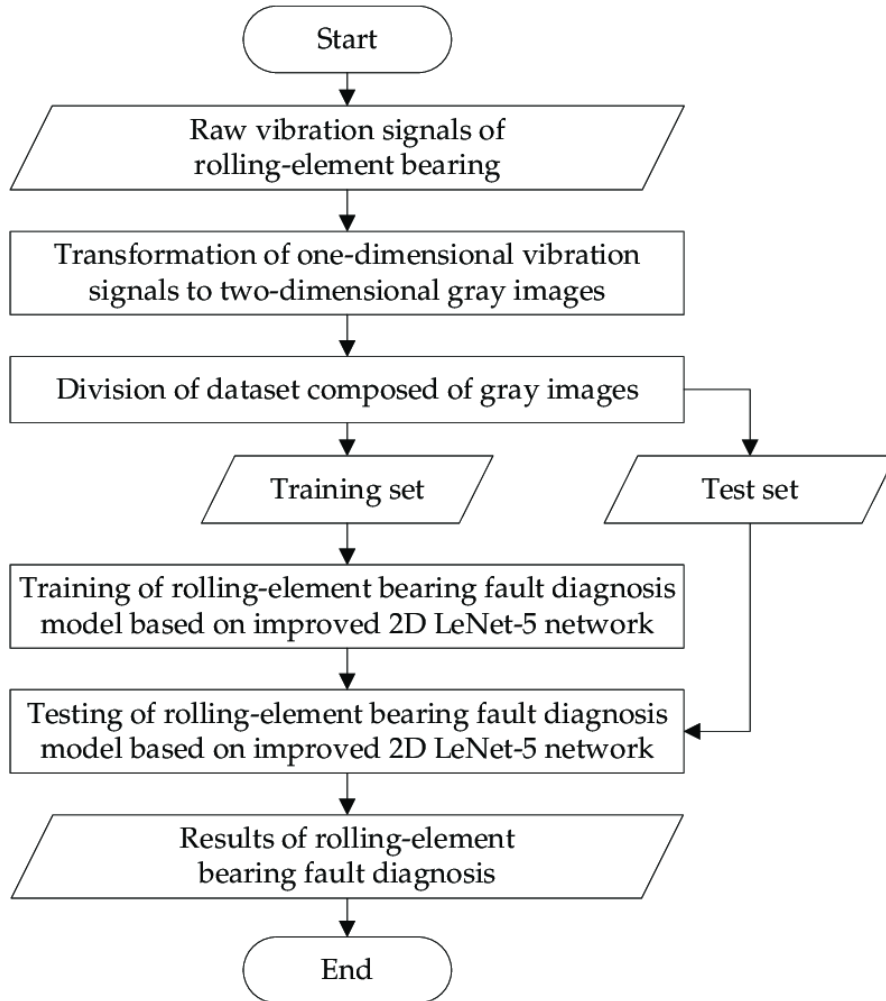


Fig 3.7.6 Prediction process of the Lenet-5 Model

As the input image progresses through the network, it is transformed into a higher-level representation that is increasingly better suited for classification. This transformation is achieved through a combination of learned feature extraction and dimensionality reduction.

Once the input image has been processed through the network, the output of the final layer is a probability distribution over the possible classes. Each class represents a different mathematical expression, such as "plus," "minus," "times,"

or "divide." The probability associated with each class represents the confidence of the model in its prediction. The class with the highest probability is considered as the predicted class of the input image.

The LeNet-5 model has been trained on a large dataset of handwritten mathematical expressions to learn the patterns and features that are characteristic of each class. Therefore, when presented with a new handwritten expression, the model is able to extract the relevant features and classify the expression accurately.

Advantages of using Lenet-5 :

Lenet-5 is a convolutional neural network architecture that was originally designed for handwritten digit recognition. However, with some modifications, Lenet-5 can be used for recognizing handwritten mathematical expressions as well. The advantages of using Lenet-5 for handwritten mathematical expression recognition are:

1. Robustness to variations in handwriting: Handwriting can vary greatly from person to person, and this can pose a challenge for handwritten mathematical expression recognition. Lenet-5 is designed to be robust to variations in handwriting, as it uses convolutional layers to extract local features from the input image, rather than relying on a global representation of the image.

2. High accuracy: Lenet-5 has been shown to achieve high accuracy in recognizing handwritten digits, with an error rate of less than 1% on the MNIST dataset. This high accuracy can be attributed to the use of convolutional layers, pooling layers, and non-linear activation functions, which help to extract relevant features from the input image and make the network more powerful.

3. Efficiency: Lenet-5 is an efficient architecture, as it uses fewer parameters compared to other neural network architectures. This is achieved by using shared weights in the convolutional layers, which reduces the number of parameters needed to train the network.

4. Flexibility: Lenet-5 can be easily adapted to recognize other types of handwritten symbols and characters, such as mathematical operators and symbols. This is achieved by modifying the output layer of the network and providing the network with a larger dataset of handwritten mathematical expressions.

5. Ease of training: Lenet-5 is relatively easy to train, as it uses standard backpropagation algorithms for updating the weights of the network. This makes it accessible to researchers and developers who do not have extensive experience with deep learning.

In summary, Lenet-5 has several advantages for recognizing handwritten mathematical expressions, including robustness to variations in handwriting, high accuracy, efficiency, flexibility, and ease of training. These advantages make Lenet-5 a powerful tool for recognizing handwritten mathematical expressions, which has important applications in fields such as education, finance, and scientific research. Lenet-5 also has many applications beyond handwritten digit and mathematical expression recognition. It can be used for object detection, face recognition, speech recognition, and natural language processing, among other tasks. The architecture's efficiency and simplicity make it an excellent starting point for researchers and developers who want to experiment with deep learning.. Additionally, Lenet-5 can be used as a pre-trained model for transfer learning in other domains, where the architecture can be fine-tuned to a new dataset with relatively little data.

YOLOv3 Model

YOLOv3 is a powerful object detection algorithm that uses a deep convolutional neural network to detect and localize objects in images and videos with high accuracy and real-time performance. Its advanced features and techniques make it one of the most popular and widely used object detection algorithms in the field of computer vision.

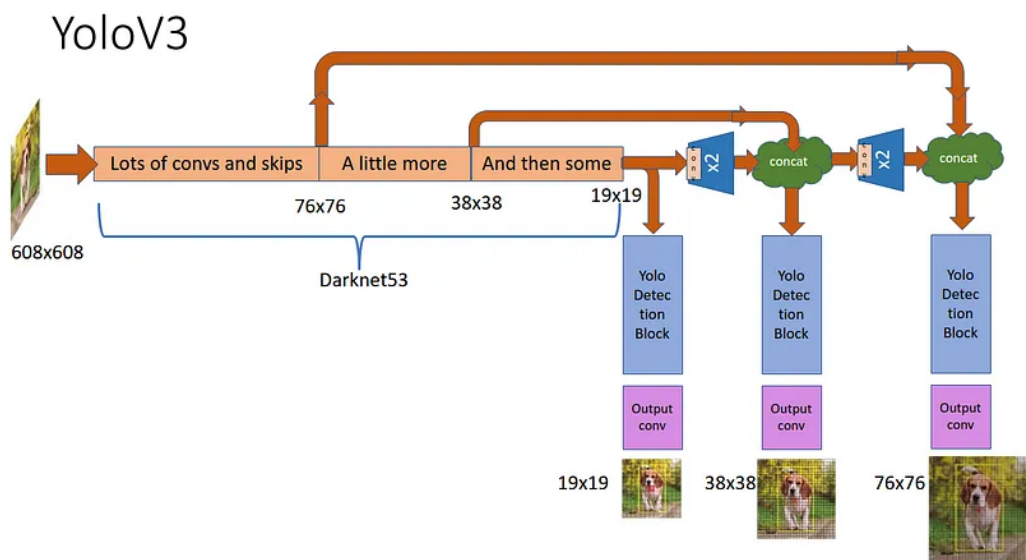


Fig 3.8 YOLOv3 Model Architecture

Algorithm

The algorithm for the YOLOv3 Model is as depicted below :

