Document Template Identification and Data Extraction using Machine Learning and Deep Learning Approach

by

Kaushik Roy
20101185
Md Fuad Islam
20101060
Md Minhazul Islam Rimon
20101078
Tasnim Mobarak
20101296
Mysha Samiha Priota
20301205

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

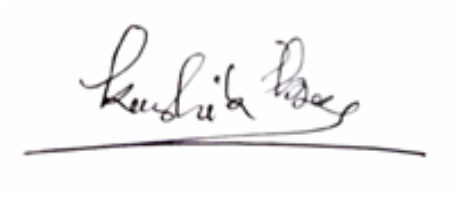
Department of Computer Science and Engineering
School of Data and Sciences
Brac University
January 2024

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

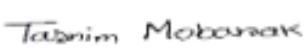
Student's Full Name & Signature:



Kaushik Roy
20101185

Md Fuad Islam
20101060

Md Minhazul Islam Rimon
20101078



Tasnim Mobarak
20101296

Mysha Samiha Priota
20301205

Approval

The thesis/project titled "Document Template Identification and Data Extraction using Machine Learning and Deep Learning Approach" submitted by

1. Kaushik Roy(20101185)
2. Md Fuad Islam(20101060)
3. Md Minhazul Islam Rimon(20101078)
4. Tasnim Mobarak(20101296)
5. Mysha Samiha Priota(20301205)

Of Fall, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 9, 2024.

Examining Committee:

Supervisor:

(Member)

Dr. Md. Khalilur Rhaman
Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

We maintain utmost confidentiality regarding the information we have gathered from Brac University exam transcripts, and it is not subject to future publication risk.

Abstract

As the world keeps progressing and we continue on our path to a technologically advanced tomorrow, the demand for quick data processing and organization is becoming more and more necessary. People now have access to technology more than ever before. Nowadays, technology allows for the processing and storing of nearly every kind of data. However, procedures requiring paper are still in place and the time-consuming process of moving these data from paper to computers is laborious which reduces work efficiency. Our goal is to make this tedious and time-consuming process fast and efficient, by directly converting the information of the manually checked scripts into digital data. Our research strategy involved gathering information from Brac University examination scripts, digitizing the verified scripts' data, and then uploading it to a spreadsheet file. The goal of the process is to make Brac University's grade-processing system quicker, more effective, and less tiresome for the teachers. Three machine learning models and three deep learning models as well as one transfer learning model were utilized for this study. Three common measures were used to evaluate the results which are precision, recall and F1-score. The KNN model showed up to 85% accuracy, whilst SVM showed 87% and SGDClassifier showed 81% accuracy. Meanwhile CNN and YOLOv8 showed 98.6% and 98.8% accuracy respectively. Since YOLOv8 is providing the best accuracy, we will be using this to create an interface that will carry out the complete data transformation process from beginning to end. Starting with capturing the image, processing it to identify the areas from which the data will be collected, and finally extracting the data, in the entire process YOLOv8 is going to be used. In the end, we will obtain precisely extracted data from handwritten exam scripts, which will be arranged in a spreadsheet, digitizing the laborious task of manually inputting each and every grade in a spreadsheet.

Keywords: CNN, KNN, YOLOv8, SVM, SGD classifier, Deep learning model, Machine learning model.

Dedication

Our effort is focused on the precise detection and identification of information from exam scripts in order to alleviate the burden that Brac University teachers endure each semester. We hope that our little contribution can have a big impact on future research efforts by other academics who want to use deep learning techniques to make test script verification easier for teachers.

Acknowledgement

We never could have completed our thesis without facing significant challenges without the grace of the almighty Allah. We express our gratitude to Dr. Md. Khalilur Rahman, our Supervisor, for his invaluable counsel and unwavering support. Additionally, we thank Mr. Sayantan Roy Arko, who served as our research mentor. Our research methods were greatly influenced by their knowledge and advice. Their perceptive criticism elevated our study to a new level and led us to think differently. In conclusion, we would like to express our gratitude to our parents for their thoughtful support and direction throughout our study.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

CNN Convention Neural Network

CTC Connectionist Temporal Classification

$ESPCN$ Efficient Sub-Pixel Convolutional Neural Network

$LSTM$ Long Short- Term Memory

$MNIST$ Modified National Institute of Standards and Technology

MT Morphological Transformation

$RCNN$ Region-based Convolutional Neural Networks

$ResNet$ Residual Network

RNN Recurrent Neural Network

VGG Visual Graphics Group

Chapter 1

Introduction

1.1 Motivation

In our daily life, we go through and process an impressive amount of documents. Documents like birth certificates, income tax, different types of government forms, passports, exam papers etc. Almost in all cases, we are required to extract data from these documents and copy them to a different virtual storage. These processes are tedious and time consuming. And in more than enough cases, there are bound to be some mistakes made which makes them even more obscure. That's why we need a system that will automate these processes and make these tasks more efficient.

And For recognizing information which needs to be extracted from documents, we

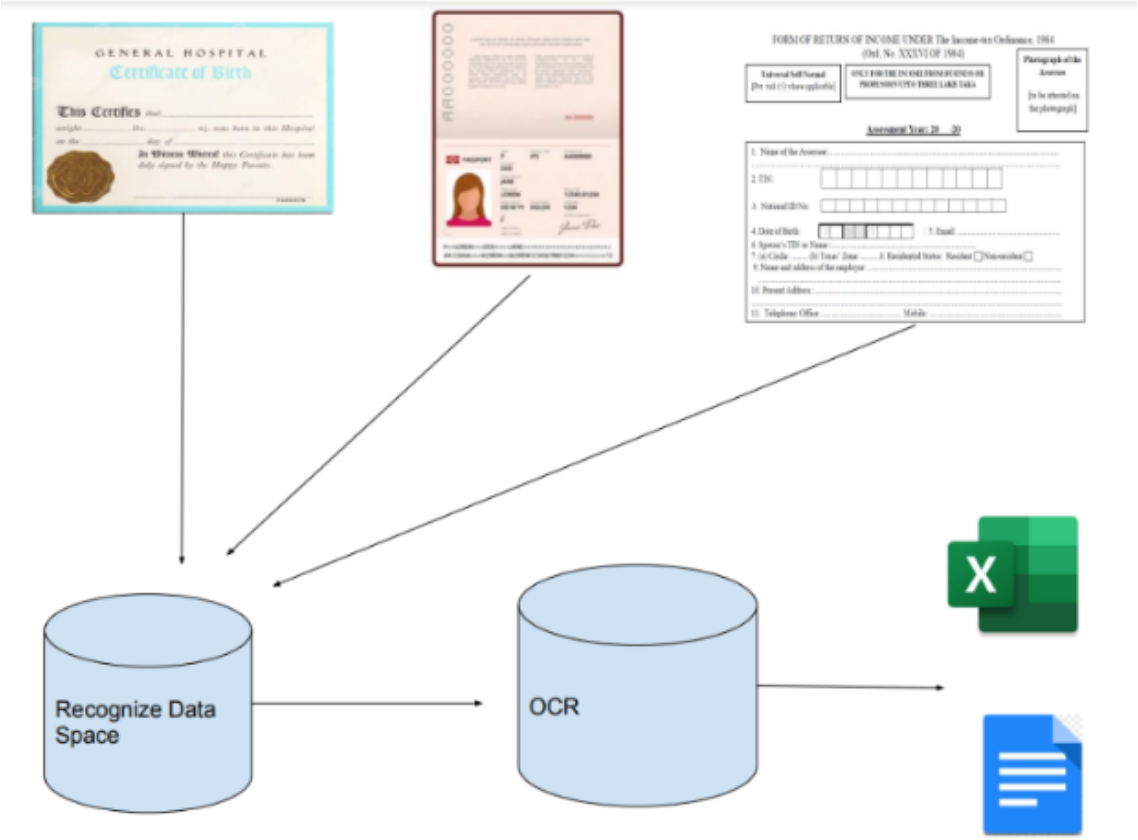


Figure 1.1: General Process of the Application.

have chosen to go with both Machine Learning and Deep Learning models. The data these models are going to extract will be reexamined a second time , if it is available. And upon re examining taken data, the secondary keys will be placed following primary keys. With that, the human error will significantly increase plus huge decrease in labor time compared to data extraction via humans. This technique will remove the monotonous working of every teacher who went through collecting data of every student's mark in a datasheet. And hopefully, we will be able to expand such a process to every form of document/analog form available.

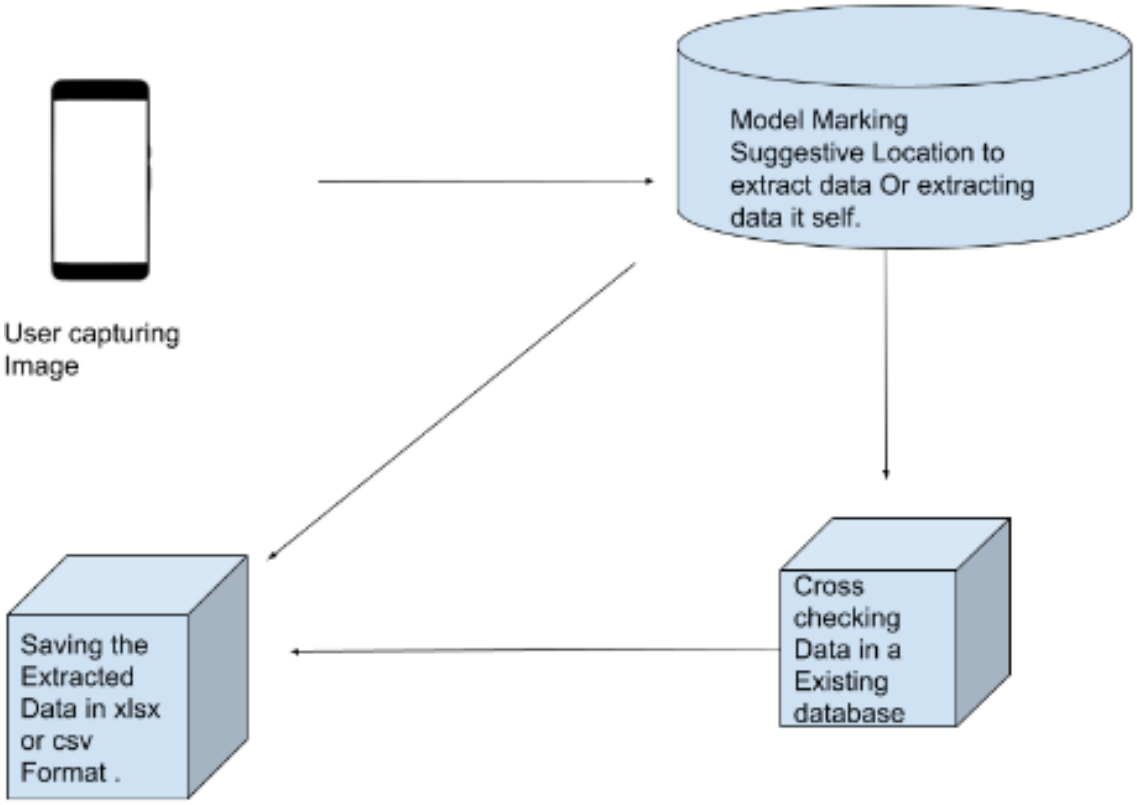


Figure 1.2: Steps of working process of the Application.

1.2 Research Problem

As days go by, the use of the digital environment has been exponentially increasing and inputting data in those digital environments manually is a monotonous labor . Such As, manually updating a datasheet with the information stored in a students exams answer scripts such as student id, section and marks obtained in questions. We are trying to automate such laborious processes by using either machine learning or deep learning. We choose machine learning techniques to implement such automation in a cpu based machine and deep learning for better performance and accuracy. Besides that, we hope to achieve that our models can easily detect the unique English handwriting of Bangladeshi people. We also hope to improve segmentation techniques, ultimately achieving better efficiency.

1.3 Research Objective

As previously mentioned, our research will be focused on finding an efficient approach for extracting data from form-like documents and storing the data in spreadsheets for further analysis. Thus, our research objectives are:

- Identify appropriate pre-processing techniques for captured images and implement.
- Compare different OCR models to find a suitable one for the task.
- Fix orientation of the documents for efficient data extraction.
- Identify appropriate segmentation techniques for extracted regions and fine-tune all segmentation faults for accurate recognition.
- Implement trained OCR model on said segmentations and accurately predict and store data.

Chapter 2

Related Work

Zhengchao et al.[3] combined CNN and RNN to recognize scene text. CNN was used to extract features from an input image. Different descriptors like VGG16, VGG19, ResNet34, ResNet50 were used for feature extraction. Now extracted data was sequential feature map and CNN has a disadvantage in such cases. This is where RNN which can extract sequential objects of arbitrary lengths comes in. After experimenting with different neural networks they managed to combine advantages of both CNN and RNN and claimed that a deeper CNN with deep descriptor will be more effective in predicting scene texts.

