Hand Written Data Digitization using an Anchor based Multi-Channel CNN (MCCNN) trained on a Hybrid Dataset (*DST*)

FINAL REPORT

Abhinandan Chiney
Ishan Kohli
Manish Kadam
Santosh N Kulkarni
Snehlata Jain
Ishwar Sukheja
Rohit Agarwal

Mentor: Malarvizhi Subramaniyan

Table of Contents

Abstract

Purpose: To develop a holistic system for handwritten English character recognition for manually filled forms by systematically synthesizing a robust hand written textual character dataset for acceptable representation of handwriting.

Materials and Methods: As part of this study, 572 copies of a form were filled by over 200 different individuals to introduce real life variation. These forms were then scanned and each hand written character in the forms was labelled and extracted using standard image processing techniques. The dataset of 84,712 character images created by this method (HW-dataset) comprised of both alphabetical and numerical characters. Three hybrid datasets (*DST*) were then formed by combining EMNIST datasets and the HW-dataset based on Digits (*DGTDST* – 329,668 character images), Alphabets (*LETDST* – 163,085 character images) and mixture of Digits and Alphabets (*MIXDST* – 189,586 character images). An anchor based image extraction technique was used in conjunction with a Multi-Channel CNN (MCCNN) model, trained on the three versions of *DST*, to automate the process of digitization of hand written forms. Three other methods including using Single Shot Detection and Tesseract-OCR have been evaluated with poor results.

Results: The classification accuracies of the MCCNN for *LETDST*, *DGTDST* and *MIXDST* are 93%, 96% and 93% respectively for unseen test data. Models trained on only the EMNIST dataset perform poorly when tested on unseen test data from *DST* datasets. An anchor based object detection used in conjunction with MCCNN trained on *DST* produces excellent results in digitizing entire unseen hand filled forms.

Conclusion: Touch free solutions will gain prevalence due to the emergence of threat of fomites in the world. In such a space, manual handling of forms for the purpose of data entry, digitization and information handling will be considered as potential health and safety hazards. The absence of good handwritten datasets for training Deep Learning models has prevented significant progress in this area. The solution presented in the current work uses a combination of models which is trained on a hybrid hand written data set with high variability. The model developed as part of this study is well suited for enabling touch free handling of documents.

Introduction

Handwriting recognition is a popular problem often encountered in modern day Machine Learning and Computer Vision. The unreliability and laborious nature of manual digitization techniques have made these problems very appealing for researchers in the above mentioned fields. Researchers and investigators have looked at various methods and techniques for resolving this issue. However, it is still unresolved to the satisfaction of the community (Shruthi & Patel, 2015). Although printed documents have been digitized, handwritten documents pose special problems. Recognizing handwritten documents like forms come with unexpected challenges like handwriting type, legibility, and some inherent noise like printed bounding boxes on form documents. These make the handwriting recognition problems more complex (Amir & Jindal, 2014). Common variations include characters having very similar shapes, peculiar distortions which are characteristics of specific handwritings and thickness variations among written characters due to use of different writing material and instruments among many others. Even use of different scanners alter the resolution of the images that are being used for training models (Sharma, Patnaik, & Kumar, 2013).

In the present study, it is aimed to develop a system that will digitize hand filled forms and save them as digital documents for further processing as per business needs. For this purpose, a motor vehicle insurance claim form (available at https://www.reliancegeneral.co.in/Downloads/Motor_Claim_Form.pdf) is chosen. Motor vehicle insurance claim form, due to its high need for digitization using manual methods before further processing, is apt case study for this problem. The present work considers the Motor Vehicle claim form filled in Upper Case letters and extracts the values against each demographic field such as Name, Address, Date of birth among others and save to an excel sheet. It is proposed to use a custom Object Detection Algorithm in conjunction with an image processing CNN to achieve this goal. It is additionally proposed that as a part of this work, the efficacy of standard text recognition data sets like Extended MNIST (EMNIST) be tested in recognizing real handwritten data in comparison to a human labelled dataset.

Literature Review

Common object detection methods use modified classifiers to perform detection. To detect an object, a classifier for the given object is evaluated at various locations in the image. Methods like Deformable Parts Models (DPM) use a sliding window technique where the said classifier is run at pre-decided spaced locations over the image (Felzenszwalb, Girshick, McAllester, & Ramanan, 2010). More recent region based methods like R-CNN methods first learn and propose potential bounding boxes in a given image and then execute a classifier on these proposed bounding boxes (Girshick, Donahue, Darrerr, & Malik, 2014). These methods are slow and less flexible compared to state of the art methods. Redmon *et. al.* (Redmon, Divvala, Girshick, & Farhadi, 2015) proposed a unified method for object detection and classification. The authors dubbed their system as You Only Look Once (YOLO) as the system looks at an image only once to predict what objects are present and where they are. The underlying system uses simultaneous regression and classification algorithms to classify objects with their locations.

Convolutional Neural Network (CNN) is one of the most commonly used Deep Learning Architectures for image processing problems. Such methods work by extracting multiple features ranging from simple features like edges and curves to more complex features like textures, automatically (LeCunn & Bengio, 1995). CNN as a result is a very popular technique for recognition of handwritings. Some investigators (Chen, Wang, Fan, Sun , & Satoshi, 2015) reported an accuracy rate of 99.59% on MNIST dataset using a modified CNN.

(Palehai & Fanany, 2017) proposed a combination of CNN and Support Vector Machines (SVM) to tackle the problem of handwriting recognition on form documents. The authors used CNN to extract the features from hand written forms and then subsequently passed that information to SVM for classification of the features into alphabets and words. The authors reported an accuracy rate of 98.85% on numeral characters, 93.05% on uppercase characters, 86.21% on lowercase characters, and 91.37% on the merger of numeral and uppercase characters. This is reported to be an improvement over the existing stand-alone CNN-Artificial Neural Network based methods. Nasien *et.al*. (Nasien, Haron, & Yuhaniz, 2010) propose a Freeman Chain Code with SVM to remove features from a standard NIST dataset consisting of uppercase, lowercase, and merger of uppercase and lowercase. Hussain and Vanlalruata

proposed (Hussain & Vanlalruata, 2018) a hybrid approach to extract features. They used a combination of zoning and topological feature to achieve this goal. To evaluate the performance of their model, the authors carried out an experiment using four different types of Artificial Neural Network (ANN) architectures. A Back Propagation Neural Network (BPNN) was found to have the highest accuracy of about 98%. Al Islam and Khan. (Al Islam & Khan, 2019) developed a CNN based classification algorithm for recognising Bengali numerical and mathematical expressions. YOLOv3 was used by the authors for object detection. The model was very successful achieving an end accuracy of over 98% for numbers and mathematical symbols. This method has shown great promise for development of a fast and accurate hand writing identification system.

Other researchers have used Long Short Term Memory (LSTM) based methods to identify and isolate hand written text characters. Breuel *et. al.* (Breuel, Ul-Hasan, Al-Azawi, & Shafait, 2013) applied bidirectional LSTM networks to the problem of machine-printed Latin and Fraktur recognition. The results presented in this paper show that the combination of line normalization and 1D-LSTM yields excellent OCR results for both Latin/Antiqua OCR and Fraktur OCR. LSTM based methods are not popular for handwriting recognition due to their "black-box" nature.

One of the more interesting developments in the field of image and text classification is the introduction of Multi-Channel CNNs (MCCNN) (Kim, 2014). These CNNs use two or more convolutional channels to prepare different feature maps based on their learnings. The loss is then cumulatively taken into account during the Back Propagation Step (BPS) to modify weights of both the channels. This is especially useful when the production data is expected to have large amount of variations. These types of CNN have been used for textual data with great effect (Guo, Zhang, Liu, & Ma, 2019).

Data Collection

This section of the report elaborates the data collected and used for this study. In addition some of the methods used to make the data usable are also discussed here.

Data

As explained in the foregoing discussion, one of the objectives is to showcase the importance of the choice of dataset used for training and development of the model. As a part of this

study, a standard dataset (Extended MNIST) is tested for its efficacy in training a handwriting based character classification model in compassion with a hand labelled dataset. A combination of the above datasets is also tested. The details are of the three datasets is described in the next few sections of the report.

Extended-MNIST dataset

The EMNIST dataset is a set of handwritten character digits derived from the NIST Special Database 19 and converted to a 28x28 pixel image format (NIST, 2019). The parent NIST Special Database 19 contains NIST's entire corpus of model training materials for hand written document and character recognition. There are six different splits provided in the EMNIST dataset. A short summary of the dataset is provided below:

- EMNIST ByClass: 814,255 characters with 62 unbalanced classes.
- EMNIST ByMerge: 814,255 characters with 47 unbalanced classes.
- EMNIST Balanced: 131,600 characters with 47 balanced classes.
- EMNIST Letters: 145,600 characters with 26 balanced classes.
- EMNIST Digits: 280,000 characters with 10 balanced classes.
- EMNIST MNIST: 70,000 characters with 10 balanced classes.

For the present study, EMNIST Balanced dataset is used to train models which will be used for classes where mixture of alphabets and numbers are expected. This set contains 131,600 characters with 47 classes. 10 numerical, 26 upper case alphabets and 11 lower case alphabets make up the dataset. In this dataset, some of the lower case alphabets are merged with the upper case alphabets. This is done to address an interesting problem in the classification of handwritten digits, which is the similarity between certain uppercase and lowercase letters (Cohen, Afshar, Tapson, & van Schaik, 2017). The motivation to choose the balanced dataset over other more populous datasets is to prevent inherent bias in the dataset. An unbalanced dataset often leads to a biased and overfit model.

As this project aims on only numerical and upper case alphabets, the Balanced EMNIST dataset is truncated to contain only the upper case and numerical characters. This reduces the data set to 96000 characters distributed over 36 character classes. For purpose of training the developed CNN model, this data is divided in to a training and a testing dataset. The distribution of the classes in the training data set is shown in Figure 1.

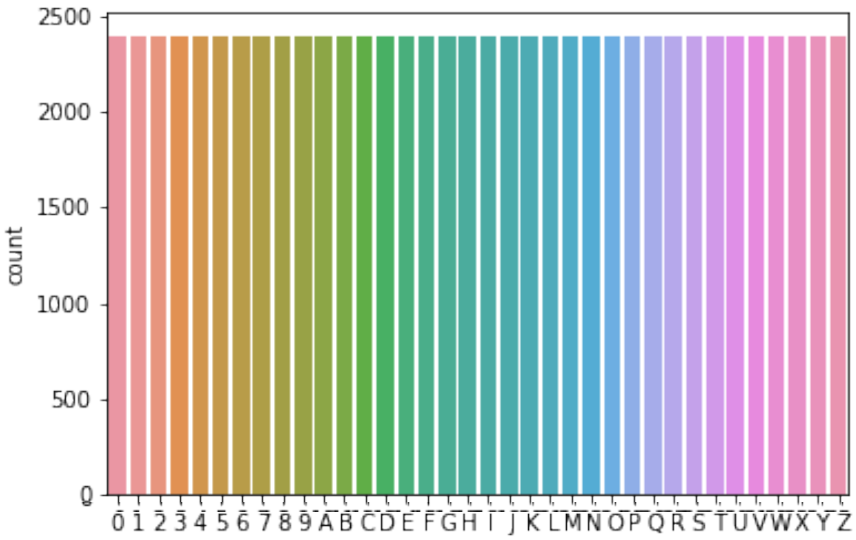


Figure 1: Distribution of classes in the training dataset made from EMNIST

A representation of a typical character in the EMNIST dataset is shown in Figure 2. The characters are in Binary colour mode with the back ground being solid black (pixel value: 0) and the characters described by arranging white pixels (pixel value: 255) accordingly. The characters are rescaled to 28 by 28 pixels to maintain consistency with the established MNIST dataset (Cohen, Afshar, Tapson, & van Schaik, 2017).

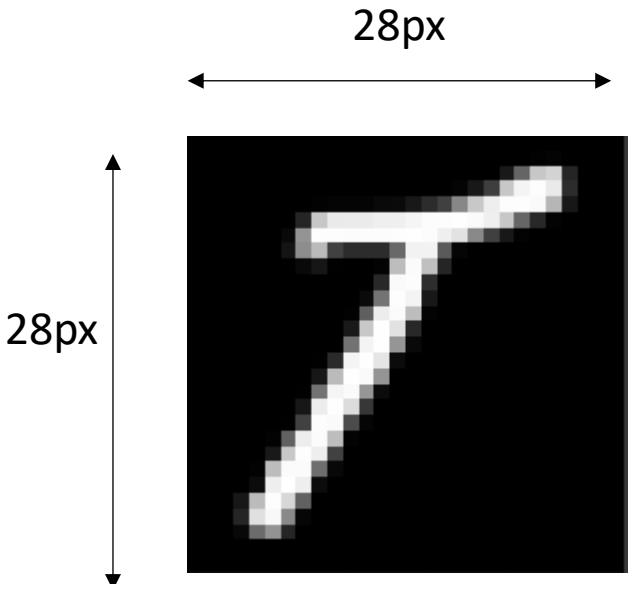


Figure 2: Character from EMNIST dataset showing its 28 by 28 pixel scaled size(Character label: T)

For numerical entries on the form, a separate instance of the previously developed model is used. EMNIST digits dataset contributes to training this model. This dataset like the Balanced dataset consists of equal number of data for each of the 10 classes – 28000.

For recognising characters in an alphabet only field, the developed model is trained on EMNIST letters dataset. This dataset is also balanced with 26 classes. There are 3984 data items per class.

To each of the above datasets, a new class depicting "BLANKS" have been added to enable the model to decipher BLANK spaces on the form. This is done to add to the readability of the output after detection and classification.

Hand Labelled Dataset from filled forms (HW-datasset)

For the purpose of creating this dataset, 617 forms have been filled. Care has been taken to distribute this form among many different sections of society to get a good variation in the handwritings. This is done so as to give the model an opportunity to learn as many variations in handwriting as possible. This will ensure the model has lower chance of over fitting with actual production data. The forms are scanned after they are filled and collected. Scanning is done with the following parameters to ensure uniformity.