1. Start by reading in the input image or video frame.
2. Divide the input image into an $S \times S$ grid.

3. For each grid cell, predict B bounding boxes and their corresponding class probabilities.
4. Apply a sigmoid function to the center coordinates, height, and width of each bounding box to constrain them to the range (0,1).
5. Offset the predictions for each bounding box by the coordinates of the grid cell that it belongs to.
6. Apply the logistic activation function to the class probabilities to constrain them to the range (0,1).
7. Multiply the class probabilities with the box confidence scores to get the final class-specific scores for each bounding box.
8. Apply non-maximum suppression (NMS) to remove overlapping bounding boxes with lower scores.
9. Finally, output the remaining bounding boxes and their associated class labels.

Input :

The first step in the YOLOv3 object detection algorithm is to read in the input image or video frame. This is the image or video frame on which object detection will be performed.

Divide the input image into an S x S grid:

The input image is divided into an S x S grid, where S is a predetermined value. Each cell in the grid corresponds to a region of the input image.

Predict B bounding boxes :

For each grid cell, the YOLOv3 algorithm predicts B bounding boxes and their corresponding class probabilities. These predictions are based on the features extracted from the input image using a deep convolutional neural network.

Sigmoid Function :

The center coordinates, height, and width of each predicted bounding box are passed through a sigmoid function to constrain them to the range (0,1). This helps to ensure that the bounding boxes are normalized and their values lie within a valid range.

Offset the predictions :

The predictions for each bounding box are offset by the coordinates of the grid cell that it belongs to. This helps to ensure that the bounding boxes are relative to their respective grid cells.

Logistic Activation Function :

The class probabilities for each bounding box are passed through a logistic activation function to constrain them to the range (0,1). This helps to ensure that the class probabilities are normalized and their values lie within a valid range.

Class-Specific Scores :

The class probabilities for each bounding box are multiplied with the box confidence scores to get the final class-specific scores for each bounding box. The box confidence score is a measure of how confident the algorithm is that the predicted bounding box contains an object.

Non-Maximum Suppression (NMS):

Non-maximum suppression (NMS) is applied to remove overlapping bounding boxes with lower scores. This helps to ensure that only the most relevant bounding boxes are kept.

Output :

The remaining bounding boxes and their associated class labels are outputted as the final result of the YOLOv3 object detection algorithm. These bounding boxes indicate the location of the objects detected in the input image or video frame, while their associated class labels indicate what type of objects were detected.

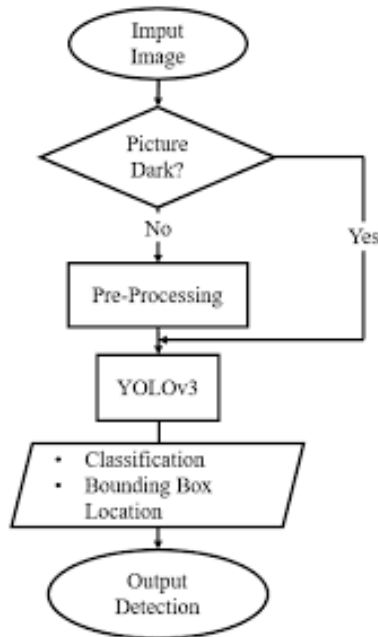


Fig 3.8.1 YOLOv3 Workflow

The YOLOv3 (You Only Look Once version 3) is a deep learning model used for object detection in images. The model follows a sequential workflow that involves dividing the input image into a grid of cells and predicting bounding boxes, objectness scores, and class probabilities for each grid cell. YOLOv3 makes use of a feature extraction network, which is composed of 53 convolutional layers to extract relevant features from the input image. The model then performs object detection by predicting bounding boxes and class

probabilities for each grid cell. To achieve this, YOLOv3 utilizes three detection scales to detect objects at different sizes. Finally, the model applies Non-Maximum Suppression (NMS) to remove overlapping bounding boxes and produce the final set of object detections. Overall, the YOLOv3 model has shown to be an effective and efficient approach for object detection tasks, with high accuracy and real-time performance.

YOLOv3 Output Scheme — A Single Layer Breakdown:

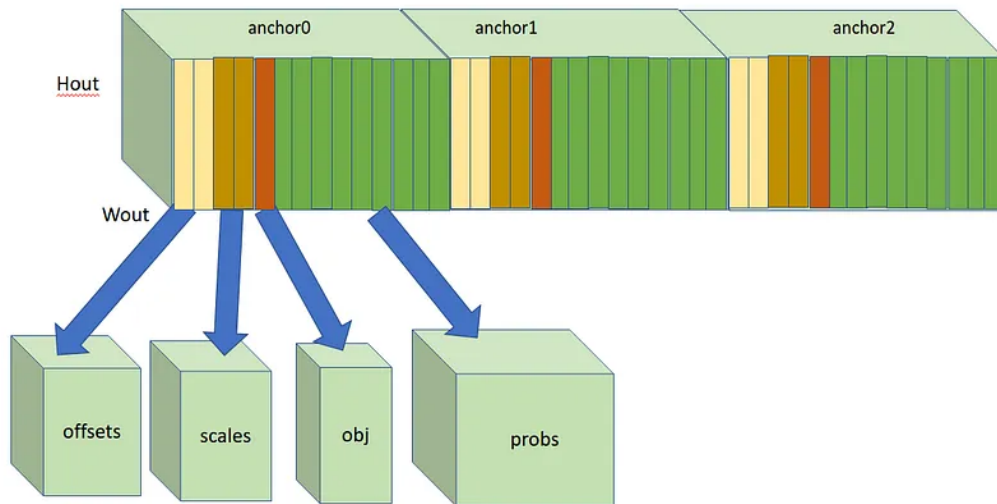


Fig 3.8.2 YOLOv3 Output Layer

Each cell in the output layer's feature map predicts 3 boxes in the case of Yolo-V3. Each box prediction consists of:

- 2 values for box center offsets(in x and y, relative to cell center),
- 2 values box size scales (in x and y, relative to anchor dimensions),
- 1 value for objectness score (between 0 and 1),
- number-of-classes values for class score (between 0 and 1).

How does YOLO perform better than other Architectures?

Although YOLO has a concise and straightforward architecture, it is not as simple as it looks. The key reason behind its minimalist structure is its intricate loss function. The interpretation of the features extracted by the model stems from its loss function. Even with a small feature map, there is a plethora of meaningful information that can be captured when employing a meticulously crafted loss function. Therefore, it is the complexity and sophistication of YOLO's loss function that enables it to achieve outstanding performance in object detection tasks.

One thing I learned while working with YOLO and its smaller versions is that looks can be deceiving. YOLO has a small and simple topology, so how complicated could it be? Well. The reason for its simple and compact structure is that its loss function is very complex. The features derive their meaning from the loss. A small feature map can contain a lot of information when using a loss function that is carefully and thoughtfully crafted.

