**Automatic Recognition of Handwritten Grades on**

**Exam Cover Sheets**

Contents

[**Automatic Recognition of Handwritten Grades on** 1](#_Toc116580106)

[**Exam Cover Sheets** 1](#_Toc116580107)

[Abstract 1](#_Toc116580108)

[Introduction 1](#_Toc116580109)

[Literature Survey 2](#_Toc116580110)

[Related Knowledge 5](#_Toc116580111)

[RCNN 6](#_Toc116580112)

[Fast RCNN 7](#_Toc116580113)

[Faster RCNN 8](#_Toc116580114)

[Region Proposal Network 9](#_Toc116580115)

[Anchor Box 10](#_Toc116580116)

[Objectness Score 10](#_Toc116580117)

[Feature Sharing of RPN and Faster RCNN 11](#_Toc116580118)

[Region of interest (ROI) 11](#_Toc116580119)

[Training of faster RCNN 11](#_Toc116580120)

[ResNET\_50 12](#_Toc116580121)

[ResNet -34 architecture 13](#_Toc116580122)

[Structure of the ResNet-50 13](#_Toc116580123)

[Problem Statement 13](#_Toc116580124)

[Methodology 13](#_Toc116580125)

[Dataset 13](#_Toc116580126)

[Data Preparing and Preprocessing 14](#_Toc116580127)

[Train Test Split 16](#_Toc116580128)

[Model Development and Training 16](#_Toc116580129)

[Evaluation Measures 17](#_Toc116580130)

[Results 17](#_Toc116580131)

[Faster RCNN Results 18](#_Toc116580132)

[Yolo V5 Results 19](#_Toc116580133)

[Comparative Analysis of Both Models 22](#_Toc116580134)

[Conclusion 23](#_Toc116580135)

[References 23](#_Toc116580136)

[Appendix 27](#_Toc116580137)

[Yolo V5 Training Code 27](#_Toc116580138)

[Code for Faster RCNN 30](#_Toc116580139)

Table of Figures

[Figure 1: Style of writing digits in different languages. 5](#_Toc116580140)

[Figure 2: Steps for object detection by Faster RCNN 8](#_Toc116580141)

[Figure 3: RCNN operations for object Detection. 8](#_Toc116580142)

[Figure 4: Fast RCNN architecture Design. 9](#_Toc116580143)

[Figure 5: Faster RCNN architecture for object Detection. 11](#_Toc116580144)

[Figure 6: Extraction of Regions with variation of anchors. 12](#_Toc116580145)

[Figure 7: Example of ResNet 50 working 14](file:///E:\ICRL\41%20Handwritten%20Marks%20Recognition\Introduction%20(AutoRecovered).docx#_Toc116580146)

[Figure 8: Overview of Dataset Samples. 16](#_Toc116580147)

[Figure 9: Overview of Dataset Annotated Samples. 17](#_Toc116580148)

[Figure 10: Class Distribution in Dataset. 18](#_Toc116580149)

[Figure 11: Architecture of Yolo v5 Model. 19](#_Toc116580150)

[Figure 12: Bar Chat of learning rate with the training of model. 20](#_Toc116580151)

[Figure 13: Accuracy and loss plots of the Faster RCNN model. 21](#_Toc116580152)

[Figure 14: Confusion Matrix of Yolo V5 Model. 21](#_Toc116580153)

[Figure 15: Accuracy and Loss Plots of the Yolo v5 Model. 22](#_Toc116580154)

[Figure 16: Precision, Recall and F1 Curve During Training. 23](#_Toc116580155)

[Figure 17: Overview of Yolo v5 Predicted samples. 24](#_Toc116580156)

Table of Tables

[Table 1: Selected Evaluation measures of both Models. 24](#_Toc116580157)

Abstract

The calculating the sum of the student grades manually or entered the student grades into the portal is very time-consuming task with high risk of human error. For instance, if the class have the more than 200 students in different sections and each student appear two times in the exam (mid-term and final term) then the 400-exam sheet need to be calculated the sum of students grade. Moreover, every exam normally based on 5 to 7 questions, means the administration department need to 400\*7 entries in the storage device. Collectively the whole procedure is very time consuming, risk of errors and required the full attention. To rescue these challenges there is need to automate the process of entering the students grade from cover sheet into database and calculate the final grade of the student automatically. Here we proposed the offline digit recognition approach that automatically recognize the handwritten digit from the captured image of cover sheet. In this regard, Faster RCNN with ResNet50 model and yolo v5 object detection model was used for the recognition of digits. The Faster RCNN model showed the # accuracy score for our customized dataset.

Introduction  
The automatic recognition of numerical numbers and characters have the numerous applications including the verification of signatures in banking [1], name and address recognition from user submitted form [2] and recognition of clinical data [3]. The main agenda of these application is automatically recognizing the digits and characters from text document. To achieve this challenge, there is need to convert the text document in the scanned document or in the format that computer can understand. The text of the document may be a printed text or handwritten text. Hence, the information of the document can be categorized into two different categories i.e., handwritten or printed. In the schools, colleges and universities, the number sections for compulsory subjects are around 25-30 and each section normally have the 40 students. As the current educational system is based on mid term and final examination and each exam mostly based on 5-9 questions, the related supervisor or lecturers need to fill the marks sheet or enter in the portal more than 60000 times. In addition, the relative teacher also needs to calculate the total score of the student more than 60000 times. To finding the student names on the portal or marks sheet and fill the marks accordingly is also a very time-consuming task. The calculation of total scores more than 60000 times by the human increases the risk of false calculation or error. By viewing all of these hurdles and consumption of time and recourses, there is need of a mechanism that reduce the time and increase the efficiency of calculating the final score.

The alternative way of recognizing the marks on cover page or mark sheet is the optical mark recognition (OMR). The main objective of this technique is to extract the information from the specific location in the document [4]–[6]. The traditional OMR scheme basically worked with the light beam of scanner and the reflectivity of the light at the specific positions in the document are used to recognized the marks. The most popular and widely used application of OMR is the marking of students exams based on multiple choice questions. In this application, the HB or 2B pencil marks are used to fill the circle shaped bubbles on the exam sheet. The OMR application pass the scanner light and recognize the reflectivity from dark area. According to the reflectivity, it recognizes the selected option of multiple-choice question and also mark it accordingly. Majority of the OMR applications recognize the student-id or registration-id, student total score for multiple choice questions [7], [8] and automatically export the total score in the file with student registration id. OMR applications are very useful in educational premises and give the significant accuracy and efficiency in recognizing the marks. But the main disadvantage of the OMR application is that the OMR is just limited to the marking of multiple-choice questions. Alternatively, OMR can’t help to our all problems because most of the subjects examination could not be fulfill by the multiple-choice only. Normally, the exam paper is based on the multiple-choice, short questions and long questions and the student write the answer in the descriptive form rather than the filling of bubbles. Resultantly, the descriptive question needs to be understood by the teachers and marked them manually on the exam sheet. The manually marking of student exam paper and calculating the final score is not a big problem if the amount of student is not much high. But in real environment, the situation is totally changed. The total marking of exam and finding out the student ids in a big list will raise many problems.

To rescue this big problem, here is another technique refer as optical character recognition (OCR). OCR has the ability to recognize the typed text and handwritten text by converting the text document in the text image with the help of scanner. OCR systems are divided into two different categories labeled as offline recognition and online recognition. In the offline text recognition technique, the handwritten text document is converted into the machine-based image through scanning and then all the text is extracted from the image. While in the online recognition techniques, text information and numerical digits are typed from any typing device and then the recognition is made from the types document. The offline character recognition systems are more complex compare to the online recognition systems due to the different writing styles of users and noise during the writing of the text and scanning of the text document. Moreover, the text recognition systems are mainly worked on two different strategies labeled as segmentation then recognition [9] and segmentation-based recognition [10]. In the primary technique, the segmentation module returns the single character for recognition and the recognition module then recognize the single character. The major limitation of this technique is the false segmentation when the segmentation module does not extract the single characters accurately according to the defined rules. To increase the system's resilience, contextual data is frequently incorporated throughout the segmentation process. The best segmentation-recognition grade of the input picture must be expressed in the final choice under the second technique, which is based on a probabilistic premise. Typically, the segmentation module of the system generates a list of possibilities, and the recognition module subsequently assesses each assumption. The list is then postprocessed while taking the context into account. Although this method provides greater dependability than the previous one, its primary disadvantage is the computing work required to compare all the created possibilities. Additionally, different schemes including fragments, isolated characters, and linked characters must be distinguished by the recognition module. This technique allows either explicit segmentation based on cutoff algorithm [11], or implicit segmentation where each pixel column is a candidate place for a cut [12] [13].

In this proposed work, we proposed the recognition module for the automatic recognition of handwritten grades on exam cover sheets. The recognition module will be based on the segmentation-based recognition technique. The recognition module will segment the marks on the exam sheet and then recognize the segmented area. For the recognition of text on the handwritten exam sheet, we proposed the Faster RCNN and Yolo V5 deep learning based models.

# Literature Survey

In recent years, the field of pattern recognition and image processing has focused on the idea of handwritten character recognition, and optical character recognition on handwritten scripts has gained widespread acceptance. This section presents a thorough study of the existing approaches in handwriting recognition systems that rely on different machine learning algorithms. Even though the recognition of printed material is now regarded as a solved problem, the identification of handwritten content is still a difficult task due to the high variance of different hand writing styles, font styles, font sizes, boldness of text and orientation of alphabets and digits among the different peoples. The interpretation of handwritten documents has been proposed using a variety of machine learning techniques. The automatic grading of handwritten responses and the identification of handwritten letters and numbers in multiple languages are only a few examples of the several handwriting recognition systems employing machine learning classifiers that are described in this section. The format of writing digits in different languages is presented in below Figure 1.

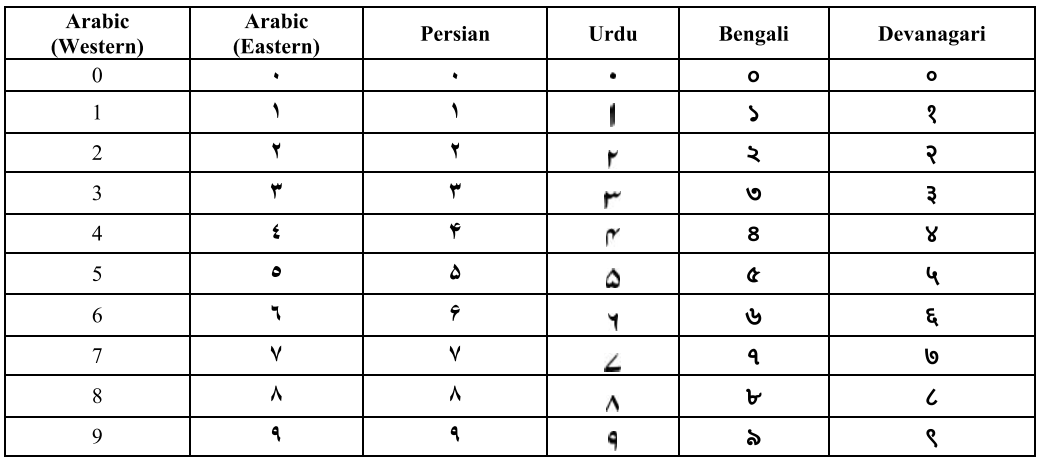


Figure : Style of writing digits in different languages.

With the help of CNN, Brown [14] presented an automated method to score handwritten number responses on scanned answer sheets using the MNIST offline handwritten dataset. The accuracy of the CNN approximation of the student responses was 95.6%. In a study proposed by the author [15], a linear regression model was used for the for assessment of SAT test. The proposed study used the Kaggle dataset and converting the characters and digits into the features. The linear regression model showed the approximately 87% accuracy score. Multiple - choice questions answers were scored automatically in [16], allowing individuals to print and scan all of the answer sheets. Each answer sheet required 35 seconds of training time. The study [17] also proposed the random forest model for the recognition of handwritten student marks from the cover sheet. The proposed random forest model used the 3960 scanned documents for training and testing procedure and also produce the 89% accuracy. Srihari et al. [18] also proposed the marking scheme for the comprehensive types exams. They proposed the study used the handwritten 300 documents based on the essays and split the data with 50% ratio for training and testing data. For marking the handwritten essay, ANN classifier was used that present the approximately 87% accuracy score. Using the 13000 document Kaggle dataset as their starting point, Mahana et al. [19] created an automated method for scoring essays. Linear regression model was also used for extracting several features from the document of training set and got the 91.85% accuracy score after training.

