Journal of
Sensor and Actuator Networks

MDPI

Article

Enhancing Optical Character Recognition on Images with Mixed Text Using Semantic Segmentation

Shruti Patil [1,*], Vijayakumar Varadarajan [2,3,4,*], Supriya Mahadevkar [5], Rohan Athawade [6], Lakhan Maheshwari [6], Shrushti Kumbhare [6], Yash Garg [6], Deepak Dharrao [6], Pooja Kamat [6] and Ketan Kotecha [1]

1 Symbiosis Centre for Applied Artificial Intelligence (SCAAI), Symbiosis Institute of Technology, Symbiosis International (Deemed University), Pune 412115, India
2 School of Computer Science and Engineering, The University of New South Wales, Sydney, NSW 2052, Australia
3 School of NUOVOS, Ajeenkya D Y Patil University, Pune 412105, India
4 Swiss School of Business and Management, SSBM Geneva, 1213 Geneva, Switzerland
5 Ph.D Research Scholar, Symbiosis Institute of Technology, Symbiosis International (Deemed University), Pune 412115, India
6 Department of Computer Science Engineering, Symbiosis Institute of Technology, Symbiosis International (Deemed University), Pune 412115, India
* Correspondence: shruti.patil@sitpune.edu.in (S.P.); vijayakumar.varadarajan@gmail.com (V.V.)

Abstract: Optical Character Recognition has made large strides in the field of recognizing printed and properly formatted text. However, the effort attributed to developing systems that are able to reliably apply OCR to both printed as well as handwritten text simultaneously, such as hand-filled forms, is lackadaisical. As Machine printed/typed text follows specific formats and fonts while handwritten texts are variable and non-uniform, it is very hard to classify and recognize using traditional OCR only. A pre-processing methodology employing semantic segmentation to identify, segment and crop boxes containing relevant text on a given image in order to improve the results of conventional online-available OCR engines is proposed here. In this paper, the authors have also provided a comparison of popular OCR engines like Microsoft Cognitive Services, Google Cloud Vision and AWS recognitions. We have proposed a pixel-wise classification technique to accurately identify the area of an image containing relevant text, to feed them to a conventional OCR engine in the hopes of improving the quality of the output. The proposed methodology also supports the digitization of mixed typed text documents with amended performance. The experimental study shows that the proposed pipeline architecture provides reliable and quality inputs through complex image preprocessing to Conventional OCR, which results in better accuracy and improved performance.

Keywords: image pre-processing; handwritten words; machine-printed words; semantic segmentation; optical character recognition; deep learning; U-Net; handwritten documents; computer vision; Google Vision

Citation: Patil, S.; Varadarajan, V.; Mahadevkar, S.; Athawade, R.; Maheshwari, L.; Kumbhare, S.; Garg, Y.; Dharrao, D.; Kamat, P.; Kotecha, K. Enhancing Optical Character Recognition on Images with Mixed Text Using Semantic Segmentation. *J. Sens. Actuator Netw.* **2022**, *11*, 63. https://doi.org/10.3390/jsan11040063

Academic Editors: Lei Shu, Adnan Al-Anbuky, Stefan Fischer, Joel J. P. C. Rodrigues and Mário Alves

Received: 23 July 2022
Accepted: 15 September 2022
Published: 3 October 2022

1. Introduction

1.1. Machine Printed and Handwritten Text

Several applications of conventional optical character recognition systems fall into the category of digitizing some form of printed or handwritten text. On a high-level basis, invoice imaging, legal industry, banking, healthcare, captcha, institutional repositories/digital libraries, optical music recognition, automatic number recognition and handwriting recognition are a few of the domains in which the use of OCR is prevalent [1], and in some cases, pivotal. Further study of some of these domains reveals that out of those mentioned, invoice imaging [2], banking, healthcare, automatic number recognition and automatic handwriting recognition are those that were found to apply to data that may

or may not include handwritten text, and in the cases that do, almost certainly alongside printed text. Concerning this observation, a distinction between the kinds of text on which optical character recognition is used can be made, that is:

- Machine printed or formatted text: typed text that follows a given font and format, most commonly found in printed images. This type of text displays characteristics such as font, uniform size, varying styles and colors, bolding, italics, etc.
- Handwritten text: text that a human writes. This type of text is variable, non-replicable non-uniform. It cannot be categorized into different styles and fonts that are minute enough to capture all of the features that distinguish handwriting from one another.