Shubham et al.[5] used CNN (convolutional neural network) model for recognizing handwritten texts. They used the MNIST dataset for both training and testing purposes. In their approach, they first pre-processed the data via normalization, rotating and reversing, input image filtering. With data processed and fit for training, they used the SEQUENTIAL model which consisted of 8 linear stacks of layers. Lastly, they used various optimizers available for keras to train their model. After experimenting with all the models,they found out Adamax, a first order gradient-based optimization method provided the most accurate results.Also they used 2 convolutional layers for maximizing accuracy. With Adamax, the trained model had an accuracy of 87,1 percent. Though other optimizers like Adadelta, Adam, SGD provided close enough accuracy, Adamax reduced the training time significantly. With better data pre-processing methodology the accuracy, efficiency of this model can be improved and for our work using a 2 convolutional layers model with Adamax optimizer provided by keras is a viable option.

Again, Himank et al.[7] proposed faster and efficient pre-processing methodology. Here geometric rectification, pre- processing, image detection, text extractions are key elements that are used for recognizing text data. Findings of their experiments suggest that though their approach reduces the time complexity and simplifies the issue at hand for untrained text fonts the approach seems to be lacking. However, with trained text fonts the accuracy rate of successfully digitizing texts is around 80 percent. We can improve the OCR model to make it adaptable to untrained fonts and use the proposed methodology in our endeavor.

Jyothi et al. (2020) [9], used OCR model to detect text and separate them into different graphical portions. Their proposed OCR model can classify different iden-

tity documents such as, passport, license into different categories. First of all, their model categorized the photos and derived information from the text extraction module and then the identification details were stored in the database. A new neural network based OCR engine LSTM(Long Short- Term Memory) is used to detect character patterns in this research. The research is mainly focused on the main project's purpose, technologies employed, used databases and the functional procedures. But they couldn't define a best solution for making the algorithm more efficient.

Anarghya et al. (2020) [10], worked on stroke detection and Hog transformation method. Their strategy presents a method towards character identification and recognition that combines the advantages of feature extraction methods associated with components distribution. The algorithm focuses on the clarity of text background segmentation. After reviewing some research papers and algorithms, they proposed integration of the Hog transformation and stroke detection method to achieve higher accuracy in localization and recognition in OCR technology. This approach highlights its efficiency in dealing with different text patterns, light and shadow conditions, and linguistic issues. Standard datasets have been demonstrated and improvement on previous OCR technology has played a significant role in providing a solution for real-time scene text recognition. This paper used MODI which can create difficulty in getting contents from books.

Yi Jiang et al.[13] proposed a text recognition algorithm to solve text segmentation difficulties, dictionary dependence using Attention mechanism and connection time classification.Here, low resolution images are upgraded using ESPCN into high resolution images. Afterward via multi-scale CNN, features are extracted. Later using Attention - CTC said encoded feature sequences are decoded. Traditional RCNN has great success in recognizing text but the training takes a significant amount of time and the calculations are large. Via the algorithm identifying the target is simpler and takes less time and calculation. In their work, they came to the conclusion that training models based on Attention CTC avoids the gradient disappearance problem of RNN. Proposed algorithm here can be advantageous if it can be implemented on android devices as effectively reducing time needed to operate would lessen the burden on the hardware.

Shruti et al.[15] used semantic segmentation and other pre-processing methodology to recognize mixed data (handwritten, printed texts) faster. In their work, they prioritize pre-processing data over improving conventional OCR models. Their approach focused on using any OCR engine readily available and getting optimal results by inputting fit data for training which they would get from their elaborate pre-processing techniques. First the input image will be pre-processed where binarization, noise reduction, slant removal, text alignment issues were performed. The resultant image was then processed using Liner removal, Gray Scale Conversion, Gaussian Blurring, Thresholding, 3 Channel Re-Conversion. Afterward, using U-net image segmentation was performed and on the output using OCR engines digital text was generated from handwritten/printed text. In their image processing module, there was a label isolation model that allowed them to segregate handwritten texts from printed text. This module can be used in our research for using

segregated printed text for recognizing and categorizing different labels. Also, their pre-processing approach is also very viable for our approach as we want to recognize texts into different segments based on labels. Their work focused on forms which are an example of mixed data while our work focuses on exam transcripts information page where student information and grading is provided in mixed data as well.

For scene text recognition, Xia et al. (2022) [16] suggested a Transformer-based encoder-decoder structure with a two-stage attention mechanism. A first-stage attention module that combines spatial attention and channel attention captures the overall placement of the text in the image at the encoder, and a second-stage attention module at the decoder precisely determines the position of each letter in the text image.This study suggests that this two-stage based strategy can improve recognition precision and more precisely determine the position of the text. In order to provide more robust features for the encoder, they also developed a multi-branch feature fusion module. This module can merge features from several receptive fields. According to this study, this framework is better for learning than the RNN-based STR approach and can speed up training. The encoder of this framework uses convolutional layers instead of linear ones because the input is an image, but it also adds that the original Transformer encoder's structure is kept.

Chapter 3

Methodology

3.1 Dataset

The dataset used for this OCR implementation project is entirely composed of handwritten digits of anonymous students ranging from 0 to 9 . The handwritten digits were carefully obtained from BRAC university exam papers, lending the dataset credibility and practical value. It consists of a total of approximately 10,000 handwritten digits. And all of those digits have been mixed up with MNIST data for better predicting performance of the models . The following MNIST dataset has about 60,000 variations of each digit. Image of every digit including ours was grayscale and binarized .

3.2 Model

3.2.1 CNN

A type of CNN, machine learning, is called either a convolutional neural network or a convnet. This special type of artificial neural network is used for different jobs and data types. CNNs have demonstrated impressive progress in a number of areas, the efficiency issue still has to be resolved immediately. Model storage and model prediction speed issues are two subsets of the efficiency problem[8].

Typically intended for image recognition and pixel data processing tasks. A CNN model uses matrix multiplication and other linear algebra concepts to find patterns in an image. Extensive Training Convolution, pooling and fully connected (FC) layers make up the three layers of CNN. The convolutional layer comes before the FC layer and ends before it.CNN becomes more complex when moving from convolutional layer to FC layer. With increasing complexity, CNN can recognize the most important regions of the image and more complex details, until it finally recognizes the object as a whole. Most of the calculations are done in the central component of the CNN, the convolution layer. The first layer of twist can be followed by another.in this layer, an examination of the image and its receptive fields occurs during the convolution process by a kernel or filter to identify any features present. The entire image is covered multiple times while calculating the dot product between input pixels and filters at each iteration. Spiral features are produced as extreme products from sets of points which ultimately convert images into numerical values

for appropriate analysis using CNNs. Pooling layers lose information but enhance efficiency within CNNs based on previously obtained feature maps before classifying them in FC (fully connected) layers where every activation unit connects with previous layer inputs. Each layer uses filtering/kernelling techniques analyzing different levels/details starting low-level basic functions then advancing towards sophisticated representations aligned with specific detected recognitions.The output generated after processing through all inverted/partially-detected images gets recognized finally by identifying individual objects or their description via FC Layers in completion.

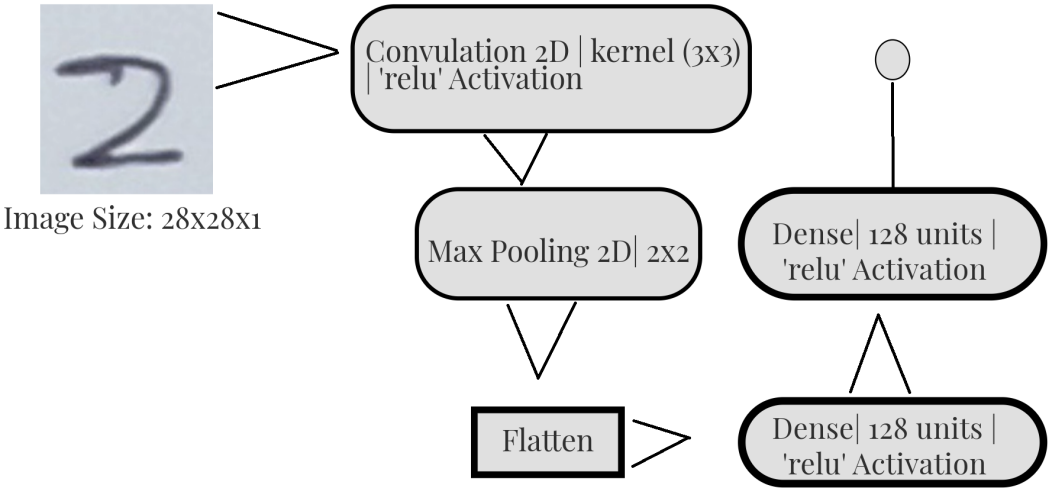


Figure 3.1: CNN.

During convolution, the input image is passed through several such filters. Each filter performs its task and passes its results to the filter of the next layer when it activates certain functions in the image. As tens, hundreds, or even thousands of layers are added, the processes are repeated because each layer can recognize different features. Finally, it can detect the whole object with image data processed by CNN's all layers.

3.2.2 KNN

The K-nearest neighbors algorithm, denoted as KNN or k-NN, is a supervised learning classifier that operates with nonparametric principles and utilizes data point proximity for prediction and classification purposes. The primary goal of the KNN method is to identify the K neighbors whose distance or similarity to the sample to be categorized is closest after first calculating the distance or similarity between the sample to be classed and the training sample of a given category[14]. It has diverse applications including regression tasks but mainly focuses on accurate results in the field of classification by taking into account the tendency of similar points to assemble together. Furthermore, discovering neighboring datapoints requires distance computations between surveying points and other relevant data sets assisting

it to recognize nearest ones effectively.This approach involves the creation of decision boundaries using distance measurements to divide survey points into distinct regions. The accuracy of classification for a given query point in the implementation of K-NN Algorithm depends on how many neighbors are analyzed. Neighbors with closely situated data belong to similar class groupings, leading to precise classifications by taking each other's similarities into account.To ensure accurate results while maintaining evenness in grouping per classes and selecting suitable radius parameters is crucial.As a result, reliable and understandable outcomes can be obtained when k=1. Because different values may result in over- or under-correction, the determination of k may require careful consideration. Larger values of k can result in a larger deviation and smaller variance, while smaller values can have a small variance and a large variance. The input data plays an important role in determining the value of k, as data with more noise or outliers will likely improve at larger values of k. The ability of KNN to handle complex decision constraints and non-linear relationships in data is a significant advantage in image classification. This is especially useful when working with complex image variations and patterns. Because KNN makes no assumptions about the distribution of the underlying data, it can be applied to a variety of image datasets where feature relationships may not follow a given mathematical model. KNN is also very efficient in handling multimodal and multiclass classification tasks, which allows it to be flexible in situations where images may belong to multiple classes or classes. Because it is non-parametric, it can dynamically adapt to changes in the dataset and is therefore robust to new and previously undiscovered image patterns.It is important to remember that despite the advantages of KNN, in some image classification scenarios, scalability and computational efficiency can arise, especially when the dataset size increases. For large data sets, calculating the distances between data points can be computationally expensive. Thus, the compromises between simplicity and scalability, as well as the unique characteristics of the image dataset, determine which algorithm is best.

3.2.3 SVM

Support vector machines (SVMs) are a supervised learning algorithm utilized in machine learning to tackle classification and regression tasks. In particular, SVMs excel at solving binary classification problems involving the separation of dataset elements into two groups. The most efficient classifier is determined via SVM, and support vectors play a significant role in this computation[6].

The objective of an SVM program is to identify the optimal decision boundary between data points belonging to different classes within high dimensional spaces called hyperplanes. To ensure effective differentiation, these programs strive for maximal margin distances- that quantify gaps separating related hyperplane points with their nearest opposing class counterparts accurately. In cases where complex or curved lines demarcate various feature space divisions rather than straight ones, nonlinear support vector machines use mathematical techniques such as transformation functions using kernel tricks on higher-dimensional datascapes suitable for determining borders more successfully—demonstrating critical usefulness across domains heavily reliant upon intricate datasets' analyses when attempting accurate

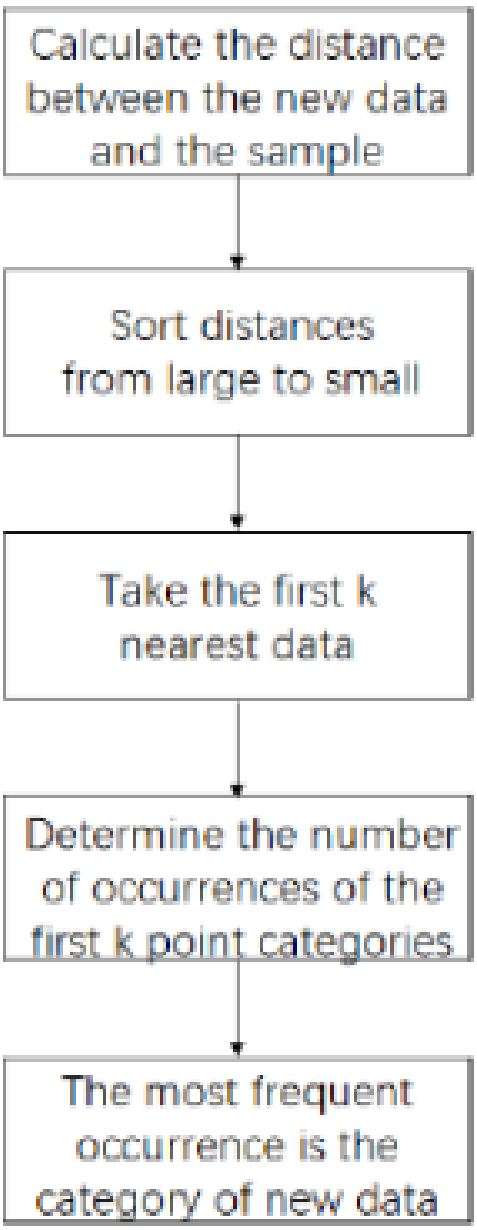


Figure 3.2: KNN[11].

modeling processes efficiently. The Support Vector Machine (SVM) paradigm has emerged as a highly effective tool for categorization. The most robust mathematical model for regression and classification is SVM[1].