An OMR system was launched by Saengtongsrikamon et al. [20] and implemented as an OMR machine using neural networks before being integrated into a scanner. With a 95.24% accuracy rate, this application recorded and graded the responses to multiple-choice questions. The OMR machine has the capability to scanned the 1000 documents of different resolutions. Using a unique local feature extraction technique, the authors of study [21] designed a multilingual handwritten number recognition system. For the multi-lingual recognition system, MADBase, MNIST, MPU-UD, ICDAR, HODA and DHCD was used based on the Arabic, English, Urdu, Bengali, Persian and Devanagari language respectively. The same author also proposed another study [22] with enhanced features set that was classified by the various machine learning models. According to the study, random forest model performed well for the recognition of handwritten numbers in different languages with the accuracy rate of 96.73%.

In the study [23] the author suggested an image processing technique for autonomously grading answer sheets with multiple choice questions. This method allowed each user to print whichever response sheet they wanted, and after printing, they used a standard scanning and computing device to evaluate the mark sheet. Out of the complete 1000 answer sheets, the suggested approach required 1.4 seconds of training time per sheet and identified 100% accuracy. Muangprathub et al. [24] also proposed a KNN based technique for the automatically grading the scanned MSQs sheet. In all, 560 MSQs sheets were assessed with the trained model. The performance of the trained model was three times faster than the manual marking with approximately 100% accuracy on completely filled bubble keys whereas the accuracy for incomplete, overflow and deleted bubbles sheet, the accuracy was 99%, 62%, and 93% respectively. Using a scanning device that could assess a multiple-choice exam, Patole et al. introduced a novel method for scoring multiple-choice exams in [25]. This project used the processing capabilities of C++ with the C# language to assess each student's academic achievement and solicit comments from the students regarding the staff. Finally, compared to typical evaluation systems, the software was able to offer advantages such as improved scalability and adaptability for asynchronous mode of assessment. A technique for rectifying multiple-choice answer sheets using mathematical formalism and k-nearest neighbors was proposed by Tayana et al. [26]. (k-NN). The database utilized to manipulate the photos includes 10 basic image files with 26 questions and four possible answers for each, as well as 680 certified answer sheets. A 99.85% total accuracy rate was attained. The researcher of [27] suggested a method for grading a specially created multiple-choice question paper that included 10 questions and five possible answers. The proposed system presented 82% accuracy score.

An author of study [28] also presented the handwritten character recognition framework based on the CNN. In order to recognize multilingual handwritten numbers, Latif et al. [29] suggested a deep learning framework employing Deep Convolutional Neural Networks. The Arabic, English, Persian and Urdu language databases labels as MADBase, MNIST, HODA and PMU-DB was constructed due to the lack of pre-existing dataset accessible for this languages), and DHCD were the databases utilized to assess the correctness of the proposed technique (Devanagari). And the aggregate accuracy that was as a consequence achieved for every language was 99.322%. A study proposed by the Singh. N. [30] also suggested the ANN based model for the recognition of handwritten characters of Devanagari with the accuracy score of 98% approximately. The proposed ANN model consumed the 2.33 seconds for the Devanagari characters identification of 400 documents. Kumar et al. [31] also proposed a SVM with kernel tuning and MLP model for the recognition of English characters. collection of distinct handwritten characters that were written by various authors. Each character contributed 27 characteristics during training, which were used for the training of the SVM with an accuracy of 80.96%. An examination of the English handwriting recognition method based on CNN was proposed by Rao et al. in [32]. The datasets utilized for this were the SVHN and MNIST databases. The handwritten characters recognition accuracy results from MNIST and SVHN Databases were 94.65% and 95.1%, respectively. Jong et al. [33] research proposal on ensemble learning machine-based handwritten English letter recognition systems.

Although Montazer et al. [34] suggested an extensive technique by utilizing neuro-fuzzy inference engine to recognize the Arabic and Persian decimal numbers, a probabilistic neural network (PNN) strategy for the recognition of the handwritten Indian numerals has been presented by the study [35]. The proposed technique used the center of gravity and set of vectors from the bounding box to the digit. Finally, Impedovo et al. developed an adaptive zoning strategy for handwritten digit recognition [36], [37] where the features are retrieved in accordance with an ideal zoning distribution. This method utilizes zoning features and is based on an evolutionary algorithm. The experimental tests demonstrate the latter's efficacy in comparison to conventional zoning planning methods.

An author of the study [38] proposed a Bayesian network-based framework for recognizing the handwritten characters of Arabic language by implementing the Discrete Cosine Transformation function. The proposed framework was evaluated using the seventy thousand Arabic characters written by the distinct 700 writers in the Arabic language dataset (ADBase). The proposed Bayesian network-based framework presented the 85.26% accuracy score. For the purpose of recognizing Farsi/Arabic handwritten digits, an author of study [39] suggested an approach based on a collection of Singular Value Decomposition (SVD) models and multiphase Particle Swarm Optimization (PSO). The authors of the study used the entries of HODA database for the evaluation of proposed models and got the 97.02% accuracy. Musleh, Halawani and Mahmoud [40] also proposed the study for the recognition of handwritten digit of Arabic language. The proposed study used to two classifications algorithm based on the SVM classifier and syntactic fuzzy classifier. Firstly, the SVM model was used for the classification of zero and non-zero digits of Arabic language. Secondly after the recognition of non-zero digits, the 1-9 Arabic digit were classified by the syntactic fuzzy algorithm. The 32695 sample digits of Arabic database were used for the evaluation of both models. The SVM and fuzzy logic achieved the 99.55% and 98.01% accuracy score for zero/non-zero classification and 1-9 digits classification respectively. It needs to be noticed that the accuracy automatically improves by categorizing the numbers into zero and nonzero due to the misunderstanding that exists with 0 and other numbers, like as 5 Arabic writing. For the identification of single handwritten digits, [41] suggested a brand-new, large-margin domain adaption approach. Additionally, they created a framework for ensemble projection feature training to take use of the unlabeled data that accessible in the particular domain. Testing of this strategy on three common datasets—MNIST (Western Arabic), USPS, and ICDAR—showed that it outperforms other well-known domain adaptation techniques with a accuracy score of 89.86% in both domain adaptation situations.

In the research proposed by the author of [42], developed a comprehensive handwritten dataset of Persian language. The developed dataset was based on the Persian characters including the writers name, dates in word and numerical form, writer age, and different string. The dataset samples were based on numerical string collected from 500 writers. In the study proposed by the Boukharouba and Bennia [43], used a feature extraction technique for the identification of offline Persian language digits. The researchers of the proposed study also used the SVM classifier with the extracted features for Persian digit recognition. The samples of HUDA database were used to evaluate the complete framework and achieved the 98.55% accuracy score. Karimi et al. [44] also proposed a framework for the identification of offline Persian language character recognition. The proposed framework based on the feature extraction technique and machine learning classifier. The authors extracted the 115 features and classify them with ensemble learning algorithm. The proposed framework was evaluated using the test samples of TMU database and attained the 92.8% accuracy score.

A zone sampling method utilizing multi-objective evolutionary algorithms was suggested by Sarkhel et al. [45]. For distinct area sampling, they employed the two different algorithms labeled as Non-dominated Sorting Harmony-Search Algorithm (NSHA) and the Non-dominated Sorting Genetic Algorithm-II (NSGA-II). The proposed technique used the support vector machine classifier and used the two distinct Bengali databases based on handwritten characters for the evaluation of the model. The trained SVM classifier got the 98% accuracy score. In another study also proposed by the Basu et al. [46] used the two distinct features set for the recognition of handwritten Bengali characters. The extracted features set were combined with the MLP classier for recognition. The proposed model was evaluated using the 6000 handwritten samples and got the 95.1% accuracy score.

Resistance Random Access Memory (RRAM) was suggested by Wang et al. [47] as the embedded system of a neural network for the detection of offline digits. The proposed algorithm was tested with the samples of MNIST (Arabic) and achieved the 81% accuracy. A study suggested by the Ali and Ghani [48] extracted the features based on transformation and Discrete Cosine transformation-based features. They also combine the both features set and implemented the Hidden Markov model for Arabic digit recognition. The proposed HMM model was evaluated using the MNIST (Arabic) handwritten samples and achieved the 97.2% accuracy score. A number identification approach based on the Extended Label Propagation method and combined with entropy-based features to account for each digit confidence coefficient was proposed by Jie et al. in [49]. The label propagation method receives the updated confidence coefficients and is then retrained using the updated coefficients. The proposed method was also evaluated with the numbers of MNIST (Arabic) database and attained the 98% accuracy. In study [50], three types of features were used for the recognition of Devanagari characters. The features were based on Density, moment features including top, bottom, right, left, and features of descriptive moment. The combination of extracted features was used with the different classifiers of machine learning. Lastly, the meta-pi network was used to gain the best recognition result from the trained classifiers.

# Related Knowledge

For the recognition of handwritten digits on the exam sheet, the proposed study used the Faster RCNN and yolo V5 model. Both models will detect the area on exam sheet and recognize the handwritten character in the detected area. In other words, the proposed study used the segmentation-based recognition for handwritten marks on cover sheet. Usually, the object detection algorithms are consisted of three major steps as shown in below Figure 2 (extracted from [[51]]). The first step is the generation of bounding boxes or region boxes. The generated region boxes are the candidates that may have the object in it. The most commonly used region boxes generation algorithms are selective search and Edge Boxes. Secondly, each region box is converted into the feature vector of fix length that play a critical role in the detection of object by classification algorithm. The vector should truly explain the proposed region in case of any applied transformation like scaling and shearing. Lastly the converted vector used by the classification algorithm usually SVM for the classification of object in proposed region. The converted feature vector assigned by the background class label or relative object label by the classification algorithm. The complexity of the classification algorithm increases with the increasing of object types increased.

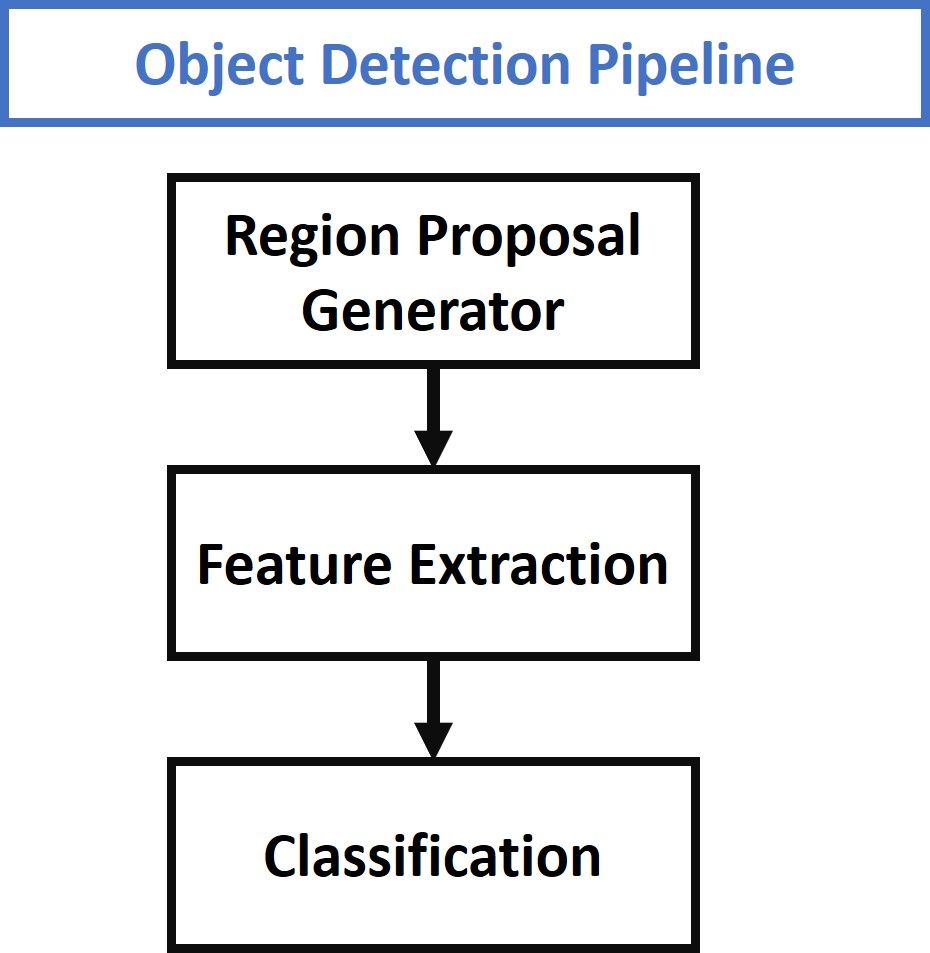


Figure : Steps for object detection by Faster RCNN

## RCNN

RCNN is pretrained region based convolutional neural network model that can classify the 80 different types of objects in the image. The main contribution of RCNN in the object detection pipeline is the automatic extraction of feature vector using the CNN. The rest of the step for object detection in RCNN are similar to the pipeline. The operations of the RCNN for object detection in an image is presented by the Figure 3 (extracted image from [51]).

