information

Article

Artificially Intelligent Readers: An Adaptive Framework for Original Handwritten Numerical Digits Recognition with OCR Methods

Parth Hasmukh Jain [1], Vivek Kumar [2,*], Jim Samuel [1], Sushmita Singh [3], Abhinay Mannepalli [1] and Richard Anderson [1]

[1] Edward J. Bloustein School, Rutgers University, Piscataway, NJ 08854, USA; pj269@scarletmail.rutgers.edu (P.H.J.); jim.samuel@rutgers.edu (J.S.); am2977@scarletmail.rutgers.edu (A.M.); rianders@docs.rutgers.edu (R.A.)
[2] Department of Mathematics and Computer Science, University of Cagliari, 09124 Cagliari, Italy
[3] School of Computer Science and Mathematics, Liverpool John Moores University, Liverpool L3 2AF, UK; sushmitafordata@gmail.com
* Correspondence: vivek.kumar@unica.it

Abstract: Advanced artificial intelligence (AI) techniques have led to significant developments in optical character recognition (OCR) technologies. OCR applications, using AI techniques for transforming images of typed text, handwritten text, or other forms of text into machine-encoded text, provide a fair degree of accuracy for general text. However, even after decades of intensive research, creating OCR with human-like abilities has remained evasive. One of the challenges has been that OCR models trained on general text do not perform well on localized or personalized handwritten text due to differences in the writing style of alphabets and digits. This study aims to discuss the steps needed to create an adaptive framework for OCR models, with the intent of exploring a reasonable method to customize an OCR solution for a unique dataset of English language numerical digits were developed for this study. We develop a digit recognizer by training our model on the MNIST dataset with a convolutional neural network and contrast it with multiple models trained on combinations of the MNIST and custom digits. Using our methods, we observed results comparable with the baseline and provided recommendations for improving OCR accuracy for localized or personalized handwritten text. This study also provides an alternative perspective to generating data using conventional methods, which can serve as a gold standard for custom data augmentation to help address the challenges of scarce data and data imbalance.

Keywords: OCR; adaptive; custom; digits; MNIST; informatics; machine learning; deep learning

Citation: Jain, P.H.; Kumar, V.; Samuel, J.; Singh, S.; Mannepalli, A.; Anderson, R. Artificially Intelligent Readers: An Adaptive Framework for Original Handwritten Numerical Digits Recognition with OCR Methods. *Information* **2023**, *14*, 305. https://doi.org/10.3390/info14060305

Academic Editors: Eftim Zdravevski, Petre Lameski and Ivan Miguel Pires

Received: 12 April 2023
Revised: 16 May 2023
Accepted: 19 May 2023
Published: 26 May 2023

1. Introduction

We are seeing a rapid increase in the breadth and depth of adaptive artificial intelligence (AI) solutions to improve performance in areas such as computer vision for character recognition, which is receiving significant industry attention [1,2]. Optical character recognition (OCR) is an AI method that mimics the human-intelligence capability of visual recognition for the computational identification of machine and handwritten text and digits from a broad range of images containing text [3,4]. This includes identification of text or digits from images of typed text, handwritten text, or printed text into machine-encoded text, either from a scanned document as a pdf file, a picture of a piece of paper in png or jpeg format, or a scene photo with text in it, such as text on a coffee cup, text on the cover page of a book, or the license number on number plates. The number of researchers, academic centers and labs, and companies researching and developing computer vision and OCR solutions has risen significantly over the past several years [2]. Hence, an array of OCR solutions and tools are available widely. We discuss a few OCR applications in the context of our research below.

OCR is increasingly used for automation and enhancement of workflow and business processes. OCR is used by organizations to manage handwritten and printed text and document types identification and to parse data according to complex rules. OCR also has social and societal functional capabilities; for example, OCR can be used to provide an assistive solution for those with vision impairment by converting text images into text-to-speech [5]. One of the significant challenges faced by hospitals is the loss of patient medical information. OCR technologies can support electronic medical records and health information capture. Additionally, OCR can be used to automate information extraction from test results for input into hospital information management systems (HIMS), supporting the effective representation of patients' medical histories. OCR tools such as Scayne and Expensify are used in finance, accounting and business process management for functions such as receipt recognition [6]. OCR is also used in loan processing, security and fraud detection, government sector services such as scanning legacy identification documents and automated license plate detection, and for supporting cultural and historical heritage initiatives, while fairly robust solutions are available for typed or printed machine-readable texts, automated handwriting recognition systems are yet to provide high levels of accuracy due to the challenges of variations in handwriting styles and composition of visual structure. Resolving these challenges remains an open problem, and improving handwriting OCR can lead to very valuable insights for decision support through the inclusion of handwritten data in the evaluation of public sentiment and other textual analytics solutions [7,8].

It is, thus, evident that OCR is an important technology and AI capability. It is, therefore, important to improve the accuracy of OCR tools with handwritten text, especially for specific handwriting styles, such as the handwriting of a particular doctor or a handwriting style from a particular region. Our work contributes to the body of knowledge of OCR methods by introducing the idea of fine-tuning MNIST with training data from a custom dataset belonging to a particular style to improve the accuracy for that specific style of handwritten digits. The model, method, and process used for OCR with such custom personalized or localized data can be extended to any handwritten style. Our work can also provide an alternative mechanism for generating custom data and conversion to MNIST format using conventional methods and can serve as a standardized mechanism for data augmentation to address the challenges associated with scarce data and data imbalance. The main contributions of this paper are as follows:

- We create a new handwritten custom dataset from a single contributor to ensure overall consistency of style.
- We train and apply a convolutional neural network to MNIST data for validation and then extend the method with custom data.
- Furthermore, we provide insights into the effects of training the model with variations in the proportions of custom data with and without rotations
- We provide a framework for adaptive OCR applicable to handwritten documents from a single source or belonging to a particular style of writing.
- Finally, we provide a comprehensive discussion of how the adaptive framework could be implemented and potential improvements with alternate technological solutions.

The rest of the paper is organized as follows. Section 2 presents the literature background of notable OCR methods. Section 3 mentions the problem statements and provides details of the dataset, and preprocessing strategies used for the experiments. Section 4 presents the stages involved in OCR and the architecture of the employed deep learning (DL) approaches. Section 5 sums up the total experiments performed and presents the experimental outcomes. Section 6 presents the analyses of the obtained results and the limitations of the employed methods. Finally, Section 7 provides the concluding remarks and the future research direction.

2. Related Works

In this section, we review the existing works related to OCR. OCR systems can be classified into two groups depending on the kind of input: handwriting recognition and

machine-printed character recognition. OCR, to some extent, is considered a solved problem for standard machine text images. Machine-printed character recognition is a relatively straightforward problem since characters are typically of uniform dimensions, and their positions on the page can be predicted. The image input for this model can be in any format, such as JPEGs, PDFs, and documents. However, handwritten text recognition, a critical component of OCR, is still challenging. Unlike printed text, the wide range of human handwriting styles and the poor quality of handwritten text pose significant challenges in converting it to machine-readable language [9].

Handwritten Character Recognition plays a crucial role in the postal services industry by automating the process of address interpretation and facilitating efficient mail sorting. In earlier approaches, high-volume applications utilized task-specific readers to achieve high system throughput. These readers focused on specific fields of interest, taking advantage of the similarity in size and layout structure among similar documents. This enabled image scanners to extract the desired information efficiently, reducing image processing and text recognition time [10]. Address readers, another type of earlier system, were designed to locate the destination address block on mail pieces and read the ZIP code within that block. In cases where additional fields were read with high confidence, the system could generate a nine-digit ZIP code and apply a corresponding bar code on the envelope [11]. Modern systems in postal services have embraced advanced techniques, such as Hidden Markov Models (HMM), for handwritten address recognition. Kornai conducted a study on two variants of an HMM-based postal OCR system, one with pre-segmentation and one without. This study reported promising results on the CEDAR dataset, contributing to understanding handwritten address recognition [12]. Furthermore, region-specific use cases have emerged, such as recognizing Sinhala handwriting using OCR and image processing technologies [13]. These specific applications cater to the unique challenges of different languages and writing systems. Grid-based approaches have also gained attention in offline handwritten word recognition. Patel and Reddy explored the impact of a grid-based approach using Principal Component Analysis (PCA) for improved representation of handwritten Kannada words. Their study focused on recognizing district names within the Karnataka state, showcasing the applicability of grid-based methods in specific contexts [14], while significant progress has been made, challenges still exist in achieving high accuracy rates and adapting to varying handwriting styles. Ongoing technological research and development will continue to enhance the efficiency and effectiveness of postal services, ultimately improving the accuracy and speed of address interpretation and mail sorting processes.