SVMs work by transforming the input data into a higher dimensional feature space. This transformation can be used to classify a data set more successfully or to more easily find a linear separation. SVMs use a kernel function to achieve this. The basis function allows the SVM to implicitly compute the dot products between the transformed feature vectors and avoids expensive, pointless computations in extreme cases, instead of computing the coordinates of the transformed space.SVMs can handle both linearly and nonlinearly separable data. They achieve this by using different types of kernel functions, including a radial basis function (RBF) kernel, a polynomial kernel, and a linear kernel. SVMs can efficiently extract complex relationships and patterns from data thanks to these kernels. In the training phase, SVMs determine the ideal hyperplane in a higher-dimensional space, also called the kernel space, using a mathematical formulation. Because it minimizes classification errors and maximizes the margin between data points of different classes, this hyperplane is very important.Because it allows merging data from the original feature space into the kernel space, the kernel feature is essential for SVMs. The choice of kernel function can significantly affect the performance of the SVM algorithm. The characteristics of the data determine the best kernel function for a given problem.The following are some of the most commonly used kernel functions in SVMs:Linear kernel, data is mapped into a higher dimensional space where it is linearly separable using the simplest kernel function.Polynomial kernel, this kernel function can be used to transform data into a higher dimensional space if it is nonlinearly differentiable. It is more efficient than a linear core.RBF kernel, in SVMs this is the most commonly used kernel function because it works well for many classification problems.Choosing a kernel function for an SVM algorithm involves a trade-off between accuracy and complexity. Better accuracy can be achieved with more efficient kernel functions, such as the RBF kernel, than with simpler kernel functions, but training the SVM algorithm with them requires more computation time and data. But as technology advances, this is becoming less and less of a problem. After training, SVMs can identify which side of the decision boundary the new unseen data points belong to and classify them. The title of the class associated with the decision boundary and side is a result of SVM. Image classification tasks with a large number of features are suitable for SVM. High dimensional images are those where every pixel and feature adds to the overall image. SVM's ability to process high-dimensional data is essential for capturing nuances and variations in images and for finely distinguishing between classes.SVM is also known for its robustness against overfitting because it searches for a decision boundary that maximizes the margin between classes. This feature makes SVM able to generalize to new data, which is important for image classification because the model must perform accurately on different recent images. But it is important to consider the computational complexity of SVMand, especially when datasets grow. Training an SVM model can be computationally demanding, and choosing the right kernel function requires careful analysis of image data. However, SVM is a powerful tool for image classification tasks due to its ability to handle high-dimensional nonlinear data, especially when accurate discrimination of complex visual patterns is crucial.

3.2.4 YOLOv8

The latest model in the YOLO series is called YOLOv8. YOLOv8, an upgrade from YOLOv6, substantially enhances performance and renders the model quick, precise, and user-friendly[17]. With its five components—structural design, loss computation, training data improvement, training strategy, and model inference process—YOLOv8 builds upon YOLOv5, and it may be used to produce nearly any kind of visual direction[18].

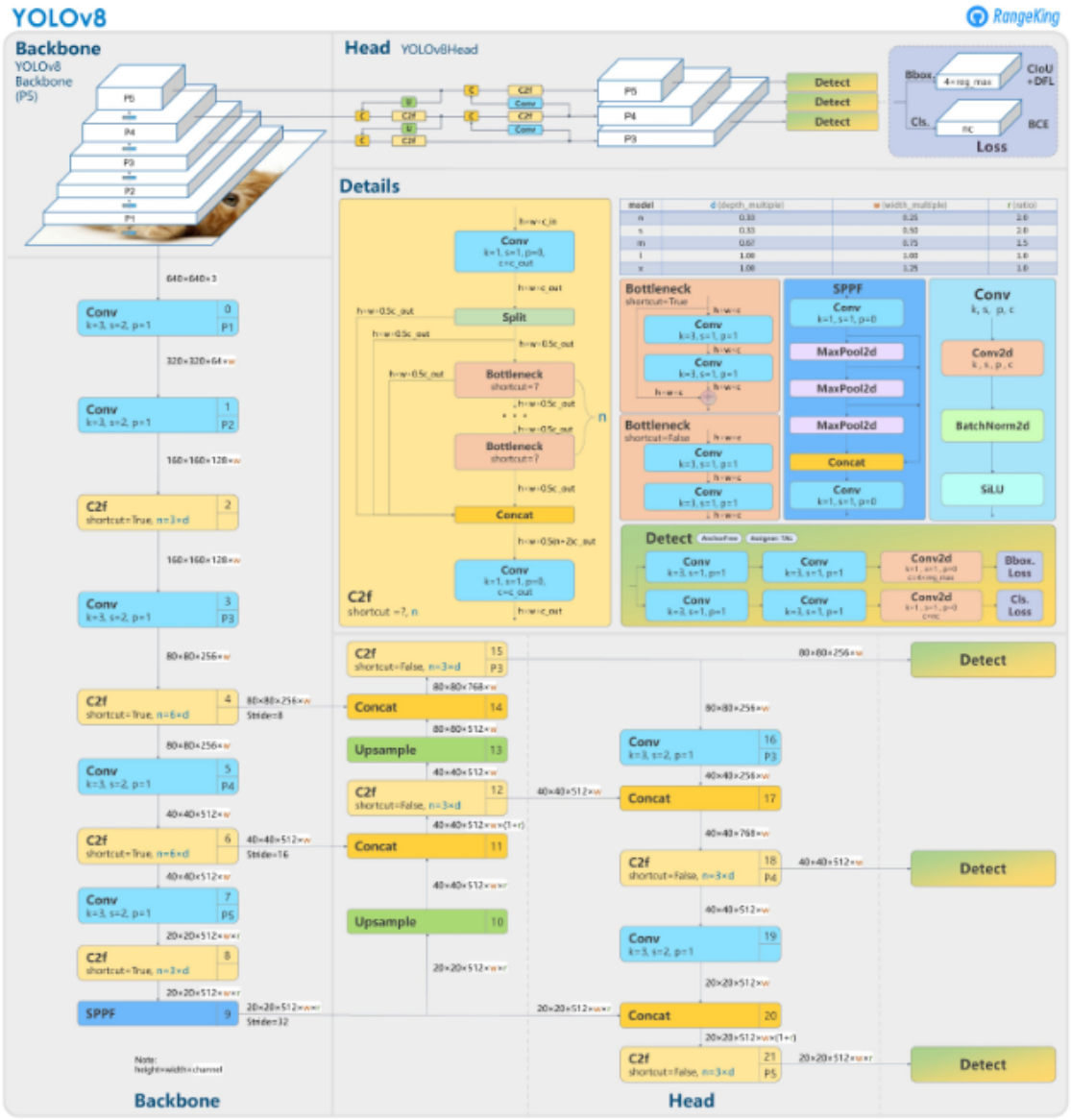


Figure 3.3: YOLO [19].

You Only Look Once, or YOLO, this model series got its name from - they can accurately predict every object in the image with just one forward motion.The main difference they introduced was how the YOLO models handled the problem. Rather than framing the object recognition task as a classification question, the study authors reframed it as a regression challenge (predicting bounding box coordinates). YOLO models are pre-trained using huge datasets like ImageNet and COCO. Thanks to this, they can now be both a student and a master at the same time. In addition to being faster to train, YOLO models can also achieve good accuracy with smaller model sizes. They are more accessible to developers like us because they can be trained on a single GPU.The latest version of these YOLO models is called YOLOv8 (early 2023). Some notable changes with its predecessors include the addition of C3 convolutions, anchorless detection, and mosaics. The YOLO (You Only Look Once) series has evolved into YoloV8, which focuses on real-time object detection in images and videos. It creates a grid of the image that predicts the bounding boxes and class probabilities for each object in the grid cell. YOLO models' strength lies in object location and detection, although these include classification to some level (giving class labels to detected objects).

3.2.5 SGDClassifier

SGDClassifier is an iterative model for large datasets[2].For machine learning tasks involving linear classification, SGDClassifier (Stochastic Gradient Descent Classifier) is a powerful optimization algorithm. An integral part of scikit-learn, this classifier efficiently trains models on large datasets using a stochastic gradient descent optimization technique. It works particularly well in situations where more computationally demanding gradient descent techniques are not possible. Because SGDClassifier supports a variety of loss functions, it is adaptable to a variety of tasks, including logistic regression and linear support vector machines (SVM). By iteratively updating the model parameters using a randomly selected subset of the training data, SGDClassifier achieves a balance between accuracy and computational efficiency. Due to its efficiency, scalability and versatility, the stochastic gradient descent classifier (SGDClassifier) is a very attractive choice for image classification applications. SGDClassifier's stochastic optimization technique works well for image classification, where datasets are often huge and high-dimensional. The technique is computationally efficient and well-suited to handling the complexity associated with image collections because it processes the training data in small, random sets. Support vector machines (SVMs), logistic regression, and linear classifiers are examples of linear models that SGDClassifier can handle, which is one of its main advantages in the image classification space. Its adaptability allows practitioners to use multiple linear models according to the characteristics of the image data and the requirements of the classification task. In addition, the stochastic properties of the algorithm facilitate its traversal over large feature spaces, which makes it suitable for scenarios where the number of pixels or features in an image is important. The adaptive learning and continuous learning of SGDClassifier makes it particularly useful in online learning environments where models need to be updated frequently as new information emerges. In addition, the SGDC classifier can converge quickly due to its efficient optimization procedure, making it a desirable choice when computational resources are limited. SGDClassifier is a practical and efficient choice for

professionals dealing with diverse and large-scale image data sections because it can process large-scale datasets and provide relatively accurate results, especially when used with linear image classification models.

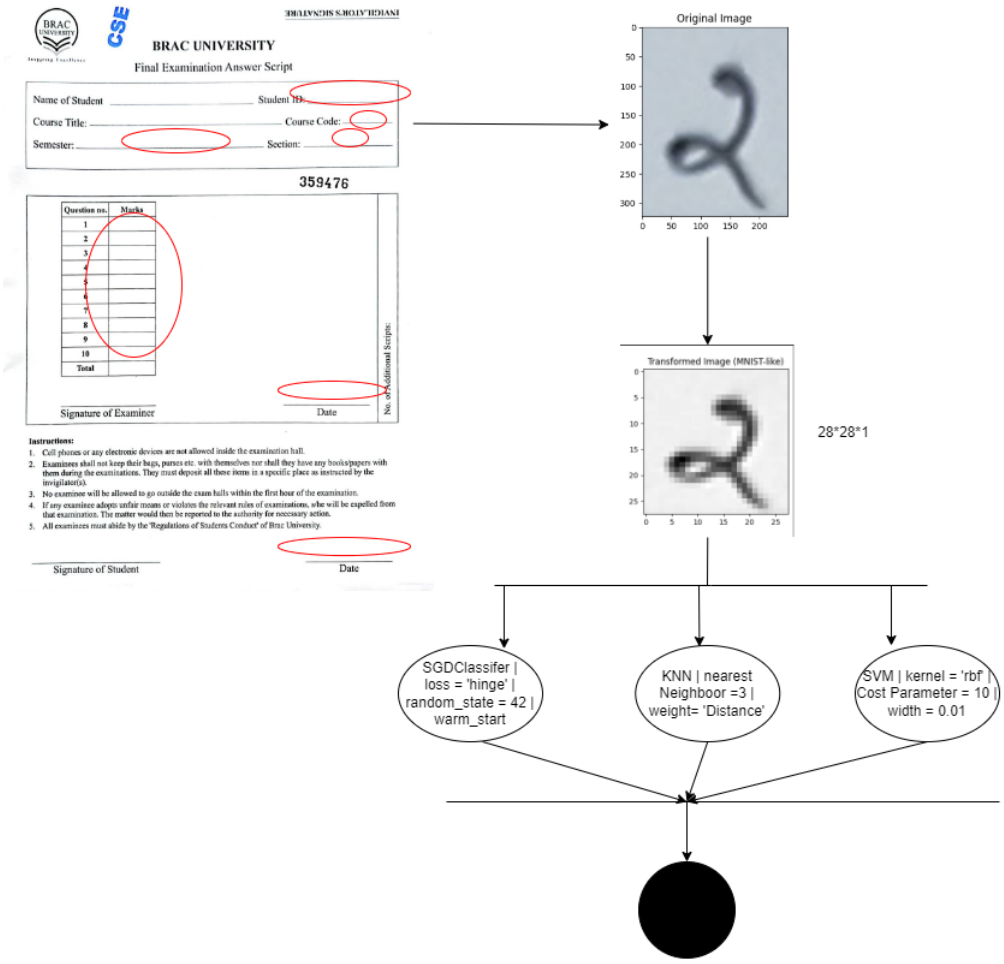


Figure 3.4: SGDC.