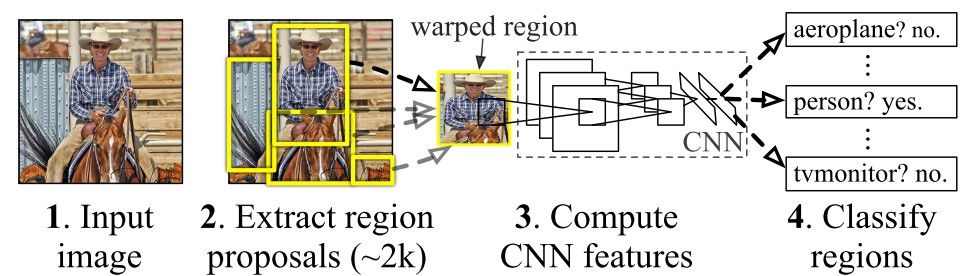


Figure : RCNN operations for object Detection.

The operation of RCNN for the object detection is based on three different steps that are following

1. In the first step, RCNN model proposed the 2000 region using the well-known selective region algorithm.
2. In the second phase, all the proposed region by selective region were converted into the fixed size region and then converted into the 4096-length feature vector.
3. Lastly the fix length feature vectors are classified by the pretrained classification algorithm (SVM).

But the RCNN model have the several drawbacks that badly impact on the performance of the RCNN. The list of RCNN drawbacks is following.

1. Firstly, the RCNN module are working independently that label it as multi step model.
2. It stored the proposed region and features vectors extracted by the CNN that require the very large storage capacity.
3. The regions are extracted by the selective search algorithm in RCNN that consume too much time.
4. Each proposed region is individually passed to the CNN for feature vector that turn the RCNN into impossible algorithm for real time object detection.

## Fast RCNN

Fast RCNN is the upgraded form of RCNN as the name exhibits. The Fast RCNN rescue the different problems faced in RCNN but the major problem overcome by the Fast RCNN was speed. Following is a diagram (Figure 4) of the Fast R- CNN design. Unlike R-CNN, which has three stages, the model only has one. It only takes an image as input and outputs the probability for each class and bounding boxes of the items that were detected. An ROI Pooling layer receives the extracted features from the previous convolutional layer as input. To retrieve a fixed-length feature representation from each region proposal, that is the rationale. The extracted fixed size feature vectors are passed to the FC layer then the output of this layer further shared with two layers label as Softmax layer for the assessment of target class score and FC layer for the prediction of bounding boxes around the detected object.

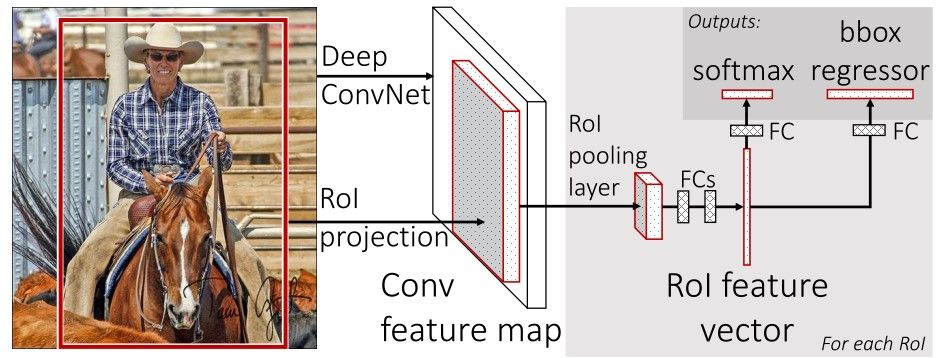


Figure : Fast RCNN architecture Design.

The contributions of Fast RCNN are following

1. Suggest the new layer label as ROI pulling layer that extract the same length features vector from each proposed region.
2. As described above, the RCNN is a multi-stage detection model, Fast RCNN is based on the single network as show in Figure 4 (extracted image from [51]).
3. Faster RCNN increased the speed of detection by sharing the calculation among all proposed region while the RCNN doing the calculation for each region independently.
4. It did not store the extracted features that make it possible to train with less storage capacity.

Although the Fast RCNN overcome the many challenges faced by the RCNN model, but the detection process takes too long time. It is due to the region proposal algorithm (Selective search) that cannot be restrict to detect some specific objects.

## Faster RCNN

Faster RCNN is most upgraded version of RCNN model. Its main contribution is the development of its own region proposal network. All the contributions of the Faster RCNN are following:

1. Faster RCNN proposed its own region proposal algorithm label as RPN (region proposal network). RPN is a convolutional neural network-based model that main function to generate the region proposal of different ratios and different scales.
2. Moreover, the RPN is also based on the attention mechanism that tell the detection algorithm where to focus.
3. This study presented the idea of anchor boxes as an alternative to pyramids of pictures (i.e., numerous occurrences of the image of different scales) or pyramids of filters (i.e., multiple filters with varied sizes). A reference box with a certain scale and aspect ratio is called an anchor box. If there are numerous reference anchor boxes, then the same region can have different sizes and aspect ratios. You can think of it as a pyramid made out of reference anchor boxes. Following that, each region is mapped to a separate reference anchor box, allowing for the detection of objects with various scales and aspect ratios.

The architecture of the Faster RCNN is mainly consist of 2 components that are described below in presented in Figure 5 (extracted from the [51])

* RPN Module (Use for the extraction of region proposals)
* Faster R-CNN (Used for the detection of object in proposed region)

Producing region suggestions is the responsibility of the RPN component. It uses neural networks to implement the idea of attention, which directs the Fast R-CNN component module to seek for object in the frame. The procedure of Faster R-CNN for detecting the object is following:

1. RPN module generate the proposed region in the input image.
2. All the proposed regions are converted into the fix length feature vector using the ROI pooling layer of CNN.
3. The converted fix length feature vectors are classified by the classification algorithm (Faster R-CNN).
4. The classification algorithm returns the score of the predicted class and the boundary of the detected object.

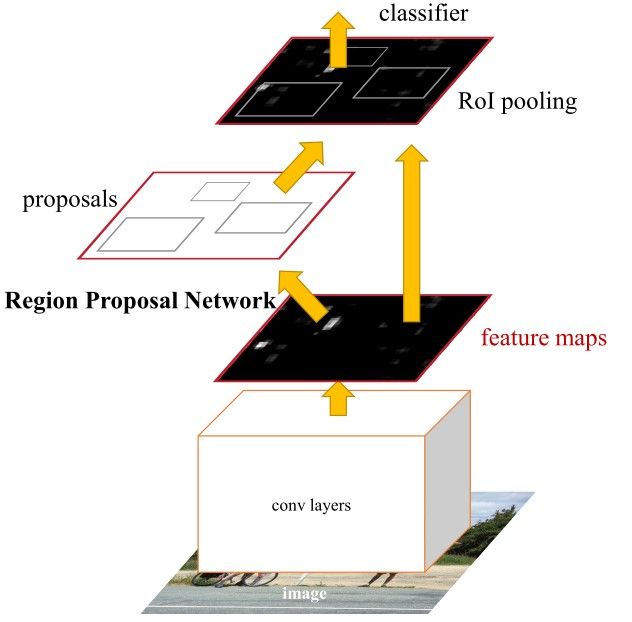


Figure : Faster RCNN architecture for object Detection.

### Region Proposal Network

The RCNN and Fast RCNN are comes with the selective search algorithm for the extraction of region proposals from the image. The extracted region is passed to the CNN that classify it. But the Faster RCNN comes with its own region proposal network (RPN) that proposed the regions. It has the following benefits

1. The RPN module can generate the region that may be customized according to the mask of detection object.
2. The RPN module also share the same convolutional layers used by the detection module. Hence the RPN module did not require the extra time for training unlike the selective search.
3. As the RPN and detection module share the similar convolutional layer, both models can be merged into single model and trained only once.

The final convolution shared with the Fast R-CNN serves as the foundation for the RPN's work. The feature map is traversed through a moving window based on a n\*n square rectangular window. Several potential region recommendations are produced for each window. These proposals will be evaluated based on their "objectivity score," therefore they do not represent the final proposals.

### Anchor Box

The feature map of the final common convolution layer is shown in the following Figure # to be passed via a rectangular moving window of size n\*n, where the size of n is 3 for the VGG-16 network. Each window generates the k region proposals that are specified according the predefined reference box also known as anchor box. For the parametrization of generated regions, the parameters are scaling and aspect ratio. Normally, three different scales and three distinct aspect ratios are used that generates the total nine regions for each window. The number of generated regions from each window can be vary and each generated region must be different from other in term of scale and aspect ratio. The example of anchor variation is presented in below Figure 6.

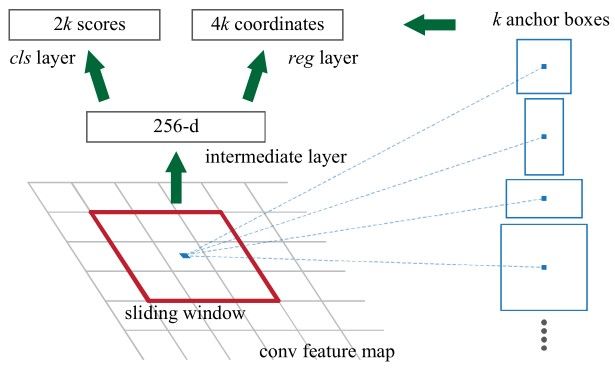


Figure : Extraction of Regions with variation of anchors.

### Objectness Score

The output layer of Faster RCNN returns the 2 values for each proposed region. If the first value is one and second is zero then the proposed region is labeled as background. If the first value is 0 and second value is 1 then the proposed region will be labeled as object. By using the technique of area intersection over union (IOU), each region is assigned by the positive and negative Objectness value for the training od region proposal network. The IOU is the fraction of common area between anchor box and ground truth box over the union of both boxes. The IOU range value is 0.0 to 1.0. If the both boxes are separate from each other then IOU have the 0.0 value that increase gradually with the closeness of both boxes and reached to 1.0 when both boxes are overlapped. The Objectness score pf region proposal as negative or positive is assigned based on the following four conditions using IOU:

1. If the IOU value of anchor box and ground truth box is greater then 0,.7 then the positive label is assigned to the anchor box.
2. If the overlap value of the anchor is not more than the 0.7 then the anchor is labeled with positive but the highest value will assign to the anchor.
3. If the IOU value of anchor is not more then the 0.3 then the anchor assigned with negative value.
4. The anchor is assigned not positive or negative if the overlap area of the anchor with ground truth is more then 0.3 but not more then 0.5. These anchors are considered as background and did not contribute in the training of the model.

