Electronic Letters on Computer Vision and Image Analysis 23(2):1-28, 2024

Deep Learning based-framework for Math Formulas Understanding

Kawther Khazri Ayeb[1], Afef Kacem Echi * and Takwa Ben Aïcha Gader [1]

[1] *University of Tunis, ENSIT-LaTICE, 05 Avenue Taha Hussein, Tunis, Tunisia*

Received 19th of January, 2024; accepted 7th of August 2024

Abstract

Extracting math formulas from images of scientific documents and converting them into structured data for storing in the database is the premise of their further utilization. As math formulas are hard to extract and recognize automatically, rapidly, and effectively, we proposed a deep learning-based system. Firstly, this system exploits the pre-trained YOLOv8 model and fine-tunes it using the mathematical formula dataset and specific combination features to detect and classify the formula inside and outside the text. Once extracted, we developed a robust end-to-end math formula recognition system. This system automatically identifies and classifies math symbols using Faster R-CNN for object detection. It then employs a Convolutional Graphical Neural Network (ConvGNN) to analyze the layout of the math formula. This approach is effective because the formula can be better represented as a graph that captures complex relationships and dependencies among objects. ConvGNN can predict formula linkages without resorting to laborious feature engineering. Experimental results on the IBEM and CROHME 2019 datasets reveal that the proposed approach can accurately extract isolated formula with mAP of 99.3%, embedded formulas with mAP of 80.3%, detect symbols with mAP of 87.3%, and analyze formula layout with an accuracy of 92%. We also showed that our system is competitive with related work.

Key Words: Formula extraction and recognition, Symbol detection and classification, formula layout analysis, YOLOv8, Faster R-CNN, ConvGNN.

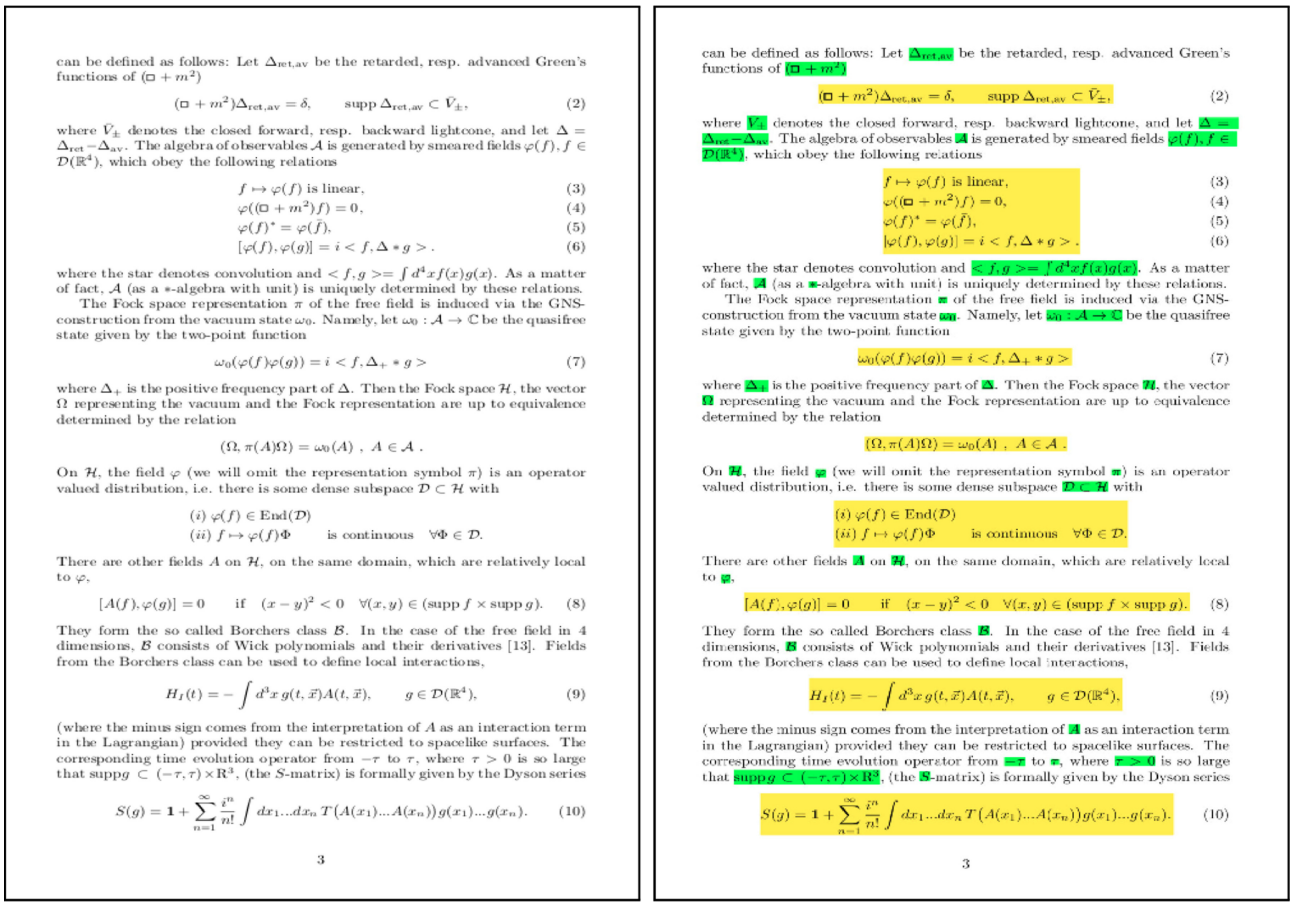
1 Introduction

Over the past twenty years, there has been an increased focus on Automatic Technical Documents Processing and Understanding (TDPU) due to its significant practical uses. TDPU builds upon previous advancements in OCR, natural language understanding, pattern recognition, and image understanding. Within the field of TDPU, there is a specific sub-area dedicated to understanding math formulas. This area focuses on detecting math formulas within documents and utilizing parsing methods to understand the formulas. More precisely, to understand mathematical formulas in documents, the process is divided into four sub-processes. First, identification and segmentation focus on detecting and isolating formulas in documents. Secondly, symbol recognition is utilized in formulas. Thirdly, layout recognition is used to identify the spatial relationships among symbols. Lastly, content representation and analysis aim to compute the outcome of the math formulas.

Correspondence to: <kawther.khazri1984@gmail.com>

Recommended for acceptance by Angel <D. Sappa>
https://doi.org/10.5565/rev/elcvia.1833
ELCVIA ISSN:1577-5097

As shown in Figure 1, formulas have a precise syntax requiring accurate content knowledge and a secure spatial delimitation. Mathematical formulas are frequently integrated within regular text in images of scientific documents. They can vary from numbers and variable names.



(a) Original page. (b) Highlighted bounding boxes.

Figure 1: An example of processing a page of the IBEM dataset. Embedded math formulas are in green, and isolated ones are in yellow. [13]

It is worth noting that while significant progress has been made in recent years, math formula extraction from images and their recognition are still active research areas, and there may not be a perfect solution for all scenarios. Each document's unique characteristics may require customized approaches and techniques for accurate extraction and recognition. Here are some common problems encountered during the process:

- Image Quality: The quality of the image itself can greatly impact the accuracy of math formula extraction. Blurriness, low resolution, noise, or uneven lighting can make it challenging for optical character recognition (OCR) algorithms to interpret the symbols and structure of the formulas correctly.
- Complex Notation: Mathematical notation can be highly complex and varied, including symbols, superscripts, subscripts, fractions, integrals, matrices, and other mathematical constructs. Dealing with such intricate structures requires sophisticated algorithms capable of accurately understanding and parsing the different elements.
- Symbol Recognition: Mathematical symbols can have different fonts, styles, and variations, making symbol recognition a difficult task. OCR algorithms might struggle with accurately identifying less common symbols or symbols with ambiguous representations.

- Contextual Understanding: Math formulas often rely on the context provided by surrounding text or equations to infer their meaning correctly. Extracting formulas in isolation without considering the surrounding text can lead to misinterpretation or incorrect parsing.
- Layout and Alignment: The layout and alignment of math formulas within the document can vary, such as inline formulas, displayed formulas, or formulas within tables or captions. Handling these different layouts and aligning the extracted formulas properly can be challenging.
- Handwritten Formulas: In some cases, scientific papers may include handwritten formulas, which introduce additional complexity. Recognizing and accurately transcribing handwritten symbols and formulas can be significantly more challenging than dealing with printed text.

Addressing these challenges requires combining techniques, including image preprocessing, OCR algorithms specialized in math formula recognition, context-aware parsing, and post-processing steps to refine the extracted formulas. Additionally, training data that specifically includes a wide range of math notation and symbol variations can improve the performance of the extraction process.

Traditional rule-based techniques for math formula extraction do not scale well across a wide range of formula types. They could be more robust for expressions with slight typos and lexical ambiguities. This work employs deep learning classifiers to identify math formulas in a given document image. We used the YOLOv8 for object detection, image classification, and instance segmentation tasks. While YOLOv8 is a general-purpose object detection model, it can be advantageous for math formula detection due to the following reasons:

- Real-time Detection: YOLOv8 is known for its fast inference speed, making it suitable for real-time or near real-time applications. This can benefit math formula detection, especially when quick and efficient processing is required.
- Simplicity and Efficiency: YOLOv8 follows a single-stage detection approach, directly predicting bounding boxes and class probabilities in a single pass through the network. This simplicity makes YOLOv8 computationally efficient compared to other multi-stage detectors, making it easier to deploy on various devices.
- Handling Small Objects: Math formulas can consist of small symbols or subscripts, which can be challenging to detect accurately. YOLOv8 is designed to handle objects of different sizes, including small objects, by using anchor boxes of various scales and aspect ratios. This capability can help improve the detection performance of small math formula components.
- High Localization Accuracy: YOLOv8 employs anchor-based detection, which helps improve the localization accuracy of detected objects. Accurate localization is crucial for math formulas, enabling precise extraction of symbols, subscripts, and superscripts.
- Flexibility and adaptability: YOLOv8 can be customized and fine-tuned for specific tasks and domains. By training YOLOv8 on a custom dataset of math formulas, it can learn to detect formulas with higher accuracy and adapt to specific formula detection requirements.
- Availability of pre-trained models: There are pre-trained versions of YOLOv8 available, trained on large-scale datasets like COCO or ImageNet. These pre-trained models can be a starting point for math formula detection tasks, allowing for transfer learning and faster convergence during training.

Once the formulas are extracted, we develop a reliable system for recognizing math symbols. We use faster R-CNN object detection technology to achieve this. Additionally, we implement a convolutional graphic neural network (ConvGNN) to analyze the layout of mathematical formulas accurately. This approach effectively represents the formula as a graph with complicated relationships and interdependencies. ConvGNN eliminates the need for time-consuming feature engineering and can accurately predict formula linkages.

The remainder of the paper is organized as follows: Section 2 includes a review of related works. Section 3 describes the proposed systems for formula and symbol detection and classification and for formula layout analysis using deep learning models. After that, experimental findings are discussed in section 4. Finally, in section 5, some conclusions and prospects are given.

2 Related Works

Math recognition systems currently employ various methods for detection and recognition. According to [1], there are three categories of formula detection methods: character-based, image-based, and layout-based. OCR engines are used in character-based methods. Any characters not recognized by the engine are considered potential math expression elements. Image-based methods use image segmentation, and layout-based detection uses features such as line height, line spacing, and alignment from typesetting information, possibly along with visual features. Similarly, various layout-based methods are employed to parse math formulas, including syntax-based approaches, graph search approaches, and image-based RNNs, which generate LATEX strings as output. Next, we will summarize the contributions and limitations of some existing math formula detection and recognition systems.

In the paper cited as [2], the INFTY system was proposed that utilizes two recognition engines working in tandem to achieve simultaneous character recognition and math-text separation. One engine is a non-specialized OCR engine for math documents, while the other is a symbol recognition engine. After recognition, the system performs structural analysis of the math expressions by identifying the minimum cost spanning trees in a weighted digraph representation. The system achieves precise recognition results through a formula structure recognition approach based on graph search.

The technique of utilizing symbol data from PDFs instead of relying on OCR to analyze images was initially introduced by a research study conducted by [3]. They employed a pattern recognition approach to recognize formulas from PDF files through expression grammar. The system has the advantage of being faster than conventional rendering and analysis of document images and has better precision by utilizing PDF character information. In a subsequent study [4], the authors successfully reconstructed fonts not originally embedded in a PDF file. They accomplished this by mapping unicode values to standard character codes when applicable. Additionally, they utilized connected components analysis to pinpoint characters with matching style and spacing from a grouping provided by pdf2html. This enabled them to obtain precise bounding boxes.