- Format: .TIFF (Lossless)
- DPI: 400 dpi
- Resolution: 3312 by 4677 pixels
- Grey Scale
- Darkened mode
- Auto detection of paper to prevent shifting during individual scans

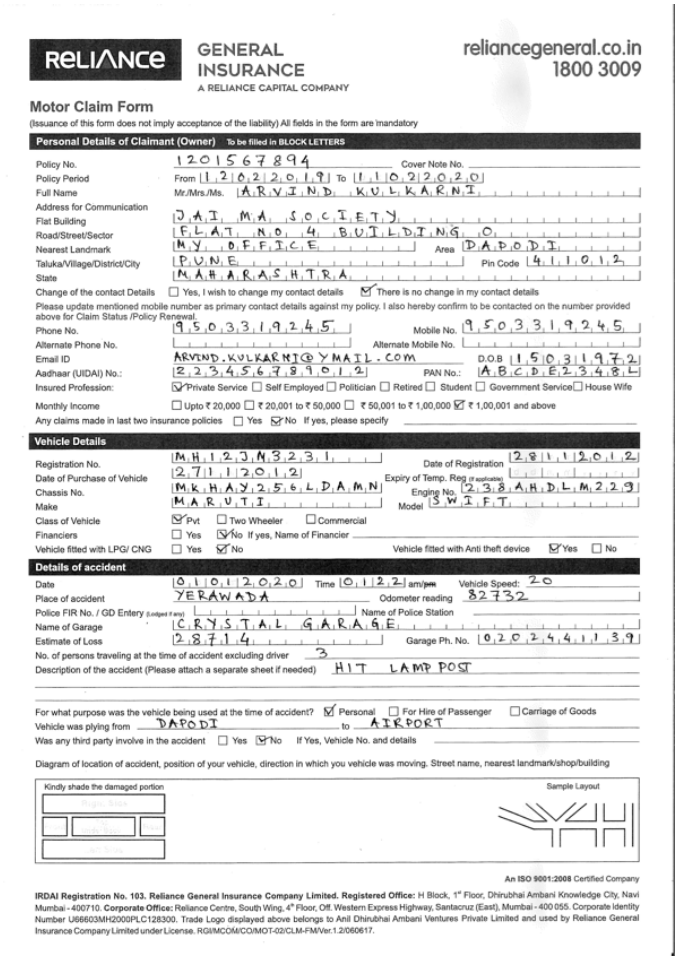
A typical scanned filled form is shown in Figure 3.

As a first step to cleaning up and using the filled forms as data, they are cleaned up to prevent excessive noise in the data. As a result 45 forms were discarded and they did not meet the standards of clarity needed to produce a deep learning training dataset. 572 forms were used to proceed to the next step of processing.

These hand filled formed were then meticulously labelled using an open source graphical image annotation tool, Labelimg (Tzutalin, 2015). A sample labelled form is shown in Figure 4. Care has been taken while labelling so as to avoid noise like printed bounding boxes and

alphabet orientation. In some cases these are deliberately inserter to get artificially create some variations in the training data set. A model training over a difficult dataset with higher number of variation is likely to perform better to production data as opposed to a model trained on simple and straight forward data.

LabelImg generates labels and their coordinates as XML files. A custom python script is developed (Appendix I) to consume the generated XML dataset into a more friendly Comma Separated Values (CSV) format. A typical CSV file (and the parent XML file) contains the following attributes of the image:



RELIANCE GENERAL INSURANCE
A RELIANCE CAPITAL COMPANY
reliancegeneral.co.in
1800 3009

Motor Claim Form
(Issuance of this form does not imply acceptance of the liability) All fields in the form are mandatory

Personal Details of Claimant (Owner) To be filled in BLOCK LETTERS

Policy No. 120156789 4 Cover Note No.
Policy Period From 12/02/2019 To 11/02/2020
Full Name Mr./Mrs./Ms. ARVIND KULKARNI
Address for Communication
Flat Building JAI MA SOCIETY
Road/Street/Sector FLAT NO 4 BUILDING O
Nearest Landmark MY OFFICE Area DADODI
Taluka/Village/District/City PUNE Pin Code 411012
State MAHARASHTRA
Change of the contact Details ☐ Yes, I wish to change my contact details ☑ There is no change in my contact details
Please update mentioned mobile number as primary contact details against my policy. I also hereby confirm to be contacted on the number provided above for Claim Status /Policy Renewal.
Phone No. 9503191245 Mobile No. 9503191245
Alternate Phone No. Alternate Mobile No.
Email ID ARVIND.KULKARNI@YMAIL.COM D.O.B 15/03/1972
Aadhaar (UIDAI) No.: 223456789012 PAN No.: ABCDE2348L
Insured Profession: ☑ Private Service ☐ Self Employed ☐ Politician ☐ Retired ☐ Student ☐ Government Service ☐ House Wife
Monthly Income ☐ Upto ₹ 20,000 ☐ ₹ 20,001 to ₹ 50,000 ☐ ₹ 50,001 to ₹ 1,00,000 ☑ ₹ 1,00,001 and above
Any claims made in last two insurance policies ☐ Yes ☑ No If yes, please specify

Vehicle Details

Registration No. MH12JN3231 Date of Registration 28/11/2012
Date of Purchase of Vehicle 27/11/2012 Expiry of Temp. Reg (if applicable)
Chassis No. MKHAY256LDAMN Engine No. 238AHDLM2293
Make MARUTI Model SWIFT
Class of Vehicle ☑ Pvt ☐ Two Wheeler ☐ Commercial
Financiers ☐ Yes ☑ No If yes, Name of Financier
Vehicle fitted with LPG/ CNG ☐ Yes ☑ No Vehicle fitted with Anti theft device ☑ Yes ☐ No

Details of accident

Date 01/10/2020 Time 01/22 am/pm Vehicle Speed: 20
Place of accident YERAWADA Odometer reading 32732
Police FIR No. / GD Entery (Lodged if any) Name of Police Station
Name of Garage CRYSTAL GARAGE
Estimate of Loss 28714 Garage Ph. No. 02024411139
No. of persons traveling at the time of accident excluding driver 3
Description of the accident (Please attach a separate sheet if needed) HIT LAMP POST

For what purpose was the vehicle being used at the time of accident? ☑ Personal ☐ For Hire of Passenger ☐ Carriage of Goods
Vehicle was plying from DADODI to AIRPORT
Was any third party involve in the accident ☐ Yes ☑ No If Yes, Vehicle No. and details

Diagram of location of accident, position of your vehicle, direction in which you vehicle was moving. Street name, nearest landmark/shop/building

Kindly shade the damaged portion
Sample Layout

An ISO 9001:2008 Certified Company

IRDAI Registration No. 103. Reliance General Insurance Company Limited. Registered Office: H Block, 1st Floor, Dhirubhai Ambani Knowledge City, Navi Mumbai - 400710. Corporate Office: Reliance Centre, South Wing, 4th Floor, Off. Western Express Highway, Santacruz (East), Mumbai - 400 055. Corporate Identity Number U66603MH2000PLC128300. Trade Logo displayed above belongs to Anil Dhirubhai Ambani Ventures Private Limited and used by Reliance General Insurance Company Limited under License. RGI/MCOM/CO/MOT-02/CLM-FM/Ver.1.2/060617.

Figure 3: Typical scanned filled form for the purpose of creating a hand filled and labelled dataset to complement EMNIST in training the model

- Path and File name
- Image Width
- Image Height
- Class of the Label

- Minimum X coordinate of the bounding box manually drawn around the class
- Minimum Y coordinate of the bounding box manually drawn around the class
- Maximum X coordinate of the bounding box manually drawn around the class
- Maximum Y coordinate of the bounding box manually drawn around the class

Using a custom OpenCV based python script, these coordinates are located on the forms and the character in each of the bounding box is extracted and resized to 28 by 28 pixel to be consistent with EMNIST dataset. One of the challenges that is faced while using these characters is that they have a white back ground. The characters are themselves made of

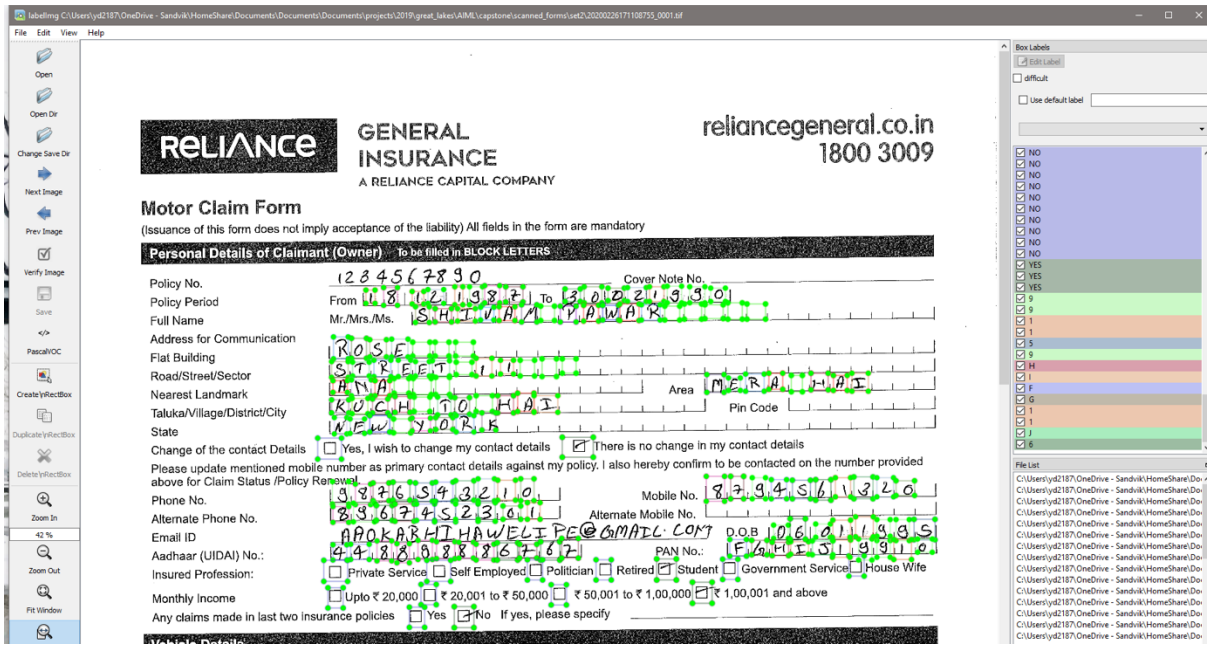


Figure 4: Sample filled and labelled form. Annotation tool LabelImg used for character wise annotation

black pixels (Figure 5). This is corrected before eventual consumption (Figure 6).

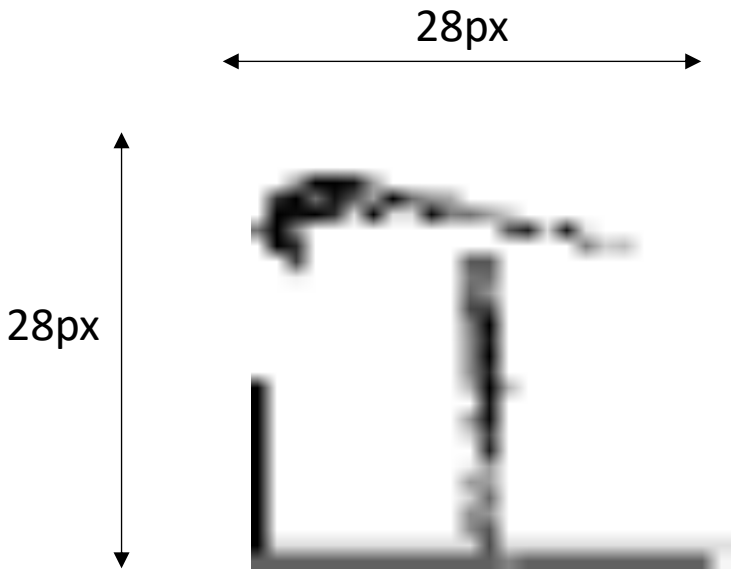


Figure 5: Raw image of character extracted from the form using Custom code (Character label: T)

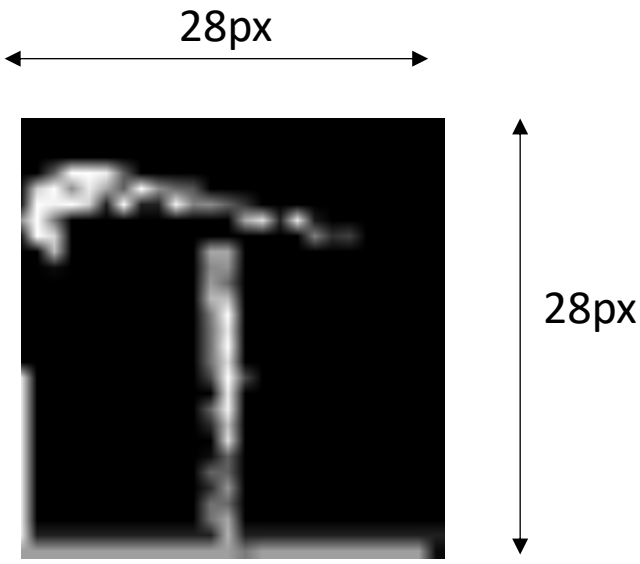


Figure 6: Final resized and background/foreground corrected image from the form (Character label: T)

Combination of EMNIST and Hand labelled dataset from filled forms

A combination of the EMNIST dataset created above with 96,000 characters with the enhanced dataset created by hand labelling of the forms is also formed. It is also noted that data labelled as BLANKs have also been included in the dataset. This is important as BLANK

entries form an integral part of the data entry. This increases the number of classes for this problem to 37 classes (10 numerical, 26 alphabets and 1 BLANK). The "Test – Labelled Forms (unseen data)" consists of data only from the labelled forms and does not contain any EMNIST data. This is done so as to replicate actual production data.

A summary of the dataset is given in Table 1.

Table 1: Summary of datasets used for training the models

	Train -EMNIST	Validation - EMNIST	Train – Labelled Forms	Test – Labelled Forms (unseen data)	Total training data	Data Set Code
For **mix** model with blanks (37 classes)	88437	16437	70427	14285	158864	MIXDST
For **letters** model with blanks (27 classes)	90837	22813	42013	7422	132850	LETDST
For **digits** model with blanks (11 classes)	242037	42037	39193	6401	281230	DGTDST

The class wise distribution is shown in Figure 7. The issue of class imbalance is resolved by the presence of EMNIST dataset as shown in Figure 8. The sequence of pre-processing of data is shown in Figure 9.