The summary of the all conditions described above are presented by the below equations:

### Feature Sharing of RPN and Faster RCNN

The main two component of the Faster RCNN model are region proposal network and classification model. Both component of the model is independent and can be learned individually. But for the Faster RCNN architecture, it is possible to build a single network that used by the RPN and classification model. In addition, the single network will train only once from both components. This idea is also used by the RPN and Faster RCNN components that use the same convolutional network. The weights of the model trained only once but shared by the both components and this technique is known as feature sharing.

### Region of interest (ROI)

The region of interest (ROI) is the portion of the image or the dataset that has been selected for the specific objective. Any of following may make up a dataset: the waveform or the single Dimensional dataset on waveform, an ROI is the time and frequency interval (the graph of little quantity plotted against time). a picture and 2-dimensional dataset.  The borders on the sketch or the image of item that make up an ROI are specified. These ROI are surfaces or a contour that define the physical item, whether it is the volume and 3dimensional dataset. A ROI is a 3dimensional dataset for the particular moment or span of time when it comes to varying 3-dimensional dataset of the item altering in the shape over duration. The Medical images are where ROI is very frequently utilised, such example as the specific area of the 2-dimensional, 3-dimensional, or 4-dimensional images that is the concern throughout the diagnostic and of significance during studies. Whenever the organ's movements have ROI for the physician's assessment or the researcher's investigation may depend on how an object travels throughout the particular duration and the period of time.

### Training of faster RCNN

Three methods were described in Faster-RCNN research paper [3] for training the both a RPN as well as Fast-RCNN despite utilizing convolution layers:

* An alternating training phase.
* An approximating joint training phase.
* The non-approximating joint training phase.

**An alternating training phases**

The very first technique is known as the alternating training, and it involves initially learning RPN to produce regional suggestions. The pretrained network on the ImageNet is used to initialise overall weights of shared convolution layers. The RPN's additional parameters are initially initialised at random bases. These weights of RPN as well as the sharing convolution layers are adjusted just after RPN generates boxes of regional proposals. The Faster R-CNN unit is trained using a RPN's produced proposals. In instance, some tuned parameters are used by an RPN to initialise some weights of shared convolution layers. The initialization of other Faster R-CNN parameters is randomly. The sharing layers' including the weights of Faster R CNNs are both adjusted during training. This RPN is once more trained using a fine-tuned parameters from a sharing layer. Alternate learning, which is used in every test, is a preferred method of instructing two components.

**An approximating joint training phase**

Second technique, referred to as approximation of joint-training, treats a RPN with Faster R-CNN as one system rather than two distinct components. In this instance, a RPN creates a regional proposal. These proposals then input straight into a Faster R-CNN, that determines a position of objects, while changing these weights of either an RPN or a sharing layer. The overall weights of Fast R-CNNs were adjusted only when a Faster R-CNN generates its outcomes. These gradients of every weight without reference to regional proposals are disregarded since overall weights of RPN a as well as sharing layers are really not modified after regional proposals are formed. This makes this procedure less accurate than the initial technique (even when outcomes remain near). On a contrary side, there is the 25–50% reduction in network training time.

**The non-approximating joint training phase**

Using the ROI Deformation layer, overall gradients of a weights according to suggested anchor boxes can be computed in approximation joint training technique.

## ResNET\_50

Residual networks are referred as ResNet and in 2015 it is proposed in the research paper titled as Deep residual learning for the image recognition in digital image processing domain. This neural network was proposed by the Kaiming He, Shaoqing Ren, Xiangyu Zhang and Jian. This method was incredibly effective, as evidenced by this fact that all ensembles took first place at ILSVRC-2015 classifying challenge with just 3.57% inaccuracy. Furthermore, it won 2015 ILSVRC including COCO contests for an ImageNet detection, ImageNet localization, COCO detection, as well as COCO segmentation. There are other versions of the ResNet which use a same basic idea but have various count of layers. The form which can function using 50 artificial neural network-based layers is referred known as a Resnet50. Artificial intelligence specialists add extra layers while using deep convolution neural network to address the computer vision challenge. Because various layers may be program for the variety of jobs to produce very precise outcomes, such additional layer's aid in more effective solution of complicated issues. Although number of the stacked layers might enhance a model's characteristics, the deep network can reveal degradation problem. In other words, as neural networking layer count rises, performance levels may eventually become overloaded as well as begin to gradually deteriorate. As a consequence, this model's accuracy declines in both a training dataset and a testing dataset. The overfitting did not cause this deterioration. Alternatively, it can be a result of network's configuration, the optimization mechanism, or much more crucially an issue with the vanishing or exploding gradients.

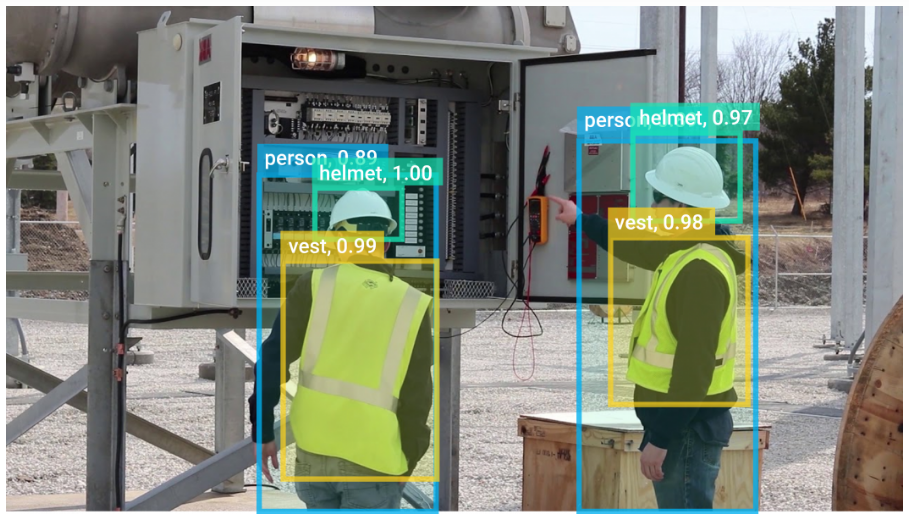


Figure : Example of ResNet 50 working

The ResNet was developed by using the intent of solving the particular issue. Remaining blocks are used in the deep-residual networks to boost algorithms precision. The advantage of such kind of artificial neural networks is an idea of "skips connections," that is on the foundation of the residual-blocks.

These skips connections operate in 2 different manners. Initially, authors resolve this problem of a vanishing-gradient by creating a different shortcut for gradient to use. They also give a model an ability to learns the identity function. By doing that, it is made clear this the network's increased levels don't function any worse than its low layers. In summary, layers acquire an identity-functions much more quickly thanks to residual-blocks. The ResNet hence reduces the number of inaccuracies without increasing an effectiveness of the deep neural network with additional neural network layers. To look at it another way, these skipped connections combine these results of earlier layers with the results of these stacked layers, enabling the training of far deep networks than was before feasible.

### ResNet -34 architecture

A Resnet-34 structure, which utilised inserting shortcuts connections to transform the plain model into a corresponding residual-network, was an initial ResNet structure. Throughout this instance, the convolution network included the 33 filters, whilst a plain structure was influenced by the VGG artificial neural network (VGG-16 and VGG-19). The ResNet, nevertheless, are simpler as well as contain some filters than the VGG-Net. Overall efficiency of a 34-layers ResNet is 3.6 billion FLOPs, as opposed to the 1.8 billion FLOPs for shorter 18-layers ResNet. It also adhered to 2 basic designs principles. For similar output features map dimension, each layer has a same count of filtering, as well as to maintain temporal complexity for each layer, a number of the filters twice when an output features map dimensions is cut in half. There were 34 total weighted layers in it. This basic networking now has shortcut interconnections. An Identity shortcut was employed directly even though an input as output size was the exact same. There were two possibilities to take into account as size increased. An initial was because identity mappings would be still performed by shortcut whereas additional zero values would be buffered to account for growing dimensionality. The projecting shortcuts to matching sizes was alternate choice.

### Structure of the ResNet-50

A Resnet50 framework is depends on concept mentioned previously, however there is a significant distinction. Because of worries about the length of temporal required to training of layers, a building-block in the instance was changed into the bottleneck architecture. Rather than a preceding two layers, utilised the stack of three. In order to create a Resnet 50 design, every of 2-layers blocks in the Resnet34 were changed to the 3-layers bottleneck block. Compared to 34-layers ResNet network, it has substantially high accuracy. The throughput of this 50-layers ResNet is 3.8 billion FLOPS.

# Problem Statement

The proposed study used the Faster RCNN and yolo v5 model for the handwritten digit recognition on the cover sheet of exam paper. The proposed model will follow the technique of segmentation-based recognition for the detection and classification oh handwritten digits.

# Methodology

The methodology section will discuss the dataset, preprocessing of the dataset and the procedure for the recognition of handwritten digits with deep learning models.

## Dataset

For the detection of handwritten digits recognition, a custom dataset was prepared by capturing the snapshots of cover sheet. In the dataset, each captured image is based on the cover sheet of exam paper that contain the grades of the student against different question. The grades of the student are marked in a grid on the exam sheet with handwritten digits range of 0-9. Each cell of the grid contains the grades of student that have 0 minimum value and 9 maximum values as show in Figure 8. The total ## images for distinct handwritten cover sheets were captured.

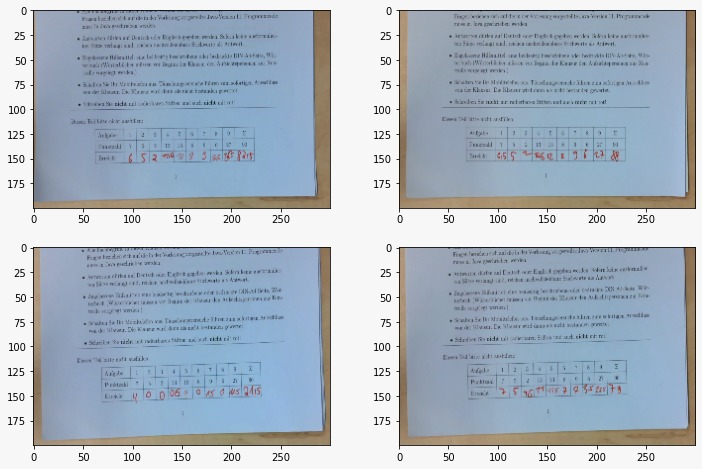


Figure : Overview of Dataset Samples.

## Data Preparing and Preprocessing

After capturing the images of exam paper cover sheet, the grid area of the images was annotated by the annotated tool. In the annotation of images, each digit was annotated separately as show in Figure 9. For instance, if they obtained mark of the student is 6.5 then the two bounding boxes were created for 6 and 5 digits separately. The proposed study did not consider the floating point in the recognition of digits. By following the annotation of the dataset, the preprocessing of the dataset was performed. In the preprocessing of the dataset, all the images were converted into the same size with the dimension 512\*900. Later, the annotation format of the annotated images was not suitable for the Faster RCNN model. The annotation format was converted into the desired format of Faster RCNN. Initially the annotation format contains the initial point and last point with x-axis and y-axis values. But the Faster RCNN model required the initial point axis value with height and width of annotation box. It calculates the last point internally with the help of height and width of the annotation. So, the current annotation format of the cover sheet images was converted into the Faster RCNN desired format.



Figure : Overview of Dataset Annotated Samples.

Further we explore the dataset for the better understanding of the proposed problem. In the dataset exploration, the class distribution of the dataset was calculated. The class distribution of the dataset reveals the number of instances against each class in the dataset. Our class distribution analysis also found out the count of annotations against each digit from 0 to 9. The class distribution of exam cover sheet dataset in plotted in the below bar chat (Figure 10).