Many images, regardless of origin, machine-printed, and handwritten text may coexist, giving way to urgent issues in the recognition pipeline [3]. This, along with the fact that documents can be understood as physical processes and conditions that occur to a given text [4], poses a challenge in the seamless digitization of noisy documents wherein the two types of text defined above coexist. One such instance of coexistence of machine-printed and handwritten text that is undeniably abundant forms [5].

As common as they are, forms are often filled out in handwriting, where attributes of forms such as titles along with field names and labels are all machine-printed text. Because forms are an ideal manifestation of the coexistence of handwritten and printed text, the methodology proposed in this paper aims to optimize the recognition of text in a form. Many handwritten documents have scrambled language, sloppy handwriting, and various writing styles, making it difficult to extract the right information from them.

Handwriting recognition systems get more challenging when there is noise, such as a bounding box on a form page. These are the parameters that affect recognition accuracy [6].

1.2. An Overview of Optical Character Recognition

Optical character recognition (OCR) is the classification of optical patterns in a digital image corresponding to alphanumeric or other characters [7]. Different methods of performing character recognition via what is now referred to as optical character recognition have been prevalent since as early as the 1900s. This means that OCR and many of the techniques used to achieve it are in no way adolescent, and so, modern techniques may be leveraged [8]. OCR is used to read the text included in an image, typically a handwritten or printed document that has been scanned. OCR can also classify different document formats automatically and arrange them in accordance with predetermined guidelines. Organizing and monitoring invoices, for instance, according to the type of seller or product.

The development of modern optical character recognition systems finds its roots in the requirement of digitizing text found in images. Initial testing and development of OCR methods were typically performed using images with machine-printed texts [9], and so for many of the years wherein research on OCR was at its prime, most methodologies were optimized to recognize printed text. It was not until the year 2000 that OCR oriented towards recognizing handwritten text rose in importance, and techniques that included Artificial Neural Networks (and their expansive domains, such as Deep Neural Networks) were utilized in order to supercharge OCR methods [10,11]

To distinguish a character or word from the backdrop in an image, pre-processing is required. It contains:

Binarization: It reduces an image to pixels that are only black and white. A threshold value is fixed to do this conversion. A white pixel is thought to exist if the value is greater; otherwise, a black pixel.

- Noise reduction: This technique clears the image of all extraneous dots and patches.
- Text alignment issues: Some text may be skewed. Skew correction aids with text alignment.
- Slant Removal: This technique is used to eliminate the slant from the text that may appear in some images in the dataset [12].

An exhaustive internet search reveals that OCR systems that allow the recognition of both machine-printed and handwritten text are difficult to find, and if at all, are not very optimized to do the same. While it is not denied that OCR engines that recognize both handwritten as well as formatted/machine-generated texts may exist, this paper proposes a methodology that allows the optimization of one such OCR engine to provide better-recognized outputs, that, speaking on a high-level understanding basis, can only be described as applying deep learning-enabled semantic segmentation technique to obtain blocks of mixed type information and subsequently feeding those blocks individually to an OCR engine to digitize the text.

Table 1 outlines some conclusive results of tests conducted on some popular online OCR engines. It can be easily deduced that Google Cloud Vision outperforms the rest due to its superior recognition capabilities.

Table 1. Comparison of Different Popular OCR Engines.