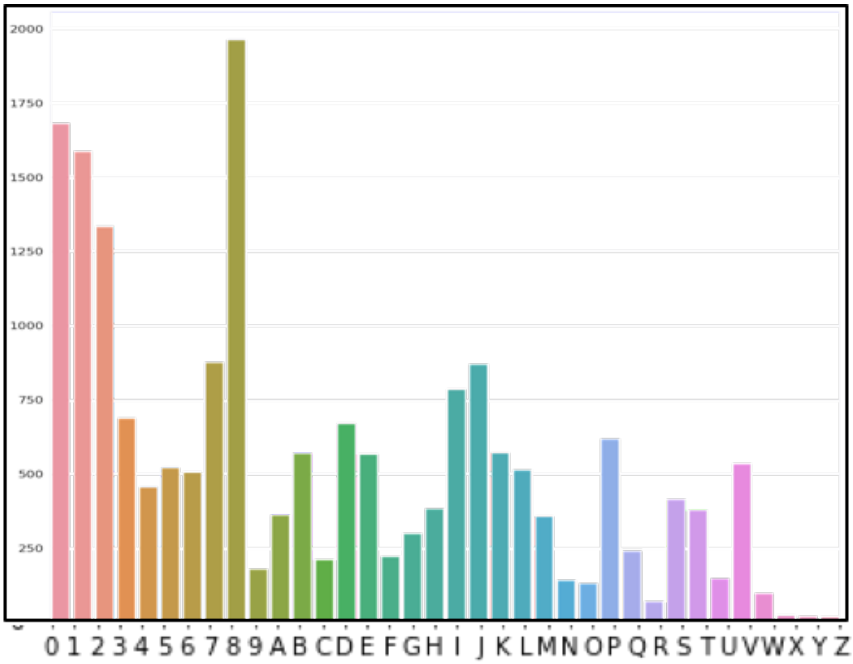


Figure 7: Class wise distribution of data for hand filled forms

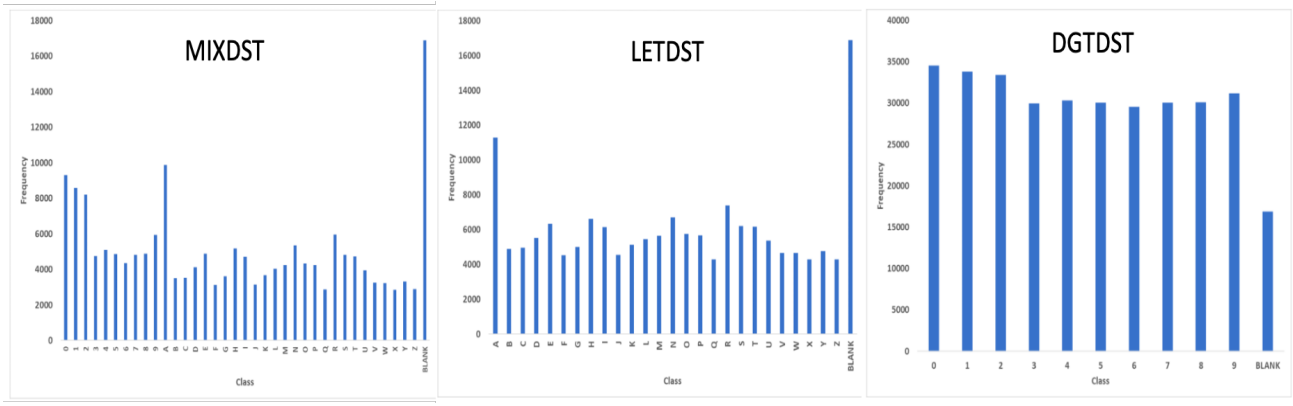


Figure 8: Class wise distribution of data for combination of EMNIST Balanced and hand filled forms

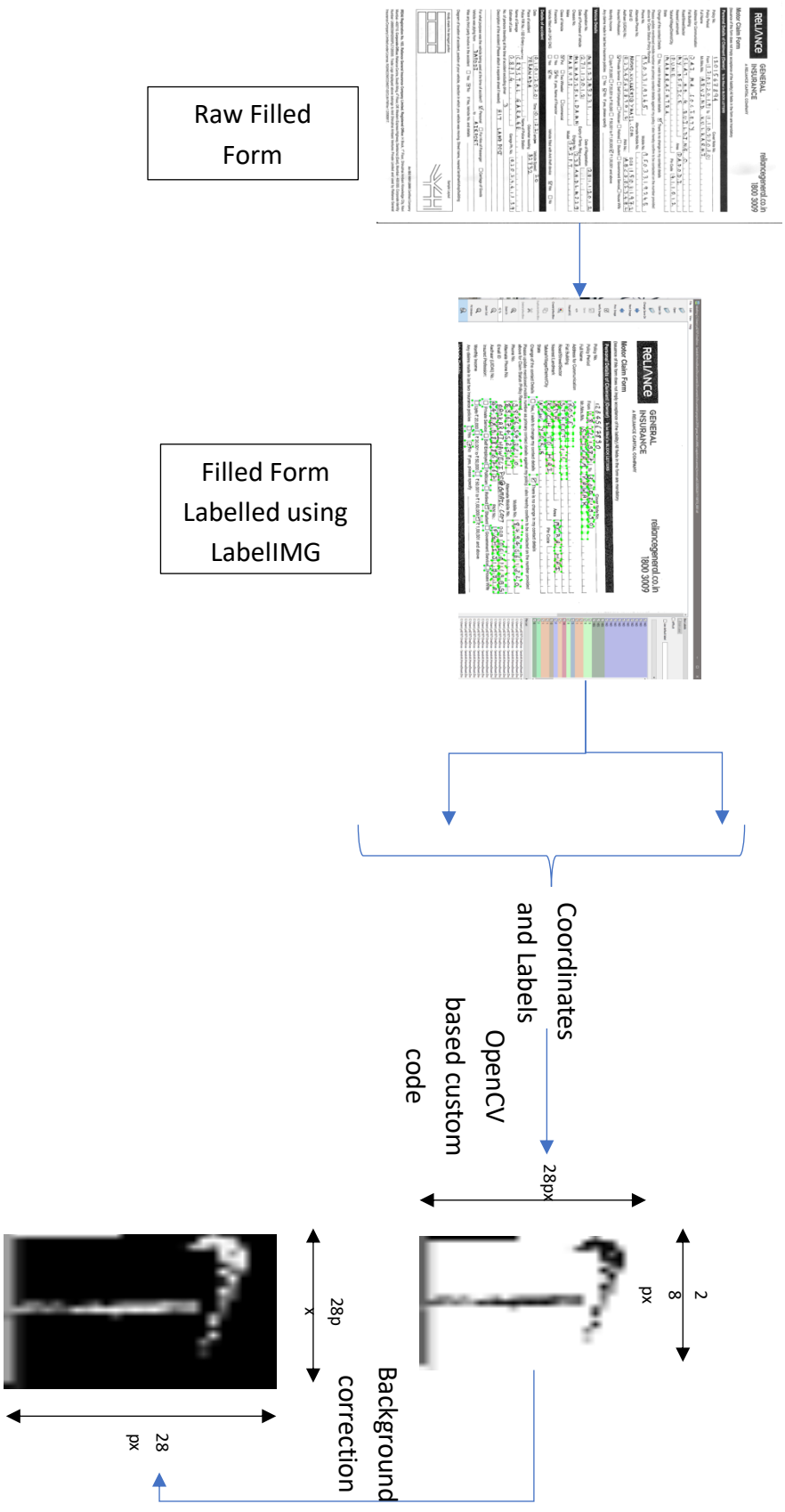


Figure 9: Work flow for pre-processing o hand labelled image from form

Fields level Labelling of Forms

Another set of labelled data set is created for Fields based Single Shot Detection Method (FSSDM). Here the entire field data is labelled for all the forms. Figure 10 shows a sample labelled form.

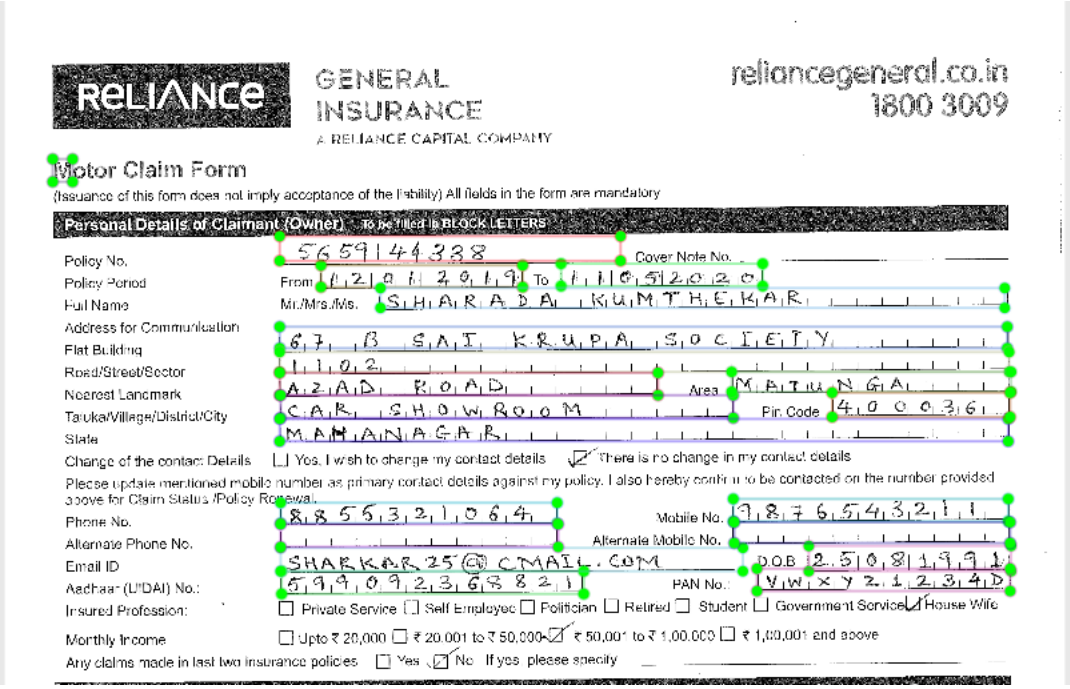


Figure 10: Field level labelling of forms. Note the "M" in the corner labelled for Object detection.

Model Development and Method

Based on the literature review conducted, it is proposed in the present work that a custom Object Detection Algorithm will be used in conjunction with a Deep CNN to identify and locate characters on the forms. A similar method has previously been used by (Al Islam & Khan, 2019) to identify Bengali handwritten characters with great success.

The CSV generated from the XML that is created by LabelImg, contains the labels and the local coordinates of the characters on the forms. The coordinates are sent to an Object detection algorithm which detects bounding boxes around potential regions of hand written text. The class label data along with the cropped images inside the bounding box is pre-processed as discussed in the previous Sections, is sent to Deep CNN based classification algorithm where the model learns to classify the object inside the bounding box as different alphabets.

The development of the object detection module as well as the text classification module are discussed separately in the upcoming sections. A dedicated discussion is also reserved for merging the two modules.

Development of the Deep CNN based classification model

LeCunn and Bengio (LeCunn & Bengio, 1995) first introduced the concept of Convolutional Neural Networks (CNN). These networks in their most common form are a combination of a series of layers performing a mathematical operation called convolution and dense fully connected layers that are reminiscent of Artificial Neural Networks (ANN). ANN due to their fully connected layers, are not able to process heavy information like pixel information of images efficiently. The convolutional layers allow the images to be scaled down to manageable proportions by use of convolutions. In the present work, two variations of CNNs have been developed and evaluated – Single-Channel Conventional CNN (SCCNN) and Multi-Channel CNN (MCCNN).

Single-Channel Conventional CNN (SCCNN)

Several network architectures have been tried and tested and the architecture adopted in the present work due to its robust performance across different datasets with respect to different kinds of datasets is shown in Figure 11. This network has a total of 827,510 trainable parameters and 290 non-trainable parameters.

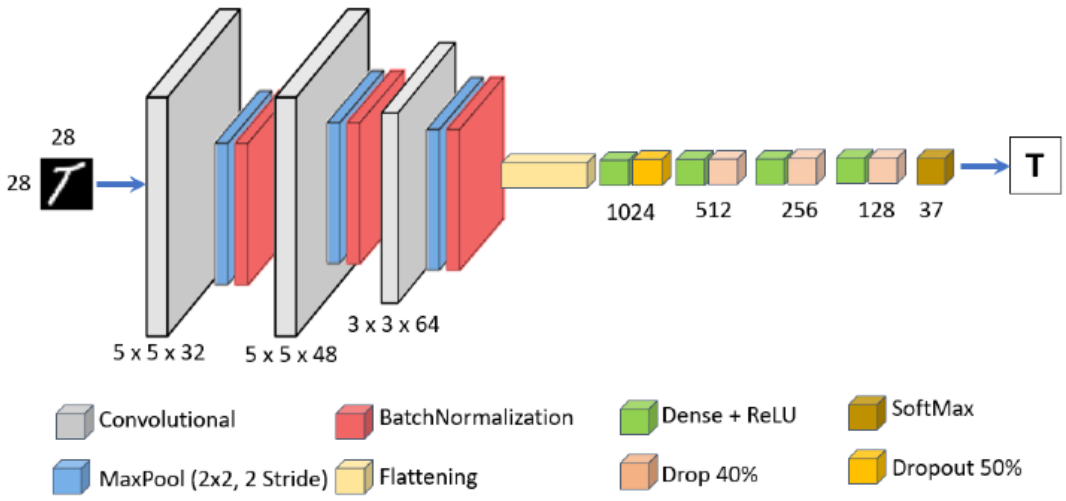


Figure 11: Architecture of the Convolutional Neural Network (CNN) followed by Dense Neural Network (DNN) used in the current work

The input to this network is normalized pixel values of the extracted character images. It is a standard architecture of CNN. The network contains three convolutional layers, each with their own 2D MaxPooling having filter shape (2,2) with stride length of two and Batch Normalization layers. The output of the convolutional layers is flattened for consumption of the subsequent fully connected layers. There are four fully connected layers in this network. The output layer has thirty six neurons to account for ten numerical characters (0-9) and twenty six upper case alphabets (A-Z). ADAM optimizer proposed by (Kingma & Ba, 2015) is used along with a categorical cross entropy loss function for weight modification as part of the Back Propagation process.