Improvement of OCR accuracy for handwritten text remains an important issue for other industries as well, such as healthcare (handwritten medical notes by healthcare practitioners with varying styles), insurance (handwritten forms, mail envelopes, and handwriting from digital devices—stylus and touch screen handwriting), and banking (handwritten letters, checks, forms, and documentation, especially in developing and underdeveloped nations) [2,15,16]. In healthcare, OCR can be adapted to improve accuracy with individual medical practitioners. Similarly, OCR can be adapted in banking and insurance to enhance accuracy with individual users, clients, and executives in organizational processes with repetitive handwritten input, while postal services use OCR for handwriting, there is no need to customize it for the user. However, even postal services could customize based on the regional styling of characters, wherein the characters may vary within the same language based on regional grouping [17,18].

Handwritten text recognition systems are available as online and offline applications. Online systems work in real-time when users write the characters into a mechanism such as a touch screen or another touch-sensitive interface. They are structured differently since they can store time-based information, such as speed, velocity, number of strokes, and writing direction. Offline recognition systems, on the other hand, work with static data, i.e., bitmap images as input, and often require extensive pre-processing to perform these tasks. Consequently, the recognition task can be more challenging. More recent technologies

based on the transformers architecture have shown promise in reading handwritten text, but the need to improve accuracy further remains [19].

The quality of OCR-generated text can impact various downstream natural language processing tasks, including NER and dependency parsing in certain workflows. Retrieval results on OCR-generated text should be viewed cautiously and manual supervision is recommended, as they can result in false positives. Topic models can also be affected by OCR quality in certain workflows, leading to increasingly divergent topics. It is, therefore, recommended to use high-quality OCR above 90 percent and better [20]. A new architecture for OCR-based image captioning called Long Short-Term Memory Relational Aware Pointer Network (LSTM-R) uses Long Short-Term Memory and a relation-aware pointer network to incorporate the geometrical relationships between OCR tokens. The relation-based pointer network copies words from OCR tokens based on their geometric relationship, and the model is optimized with multi-label loss. Experimental results demonstrated the effectiveness of LSTM-R over benchmarks, achieving state-of-the-art results on the TextCaps dataset [21]. It is also important to consider potential directions for developing post-OCR processing approaches. It is possible to create artificial materials to train machine learning models, utilize external resources for neural network-based models, improve error detection and correction tasks, and analyze performance with different OCR errors; it is also useful to focus on 'handling errors' involving word segmentation, consider 'different weights for evaluation metrics', create datasets with more detailed information, and develop 'post-processing approaches for languages other than English' [22]. Analysis by [23] revealed that most of the OCR software is incapable of accurately detecting variations in fonts and formats. Furthermore, while it can effectively convert text images, it fails to appropriately convert mathematical equations and symbols, providing only plain text as output. This highlights the need for new and improvised OCR tools to be developed in the future that can effectively perform the identification of mathematical equations and symbols [23].

According to [24], creating a dataset specific to a particular language and an associated period is highly beneficial for OCR correction workflows. Interestingly, using only a language-specific dataset based on the Bible created more errors than corrections. Ref. [24] considered this as being similar to a person from the biblical era attempting to fix OCR errors in modern texts. Furthermore, the error generation algorithm for Hebrew historical newspapers required only 105 manually corrected articles, which is significantly less than the amount of labeled training data needed for a neural network [24]. Ref. [25] showed that document AI and Textract had consistently lower error rates than Tesseract on various documents with different noise levels and fonts in English and Arabic.

Tesseract was more sensitive to noise and performed better only on noise-free documents in English. Further analysis by noise type showed that Textract and Tesseract performed better on grayscale images than on color images and struggled with blur, and Tesseract was more sensitive to salt and pepper noise. In conclusion, Document AI and Textract had better OCR accuracy than Tesseract, and accuracy varied across languages and noise types [25].

It is also important to take a quick view of some of the state-of-the-art solutions available currently. Popular OCR solutions include Tesseract, EASY OCR, Keras OCR, PyPDF2, DocTr, and TrOCR. Tesseract was initially proposed in [26]. Tesseract was one of the first architectures to handle white-on-black text with ease [26]. Tesseract was initially developed at HP in 1984 and 1994. Tesseract was open-sourced in 2005, then acquired by Google, and it works well with Tensorflow. Tesseract solves the problem of text localization, identifying where the text is in the document or an image [27]. Tesseract 4.00 has a new neural network subsystem as a text line recognizer. It has its origins in [28]. Easy OCR is the most straightforward way to implement OCR, and Easy OCR can be implemented with just a few lines of code. It is a Python package, and Jaided AI [29] maintains it. It was released in July 2020 and received a recent update in June 2022. EasyOCR uses Pytorch and OpenCV in the backend [30]. Keras OCR [31] provides end to end training pipeline to build new OCR models. It is a slightly better and packed version of Keras CRNN implementation and

CRAFT text detection model. It provides a very accurate API for training a text detection and OCR pipeline [32]. Convolutional Recurrent Neural Network is a CNN, RNN, and CTC (Connectionist Temporal Classification) loss combination for image-based sequence recognition applications, such as scene text recognition and OCR [33]. The convolution is used to extract the feature sequence from an input image. The Recurrent neural network predicts the label distribution for each frame. Furthermore, a transcription layer translates the per-frame predictions into a final label sequence [34]. A convolutional neural network is used to construct Character Region Awareness For Text ('CRAFT') Detection to provide the character region and affinity scores. Individual characters are localized in the image using the region score, and each character is grouped into one instance using the affinity score [35].

PyPDF2 is a Pure-Python library that Fenniak [36] created as a PDF toolkit. It has the ability to extract document information [37], divide and merge documents page by page, crop pages, combine numerous pages into one, and encrypt and decode PDF files. DocTR was published by Mindee in March 2021 [38]. Its latest version was released in March 2022. It achieves end-to-end OCR in a two-stage approach; in the first stage, it implements text detection using Resnet 50 architecture, and in the second stage, it achieves text recognition by using CRNN. It can process both PDFs and images. It also detects rotated images. It can show text localization along with confidence. Its overall good implementation and gives overall accurate results [39]. More recently, we have seen the use of transformers for converting textual or handwritten data to machine-encoded text. The transformer concept and architecture were first introduced in the famous paper by [40], and since then, transformers have been increasingly used in NLP, computer vision, and other AI tasks. A Transformer architecture is a stack of encoder and decoder layers. Encoder and Decoder are composed of modules stacked on top of each other multiple times. The encoder will convert the English text into numerical representations. Those numerical representations are fed into a Decoder, and the decoder converts to desored output for further processing. Microsoft's TrOCR models are encoder–decoder models with a text Transformer as the decoder and an image Transformer as the encoder [41]. The Hugging Face version of the TrOCR model is popular because of its longevity and simplicity. With many other implementations, it is observed that the same model or libraries cannot recognize handwritten and typed text with high accuracy. The TrOCR model can convert both text types into machine-encoded text. Furthermore, experiments have shown that the TrOCR model outperforms the current state-of-the-art models on printed and handwritten text recognition tasks. Table 1 list the notable open-source OCR models.

Table 1. Overview of open-source OCR models.

Sr No.	Name	Brief Description	Authors	Format
1.	Tesseract	Tesseract is performing well for high-resolution images	[42]	Machine text
2.	TrOCR	It can produce very accurate results for machine and handwritten text and is based on Transformers	[41]	Handwritten and Machine text
3.	KerasOCR	Keras-OCR is designed for images. It produces decent results when text is embedded in a picture and the fonts and colors are not coordinated	[31]	Machine text
4.	DocTr	State-of-the-art performances on public document datasets, comparable with GoogleVision	[38]	Machine text
5.	Easy OCR	EasyOCR is a lightweight model that gives accurate results with organized texts like pdf files, receipts, and bills. It supports 80+ languages	[43]	Machine text
6.	PyPDF2	Can be used only for PDF and is not accurate in some cases.	[36]	Machine text

3. Problem Formulation, Datasets, and Preprocessing

This section explains the problem statements tackled, the experimental dataset used, the process involved in generating *Custom* dataset, and preprocessing strategies implemented to prepare the input data for our experiments.

3.1. Problem Formulation

In this work, we explored the idea of developing an adaptive OCR framework and provide reports from our OCR experiments in working with custom data of a unique but consistent style from a single source in addition to baseline MNIST data. Thus, we contribute to addressing the challenge of improving the accuracy of OCR mechanisms with adaptive methods achieved through localized training of models with the inclusion of custom data in the training process.

3.2. Datasets

For our work, we have used two datasets, namely the Modified National Institute of Standards and Technology database (*MNIST*) and our own generated dataset. For ease of understanding, we termed our dataset as the *Custom* dataset.