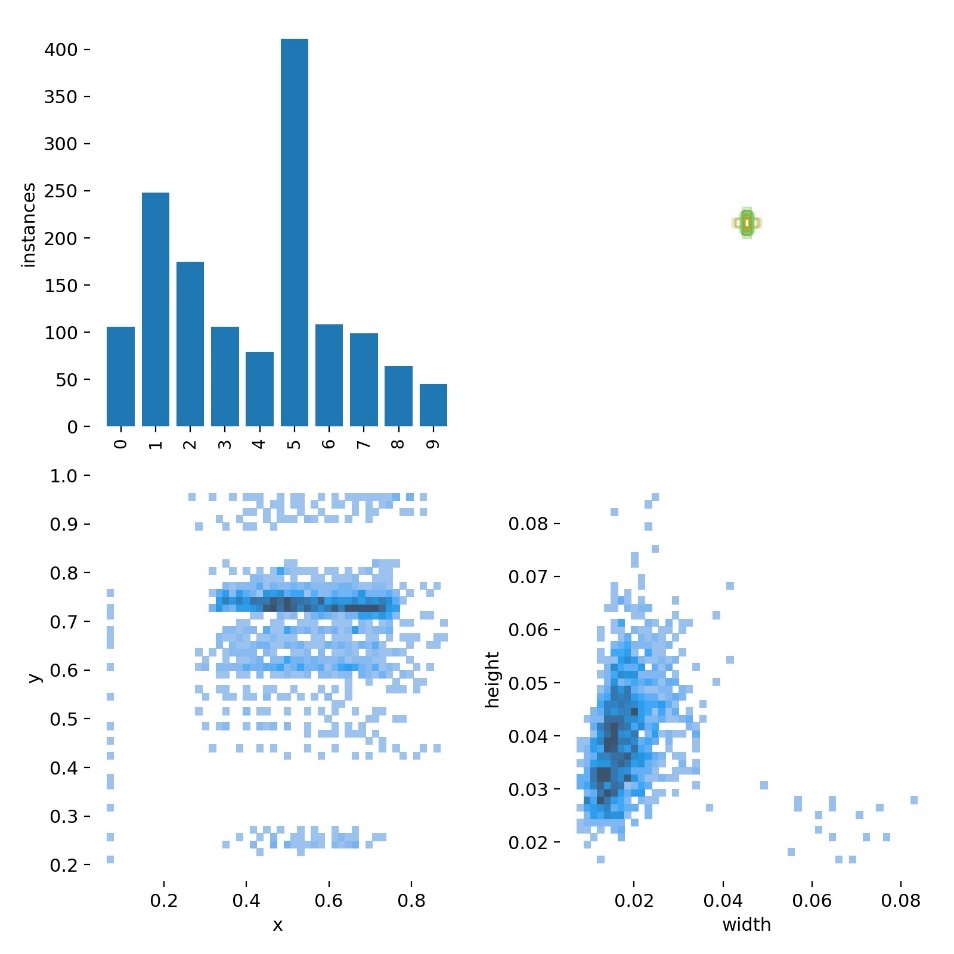


Figure : Class Distribution in Dataset.

## Train Test Split

By following the complete preprocessing of the dataset, the dataset was split into two different sub groups labeled as train set and test set. For splitting the dataset into two different groups we used the train test split module of python library named as scikit learn. It takes the dataset and divide the data into first group with some ration and rest of the dataset into the second group. We divide our exams cover sheet dataset into training and testing set with the ratio of 80% and 20% respectively. The train test split module divided the 80% of the samples into training set by randomly selecting the samples and rest of the 80% samples into the test set. After splitting the dataset, the train and test groups contain the # and # cover sheet images respectively.

## Model Development and Training

For the recognition of handwritten digits on the exams cover sheet, two different deep learning models named as Faster RCNN and Yolo V5 was used. We did not develop the model from scratch. For the recognition of digits, the proposed study used the transfer learning approach. In transfer learning approach, pretrained model was loaded with its trained weights. The dimensions of the input and out layer was changed according to the selected dataset and start training with the pretrained weights of the model.

In the proposed study we also used the pretrained Faster RCNN and Yolo V5 model for handwritten digits recognition. The Faster RCNN model was used with the RPN, FPN\_3x, and ResNet50 module. The RPN module of the Faster RCNN was used to extract the regional proposal that were converted into the feature set by the last ROI pooling layer of module. As the Faster RCNN uses the multiple anchors of different ratios and scales that generates the multiple features maps. For selecting the most suitable feature map among the multiple generated features map, FPN\_3x strategy was used to pick the most suitable feature map for proposed region.

For yolo v5 model, we also load the small weights on the architecture of the model. We did not freeze any layer of the model and train all the layer of the model for digits recognition. After loading the initial weights of the model, the input and output dimension of the model was adjusted according to the image size and number of classes respectively. The tensor board representation of yolo v5 model is presented in below Figure 11 (extracted from [52]).

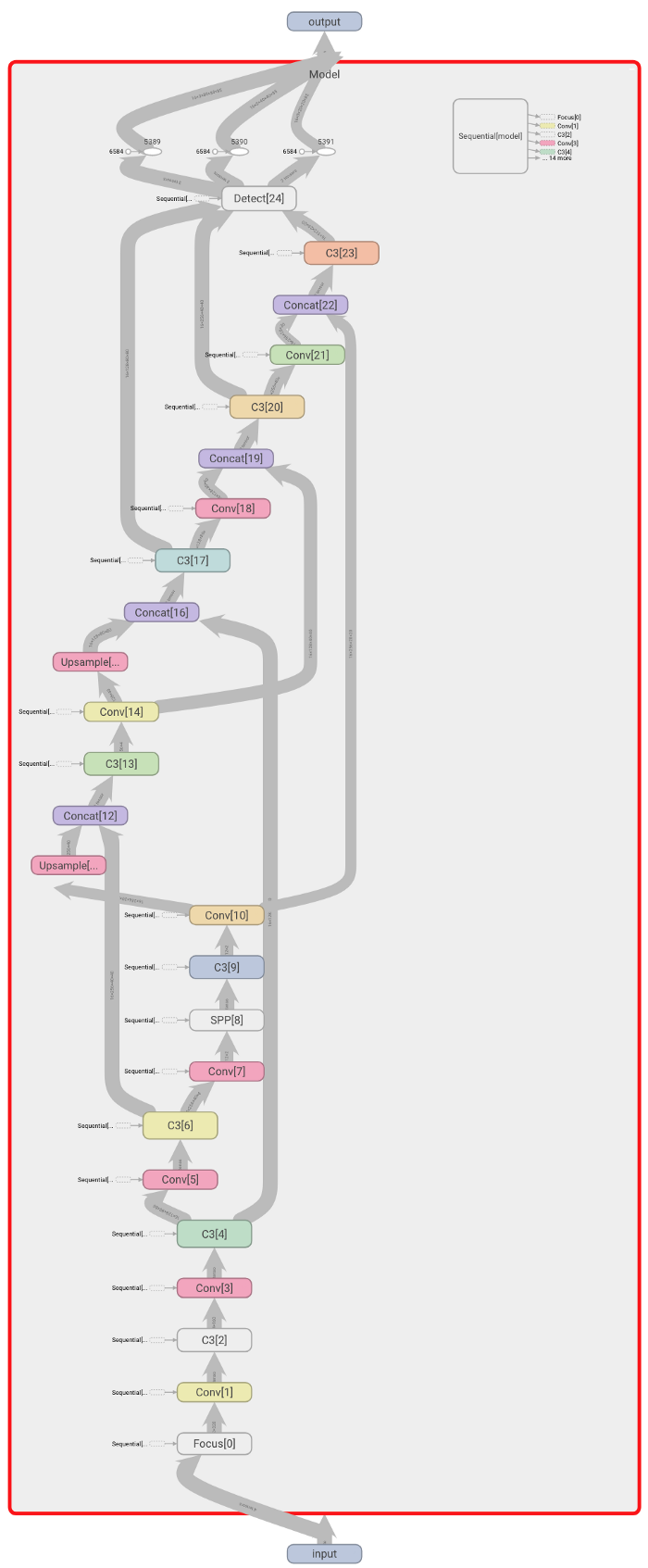


Figure : Architecture of Yolo v5 Model.

By following the initializing of both object detection models (Faster RCNN and Yolo v5), the training samples of the dataset was used for the training of the model. The default value of all the hyper parameters was used at the time of the training. After the training of both models, test group images were used to evaluate the performance of the model.

## Evaluation Measures

Evaluation measures include the criteria or strategy to measure the performance of applicant. In the proposed study, average precision (AP) and mean average precision (MAP) was used to evaluate the performance of trained models. Precision is the fraction of accurately predicted the positive instances over the all-predicted instances as positive as shown in below equation. Average precision is the weighted average of precision for each class individually on different threshold in case of object detection. While the mean average precision is the mean of all AP. The below equations show the formula for calculating the precision, average precision and mean average precision.

# Results

For the recognition of handwritten digits recognition on the cover sheet, the Faster RCNN and yolo v5 models were trained n the proposed study. The below section will discuss the extracted results of both models in detailed.

## Faster RCNN Results

Faster RCNN model with RPN, FPN\_3x and ResNet\_50 module for the recognition of handwritten digits on the exam sheets. The RPN module used for the extraction of regions from input image and FPN use to assign the ROI to features map and the ResNet50 was used for the classification of the proposed regions. By following the complete development of the Faster RCNN model, the model was trained with the # training samples of the handwritten digits dataset. The learning of the model was gradually decreased with the training of the of the model as shown in Figure 12.

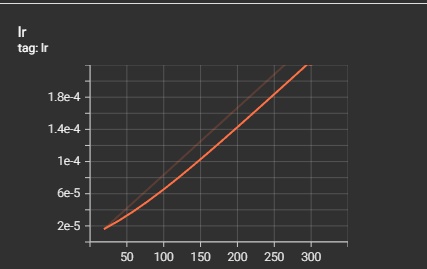


Figure : Bar Chat of learning rate with the training of model.

The faster RCNN model used the Adam as optimizer, categorical cross entropy as loss function and accuracy as evaluation matrix for training. As the classification model of Faster RCNN returns the two values, one for region and second for object type in the region, Faster RCNN also return two values that help in finding the accuracy and loss of regions and objects. By following the complete learning of the model, Faster RCNN showed the 0.82 accuracy score for objects classes and 0.25 class accuracy of foreground accuracy respectively. Moreover, the trained model also showed the 0.05 and 0.6 loss for region boxes and objects types in Figure 13.

|  |  |
| --- | --- |
|  |  |
|  |  |

Figure : Accuracy and loss plots of the Faster RCNN model.

## Yolo V5 Results

For the handwritten digit recognition on the cover sheets of exam paper, yolo v5 model was also trained on the selected dataset. The # training samples of the dataset was used for the learning of the model. The small initial weights of the model at the time of training. The 0.01 learning rate, with Adam optimizer was used for the training of the model. The model was also trained with 100 epochs and 10 batch size. The trained model was evaluated using the test images of the dataset and model showed the % accuracy score. The confusion matrix of the model is presented in Figure 14 that reveal that the model did not significantly perform for different classes. As the confusion matrix showed that the 60% samples of digit 5 and 40% samples of background are accurately classified while the rest of the digits were not classified by the model.

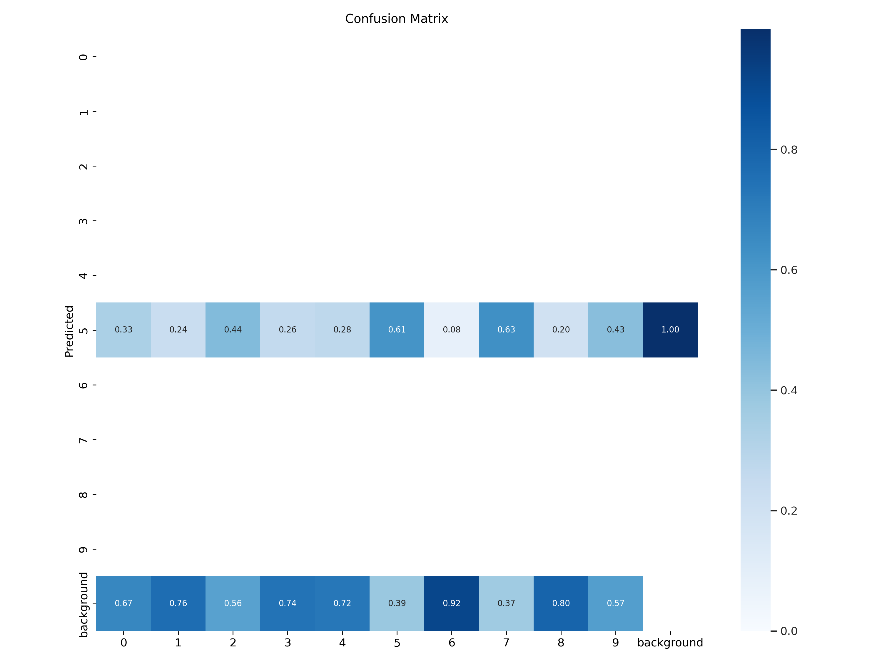


Figure : Confusion Matrix of Yolo V5 Model.