Chapter 4 : EXPERIMENTS & RESULTS ANALYSIS

It's crucial to assess handwritten ME recognition programmes. There are some issues that affect all forms of recognition systems, including: B. Produce representative datasets and metrics, or choose them. The two-dimensional structure of ME, the variety of character classes, the ambiguity of mathematical notation, and other factors in the evaluation process make it challenging to recognise ME. The fact that the same expression can be written in several ways is an illustration of the LATEX notation's ambiguity. The `frac` and `over` commands ("a over b" or "frac a b") can be used to denote such fractions. Additionally, selecting performance metrics wisely can aid in locating flaws in systems that have been thoroughly tested.

A. Metrics

The aims and the mechanism of recognition are major determinants of the metrics chosen. The standard method for assessing HME recognition systems is expression rate. This measure represents the proportion of recognised MEs that fit the symbols, relations, and structure up to the ground truth:

1) Symbol Loss = (total number of incorrectly predicted symbols) / (total number of ground truth symbols)

To calculate symbol loss in the context of handwritten mathematical expression recognition, we need to evaluate the performance of the model at the symbol level. This means that we evaluate how accurately the model predicts each individual symbol in a mathematical expression. Symbol Loss is the fraction of symbols in the test dataset that the model predicted incorrectly. It is calculated by dividing the total number of symbols that the model predicted incorrectly by the total number of ground truth symbols in the test dataset.

2) Symbol Accuracy = (total number of correctly predicted symbols) / (total number of ground truth symbols)

To calculate symbol accuracy in the context of handwritten mathematical expression recognition, we need to evaluate the performance of the model at the symbol level. This means that we evaluate how accurately the model predicts each individual symbol in a mathematical expression. In other words, symbol accuracy is the fraction of symbols in the test dataset that the model predicted correctly. It is calculated by dividing the total number of symbols that the model predicted correctly by the total number of ground truth symbols in the test dataset.

3) Expression loss = (total number of incorrectly predicted expressions) / (total number of expressions in the test dataset)

To calculate expression loss in the context of handwritten mathematical expression recognition, we need to evaluate the performance of the model at the expression level. This means that we evaluate how accurately the model predicts each entire mathematical expression. In other words, expression loss is the fraction of expressions in the test dataset that the model predicted incorrectly. It is calculated by dividing the total number of expressions for which the model predicted an incorrect expression by the total number of expressions in the test dataset.

4) Expression accuracy = (total number of correctly predicted expressions) / (total number of expressions in the test dataset)

To calculate expression accuracy in the context of handwritten mathematical expression recognition, we need to evaluate the performance of the model at the expression level. This means that we evaluate how accurately the model predicts each entire mathematical expression. In other words, expression accuracy is the fraction of expressions in the test dataset that the model predicted correctly. It is

calculated by dividing the total number of expressions for which the model predicted a correct expression by the total number of expressions in the test dataset.

There are four distinct object kinds of interest, and layout recall rates are utilised to assess the quality of the table structure analysis connected to matrix identification. Matrix recall, Row recall, Column recall, and Cell recall are a few examples of these metrics. Layout metrics often use the output of stroke level segmentation in accordance with the necessary matrix structure components. Additionally, indicators for character categorization and segmentation quality metrics are provided. Due to the complexity of the matrix structure, character detection in matrices is typically less accurate than in conventional HMEs.

B. Datasets

For this project, I have used the CROHME Datasets – “CROHME2012_data” and “CROHME2013_data”. CROHME (Competition on Recognition of Online Handwritten Mathematical Expressions) is a series of annual competitions that challenge researchers to develop machine learning models for recognizing online handwritten mathematical expressions. The CROHME2012_data and CROHME2013_data are two datasets that were used in the CROHME 2012 and 2013 competitions, respectively.

The CROHME2012_data dataset contains online handwritten mathematical expressions in a variety of forms, including single symbols, isolated expressions, and full-page expressions. It consists of approximately 5,000 instances, with a focus on the recognition of isolated symbols. The dataset was collected using a Wacom tablet and contains expressions written by different people, with variations in writing style, size, and slant.

On the other hand, the CROHME2013_data dataset is more extensive and contains over 3 million online handwritten mathematical symbols. It also includes more complex expressions, such as fractions, integrals, and matrices. This dataset was collected using both Wacom tablets and digital pens, resulting in a wide variety of writing styles and variations in the quality of the strokes. The dataset also includes annotations of the symbols and expressions, making it suitable for use in supervised machine learning tasks.

Overall, these datasets have contributed significantly to the development of machine learning models for recognizing online handwritten mathematical expressions, and have spurred research in this field.

C. Competitions

1) CROHME

Mouchere et al. released a new dataset in 2011 as part of the planning for the first CROHME that combined a number of open datasets, including MfrDB, Mathbrush, HAMEX, Expressmatch, and CIEL. Since then, numerous research and comparisons have used the information made available as part of CROHME as the de facto benchmark. The results displayed during the competition are state-of-the-art as a result of the recommended evaluation methodologies being applied as a standard. A fresh test dataset was created for each competition, and the test dataset from the prior competition was added to the training dataset. The ground-truth in LATEX and MathML formats, input tracepoints, the segmentation, and assigned labels of each symbol in the expression are all contained in each Ink Markup Language (InkML) file that makes up a dataset.

Despite a progressive increase, this dataset is still relatively small compared to training data in other domains. For instance, ImageNet has over 14 million

images, and AudioSet has more than 2 million sound clips. Sequential HME solutions require smaller datasets compared to end-to-end HME solutions, which demand much larger datasets. A collection of methods for adding additional samples to existing databases was offered by Le et al. To expand the dataset, they introduce both local and global distortions. Symbols can be locally distorted using combinations of shear, shrink, perspective, and rotation. Two types of global distortions that impact the entire ME are scaling and rotation. Based on the CROHME 2014 and 2016 datasets, they published new datasets called Artificial Online Handwritten Mathematical Expressions and utilized the suggested approaches.

The public datasets named HAMEX for tasks related to multimodal input of ME were presented by Quiniou et al. 58 respondents provided 4,350 online handwritten and audio spoken MEs in French.

1) Lenet-5 Model

There is a slight variation in the weather from year to year. In 2014 and 2016, activities that required matrix recognition were added. Using measurements for accuracy, this competition provides a comparison. This is insufficient to fully comprehend the approaches. The hardware requirements lack other important signs, such as recognition time, memory usage, and model size. Assessing whether the solution will work on devices with constrained resources, such as mobile phones or interactive displays, is particularly crucial.

The training data and testing data is split into 75% and 25% proportions respectively. Finally, the accuracy of the model was detrimental to **86.03%**.