1. **MNIST:** The *MNIST* is a benchmarking [44] of 60,000 train handwritten digits and a test set of 10,000 (10 class labels) with each example represented as an image of 28 $\times$ 28 gray-scale pixels. Images are normalized to fit in a 20 $\times$ 20 pixel box while maintaining their aspect ratio. Images are centered in a 28 $\times$ 28 image. One column for each of the pixels because there are 28 $\times$ 28 pixels, there are 784 columns, and +1 for the labels. Pixel values are between 0 and 255, where 0 means black and 255 means white.
2. **Custom:** *Custom* dataset consists of 240 training digits, 40 testing digits, and 20 validation digits, ensuring that the model is trained on a diverse set of data and can generalize well to new data points.
   The *Custom* dataset is also made available on our Githubrepository [45].

3.3. Custom Dataset Collection

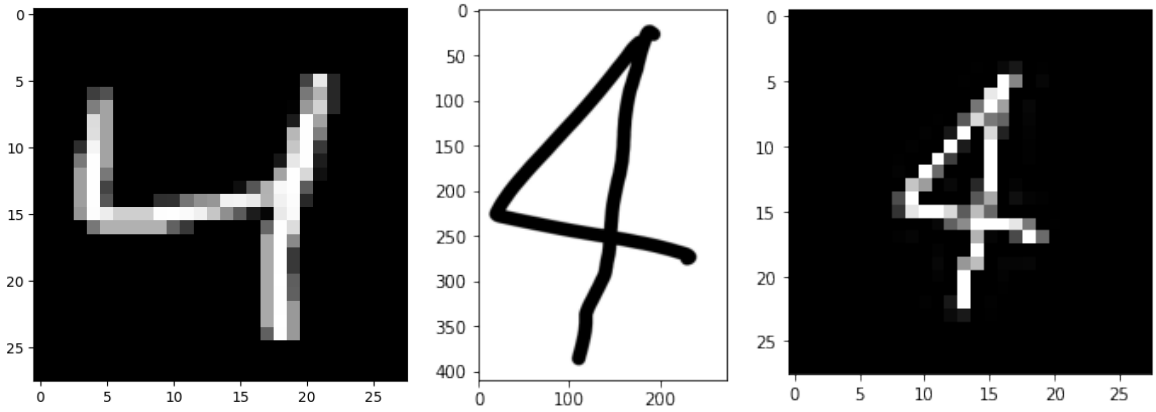
For data collection, we have to create images of the digits. The following steps were taken to accomplish this. Since writing digits on paper and taking a photograph of them can be a challenging and time-consuming process, Microsoft Word version 10 was utilized to make images of the digits. The "draw with trackpad" option in the drawing tab of Microsoft Word was used for this purpose. Subsequently, screenshots of each digit were taken and grouped into batches of 10. Each batch contained digits from 0 to 9, with 10 digits per batch, and the name of the image was the gold label of what the digit represented.

- **Preparing the Custom Dataset:** Our *Custom* dataset consists of 300 handwritten digits collected using the method described in the data collection section. Two users wrote these digits, and we divided the dataset into three parts, each containing 100 digits. We then performed a stratified split on each of the three datasets to create training and testing sets. Specifically, we divided each set into 80 training images and 20 testing images, ensuring each category had an equal number of labels. This stratified split helped to prevent any bias in the model. Next, we concatenated the training images, creating a set of 240 training digits, and concatenated the testing images, creating a set of 60 digits. We then performed a stratified fold on the testing images, splitting them into sets of 40 and 20. The 20 digits were used as a validation dataset.

3.4. Preprocessing the Custom Dataset

To ensure consistency with the MNIST dataset, which has images of a fixed size of 28 $\times$ 28 pixels, we need to standardize our images to the same size. When digitizing a handwritten number, we followed several steps to ensure the accuracy and clarity of the resulting image. Firstly, the handwritten number image is loaded, and then, it is converted to a binary image format using the "L" mode, which only allows each pixel

to be black or white. A white canvas of 28 × 28 is created the standard size for all MNIST images. The width or height of the image is then determined, and the image is resized such that the greatest dimension is 20 pixels, while the smaller dimension is scaled proportionally. To improve the image quality, the antialias function of pillow Library is applied, which smoothens jagged edges by averaging the colors of the pixels at the boundary and sharpening it. Subsequently, the resized image is pasted onto the 28 × 28 pixel white canvas, with 4 pixels from the top or side of the largest dimension used to center the picture. The smallest dimension is placed halfway between the original size of 28 and the scaled picture to center the image. After obtaining the pixel values of the new image, the values are adjusted to a range of 0 to 1, where 1 represents black and 0 represents white, using a formula to invert and normalize the values. As a result of these steps, the resulting image of the handwritten number is clear and accurate, making it easier to analyze and process. The original image (Figure 1b) was transformed into the MNIST-style image shown in Figure 1c using the process mentioned above. Figure 1a and our transformed digit Figure 1c are quite similar in appearance. However, our transformation process resulted in a sharper image compared to the original figure.



(**a**) (**b**) (**c**)

Figure 1. Preprocessing the *Custom* dataset. (**a**) MNIST image from the MNIST Dataset. (**b**) Original custom image. (**c**) Transformed into MNIST format.

4. Materials and Methods

This section mentions the computational resource used to conduct the experiments, the phases involved in performing OCR, and the architecture of the employed CNN-based DL model. The computational resource used to perform the experiments is mentioned in Table 2.

Table 2. Hardware resource specifications.

Item	Specification
CPU	Intel Core i3-7100 (-HT-MCP-) CPU @ 3.90 GHz
GPU	NVIDIA T4
CUDA	Version 10.1
OS	Ubuntu 17.10
Python	Version 3.10
Tensorflow	Version 2.12.0

4.1. Major Phases of Performing an OCR

The OCR workflow combines multiple phases, as described by [46].

1. Source data input—this phase involves using an external optical device, such as a scanner or camera, to generate images that contain relevant alphanumeric text or symbols.

2. Data preparation—covers various preprocessing operations, such as noise removal, to enhance source image quality and alignment of the image with the required standardized input format.
3. Partitioning—multiple characters in the image are split into individual items in this phase so that a recognition engine can process the partitioned characters appropriately.
4. Feature extraction—the partitioned characters are further "processed to extract different features." The characters are then 'recognized' based on their alignment with these features.
5. Classification—based on partitioning and feature extraction, the target image's features are mapped to several categories and classified to appropriate values.
6. Options—characters within source files can be classified using multiple techniques for pattern recognition using fundamental and statistical approaches.
7. Validation and improvement—upon the completion of initial classification exercises, it may be observed that the results are rarely perfect. This is especially true for handwritten content with multiple authors, multifaceted fonts, and intricate languages. Quantitative, conceptual, linguistic, and probabilistic remedial approaches and fine-tuning can be performed to improve the accuracy of OCR systems.

4.2. Architecture of CNN-Based DL Model

CNNs have become popular for image classification tasks due to their exceptional ability to extract features from images [47]. Our study utilized the MNIST dataset, which comprises 70,000 handwritten digits from 0 to 9 represented in a 28-by-28-pixel grayscale format. To further enhance the dataset, we added our data to the original MNIST dataset, resulting in an augmented dataset used to train the CNN. The trained model was then evaluated on our custom test data to assess its accuracy and generalizability. The process flowchart with *Custom* data is shown in Figure 2.

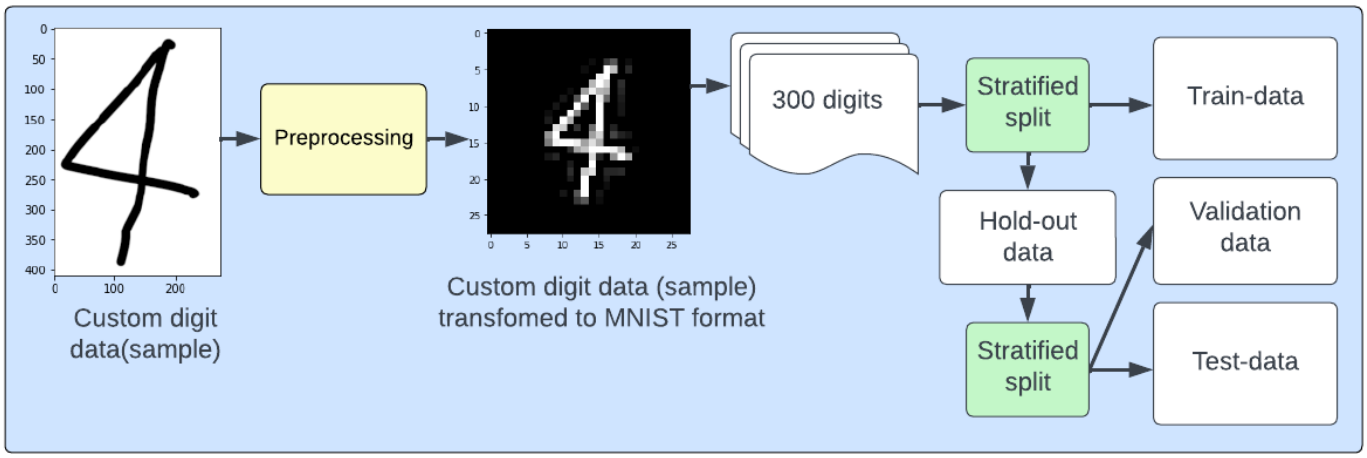


Figure 2. Process flowchart with *Custom* data.