Results and Model Performance

As described in the earlier sections, the model has been tested using three different datasets. The creation of these datasets have been explained in a forgoing discussion. The performance using four different metrics is shown in Table 2.

Table 2: Model performance with three different datasets and over four different performance metric. Note the performance is reported only for the mix data for this model

Model trained on	**Train Accuracy**	**Validation Accuracy**	**Testing Accuracy**	**Testing F1 Score**	**Testing Recall Score**	**Testing Precision Score**
EMNIST	93%	92%	21%	0.29	0.21	0.48
Scanned and human labelled characters from forms	83%	81%	72%	0.70	0.68	0.72
EMNIST + Scanned and human labelled characters from forms **(MIXDST)**	90%	91%	82%	0.76	0.75	0.78

As explained in a foregoing discussion, the testing is performed on random unseen data selected from labelled forms. It is seen that the EMNIST dataset, while, is an extensive dataset providing a good standard for handwritten alphabets and numerical characters, it does not capture the variations of a real handwritten dataset created by several independent writers. The dataset formed only by the human labelled dataset does significantly better in predicting the character classes in the test dataset. From the training and validation accuracy, it is understood that the data is significantly more difficult as compared to the standard EMNIST dataset. However, this dataset also suffers from poor performance. This dataset also has a drawback in the fact that creating this database is a laborious and time consuming process. This called for a third database to be created and tested as a part of this exercise. A combination of data extracted from hand filled forms and EMNIST proves to be the best dataset for an exercise like this where it is important to capture the variations in real handwritings. This is due to the fact that EMNIST provides for a good opportunity to learn basic features while the hand filled dataset provides the inherent variations in each alphabet that a handwritten dataset should have. The performance metric show great improvement, however even this model suffers from over fitting. The class specific confusion matrix is shown in Figure 12.

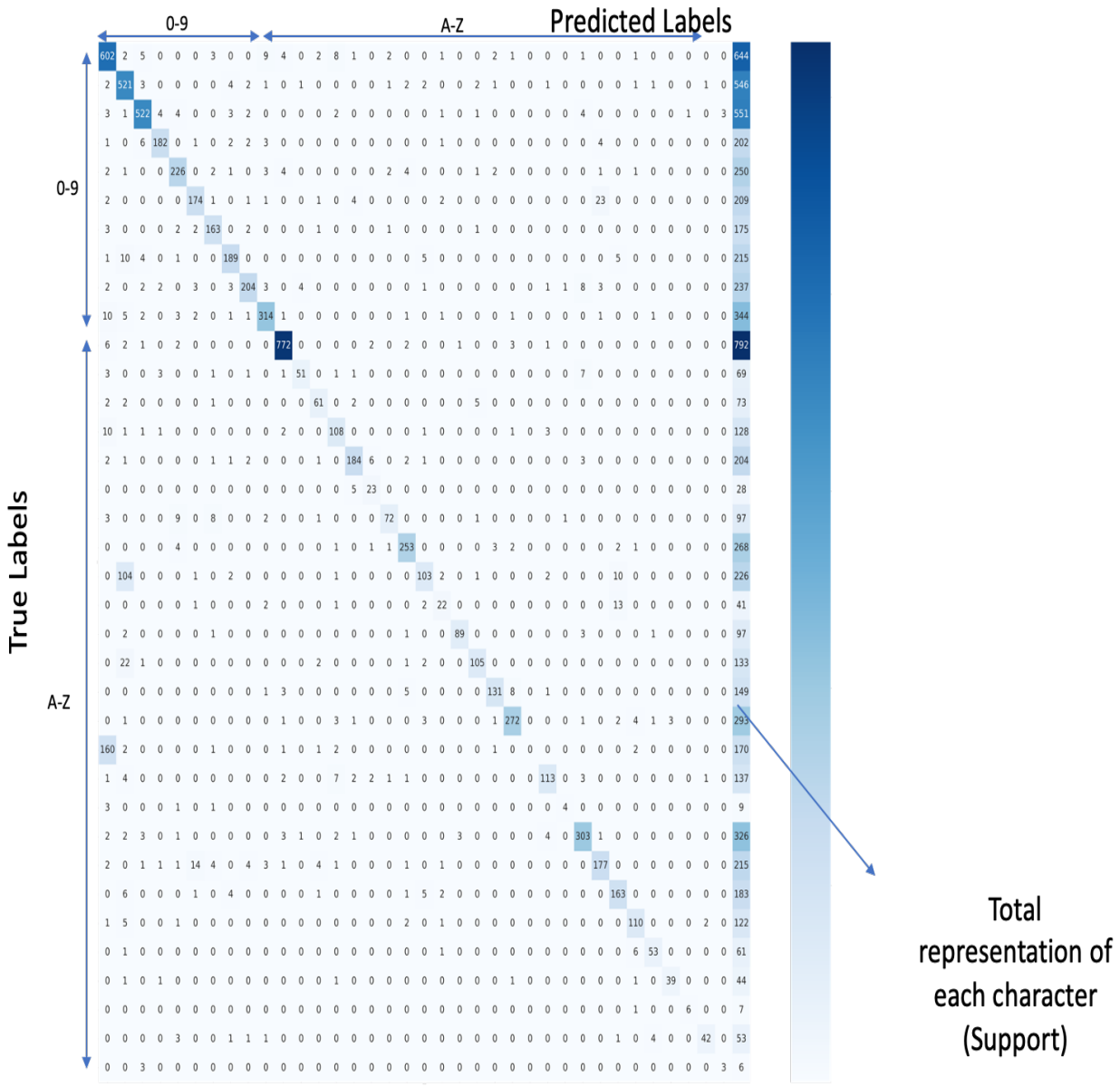


Figure 12 : Character wise confusion matrix generated by the SCCNN model trained on combination of EMNIST dataset and characters extracted from hand filled hand labelled forms on unseen test set

Some of the common mistakes that are committed by the model are listed in Table 3.

Table 3: Common character mistakes committed by model

True Label	Predicted Label
O	0
8	3
6	G
5	S
0	D
I	1, L, T

This exercise shows the importance of having the right dataset for the right purpose. Standard datasets like the NIST, MNIST and EMNIST while provide a good starting point but by themselves are not enough to capture the variations encountered in real production data. Another important observation here is that there is considerable misclassification with respect to numbers and alphabets. Hence, it has been decided that to enhance the overall accuracy of the model(s), three models be combined to predict the outcome. One model (trained on MIXDST) will handle mix data fields like address, PAN card details etc. A second model (trained on DGTDST) will handle numerical only data fields like Dates and Mobile Numbers. Alphabet only data fields like will be handled by a third model (trained on LETDST).

Since the MIXDST is the most complicated dataset, the models hence forth will be benchmarked on their performance on the MIXDST.

Multi-Channel CNN (MCCNN)

As seen from the foregoing discussion, the conventional SCCNN gives satisfactory results. The results however, do not boast of very high accuracy. At this point a different kind of CNN is investigated – Multi-Channel CNN. As discussed earlier, these models are preferred where the data variation in production is supposed to be very high. Such models are very good at learning features of Augmented datasets. Since handwritten data has very high variation in style and pattern, MCCNNs are expected to outperform SCCNNs for such data.

In the present work a MCCNN is developed with the architecture shown in Figure 13.

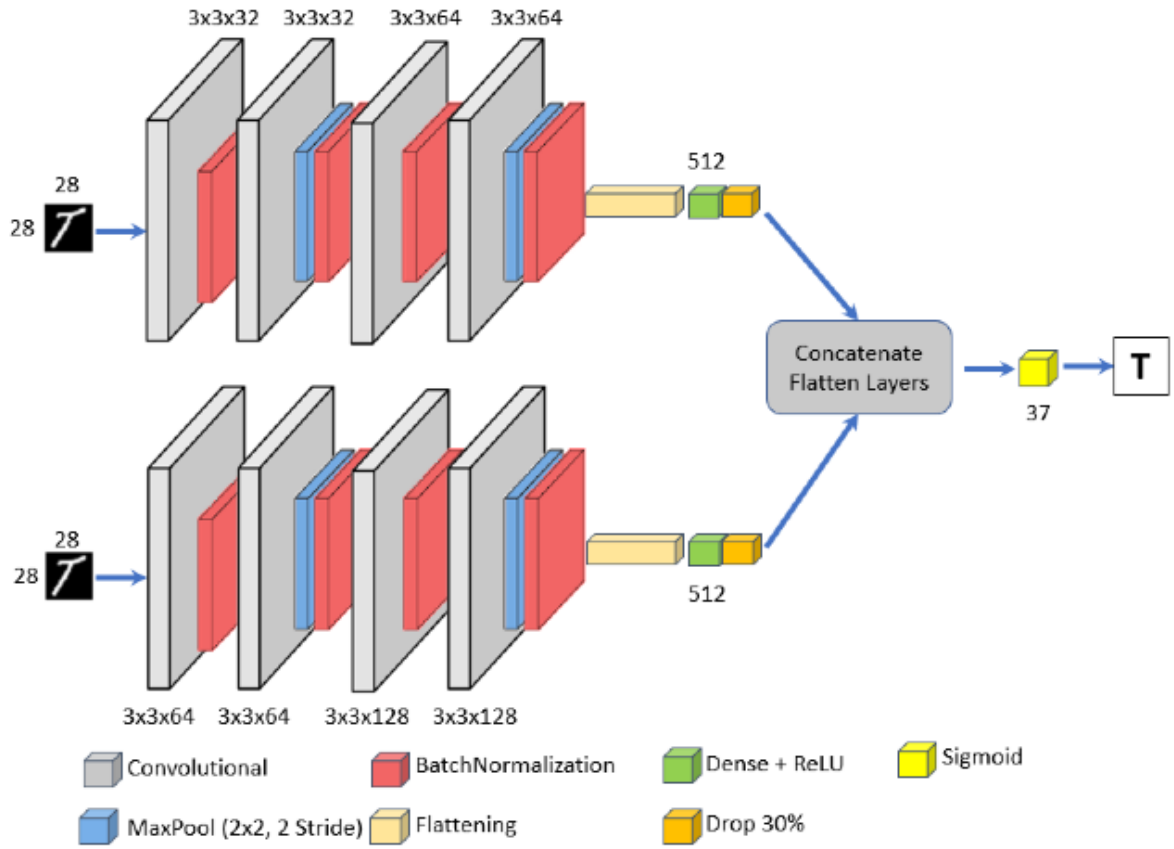


Figure 13: Architecture of the Multi-Channel Convolutional Neural Network used

This model has 1,939,013 trainable parameters and 3,200 non-trainable parameters. This model proved to be very robust and hence was used for all three purposes (alphabet only data, numerical only data and mix data).

The input to each of the channels of this network is normalized pixel values of the extracted character images. Each channel is like a standard architecture of CNN. The output of the flattening layer of each Convolutional channel is concatenated and then sent to 37 neuron output layer. ADAM optimizer proposed by (Kingma & Ba, 2015) is used along with a categorical cross entropy loss function for weight modification as part of the Back Propagation process.

This model is trained on 3 different datasets (Table 1), yielding three different sets of model weights. When the model is used to predict on the actual data, the appropriate set of model weights is called upon and used depending on whether the predicting class can be a mix of alphabets and numbers, only alphabets or only numbers.

Results and Model Performance

The model performance of the model is judged based on the accuracy, Recall, Precision and F1-score when predicting on the MIXDST test dataset (14285 images). This test dataset contains data only from Labelled forms. This is done so as to mimic actual production data.

The details of the performance of the model trained on MIXDST dataset when tested on production like data is given in Table 4.

Table 4: Performance of the model trained on MIXDST when tested on production like data

Model trained on	**Train Accuracy**	**Validation Accuracy**	**Testing Accuracy**	**Testing F1 Score**	**Testing Recall Score**	**Testing Precision Score**
EMNIST + Scanned and human labelled characters from forms **(MIXDST)**	93%	93%	93%	0.90	0.89	0.92

Figure 14 shows a comparison accuracy of the MCCNN model with SCCNN model for the same production like dataset. Figure 15 shows performance with respect to other performance metrics. It is seen that there is significant improvement in the performance of the model.

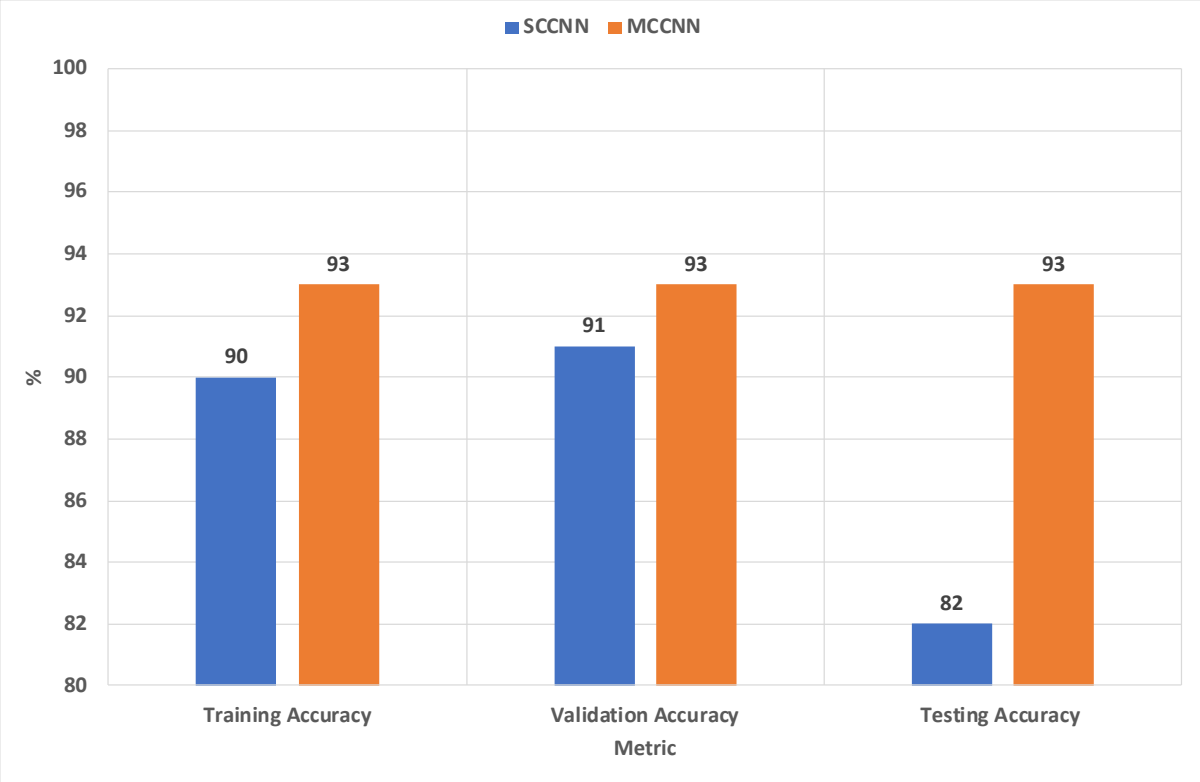


Figure14: Comparison of accuracy of the MCCNN and SCCNM models with production data