Provider	No. Correct	No. Incorrect	No. No Result	Precision	Recall
Microsoft Cognitive Services	142	76	283	65%	44%
Google Cloud Vision	322	80	99	80%	80%
AWS Recognition	58	213	230	21%	54%

1.3. An Overview of Semantic Segmentation Using Deep Learning

Semantic segmentation has infamously been considered the grandest challenge to achieve and implement in the recent rise of computer vision. It remains one of the most popular methods of segmentation due to the distinctive characteristic it possesses that sets it apart from other image classification methods: it does not require prior visual concepts or knowledge of the objects being classified. While object classification classifies objects for which it has labels beforehand, dogs and cats, an ideal image segmentation algorithm also segments unknown objects [12].

Semantic segmentation can be thought of as a method of identifying objects within an image by classifying each pixel as whether it belongs to that class (or multiple classes) in the image based on its semantic properties. Due to this similar vast semantic segmenting prowess that algorithms like U-Nets [13] and FCNs [14] provide, they could be instrumental in improving OCR results.

To summarize the contribution of this work is as follows:

(1) This research aims to explore an elaborate pre-processing methodology to enhance the recognition of images containing mixed-type text (i.e., handwritten and machine-printed text).
(2) The work gives background study, limitations and the new approach for data digitization.
(3) The proposed pixel-wise classification technique to accurately identify the area of an image containing relevant text, to feed them to a conventional OCR engine in the hopes of improving the quality of the output.
(4) The proposed methodology also supports the digitization of mixed typed text documents with amended performance. Experimental study shows that the proposed pipeline architecture provides reliable and quality inputs through complex image preprocessing to Conventional OCR, which results in better accuracy and improved performance.

2. Related Work

The idea of using segmentation as a pre-processing step to enhance OCR is not a new one; the review conducted in Page Segmentation in OCR System—A Review [15] mentions that segmentation techniques used in OCR Systems can be aptly summarized as top-down, bottom-up, and hybrid approaches. These approaches employ complex, mathematically-inclined algorithms to find and segment lines, words, and characters.

While results from approaches such as these are staggering, they cannot aid in marginally enhancing the results obtained by OCR while considering images with a combination of machine-printed and handwritten text. Chargrid-OCR: End-to-end Trainable Optical Character Recognition through Semantic Segmentation and Object Detection [16] emphasizes a concept that is not dissimilar to the one emphasized in this paper: converting the task of OCR to a task of object detection and segmentation. The Chargrid-OCR architecture is a custom-made one that allows for the character-wise semantic segmentation and classification of text in images. It disregards the need for image pre-processing and enhancement techniques as its encoder-decoder architecture learns directly from the raw pixel input that images provide. Apart from the fully custom architecture that Chargrid-OCR uses, it is essentially a method to perform OCR itself and much less to enhance the results of existing OCR systems. This is where the similarity between our methodology and Chargrid-OCR halts.

Authors [17] proposes another deep-learning approach, but with specific emphasis on the recognition of handwritten text. A word-level segmentation is performed, along with pre-processing steps to allow the smooth final classification of segmented words. Authors [17] equires a large corpus of words for the accurate classification of the segmented handwritten words. Like [16,17] proposes a method of performing optical character recognition itself. It seems that the domain of deep learning in OCR, regardless of the type of text, almost always defaults to the classic coupling of segmentation followed by classification on a certain level. Dedicated research towards pre-processing techniques for enhancing handwritten text recognition is lacking; Ref. [18] suggests and discusses pre-processing techniques such as underline removal and skew correction as priors to recognition; however, upon closer observation, it is found that a number of these techniques (inclusive of those listed by [18]) are not properly distinguishable from those used for the conventional recognition of machine-printed text [19,20].