For our experiments, we built a CNN Architecture, as shown in Figure 3, with the following parameters:

1. In our model, we stack multiple Conv2D layers to extract increasingly complex features from an input image.
2. Each Conv2D layer has parameters such as the number of filters, kernel size, padding, and activation function, which are tuned to optimize the performance of the network. The Conv2D layers are used to build feature maps from the data.
3. The kernel size of the filters is usually (3×3), and we use padding to ensure that the filters fit the input image. We also use the ReLU activation function to introduce nonlinearity in our model. Max-Pooling is used to reduce dimensionality. In Max-

Pooling, the output value is just the maximum input value in each patch (for example, the maximum pixel in a span of three pixels).

4. The next step is to flatten out the output from the last pooling layer because the input of the fully-connected layer is not a 2D vector but a 1D vector. To flatten means to convert a 2D matrix into a 1D matrix.
5. A fully-connected hidden layer is added to perform classification. The fully-connected layers combine the features extracted from the convolutional layers to create a model. Between fully-connected layers, dropout layers are added to remove specific neurons' contributions to the following layer while leaving the rest intact. Dropout layers are applied to reduce overfitting.
6. Finally, an activation function as softmax is used to classify the outputs as digits as 0, 1, 2 . . . 9.

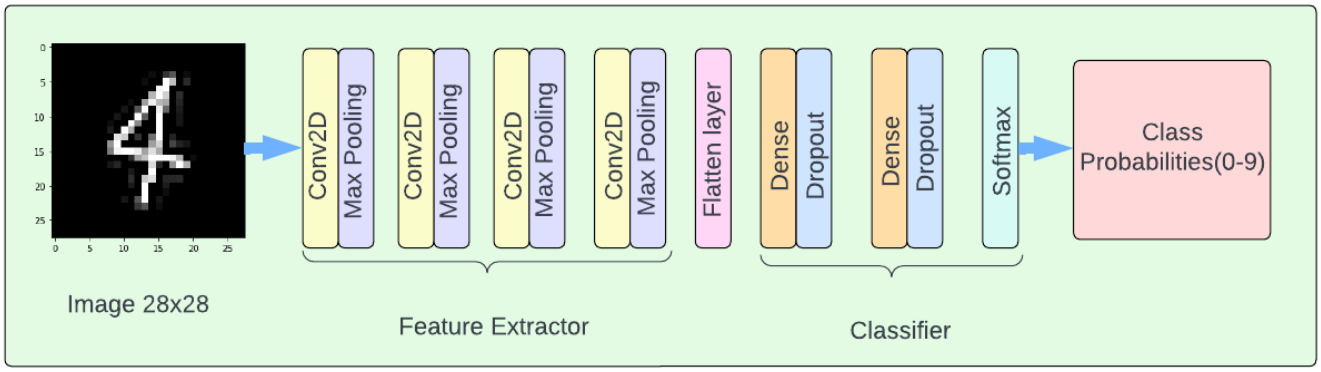


Figure 3. Model flowchart.

The model information is presented in Table 3.

Table 3. Model summary.

Layer (Type)	Output Shape	Number of Parameters
conv2d (Conv2D)	(None, 28, 28, 16)	160
max_pooling2d (MaxPooling2D)	(None, 14, 14, 16)	0
conv2d_1 (Conv2D)	(None, 14, 14, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 7, 7, 32)	0
conv2d_2 (Conv2D)	(None, 7, 7, 64)	18,496
max_pooling2d_2 (MaxPooling2D)	(None, 3, 3, 64)	0
conv2d_3 (Conv2D)	(None, 3, 3, 64)	36,928
max_pooling2d_3 (MaxPooling2D)	(None, 1, 1, 64)	0
flatten (Flatten)	(None, 64)	0
dense (Dense)	(None, 64)	4160
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 64)	4160
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 10)	650

5. Experiments and Results

In this section, we describe the details of our experiments, the process, and the outcomes. For ease of understanding, we have grouped each experiment and its result. We have used the Accuracy and F-1 score [48] metrics below to measure the performance of our employed classification models given by the following equations:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

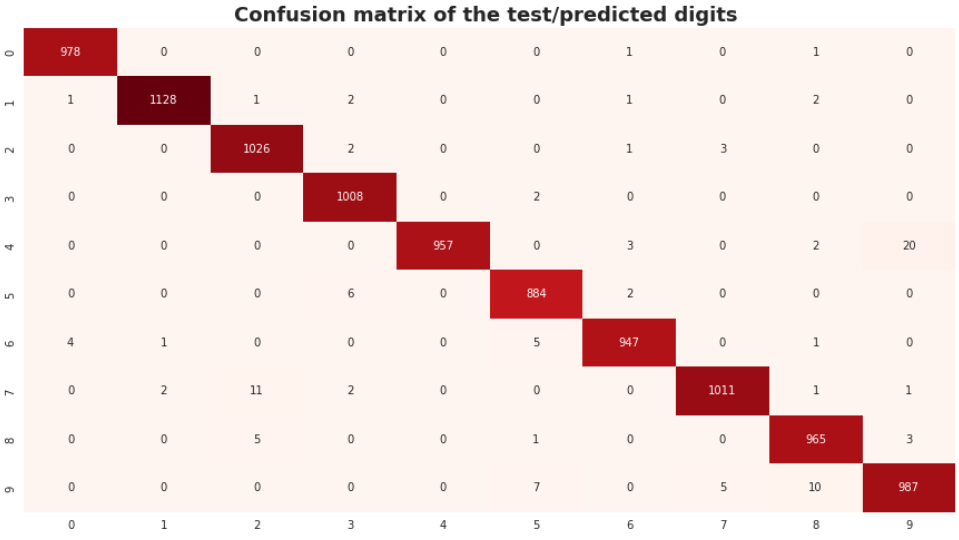
$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2}$$

where TP, FP, TN, and FN are True positive, False positive, True Negative, and False Negative, respectively.

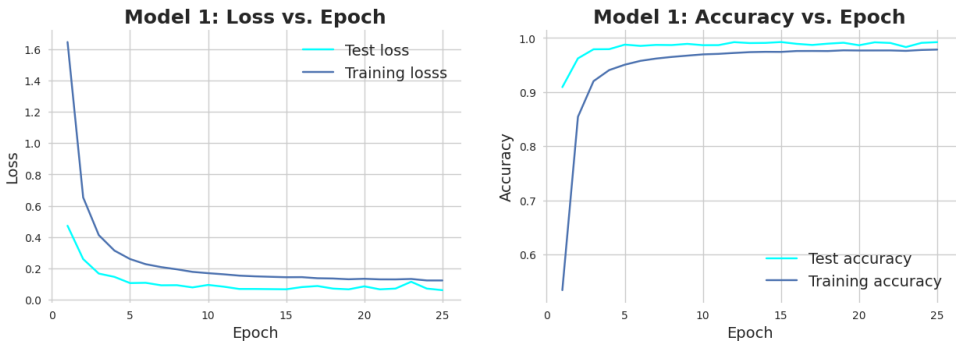
5.1. Experiment 1

In this experiment, we trained the CNN Model on 60,000 MNIST images and tested the model on 10,000 MNIST test data points. The output from this experiment is not shown in Table 4 because it was a standard experiment with MNIST data, and we only used it to validate our base process.

Results: Overall, the model performed well and generalized well to different data points, with very few mispredicted labels, as shown in Figure 4a. Furthermore, Figure 4b depicts the Loss vs. Epoch and Loss vs. Accuracy graph, demonstrating that the validation accuracy and validation loss were consistent with the training accuracy and training loss. The small gap between training and validation accuracy indicates that the model does not overfit and can generalize well. This model will serve as our baseline for comparison to other models. The model achieved an accuracy of 97.90% and a validation accuracy of 99.18%.



(**a**) Confusion matrix



(**b**) Graph

Figure 4. Experiment 1 outcome.

5.2. Experiment 2

In this experiment, the model was trained on a dataset of 60,000 images from the MNIST dataset and was then tested on our *Custom* test dataset consisting of 40 images. The MNIST test set, which consists of 10,000 images, was used as the validation dataset for this model.