The precision and recall of the model also calculated for the trained yolo v5 model. The plot of the of the precision, recall and precision-recall during the training of the model on validation set is presented in Figure 15. The model showed the 0.038, 0.035 and 0.040 box loss, object loss and class loss for training data respectively. While the model showed the 0.031, 0.030 and 0.038 box loss, object loss and class loss for test images respectively. The plot of all the losses on train and test instances during the learning of the model is presented in Figure 15.

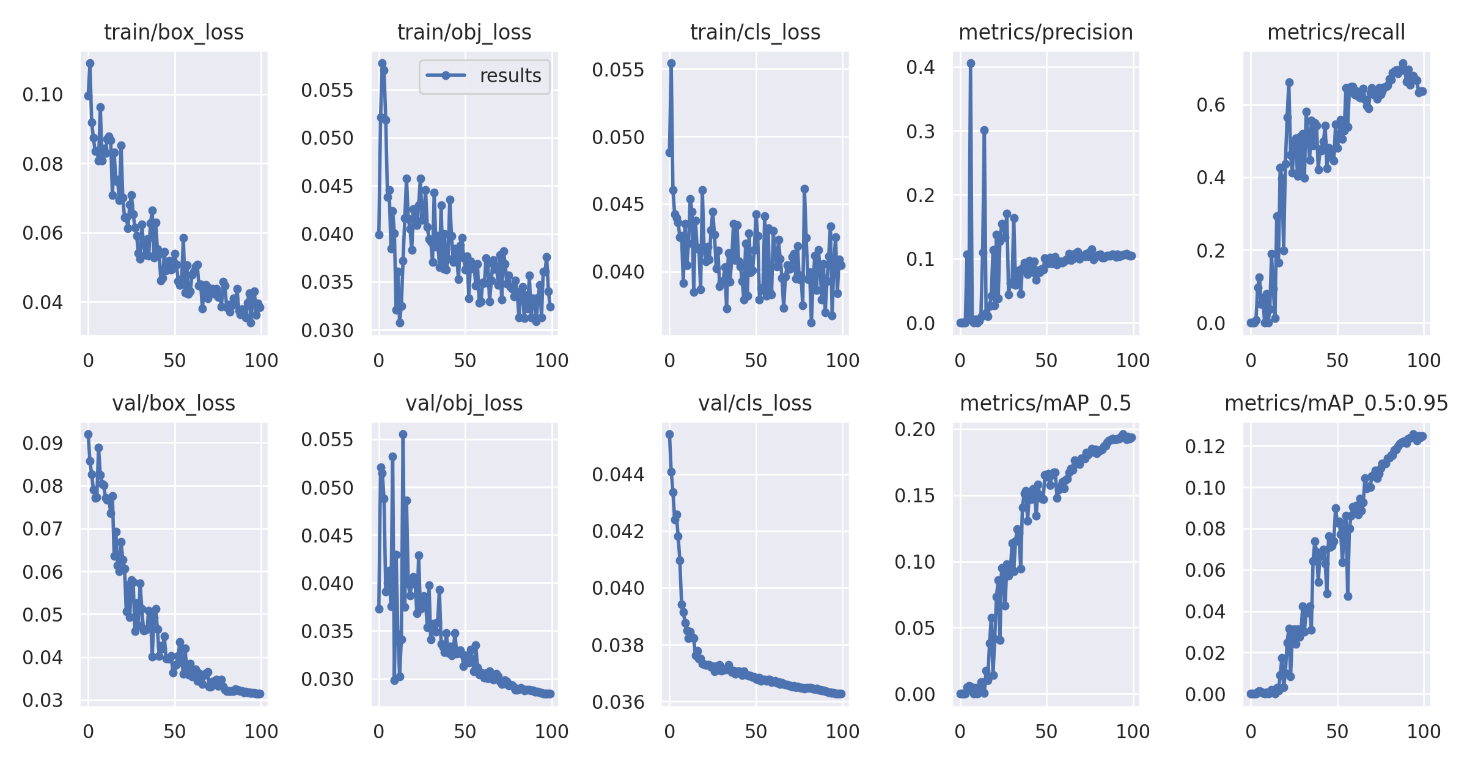


Figure : Accuracy and Loss Plots of the Yolo v5 Model.

The precision recall and the f1-score of the yolo v5 trained model was also calculated with different confidence scores. The confidence score shows the classifier's level of assurance and the likelihood that the box includes an object of interest. In a perfect world, the confidence score would be zero if there was no object in that box. For tighter bounding boxes, the confidence score often tends to be greater. The plots of the precision, recall and f1-score with 0 to 1 confidence score are shown in Figure 16.

|  |  |
| --- | --- |
|  |  |
|  | |

Figure : Precision, Recall and F1 Curve During Training.

Lastly, the few samples from the test group were predicted with the trained yolo v5 model and then illustrated using the matplotlib library of python for demonstration purpose. The predicted samples by the yolo v5 model are also shown in below Figure 17.

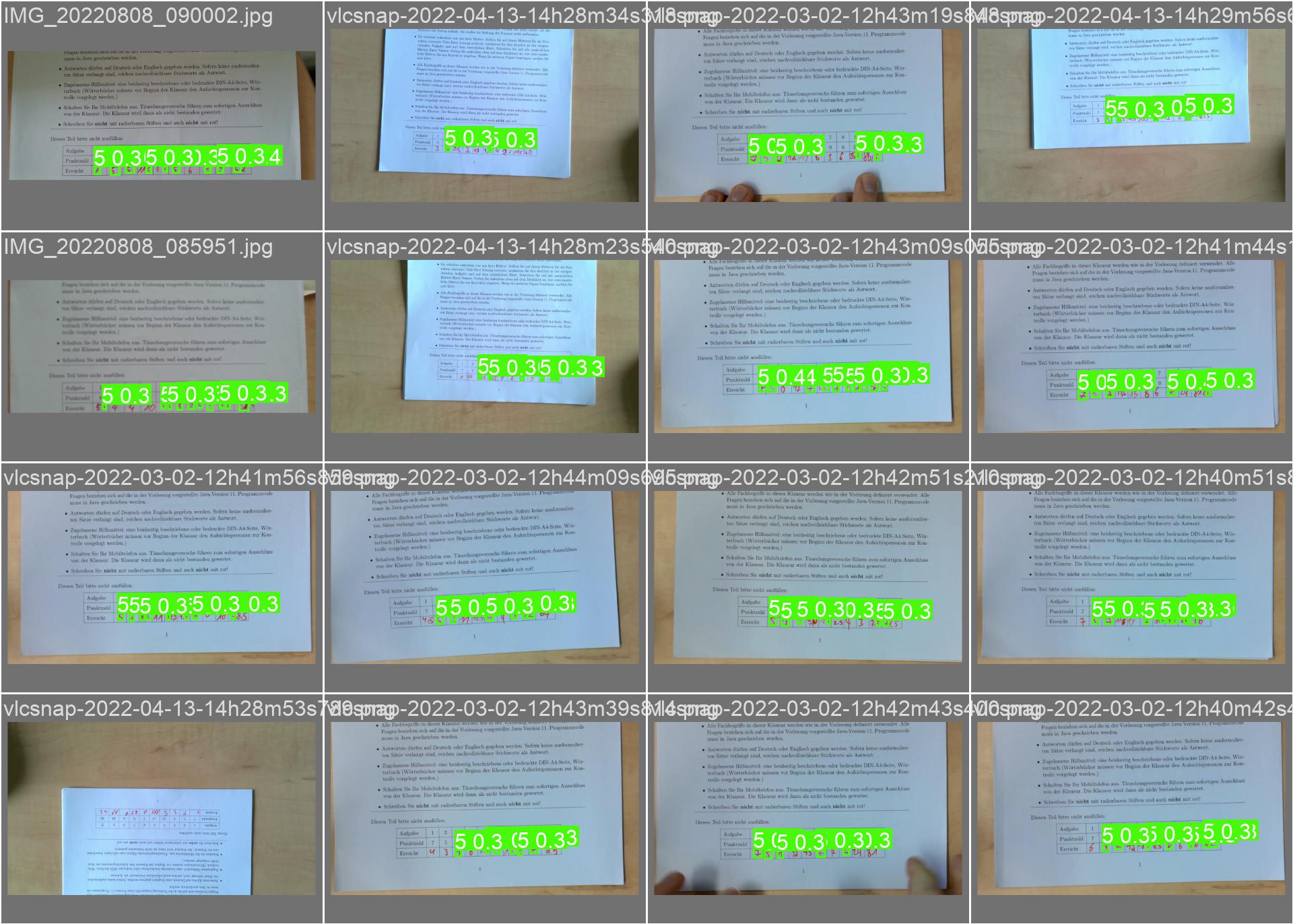


Figure : Overview of Yolo v5 Predicted samples.

## Comparative Analysis of Both Models

By following the training and evaluation procedure of both models, a comparative study was performed. The purpose of the comparative study is to analyze the best model for the recognition of handwritten digits on the cover sheet. The criteria for selecting the optimal model, accuracy and mean average precision evaluation measure were selected. On the basis of these evaluation measure, both models were analyzed and pick the best model with highest evaluation scores.

As the results section of both models clearly represent that the accuracy of Faster RCNN was 85% and the accuracy of Yolo v5 was #. There was a huge difference in the accuracy score of the both models. Moreover, the yolo v5 model was unable to detect the almost all classes except the background and digit 5 class. Collectively the performance of the Faster RCNN is more robust compare to the yolo v5 model on the basis of accuracy score. Similarly, the mAP of the Faster RCNN was # while the mAP of yolo v5 was #. By comparing the both evaluation measures, the proposed study reveal the Faster RCNN as best model for the recognition of offline digits on the cover sheet of exam papers. The comparative results of both models are also presented in Table 1.

Table : Selected Evaluation measures of both Models.

|  |  |  |
| --- | --- | --- |
| Evaluation Measure | Faster RCNN | Yolo v5 |
| Average Precision | 0.75 | 0.19 |
| Mean Average Precision (50%) | 0.72 | 0.10 |

The extracted evaluation measures of the models were also plotted comparatively for the comparison of both models. The comparison bar chat of both models based on extracted evaluation scores is presented in Figure #.

Although, the accuracy and other evaluation measures are lower values compare to the research work published in the literature. Numerous studies have already published with more than 90% accuracy score and significant score for other evaluation measures. But all the published wok is based on the numerous databases i.e., these studies used the sample data from different dataset bases for the training of the proposed model. The parameters of real-world environment are very heterogeneous and complex. Majority of the proposed work with significant accuracy score are unable to recognize the offline characters in real world environment. There are many reasons that involved in the inconsistency of the model. One of the main reasons is the non-multi-modality of the dataset. In other words, the real-world data is very heterogeneous, noisy, and unclear but the existing databases data did not. Due to these factors, the high accuracy models are unable to perform in real world environment. In the proposed study we used customized data of real-world environment with multi-modality factor, noise factor and blurring factor. The proposed study trained the model with noisy and blurred that cause the fall in accuracy score. But the trained model on our customized dataset will perform similarly in real world scenarios as performing in the controlled environment. Although, there is a need to robust the both model or proposed new model for significant result with the customized dataset in future studies.