Table 2 describes the comparison of the U-Net and Mask RCNN architectural models with respect to the layers used and the advantages and disadvantages of each. Table 3 shows the summary, layers used, and the advantages and disadvantages of object detection and semantic segmentation. Finding instances of items of a specific class inside an image is the challenge of object detection. Modern techniques can be divided into two primary categories: one-stage approaches and two-stage methods. Examples of one-stage algorithms that emphasize inference speed are YOLO, SSD, and RetinaNet. Whereas, Semantic image segmentation aims to assign a class of what is being represented to each pixel of an image. The term "dense prediction" refers to the fact that we are making predictions for each pixel in the image.

Table 2. Comparison of U-Net and Mask RCNN.

Architecture	Model	Layers	Advantage	Disadvantage
U-Net	Built upon the Fully Convolutional Network (FCN) which is modified in a manner that yields better segmentation in medical imaging.	Separated into 3 parts: The contracting/ down sampling path bottleneck The expanding/up sampling path	Combines location information with contextual information to obtain general information which is necessary to predict a good segmentation map.	Instance segmentation is difficult because the output is a binary segmentation mask for the whole input image.
Mask R-CNN	Built on top of Faster R-CNN. So, in addition to the class label and bounding box coordinates for each object, it will also return the object mask.	Combines the 2 networks—Faster RCNN and FCN in one mega architecture.	Efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance.	Although it does not cost a lot to modify the Mask RCNN, it still has the limitation that the mask generation cannot completely cover the edge of the target.

Table 3. Comparison between Object Detection and Semantic Segmentation.

Technique	Summary	Layers	Advantage	Disadvantage
Object detection	Classifies patches of an image into different object classes and creates a bounding box around that object.	There are 2 classes of object classification-object bounding boxes and non-object bounding boxes.	Acts as a combination of image classification and object localization.	Bounding boxes are always rectangular, so it does not help in determining the shape of the object if it contains some curvature part. It cannot accurately estimate measurements such as the area or perimeter of an object from the image.
Semantic segmentation	Gives a pixel-level classification in an image, that is, classification of pixels into its corresponding classes.	Each pixel is labeled with the class of the object (person, dog...) and non-objects (tree, road...).	A further extension of object detection.	This technique is more granular than bounding box generation, hence it helps in determining the shape of each object present in the image.

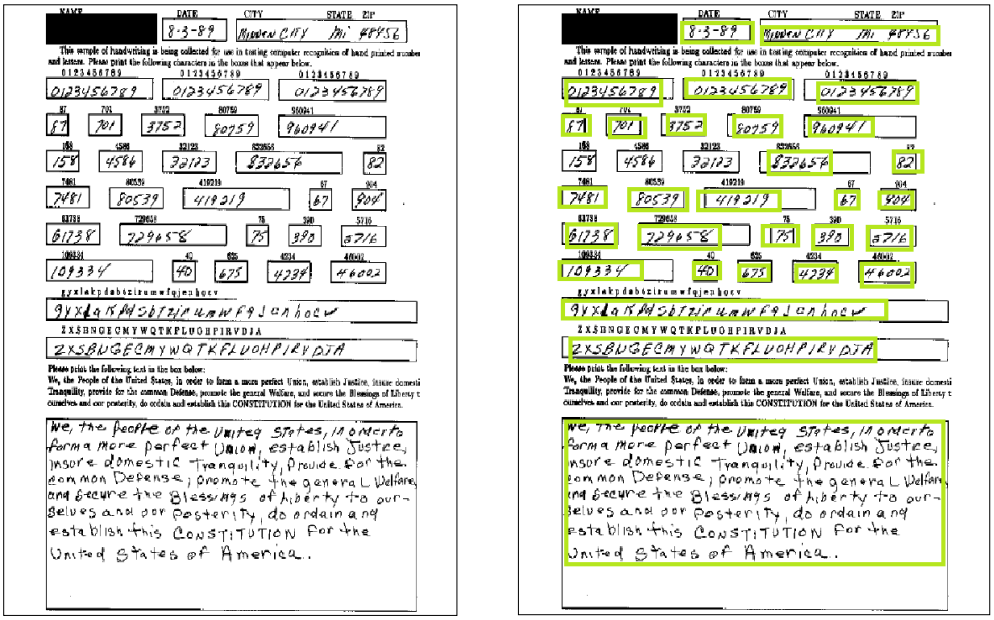
3. Proposed Idea and Methodology

3.1. Background Issue Overview

The idea proposed in this paper aims to explore an elaborate pre-processing methodology to enhance the recognition of images containing mixed-type text (i.e., handwritten and machine-printed text).