Results: To evaluate the model's performance on our *Custom* dataset, we used the same model as before and tested it on 40 images. The results showed that five of the 40 digits were misclassified, as seen in Figure 5. Notably, the digit "2" had the lowest accuracy of classification.

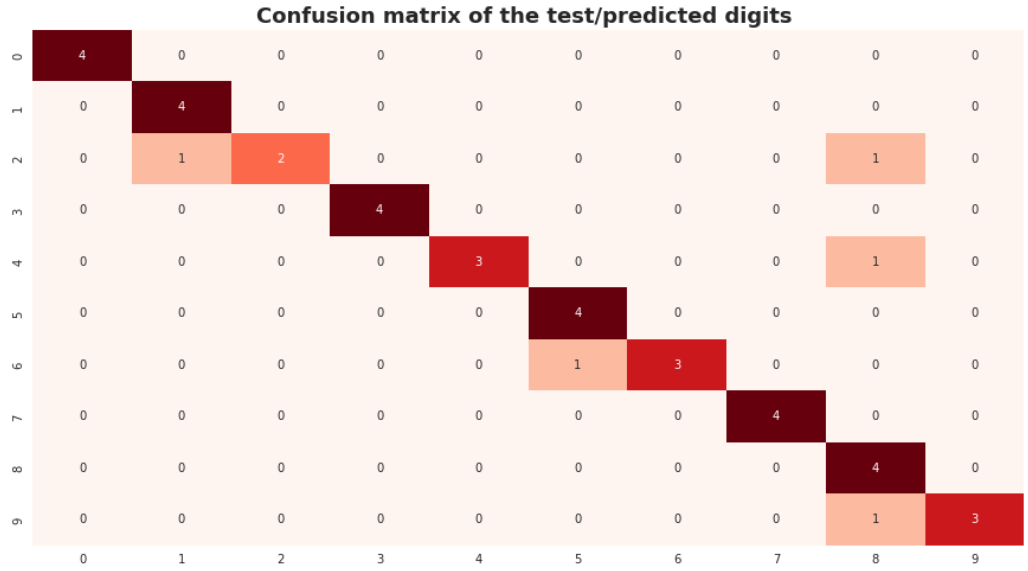
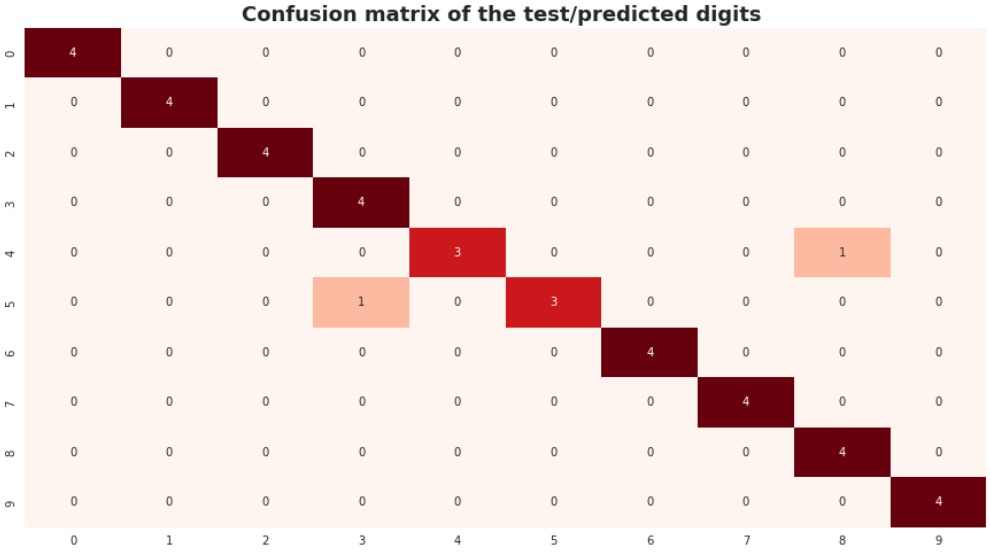


Figure 5. Confusion matrix of experiment 2.

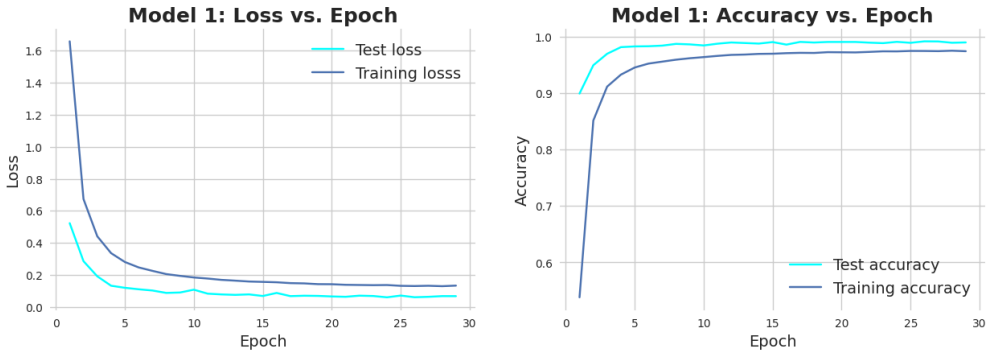
5.3. Experiment 3

The model was trained on a combined dataset comprising 60,000 images from the MNIST dataset and 240 images from our *Custom* dataset. To prevent overfitting and ensure that the model generalizes well, we used the MNIST test set, which contains 10,000 images, along with 20 digits from our *Custom* validation dataset, as the validation dataset for this model. After the training phase, we evaluated the model's performance on our *Custom* test dataset of 40 images.

Results: The evaluation of the model's performance on our *Custom* test dataset revealed that two of the 40 digits were misclassified, as shown in Figure 6a. Notably, the digits "4" and "5" were the ones that were misclassified. The graph shown in Figure 6b indicates that there is only a tiny gap between training and validation accuracy and loss, indicating that the model has maintained its generalizability and can fine-tune itself over our *Custom* dataset, achieving a low number of misclassification. The model achieved an accuracy of 97.07% on the combined dataset and a validation accuracy of 98.91%. These results suggest that the model performs well on both the MNIST and our *Custom* datasets, indicating that it has learned relevant common features across different datasets. These findings demonstrate the effectiveness of using a combined dataset and diverse validation data to improve the model's performance on new data points.



(**a**) Confusion matrix



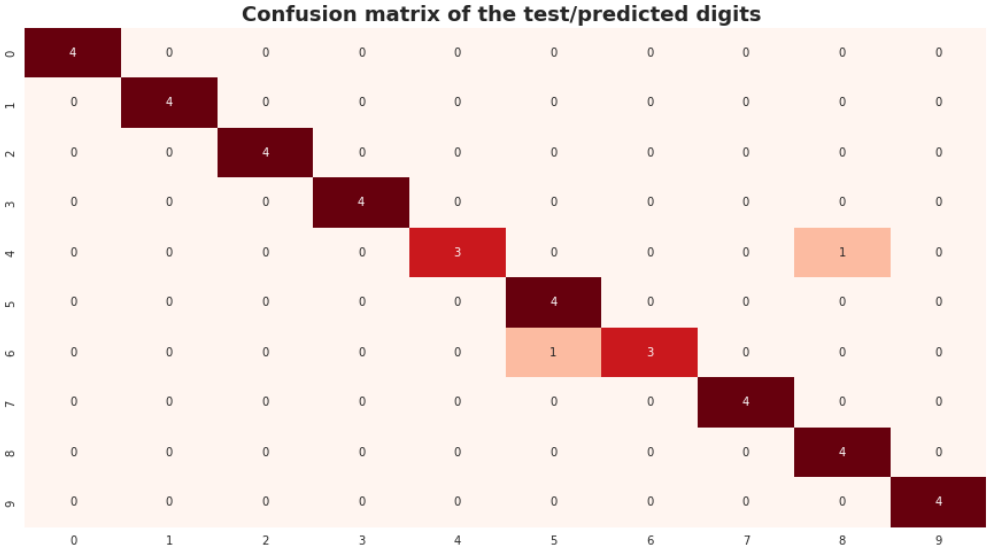
(**b**) Graph

Figure 6. Experiment 3 outcome.