3.3 Proposed Methodology

For the purpose of training and testing our dataset, we have tuned some of our models. Like for KNN we have set the nearest neighbor to 3 and weights set to distance. In particular, this setup can be useful for handwritten digit categorization. When it comes to digit identification, the image's patterns and characteristics are essential in identifying the digit. The model becomes sensitive to local patterns and subtleties unique to each digit by taking into account the three nearest neighbors and assigning greater weight to those that are closer in feature space. This is particularly useful for handwritten numbers, since different writing styles may be used and little details may contain a lot of information. The 'distance' weighting procedure further makes sure that the contributions of nearby data points are weighted in accordance with their closeness, highlighting the significance of more pertinent and comparable instances in producing precise forecasts. This can work especially well for handling slight variances in handwritten digit presentation or writing style variations. By utilizing the local correlations and patterns found in the dataset, the KNeighborsClassifier setup with k=3 and distance-based weighting can, all things considered, provide a reliable and adaptable method for handwritten digit recognition.

Then for SVM we used Gaussian kernel along with regularization parameter set to 10 and Gaussian width set to 10. By efficiently converting the input space into a higher-dimensional space, the kernel enables the SVM to identify intricate, non-linear correlations in the data. Since the innate patterns and differences in writing styles could not be linearly separable in the original feature space, this is important for the classification of handwritten digits. The classifier can fit the training data more freely with a greater value of C, perhaps catching complex patterns. A larger C value would enable the model to more effectively accommodate the intricacies of various digit representations, which is important for handwritten digit classification when identifying minute differences in writing styles is essential. The Gaussian kernel's width is set by the gamma=0.01 parameter. The decision border is smoother when the gamma value is less since it indicates that each training sample has a greater effect. This helps guarantee that the model generalizes effectively to new data and helps reduce overfitting, which can be advantageous for handwritten digit categorization. To sum up, the SVM configuration with an RBF kernel, C=10, and gamma=0.01 works well for classifying handwritten numbers. It is a strong option for identifying various writing styles in digit pictures because the RBF kernel allows the model to capture non-linear correlations, the larger C value allows for greater adaptability to complicated patterns, and the smaller gamma value helps minimize overfitting.

Using specified parameter values, the SGDClassifier initializes a Stochastic Gradient Descent (SGD) classifier (loss='hinge', max_iter=1, random_state=42, tol=None, warm_start=True). Let's dissect this arrangement and talk about why it could work well for classifying handwritten digits. The hinge loss function, which is frequently used in conjunction with support vector machines (SVMs), is specified via the loss='hinge' option. The hinge loss works particularly well when trying to optimize the margin between various classes and is a good fit for binary classification

applications. The hinge loss can aid in the creation of a decision boundary that best divides the classes when it comes to handwritten digit classification, where the objective is to differentiate between various digits. The maximum number of iterations (epochs) that can be used in the training process is set to one by the max_iter=1 option. This suggests that the training data will only be shown to the model once. This could be helpful in situations requiring a little amount of CPU power or in online learning scenarios where the model needs to be updated gradually with fresh data. It's crucial to remember that a model may need more iterations to get an ideal solution if this low value is insufficient, and more iterations may be required for improved performance. Reproducibility is guaranteed by setting the random seed to 42 using the random_state=42 argument. This implies that the classifier will provide consistent results when run with identical data and parameters, which is crucial for reliable testing and assessment. The tolerance for convergence is specified by the tol=None option. If you set it to None, then regardless of whether the model has converged, the training process will go on until the maximum number of iterations (max_iter) is achieved. In situations when exact convergence is not a crucial factor, this is an acceptable option. Building on the prior technique, the model may be trained progressively with the warm_start=True option. This makes it an appropriate option for handwritten digit classification tasks that can require ongoing updates and additions to the training dataset. It is especially helpful in situations when the model has to be updated with new data in a dynamic or online learning context. In conclusion, handwritten digit classification tasks may benefit from the configuration of the SGD classifier with hinge loss, one iteration, and warm start. This is particularly true in situations where computing resources are constrained or when the model must be updated gradually over time with fresh data.

After that, the CNN model may be built sequentially thanks to the Sequential() method, which initializes a linear stack of layers. First, we have a convolutional layer represented by Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)), which has 32 filters (sometimes called kernels) of size 3x3. This layer's purpose is to find certain patterns and characteristics in the handwritten digit pictures, which have 28 by 28 pixels. The model gains non-linearity via the 'relu' activation function, which helps it learn intricate representations. Max pooling is carried out by the next layer, MaxPooling2D(pool_size=(2, 2)), which reduces the spatial dimensions of the representation while keeping the most crucial data. By doing this step, the model becomes more resilient to changes in finger location and computationally efficient. To prepare the output of the preceding layer for input into a highly linked layer, the Flatten() layer flattens it into a one-dimensional vector.

The following fully connected layer has 128 neurons and a rectified linear unit (ReLU) activation function. It is called Dense(128, activation='relu'). High-level abstractions and intricate linkages in the data are further captured by this layer. Dense(10, activation='softmax'), the last layer, is a densely linked layer made up of 10 neurons, which represent the 10 digit classes (0 through 9). The network's output is transformed into probability distributions using the 'softmax' activation function, which yields a likelihood for each digit class. Convolutional and densely linked layers of the CNN architecture, together with the right activation functions, pooling, and flattening, are, in short, ideal for collecting hierarchical characteris-

tics in handwritten digit pictures. 'categorical_crossentropy' loss function and the 'adam' optimizer are good options for effective training and precise classification. The best model is retained for later use thanks to the model checkpoint.

For YOLOv8, we have used Adam optimizer along with the basic model.

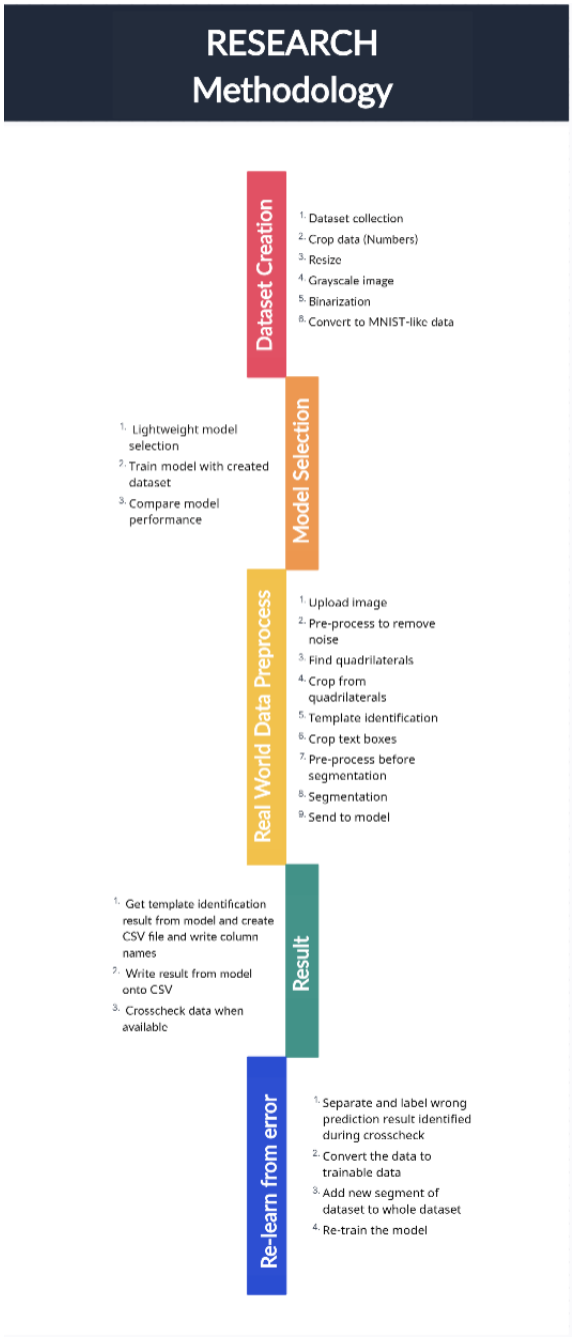


Figure 3.5: Research Methodology.

3.4 Workplan

This research aims to train its own custom model after gathering appropriate amounts of data and applying appropriate processing methods so the model will run efficiently and accurately.And then a real world deployment will be done as a contribution. So, the work plan for this research will be :

- Template format: For the time being, we aim to apply this model on BRAC university exam script as real world deployment. So, we will gather this template and its variations.
- Text data gather: We aim to gather our own dataset. In this case, we will gather marked answer scripts and gather the data from that. We will maintain confidentiality as it contains sensitive data.
- Template pre-processing: Before using it into a Machine learning model, we want to process it to cut out unnecessary areas such as logos, images etc.
- Data pre-processing: Same as template, we want to improve the quality of the characters from our data set. So, we will add a few processing methods.
- Model implementation: With the dataset created and processed, we will apply it on models found based on our literature review.
- Testing and Evaluation: In order to measure the performance of the model we will use different real world documents for testing. Effectiveness will be measured with appropriate metrics to ensure user satisfaction in real world scenarios.
- Cross-check method: If the model can not identify a particular scenario, we will save that event and try to label that from an existing cross checking database if it exists or save the edit being done. Then add that case back into the database so the same mistake won't be made two times.
- Deployment: With appropriate model design and evaluation with real world tests, we will deploy it as an appropriate form with security so that teachers can use it as a tool to upload student data seamlessly.
- Maintenance: The human handwriting is different from one another and it continues to differ as time progresses. To ensure the performance of the system and handle future gimmicks, a scheduled maintenance will be done to make appropriate tweaks.

Chapter 4

Dataset Creation

4.1 Image Collection

In order to create a representative and diverse dataset, a thorough and systematic procedure was used to gather samples from various sources found in the academic archives of the institution. We were able to obtain access to an extensive and diverse set of previous exam papers with the express consent and collaboration of the university administration. These papers covered a wide range of courses, subjects, and examination types, including final and midterm papers across different faculties. This systematic strategy performed a critical role in guaranteeing that our dataset included a wide range of handwriting styles, which is essential for the reliability of OCR systems. We ensured that the handwriting samples had a wide range of variability by including papers from several academic subjects. This diversity goes beyond only the sorts of writing pupils produce; it also includes variances in the writing implements that they use, such as different kinds of pencils and pens, which can have an impact on the clarity and thickness of their strokes. Moreover, the addition of different ink colours enriches the dataset. Although the most prevalent colours are black and red, there are also additional colour like blue, which gives OCR systems a more difficult and realistic scenario to understand. This variance is significant because it replicates real-world scenarios in which OCR systems could run into issues with various ink colours. In a study[12], upon inspecting language practice of three different origin, researcher have concluded that there are significant diversity on their writing of English letters. And thus we have come to a conclusion that people of Bangladesh has unique English handwriting and our collected data will help the model recognize said handwriting well.

Furthermore, several types of paper are utilised for these exam sheets, such as recycled or slightly tinted papers, in addition to typical white sheets, which may affect the handwritten numbers' backdrop contrast. A further degree of difficulty in digit recognition arises from this diversity of paper types, which replicates the various scanning settings and paper characteristics that OCR systems may face in real-world applications. The students that took these tests across multiple semesters represented a wide demography that contributed to this dataset. This feature of the dataset makes the handwriting samples more representative and guarantees that they aren't biassed towards any one course or student group. Exam papers from several courses and semesters are included in the dataset, which makes it a more

comprehensive resource for training and assessing OCR algorithms since it covers changes and trends in handwriting styles over time.

To summarise up, the handwritten digits in our dataset were primarily sourced from this large collection of documents from the academic archives of the university. This approach to data collecting guarantees that the dataset is both diverse and sizable, closely reflecting the real-world situations where OCR systems are used. As a powerful and demanding resource for creating and testing sophisticated OCR systems, the dataset captures a broad diversity of handwriting styles, utensils, ink colours, and paper kinds.

4.2 Image Processing

The dataset was divided into training and validation subsets, allowing for model evaluation and development. The MNIST dataset structure served as inspiration for the dataset's construction, but a special method suited to the context of handwritten numbers taken from university test papers was also used. Using custom scripts, the procedure started with the difficult extraction and snipping of individual digit images from the exam papers. The rigorous manual cropping process guaranteed the separation of every handwritten digit while preserving the uniform size and orientation consistent with the standard format of the MNIST dataset. These images may have varied in size due to variances in writing size and paper alignment. A robust set of preparation procedures was carefully designed to ensure the consistency and interpretability of the dataset. The standardization of the data across different samples was made possible in large part by normalization procedures. In order to reduce any biases caused by differences in writing styles and ink intensity, this involved scaling the pixel values of the handwritten digits to a common range. Additionally, contrast enhancement techniques were used to maximize the readability of the numbers, guaranteeing consistency in their appearance and supporting the extraction of features.

Additionally, to maintain uniformity across the dataset, each cropped image was meticulously resized to align its proportions to a predetermined standard, such as the MNIST collection's convention of 28x28 pixels. Initially the cropped photos were uniformly enlarged utilizing interpolation methods to establish a consistent format. Then image processing libraries and methods used to maintain the aspect ratio and content integrity of the handwritten numbers. Moreover, The coloured images were converted to grayscale and a binary format to reduce complexity. Binary images only have two values: 0 denotes the absence of information or background, and 1 denotes the presence of information or foreground (for the handwritten numeral). This simplicity improves processing efficiency by streamlining subsequent computational procedures. The emphasis changes from grayscale or color gradients to a distinct delineation of the digits' shape and edges. This makes feature extraction easier, allowing models to focus on important structural aspects for classification or identification tasks. Running consistency tests are conducted on the dataset to detect any anomalies or discrepancies in the recognized digits' alignments. This may include manual visual inspections, statistical analysis of digit distribution, checking for missing or duplicate entries, and assuring digit size and orientation consistency.