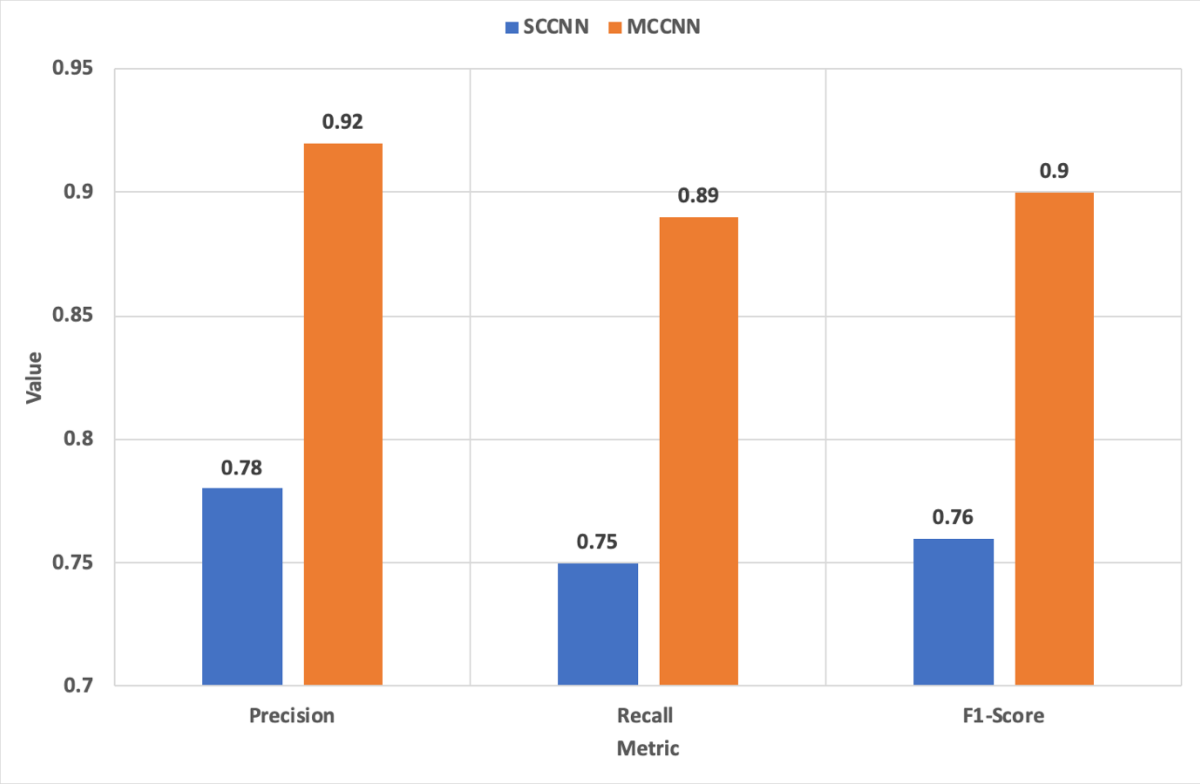


Figure15: Comparison of model performance metrics for SCCNN and MCCNN models on production data

The class specific confusion matrix is shown in Figure 16 for MCCNN. Based on the significant improvement seen on the MIXDST over SCCNN, this model architecture is further trained

DGTDST and LETDST datasets to generate corresponding weight files. The performance of the model on production like dataset for those models is shown in Table 5. It is seen that the MCCNN model performs exceedingly well for all kinds of dataset and hence is chosen for rest of the present work.

Table 5: Performance of MCCNN model on different kinds of datasets

	Training accuracy	**Validation accuracy**	**Testing accuracy (unseen production data)**
MCCNN - LETDST	96%	96%	93%
MCCNN - DGTDST	99%	99%	96%

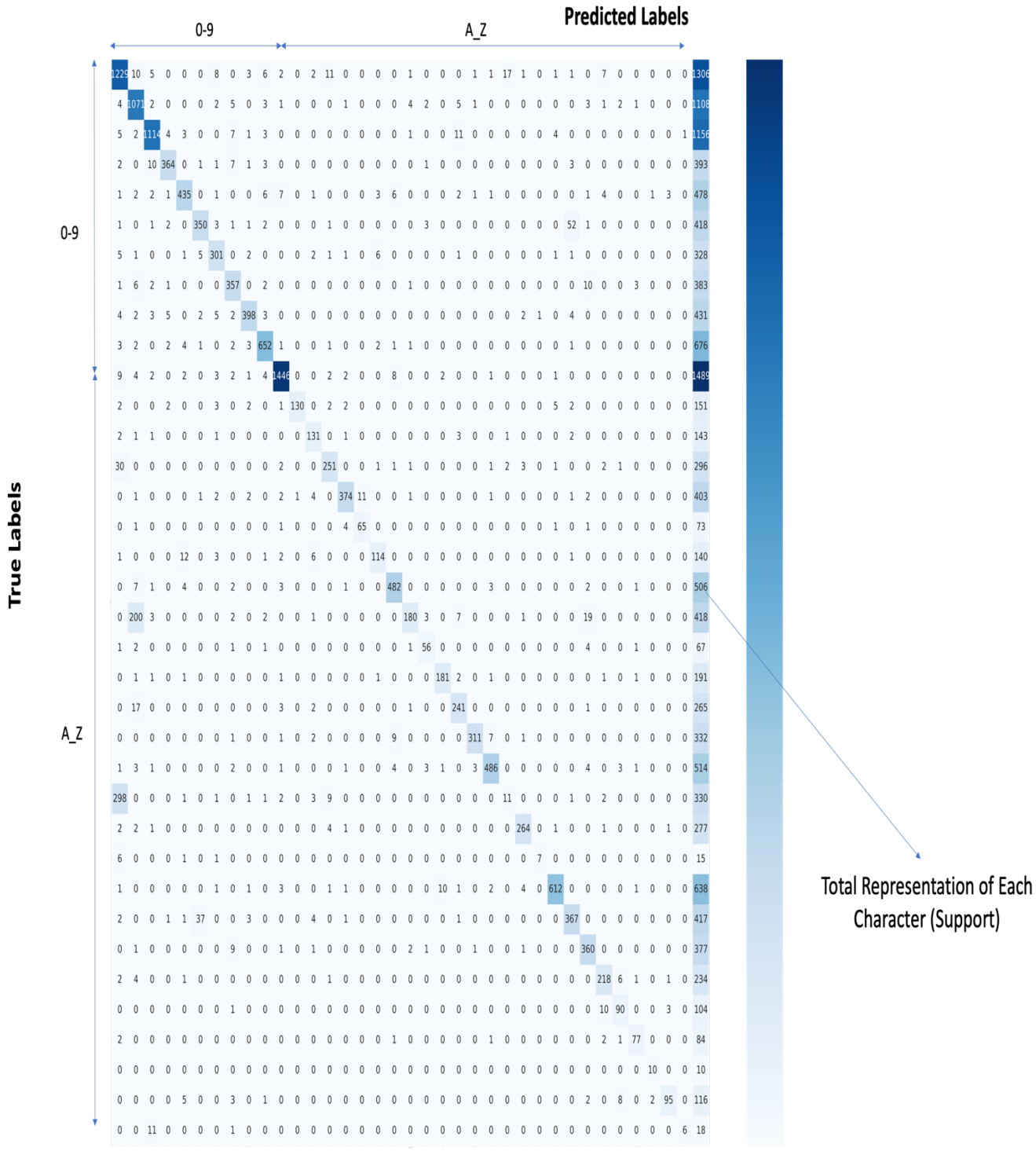


Figure 16: Character wise confusion matrix of the prediction by MCCNN trained on MIXDST and tested on unseen production like data

It is also seen that the models perform much better when split up as per data requirements. Based on this observation, it has been decided to use the three models trained on MIXDST, LETDST and DGTDST datasets in unison to improve overall accuracy of the eventual model.

Development of the Object Detection Algorithm for Handwriting Detection

As is evident from the problem of digitising a hand filled form, it is necessary that the region of hand written text be identified before classification. For this purpose, four different methods have been identified and evaluated. They are as follows:

1. Anchor based method with OpenCV Template Matching backend (AM),
2. Fields based Single Shot Detection Method (FSSDM),
3. Character based Single Shot Detection Method (CSSDM), and
4. Tesseract with Python wrapper (PyT).

The upcoming sections of the report will discuss each of these methods in details.

Anchor based method with OpenCV Template Matching backend (AM)

As the primary document to be digitised is a form, it is seen that there are regions demarcated for writing down the values of each field by hand. As this is a standard form, there are printed guidelines mentioning the class of information to be filed. These regions in the forms are printed, fixed and constant for the entire dataset and is expected to be constant for the production dataset as well. This provides an excellent opportunity to mark an anchor location on the form and use relative distance analysis to locate the start of the hand written area for a given class and the corresponding class. In this case the printed letter “M” is used as an anchor box. This concept is shown in Figure 17. This anchor is extracted using OpenCV’s template matching functionality. This method uses a sliding window technique to determine which area in a parent image matches best with a given patch of image (OpenCV Dev Team, n.d.).

After the coordinates of the anchor box “M” is extracted, the field level labelling described in Figure 10 for 572 forms is then used to mark the full region of the handwritten area for a given class. This is obtained by using the average value of width and height of each field over 572 forms. As the number of forms labelled in high, this average value is a good representative of the true value of width of each class. Some of these values are listed in Table 6. Using these values, entire rows of handwritten data is extracted. Figure 18 shows some of these rows.

Table 6: Typical distance of classes from anchor box

policyno	policyfrom
row1_xmin = M_xmin + 712 row1_ymin = M_ymin + 250 row1_xmax =M_xmax + 1726 row1_ymax = M_ymax + 270	row1_xmin = M_xmin + 841 row1_ymin = M_ymin + 330 row1_xmax = M_xmax + 1403 row1_ymax = M_ymax + 355
flat	road
row1_xmin = M_xmin + 706 row1_ymin = M_ymin + 520 row1_xmax = M_xmax + 2929 row1_ymax = M_ymax + 552	row1_xmin = M_xmin + 709 row1_ymin = M_ymin + 602 row1_xmax = M_xmax + 2935 row1_ymax = M_ymax + 625
city	state
row1_xmin = M_xmin + 709 row1_ymin = M_ymin + 740 row1_xmax = M_xmax + 2067 row1_ymax = M_ymax + 760	row1_xmin = M_xmin + 709 row1_ymin = M_ymin + 815 row1_xmax = M_xmax + 2926 row1_ymax = M_ymax + 840
pin	phoneno
row1_xmin = M_xmin + 2438 row1_ymin = M_ymin + 730 row1_xmax = M_xmax + 2926 row1_ymax = M_ymax + 758	row1_xmin = M_xmin + 715 row1_ymin = M_ymin + 1070 row1_xmax = M_xmax + 1509 row1_ymax = M_ymax + 1095
email	alternatephoneno
row1_xmin = M_xmin + 718 row1_ymin = M_ymin + 1212 row1_xmax = M_xmax + 2056 row1_ymax = M_ymax + 1242	row1_xmin = M_xmin + 709 row1_ymin = M_ymin + 1142 row1_xmax = M_xmax + 1511 row1_ymax = M_ymax + 1163
adhar	dob
row1_xmin = M_xmin + 712 row1_ymin = M_ymin + 1292 row1_xmax = M_xmax + 1591 row1_ymax = M_ymax + 1310	row1_xmin = M_xmin + 2360 row1_ymin = M_ymin + 1208 row1_xmax = M_xmax + 2920 row1_ymax = M_ymax + 1238

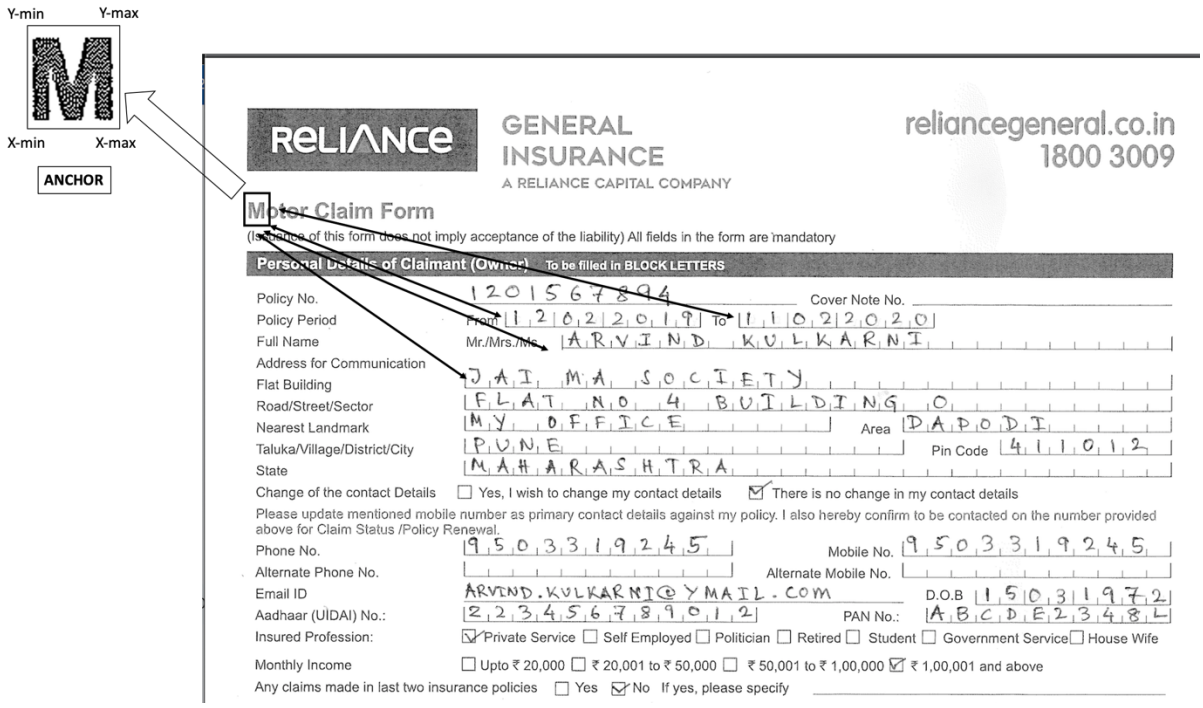


Figure 17: Concept of using an anchor to determine locations of handwritten regions and the corresponding classes

Class	Extracted Region
Policy From	12 02 2019
Policy To	11 02 2020
Name	ARVIND KULKARNI
Flat	JAI MA SOCIETY
Landmark	MY OFFICE
PAN	ABCDE2348L
Date of Birth	15 03 1972

Figure 18: Handwritten text regions extracted based on anchor coordinates

Once the hand written region is extracted for each class, each character in those regions was extracted to be fed to the appropriate MCCNN as discussed in the last section. This is achieved by analysing the character based labelling data extracted by LabelImg (Figure 4). It is seen that

that over 572 labelled form images and over 84,000 characters labelled, each of the handwritten character occupies an average of 78x78 pixels (Figure 19). This average value works for most of the characters. Starting from the left most character, other characters are extracted at intervals of 78 pixels as shown in Figure 20.