# Conclusion

In the proposed work, handwritten digits on the cover sheet of the exam paper were recognized. For the recognition of the offline digits, two well-known deep learning model labeled as Faster RCNN and Yolo v5 were proposed. A custom dataset was prepared by capturing the images of cover sheet for the training of the both models. Faster RCNN model showed the overall # accuracy score while the yolo v5 model showed the # accuracy score. As the results section showed that the yolo v5 results are not significant but the Faster RCNN model perform well compare to the yolo v5 model. Although, the performance of the faster RCNN is lower relative to the online published study but the use of noisy and unclear data for the training of the model make is more robust compare to the other models. The # # score is slightly low relative to other studies but the customized dataset makes it robust enough to perform similarly in real world environment unlike the high score published studies. Finally, the proposed work concluded that current score of the Faster RCNN with our customized dataset is more robust to recognize the handwritten digits on cover sheets relative to models of published studies.

# References

[1] K. T. Mane and V. G. Pujari, “Signature matching with automated cheque system,” *2013 International Conference on Intelligent Systems and Signal Processing, ISSP 2013*, pp. 166–169, 2013, doi: 10.1109/ISSP.2013.6526895.

[2] S. N. Srihari and S. Lee, “Automatic handwriting recognition and writer matching on anthrax-related handwritten mail,” *Proceedings - International Workshop on Frontiers in Handwriting Recognition, IWFHR*, pp. 280–284, 2002, doi: 10.1109/IWFHR.2002.1030923.

[3] P. G. Biondich, J. M. Overhage, P. R. Dexter, S. M. Downs, L. Lemmon, and C. J. McDonald, “A modern optical character recognition system in a real world clinical setting: some accuracy and feasibility observations.,” *Proceedings of the AMIA Symposium*, p. 56, 2002, Accessed: Oct. 10, 2022. [Online]. Available: /pmc/articles/PMC2244242/?report=abstract

[4] K. Chinnasarn and Y. Rangsanseri, “Image-processing-oriented optical mark reader,” *https://doi.org/10.1117/12.365883*, vol. 3808, pp. 702–708, Oct. 1999, doi: 10.1117/12.365883.

[5] M. E. R. Bezerra and A. L. I. de Oliveira, “Optical character recognition,” *http://dx.doi.org/10.3810/pgm.1998.03.410*, pp. 207–230, Jan. 2015, doi: 10.3810/PGM.1998.03.410.

[6] Stephen. Haag and Maeve. Cummings, “Management information systems for the information age,” p. 554, 2013, Accessed: Oct. 10, 2022. [Online]. Available: https://books.google.com/books/about/Management\_Information\_Systems\_for\_the\_I.html?id=Vk7HygAACAAJ

[7] “Ogami OMR - Free download and software reviews - CNET Download.” https://download.cnet.com/Ogami-OMR/3000-10743\_4-75670602.html (accessed Oct. 10, 2022).

[8] “Remark Office OMR Software · Remark Software.” https://remarksoftware.com/products/office-omr/ (accessed Oct. 10, 2022).

[9] T. M. Breuel, “A system for the off-line recognition of handwritten text,” *Proceedings - International Conference on Pattern Recognition*, vol. 2, pp. 129–134, 1994, doi: 10.1109/ICPR.1994.576889.

[10] “Neural Networks for Pattern Recognition - Christopher M. Bishop, Professor of Neural Computing Christopher M Bishop - Google Books.” https://books.google.com.pk/books?hl=en&lr=&id=T0S0BgAAQBAJ&oi=fnd&pg=PP1&dq=+Neural+Networks+for+Pattern+Recognition.&ots=jN8YwIayok&sig=O0IVXtsTBMwliG\_6S4wJZ2BtdLw&redir\_esc=y#v=onepage&q=Neural%20Networks%20for%20Pattern%20Recognition.&f=false (accessed Oct. 13, 2022).

[11] L. Heutte *et al.*, “Multi-Bank Check Recognition System: Consideration on the Numeral Amount Recognition Module,” *http://dx.doi.org/10.1142/S0218001497000251*, vol. 11, no. 4, pp. 595–618, Nov. 2011, doi: 10.1142/S0218001497000251.

[12] T. M. Ha, M. Zimmermann, and H. Bunke, “Off-line handwritten numeral string recognition by combining segmentation-based and segmentation-free methods,” *Pattern Recognit*, vol. 31, no. 3, pp. 257–272, Mar. 1998, doi: 10.1016/S0031-3203(97)00050-2.

[13] H. Tokahashi and T. D. Griffin, “Recognition enhancement by linear tournament verification,” pp. 585–588, Dec. 2002, doi: 10.1109/ICDAR.1993.395667.

[14] M. Brown, “Automated Grading of Handwritten Numerical Answers,” *Computer Engineering*, Jun. 2017, Accessed: Oct. 13, 2022. [Online]. Available: https://digitalcommons.calpoly.edu/cpesp/221

[15] K. W. Murray and N. Orii, “Automatic Essay Scoring”, Accessed: Oct. 13, 2022. [Online]. Available: http://freevocabulary.com/

[16] M. Alomran and D. Chai, “Automated Scoring System for Multiple Choice Test with Quick Feedback,” *International Journal of Information and Education Technology*, vol. 8, no. 8, pp. 538–545, 2018, doi: 10.18178/IJIET.2018.8.8.1096.

[17] M. Čupić, K. Brkić, T. Hrkać, Ž. Mihajlović, and Z. Kalafatić, “Automatic recognition of handwritten corrections for multiple-choice exam answer sheets,” *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2014 - Proceedings*, pp. 1136–1141, 2014, doi: 10.1109/MIPRO.2014.6859739.

[18] S. Srihari, J. Collins, R. Srihari, H. Srinivasan, S. Shetty, and J. Brutt-Griffler, “Automatic scoring of short handwritten essays in reading comprehension tests,” *Artif Intell*, vol. 172, no. 2–3, pp. 300–324, Feb. 2008, doi: 10.1016/J.ARTINT.2007.06.005.

[19] M. Mahana, M. Johns, and A. Apte, “Automated Essay Grading Using Machine Learning”.

[20] C. Saengtongsrikamon, P. Meesad, and S. Sodsee, “Scanner-Based Optical Mark Recognition,” *undefined*, 2009.

[21] J. M. Alghazo, G. Latif, L. Alzubaidi, and A. Elhassan, “Multi-language handwritten digits recognition based on novel structural features,” *Journal of Imaging Science and Technology*, vol. 63, no. 2, 2019, doi: 10.2352/J.IMAGINGSCI.TECHNOL.2019.63.2.020502.

[22] J. M. Alghazo, G. Latif, A. Elhassan, L. Alzubaidi, A. Al-Hmouz, and R. Al-Hmouz, “An Online Numeral Recognition System Using Improved Structural Features-A Unified Method for Handwritten Arabic and Persian Numerals”.

[23] D. Chai, “Automated marking of printed multiple choice answer sheets,” *Proceedings of 2016 IEEE International Conference on Teaching, Assessment and Learning for Engineering, TALE 2016*, pp. 145–149, Feb. 2017, doi: 10.1109/TALE.2016.7851785.

[24] J. Muangprathub, O. Shichim, Y. Jaroensuk, and S. Kajornkasirat, “Automatic Grading of Scanned Multiple Choice Answer Sheets,” *International Journal of Engineering & Technology*, vol. 7, no. 2.23, pp. 175–179, Apr. 2018, doi: 10.14419/ijet.v7i2.23.11910.

[25] S. Patole, A. Pawar, A. Patel, A. Panchal, and J. 5 Ravindra, “AUTOMATIC SYSTEM FOR GRADING MULTIPLE CHOICE QUESTIONS AND FEEDBACK ANALYSIS,” *International Journal of Technical Research and Applications*, vol. 39, pp. 16–19, 2016, Accessed: Oct. 13, 2022. [Online]. Available: www.ijtra.com

[26] A. M. Tavana, M. Abbasi, and A. Yousefi, “Optimizing the correction of MCQ test answer sheets using digital image processing,” *2016 8th International Conference on Information and Knowledge Technology, IKT 2016*, pp. 139–143, Dec. 2016, doi: 10.1109/IKT.2016.7777754.

[27] A. A. Abbas, “An Automatic System to Grade Multiple Choice Questions paper based exams”.

[28] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, “Convolutional neural network committees for handwritten character classification,” *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 1135–1139, 2011, doi: 10.1109/ICDAR.2011.229.

[29] G. Latif, J. Alghazo, L. Alzubaidi, M. M. Naseer, and Y. Alghazo, “Deep Convolutional Neural Network for Recognition of Unified Multi-Language Handwritten Numerals,” *2nd IEEE International Workshop on Arabic and Derived Script Analysis and Recognition, ASAR 2018*, pp. 90–95, Oct. 2018, doi: 10.1109/ASAR.2018.8480289.

[30] N. Singh, “An Efficient Approach for Handwritten Devanagari Character Recognition based on Artificial Neural Network,” *2018 5th International Conference on Signal Processing and Integrated Networks, SPIN 2018*, pp. 894–897, Sep. 2018, doi: 10.1109/SPIN.2018.8474282.

[31] P. Kumar, N. Sharma, and A. Rana, “Handwritten Character Recognition using Different Kernel based SVM Classifier and MLP Neural Network (A COMPARISON),” *Int J Comput Appl*, vol. 53, no. 11, pp. 975–8887, 2012.

[32] Z. Rao *et al.*, “Research on a handwritten character recognition algorithm based on an extended nonlinear kernel residual network,” *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 12, no. 1, pp. 413–435, Jan. 2018, doi: 10.3837/TIIS.2018.01.020.

[33] S. H. Jeong, Y. S. Nam, and H. K. Kim, “Non-similar candidate removal method for off-line handwritten Korean character Recognition,” *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, vol. 2003-January, pp. 323–328, 2003, doi: 10.1109/ICDAR.2003.1227682.

[34] G. A. Montazer, H. Q. Saremi, and V. Khatibi, “A neuro-fuzzy inference engine for Farsi numeral characters recognition,” *Expert Syst Appl*, vol. 37, no. 9, pp. 6327–6337, Sep. 2010, doi: 10.1016/J.ESWA.2010.02.088.

[35] S. Impedovo, F. M. Mangini, and G. Pirlo, “A new adaptive zoning technique for handwritten digit recognition,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8156 LNCS, no. PART 1, pp. 91–100, 2013, doi: 10.1007/978-3-642-41181-6\_10/COVER.

[36] F. M. Mangini and G. Pirlo, “ADAPTIVE ZONING DESIGN BY SUPERVISED LEARNING USING MULTI-OBJECTIVE OPTIMIZATION,” *http://dx.doi.org/10.1142/S1469026814500047*, vol. 13, no. 1, Mar. 2014, doi: 10.1142/S1469026814500047.

[37] F. A. Al-Omari and O. Al-Jarrah, “Handwritten Indian numerals recognition system using probabilistic neural networks,” *Advanced Engineering Informatics*, vol. 18, no. 1, pp. 9–16, Jan. 2004, doi: 10.1016/J.AEI.2004.02.001.

[38] J. H. Alkhateeb and M. Alseid, “DBN - Based learning for Arabic handwritten digit recognition using DCT features,” *2014 6th International Conference on Computer Science and Information Technology, CSIT 2014 - Proceedings*, pp. 222–226, 2014, doi: 10.1109/CSIT.2014.6806004.

[39] H. Salimi and D. Giveki, “Farsi/Arabic handwritten digit recognition based on ensemble of SVD classifiers and reliable multi-phase PSO combination rule,” *International Journal on Document Analysis and Recognition*, vol. 16, no. 4, pp. 371–386, Dec. 2013, doi: 10.1007/S10032-012-0195-7/TABLES/9.

[40] D. Musleh, K. Halawani, and S. Mahmoud, “Fuzzy Modeling for Handwritten Arabic Numeral Recognition,” *The International Arab Journal of Information Technology*, vol. 14, no. 4, 2017.