By methodically carrying out these verification and alignment stages, the dataset's consistency is ensured, proving the accuracy of digit identification and alignment throughout the dataset. This thorough technique ensures a high-quality dataset, laying the foundation for strong model training and precise OCR implementation.

Chapter 5

Implementation

5.1 Library List

The libraries that were used for our implementation are: matplotlib, skimage, sklearn, numpy, pickle, Seaborn, os, csv, tensorflow, keras, PIL, OpenCV. Other than the libraries some additional packages and modules were also used for our implementation which are: attrs, backcall, bleach, cycler, decorator, defusedxml, entrypoints, imageio, importlib-metadata, ipykernel, ipython, ipython-genutils, ipywidgets, jedi, Jinja2, joblib, jsonschema, jupyter, jupyter-client, jupyter-console, jupyter-core, kiwisolver, MarkupSafe, mistune, nbconvert, nbformat, networkx, notebook, packaging, pandocfilters, parso, pexpect, pickleshare, Pillow, prometheus-client, prompt-toolkit, ptyprocess, Pygments, pyparsing, pyrsistent, python-dateutil, PyWavelets, pyzmq, qtconsole, QtPy, scipy, Send2Trash, six, terminado, testpath, threadpoolctl, tifffile, tornado, traitlets, wcwidth, webencodings, widgetsnbextension, zipp.

5.2 Data Pre-processing

For our dataset we used a combination of MNIST dataset and our primarily collected dataset. As our primary dataset was lacking in volume, we combined it with MNIST dataset and divided it into training, testing and validation sets. For training, testing and validation we divided it into respectively 80%, 10% and 10%.

For data preprocessing, we first resized our images to lower computational cost. Then, we converted the images into grayscale and later converted them into binary images. Afterwards, we did some model specific pre-processing steps. For example, in the case of CNN model after all the previous steps, we additionally set the data type to 'float32' and also normalized the pixel value to range between 0 and 1 instead of 0 and 255. For YOLOv8 model, instead of resizing the image to 28*28, we resized it to 32*32.

5.3 Model Selection

We had shortlisted five machine learning and deep learning models from our literature review for our task. However, we would only use one of these models for designing our interface. Thus, we trained all five models with our combined dataset and compared their results. Out of all the models, we chose the one with highest accuracy for our implementation.

5.4 Training Model

We trained all five models individually with our combined dataset. Firstly, for training CNN models, we added a few custom layers in addition to base layers as discussed in chapter 3. With our combined dataset we trained the modified version of CNN model. We set the batch size to 256 and epoch to 20 for this model. For CNN, we used Adam optimizer and set the loss function to 'categorical_crossentropy'. Then, for the YOLOv8 model we loaded a pre-trained model as it is a transfer learning model and set the epoch to 8 and batch size to 16. Same as CNN we used Adam optimizer for YOLOv8 as well. For KNN, we set the weight to 'distant' and n_neighbour to 3. For SVM, we set the kernel function to Radial Basis Function, set the regularization parameter to 10 and gamma value to 0.1. Lastly, for SGDC we set the loss function to 'hinge', random state to 42, tol to None, number of iterations to 1 and enabled warm start capability.

After training all the models in a given manner and comparing their accuracy, we found that YOLOv8 had the best outcome of all. The result of our findings will be explained in detail in chapter 6.

5.5 Creating Model Interface

5.5.1 Data Insertion

The model interface is designed in a manner that would allow users to scan and extract data from multiple documents at a time. For our purpose, we choose google drive to store our input images. Users can upload from one image to any number of images they want in said google drive. Our interface will then connect and import all the input images to our environment.

5.5.2 Fix Orientation and Extract Data Region

We know in form-like documents there are boxes that users use to fill out their information. In some documents, these boxes are profound and in some cases they are not just like in Fig 5.1 and Fig 5.2. No matter what type of documents it is, one thing is very evident that a certain type of documents will always have a fixed dimension. To elaborate, if there is an information table in the document that asks for users name, id, course name, section, course code every copy of the said kind of documents will have the same height and width in the exactly same position. Our goal was to fix the orientation of all input images so that we can extract regions containing necessary data.

Question no.	Marks
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

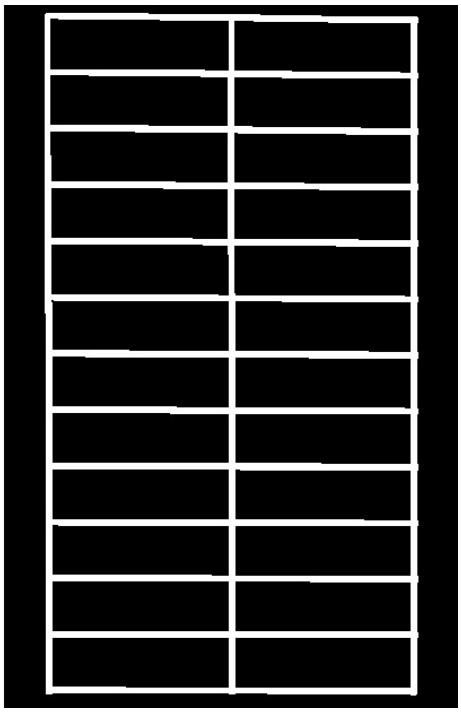


Figure 5.1: Profound Box.

Name of Student _____ Student ID: _____

Course Title: _____ Course Code: _____

Semester: _____ Section: _____

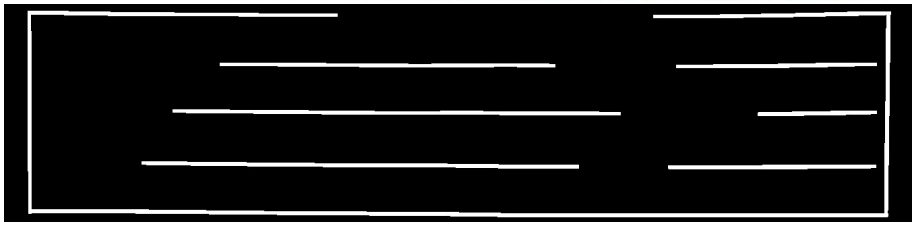


Figure 5.2: Unprofound Box.

There are mainly two known ways to fix document orientation. One is using machine learning and another is using image processing. In our experiment, we used the latter one. There are few steps to getting a perfectly oriented document image which are pre-processing, finding quadrilaterals, extracting images using said quadrilaterals points. We will be using Fig 5.3 for our demonstration.

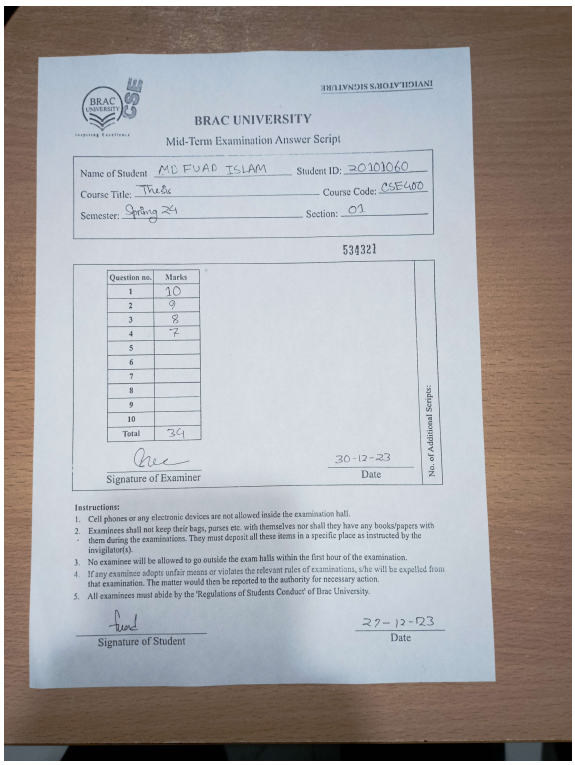


Figure 5.3: Test Script.

Starting with pre-processing, it is necessary for achieving a clear, noise-free image that will be less resource intensive and accelerate computational capability. Firstly, we started with resizing the image. By resizing the image, we reduce computation required for each image. However this also reduces details as well. Secondly, we implemented denoising methods. Denoising will reduce noise from images, making its features more prominent. Images taken with poor camera quality or captured in a low light environment tend to have more noise which impairs the judgment of models used to process said images. For denoising we used opencv library function "fastN1MeansDenosing". In said function, non-local means denoising algorithm is used to detect similar patches of pixel, as similar patches tend to have similar noises. The function ensures noises are removed while preserving the details of the image. In Fig 5.4, we have given a side by side comparison between the real image and denoised image. Lastly, we implemented Otsu's thresholding function. For thresholding, we first converted the three channel image to one channel image. Then we used the new gray scaled image for thresholding which provided us with a binary image. This marks the end for pre-processing the images, in Fig 5.5 we have shown a comparison of real world image vs processed image.

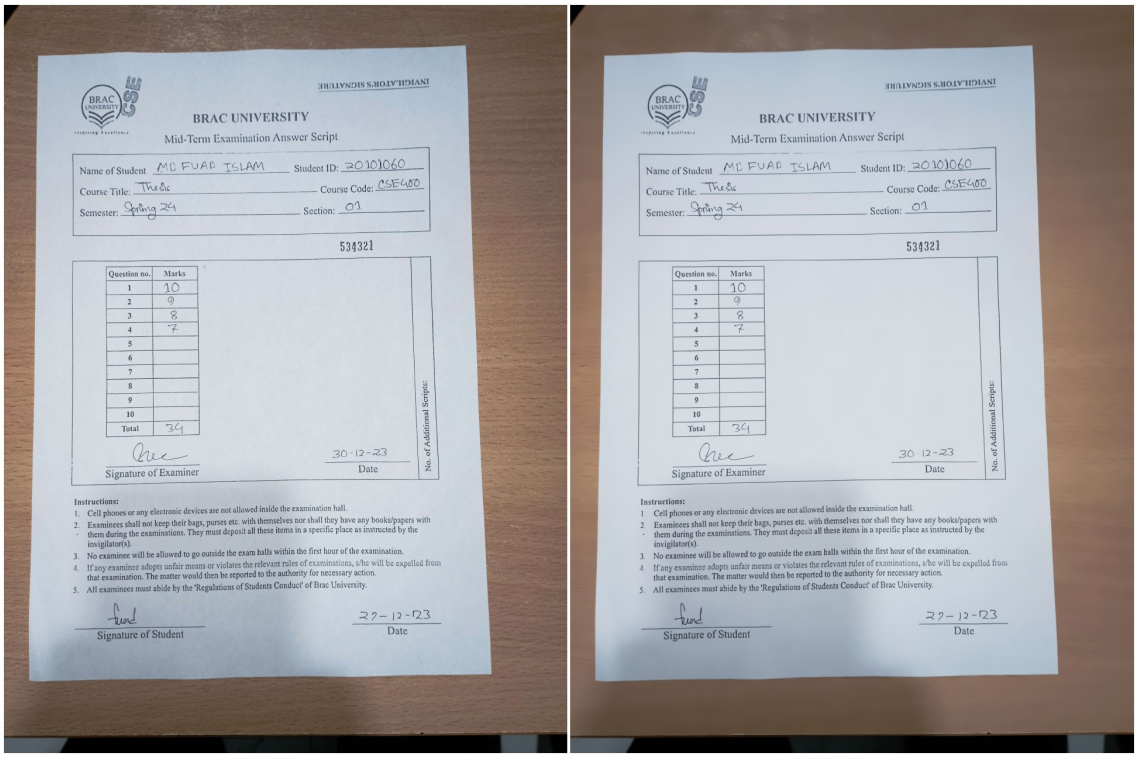


Figure 5.4: Original Image VS Denoised Image.