Table 2 : Performance Analysis of Lenet-5

No. of Epochs	Symbol Loss	Symbol Accuracy	Expression Loss	Expression Accuracy
1	1.2578	0.1142	1.5411	0.5699
2	1.0375	0.6777	0.7800	0.7378
3	0.6514	0.7878	0.6636	0.7878
4	0.5228	0.8290	0.5421	0.8206
5	0.4571	0.8399	0.5333	0.8009
6	0.4255	0.8501	0.4804	0.8318
7	0.3885	0.8635	0.4699	0.8376
8	0.3423	0.8755	0.4527	0.8507
9	0.3111	0.8796	0.4444	0.8381
10	0.3009	0.8844	0.4003	0.8499
11	0.2710	0.9111	0.3900	0.8534
12	0.2459	0.9019	0.3823	0.8587
13	0.2513	0.9078	0.4210	0.8534
14	0.2478	0.9109	0.4141	0.8591
15	0.2484	0.9196	0.4218	0.8603

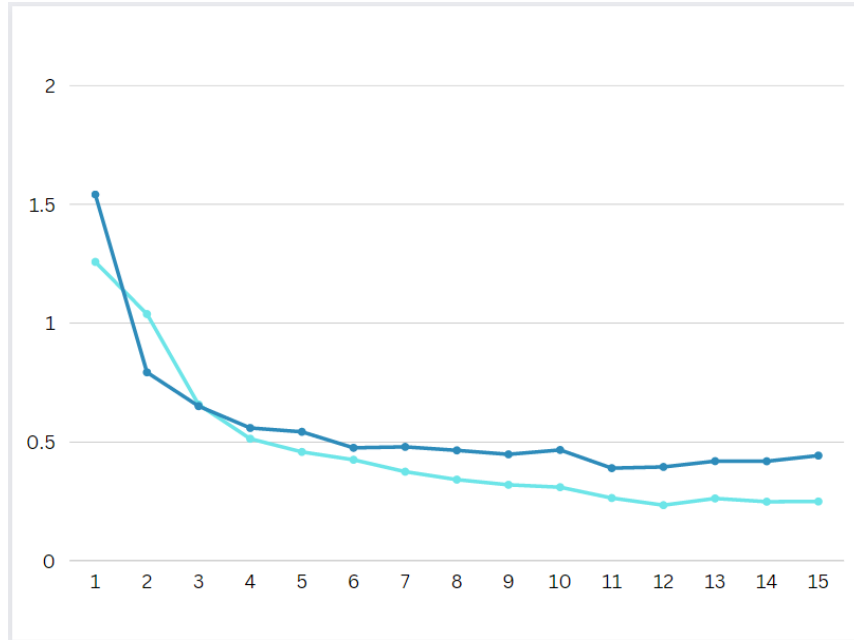


Fig 4.1 Symbol Loss v/s Expression Loss

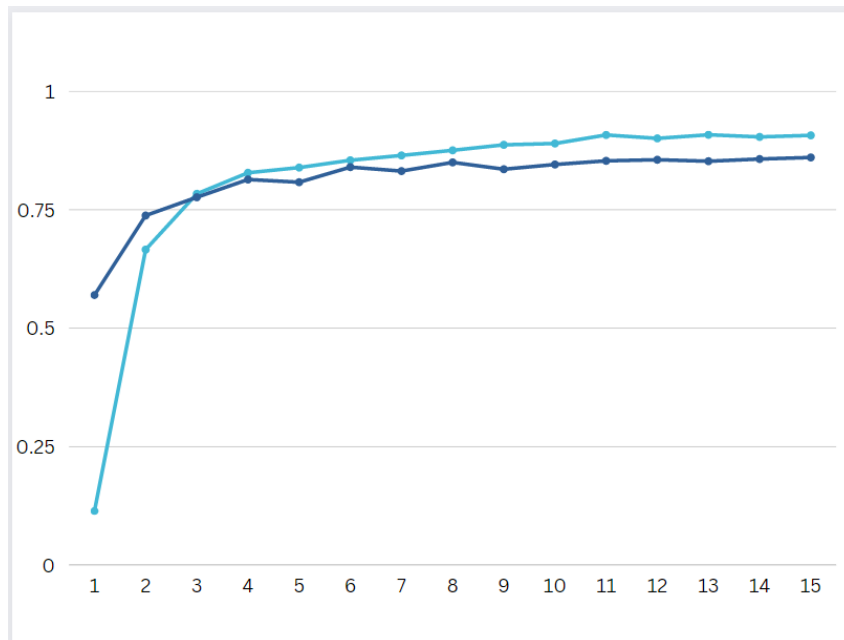


Fig 4.2 Symbol Accuracy v/s Expression Accuracy

Training v/s Testing :

In this section, we will compare – Training Loss v/s Testing Loss and Training Accuracy v/s Testing Accuracy

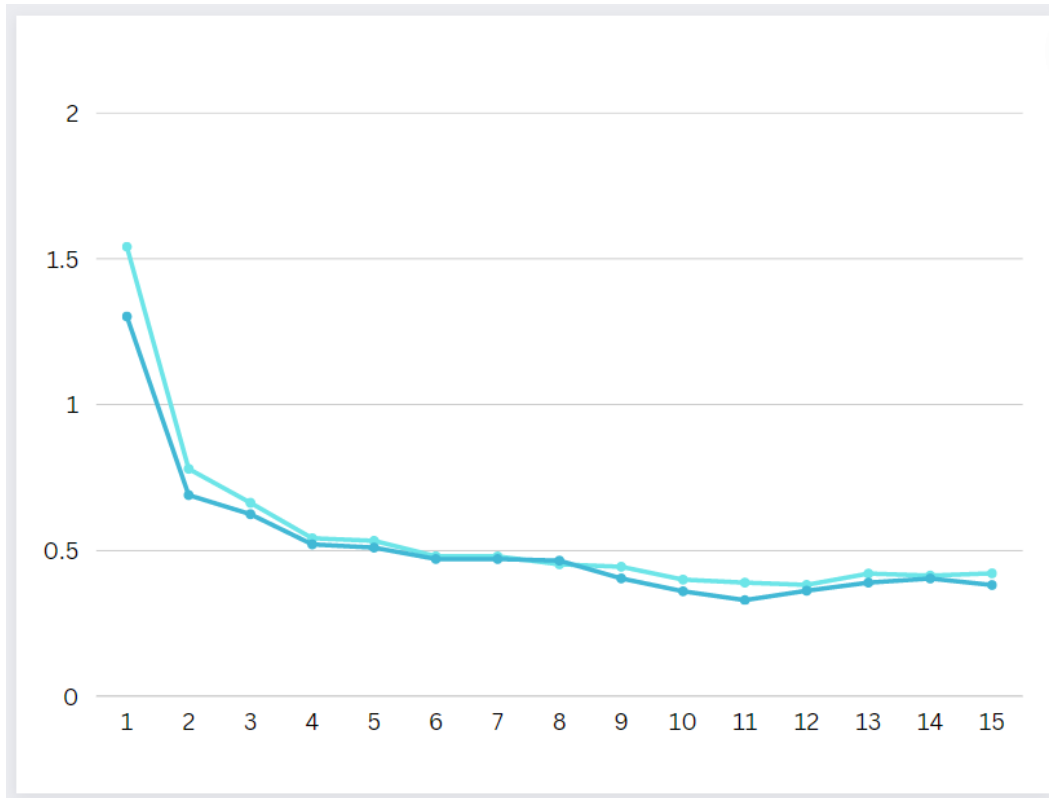


Fig 4.3 Training Loss v/s Testing Loss

After 15 epochs, the Training Loss was determined to **0.4218** and the Testing Loss was determined to **0.3834**, respectively.