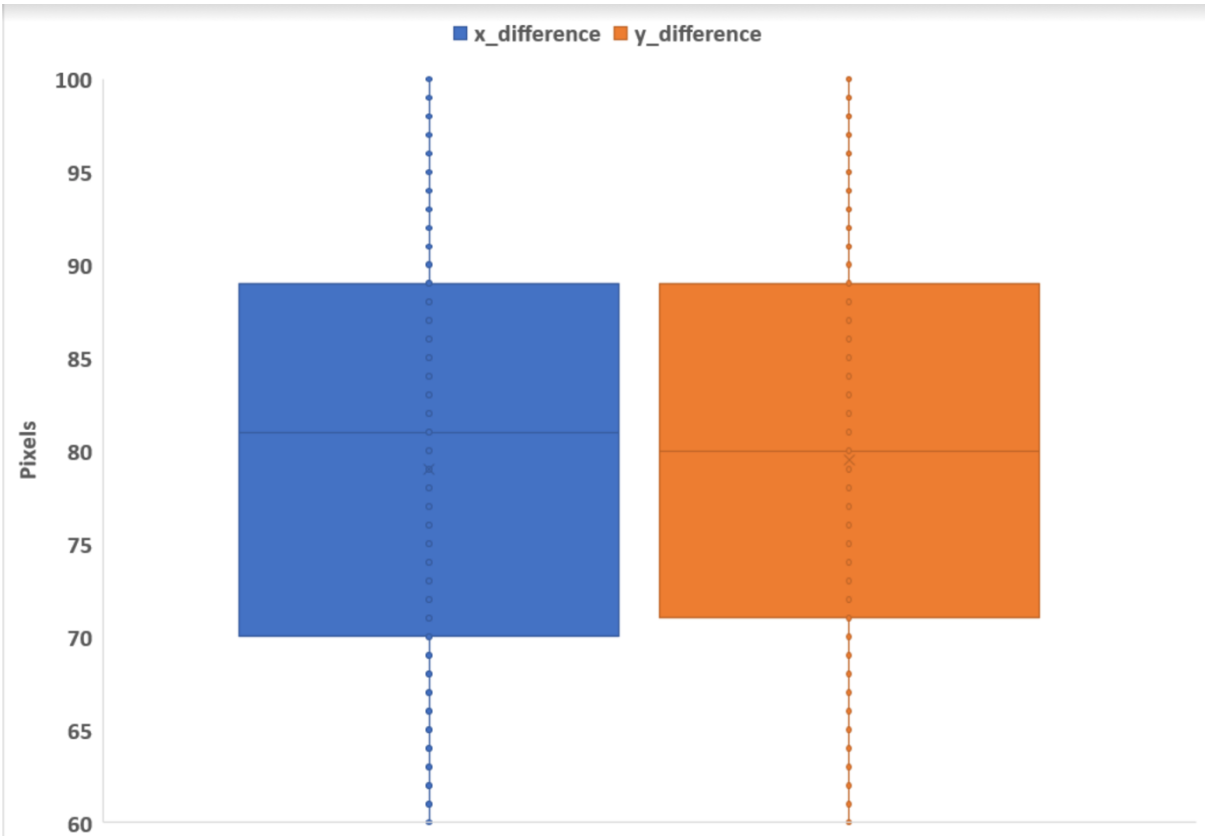


Figure 19: Average size of a character box in the forms (over 517 forms and 85000 characters)

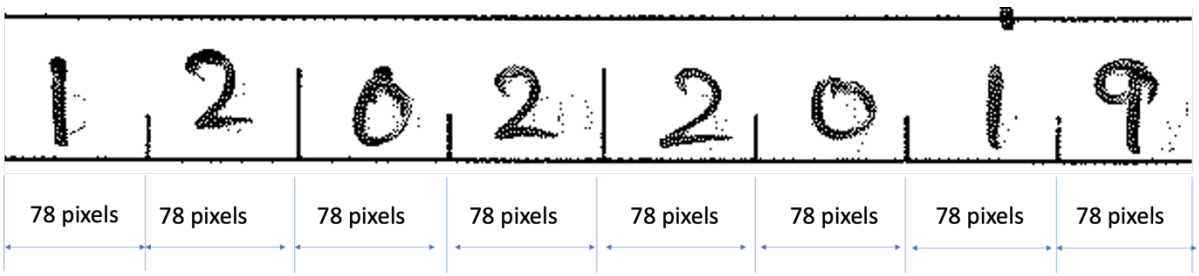


Figure 20: Character level extraction process from hand written region

The image string like the one shown in Figure 20 when extracted leads to the a series of individual character level images. Such a set is shown in Figure 21.

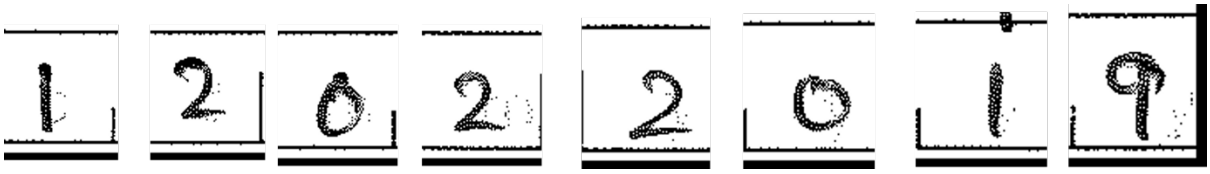


Figure 21:Extracted characters for image string in Figure 20

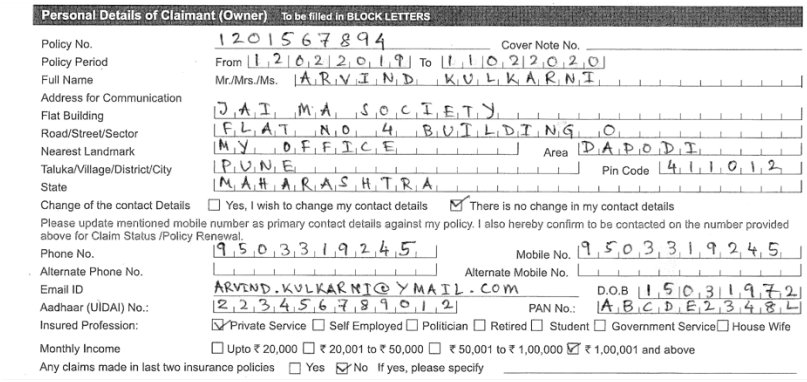
The character level images that are extracted are then forwarded to the MCCNN Model for further classification. The choice of model based on the field class is shown in Table 7.

Table 7: Model Selection for Character Classification based on Field Class (Green cells indicate the model used)

Field Class	LETDST	DGTDST	MIXDST
Policy to			
Policy from			
Name			
Landmark			
City			
State			
PIN			
PAN			
Mobile No.			
Alternate Mobile No.			
Flat			
Road			
Aadhar			
DOB			

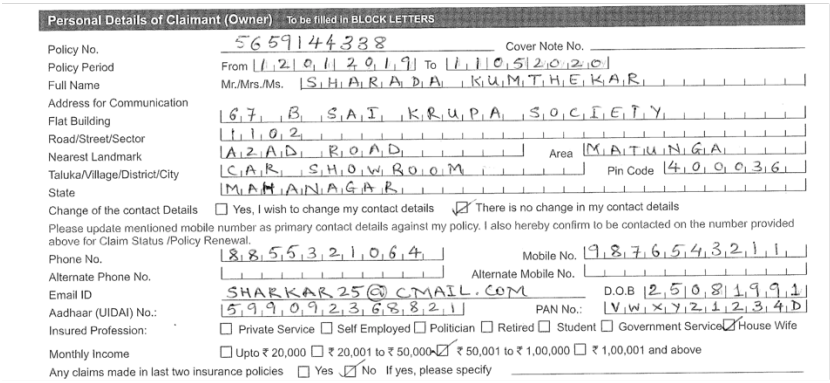
The classified output by the three models in collated in a separate CSV file.

Figures 22 shows the predicted output for a form from the training set. Figure 23 shows the predicted output for a form from the unseen production set.



policyfrom	12022019
policyto	11022020
name	ARVIND KULKARNI
flat	JAI MA SOCIETY
road	FLAT NO 4 BUILDIN3 0L
landmark	MY OEEICE L LLL
area	DAEODI
city	PUNE
state	MAHARASHTRA L
pin	411012
phoneno	9503319245
mobileno	9503319245
alternatephoneno	
alternatemobileno	
adhar	223456389012
dob	15021972
pan	ABCDE2348I

Figure 22: Predicted output for the form depicted in Figures 17, 18, 20 and 21



policyfrom	12012019
policyto	11052020
name	SHARADA KUMJHEKAR
flat	67 B SAI KRUPA SOCIETY
road	1102
landmark	AZAD ROAD
area	MATWNGA
city	CAR SHOWROOM
state	MAMANAGAR
pin	400036
phoneno	8855321064
mobileno	9876543211
alternatephoneno	
alternatemobileno	
adhar	599092368821
dob	25081991
pan	VNXY212340

Figure 23:Predicted output for production level form

Some observations from the predictions

1. The prediction of the model is excellent for most of the form. Some fine tuning can convert it to a full scale software.
2. Printed box demarcating the writing area often interferes with the model prediction. Figure 22 shows a couple of instances “F” has been predicted as “E” due to the presence of a printed at the bottom of the character level image.
3. The printed bounding boxes on the form also are recognised as “L” or “I” in some of the cases.

4. Similar looking alphabets like “D”, “O”, “G”, “Q” and “I”, “L” are often misclassified. Other combinations include “I”, “T” also.
5. The method works very well for forms which have no scanning anomalies. However, the method fails when scanning of the form is not appropriate. Such situations can be handled by streamlining the scanning process or by accounting for an error margin (using trigonometry) for calculation of relative distance from anchor boxes. Using two anchor points on the form instead of one can also help. This is under investigation.

Fields based Single Shot Detection Method (FSSDM)

The AM method relies on some form of manual intervention by which the anchors and the relative distances of the various field classes from the anchor are determined. In an attempt to develop a holistic and automated method to identify the location of the fields, a Single Shot Detention (SSD) method has been evaluated. The objective of the SSD algorithm is to perform the same task as the previously discussed Anchor Based Template Matching Method. Once the field classes are extracted with respect to handwritten regions, the same method of isolating character images using a 78x78 pixel cropping matrix is proposed to be followed. Tensorflow Object Detection API (Tensorflow Object Detection API, n.d.) has been used to develop a Mobilenet based SSD algorithm.

The TensorFlow Object Detection API does not produce training and testing accuracies by default. However, it computes the loss values per iteration and it is used to measure the model performance. The low loss values are preferred for better detection.

An Object Detection model performs well when the training and testing images satisfy below characteristics:

- The content of the bounding boxes of similar objects (characters or fields) must be approximately identical at pixel level across different forms.
- The content of the bounding box of one object (character or field) must be different at pixel level from another object in the same form.

Unfortunately, both of above premises are not satisfied with the scanned forms used in the present work. This is mainly because one particular set of characters can be present in

multiple fields. Secondly, the fields can have many blank spaces and there is no fixed pattern of spaces, either in one field or across all fields.

The labelled training data is created by using LabelImg as discussed before (Figure 10). The classes that are labelled are given in Table 8.

Table 8: List of pre-identified fields for handwritten text recognition

adhar	alteranatephoneno	Alternatemobileno	alternatephoneno	Area
city	Dob	email	flat	landmark
Mobileno	Name	pan	phoneno	Pin
Policyfrom	Policyno	policyto	road	State

Configurations for the ssd_mobilenet_v1 object detection model and the details of the training data are descried in Table 9.

Table 9: Parameters for ssd_mobilenet_v1 model development and training data

Number of total forms	571
Number of valid forms	569
Number of fields per form	20
Number of fields expected from the forms	11380
Number of valid fields from the forms	10748
Train : Test ratio	0.8 : 0.2
Number of steps	400
Input IMAGE_SIZE	828 X 1169 pixels
Feature Extractor	ssd_mobilenet_v1

Figure4 shows an example of a test images with bounding box as produced by the object detection model.