Zhang et al. [5] adopt a dual extraction technique, drawing inspiration from [3], for PDF character extraction. They rely on a PDF parser and an OCR engine to complement PDF symbol extraction. The approach involves recursively segmenting and reconstructing the formula using symbols detected on the main baseline, enabling an in-depth analysis of the formula structure. The recognition rate in INFTYReader [2] was improved by the authors of [7]. They achieved this by utilizing the extracted PDF character information from PDFMiner. In certain PDFs, certain characters are made up of multiple glyphs, like big braces or square roots that usually have a radical symbol connected to a horizontal line. A study by [3] used overlapping bounding boxes to identify these complex characters in current PDFs that use Type 1 fonts.

Deng et al. [8] introduced the concept of image-based detection with RNN-based recognition inspired by RNN-based image captioning techniques. In a recent study by Phong et al. [14], they utilized a YOLO v3 network based on a Darknet-53 network with 53 convolutional layers for feature extraction and detection. To improve recognition, they employed an advanced end-to-end neural network called Watch, Attend, and Parse (WAP). The system utilizes a GRU with attention-based mechanisms in its parser, which can lead to slow processing times due to pixel-wise computations. Diagnosing errors in recurrent image-based models is difficult due to the absence of a direct correlation between input image regions and the resulting LATEX output strings. The proposed system has been tested on the Marmot public dataset. The obtained accuracies of detecting isolated and inline expressions are 93% and 73%, respectively. Meanwhile, accuracies of the recognition for isolated and detected expressions are 51.77% and 45.50%, respectively.

In [10], the authors presented algorithms that can detect and recognize mathematical expressions using a unified system. They introduced an enhanced PDF symbol extractor, SymbolScraper, to accurately identify symbol locations. Additionally, they developed a new Scanning Single Shot Detector, named ScanSSD, that uses visual features to identify mathematical formulas. The ScanSSD was created by modifying the Single Shot Detector (SSD) [11] to work efficiently with large document images. To recognize expression structure, they utilized the Query-driven Global Graph Attention (QD-GGA) model [12]. This model uses CNN-based features with attention to extract formula structure, similar to [2]. However, unlike [2], the QD-GGA model trains the features and attention modules concurrently in a feed-forward pass for multiple tasks, including symbol classification, edge classification, and segmentation. This results in faster training and execution for the system.

A new system proposed by Phong (2022) [21] involves transforming binary document images into grey images using the distance transform method. This process enhances the contrast between the expressions and the background, improving the accuracy of expression detection. The transformed images are then inputted into the Faster RCNN for formula detection. To further enhance accuracy, the anchor boxes of the RPN are optimized for generation. The system also detects isolated expressions within the transformed images.

3 Proposed System

In this work, we proposed systems that differ from previous works. Here are some important distinctions:

- Our system focuses on formula detection, classification, and recognition and its evaluation.
- Grammar and manual segmentation are not required.
- Symbol and formula detection is based on the Faster RCNN and the YOLOv8 object detection deep learning models, respectively.
- Formula layout analysis is based on convGNN, with outputs generated more quickly and easily diagnosed errors than RNN models.
- Structure recognition errors can be directly observed in graphs grounded in input image regions.

3.1 Proposed System for Formula Detection and Classification

YOLOv8 is an improved version of the popular You Only Look Once (YOLO) object detection model. While YOLOv8 refers to several different implementations and variations, the architecture generally follows the YOLO design principles. Here's an overview of the YOLOv8 architecture:

- Input and Preprocessing: The model takes an input image divided into a fixed grid of cells. The input image is typically resized to a predetermined size suitable for processing.
- Backbone Network: It consists of multiple convolutional layers, residual blocks, and pooling operations. The purpose of the backbone network is to extract high-level features from the input image.
- Neck: It is an intermediate component inserted between the backbone and the detection head. It usually includes additional convolutional layers and feature fusion operations. The neck helps combine features of different scales and enhances the model's ability to detect objects of various sizes.
- Detection Head: It is responsible for generating the final predictions for object detection. It typically consists of a series of convolutional layers, which progressively refine the feature representations.The detection head predicts bounding box coordinates, objectness scores (to determine if an object is present), and class probabilities for different object categories. YOLOv8 often employs anchor-based detection, where anchor boxes of predefined sizes and aspect ratios facilitate localization and classification.

- Output: It is a set of predicted bounding boxes, along with their associated class labels and confidence scores. Non-maximum suppression (NMS) is commonly applied to remove duplicate or overlapping detections, keeping only the most confident ones.

While YOLOv8 is primarily designed for general object detection tasks, including detecting various objects in images, it can potentially be adapted for mathematical formula detection and classification as well. To use it, we needed a custom dataset that included annotated images of mathematical formulas. The annotations define the bounding boxes around the formulas to train the model. Each bounding box annotation contains the coordinates and label of the formula. Here's a general outline of the steps we followed to adapt YOLOv8 for mathematical formula detection:

- Dataset Preparation: We used the IBEM: a dataset of images containing mathematical formulas annotated with bounding box coordinates and labels. The labels represent the mathematical formula's class, such as an "embedded" or an "isolated" formula. It is important to note that creating a high-quality annotated dataset for math formulas or expressions can be time-consuming and requires expertise.

- Model Selection: We used the annotated dataset to train the YOLOv8 model. YOLOv8 uses a deep convolutional neural network architecture trained on the large-scale dataset ImageNet. However, in this case, we needed to fine-tune the model on the used specialized formula dataset.

- Fine-tuning and Evaluation: We performed iterations of training and evaluation to improve the model's performance. We used appropriate evaluation metrics to assess the model's accuracy and adjusted the training parameters as needed. Training YOLOv8 for mathematical formula detection might require a significant amount of labeled data and computational resources. Additionally, the accuracy of the model depends on the quality and diversity of the training dataset.

- Testing and Inference: Once the model is trained, we used it to detect mathematical formulas in new images. We provided the image as input to the trained YOLOv8 model and processed the output bounding box predictions to extract the detected formulas. Note that the performance of the object detection model depends on factors like image quality, variation in mathematical notations, and the complexity of the formulas or expressions.

3.2 Proposed System for Math Formula Recognition

We propose splitting the off-line handwritten math formula recognition problem into two subsequent tasks. The first challenge is recognizing math symbols in images and determining their bounding boxes. The second task uses a graph-based technique to identify the formula's structure.

3.2.1 Math Symbol Detection and Classification

As input, our system receives a scanned image of a formula and a model that ensures detection and classification. Faster R-CNN, a pre-trained object detection model, was trained using the CROHME 2019 dataset to build this model. The bounding boxes of classified symbols are then computed.

Deep Convolutional Neural Networks have improved the accuracy of current object-detecting systems dramatically. According to [20], among the different approaches, faster R-CNN, a supervised learning algorithm, has provided state-of-the-art accuracy and efficiency, which justifies our pick of this model. After receiving an image, the algorithm outputs a list of object-bounding box coordinates and the related object class for each box. It begins by importing the Faster R-CNN Inception Resnet model to be trained, followed by the validation and training data sets.

In Figure 2, we see that Faster R-CNN is best understood as a three-part neural network consisting of a feature extractor, a Region Proposal Network (RPN), and a region classifier.

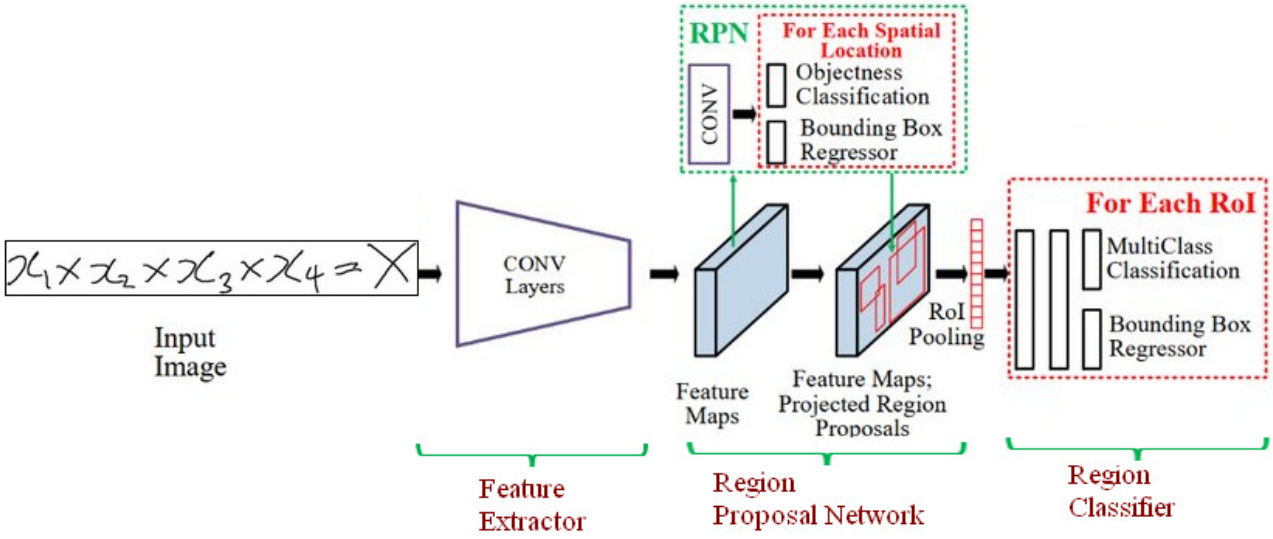


Figure 2: Architecture of Faster R-CNN.

- Feature Extractor: It is often a Deep CNN (DCNN) with no fully connected layers. It takes an input image and outputs a feature map. The feature extractor accepts images of varying widths and heights but performs a preprocessing phase in which the images are scaled to the same minimum dimension (M). The type of feature extractors used has a significant impact on the network's accuracy and computational cost. The larger the depth of the network, the longer the inference time and the higher the accuracy.

- Region Proposal Network: It is a two-layer, fully convolutional neural network. One is the box regression layer, and the other is the classification layer. It begins by running the image through CNN to generate a feature map. The algorithm then creates anchor boxes representing each object class with specific height and width based on the objects in the training dataset. Anchor boxes are then tiled across the image. Predictions for each anchor box are calculated using the previously generated convolutional feature map. The regression layer computes four values describing a bounding box relative to the anchor. The classification layer generates two values representing the probability of the bounding box containing an object. It is a full CNN with two layers. The box regression layer is one, and the classification layer is another. The image is first processed through CNN to build a feature map. Based on the object classes in the training dataset, the algorithm then generates anchor boxes with specific heights and widths for each object class. The image is then tiled with anchor boxes. The previously produced convolutional feature map calculates predictions for each anchor box. The regression layer calculates four values that describe a bounding box concerning the anchor. The classification layer makes two values representing the probability of the bounding box containing an object.

- Region Classifier: The proposals of the RPN crop the respective regions from the features map. The object's class is then determined by a small neural network classifier, which then refines the box using the cropped regions as input. This network, like the RPN, optimizes using a softmax and regression loss.

Using the faster R-CNN model that has already been trained, math symbols are identified and classified (see Figure 3). The math formula layout analyzer then uses the vectorized class and placement information for each symbol as its input.

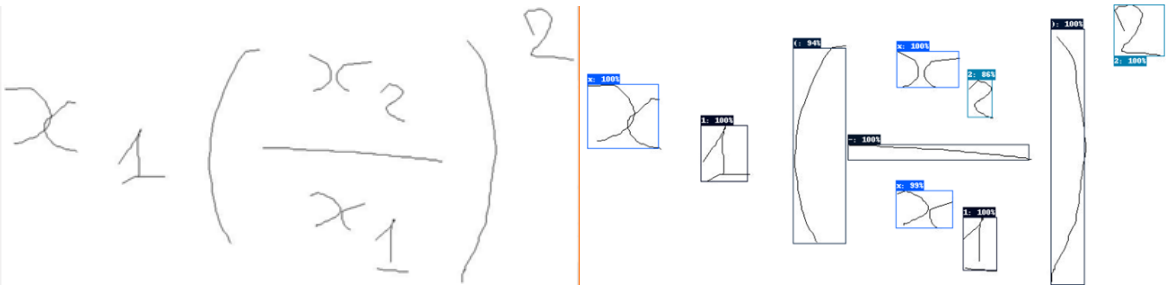


Figure 3: Math symbol detection and classification output example.