[41] S. A. Azeem, M. el Meseery, and H. Ahmed, “Online Arabic handwritten digits recognition,” *Proceedings - International Workshop on Frontiers in Handwriting Recognition, IWFHR*, pp. 135–140, 2012, doi: 10.1109/ICFHR.2012.249.

[42] J. Sadri, M. R. Yeganehzad, and J. Saghi, “A novel comprehensive database for offline Persian handwriting recognition,” *Pattern Recognit*, vol. 60, pp. 378–393, Dec. 2016, doi: 10.1016/J.PATCOG.2016.03.024.

[43] A. Boukharouba and A. Bennia, “Novel feature extraction technique for the recognition of handwritten digits,” *Applied Computing and Informatics*, vol. 13, no. 1, pp. 19–26, Jan. 2017, doi: 10.1016/J.ACI.2015.05.001.

[44] H. Karimi, A. Esfahanimehr, M. Mosleh, F. M. J. Ghadam, S. Salehpour, and O. Medhati, “Persian Handwritten Digit Recognition Using Ensemble Classifiers,” *Procedia Comput Sci*, vol. 73, pp. 416–425, Jan. 2015, doi: 10.1016/J.PROCS.2015.12.018.

[45] R. Sarkhel, N. Das, A. K. Saha, and M. Nasipuri, “A multi-objective approach towards cost effective isolated handwritten Bangla character and digit recognition,” *Pattern Recognit*, vol. 58, pp. 172–189, Oct. 2016, doi: 10.1016/J.PATCOG.2016.04.010.

[46] S. Basu, R. Sarkar, N. Das, M. Kundu, M. Nasipuri, and D. K. Basu, “Handwritten Bangla digit recognition using classifier combination through DS technique,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3776 LNCS, pp. 236–241, 2005, doi: 10.1007/11590316\_32/COVER.

[47] W. Wang *et al.*, “A hardware neural network for handwritten digits recognition using binary RRAM as synaptic weight element,” *2016 IEEE Silicon Nanoelectronics Workshop, SNW 2016*, pp. 50–51, Sep. 2016, doi: 10.1109/SNW.2016.7577980.

[48] S. S. Ali and M. U. Ghani, “Handwritten Digit Recognition Using DCT and HMMs,” *Proceedings - 12th International Conference on Frontiers of Information Technology, FIT 2014*, pp. 303–306, Jun. 2015, doi: 10.1109/FIT.2014.63.

[49] M. Jie, I. A. Aziz, H. Hasbullah, and S. A. B. Azizan, “Handwritten digits recognition based on improved label propagation algorith,” *2016 3rd International Conference on Computer and Information Sciences, ICCOINS 2016 - Proceedings*, pp. 345–350, Dec. 2016, doi: 10.1109/ICCOINS.2016.7783239.

[50] R. Bajaj, L. Dey, and S. Chaudhury, “Devnagari numeral recognition by combining decision of multiple connectionist classifiers,” *Sadhana 2002 27:1*, vol. 27, no. 1, pp. 59–72, 2002, doi: 10.1007/BF02703312.

[51] “Faster R-CNN Explained for Object Detection Tasks | Paperspace Blog.” https://blog.paperspace.com/faster-r-cnn-explained-object-detection/ (accessed Oct. 11, 2022).

[52] “Object Detection Algorithm — YOLO v5 Architecture | by Surya Gutta | Analytics Vidhya.” https://medium.com/analytics-vidhya/object-detection-algorithm-yolo-v5-architecture-89e0a35472ef (accessed Oct. 12, 2022).

# Appendix

## Yolo V5 Training Code

#!pip install tensorboard==2.4.1

!pip install torch #YOLOv5 runs on top of PyTorch, so we need to import it to the notebook

# !pip install torch

import torch # YOLOv5 implemented using pytorch

from IPython.display import Image #this is to render predictions

!git clone https://github.com/ultralytics/yolov5

import os

from random import choice

import shutil

#arrays to store file names

imgs =[]

xmls =[]

#setup dir names

trainPath = '/content/yolov5/dataset/images/train'

valPath = '/content/yolov5/dataset/images/val'

crsPath = '/content/images' #dir where images and annotations stored

#setup ratio (val ratio = rest of the files in origin dir after splitting into train and test)

train\_ratio = 0.8

val\_ratio = 0.02

#total count of imgs

totalImgCount = len(os.listdir(crsPath))/2

#soring files to corresponding arrays

for (dirname, dirs, files) in os.walk(crsPath):

for filename in files:

if filename.endswith('.jpg') or filename.endswith('.png'):

imgs.append(filename)

else:

xmls.append(filename)

#counting range for cycles

countForTrain = int(len(imgs)\*train\_ratio)

countForVal = int(len(imgs)\*val\_ratio)

print("training images are : ",countForTrain)

print("Validation images are : ",countForVal)

trainimagePath = '/content/yolov5/dataset/images/train'

trainlabelPath = '/content/yolov5/dataset/labels/train'

valimagePath = '/content/yolov5/dataset/images/val'

vallabelPath = '/content/yolov5/dataset/labels/val'

#cycle for train dir

for x in range(countForTrain):

fileJpg = choice(imgs) # get name of random image from origin dir

fileXml = fileJpg[:-4] +'.txt' # get name of corresponding annotation file

#move both files into train dir

#shutil.move(os.path.join(crsPath, fileJpg), os.path.join(trainimagePath, fileJpg))

#shutil.move(os.path.join(crsPath, fileXml), os.path.join(trainlabelPath, fileXml))

shutil.copy(os.path.join(crsPath, fileJpg), os.path.join(trainimagePath, fileJpg))

shutil.copy(os.path.join(crsPath, fileXml), os.path.join(trainlabelPath, fileXml))

#remove files from arrays

imgs.remove(fileJpg)

xmls.remove(fileXml)

#cycle for test dir

for x in range(countForVal):

fileJpg = choice(imgs) # get name of random image from origin dir

fileXml = fileJpg[:-4] +'.txt' # get name of corresponding annotation file

#move both files into train dir

#shutil.move(os.path.join(crsPath, fileJpg), os.path.join(valimagePath, fileJpg))

#shutil.move(os.path.join(crsPath, fileXml), os.path.join(vallabelPath, fileXml))

shutil.copy(os.path.join(crsPath, fileJpg), os.path.join(valimagePath, fileJpg))

shutil.copy(os.path.join(crsPath, fileXml), os.path.join(vallabelPath, fileXml))

#remove files from arrays

imgs.remove(fileJpg)

xmls.remove(fileXml)

#rest of files will be validation files, so rename origin dir to val dir

#os.rename(crsPath, valPath)

# shutil.move(crsPath, valPath)

## Code for Faster RCNN

!python -m pip install pyyaml==5.1

import sys, os, distutils.core

!python -m pip install 'git+https://github.com/facebookresearch/detectron2.git'

import torch, detectron2

!nvcc --version

TORCH\_VERSION = ".".join(torch.\_\_version\_\_.split(".")[:2])

CUDA\_VERSION = torch.\_\_version\_\_.split("+")[-1]

print("torch: ", TORCH\_VERSION, "; cuda: ", CUDA\_VERSION)

print("detectron2:", detectron2.\_\_version\_\_)

# Some basic setup:

# Setup detectron2 logger

import detectron2

from detectron2.utils.logger import setup\_logger

setup\_logger()

# import some common libraries

import numpy as np

import os, json, cv2, random

from google.colab.patches import cv2\_imshow

# import some common detectron2 utilities

from detectron2 import model\_zoo

from detectron2.engine import DefaultPredictor

from detectron2.config import get\_cfg

from detectron2.utils.visualizer import Visualizer

from detectron2.data import MetadataCatalog, DatasetCatalog

from detectron2.structures import BoxMode

def get\_dict(path):

files =os.listdir(path)

idx=-1

dataset\_dict=[]

for file in files:

if file[-4:]==".jpg" or file[-4:]==".png":

idx +=1

record={}

filename=os.path.join(path,file)

height,width=cv2.imread(filename).shape[:2]

record["file\_name"]=filename

record["image\_id"]=idx

record["height"]=height

record["width"]=width

annotations=open(path+"/"+ file[:-4]+".txt","r")

objs=[]

for annotation in annotations.readlines():

data=annotation.split(" ")

x\_min=np.float(data[1])-(np.float(data[3]))/2

y\_min=np.float(data[2])-(np.float(data[4]))/2

w\_1=np.float(data[3])

h\_1=np.float(data[4])

obj={"bbox":[x\_min\*width,y\_min\*height,w\_1\*width,h\_1\*height],

"bbox\_mode":BoxMode.XYWH\_ABS,

"category\_id":int(data[0])}

objs.append(obj)

record["annotations"]=objs

dataset\_dict.append(record)

return dataset\_dict

# registering train data

d=0

DatasetCatalog.register("/content/dataset/train", lambda d=d: get\_dict("/content/dataset/train"))

MetadataCatalog.get("/content/dataset/train").set(thing\_classes=[str(i) for i in range(0,10)])

digits\_metadata = MetadataCatalog.get("/content/dataset/train")

for d in random.sample(k, 3):

img = cv2.imread(d["file\_name"])

visualizer = Visualizer(img[:, :, ::-1], metadata=digits\_metadata, scale=0.5)

out = visualizer.draw\_dataset\_dict(d)

cv2\_imshow(out.get\_image()[:, :, ::-1]

from detectron2.engine import DefaultTrainer

cfg = get\_cfg()

cfg.merge\_from\_file(model\_zoo.get\_config\_file("COCO-Detection/faster\_rcnn\_R\_50\_FPN\_3x.yaml"))

cfg.DATASETS.TRAIN = ("/content/dataset/train",)

cfg.DATASETS.TEST = ()

cfg.DATALOADER.NUM\_WORKERS = 2

cfg.MODEL.WEIGHTS = model\_zoo.get\_checkpoint\_url("COCO-Detection/faster\_rcnn\_R\_50\_FPN\_3x.yaml") # Let training initialize from model zoo

cfg.SOLVER.IMS\_PER\_BATCH = 8 # This is the real "batch size" commonly known to deep learning people

cfg.SOLVER.BASE\_LR = 0.00025 # pick a good LR

cfg.SOLVER.MAX\_ITER = 300 # 300 iterations seems good enough for this toy dataset; you will need to train longer for a practical dataset

cfg.SOLVER.STEPS = [] # do not decay learning rate

cfg.MODEL.ROI\_HEADS.BATCH\_SIZE\_PER\_IMAGE = 128 # The "RoIHead batch size". 128 is faster, and good enough for this toy dataset (default: 512)

cfg.MODEL.ROI\_HEADS.NUM\_CLASSES = 10 # only has one class (ballon). (see https://detectron2.readthedocs.io/tutorials/datasets.html#update-the-config-for-new-datasets)

# NOTE: this config means the number of classes, but a few popular unofficial tutorials incorrect uses num\_classes+1 here.

os.makedirs(cfg.OUTPUT\_DIR, exist\_ok=True)

trainer = DefaultTrainer(cfg)

trainer.resume\_or\_load(resume=False)

trainer.train()

# Inference should use the config with parameters that are used in training

# cfg now already contains everything we've set previously. We changed it a little bit for inference:

cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT\_DIR, "model\_final.pth") # path to the model we just trained

cfg.MODEL.ROI\_HEADS.SCORE\_THRESH\_TEST = 0.30 # set a custom testing threshold

predictor = DefaultPredictor(cfg)

# registering test data

d=0

DatasetCatalog.register("/content/dataset/val", lambda d=d: get\_dict("/content/dataset/val"))

MetadataCatalog.get("/content/dataset/val").set(thing\_classes=[str(i) for i in range(0,10)])

digits\_metadat= MetadataCatalog.get("/content/dataset/val")

from detectron2.evaluation import COCOEvaluator, inference\_on\_dataset

from detectron2.data import build\_detection\_test\_loader

evaluator = COCOEvaluator("/content/dataset/val", output\_dir="./output")

val\_loader = build\_detection\_test\_loader(cfg, "/content/dataset/val")

results\_dic = inference\_on\_dataset(predictor.model, val\_loader, evaluator)

# another equivalent way to evaluate the model is to use `trainer.test`