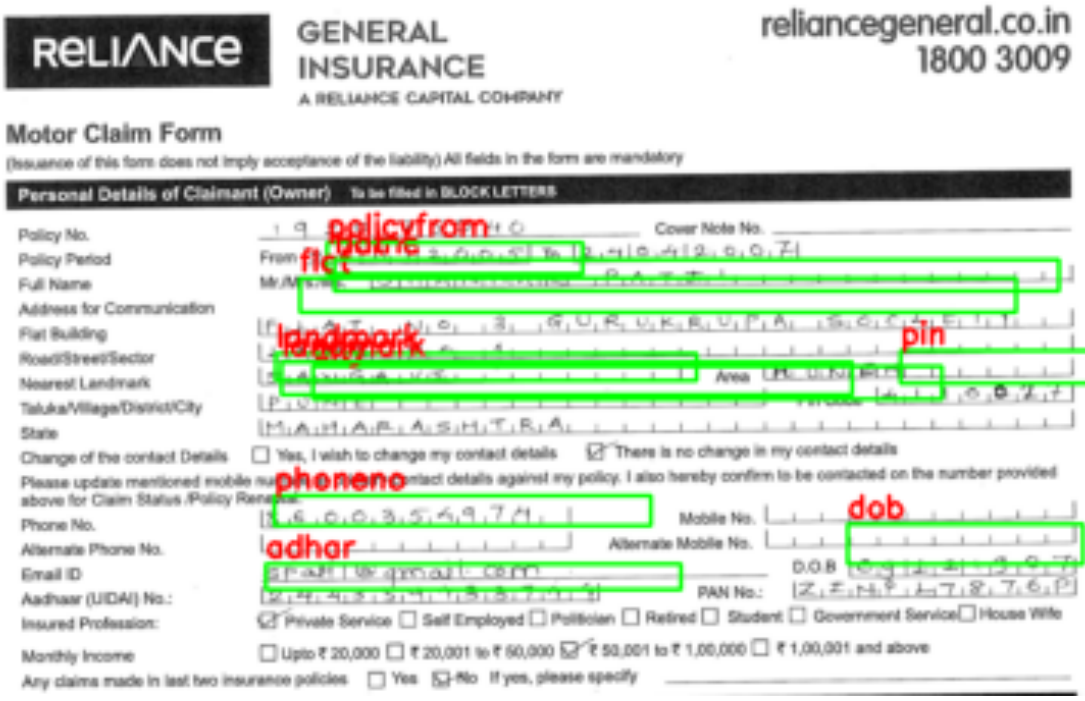


Figure 24: Predicted output of SSD Object Detection Model for field labels

Following inferences are drawn from the SSD based semi-automated object detection process.

- A bounding box corresponding to a field is approximately covering 5-10% of the entire scanned area.
- Object detection models do not perform well to correctly identify the location of every bounding box.
- Object classification cannot be directly applied on the output of this object detection model.

For the character based classification to be reliable and accurate the field level extraction of hand written data has to be reliable and accurate as well. In this semi-automated SSD based approach, it is seen that although the model is able approximately gauge the region of handwriting for each field class, it is not able to reliable capture the regions. Compared to the AM method discussed in the foregoing section, this method is less reliable and accurate. Hence, this method is not pursued further.

Character based Single Shot Detection Method (CSSDM)

The next method developed and evaluated is a filly automatic method to extract the characters and their coordinates. For this purpose, a Mobilenet based SSD technique is used as in FSSDM. The only difference between the implementations is the fact that training data

is created using a different labelling strategy. In this case unlike in the case for FSSDM, each individual handwritten character is labelled as shown in Figure 4. This information is then fed in to an SSD algorithm as a CSV file much like in FSSDM. However, as this method directly extracts the character level data, there is no need for an additional step of image classification. This is potentially a holistic digitisation method.

Configurations for the ssd_mobilenet_v1 object detection model and the details of the training data are descried in Table 10.

Table 10: Parameters for ssd_mobilenet_v1 model development and training data

Number of total forms	570
Number of valid forms	570
Train : Test ratio	0.8 : 0.2
Number of steps	75000
Input IMAGE_SIZE	708 X 1000 pixels
Feature Extractor	ssd_mobilenet_v1

Figure 25 shows the prediction of the model on unseen production data.

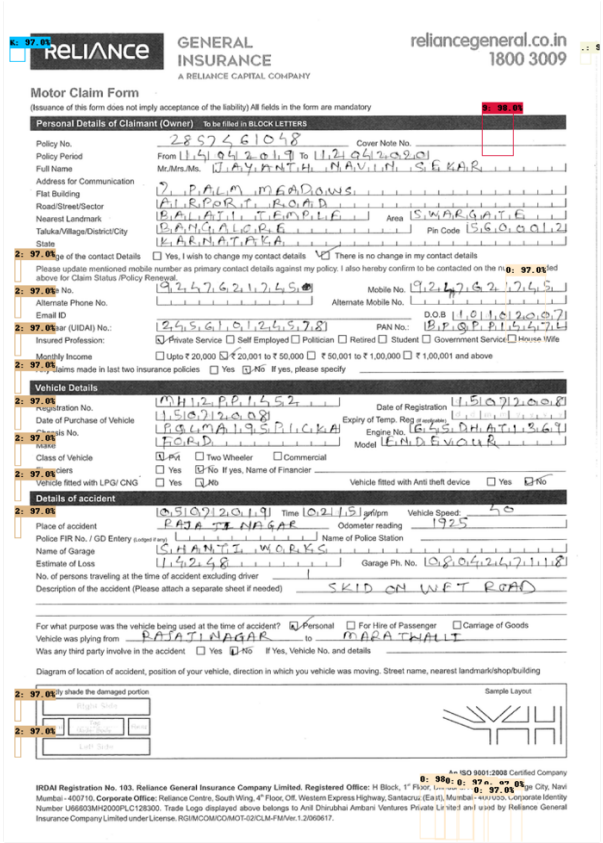


Figure 25: Prediction of CSSDM on production level test form

It is seen that the prediction is very poor and not even close to what the actual prediction should be. This is attributed to the large number of classes and the fact that all the class bounding boxes are very small as compared to the form. Even training the model from scratch did not improve the performance of the model.

Thus, this model was discarded and not pursued with.

Tesseract with Python wrapper (PyT)

Tessaract is a combination of two items – (a) an OCR engine - libtesseract and (b) a command line program – tesseract. Tesseract was originally developed at Hewlett-Packard Laboratories Bristol and at Hewlett-Packard Co, Greeley Colorado between 1985 and 1994, with some more changes made in 1996 to port to Windows. In 2005 Tesseract was open sourced by HP. Since 2006 it is developed by Google (Google, n.d.). Since this program is written and developed in C++, a C++ compiler is required to run it. Tesseract uses Leptonica library for opening input images. "Leptonica is pedagogically-oriented open source site containing software that is broadly useful for image processing and image analysis applications" (Bloomberg, 2001).

Two methods have been evaluated using a Tessaract program with Python Wrapper (PyT).

1. Using a pre trained PyT program for predicting production level forms (Universal Model)
2. Fine tuning a pre trained PyT program and then predicting production level forms

The outcome of the evaluation is described in the subsequent discussions.

Universal Model

A pretrained version of the PyT model has been evaluated and used to predict hand written text in scanned forms. PyT program initially looks to identify the regions with text in them and then isolates them to "read" them digitally. Figure 26 shows a test form where PyT has identified the region in the form which has textual data (green boxes). From this figure it is seen that PyT performs very well in identifying textual region in a document. This is expected as Tesseract OCR is basically designed for scanned document text (Ansari, Shah, Yasmin, Sharif, & Fernandes, 2018).

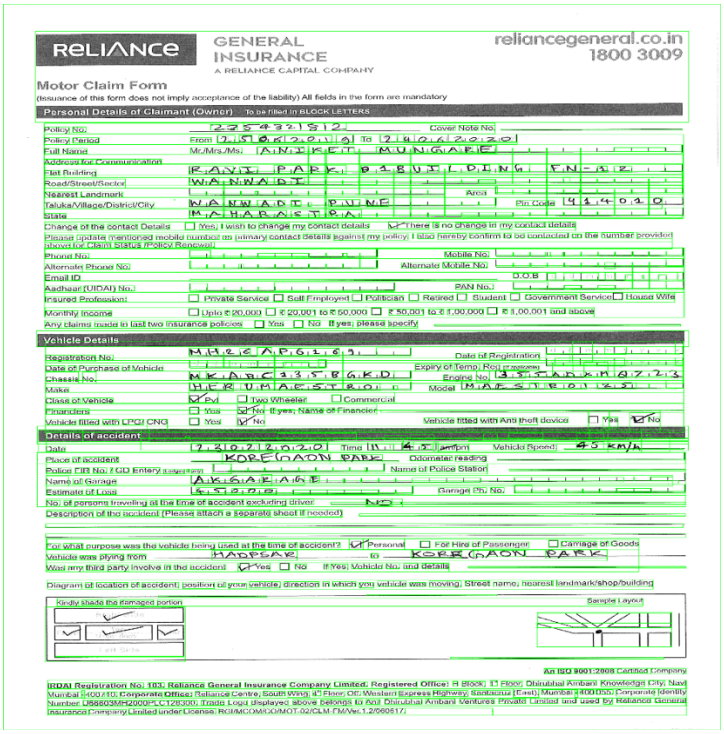


Figure. 26: Textual region detection by PyT

Although Tessaract OCR is able to predict and decipher the printed letters on the scanned form very well and with very low error, as it is not designed for hand written data, it is not able to predict hand written characters. This is shown in Figure 27. The prediction is not contextual at all and very poor. It is also seen that the printed bounding boxes also throw the model off and results in incorrect predictions.

Fine Tuned PyT Model

To overcome the inability of PyT in predicting, the model parameters are fine tuned. To achieve this, five fonts which represent hand written data are used to train the model. These are listed below and shown in Figure 28.

1. Santos Dumont
2. WC Mano Negra BTA
3. Queen Gladys
4. YouMurderer BB
5. Alanis Hand

Font details are taken from www.1001fonts.com (1001Fonts, n.d.).

These new fonts are added to the training data as per the instructions given in The Tesseract OCR documentation (tesseract-ocr, n.d.). An example of the ground truth labels created by PyT is shown in Figure 29.