3.2.2 Math Formula Layout Analysis

As announced before math formula structure analysis is implemented based on ConvGNN. ConvGNN is a popular example of Graph Neural Network (GNN), a neural network that operates on graph data structures. ConvGNN are comparable to traditional CNN programs. Just as an image is a particular case of graphs in which neighboring pixels are connected. In CNN, and similarly to a graph, each image pixel is considered a node, with neighbors determined by the filter size. The CNN computes the weighted average of the red node's pixel values and those of its neighbors. A node's neighbors are ordered and have a fixed size (see Figure 4).

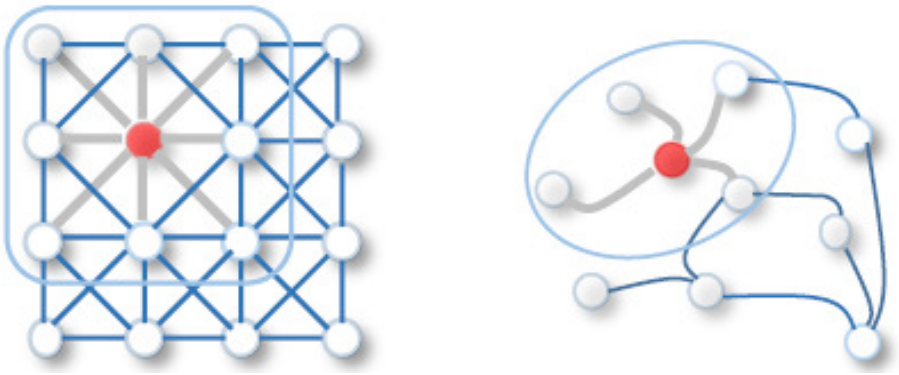


Figure 4: On the left, we have image convolution in CNN, while on the right, we have graph convolution in ConvGNN.

Like CNN, the ConvGNN calculates the weighted average of a node's neighborhood information. For instance, to obtain a hidden representation of the red node, the average value is computed by combining the node features of the red node and its neighbors. Unlike image data, the neighbors of a node are not arranged in a specific order and can vary in size (refer to Figure 4). Additionally, nodes can be connected by different links, making it challenging to apply convolutions to graphs. In the following paragraphs, we provide a brief definition of ConvGNN, explain how information is propagated through its hidden layers, and clarify how ConvGNN combines information from previous layers to create useful feature representations of nodes in graphs.

Remember that ConvGNNs are used for processing graphs and extending image convolution to graph data. The idea is to create a node representation by combining its features with its neighbors, to extract high-level node representations. To analyze a graph, either $G = (V, E)$, where V is the set of vertices and E is the set of edges, or $G = (A, X)$, the input required is :

- An input feature matrix X of size $N \times F^0$, where N is the number of nodes and F^0 is the number of input features for each node.
- An adjacency matrix A of size $N \times N$, which is a tool for implementing the graph structure representation.

The hidden layer within the ConvGNN can be expressed as follows:

$$H^i = f(H^{i-1}, A) \tag{1}$$

where $H^0 = X$ and f is a propagation rule :

Within this framework, multiple layers are denoted as H^i, corresponding to an $N \times F^i$ feature matrix. Each row in this matrix represents a feature representation of a node. Using the propagation rule f, these features are aggregated at each layer to create the next layer's features. As a result, the features become increasingly abstract with each successive layer. The only variation between different ConvGNN variants is the propagation rule f used. Initially, we utilized a basic propagation rule-based ConvGNN, and subsequently, we experimented with the spectral rule. There are two types of ConvGCNs:

- Simple ConvGNN: One of the fundamental propagation rules involves the equation below that includes the weight matrix for a given layer, denoted as W^i, and a non-linear activation function, specifically the ReLU function represented as σ. The weight matrix has dimensions of $F^i \times F^{i-1}$, with the second dimension determining the number of features in the upcoming layer.

  $$f(H^i, A) = \sigma(AH^iW^i) \tag{2}$$

  It's important to remember that the straightforward propagation rule presents a node as a sum of its neighbors' feature representations. However, there are a couple of significant downsides to this approach. Firstly, the resulting representation of a node doesn't consider its features. Secondly, nodes with high degrees will have values that overshadow their feature representation, while nodes with low degrees will have limited values. This imbalance can cause problems with exploding gradients and make it challenging to train using algorithms like stochastic gradient descent, which require careful feature scaling.

- Spectral-based ConvGNN: In this paper, we offer a theoretical explanation for a particular neural network model based on graphs. Throughout the rest of the paper, we will use this model. Specifically, we analyze a ConvGNN that operates with the following layer-wise propagation rule:

  $$f(H^i, A) = \sigma(D^{-0.5}\hat{A}D^{-0.5}H^iW) \tag{3}$$

  where

  - The degree matrix, denoted as D, is a diagonal matrix that holds information about the degree of each node in a graph. The degree of a node is the number of edges connected to it. Together with the adjacency matrix, the degree matrix is used to create the Laplacian matrix of the graph.
  - The adjacency matrix $\hat{A}$ includes self-connections (represented by the identity matrix, I) to consider the features of the node itself rather than just its neighbors' attributes. This improvement overcomes the limitation of the basic ConvGNN.
  - W: the weight matrix.
  - σ : the ReLU activation function.

  When comparing the spectral rule to simple rule propagation, the only difference lies in the choice of the aggregate function. The computation of the aggregate feature representation of the i^{th} node using the simple rule is as follows:

  $$aggregate(A, X)_i = A_iX = \sum_{j=1}^{N} A_{i,j}X_j \tag{4}$$

The equation demonstrates that the contribution of every neighbor is determined by the neighborhood described by the adjacency matrix A. When using the spectral rule, the feature representation is calculated through the following formula:

$$aggregate(A, X)_i = D^{-0.5} A_i D^{-0.5} X = \sum_{j=1}^{N} D_{i,i}^{-0.5} A_{i,j} D_{j,j}^{-0.5} X_j \tag{5}$$

To explain this propagation rule, we can start with a localized spectral filter on graphs and use a first-order approximation. Just like in the study conducted by [23], we initially utilized a Chebychev polynomial approximation. However, we discovered that using Hermite polynomials yields better outcomes. We define spectral convolutions on graphs as the product of a signal $x \in \mathbf{R^N}$ (which has a scalar value for each node) and a filter $g_\theta = diag(\theta)$ that is Fourier domain parameterized by $\theta \in R^N$. The matrix of eigenvectors of the normalized graph Laplacian is represented by U:

$$g_\theta \times x = U_{g_\theta} U^T x \tag{6}$$

$$L = I_N - D^{-1/2} A D^{-1/2} = U \wedge U^T \tag{7}$$

The function g_θ depends on the eigenvalues of matrix L, denoted by $\wedge$, and the graph Fourier transform of x, represented by $U^T x$. To evaluate this equation, one must multiply with the eigenvector matrix U, a computationally expensive process with time complexity of $O(N^2)$. Additionally, computing the eigen-decomposition of L can be challenging for large graphs. To overcome this issue, Chebyshev polynomials are defined recursively.

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \tag{8}$$

with $T_0(x) = 1$ and $T_1(x) = 2x$. The Hermite polynomials are recursively defined as:

$$T_k(x) = xT_{k-1}(x) - (k-1)T_{k-2}(x) \tag{9}$$

with $T_0(x) = 1$ and $T_1(x) = x$.

In what follows, we will go over ConvGNN's design, along with how it applies to problems involving math formula recognition. The formula is expressed as a graph, as seen in Figure 5, with nodes denoting symbol classes and links denoting their spatial relationship. We added link nodes to ensure link prediction. We then created graph G, a single undirected, unweighted network containing all of the math formulas in the Crohme dataset, with nodes nodes as symbols and their links.

As a part of our efforts to train our math formula analysis system, we have utilized the CROHME data set. This data set contains graph files that provide information about the symbols and their relationships in mathematical formulas. Before analyzing the data, we performed a preprocessing step where we grouped the strokes of the same symbol and removed any unnecessary spatial relationships, especially between distant symbols. You can refer to Figure 6 for a better understanding.

ConvGNN is a method used to create a node's representation by combining its features and those of its neighbors. This allows for the extraction of high-level node representations. In Figure 7, a single graph represents all the formulas in the data set. In this graph, nodes represent symbols and links. Our system takes a Graph $G = (A, X)$ as input where X is a feature matrix of size $N \times F^0$ (with N being the total number of nodes and F^0 being the number of input features for each node), and A is an adjacency matrix of size $N \times N$. The model propagates information through hidden layers, accumulates information from previous layers, and provides graph node feature representations.

The ConvGNN output is a labeled graph where symbols are classified into lexical units, and their relationships are identified.

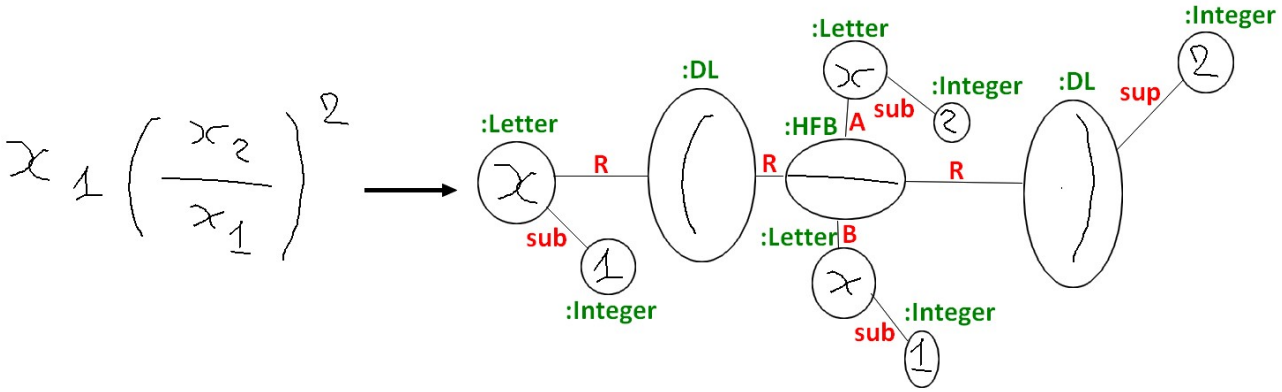


Figure 5: Graph representation of a mathematical formula.

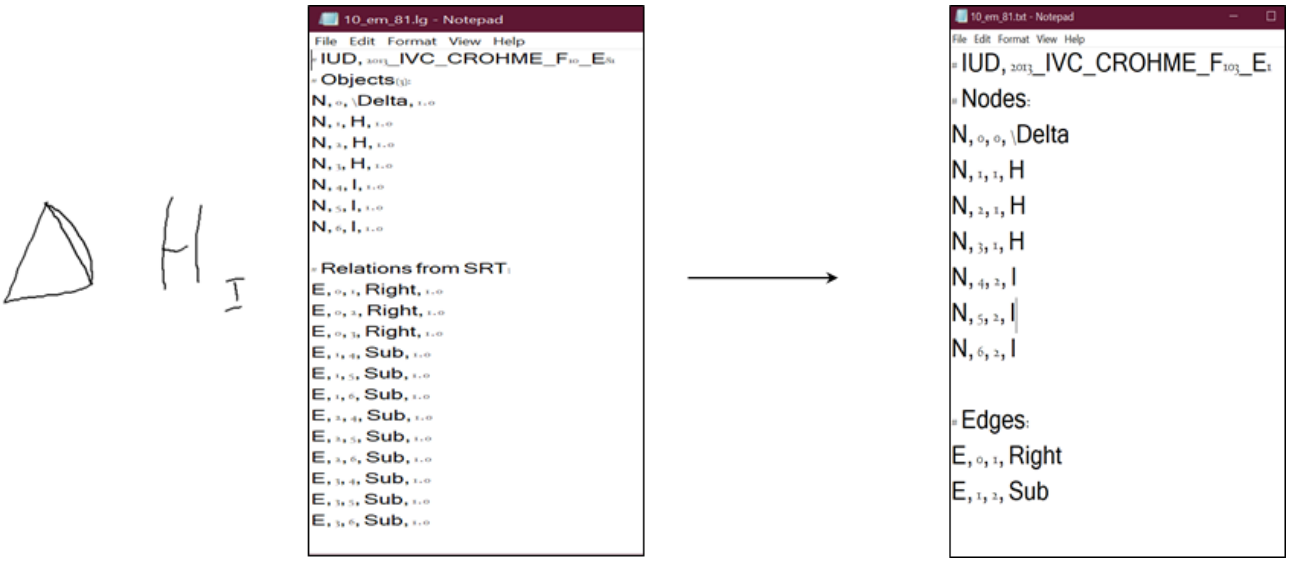


Figure 6: Preprocessing for math formula layout analysis.