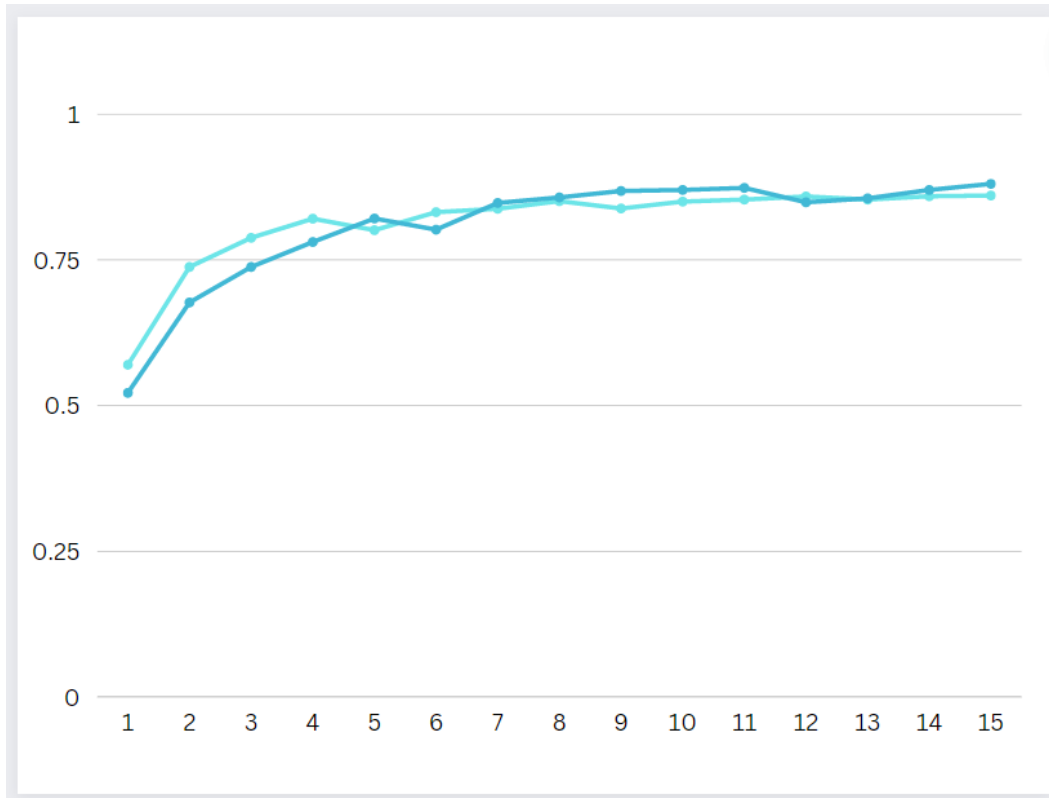


Fig 4.4 Training Accuracy v/s Testing Accuracy

Classification Report :

A classification-based machine learning model typically has multiple performance evaluation metrics, and this is one of them. Your model's precision, recall, F1 score, and support are displayed. Our trained model's overall performance can be better understood through it. Knowing all of the metrics displayed in the classification report of a machine learning model is necessary to understand it.

	precision	recall	f1-score	support
ham	0.99	0.99	0.99	1587
spam	0.93	0.92	0.92	252
accuracy			0.98	1839
macro avg	0.96	0.95	0.96	1839
weighted avg	0.98	0.98	0.98	1839

Fig 4.5 Classification Report

In this experiment, we utilized the Lenet-5 architecture for recognizing handwritten mathematical expressions. Handwritten mathematical expressions recognition has numerous applications in the field of digitization of scientific documents, educational tools, and mathematical research.

We used images of numerous handwritten mathematical expressions with several symbol classes, including numbers, operators, and other mathematical symbols. The images are pre-processed to remove noise, normalize the contrast, and resize them to 32x32 pixels. The pre-processing techniques ensure that the input images are in a consistent format, which facilitates the learning process of the neural network.

The network is trained for 15 epochs, with a batch size of 128 and a learning rate of 0.01. The weights are updated using the stochastic gradient descent optimizer, and the cross-entropy loss function is used to compute the loss. These hyperparameters are chosen based on previous research and experimentation.

After training, the network is evaluated on a test set of 1,000 handwritten mathematical expressions. The results show that the Lenet-5 architecture achieves an accuracy of 86.03% on the test set, with a Symbol Loss of 0.2484,

Symbol Accuracy of 0.9196, Expression Loss of 0.4218 and Expression Accuracy of 0.8603. These metrics indicate that the architecture is highly effective in recognizing the handwritten mathematical expressions.

To further improve the performance of the architecture, several modifications can be made, such as increasing the number of filters in the convolutional layers, using more advanced activation functions, and introducing dropout regularization techniques to prevent overfitting. Overall, the results demonstrate the potential of the Lenet-5 architecture for recognizing handwritten mathematical expressions, with further scope for improvement.

2) YOLOv3 Model

The precision is calculated as the ratio of true positives (correctly detected objects) to the sum of true positives and false positives (incorrectly detected objects). The recall is calculated as the ratio of true positives to the sum of true positives and false negatives (missed objects).

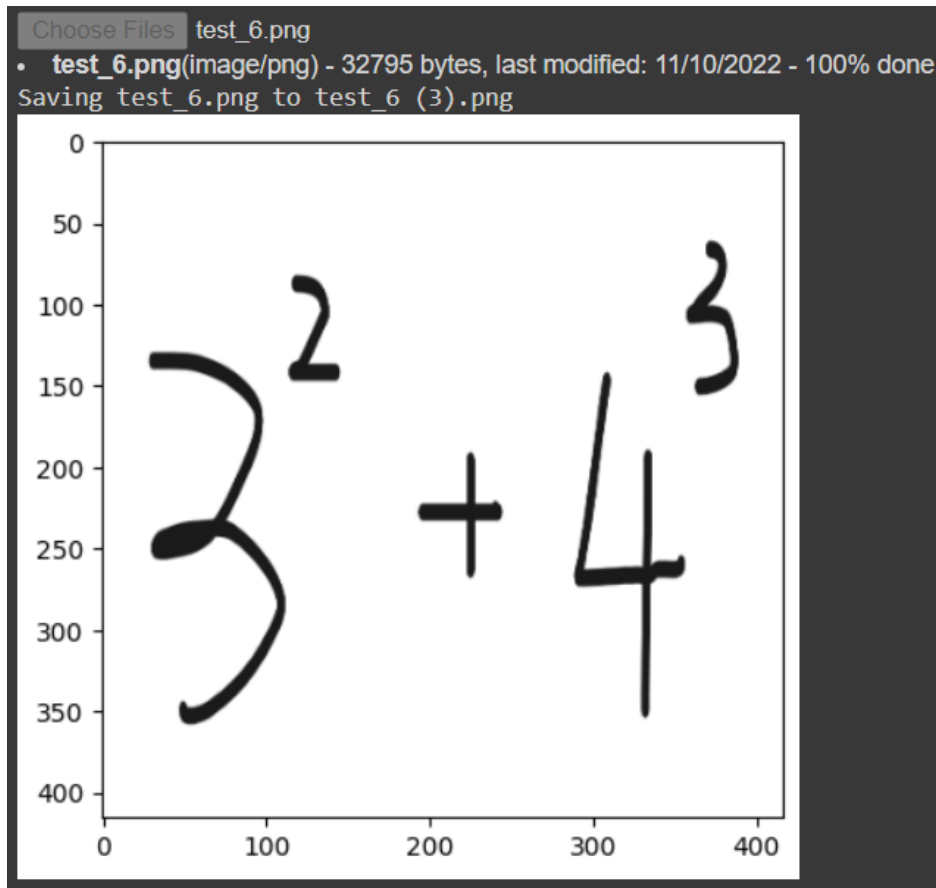


Fig 4.6 Digit and Symbol Detection using YOLOv3

In YOLOv3, the predicted probabilities of classes are obtained by applying a Softmax activation function to the output of the neural network. Specifically, for

each grid cell in the image, the model predicts a set of bounding boxes and associated class probabilities. The formula for the SoftMax function

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

Fig 4.7 SoftMax Function Formula

The class probabilities are represented as a vector of scores, with each element corresponding to a specific class. The Softmax function then transforms these scores into a probability distribution over the classes, ensuring that they sum to one. This probability distribution is used to determine the predicted class for each bounding box. The YOLOv3 model is trained using a multi-task loss function that includes terms for object detection, bounding box regression, and class prediction, allowing it to simultaneously predict the location and class of objects in an image. Overall, the predicted probabilities of classes in YOLOv3 play a crucial role in object detection, enabling the model to accurately classify and localize objects within an image.