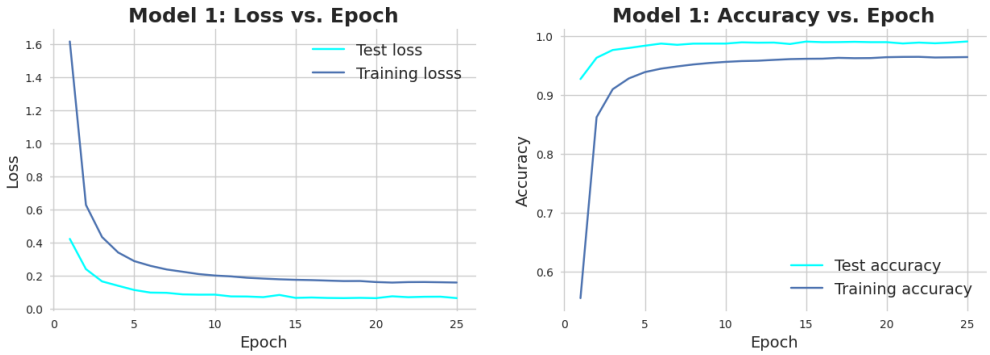
5.4. Experiment 4

This model was trained on a combined dataset of 60,000 images from the MNIST dataset, 240 images from our *Custom* dataset, and 720 rotated digits. The rotated digits were generated by applying a random rotation angle between −15 degrees to 15 degrees to our 240 digits. For validation, we used the MNIST test set, which contains 10,000 images, and added 20 digits from our *Custom* validation dataset. Following the training phase, we evaluated the model's performance on our *Custom* test dataset of 40 images.

Results: The evaluation of the model's performance on our *Custom* test dataset revealed that two of the 40 digits were misclassified, as shown in Figure 7a. As seen in the graph presented in Figure 7b, there is a slightly increased gap between the training and validation accuracy, which was expected due to the addition of our large *Custom* dataset. Although the model's generalizability decreased slightly, it still performed well on the *Custom* test dataset. The model achieved an accuracy of 96.30% on the combined dataset and a validation accuracy of 98.83%.



(**a**) Confusion matrix



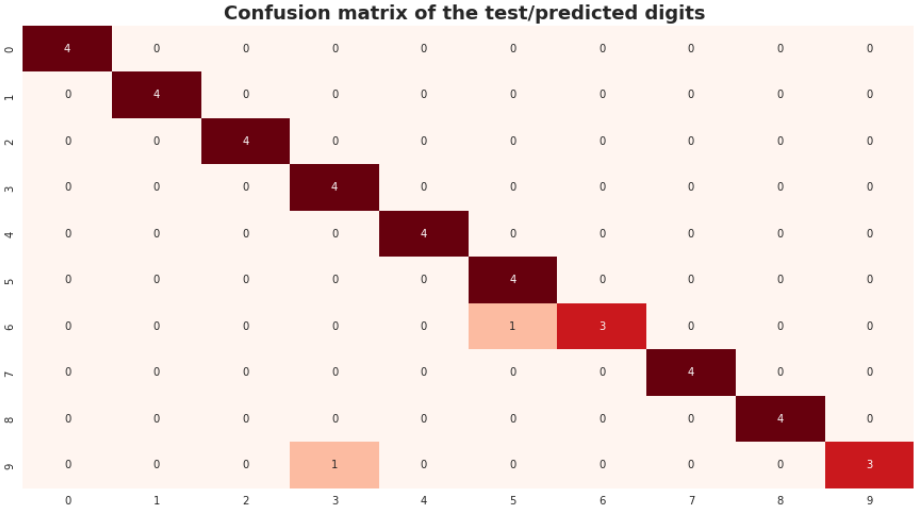
(**b**) Graph

Figure 7. Experiment 4 outcome.

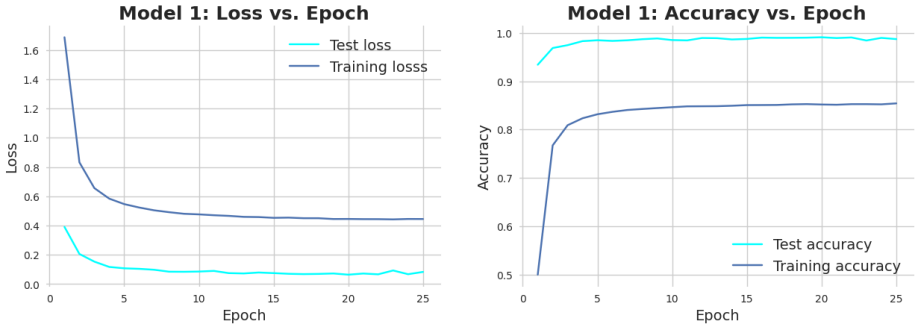
5.5. Experiment 5

This model was trained on a combined dataset of 60,000 images from the MNIST dataset, 240 images from our *Custom* dataset, and 9600 rotated digits. The rotated digits were generated by applying a random rotation angle between −15 degrees to 15 degrees to our 240 digits. For validation, we used the MNIST test set, which contains 10,000 images, and added 20 digits from our *Custom* validation dataset. Following the training phase, we evaluated the model's performance on our *Custom* test dataset of 40 images.

Results: The evaluation of the model's performance on our *Custom* test dataset revealed that two of the 40 digits were misclassified, as shown in Figure 8a. As seen in the graph presented in Figure 8b, there is a drastic increased gap between the training and validation accuracy, which was expected due to the addition of our larger *Custom* dataset. Although the model's generalizability decreased slightly, it still performed well on the *Custom* test dataset. The model achieved an accuracy of 85.30% on the combined dataset and a validation accuracy of 98.96%.



(**a**) Confusion matrix



(**b**) Graph

Figure 8. Experiment 5 outcome.

5.6. Experiment 6

The model used in this experiment is based on the transformer architecture and was first introduced in the paper by Li et al. (2021) [41]. It was trained on the IAM dataset [49], and the specific model used in this study is a pre-trained model fine-tuned on the MNIST dataset accessible via [50]. On the MNIST dataset, the model achieved an accuracy of 99.52%. However, when tested on our dataset, it misclassified two digits, as seen in Figure 9.

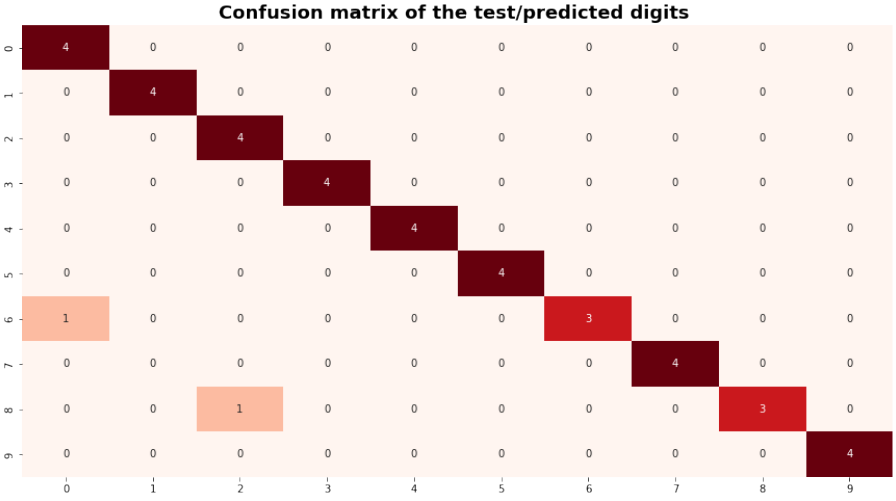


Figure 9. Confusion matrix of experiment 6.

5.7. Experimental Summary

We have provided the consolidated experimental outcomes of experiments 2 to 6 in Table 4.

Table 4. Summary of experimental results with 40 *Custom* test digits.

Model	Train Data	Validation Accuracy	Misclassified Digits	Test Accuracy
CNN	MNIST	99.18%	5	87.5%
CNN	MNIST + 240 custom digits	98.91%	2	95%
CNN	MNIST + 960 custom digits	98.83%	2	95%
CNN	MNIST + 9840 custom digits	98.96%	2	95%
TrOCR	(Pretrained model)	NA	2	95%

6. Discussion and Limitations

Computer vision and digital image processing are crucial in multimedia, artificial intelligence, and robotics. Image analysis includes segmentation, feature extraction, and classification techniques. Human–computer interaction can make things easier for users, and optimal results with less computation time and multilingual character segmentation and recognition are possible. A segmentation-free approach using Deep Neural Network (DNN) is also possible in OCR, and this work may bridge the knowledge gap in automatic interaction between human–system and system–system interactions [51].

We use a CNN to recognize an image containing a single character. Text of arbitrary length is a sequence of characters, and such problems are solved using Recurrent Neural Networks, and LSTM [52] is a popular form of RNN [53]. Modernization of the Tesseract tool involved code cleaning and adding a new LSTM model. The input image is processed in boxes, line by line, and fed into an LSTM model. Even after a lot of training, Tesseract performs better, but it still needs to be improved to work on handwritten text and weird fonts. Additionally, rotated or skewed text may cause the Tesseract to malfunction.

One major limitation we faced during the model training was the scarcity of data, as we only produced about 300 digits. This limited our ability to make significant improvements to the model, which was already trained on a much larger dataset of 60,000 digits. To address this limitation and make further progress, we would need to increase our dataset size to at least 10,000 digits.