4 Experimentation and Obtained Results

4.1 Results from the YOLOv8 for Formula Detection and Classification

We used the IBEM dataset [13], which comprises 600 documents totaling 8,272 pages. It comprises 29603 isolated and 137089 embedded expressions. Table 1 summarizes the characteristics of the IBEM data set.

Table 1: statistics of the IBEM data set.

Total no. of documents	600
Total no. of pages	8272
No. of isolated MEs	29603
No. of embedded MEs	137089

Regarding class distribution, it is evident that the IBEM data set is imbalanced, and embedded expressions are more frequent than isolated ones. This data set is divided into three sets, as specified by the ICDAR 2021 Competition on Mathematical Formula Detection [22]. We worked with the second set, including 670-

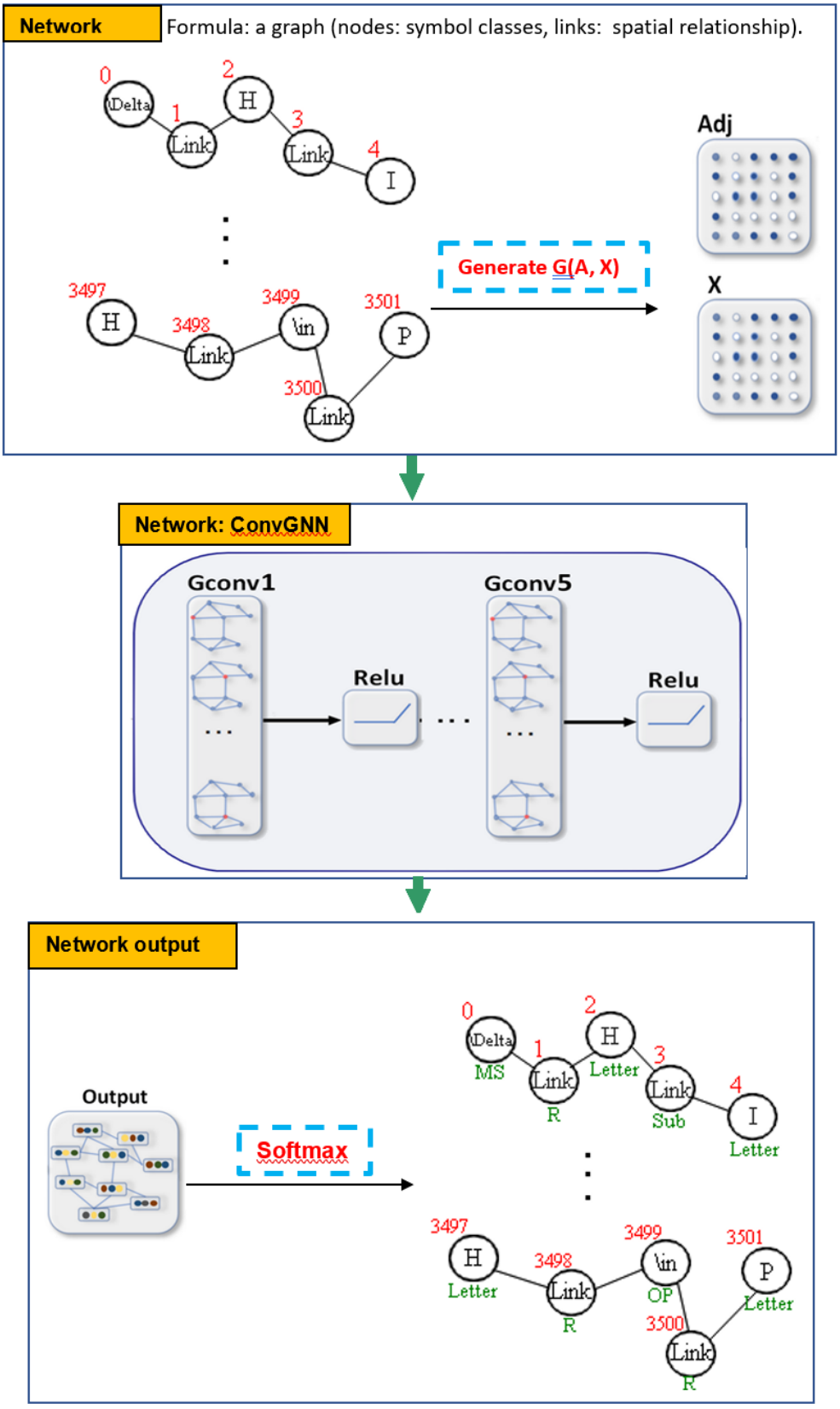


Figure 7: Architecture of ConvGNN, for mathematical formula layout analysis.

page images for training and 380-page images for validation. The data set includes image files of scanned math pages and their corresponding text files. Each text file contains information regarding mathematical expressions found in the document. This information consists of the x and y coordinates of the expression's centroid, width, height, and class (embedded or isolated).

YOLOv8 is typically trained in a two-step process: pretraining on a large-scale dataset (e.g., ImageNet) and fine-tuning on a specific detection dataset. During training, the model optimizes the loss function incorporating both localization and classification losses. The loss function penalizes incorrect predictions and encourages accurate bounding box localization and correct class predictions.

4.1.1 YOLOv8 fine-tuning

Based on our experiments, we have determined the optimal hyperparameters to be utilized in the YOLOv8 model.

- Input shape: 1447×2048.
- In our training program, we utilized a batch size of 8 and set the number of training epochs to 1000 with a learning rate of 0.01 and a momentum of 0.9. We also implemented early stopping with a window size of 50, which means training stopped if the validation loss did not decrease for 50 consecutive epochs.
- The used optimizer is SGD (Stochastic Gradient Descent).

The performance of the YOLOv8 model on the training and validation sets is illustrated in Figure 8, which showcases three distinct loss metrics: box loss, target loss, and classification loss. Box loss assesses the model's ability to accurately identify the center of an object and predict its bounding box. Target loss evaluates the probability of locating an object within a suggested region of interest. Classification loss reflects the model's proficiency in correctly classifying objects. The model demonstrated notable improvements in precision, recall, and average accuracy throughout the training process. The decreasing values of box loss, target loss, and classification loss evidence this. The model exhibited significant enhancements in precision, recall, and mean Average Precision (mAP) after 200 epochs, ultimately reaching a stable state after 400 epochs and stopped early at 801. Based on the above mentioned metrics, the best results were observed at epoch 751 and are briefly presented in Table 2.

Table 2: Training results for mathematical formula detection

	Box loss	Cls loss	Tar loss	Pre	Rec	mAP50	mAP95
train	0.86748	0.56159	0.89632	2*0.89867	2*0.86504	2*0.90957	2*0.70882
val	0.79908	0.49438	0.86164				

During the training process, the model was validated using a dataset of 380 validation images. The results are presented in Table 3.

Table 3: Results of the validation of YOLOv8 model on the validation data set

Class	Images	Instances	Precision	Recall	mAP50	mAP50-95
all	380	7941	0.896	0.867	0.91	0.709
emb	380	6595	0.825	0.755	0.83	0.552
iso	380	1346	0.967	0.98	0.989	0.865

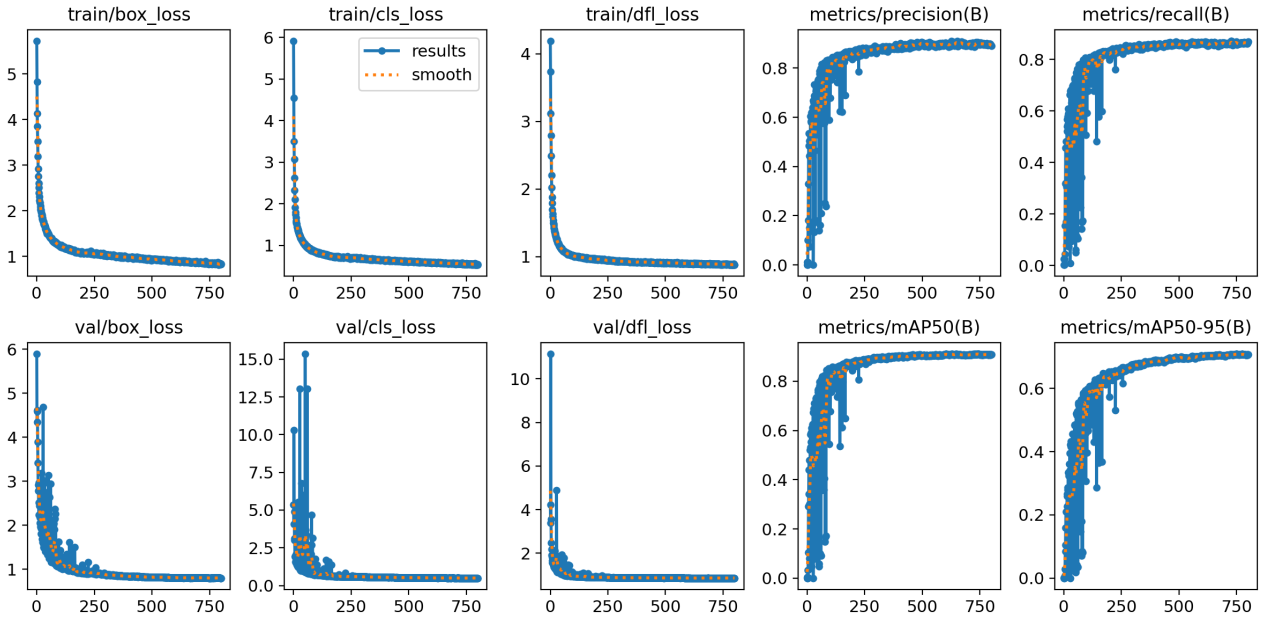


Figure 8: The training and validation curves of box loss, objectness loss, classification loss, precision, recall, and mean average precision (mAP).

4.1.2 YOLOv8 Testing

After training the model, it was applied to analyze unseen document images. We employed the mean average precision (mAP) metric to assess the model's performance, commonly used in object detection tasks. The mAP is calculated by determining each class's Average Precision (AP) and then averaging it across multiple classes (as shown in Equation 10). Precision (AP) represents the proportion of correctly identified objects (true positive detections) among all positive detections (both correct and incorrect).

$$mAP = \frac{1}{C}\sum_{c=1}^{C} AP_c; \tag{10}$$

where: C : represents the total number of classes and AP_c : represents the average precision for class c.

The mean average precision (mAP) is a comprehensive metric that provides an accurate assessment of the overall performance of a model. It is commonly employed in computer vision competitions and research to evaluate and compare the performance of various object detection models.

The precision refers to the level of accuracy exhibited by a model in correctly recognizing just pertinent objects. The metric under consideration is the proportion of true positives (TPs) in relation to the total number of detections produced by the model. (See Equation 11)

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{alldetections} \tag{11}$$

Recall is a metric that quantifies the model's capacity to accurately identify all instances of true positive predictions in relation to the total number of ground truths. (See Equation 12)

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{allgroundtruths} \tag{12}$$

A model is considered to be of high quality if it exhibits both high precision and strong recall. An ideal model exhibits no false negatives and no false positives, resulting in a precision score of 1 and a recall score of 1.

The recall and precision curves for each confidence threshold can be seen in Figures 9 and 10, respectively. The curves illustrate that choosing a high confidence threshold leads to a decrease in recall and an increase in precision. The result suggests that there are only a limited number of accurate detections, but they are very precise. On the other hand, if a less strict threshold is used, there will be numerous detections, but with many false positives. All mathematical formulas will be identified, but many incorrect ones will also be detected. Remember that recall and precision scores fall between 0.0 and 1.0, with a higher score indicating better performance. After analyzing the curves, we maintained a confidence level of 0.1 for making predictions. We have summarized the results obtained from this evaluation in Table 4.