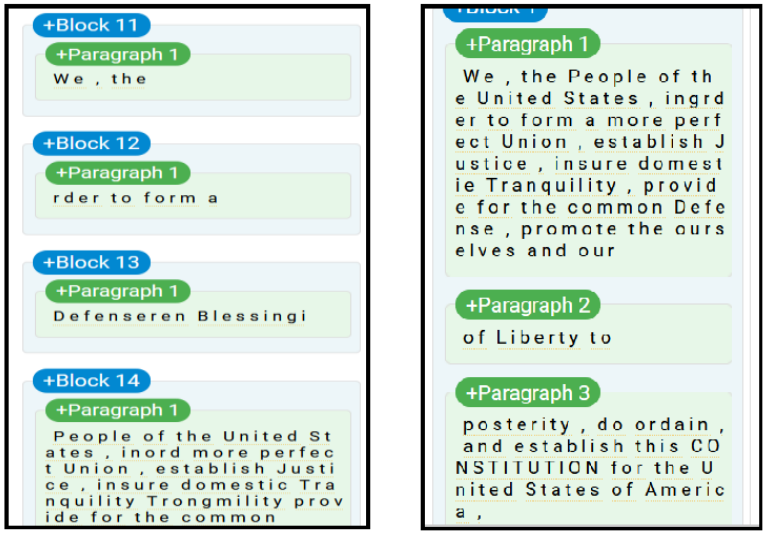
The concept of improving the output of a conventional OCR engine, strictly adhering to methods that do not contain the actual performance of recognition itself, proved to be a challenge. Upon study, the conclusion that is drawn time and again while attempting to conceive a method to improve the results of any task related to Computer Vision is undeniably one that revolves around developing innovative methods of image pre-processing. As was discussed in 2, a numerous proportion of image pre-processing algorithms already exist, a sizeable portion of which are enabled with complex mathematics. This begs the rhetoric of whether image pre-processing algorithms are nearing the boundaries of innovation or not, and if so, what the essence of the new branch it may grow into is. It is fairly simple to follow the fact that any given OCR engine in existence is likely only to be as good as the quality of the data it is fed. In other words, the performance of a given OCR engine is directly proportional to the quality of input it is fed. Naturally, most of the metrics that govern the quality of an input image to be recognized about the clarity of the OCR engine's data are expected to parse. However, to obtain a deep understanding of the reason behind the success of the methodology outlined in this proposal, we must first understand the singular feature of image quality this methodology exploits.

Figure 1a depicts a random image of a hand-filled form courtesy of [21]. Figure 1b represents the same image but highlighted with blocks about text data discovered upon feeding the unadulterated original image to the tool available at [18] powered by Google Cloud Vision API. Including the fact that not all the blocks are properly distinguishable, some of these also blocks do not capture continuous handwritten text. Figure 2a depicts the result of the same.



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

Figure 1. (**a**) Sample Hand-Filled Form, (**b**) Boxes of Text Highlighted.



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

Figure 2. (**a**) Recognized Text from Raw Input, (**b**) Recognized Text from Cropped Specific Image.

However, when the same image is cropped down to highlight just a singular block wherein handwritten text is present, and when the same unadulterated version of this block is run through the same OCR engine, the results are catastrophically different. Not only are the blocks highlighted more pronounced, but the result of the recognition service itself is also marginally improved. (Figure 2b).