Additionally, although we used a CNN model for our digit recognition task, we acknowledge that transformer models such as TrOCR have demonstrated comparable accuracy, as seen in the above table. With the potential to capture more global representations of images compared to CNN models, fine-tuning a TrOCR-based model could yield even better results. Furthermore, while our smart-OCR research concept is demonstrated using digits, the final production-level application will be expected to extend the concept to handwritten textual artifacts, full texts, and documents corpora, including historical texts [54,55].

7. Conclusions and Future Work

In this paper, we demonstrated the viability of a simple schema to develop an OCR mechanism for creating an adaptive framework for custom digit recognition models, and its logical implication of flexibility of OCR models to specific writing styles. Once developed further with state-of-the-art neural networks, such as transformers for computer vision, this approach can be applied to a variety of industry-level use cases. This could include solutions in healthcare where individual medical practitioners have different writing styles, and in fraud detection to match or distinguish handwriting styles with greater accuracy. Given the tremendous potential for adaptive OCR applications, it is advisable to move adaptive OCR research to the forefront. We are hopeful that such adaptive OCR solutions would be an important part of the rapidly advancing artificial intelligence ecosystems worldwide. Currently, most NLP research and practice use machine-readable

typed data and associated textual data distributions [56]. It would be very useful to develop OCR solutions for handwritten documents to create a seamless integration with NLP solutions, such as sentiment analysis and NLP-based socioeconomic modeling [57–59]. OCR is a mature discipline with industry-level solutions for identifying and 'reading' images of machine-printed text. However, due to the high degree of variations, OCR for handwriting recognition needs additional work. Based on the early-stage success of TrOCR, we believe there is significant potential for improving OCR solutions for handwritten text with transformer-based applications. We intend to explore the potential of fine-tuning and limited shot learning with pretrained transformer models to cater to user-specific digit recognition needs. Furthermore, the additional use of AI methods and tools to mimic human intelligence's capability to identify text in varying colors, mixed sizes, and styles and other complex forms holds great promise. As an applied direction for future research, it is possible to use OCR methods to generate data for domains such as heritage culture and preservation from images of historical texts. One of the most important research areas to build upon would be the capability of an OCR application to be 'flexible' with custom handwriting styles. Incorporating the on-demand flexibility of OCR models would be a powerful way to advance the effectiveness of OCR with *Custom* data and variations in font styles. We also aim to incorporate world knowledge in the form of triples to address the domain adaptation challenges in identifying subtle sentiments [60].

Author Contributions: Conceptualization, P.H.J., V.K., J.S., S.S., A.M. and R.A.; methodology, P.H.J., V.K., J.S., S.S., A.M. and R.A.; software, P.H.J. and J.S.; formal analysis, P.H.J., V.K. and J.S.; investigation, P.H.J., V.K. and J.S.; resources, P.H.J., J.S. and A.M.; data curation, P.H.J., J.S. and A.M.; writing—original draft preparation, P.H.J., V.K. and J.S.; writing—review and editing, P.H.J., V.K., J.S., S.S. A.M. and R.A.; visualization, P.H.J., V.K. and J.S.; supervision, V.K. and J.S.; project administration, J.S.; funding acquisition, J.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the NJ State Policy Lab (https://policylab.rutgers.edu/ (accessed on 11 April 2023)) and the Public Informatics program at Bloustein School, Rutgers University (https://bloustein.rutgers.edu/graduate/public-informatics/mpi/ (accessed on 11 April 2023)).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The *MNIST* dataset is available at (https://archive-beta.ics.uci.edu/dataset/683/mnist+database+of+handwritten+digits (accessed on 11 April 2023)) and our *Custom* dataset is available at the Github repository (https://github.com/ay7n/OCR-RUCILDigits-4 (accessed on 11 April 2023)).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
CNN	Convolution Neural Network
DL	Deep Learning
ML	Machine Learning
OCR	Optical Character Recognition
LSTM	Long Short-Term Memory
NLP	Natural Language processing
RNN	Recurrent Neural Network
NER	Named-Entity Recognition
ReLU	Rectified Linear Unit
LSTM-R	Long Short-Term Memory plus Relation-aware pointer network
HMM	Hidden Markov Models
DNN	Deep Neural Network

References

1. Samuel, J.; Kashyap, R.; Samuel, Y.; Pelaez, A. Adaptive cognitive fit: Artificial intelligence augmented management of information facets and representations. *Int. J. Inf. Manag.* **2022**, *65*, 102505. [CrossRef]
2. Thorat, C.; Bhat, A.; Sawant, P.; Bartakke, I.; Shirsath, S. A detailed review on text extraction using optical character recognition. In *ICT Analysis and Applications*; Springer: Singapore, 2022; pp. 719–728.
3. Singh, S. Optical character recognition techniques: A survey. *J. Emerg. Trends Comput. Inf. Sci.* **2013**, *4*, 2009–2015.
4. Samuel, J. A call for proactive policies for informatics and artificial intelligence technologies. *Scholars Strategy Network*, 19 December 2021.
5. Srivastava, N.; Singh, S. Netra: Smart Hand Gloves Comprises Obstacle Detection, Object Identification & OCR Text to Speech Converter for Blinds. In Proceedings of the 2018 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), Gorakhpur, India, 2–4 November 2018; pp. 1–4.
6. Januszewski, A.; Kujawski, J.; Buchalska-Sugajska, N. Benefits of and obstacles to RPA implementation in accounting firms. *Procedia Comput. Sci.* **2021**, *192*, 4672–4680. [CrossRef]
7. Samuel, J.; Rahman, M.M.; Ali, G.M.N.; Samuel, Y.; Pelaez, A.; Chong, P.H.J.; Yakubov, M. Feeling Positive About Reopening? New Normal Scenarios From COVID-19 US Reopen Sentiment Analytics. *IEEE Access* **2020**, *8*, 142173–142190. [CrossRef]
8. Ali, G.M.N.; Rahman, M.M.; Hossain, M.A.; Rahman, M.S.; Paul, K.C.; Thill, J.C.; Samuel, J. Public perceptions of COVID-19 vaccines: Policy implications from US spatiotemporal sentiment analytics. *Healthcare* **2021**, *9*, 1110. [CrossRef]
9. Manwatkar, P.M.; Yadav, S.H. Text recognition from images. In Proceedings of the 2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, India, 19–20 March 2015; pp. 1–6.
10. Srihari, S.N.; Shekhawat, A.; Lam, S.W. Optical character recognition (OCR). In *Encyclopedia of Computer Science*; Wiley: London, UK, 2003; pp. 1326–1333.
11. Srihari, S.N.; Kuebert, E.J. Integration of hand-written address interpretation technology into the united states postal service remote computer reader system. In Proceedings of the Fourth International Conference on Document Analysis and Recognition, Ulm, Germany, 18–20 August 1997; Volume 2, pp. 892–896.
12. Kornai, A. An experimental HMM-based postal ocr system. In Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing, Munich, Germany, 21–24 April 1997; Volume 4, pp. 3177–3180.
13. Ifhaam, M.; Jayalal, S. Sinhala handwritten postal address recognition for postal sorting. In Proceedings of the 2019 International Research Conference on Smart Computing and Systems Engineering (SCSE), Colombo, Sri Lanka, 28 March 2019; pp. 134–141.
14. Patel, M.; Reddy, S.L. An impact of grid based approach in offline handwritten Kannada word recognition. In Proceedings of the 2014 International Conference on Contemporary Computing and Informatics (IC3I), Mysore, India, 27–29 November 2014; pp. 630–633.
15. Nagy, G. Disruptive developments in document recognition. *Pattern Recognit. Lett.* **2016**, *79*, 106–112. [CrossRef]
16. Faizullah, S.; Ayub, M.S.; Hussain, S.; Khan, M.A. A Survey of OCR in Arabic Language: Applications, Techniques, and Challenges. *Appl. Sci.* **2023**, *13*, 4584. [CrossRef]
17. Al-Hadhrami, A.A.; Allen, M.; Moffatt, C.; Jones, A.E. National characteristics and variation in Arabic handwriting. *Forensic Sci. Int.* **2015**, *247*, 89–96. [CrossRef]
18. Bhagyasree, P.; James, A.; Saravanan, C. A proposed framework for recognition of handwritten cursive english characters using DAG-CNN. In Proceedings of the 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT), Chennai, India, 25–26 April 2019; pp. 1–4.
19. Bhunia, A.K.; Khan, S.; Cholakkal, H.; Anwer, R.M.; Khan, F.S.; Shah, M. Handwriting transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 1086–1094.
20. Van Strien, D.; Beelen, K.; Ardanuy, M.C.; Hosseini, K.; McGillivray, B.; Colavizza, G. Assessing the impact of OCR quality on downstream NLP tasks. In Proceedings of the 12th International Conference on Agents and Artificial Intelligence, Valletta, Malta, 22–24 February 2020.
21. Wang, J.; Tang, J.; Yang, M.; Bai, X.; Luo, J. Improving OCR-based image captioning by incorporating geometrical relationship. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Virtual, 19–25 June 2021; pp. 1306–1315.
22. Nguyen, T.T.H.; Jatowt, A.; Coustaty, M.; Doucet, A. Survey of post-OCR processing approaches. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–37. [CrossRef]
23. Vijayarani, S.; Sakila, A. Performance comparison of OCR tools. *Int. J. UbiComp (IJU)* **2015**, *6*, 19–30.
24. Suissa, O.; Elmalech, A.; Zhitomirsky-Geffet, M. *Optimizing the Neural Network Training for OCR Error Correction of Historical Hebrew Texts*; iConference 2020 Proceedings; iSchools Inc.: Grandville, MI, USA, 2020; pp. 1–10.
25. Hegghammer, T. OCR with Tesseract, Amazon Textract, and Google Document AI: A benchmarking experiment. *J. Comput. Soc. Sci.* **2022**, *5*, 861–882. [CrossRef]
26. Smith, R. An overview of the Tesseract OCR engine. In Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Curitiba, Brazil, 23–26 September 2007; Volume 2, pp. 629–633.
27. Ramiah, S.; Liong, T.Y.; Jayabalan, M. Detecting text based image with optical character recognition for English translation and speech using Android. In Proceedings of the 2015 IEEE Student Conference on Research and Development (SCOReD), Kuala Lumpur, Malaysia, 13–14 December 2015; pp. 272–277.