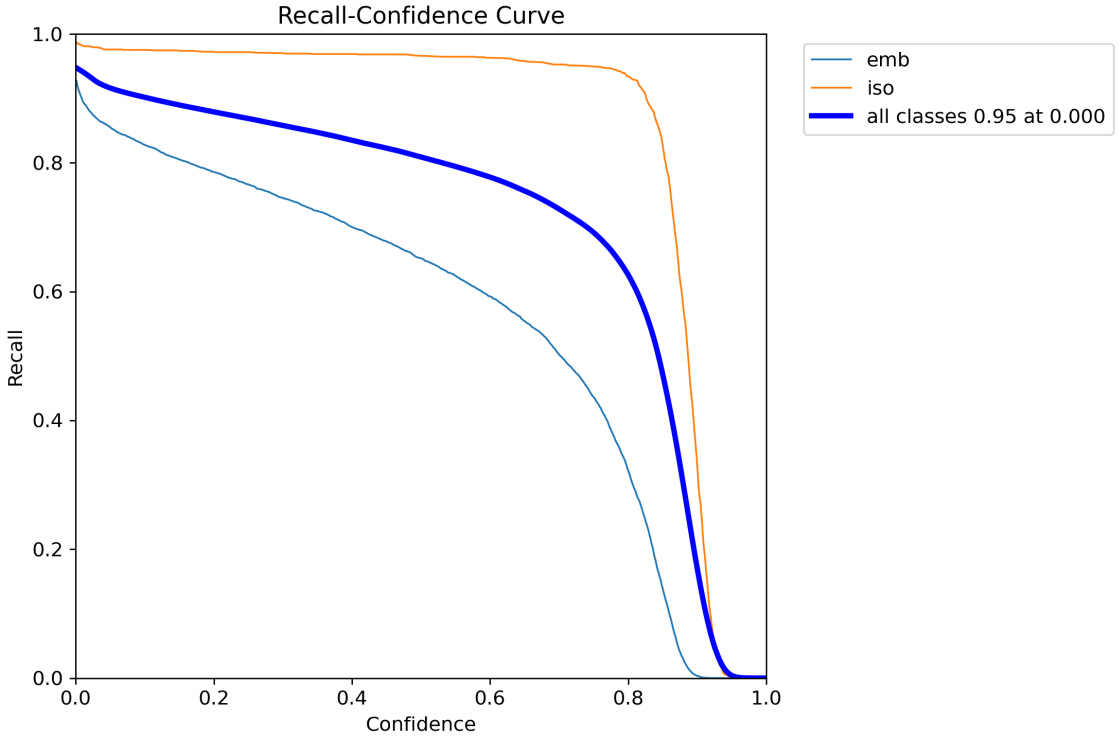


Figure 9: The recall score for our custom YOLOv8 object detection model.

Table 4: Results of the validation of YOLOv8 model on the test data set

Class	Images	Instances	Precision	Recall	mAP50	mAP50-95
all	380	7849	0.888	0.857	0.905	0.701
emb	380	6435	0.833	0.744	0.83	0.547
iso	380	1414	0.943	0.97	0.979	0.855

We also employed the confusion matrix as an evaluation metric, which is a useful tool for assessing the performance of a Machine/Deep Learning model's performance by examining its predictions' accuracy in relation to the ground truth. In the confusion matrix, each row corresponds to an actual class, while each column corresponds to a predicted class. The Figure 11 displays the confusion matrix presenting the results of our model's application on the test dataset. As observed, the model successfully detected the embedded formulas with an accuracy rate of 77% but failed to detect 23% of them. In the case of isolated formulas, our system achieved a detection rate of 98%. Our analysis revealed a complete absence of confusion between the two classes. It should be noted that the background class is inherently incorporated by default, thanks to a feature of YOLOv8's operational methodology. Specifically, YOLOv8 classifies any zone lacking an object as a back-

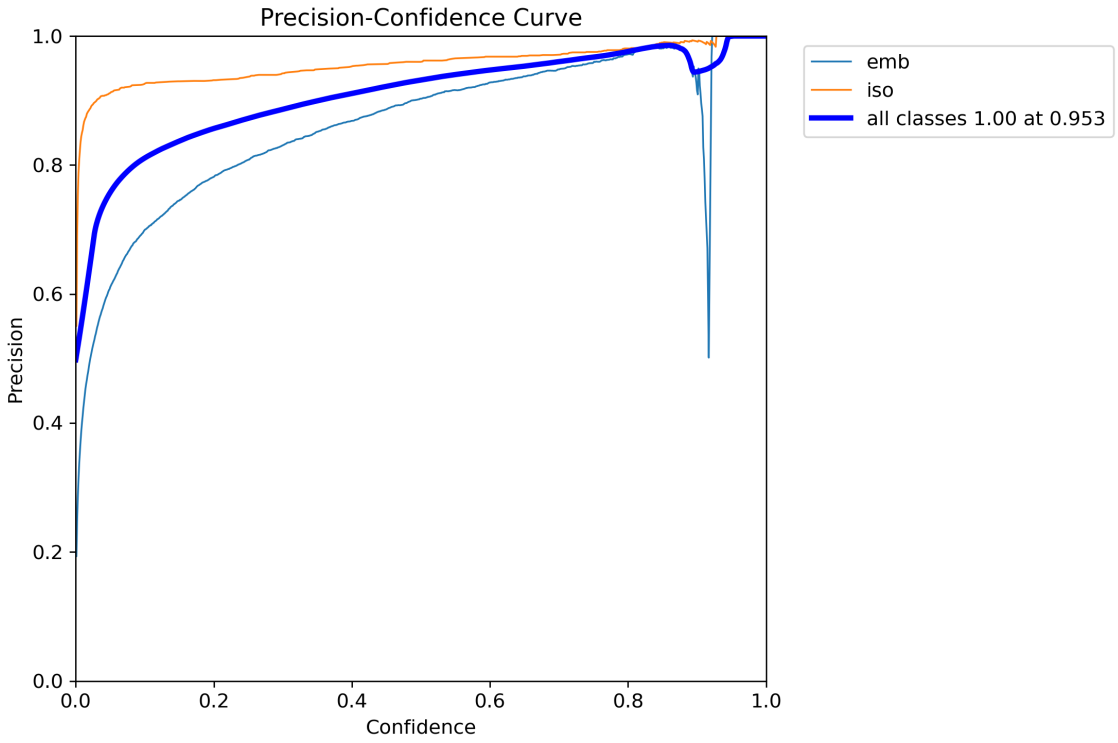


Figure 10: The precision score for our custom YOLOv8 object detection model.

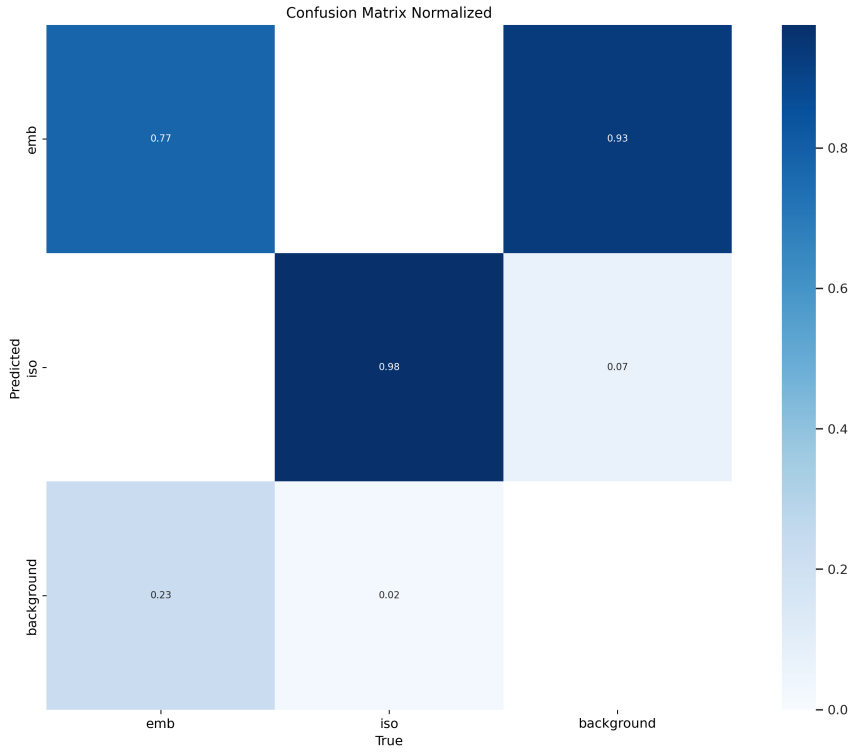


Figure 11: Confusion matrix.

ground. For this research, we focused exclusively on the two specified classes to evaluate the performance of our system.

The developed model was used to analyze document images that had not been seen before. The image in Figure 12 provides an example of the model's prediction capabilities, showing that it can accurately identify and classify various objects in document images. The prediction probabilities for these objects consistently exceed 90%, especially for isolated formulas. However, recognizing embedded formulas is more challenging due to the possibility of confusion between certain parts of the formula and the surrounding text. Table 5 illustrates the experimental findings, whereby the performance of our model is compared with that of other relevant studies, utilizing mAP50 metric.

4.2 Results from Faster RCNN for Symbol Detection and Classification and ConvGNN for Formula Layout Analysis

To develop our handwritten math expression recognition models, we rely on the widely used CROHME dataset [6]. This dataset comprises math expressions handwritten on online devices, with stroke trajectory described by (x, y) coordinates in InkML files. We use the images generated from these files for training, validation, and testing. The CROHME training set contains 8836 expressions, while the test set varies concerning the year they were released. We evaluate our system based on the CROHME 2014, CROHME 2016, and CROHME 2019 test sets, which consist of 986, 1147, and 1199 expressions, respectively. The CROHME dataset has 101 symbol classes and 6 structural relation classes (*Right*, *Above*, *Below*, *Superscript*, *Subscript* and *Inside*). This large number of symbol classes poses several challenges, including digits, characters, and operators such as +, $\times$, $\sqrt{}$. Some of the challenging categories include small symbols like *points* and *commas*, and symbols that have similar shapes such as 1, \, $\times$, x, c, and C. For a visual representation of the dataset, see Figure 13.

We utilized the CROHME dataset [6] to execute our formula analysis model. The mathematical formulas in the dataset are labeled at the symbol level, where each symbol node is assigned a specific label from a set of 20 labels to identify it accurately. For more information about the labels used, please refer to Table 6.

To conduct our experiments, we divided the math formula dataset into three sets: training, validation, and testing. This dataset contained 180 formulas, of which approximately 50% (90 formulas, generating 1737 train nodes) were used for training, 20% (36 formulas, generating 669 validation nodes) for validation, and 30% (54 formulas, generating 1096 test nodes) for testing. Our decision to use only 50% of the dataset for training and a larger set for testing was intentional, as we wanted to test the system's ability to generalize even with a limited training set. Our evaluation of the math formula recognition system relied on the accuracy metric, which measures the percentage of correctly predicted sequences.

After conducting experiments, we have established the following hyperparameters to be used in the proposed model:

- When resizing images, we maintain their aspect ratio while ensuring that their dimensions fall within a range of 640 pixels (with a maximum and minimum limit).
- We trained Faster R-CNN, Inception, and ResNet models, which are pre-trained models on the MSCOCO dataset. Despite the stark contrast between our generated images and the natural images in the dataset, we discovered that using pre-trained models significantly expedites the convergence process compared to training from scratch.
- We used 300 proposals per image.
- In our training program, we utilized a batch size of 2 and set the number of training steps to 900,000 for math symbol detection. The model was allowed to learn for 126,000 steps.

We defined the setup options and training configurations for our formula layout analyzing model as follows:

- A 5-layer ConvGNN network was proposed.

of the actio topological considerations. In the limit of $S \to S_n$ the quantum theory in this case is described by a finite degrees of freedom and is characterized by an arbitrary parameter just as in the scalar field theory . We conclude the paper in Section 4 with a summary and outlook.

2. Scalar Field

Let $\mathcal{M}$ be a flat Minkowskian m slice has the topology of a cylinder $S^1 \times R$. Let r be the radius of S^1 and let θ be the angle spanning it. Consider the following flat metric in $\mathcal{M}$ given by

$$ds^2 = dt^2 - dz^2 - r^2 d\theta^2, \tag{2.1}$$

Let S given by $z = v$ be a surface embedded in $\mathcal{M}$ where v is the parameter iting value of this parameter is given by $v = 1$. The pull-back of the above metric in $\mathcal{M}$ to the time-like surface S can be written as

$$ds^2|_{z=vt} = (1 - v^2)dt^2 - r^2 d\theta^2 = h_{ab} dy^a dy^b. \tag{2.2}$$

As v ull surface S_n given Eq metric h_{ab} inudced on S is degenerate as $S \to S_n$.

Consider a single real scalar field ϕ which is defined on the surface S. We will assume that the scalar field is valued ue degeneracy of the metric in the limit of $v \to 1$ leads to a Hamiltonian that is ill- problem we consider a renormalized field $f\phi$ where f is the renormalization parameter. The action for such a real scalar field can

$$\mathcal{S} = \int_S \sqrt{h} d^2y \; \mathcal{L} = \frac{f^2}{8\pi} \int_S \sqrt{h} d^2y h^{ab} \partial_a \phi \partial_b \phi. \tag{2.3}$$

As is evident from Eqn. (2.3), f can also be interpreted as the coupling constant of t

The field ϕ obeys the equation of motion

$$h^{ab} \partial_a \partial_b \phi = 0. \tag{2.4}$$

Figure 12: An example of formula detection by YOLOv8.

Table 5: Comparing our work with others in the field.