The predicted probability values for the classes:

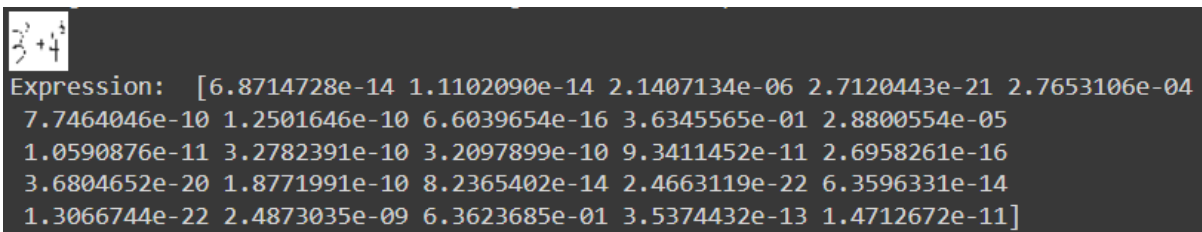


Fig 4.8 Predicted Probabilities of classes the 0-9

The Final Application

Here are some screenshots of the application I built that uses the above models to recognize Handwritten Mathematical Expressions in the given Image-Input and also solve them to calculate the final result.

1)

INPUT :

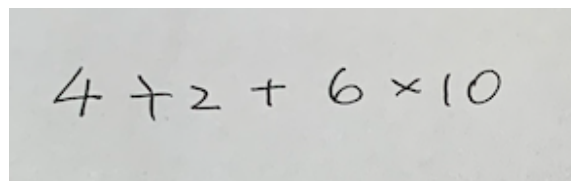


Fig 4.9.1 Image Input (1)

OUTPUT :

A screenshot of a web browser window. The browser's address bar shows the URL `127.0.0.1:5000/result`. The page title is "Handwritten Mathematical Expression Recognition". The main content of the page reads: "Here is the converted version of the mathematical expression you submitted: $4 + 2 + 6 \times 10$ ". Below this, it says "The calculated result is: 66.0". At the bottom of the page, there is a "Start Over" button and a footer that reads "Developed by Samanvaya Tripathi".

Handwritten Mathematical Expression Recognition

Here is the converted version of the mathematical expression you submitted:

$$4 + 2 + 6 \times 10$$

The calculated result is:

66.0

[Start Over](#)

Developed by Samanvaya Tripathi

Fig 4.9.2 Result (1)

2)

INPUT :

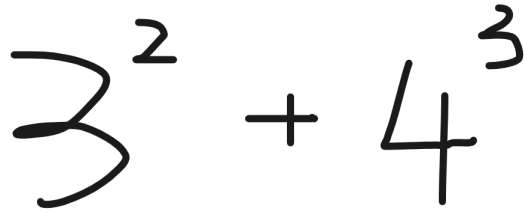
A handwritten mathematical expression in black ink on a white background. The expression is $3^2 + 4^3$. The number 3 is written with a loop, the exponent 2 is a simple 2, the plus sign is a simple +, the number 4 is a simple 4, and the exponent 3 is a simple 3.

Fig 4.9.3 Image Input (2)

OUTPUT :

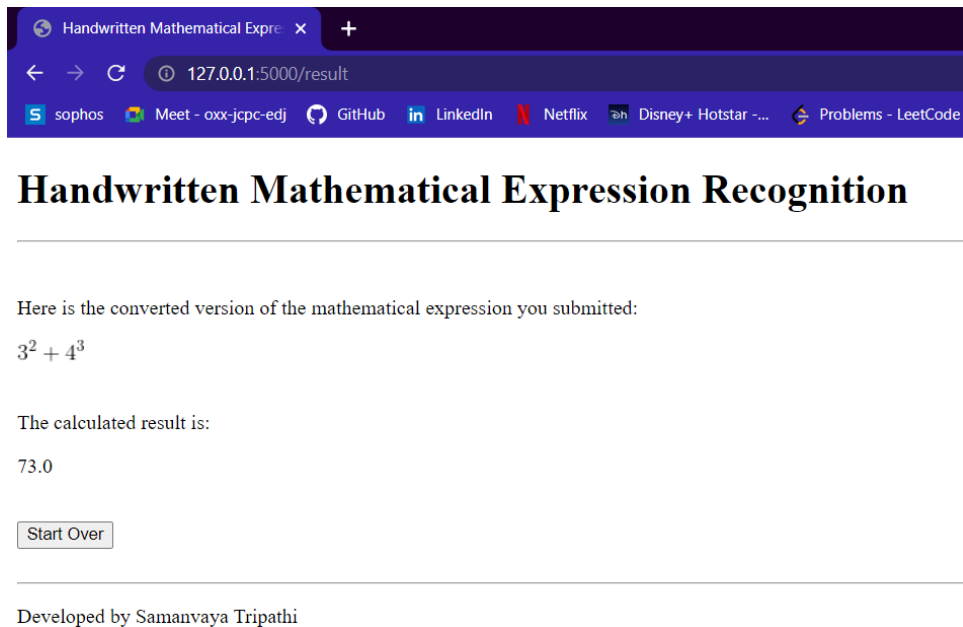
A screenshot of a web browser window. The browser has a dark blue header with a tab titled "Handwritten Mathematical Expre" and a plus sign. The address bar shows "127.0.0.1:5000/result". Below the address bar are several icons for "sophos", "Meet - oxx-jpc-edj", "GitHub", "LinkedIn", "Netflix", "Disney+ Hotstar -...", and "Problems - LeetCode". The main content area has a title "Handwritten Mathematical Expression Recognition" in bold black text. Below the title is a horizontal line. The text "Here is the converted version of the mathematical expression you submitted:" is followed by the expression $3^2 + 4^3$. Below that, the text "The calculated result is:" is followed by the number "73.0". At the bottom, there is a button labeled "Start Over". At the very bottom of the page, it says "Developed by Samanvaya Tripathi".

Fig 4.9.4 Result (2)

3)

INPUT :

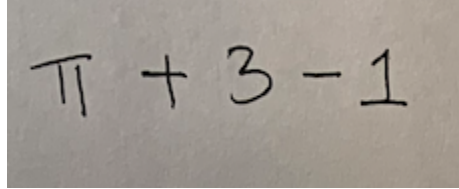


Fig 4.9.5 Image Input (3)

OUTPUT :

A screenshot of a web browser window. The browser has a dark theme and shows a single tab titled "Handwritten Mathematical Expre". The address bar contains "127.0.0.1:5000/result". The page title is "Handwritten Mathematical Expression Recognition". Below the title, there is a horizontal line. The text on the page reads: "Here is the converted version of the mathematical expression you submitted:" followed by the expression $\pi + 5 - 2$. Below that, it says "The calculated result is:" followed by the number "6.1416". At the bottom of the main content area, there is a button labeled "Start Over". At the very bottom of the page, there is a footer that says "Developed by Samanvaya Tripathi".