28. Breuel, T.M. The OCRopus open source OCR system. *Proc. Doc. Recognit. Retr. SPIE* **2008**, *6815*, 120–134.
29. Kittinaradorn, R. EasyOCR. 2020. Available online: https://github.com/JaidedAI/EasyOCR/tree/master (accessed on 11 April 2023).
30. Awalgaonkar, N.; Bartakke, P.; Chaugule, R. Automatic license plate recognition system using ssd. In Proceedings of the 2021 International Symposium of Asian Control Association on Intelligent Robotics and Industrial Automation (IRIA), Goa, India, 20–22 September 2021; pp. 394–399.
31. Troller, M. Practical OCR System Based on State of Art Neural Networks. Bachelor's Thesis, Czech Technical University in Prague, Dejvice, Czech Republic, 2017.
32. Alrasheed, N.; Prasanna, S.; Rowland, R.; Rao, P.; Grieco, V.; Wasserman, M. Evaluation of Deep Learning Techniques for Content Extraction in Spanish Colonial Notary Records. In Proceedings of the 3rd Workshop on Structuring and Understanding of Multimedia heritAge Contents, Virtual, 20 October 2021; pp. 23–30.
33. Chen, Y.; Yang, J. Research on scene text recognition algorithm basedon improved CRNN. In Proceedings of the 2020 4th International Conference on Digital Signal Processing, Chengdu, China, 19–21 June 2020; pp. 107–111.
34. Shi, B.; Bai, X.; Yao, C. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 2298–2304. [CrossRef]
35. Baek, Y.; Lee, B.; Han, D.; Yun, S.; Lee, H. Character region awareness for text detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 9365–9374.
36. FenniakM. PyPDF2·PyPI. Available online: https://pypi.org/project/PyPDF2/ (accessed on 4 April 2023).
37. Kekare, A.; Jachak, A.; Gosavi, A.; Hanwate, P. Techniques for Detecting and Extracting Tabular Data from PDFs and Scanned Documents: A Survey. *Tabula* **2020**, *7*, 415–417.
38. Mindee. docTR: Document Text Recognition. Available online: https://github.com/mindee/doctr (accessed on 4 April 2023).
39. Batra, P.; Phalnikar, N.; Kurmi, D.; Tembhurne, J.; Sahare, P.; Diwan, T. OCR-MRD: Performance Analysis of Different Optical Character Recognition Engines for Medical Report Digitization. 2023. Available online: https://www.researchsquare.com/article/rs-2513255/v1 (accessed on 4 April 2023).
40. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All You Need. *arXiv* **2017**. arXiv:1706.03762.
41. Li, M.; Lv, T.; Cui, L.; Lu, Y.; Florencio, D.; Zhang, C.; Li, Z.; Wei, F. Trocr: Transformer-based optical character recognition with pre-trained models. *arXiv* **2021**. arXiv:2109.10282
42. Smith, R.W. The Extraction and Recognition of Text from Multimedia Document Images. Ph.D. Thesis, University of Bristol, Bristol, UK, 1987.
43. Ai, J. EasyOCR. Available online: https://github.com/JaidedAI/EasyOCR (accessed on 4 April 2023).
44. Le Cun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
45. Lab, R. OCR-RUCILDigits-4. 2022. Available online: https://github.com/ay7n/OCR-RUCILDigits-4 (accessed on 4 April 2023).
46. Eikvil, L. Optical Character Recognition. 1993. Available online: http://home.nr.no/~eikvil/OCR.pdf (accessed on 4 April 2023).
47. Agrawal, A.K.; Shrivas, A.; kumar Awasthi, V. A Robust model for handwritten digit recognition using machine and deep learning technique. In Proceedings of the 2021 2nd International Conference for Emerging Technology (INCET), Belagavi, India, 21–23 May 2021; pp. 1–4.
48. Kumar, V.; Recupero, D.R.; Riboni, D.; Helaoui, R. Ensembling Classical Machine Learning and Deep Learning Approaches for Morbidity Identification From Clinical Notes. *IEEE Access* **2021**, *9*, 7107–7126. [CrossRef]
49. Cheng, L.; Bing, L.; He, R.; Yu, Q.; Zhang, Y.; Si, L. IAM: A Comprehensive and Large-Scale Dataset for Integrated Argument Mining Tasks. *arXiv* **2022**, arXiv:2203.12257
50. Aico. TROCR Digit. Available online: https://huggingface.co/spaces/aico/TrOCR-digit (accessed on 4 April 2023).
51. Karthick, K.; Ravindrakumar, K.; Francis, R.; Ilankannan, S. Steps involved in text recognition and recent research in OCR: A study. *Int. J. Recent Technol. Eng.* **2019**, *8*, 2277–3878.
52. Yu, Y.; Si, X.; Hu, C.; Zhang, J. A review of recurrent neural networks: LSTM cells and network architectures. *Neural Comput.* **2019**, *31*, 1235–1270. [CrossRef] [PubMed]
53. Williams, G.; Baxter, R.; He, H.; Hawkins, S.; Gu, L. A comparative study of RNN for outlier detection in data mining. In Proceedings of the 2002 IEEE International Conference on Data Mining, Maebashi City, Japan, 9–12 December 2002; pp. 709–712.
54. Mueller-Gastell, J.; Sena, M.; Tan, C.Z. A Multi-Digit OCR System for Historical Records (Computer Vision). Available online: http://cs230.stanford.edu/projects_spring_2020/reports/38792124.pdf (accessed on 4 April 2023).
55. Goodfellow, I.J.; Bulatov, Y.; Ibarz, J.; Arnoud, S.; Shet, V. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv* **2013**, arXiv:1312.6082
56. Samuel, J.; Palle, R.; Soares, E.C. Textual Data Distributions: Kullback Leibler Textual Distributions Contrasts on GPT-2 Generated Texts, with Supervised, Unsupervised Learning on Vaccine & Market Topics & Sentiment. *arXiv* **2022**, arXiv:2107.02025.
57. Rahman, M.M.; Ali, G.M.N.; Li, X.J.; Samuel, J.; Paul, K.C.; Chong, P.H.; Yakubov, M. Socioeconomic factors analysis for COVID-19 US reopening sentiment with Twitter and census data. *Heliyon* **2021**, *7*, e06200. [CrossRef]
58. Samuel, J.; Ali, G.G.M.N.; Rahman, M.M.; Esawi, E.; Samuel, Y. COVID-19 public sentiment insights and machine learning for tweets classification. *Information* **2020**, *11*, 314. [CrossRef]

59. Bhandari, A.; Kumar, V.; Thien Huong, P.T.; Thanh, D.N. Sentiment analysis of COVID-19 tweets: Leveraging stacked word embedding representation for identifying distinct classes within a sentiment. In *Artificial Intelligence in Data and Big Data Processing, Proceedings of ICABDE 2021, Ho Chi Minh City, Vietnam, 18–19 December 2022*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 341–352.
60. Kumar, V.; Reforgiato Recupero, D.; Helaoui, R.; Riboni, D. K-LM: Knowledge Augmenting in Language Models Within the Scholarly Domain. *IEEE Access* **2022**, *10*, 91802–91815. [CrossRef]