Works	Models	Databases	Performances
[15]	Object detection: EfficientDet-Lite0 formula calculation: Reverse Polish Notation algorithm with operator precedence rules	200 formula images for training the object detection model. 20 for testing formulas. Recognition 16 symbols. classes	Object detection: mAP 63.17% formula Recognition: accuracy 81.9%
[16]	CNN model: HMS-VGGNet for offline symbol recognition	Crohme 2014 ad 2016 (101 symbols), HASY v2 (369 symbols)	92.42% accuracy tested on Crohme 2016 and 85.05% tested on HASY v2
[17]	Feed forward back propagation neural network, SVM, KNN for printed and handwritten symbol and formula recognition	Personal database	Accuracy of 88.5% with SVM and of 78.4% with KNN
[18]	Symbol recognition: RCN, layout analysis: combine segmentation and position info. to build the Latex sequence	MNIST, Kaggle, personal dataset of 200 formulas	Symbol recognition: 49%, math formula recognition: 70%
[19]	Symbol localisation (YOLO v5 object detector) build a graph on math symbols graph reasoning network to generate SLT	Crohme 2014, 2016, and 2019, OffRaSHME	47.67% for symbol detection, 56.95% for formula recognition
[14]	YOLOv3, WAP	Marmot	YOLOv3: 93% (isolated), 73% (inline), WAP: 51.77% (isolated), 45.50% (inline)
[21]	Faster RCNN	Marmot and GTDB (part of IBEM)	85.15% (inline), 91.04% (isolated)
Ours	**YOLOv8 for formula detection, Faster R-CNN for symbol detection and classification, and ConvGNN for layout analysis**	**Crohme 2019**	**98% (isolated formula detection), 77% (embedded formula detection), 87.3% (symbol detection), 92% (layout analysis)**

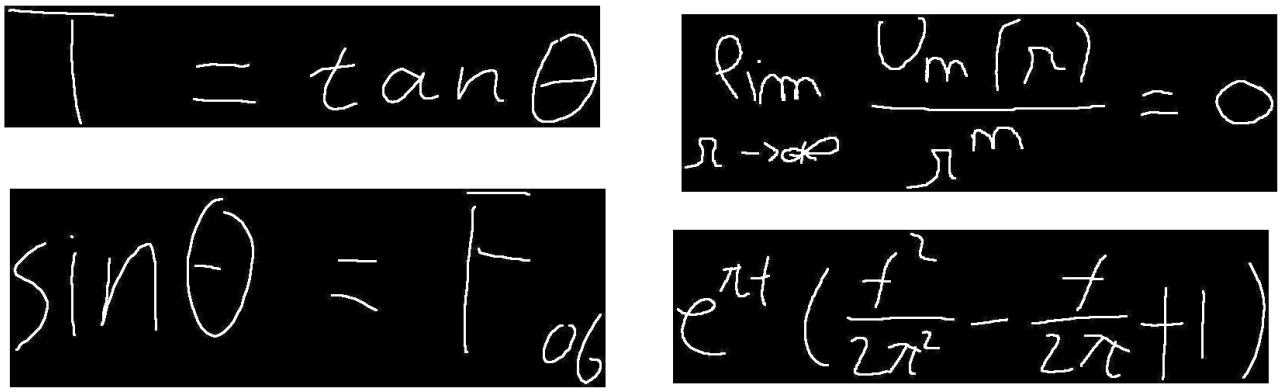


Figure 13: Here are some sample images from the CROHME dataset.

Table 6: Used labels.

Label	Signification
letter	Letters from a (resp. A) to z (resp. Z)
OP	Operator (+, -, ¡, ¿, /, etc.)
DL	Delimiter ((,), , , etc.)
Integer	digits from 0 to 9
HFB	Horizontal Fraction Bar
IS	Integral Sign
MS	Math Symbol ($\triangle$, ∞, $\emptyset$, etc.)
RS	Root Symbol
Pun	Punctuation sign (,, ..., etc.)
SPS	Summation Product Symbol
NF	Name of Function (f(x), g(x), etc.)
MF	Math Function (cos, sin, etc.)
LS	Limit Sign
Det	Determinent
R	Right
A	Above
B	Below
I	Inside
Sub	Subscript
Sup	Superscript

- Hermite filters of order 2 $(k = 2)$ were used, and we initialized each hidden layer with 18 units.
- The ConvGNN layer had an $L2$ regularization factor of 5.10^{-4} and a dropout of 0.005.
- The model was trained as a single big graph batch for a maximum of 1000 epochs with a learning rate of 0.001. We also implemented early stopping with a window size of 50, which means training stopped if the validation loss did not decrease for 50 consecutive epochs.

We assessed the suggested system based on its ability to detect symbols and recognize mathematical expressions.

4.2.1 Symbol Detection

After 126,000 steps, the mean average precision at a 50% IoU threshold is 87.3%. Check Table 7 for a list of the top and bottom ten classes regarding average precision.

Table 7: Highest and lowest performing symbols based on average precision.

best performing symbols	AP	worst performing symbols	AP
$\backslash forall$	1.000	$\backslash exists$	0.000
$\backslash Delta$	1.000	$\backslash mu$	0.000
$=$	0.998	o	0.000
$\backslash sum$	0.987	P	0.175
$-$	0.973	S	0.200
8	0.970	v	0.273
$\backslash ldots$	0.964	X	0.312
$+$	0.962	s	0.316
3	0.959	$\backslash\}$	0.333
$\backslash infty$	0.959	I	0.333

In the two figures, labeled 14 and 15, we can find information regarding the mean average precision and the average recall of three different class sizes. Symbols smaller than 32^2 are classified as small, those between 32^2 and 96^2 as medium, and the rest as large. The figures indicate that the size of symbols has a significant impact on the accuracy of the detector. Larger symbols lead to more precise predictions. Additionally, figures labeled 16, 17, and 18 provide further examples of the detection model's output. The first example shows a detection with no errors, while the second one depicts a typical confusion case between opening and closing parentheses. Finally, the last figure shows that the part in of the symbol $\backslash sin$ was erroneously detected as separate from the $\backslash sin$ symbol.

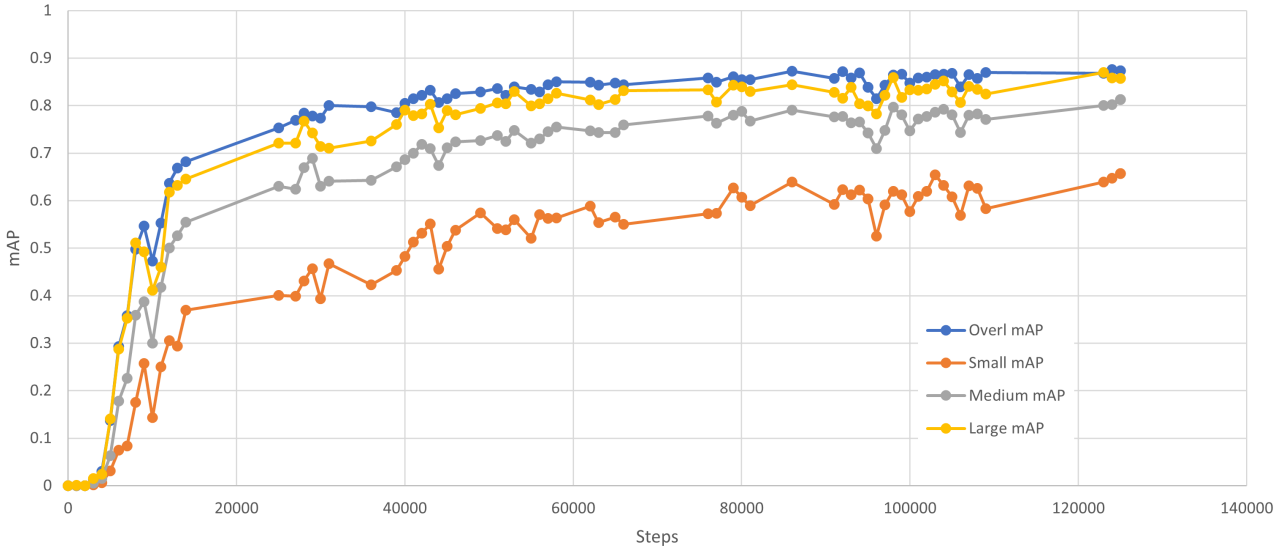


Figure 14: Calculating the Mean Average Precision (mAP) for each symbol size.

In the graph in Figure 19, we can observe the training and validation losses throughout the update steps. One notable observation is that the training and validation losses are similar, with validation loss slightly higher. Furthermore, the model's performance improves over time by noticing a decline in training and validation loss,

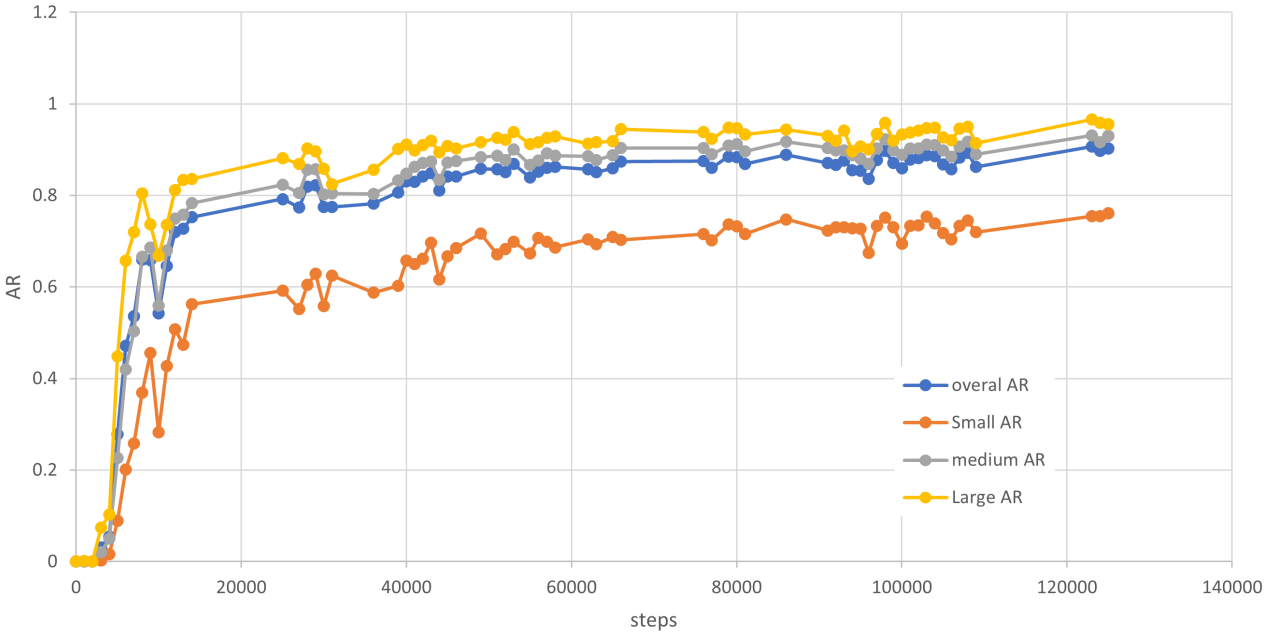


Figure 15: Calculating the average recall (AR) for each symbol size.

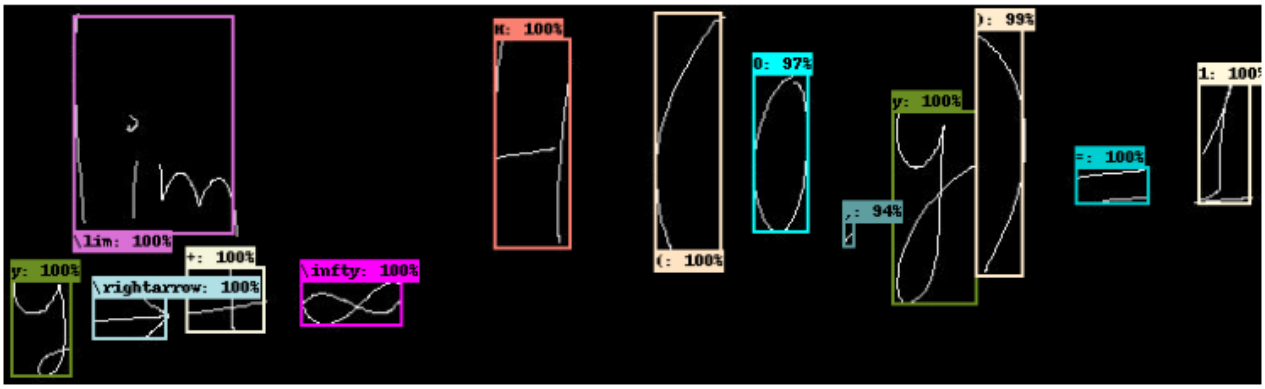


Figure 16: Detecting math symbols accurately and error-free.

followed by a stable curve, indicating a good fit. Additionally, Figure 20 demonstrates the breakdown of the validation loss into classification and localization loss, and the outcome is satisfactory.