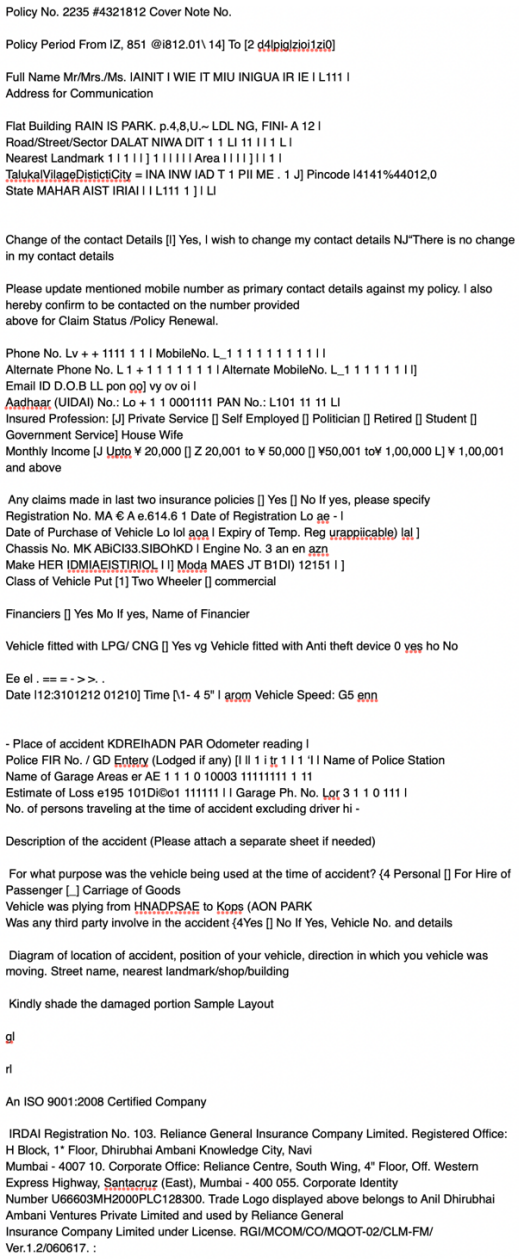


Figure 27: Prediction by PyT (Universal model) on production level form

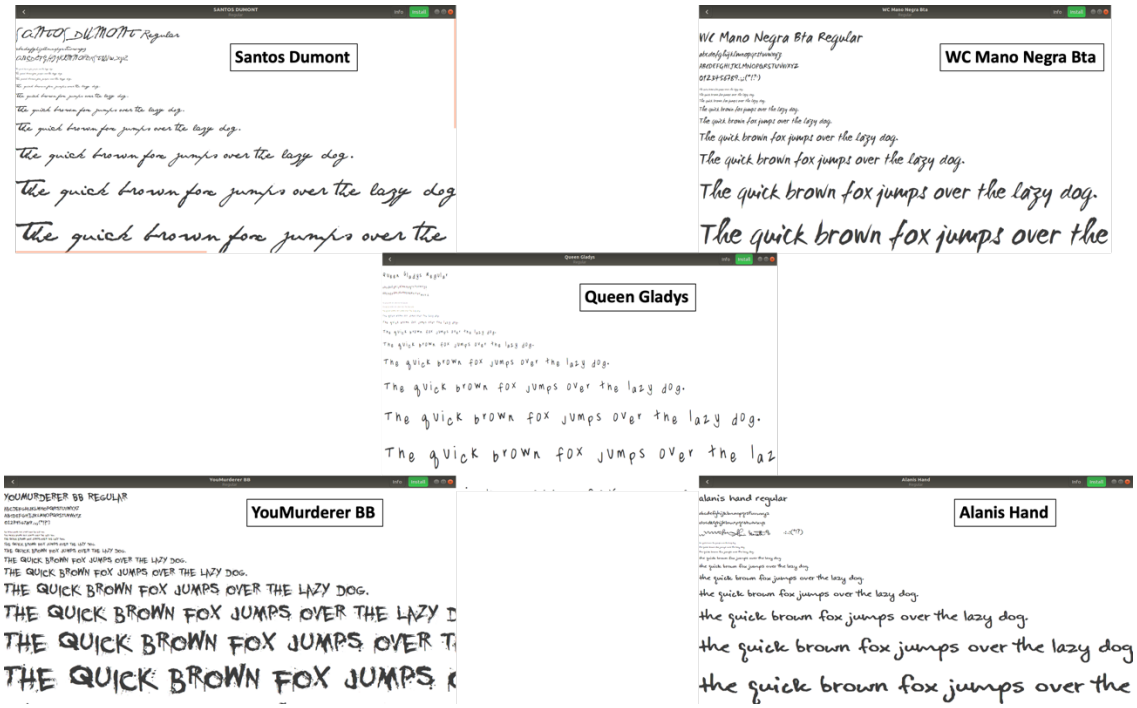


Figure 28: Fonts used for training PyT

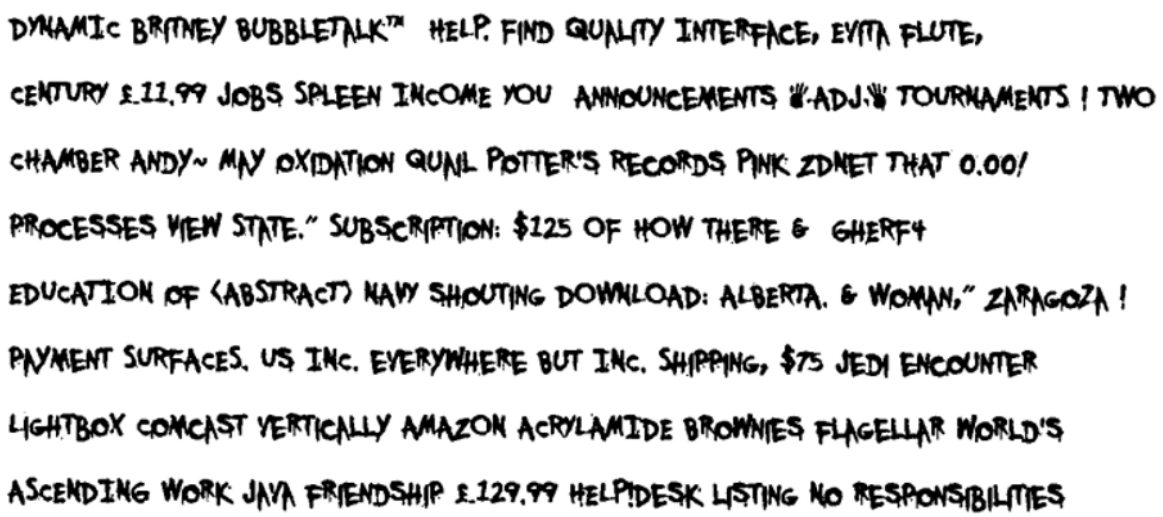


Figure 29: Example of ground truth label created by PyT for training on new fonts that resemble handwriting

The new retrained model is then tested on production level test data for efficacy in predicting hand written data. A sample prediction is shown in Figure 30. The prediction of printed data is good but is inferior to the Universal Model discussed earlier. The prediction of handwritten data has not improved at all by retraining. It is seen that in some places the prediction has

become worse as compared to the Universal Model. This again is due to the fact that the Tesseract OCR model was never designed for hand written data.

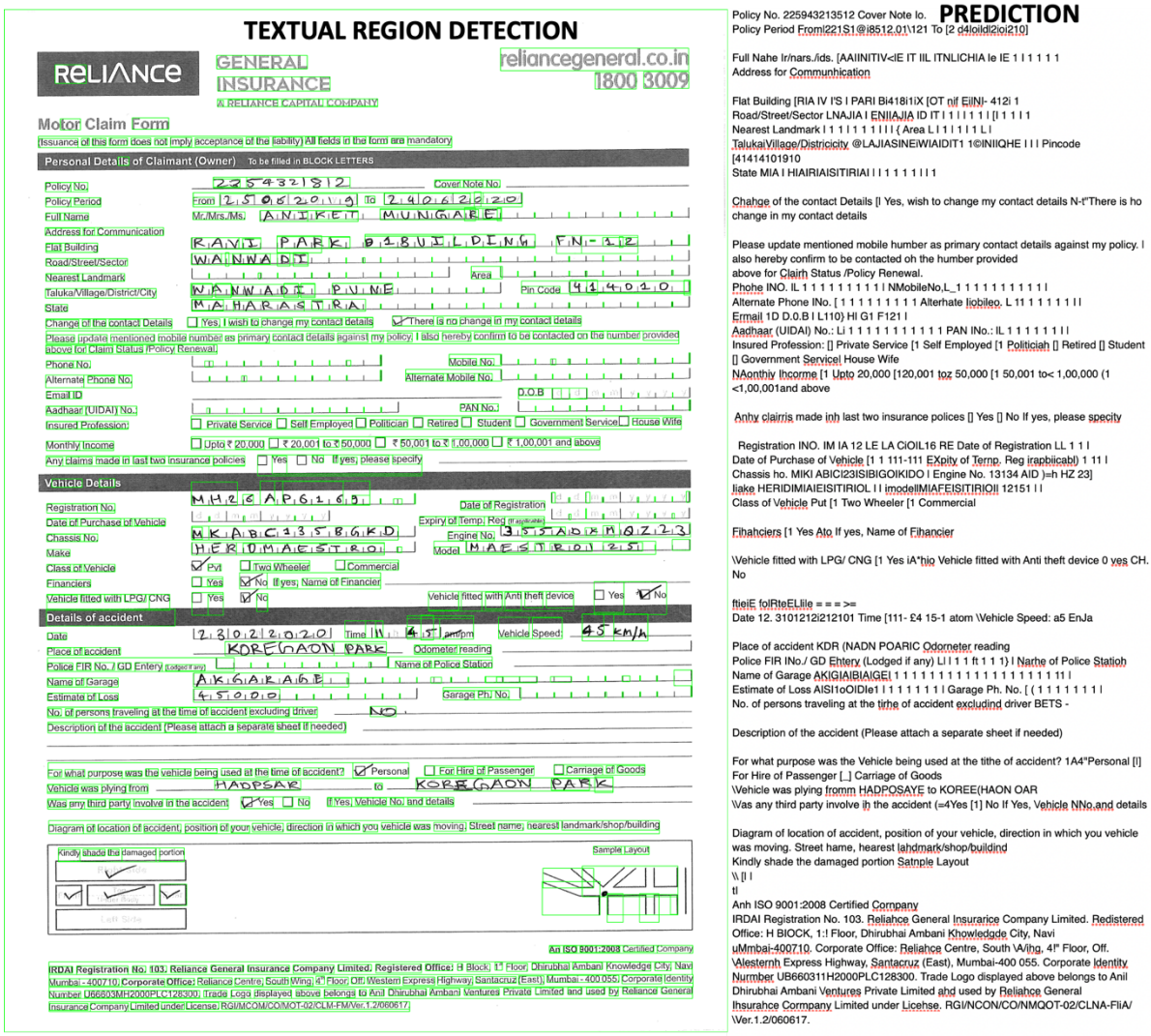


Figure 30: Textual region detection and text prediction by retrained PyT on production level test form

Another option available for retraining PyT is to train the model from scratch with custom hand written data. As the data set that is currently under investigation is a real life data set, the variations in handwriting is very large. It is practically not possible to create a font type for each handwriting style to retrain the model. It is due to this fact that a PyT based model is not pursued any further.

Summary of findings and model selection for production processes

Two different kinds of CNNs have been evaluated for text classification – Single-Channel Conventional CNN and Multi-Channel CNN. Both models have been evaluated on production data and the Multi-Channel CNN is seen to outperform the Single-Channel CNN on every performance metric. Hence MCCNN is used for all the further evaluation study where a pretrained model is not used.

Four methods have been investigated for efficacy in digitizing hand written forms. The summary of the findings in the present study is listed in Table 11.

Table 11: Summary of different methods evaluated

Method	Pros	Cons	Accuracy	Automation	Comments
Anchor based method with OpenCV Template Matching backend (AM) with MCCNN	1. Easy to automate for a particular form design 2. Scalable 3. Robust 4. High Accuracy	1. Scanning orientation determines usability 2. Redo for different form – not general 3. Adapting for new document tedious 4. High manual inference	**Very high**	High for similar documents Low for new documents	**Recommended for use**
Fields based Single Shot Detection Method (FSSDM)	1. Automated detection 2. Low manual inference	1. Poor accuracy 2. High data requirement	Poor	High	Not recommended for use
Character based Single Shot Detection Method (CSSDM)	1. Very high automation 2. No manual inference	1. Very poor	Very poor	Very high	Not recommended for use
Tesseract with Python wrapper (PyT)	1. High automation 2. Already trained for textual data	1. Not designed for hand writing 2. Very tedious and not practical to create custom fonts for each handwriting style	Very high for printed data; very poor for handwritten data	Very high	Not recommended

Based on the findings, it is recommended that **Anchor based method with OpenCV Template Matching backend (AM) with MCCNN classifier** be used for such predictions.

Conclusions and Summary

1. 572 hand written forms were filled and collected to form the dataset for method development and evaluation. Care has been taken so that the forms are filled by over 200 individuals to account for variation in handwriting reflective of real like scenario.
2. 572 forms labelled character wise and field wise.
3. Character wise labelling gave nearly 85,000 training images for development of image classifier.
4. Two image classifiers evaluated – SCCNN and MCCNN. It is seen that MCCNN has outperformed SCCNN on every performance metric
5. Four different object detection methods have been tested.
6. Of the methods tested, **Anchor based method with OpenCV Template Matching backend (AM) with MCCNN classifier** has the best accuracy and is recommended for use.
7. Other methods evaluated include two versions of SSD (Character and Field level detection) and Py-Tesseract (both Universal model as well a Fine tuned model). These models do not perform well on handwritten data.

Appendix

The number of forms used is of very high importance. Multi-Channnel CNN is used to show this.

Table 12: Influence of number of forms on accuracy

Training Dataset	Training Accuracy	Validation Accuracy (EMNIST)	Testing Accuracy (HW-test dataset)
EMNIST	94%	94%	0.2%
EMNIST + data from 67 forms	93%	94%	79%
EMNIST + data from 131 forms	93%	93%	84%
EMNIST + data from 260 forms	94%	94%	91%
EMNIST + 570 forms (h-EHm dataset)	93%	93%	93%

Bibliography

Shruthi, A., & Patel, M. S. (2015). Offline Handwritten Word Recognition using Multiple Features with SVM Classifier for Holistic Approach. *International Journal of Innovative Research in Computer and Communication Engineering, 3*(6), 5989-5995.

Amir, A., & Jindal, A. (2014). Automatic classification of handwritten and printed text in ICR boxes. *2014 IEEE International Advance Computing Conference (IACC)* (pp. 1028-1032). Gurgaon: IEEE.

LeCunn, Y., & Bengio, Y. (1995). Convolutional netwroks for images, speech and time series. In *The Handbook of Bran Theory and Neural Networks.* Cambridge, USA: MIT Press.

Chen, L., Wang, S., Fan, W., Sun , J., & Satoshi, N. (2015). Reconstruction combined training for convolutional neural networks on character recognition. *3th International Conference on Document Analysis and Recognition (ICDAR)* (pp. 431-435). IEEE.

Palehai, D., & Fanany, M. I. (2017). Handwriting Recognition on Form Document Using Convolutional Neural Network and Support Vector Machines (CNN-SVM). *The 5th International Conference on Information and Communication Technology.* Malacca, Malaysia: IEEE.

Nasien, D., Haron, H., & Yuhaniz, S. S. (2010). Support Vector Machine (SVM) for english handwritten character recognition. *2010 2nd International Conferance of Computur Engineering and Application, ICCEA 2010*, (pp. 249-252).

Hussain, J., & Vanlalruata. (2018). A Hybrid Approach Handwritten Character Recognition for Mizo using Artificial Neural Network. *International Conference on Advanced Computation and Telecommunication (ICACAT)* (pp. 1-6). Bhopal: IEEE.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 32*(9), 1627-1645.

Girshick, R., Donahue, J., Darrerr, T., & Malik, J. (2014). Rich fea- ture hierarchies for accurate object detection and semantic segmentation. *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference* (pp. 580-587). IEEE.

Redmon, J., Divvala, S. K., Girshick, R. B., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 779-788). IEEE.

Breuel, T. M., Ul-Hasan, A., Al-Azawi, M. A., & Shafait, F. (2013). High-Performance OCR for Printed English and Fraktur Using LSTM Networks. *2013 12th International Conference on Document Analysis and Recognition* (pp. 683-687). Washington DC: IEEE.

Sharma, N., Patnaik, T., & Kumar, B. (2013). Recognition for Handwritten English Letters : A Review.

Al Islam, M. A., & Khan, S. K. (2019). HishabNet: Detection, Localization and Calculation of Handwritten Bengali Mathematical Expressions. *22nd International Conference on Computer and Information Technology (ICCIT).* IEEE.

NIST. (2019, March 4). *NIST - National Institute of Standards and Technology*. Retrieved March 2020, from The EMNIST Dataset: https://www.nist.gov/itl/products-and-services/emnist-dataset

Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). *EMNIST: an extension of MNIST to handwritten letters*. Retrieved from NIST: https://arxiv.org/pdf/1702.05373v1.pdf

Tzutalin. (2015). *Git code*. Retrieved from LabelImg: https://github.com/tzutalin/labelImg

Kingma, D. P., & Ba, J. L. (2015). ADAM: A Method For Stochastic Optimization. *International Conference on Learning Representations 2015.* San Diego, CA.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv:1408.5882*.

Guo, B., Zhang, C., Liu, J., & Ma, X. (2019). Improving text classification with weighted word emb e ddings via a multi-channel TextCNN model. *Neurocomputing, 363*, 366-374.

OpenCV Dev Team. (n.d.). *OpenCV 2.4.13.7 Documentation*. Retrieved May 2020, from Template Matching: https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html

Tensorflow Object Detection API. (n.d.). *GitHub*. Retrieved May 2020, from Tensorflow Object Detection API: https://github.com/tensorflow/models

Google. (n.d.). *Tesseract OCR*. Retrieved May 2020, from GitHub: https://github.com/tesseract-ocr/tesseract/blob/master/README.md

Bloomberg, D. (2001). *Leptonica*. Retrieved May 2020, from Leptonica: http://www.leptonica.org

Ansari, G. J., Shah, H. J., Yasmin, M., Sharif, M., & Fernandes, S. L. (2018). A novel machine learning approach for scene text extraction. *Future Generastions Computer Systems, 87*, 328-340.

1001Fonts. (n.d.). *New and Fresh Fonts*. Retrieved May 2020, from 1001Fonts: www.1001fonts.com

tesseract-ocr. (n.d.). *Tessdoc - Tesseract Documentation*. Retrieved May 2020, from How to use the tools provided to train Tesseract 4.00: https://tesseract-ocr.github.io/tessdoc/TrainingTesseract-4.00.html