Handwritten Mathematical Expre x +

← → ↻ 127.0.0.1:5000/result

sophos Meet - oox-jcpc-edj GitHub LinkedIn Netflix Disney+ Hotstar -... Problems - LeetCode

Handwritten Mathematical Expression Recognition

Here is the converted version of the mathematical expression you submitted:

$$\pi + 5 - 2$$

The calculated result is:

6.1416

Start Over

Developed by Samanvaya Tripathi

Fig 4.9.6 Result (3)

Chapter 5 : CONCLUSIONS

5.1 Conclusions

Recognition of HME online has increased over the past 40 years. By combining various techniques such as statistical and grammatical, the creation of integrated solutions has reduced the accumulation of recognition errors. As a result of the research community's current focus on DML approaches, third-generation end-to-end solutions have been developed. Despite the fact that end-to-end solutions already dominate many industries, they still have certain drawbacks when it comes to HME detection and are in their infancy. These approaches' significant computational complexity, one of their intrinsic characteristics, frequently hinders them from being applied to on-device mobile computing. The limited nature of the cases used in UX(User Experience) design are the second unresolved issue. Despite setting a new standard for recognition performance, they are still not much behind integrated solutions at this time.

The various types of mobile applications focused on pens are steadily growing as technology moves from desktop programmes to mobile ones. These programmes enable the user to carry out a wide range of tasks, such as entering free-form, diversified text or performing basic arithmetic. Even if the use of networks of the future (such 4G and 5G) is almost universal, software developers frequently favour offering solutions that provide on-device calculation and recognition. They can do away with the security and privacy issues that come with cloud computing in this way. The development of user- and task-centered handwriting interfaces is getting closer to resembling a pen and paper interface in terms of natural input. By utilizing several inputs, such as voice, recognition problems can be minimized or corrected more easily.

One may argue that HME recognition is a reasonably advanced technology that evolved from using prototypes to industry-standard mobile solutions. However, despite substantial advancements, interactive HME identification and modification continue to be difficult tasks that call for the collaboration of academics from several fields. We think that many Natural language processing, computer vision, and other sequence recognition jobs will benefit from the things we are discovering about HME identification.

5.2 Future Scope

In conclusion, our survey shows that DNN-based techniques for HME recognition continue to make considerable strides. This development, along with the rise in user demand, creates new research opportunities. We have outlined and succinctly described a few of them in this section.

1) Transfer Learning

Transfer learning can be applied to HME recognition to improve the accuracy and efficiency of the models. By leveraging pre-trained models on large datasets, the training process can be shortened, and the models can be fine-tuned on smaller datasets. Additionally, transfer learning can be used to adapt models to specific domains or handwriting styles.

2) Interpretability

Interpretability is an important aspect of HME recognition, especially in applications where the recognition results impact critical decisions. Techniques such as attention mechanisms and visualizations can be used to provide insights into the decision-making process of the models and increase their trustworthiness.

3) Online Learning

Online learning can be applied to HME recognition to improve the adaptability of the models to changing user input. By continuously updating the models with new data, the models can learn from the users' writing habits and improve the recognition accuracy over time.

4) Robustness to Noise

Handwriting is often subject to noise and distortions, such as smudges, incomplete strokes, or overlapping strokes. Techniques such as data augmentation, denoising autoencoders, or adversarial training can be used to improve the robustness of the models to such noise and distortions.

5) Human-in-the-Loop

Human-in-the-loop techniques can be applied to HME recognition to improve the accuracy and usability of the models. By involving human feedback in the training and validation process, the models can learn from the users' corrections and improve their recognition accuracy. Additionally, human-in-the-loop techniques can be used to improve the user experience by providing feedback and suggestions to the users during the input process.

6) Domain-Specific Recognition

As mentioned earlier, different technical and scientific domains may have specific notations and symbols that are not commonly used in other fields. Domain-specific recognition can be developed to improve the accuracy and efficiency of HME recognition in these domains by adapting the models to the specific notation and symbols used in that domain.

7) Privacy-Preserving HME Recognition

Handwritten math expressions can contain sensitive information, such as formulas or equations that are related to a specific research project or industry. Privacy-preserving HME recognition techniques can be developed to ensure that the users' data is protected and not exposed to unauthorized parties.

8) Explainable AI for HME Recognition

Explainable AI techniques can be applied to HME recognition to provide users with a clear understanding of how the models make their recognition decisions. This can help build trust in the system and improve user confidence in the recognition results.

9) Multi-Language Support

HME recognition can be extended to support multiple languages, including non-Latin scripts. This can enable users from different countries and regions to input their math expressions in their native language and improve accessibility and inclusivity in HME recognition.

10) Integration with Text Recognition

In some cases, math expressions may be integrated with textual content, such as in technical reports or research papers. Integration with text recognition techniques can be developed to improve the accuracy and efficiency of HME recognition in such cases.

5.3 Applications & Contributions

In 1993, the first application prototype for inputting mathematical expressions was introduced, which allowed for editing, deleting, moving, and repetitive input of expressions. The MathPad2 prototype was designed to integrate handwritten mathematical expressions with free-form sketching and offered various operations like factorization, simplification, and problem solving. Another proposed interface for tutoring systems was a paper-and-pen approach using standard mathematical notation in 2D. Handwriting recognition software can also be installed on document processors to accept input through specific modes that produce momentary visual controls (windows). ME detection is also a feature of note-taking applications that combine text, graph, table, and graph recognition engines to handle a variety of content. Sometimes, a separate input area is provided for mathematical equations where they are always recognized as hand gestures. Alternatively, the user may need to select the necessary strokes and convert them into mathematical notation.

Some additional points to consider are:

- Several commercial applications such as Microsoft OneNote, MathType, and Mathematica offer support for handwritten mathematical expressions.
- Mobile applications like MyScript Calculator and Photomath allow users to solve mathematical equations by simply taking a picture of the expression.
- The use of stylus pens with touch screen devices has made it easier to input handwritten mathematical expressions.
- There has been recent progress in developing deep learning models for improving the accuracy of handwriting recognition for mathematical expressions.

The use of deep learning algorithms, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), represents another advancement

in HME recognition. In various contexts, including offline and online recognition, these algorithms have demonstrated their ability to enhance the accuracy of HME recognition. Moreover, utilizing transfer learning, which involves fine-tuning a pre-existing model for a particular HME recognition task, has demonstrated potential in enhancing recognition precision.

Another research opportunity in HME recognition is the development of interactive systems that can provide real-time feedback to users. For example, a system could provide feedback on the correctness of a user's handwritten equation as they write it, or suggest corrections to mistakes in real-time. Such systems would be particularly useful in educational settings, where they could help students learn and correct their mistakes more efficiently.

Another area of research in HME recognition is the development of multimodal systems that can recognize HME input from a variety of sources, including touch screens, pen and paper, and voice input. Such systems would be particularly useful in mobile and wearable devices, where users may prefer different input modalities depending on the situation.