Table 7 shows that certain symbols in mathematical formulas have unique structures, resulting in high precision during detection. However, some symbols are often mistaken for others, leading to lower accuracy and recall rates for the model. For instance, Table 8 highlights examples of symbols that may be missed or confused with other symbols during detection. The detection of symbols with small widths or heights, such as the dot, hyphen, and number 1, poses a significant challenge for the models and may lead to missed symbols. Additionally, the models frequently misclassify symbols that have similar shapes, like 0 and $\backslash theta$ or 9 and g, which is a common detection error. Moreover, the models may misclassify symbols with the same or similar shape but are symmetric, such as (and) or 7 and F. It is worth noting that illegible handwriting can also lead to erroneous classification, as shown in Figure 21. This type of error can be challenging even for humans to detect.

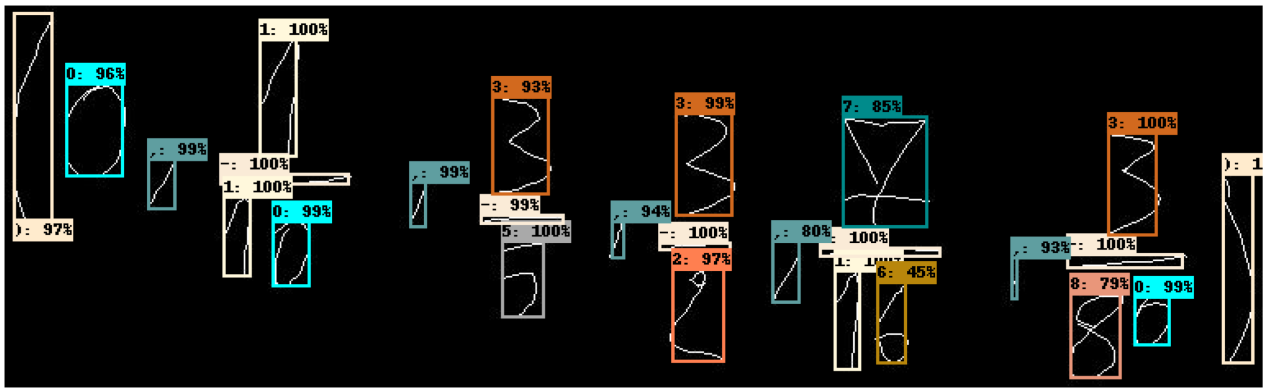


Figure 17: One common error in detection is the confusion between opening and closing parentheses.

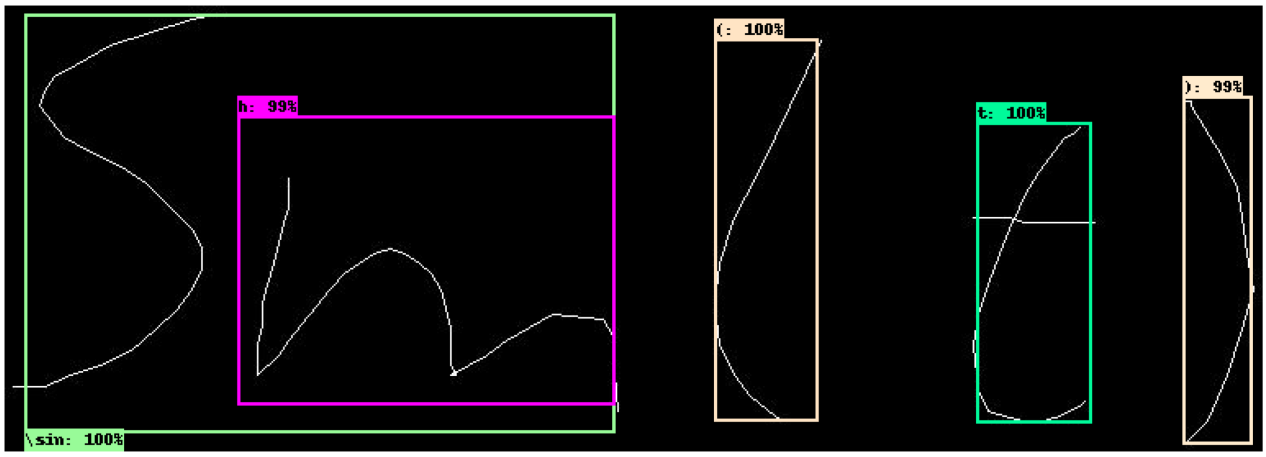


Figure 18: Separately detecting the $\backslash sin$ symbol and its composite in inside.

Table 8: Example of confused or missed symbols.

Ground truth	Prediction	Ground truth	Prediction
0	$\backslash theta$	q	9
d	1	.	miss
G	v	(	)
b	d	-	miss
7	F	1	miss
9	g	)	(
2	c	3	s
W	N	¿	¡
$\geq$	$\leq$	d	b
C	(	$\leq$	$\geq$

4.2.2 Formula Layout Analysis

We have conducted tests on four distinct models: the Multi-layer Perceptron (MLP) model, the basic ConvGNN model, and the spectral models that use Tchebychev (ConvGNN-Hermite) and Hermite (ConvGNN-Chebychev) polynomial approximation filters. The summary of our comparative evaluation experiments is

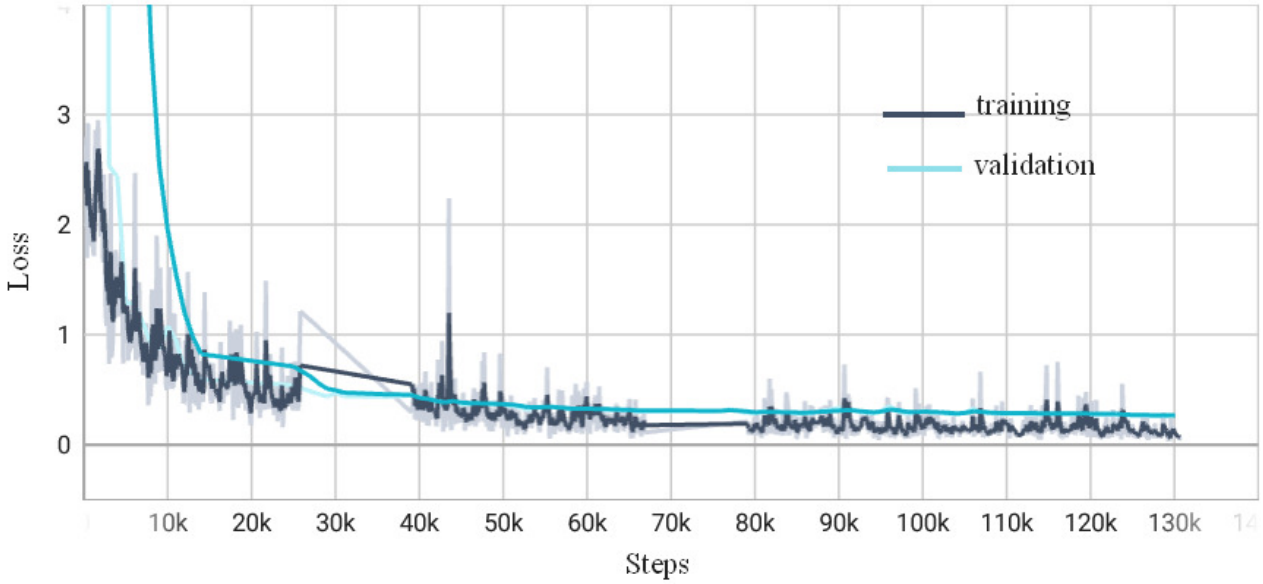


Figure 19: The losses for training and validation.

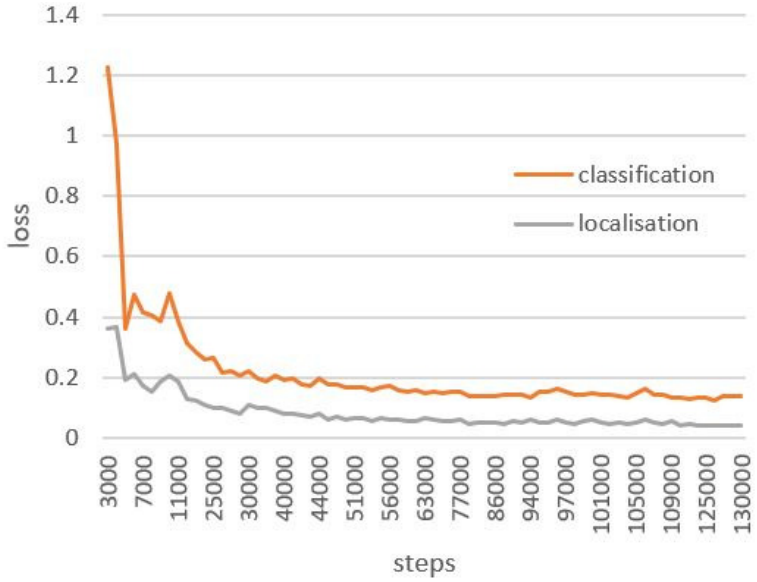


Figure 20: Validation losses for classification and localization.

presented in Table 5.

Table 9: Results from simple and spectral ConvGNNs.

Proposed Models	Train Accuracy	Runtime	Cost
Simple convGNN	0.28	0.03124	2.17124
ConvGNN-hermite	0.92	0.02992	0.31295
ConvGNN-chebychev	0.82	0.02992	0.68176

Our study evaluated the accuracy, run time, and cost of our methods on test nodes. The reported numbers

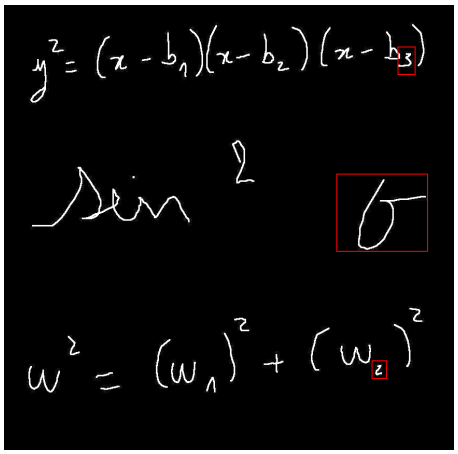


Figure 21: The handwriting was difficult to read, resulting in incorrect classification. Specifically, the number 3 was mistaken for an s, the letter G was mistaken for a v, and the number 2 was mistaken for a c.

indicate the performance of spectral ConvGNN, which uses the Hermite approximation filter and a limited number of nodes from the CROHME dataset. Our results demonstrate that this method performs exceptionally well. Despite this, it requires more training time than other methods, like MLP, due to its complexity and numerous parameters to learn in each training epoch. Furthermore, Spectral ConGNN can achieve superior performance with fewer training epochs. Displayed in Figure 22 are the accuracies for both training and validation, while Figure 23 depicts the loss for each epoch during training and validation.