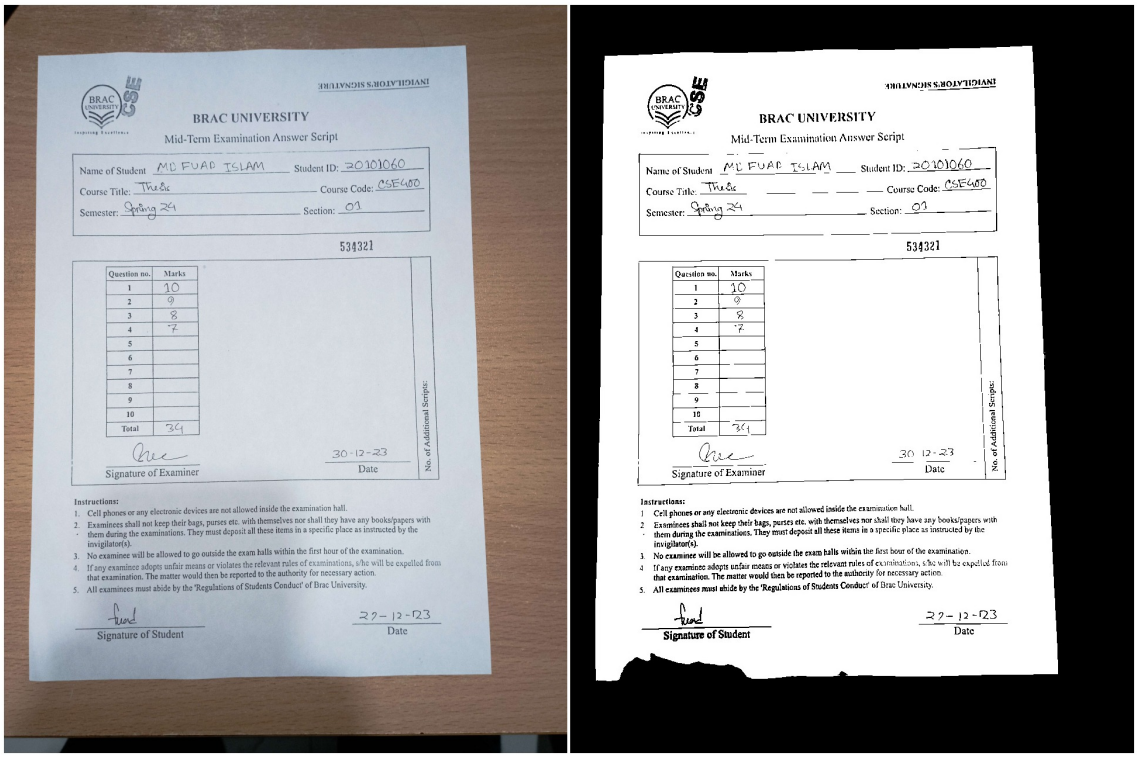


Figure 5.5: Original Image VS Processed Image.

The second step in our task is to find quadrilaterals. We had used hough lines for doing so. Again, we had to start with pre-processing the image. However, this time the pre-process was different than the first time. Our goal was to find four lines, now the text written in the document would be interfering with the task. We removed the text/details the document had so that we can find the lines. We could not include this in the first pre-processing method as we need the text/details for future steps. Thus, we will be doing it in this step. To remove the details within the document we used morphological transformations. In Fig 5.6, we have shown an example of how the image will look after processing. Now, to detect the page parameters we used another algorithm called canny edge detection. Finally, using the output from the last algorithm we can find hough lines[4]. We used opencv functions called HoughLines and HoughLinesP for getting hough lines. In Fig 5.7, an example of hough lines created from a test image is shown. As you can see in the image the lines form a quadrilateral. However, there are certain cases where the images will fail to form a quadrilateral mainly due to light source issue, overlapping documents. In Fig 5.8, two images have been doctored to visualize the problems that cause the failure. In our demonstration the image used does not have any said implication and hence was successfully processed. Lastly, to calculate the extraction points we need to find intersection points. But the task is not as easy as it seems, as there is more than one line for our perfect image as well. So, we checked the intersection angle. If the angle is 90 degrees or closer we count them as correct and so on. By doing so we get a few intersection points for all four corners. We used KMeans algorithm to find the quadrilateral in the end.

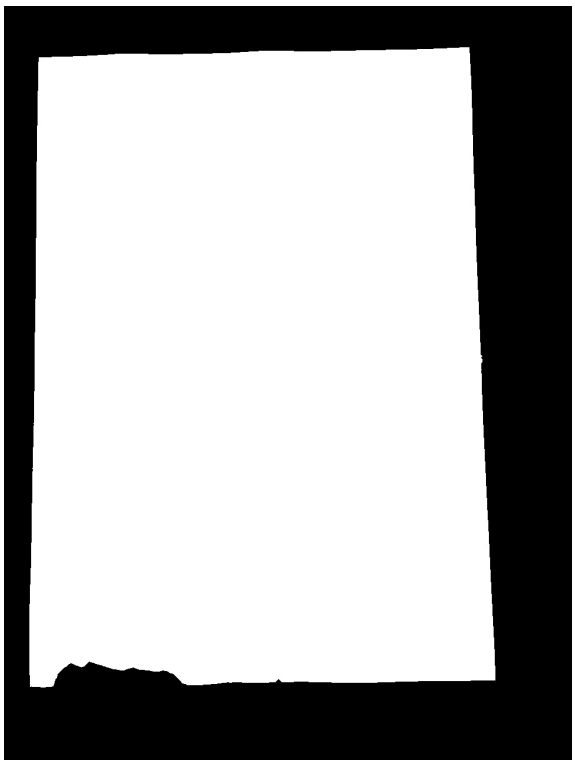


Figure 5.6: After Morphological Transformation Closed Operation.

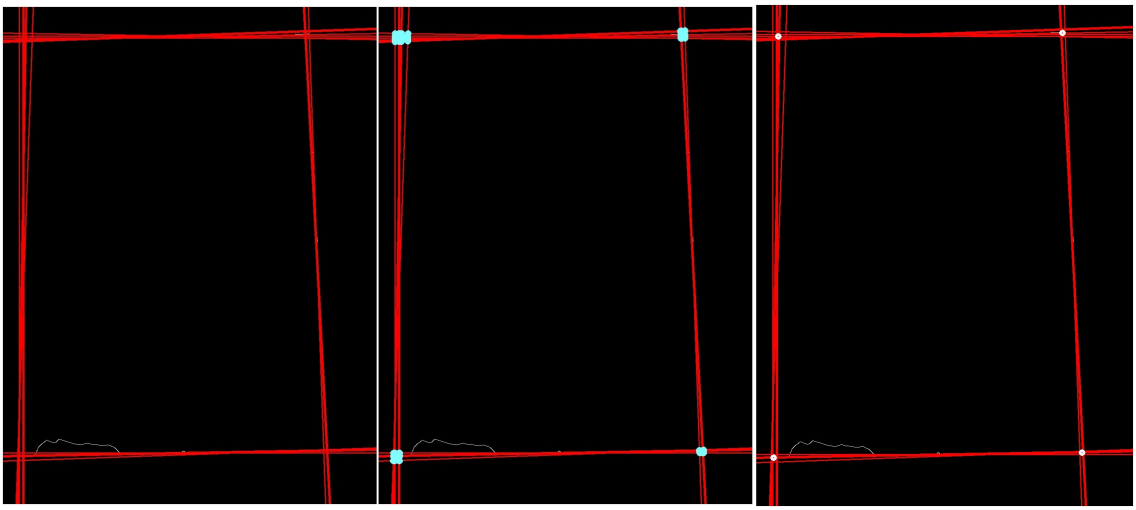


Figure 5.7: i. Hough Lines; ii. Hough Lines with Intersection; iii. Hough Lines with Grouped Intersection

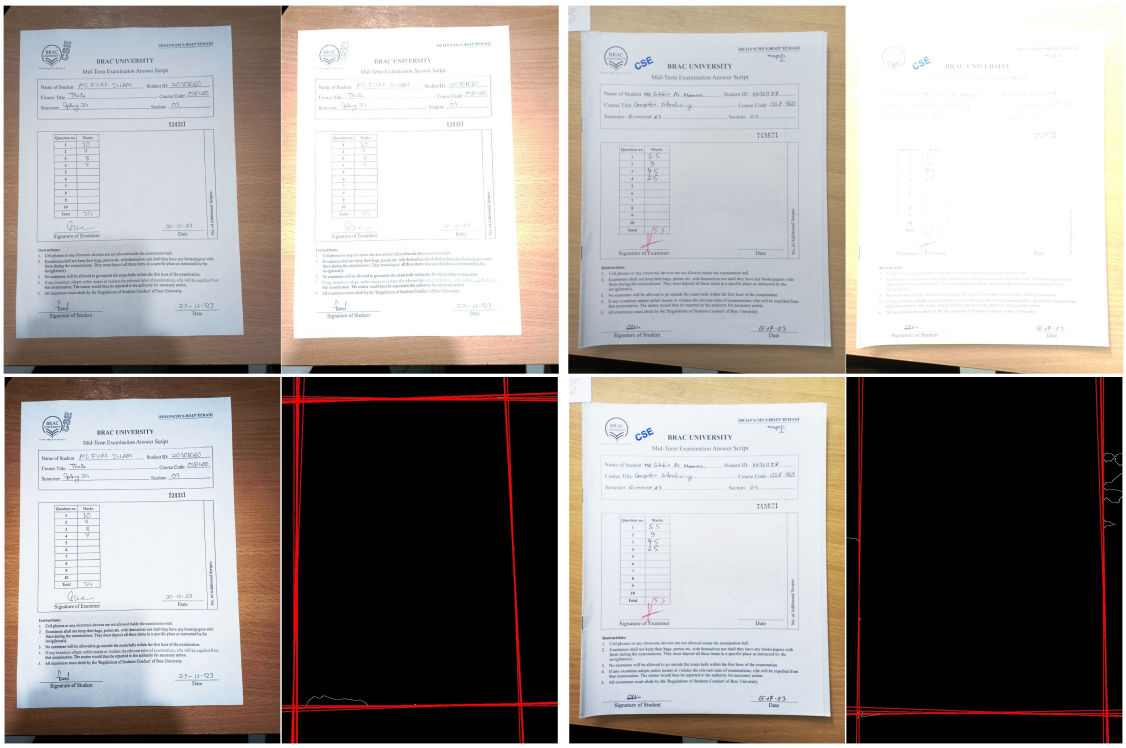


Figure 5.8: Problems while Drawing Hough Lines.

For the third step, with the quadrilateral found, the only thing left is to extract the image. We used opensCV's warp perspective function.
With the image correctly orientated, we can now extract the necessary information. As previously mentioned, all form-like documents of the same structure have the same dimensions. So, for our next task, we just need to specify the regions where data can be found. In Fig 5.9, we see the final image after fixing orientation. With the regions selected we can loop the algorithm to be able to extract regions from as many as scripts we want. In Fig 5.10, we can see examples of extracted regions.

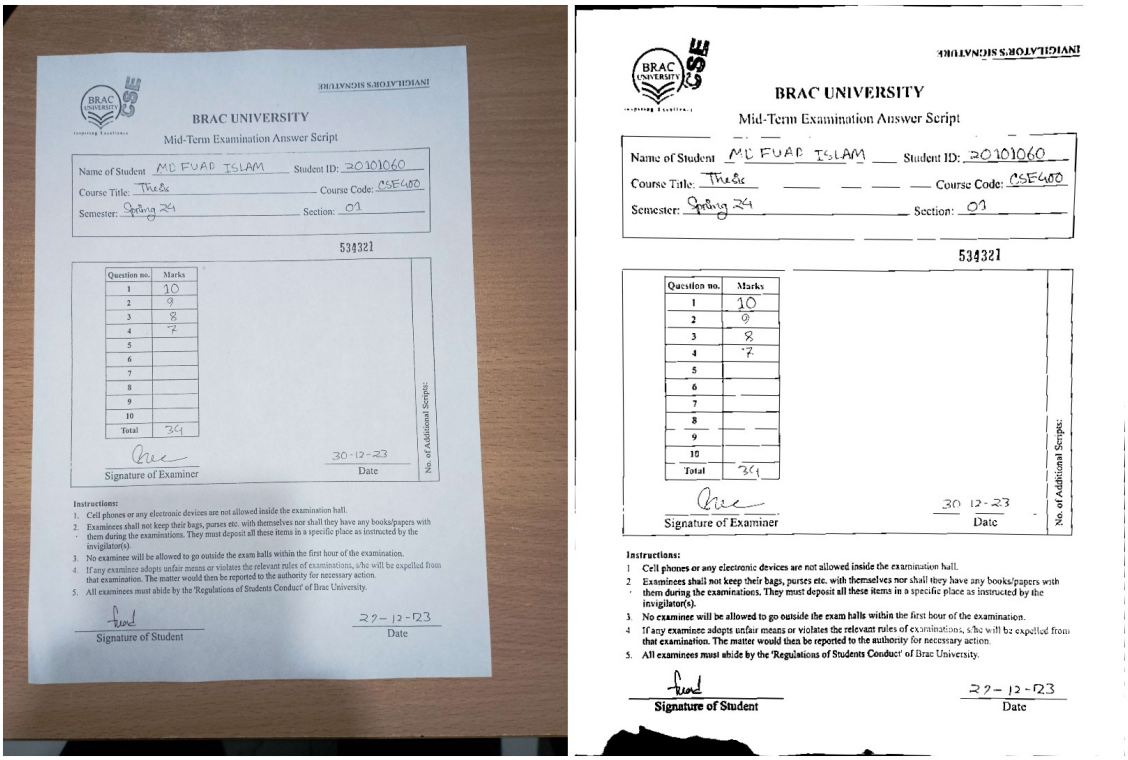


Figure 5.9: Comparison Between Original Image and Correctly Oriented Image.

5.5.3 Segmentation of Extracted Data Region

From the previous stage, we got cropped images each containing either student id, section or question marks. In this stage, we targeted these cropped images to segment them into individual digits. So, for student id there would be 8 segmented images, for section there will be either 1 or 2 segmented images and same goes for question marks.

We used contours to segment each image but before using contours we needed to prep the image. At this point all the cropped images are already processed once as a whole, however this time pre-process was done for individual crops and their end result was also varying. We also set up a function that would remove unwanted segmented images as well for improved performance.