It is not difficult to fathom that given an input, the output of any OCR engine would benefit from being provided specific areas of the image in a one-by-one. This paper proposes a methodology that employs the pixel-wise classification quality of semantic segmentation algorithms to accurately identify which areas of the image contain relevant text, crop these areas and then iteratively feed them to a conventional OCR engine improving the quality of the output.

Not only is D-Rex a methodology to enhance the results of conventional OCR, but it also houses modules related to the automatic digitization of the forms used for this project. This firstly requires a method to distinguish between field labels and the handwritten data that has been filled out, respectively, and then an algorithm to format and clean the extracted data.

3.2. Overview of Pipeline and Components

The Figure 3 image flow of input starting from entering the pipeline to obtaining the results is as follows:

1. An image of a hand-filled form of (3, 512, 512) pixels dimensions is input to the Image Pre-Processing Component. (3, 512, 512) are three dimensions of an input of a sample image. Various image processing techniques are applied to enhance the image and make it suitable for prediction during the segmentation phase.
2. The processed image enters the Block Identification and Isolation Module, where first it is used as an input to the semantic segmentation algorithm. The output of the semantic segmentation algorithm is a segmentation map. This segmentation map is used in the Block Cropping component to reference crops and store the relevant areas of the text identified by the segmentation algorithm.
3. The blocks of relevant text are sent to the Text Processing Module as well as the Label Isolation Component.
4. The Label Isolation component erases any handwritten text present in each block and outputs blocks with the only machine-printed text present.
5. The OCR Engine in the Text Processing Module accepts the collection of blocks containing all relevant text and the collection of blocks containing only labels. It uses Optical Character Recognition to recognize the text in all the blocks it has received regardless of the type of text present.
6. The recognized text, Label/Info Separation component, separates the labels from the respective information filled.
7. The post-processing component accepts both types of text and processes them to be suitable for further use.

3.3. Modules in Detail

Figure 4 shows the stepwise flow of processing a form document to extract the information from a form image. It includes an image preprocessing step, image segmentation, that is, masking of an image, Image processing using an OCR engine and, at the end, extracted digitized information as output.

(1) Dataset Generation Module

Before the modules listed in the pipeline diagram are discussed, it is important to highlight the data that would be used in the training of the segmentation module, as the segmentation algorithm would only work as well if the training data are diverse and of quality. Due to the fact that this methodology has been optimized to extract data specifically from forms, the natural type of image present in the training dataset should be those of hand-filled forms. Unfortunately, an easily available open-source dataset consisting of thousands of hand-filled forms was not available and so, custom dataset generation was chosen as a resort.

The major shortcoming related to creating a custom dataset is that the data must essentially be sourced from reality. Sourcing data from reality poses its challenges, and the problems that follow from doing the same are mirrored in the lack of quality of the dataset that usually results from it. This issue then propagates forward and manifests itself eventually in an unreliable and poorly performing deep learning algorithm.