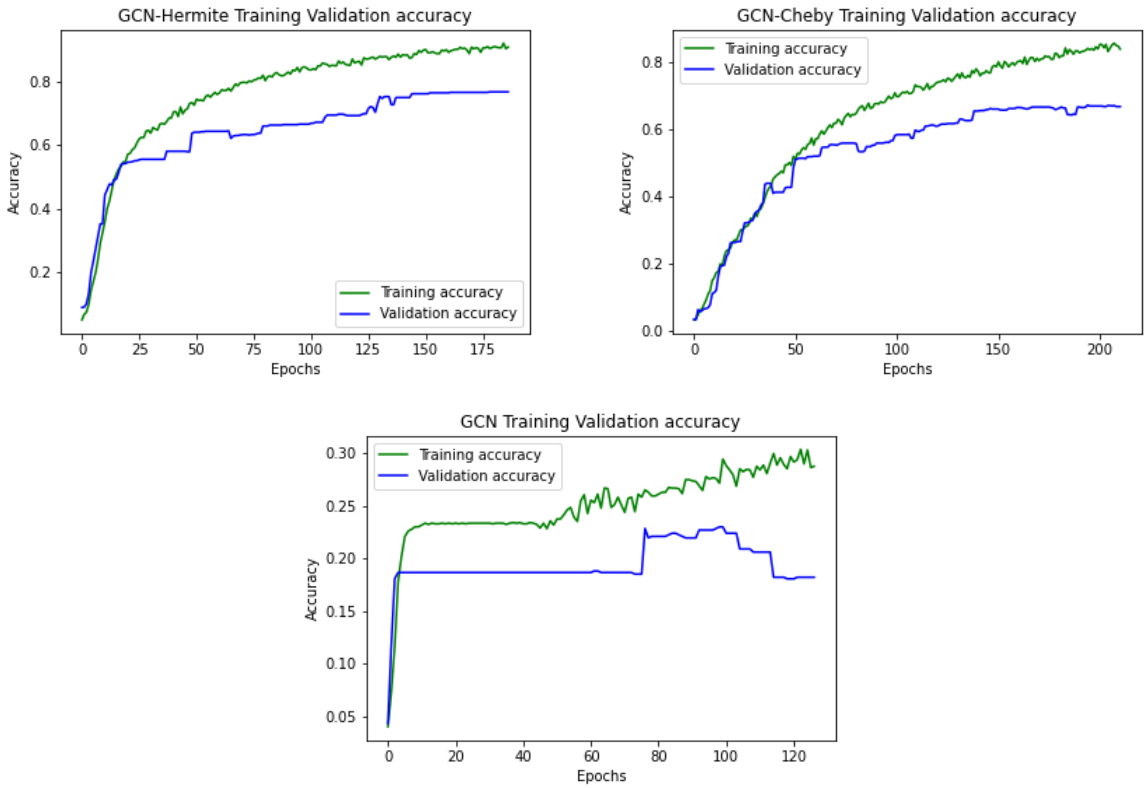


Figure 22: Training Validation Accuracy: (a) ConvGNN-Hermite (c) ConvGNN-cheby, (d) Simple ConvGNN.

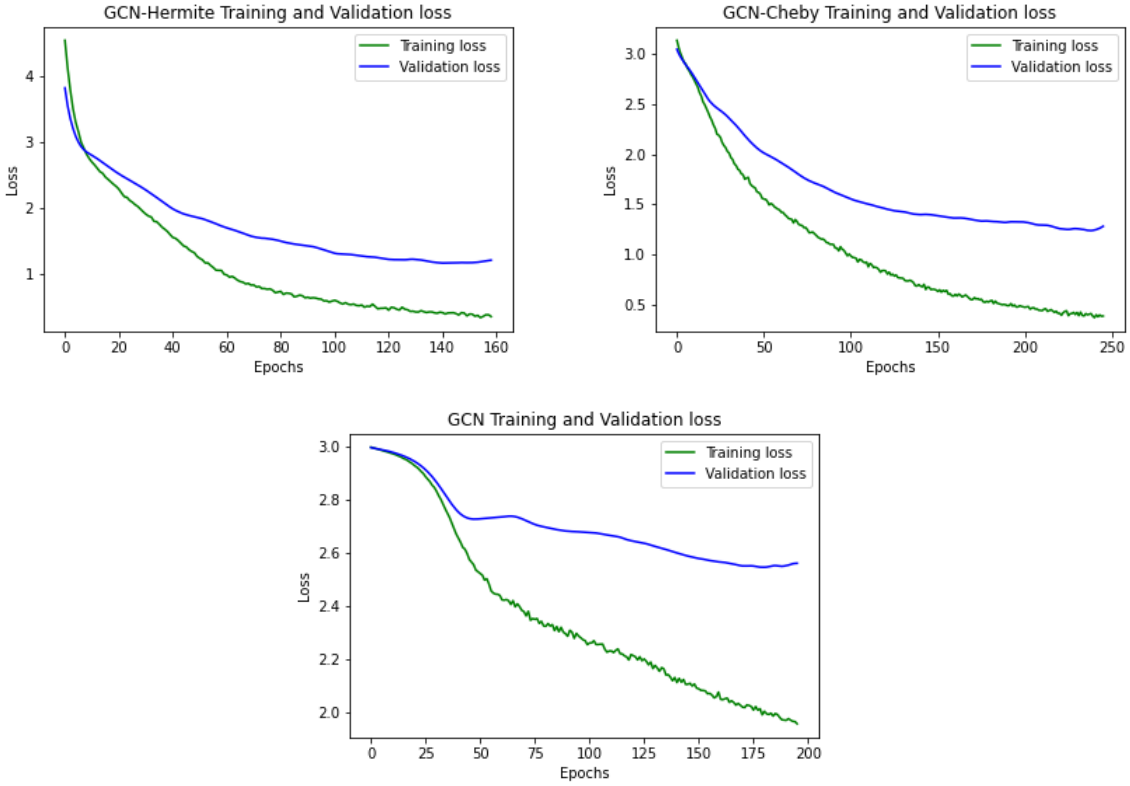


Figure 23: Training Validation Loss: (a) ConvGNN-Hermite, (c) ConvGNN-Cheby (d) Simple ConvGNN.

Our system can get good results in a few training epochs, one hundred training epochs, and even less stable compared to the MLP model, which needs more than 600 epochs to be stable. Besides, we can see that training, and validation accuracy/loss of Spectral-based ConvGNN models, especially the model using Hermite Filter, rise or descend very quickly and stably. Here is a summary of the research on recognizing mathematical symbols and formulas. The results are promising and comparable to the work done by [19], which used a graph neural network and achieved an accuracy of 92% on the same database.

5 Conclusion and Future Work

We have developed a solution using deep learning techniques to extract and recognize mathematical formulas. By employing the YOLOv8 deep learning object detection model and adapting it to math formula detection, we have been able to identify and localize math formulas within images automatically. Once formulas were extracted, we used the Faster-RCNN architecture to localize and classify symbols in the image of the formulas. We found that this model works well with offline data as well. To parse the formulas, we employed a spectral-based ConvGNN to predict the relationship between the symbols, which allows us to model the input formula image as a symbol graph. This approach is more interpretable than sequence representation. Our tests on several math formulas collected from CROHME datasets demonstrate that the ConvGNN model is effective in categorizing math symbols and identifying spatial relations. This model opens up possibilities for further research and development in graph representation learning, and we plan to test with various GNN architectures such as Graph Auto-Encoder Networks, Recurrent GNNs, and Gated Graph Neural Networks.

References

[1] Lin, X., Gao, L., Tang, Z., Lin, X., Hu, X.: Mathematical Formula Identification in PDF Documents. In: 2011 International Conference on Document Analysis and Recognition. pp. 1419-1423 (Sep 2011). https://doi.org/10.1109/ICDAR.2011.285, iSSN: 2379-2140

[2] Suzuki, M., Tamari, F., Fukuda, R., Uchida, S., Kanahori, T.: INFTY: an integrated OCR system for mathematical documents. In: Proceedings of the 2003 ACM symposium on Document engineering. pp. 95-104. DocEng '03, Association for Computing Machinery, New York, NY, USA (Nov 2003). https://doi.org/10.1145/958220.958239, http://doi.org/10.1145/958220.958239

[3] Baker, J.B., Sexton, A.P., Sorge, V.: A Linear Grammar Approach to Mathematical Formula Recognition from PDF. In: Carette, J., Dixon, L., Coen, C.S., Watt, S.M. (eds.) Intelligent Computer Mathematics. pp. 201-216. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642- 02614-0 19

[4] Sorge, V., Bansal, A., Jadhav, N.M., Garg, H., Verma, A., Balakrishnan, M.: Towards generating web-accessible STEM documents from PDF. In: Proceedings of the 17th International Web for All Conference. pp. 1-5.W4A '20, Association for Computing Machinery, New York, NY, USA (Apr2020). https://doi.org/10.1145/3371300.3383351, http://doi.org/10.1145/3371300.3383351

[5] Zhang, X., Gao, L., Yuan, K., Liu, R., Jiang, Z., Tang, Z.: A Symbol Dominance Based Formulae Recognition Approach for PDF Documents. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). vol. 01, pp. 1144-1149 (Nov 2017). https://doi.org/10.1109/ICDAR.2017.189, iSSN: 2379- 2140

[6] Harold Mouchère, "ICDAR 2019 Competition on Recognition of Handwritten Mathematical Expressions and Typeset Formula Detection", *International Conference on Document Analysis and Recognition (ICDAR)*, 2019.

[7] Suzuki, M., Yamaguchi, K. (2016). Recognition of E-Born PDF Including Mathematical Formulas. In: Miesenberger, K., Buhler, C., Penaz, P. (eds) Computers Helping People with Special Needs. ICCHP 2016. Lecture Notes in Computer Science(), vol 9758. Springer, Cham. https://doi.org/10.1007/978-3-319-41264-15

[8] Deng, Y., Kanervisto, A., Ling, J., Rush, A.M.: Image-to-markup generation with coarse-to-fine attention. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017. Proceedings of Machine Learning Research, vol. 70, pp. 980-989. PMLR (2017), http://proceedings.mlr.press/v70/deng17a.html

[9] Phong, B.H., Dat, L.T., Yen, N.T., Hoang, T.M., Le, T.L.: A deep learning-based system for mathematical expression detection and recognition in document images. In: 2020 12th International Conference on Knowledge and Systems Engineering (KSE). pp. 85-90 (Nov 2020). https://doi.org/10.1109/KSE50997.2020.9287693, iSSN: 2164-2508

[10] A. K. Shah, A. Dey, and R. Zanibbi, "A Math Formula Extraction and Evaluation Framework for PDF Documents," in Document Analysis and Recognition – ICDAR 2021, Cham, 2021, pp. 19–34. doi: 10.1007/978-3-030-86331-92

[11] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: SSD: Single shot multi-box detector. In: European conference on computer vision. pp. 21-37. Springer (2016)

[12] Mahdavi, M., Sun, L., Zanibbi, R.: Visual Parsing with Query-Driven Global Graph Attention (QD-GGA): Preliminary Results for Handwritten Math Formula Recognition. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 2429-2438. IEEE, Seattle, WA, USA (Jun 2020). https://doi.org/10.1109/CVPRW50498.2020.00293, https://ieeexplore.ieee.org/document/9150860/

[13] Anitei, Dan; Sánchez, Joan Andreu; Benedí, José Miguel: IBEM Mathematical Formula Detection Dataset. https://zenodo.org/record/4757865

[14] B. H. Phong, L. T. Dat, N. T. Yen, T. M. Hoang and T. -L. Le, : A deep learning-based system for mathematical expression detection and recognition in document images. 2020 12th International Conference on Knowledge and Systems Engineering (KSE), Can Tho, Vietnam, 2020, pp. 85-90, doi: 10.1109/KSE50997.2020.9287693.

[15] Muhammad Util Ahmadi Kamal, Nanik Suciati and Shintami Chusnul Hidayati, "Calculation of Handwriting Mathematics Expressions on Mobile Devices Using Efficientdet-Lite0 And Reverse Polish Notation", *The Journal on Machine Learning and Computational Intelligence (JMLCI)*, Dec 2021.

[16] Lanfang Dong and Hanchao Liu, "Recognition of Offline Handwritten Mathematical Symbols Using Convolutional Neural Networks", *International Conference on Information and Communication Technology (ICoICT)*, Dec 2017.

[17] Sagar Shinde1, Akil Mulagirisamy, Daulappa Bhalke, and Lalitkumar Wadhwa, "Recognition of Math Expressions and Symbols using Machine Learning", *Turkish Online Journal of Qualitative Inquiry (TOJQI)*, Jul 2021.

[18] Hai Dai Nguyen, Anh Duc Le, and Masaki Nakagawa, "Deep neural networks for recognizing online handwritten mathematical symbols", *IAPR Asian Conference on Pattern Recognition (ACPR)*, Nov 2015.

[19] Jia-Man Tang, Jin-Wen Wu, Fei Yin, and Lin-Lin Huang, "Offline Handwritten Mathematical Expression Recognition via Graph Reasoning Network", *Asian Conference on Pattern Recognition (ACPR)*, 2021.

[20] J. Huang and al., "SmartMonitor: Speed/accuracy trade-offs for modern convolutional object detectors", *in Proceedings of the 17th Computer Vision and Pattern Recognition: CVPR*, 2017.

[21] Bui Hai Phong, Thang Manh Hoang, Thi-Lan Le, "End-to-end framework for the detection of mathematical expressions in scientific document images", Expert Systems, 39(1), 2022.

[22] Dan Anitei, Joan Andreu Sánchez, José Manuel Fuentes, Roberto Paredes and José Miguel Benedí, "ICDAR 2021 Competition on Mathematical Formula Detection", International Conference on Document Analysis and Recognition,ICDAR, 2021.

[23] Thomas N. Kipf and Max Welling, "Semi-Supervised Classification with Graph Convolutional Networks", International Conference on Learning Representations, 2017