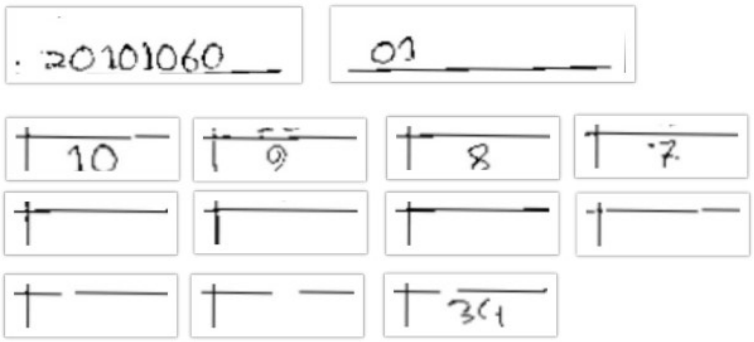


Figure 5.10: Extracted Images Preview.

We started our cropped image pre-processing by removing underlines or unwanted dots. In all form-like documents to specify each data block for the humans, there are guiding boxes or underlines which suggest that one should either write certain information in said boxes or over the underline. While extracting said certain information, our algorithm may as well parts of this underline/boxes or whole of it. So, our first step was to get rid of said lines. Firstly, we took the cropped image and did grayscale conversion, binarization. Afterward, we initiated a kernel matrix. The purpose of this kernel matrix was to extract features, in our case horizontal lines. We then performed a morphological transformation (MT) operation to find the lines. The kernel matrix was used to operate the MT operations for line detection. Upon detecting lines, we computed its contours. By having contours we could successfully create a border around the lines. We then run an algorithm which would replace the lines pixel value to be black which basically removed our lines. With the unnecessary lines gone, we further processed the cropped images by grayscale conversion and binarization. And again, we computed contours of the images. However, this time the contours were computed so that all individual digits borders would be detected like it is showcased in Fig 5.11. Now, we used an opencv function "boundingrect" to get the top left x,y coordinates and also height (h), width (w). Using each of these x,y,w,h for each of the digits, we were able to segmentize the digits. Then, we further cropped the images to get individual digit images and also store them in the google drive for further progress. Also, we stored the height and width of these individual digit images and the reason behind will be explained later.

As you can see in Fig 5.12, though the digits were segmented there still existed some anomaly. Our further progress could not have been possible until these anomalies were executed. Thus, we designed a secondary function that would flush these poor segmentation out. In our second function, we first calculated the pixel count in all the images. Now, all the cropped images will vary in height and width and henceforth in the pixel count as well. Also, a normal number ranging from 0 to 9 will

Figure 5.11: Highlighted Digits for Segmentation via Contour.

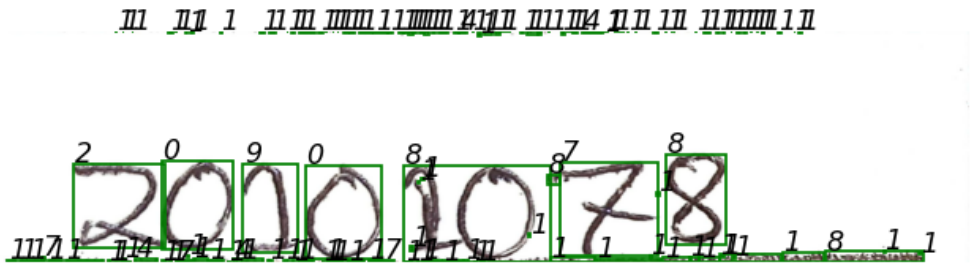


Figure 5.12: Segmentation First Phase.

always take a minimum amount of pixels which will be greater than the dots. Thus, we introduced a threshold. The threshold was designed so that any digit image pixel would be greater than the threshold value and all dots images threshold would be less than the threshold value. The second function would essentially compare the pixel count and store the outputs in a secondary location. For our other kind of anomaly removal, remember the height and width we stored on our initial function, we used them. We calculated the average height and width of the segmented images from these stored values and found the ones that are greater than the average value. We then segmented them in half and stored the new segmentation. Thus, we removed the segmentation anomaly and got clean, computable segmented images.

5.5.4 Pre-process of Segmentated Image

At this stage, we have segmented our image and from one single answer script we get around 25-30 segmented images each containing a single digit. In our demonstration image, we got 23 segmented images. However, these images vary in dimension and are still not similar to the data type the model was trained with. Fig 5.13 is a representation of the dataset that was used to train our model. In this preprocessing step, we tried to mimic our segmented images to be like the previously mentioned dataset.

First, we converted the image to grayscale. Fig 5.14 is a representation of how our segmented image looked after resizing. As we can see there were still discrepancies between the images. We tried various image processing techniques to mimic the dataset. In fig 5.15, we showcased different outcomes for different processing

Figure 5.13: Representation of our custom data.

techniques. We used histogram equalization method for our image, but for our purpose the processed image was not suitable. Then, we decided to increase the image contrast. Initially, we set the contrast control factor to 1.5 and got a satisfactory output. However, upon increasing the control factor to 1.9 we observed the details of the digits which became more profound. Thus, we choose to set the control factor to 1.9 and increase the segmented image contrast. Furthermore, we resize the images to 28*28 and set the interpolation method to bilinear interpolation. Later we did a bitwise NOT operation for the resized image, which essentially converts light pixel to dark pixel and vice versa. Lastly, we reshaped the image, changed its data type to 'float32' and normalized the pixel values to range from 0 to 1.

Figure 5.14: Resized Segmented Image Without Pre-processing.

Figure 5.15: Result of different pre-processing.

5.5.5 Prediction

For our model interface, we designed four prediction functions. Two of these functions were for digit recognition and the other two were for alphabet recognition. For digit recognition, one of the two functions was to make a string while the other function would only recognize specific images. For example, the first function would take a relevant array of segmented images like 8 segmented images of student id. The first function would then cycle through the array of segmented images and call the second function who would recognize each individual image. The second function would initialize the model, predict the image's digit, turn the data type to string and return the result to the first function. The first function that was going through all relevant segmented images would take the output from the second function and concatenate it to a string.Ultimately, the first function would return the predicted id in such a manner. The same thing happens for the two alphabet recognition model, only the model initialized in the second function there is different.

5.5.6 Second Step Verification and Creation of Improvement Dataset

In the previous section, we talked about how due to lighting problems or overlapping documents, the fix orientation algorithm can falter. Thus, we choose to have a second step verification algorithm integrated to our model interface as well.
Here the algorithm will cross check predicted id with a user provided excel sheet data, to confirm whether or not the student exists. If the student id exists in the database, the algorithm will go on, otherwise the algorithm will do a cross check with the predicted name and id with the database to find the closest matching student. The algorithm will notify the user to recheck said closest student while also storing said extracted region data for future training purposes. By doing so, the algorithm is creating a new improvement dataset for our model. The model can learn from its mistake as a dataset is being automatically updated with contents that the existing model cannot handle.

5.5.7 Output

The ultimate goal of our model interface is to take multiple answer scripts as input and as output create csv files that will contain student id, section, question marks and total marks.
From our previous segments of our model interface, we got the predicted value of the extracted region data. We created arrays to store these individual entries and we used the data of these arrays to fill in the csv file. The csv file was then also stored in the google drive for the user to access.

Chapter 6

Result Analysis

6.1 Compare Accuracy of Chosen Models

This research aims to find the best lightweight model so that we can implement it on different platforms easily while retaining excellent accuracy. So first, let us observe individuals models precision, recall and f1-score for each digit:

Table 6.1: KNN

Digit	Precision	Recall	F1-score
0	0.8	0.92	0.86
1	0.97	0.94	0.95
2	0.82	0.86	0.84
3	0.79	0.81	0.8
4	0.9	0.8	0.84
5	0.94	0.82	0.87
6	0.85	0.89	0.87
7	0.93	0.83	0.88
8	0.77	0.86	0.81
9	0.77	0.76	0.77

From the Fig 6.1 we can observe that, KNN model performed fairly well with the pre-process that we did and maintained a precision above 90% except for digits: 3,8,9. As these digits closely resemble each other so when a person is writing them, it is harder to predict it accurately. But as we managed to fix different segmentation related errors, the result is far better than other papers that used KNN as well.

Then we have SGDClassifier, from the Fig 6.2, we can see, this model overall performed worse then KNN but the results were all uniform and there wasn't much scatter of precision. Where KNN managed to detect a few digits really well and for some the result plummeted, SGDClassifier managed to maintain a fairly closely knitted result when detecting digits from hand written numbers.

Table 6.2: SGDClassifier

Digit	Precision	Recall	F1-score
0	0.84	0.89	0.87
1	0.93	0.95	0.94
2	0.79	0.87	0.83
3	0.85	0.76	0.8
4	0.81	0.83	0.82
5	0.89	0.85	0.87
6	0.86	0.89	0.87
7	0.86	0.87	0.87
8	0.8	0.81	0.8
9	0.77	0.69	0.73

Table 6.3: SVM

Digit	Precision	Recall	F1-score
0	0.87	0.91	0.89
1	0.96	0.95	0.96
2	0.83	0.89	0.86
3	0.83	0.83	0.83
4	0.85	0.85	0.85
5	0.91	0.88	0.89
6	0.87	0.91	0.89
7	0.91	0.87	0.89
8	0.85	0.81	0.83
9	0.78	0.75	0.77

From this Fig 6.3, the SVC model performed overall better than the previous two models. We can see that the precision scatter is similar to that of KNN. For a few digits, the model is performing extremely well but for some digits, the precision hits rock bottom. However, the precision has increased compared to KNN. Similarly, SGDClassifier may have better uniform result but the overall precision is higher in SVC.

Table 6.4: YOLOv8

Digit	Precision	Recall	F1-score
0	.99	1	1
1	.99	1	1
2	.99	.97	.98
3	.99	.99	.99
4	.99	1	.99
5	.97	.98	.98
6	.99	.99	.99
7	.99	.99	.99
8	.99	.99	.99
9	1	.98	.99

YOLOv8 is a really advanced model and can apply past learnings in the new dataset or model. So, it is no surprise that YOLOv8 performed really well in this case. From the Fig 6.4 we can see that the lowest precision value is 98% which is more than the highest value of the last three models. And in one case we are getting 100% precision of the digits. So, this is a really optimized and high performing model that we can consider to apply for the final product.

Table 6.5: CNN

Digit	Precision	Recall	F1-score
0	0.98	0.99	0.99
1	0.99	1	0.99
2	0.98	0.98	0.98
3	0.99	0.99	0.99
4	0.99	0.98	0.99
5	0.99	0.98	0.98
6	0.99	0.98	0.98
7	0.98	0.99	0.98
8	0.98	0.99	0.98
9	0.98	0.98	0.98

Finally we have CNN. From the Fig 6.5, we can observe that the model has similar performance to the YOLOv8 model. The precision is really high and the lowest value is similar to that of YOLOv8 : 98%. Although it did not reach any perfect precision anywhere but nevertheless, it retained a really high value in all the digits. This is a neural network model, so due to the convolution and pooling layers with

the fine tuned image pre-processing, it is performing exceptionally well in our custom dataset.

Now, let us observe the confusion matrix of these models:

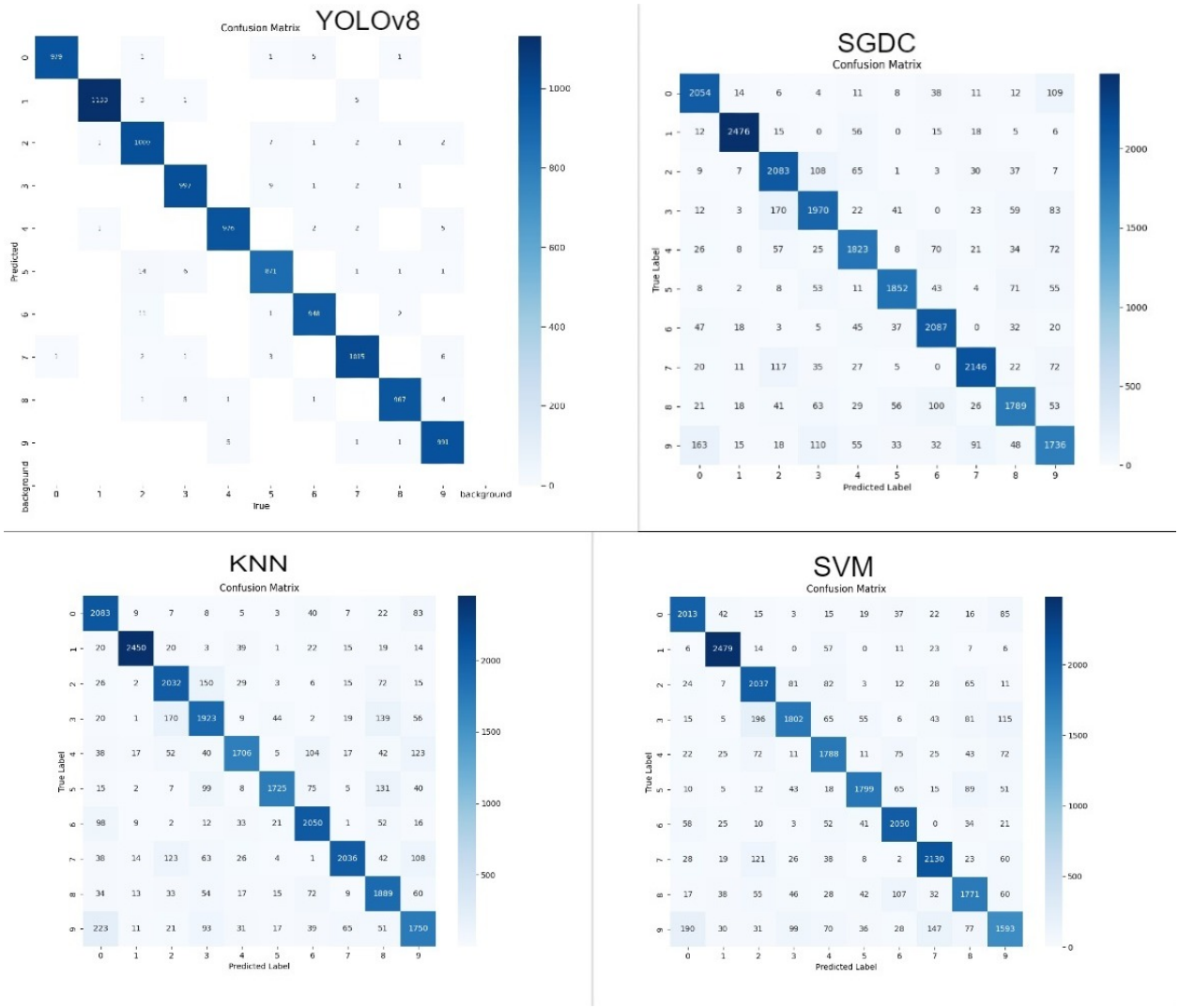


Figure 6.1: Confusion Matrix of Different Models.