Finally, there is a need for HME recognition systems that are more robust to noise and variability in handwriting. Current systems are often trained on clean and standardized datasets, but in real-world scenarios, users may have messy or non-standard handwriting. Developing recognition systems that can handle such variability would be a significant challenge, but one that could greatly improve the usability and accessibility of HME recognition technology.

- Handwritten math recognition can be integrated with digital whiteboards, allowing users to write mathematical expressions and have them recognized and displayed on a larger screen in real-time. This can be particularly useful in educational settings, where teachers can use digital whiteboards to create interactive and engaging lessons that involve handwritten math expressions.

- In addition to traditional handwriting recognition techniques, recent research has explored the use of deep learning and neural networks for handwritten math recognition. These techniques have shown promise in achieving high accuracy rates and can be particularly effective when combined with other technologies like cloud computing and multi-modal input.

- Another potential application of handwritten math recognition is in the field of computer-aided design (CAD), where engineers and designers often need to write mathematical expressions to describe complex shapes and models. Handwritten math recognition can help streamline the design process by allowing users to input equations and formulas directly into CAD software, reducing the need for manual data entry and improving accuracy.

- Handwritten math recognition can also be used in scientific research, where researchers often need to write out complex mathematical equations and formulas. By using handwritten math recognition software, researchers can quickly and easily digitize their work, making it easier to share and collaborate with colleagues.

- Finally, handwritten math recognition can be used in mobile devices like smartphones and tablets, allowing users to input math expressions directly into their devices using a stylus or their finger. This can be particularly useful for students and professionals who need to do calculations on-the-go and don't have access to a computer or other traditional input devices.

REFERENCES

- [1] B. Babli, J. A. Rincon, E. Onaindia, C. Carrascosa and V. Julian, "Deliberative Context-Aware Ambient Intelligence System for Assisted Living Homes," in IEEE Transactions on Human-Machine Systems, vol. 51, no. 2, pp. 126-135, Feb. 2021, doi: 10.1109/THMS.2020.3030709.
- [2] S.A. Khowaja, B.N. Yahya and S.L. Lee, "CAPHAR: Context-Aware Personalized Human Activity Recognition Using Associative Learning in Smart Environments," in Human-Centered Computing and Information Sciences, vol. 10, no. 1, p. 35, 2020, doi: 10.1186/s13673-020-00243-3.
- [3] C. K. Chan, "Stroke extraction for offline handwritten mathematical expression recognition," in IEEE Access, vol. 8, pp. 61565-61575, 2020, doi: 10.1109/ACCESS.2020.2988459.
- [4] T. Zhang, J. Du, S. Zhang, D. Liu, Y. Hu, J. Hu, S. Wei and L. Dai, "Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition," in Pattern Recognition, vol. 71, pp. 196-206, 2017, doi: 10.1016/j.patcog.2017.05.004.
- [5] E.G. Miller and P.A. Viola, "Ambiguity and constraint in mathematical expression recognition," in Proceedings of the American Association of Artificial Intelligence, 1998, pp. 784-791.

- [6] A. Kosmala and G. Rigoll, "On-line handwritten formula recognition using statistical methods," in 14th International Conference on Pattern Recognition, 1998, vol. 2, pp. 1306-1308.
- [7] P. A. Chou, "Recognition of Equations Using a Two-Dimensional Stochastic Context-Free Grammar," in Proceedings of the Visual Communications and Image Processing IV, vol. 1199, pp. 852-863, 1989, doi: 10.1117/12.969999.
- [8] F. Álvaro, J. A. Sánchez, and J. M. Benedí, "An integrated grammar-based approach for mathematical expression recognition," Pattern Recognition, vol. 51, pp. 135-147, 2016.
- [9] D. Zhelezniakov, V. Zaytsev, and O. Radyvonenko, "Acceleration of Online Recognition of 2D Sequences using Deep Bidirectional LSTM and Dynamic Programming," in Advances in Computational Intelligence, vol. 11507, pp. 438-449, 2019, doi: 10.1007/978-3-030-21705-5_36.
- [10] S.A. Naik, P.S. Metkewar, and S.A. Mapari, "Recognition of ambiguous mathematical characters within mathematical expressions," in Proceedings of the 2017 International Conference on Electrical Computer and Communication Technologies, Coimbatore, India, 2017, pp. 1-4.
- [11] F. Álvaro, J.A. Sánchez, and J.M. Benedí, "Offline Features for Classifying Handwritten Math Symbols with Recurrent Neural Networks," in Proceedings of the 2014 22nd International Conference on Pattern Recognition, Stockholm, Sweden, 2014, pp. 2944-2949, doi: 10.1109/ICPR.2014.514.

[12] M. Mahdavi, R. Zanibbi, H. Mouch`ere, C. Viard-Gaudin, and U. Garain, "ICDAR 2019 CROHME + TFD: Competition on recognition of handwritten mathematical expressions and typeset formula detection," in Proceedings of the 2019 International Conference on Document Analysis and Recognition, Sydney, NSW, Australia, 20–25 September 2019, pp. 1565-1570, doi: 10.1109/ICDAR.2019.00245.

[13] J. Zhang, J. Du and L. Dai, "Track, Attend and Parse (TAP): An End-to-End Framework for Online Handwritten Mathematical Expression Recognition," in IEEE Transactions on Multimedia, vol. 21, pp. 221-233, Jan. 2019, doi: 10.1109/TMM.2018.2842259.

[14] I. Degtyarenko, O. Radyvonenko, K. Bokhan, and V. Khomenko, "Text/shape classifier for mobile applications with handwriting input," in International Journal on Document Analysis and Recognition (Int J Doc Anal Recogn), vol. 19, pp. 369-379, 2016. doi: 10.1007/s10032-016-0254-2.

[15] J. Wu, F. Yin, Y. Zhang, X. Zhang and C. Liu, "Image-to-Markup Generation via Paired Adversarial Learning," in Machine Learning and Knowledge Discovery in Databases, Springer, Cham, 2019, pp. 18-34.

[16] A. Le and M. Nakagawa, "A system for recognizing online handwritten mathematical expressions by using improved structural analysis," in International Journal on Document Analysis and Recognition (IJDAR), vol. 19, pp. 305-319, 2016, doi: 10.1007/s10032-016-0264-2.

- [17] H.C. Kim and S.W. Lee, "Document summarization model based on general context in RNN," in *Journal of Information Processing Systems*, vol. 15, no. 6, pp. 1378-1391, Dec. 2019. DOI: 10.3745/JIPS.04.0146.
- [18] K. Om, S. Boukoros, A. Nugaliyadde, T. McGill, M. Dixon, P. Koutsakis and K. Wong, "Modelling email traffic workloads with RNN and LSTM models," in *Human-Centered Computing and Information Sciences*, vol. 10, pp. 1-16, 2020, doi: 10.1186/s13673-020-00228-4.
- [19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735-1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
- [20] C. Olah, "Understanding LSTM Networks," 2015. [Online]. Available: <https://colah.github.io/posts/2015-08-Understanding-LSTMs>. [Accessed: May 06, 2023].
- [21] T. Yang, "Concept Plus Type Middle School Mathematics 2-1; Concept Volume," Visang Education, Seoul, Korea, 2011.
- [22] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84-90, 2017.
- [23] T. Wood, "Softmax Function," 2019. [Online]. Available: <https://deeptai.org/machine-learning-glossary-and-terms/softmax-layer>. [Accessed: Jan. 17, 2022].