From the Fig 6.6 , we can see that CNN and YOLOv8 have the best prediction from the rest and they have some no false predictions in some cases. Although they were excellent models to begin with, in addition to the pre-processing that we did and removed many errors that we would have depended on the model to perform, the selected models demonstrated really high accuracy in our custom dataset containing handwritten dataset.

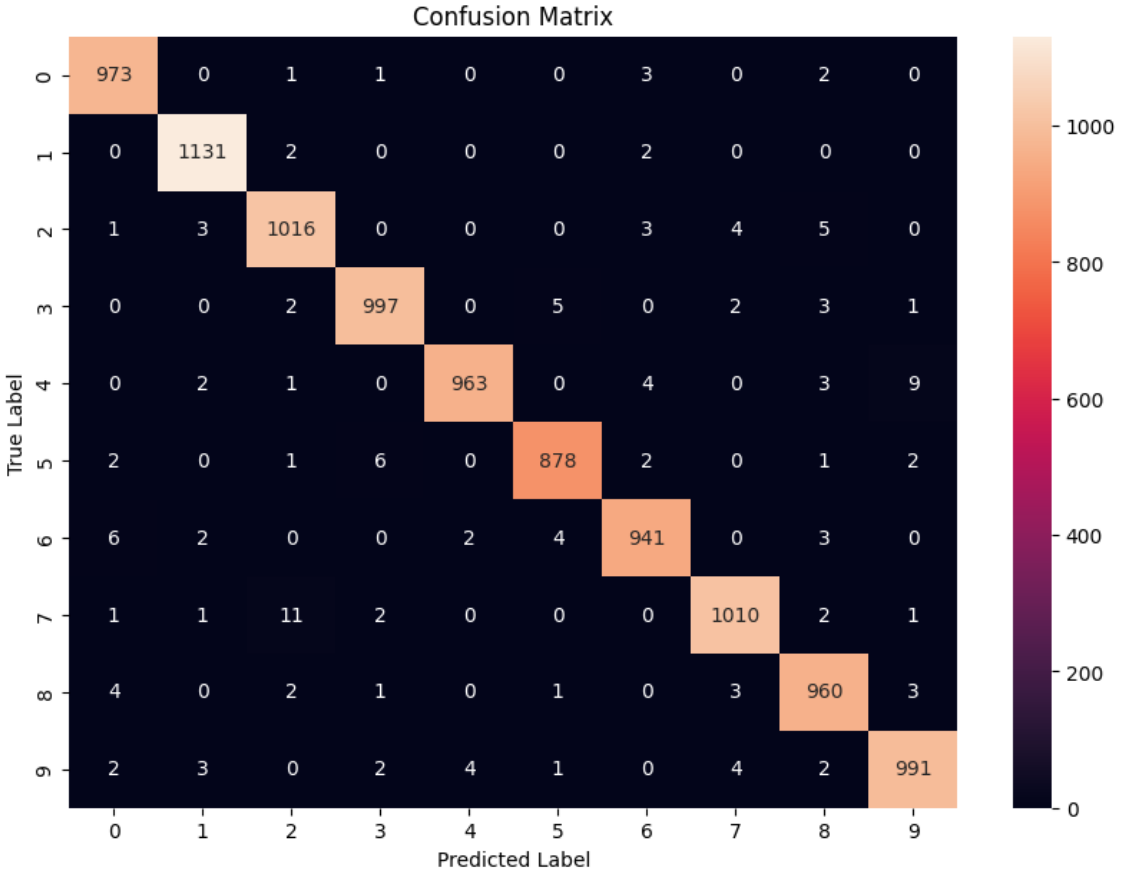


Figure 6.2: Confusion Matrix of CNN.

For further clarification and to choose a better model out of these, lets observe their accuracy:

Table 6.6: Accuracy Comparison

Model	Accuracy
KNN	85%
SGDClassifier	84%
SVC	87%
CNN	98.6%
YOLOv8	98.8%

From the Fig 6.7 , it becomes evident that CNN and YOLOv8 outperforms all other models in terms of accuracy, precision, recall and f1-score. Also, the gap between these two and the other models are really high. Here, CNN is a neural network model whereas YOLOv8 is a transfer learning based model. As our objective is to apply the best model in a real life scenario where we will encounter complex connected strings. So, having a pre-trained model which can apply that knowledge while training in our current dataset, it will be better in the long term. The reason is because we want to add the wrong predictions back to our dataset when we are using the model on real world data. So, if the model can retain its previous training and continuously update itself from the new data, it will be the perfect model in our case. So, we aim to apply this model in a real world scenario and deploy a medium which will be incorporated on this idea.

Chapter 7

Conclusion and Future work

7.1 Conclusion

To conclude, integration of ML and OCR in an android environment has great potential as it directly enhances user experience by removing the human labor and makes it more time efficient. In addition, the cross checking feature will rapidly ensure data validity and speed up the process of error handling even more. Then by enabling the processing to occur in the local environment, people can now store data with ease when visiting rural areas with no reception. Furthermore, the model uses user reference to calibrate itself for better results. So, the model can ensure each user gets their own preference in extracting data. And by the fact that everyone in the modern world currently has access to a smartphone, we can obsolete the idea of a different device needed for this type of work in terms of scanning and extracting. So, this research will directly impact the common people in empowering them and help them integrate with the processes of the modern world.

7.2 Limitation

In our research, we have managed to take an image data as input and extract it with different pre-process and put it in a computer editable format. Although, we managed to solve and improve many problems like cropping the image correctly, fixing different segmentation related errors. But, there were some problems along the way: when the background of the intended document was in a brighter contrast than expected, the cropped image contained that background. Also, images from different types of camera have different properties, we did not incorporate that when doing our case and instead went for a general format which the majority of the people had.

7.3 Future Work

While collecting our dataset, we only collected those of the students of our institution. So there is a bias of both age group and education level in the dataset. So in future, we aim to further extend our research, starting with student handwritten data from different age groups. This will allow us access to data of the effect of age and level of education in handwriting. In addition to that, we want to study the pattern of different formats simultaneously and find a connection so that we can predict the occurrence of a specific type of data in a place and its reason.

Bibliography

[1] M. A. Chandra and S. S. Bedi, "Survey on svm and their application in image classification," *International Journal of Information Technology*, Jan. 2018. DOI: https://doi.org/10.1007/s41870-017-0080-1.

[2] K. Kowsari, M. Heidarysafa, D. E. Brown, K. J. Meimandi, and L. E. Barnes, "Rmdl: Random multimodel deep learning for classification," *Proceedings of the 2nd International Conference on Information System and Data Mining - ICISDM '18*, pp. 19–28, 2018. DOI: https://doi.org/10.1145/3206098.3206111. [Online]. Available: https://arxiv.org/abs/1805.01890.

[3] Z. Lei, S. Zhao, H. Song, and J. Shen, "Scene text recognition using residual convolutional recurrent neural network," *Machine Vision and Applications*, vol. 29, no. 5, pp. 861–871, 2018. DOI: 10.1007/s00138-018-0942-y.

[4] A. Sere, F. T. Ouedraogo, and B. Zerbo, "An improvement of the standard hough transform method based on geometric shapes," *Advances in Intelligent Systems and Computing*, pp. 369–384, 2018. DOI: 10.1007/978-3-030-03405-4_25.

[5] S. S. Mor, S. Solanki, S. Gupta, S. Dhingra, M. Jain, and R. Saxena, "Handwritten text recognition: With deep learning and android," *International Journal of Engineering and Advanced Technology*, vol. 8, no. 3S, pp. 819–825, Feb. 2019, ISSN: 2249-8958. [Online]. Available: https://www.ijeat.org/wp-content/uploads/papers/v8i3S/C11730283S19.pdf.

[6] S. Y. Chaganti, I. Nanda, K. R. Pandi, T. G. Prudhvith, and N. Kumar, "Image classification using svm and cnn," in *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*, 2020, pp. 1–5. DOI: 10.1109/ICCSEA49143.2020.9132851.

[7] H. Dave, "Ocr text detector and audio convertor," *International Journal for Research in Applied Science and Engineering Technology*, vol. 8, no. 5, pp. 991–999, 2020. DOI: 10.22214/ijraset.2020.5157.

[8] J. Qin, W. Pan, X. Xiang, Y. Tan, and G. Hou, "A biological image classification method based on improved cnn," *Ecological Informatics*, vol. 58, no. 58, p. 101 093, Jul. 2020. DOI: https://doi.org/10.1016/j.ecoinf.2020.101093. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1574954120300431.

[9] J. E. Tejaswini, "Text extraction from images using ocr," *International Journal for Research in Applied Science and Engineering Technology*, vol. 8, no. 5, pp. 1805–1810, 2020. DOI: 10.22214/ijraset.2020.5291.

[10] A. U, "Text localization and recognition," *International Journal for Research in Applied Science and Engineering Technology*, vol. 8, no. 5, pp. 981–985, 2020. DOI: 10.22214/ijraset.2020.5155.

[11] Z. Fan, J.-k. Xie, Z.-y. Wang, P.-C. Liu, S.-j. Qu, and L. Huo, "Image classification method based on improved knn algorithm," *Journal of Physics: Conference Series*, vol. 1930, p. 012 009, May 2021. DOI: 10.1088/1742-6596/1930/1/012009.

[12] L. Gannetion, K. Y. Wong, P. Y. Lim, K. H. Chang, and A. F. L. Abdullah, "An exploratory study on the handwritten allographic features of multi-ethnic population with different educational backgrounds," *PLoS ONE*, vol. 17, no. 10, e0268756, Oct. 2022. DOI: https://doi.org/10.1371/journal.pone.0268756%7D. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9544031/%7D.

[13] Y. Jiang, Z. Jiang, L. He, and S. Chen, "Text recognition in natural scenes based on deep learning," *Multimedia Tools and Applications*, vol. 81, no. 8, pp. 10 545–10 559, 2022. DOI: 10.1007/s11042-022-12024-w.

[14] R. Li and S. Li, "Multimedia image data analysis based on knn algorithm," *Computational Intelligence and Neuroscience*, vol. 2022, Q. Li, Ed., pp. 1–8, Apr. 2022. DOI: https://doi.org/10.1155/2022/7963603.

[15] S. Patil, V. Varadarajan, S. Mahadevkar, R. Athawade, L. Maheshwari, S. Kumbhare, Y. Garg, D. Dharrao, P. Kamat, and K. Kotecha, "Enhancing optical character recognition on images with mixed text using semantic segmentation," *Journal of Sensor and Actuator Networks*, vol. 11, no. 4, p. 63, 2022. DOI: 10.3390/jsan11040063.

[16] S. Xia, J. Kou, N. Liu, and T. Yin, "Scene text recognition based on two-stage attention and multi-branch feature fusion module," *Applied Intelligence*, vol. 53, no. 11, pp. 14 219–14 232, 2022. DOI: 10.1007/s10489-022-04241-5.

[17] B. Xiao, M. Nguyen, and W. Q. Yan, "Fruit ripeness identification using yolov8 model," *Multimedia Tools and Applications*, Aug. 2023. DOI: https://doi.org/10.1007/s11042-023-16570-9.

[18] S. Chen, Y. Li, Y. Zhang, Y. Yang, and X. Zhang, "Soft x-ray image recognition and classification of maize seed cracks based on image enhancement and optimized yolov8 model," *Computers and Electronics in Agriculture*, vol. 216, pp. 108 475–108 475, Jan. 2024. DOI: https://doi.org/10.1016/j.compag.2023.108475.

[19] Ultralytics, *Brief summary of yolov8 model structure*. [Online]. Available: https://github.com/ultralytics/ultralytics/issues/189.