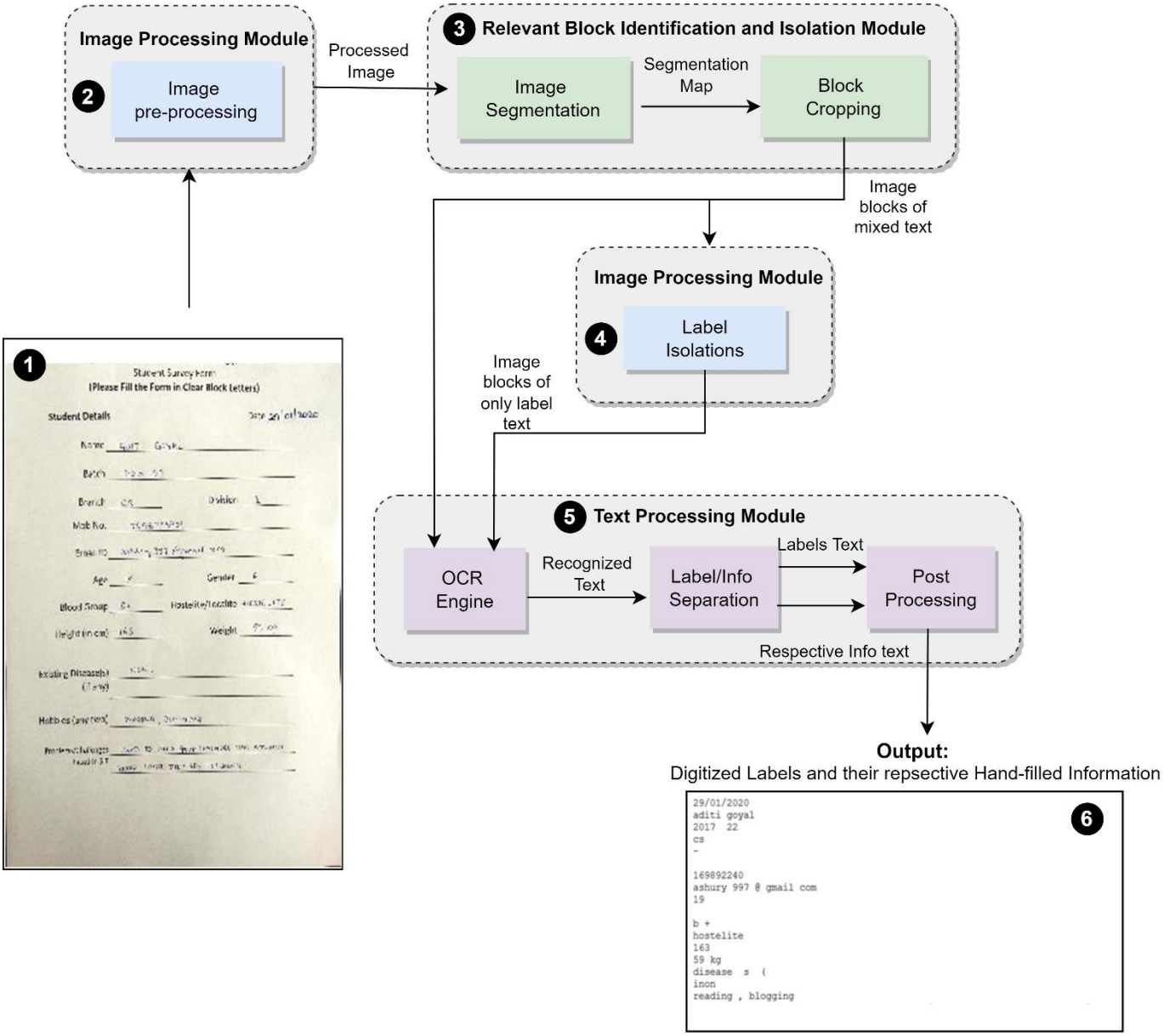


Figure 3. The Pipeline.

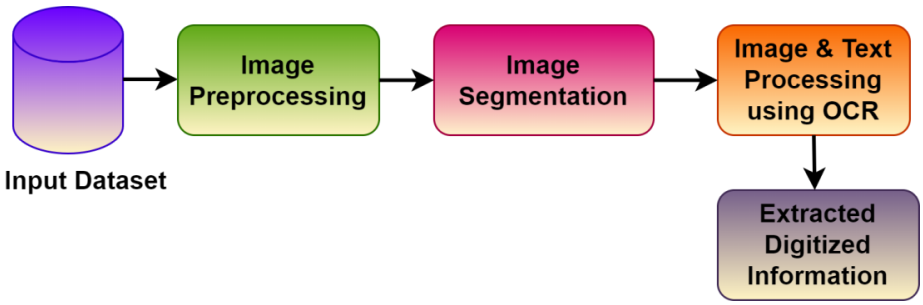


Figure 4. Modules and stepwise workflow.

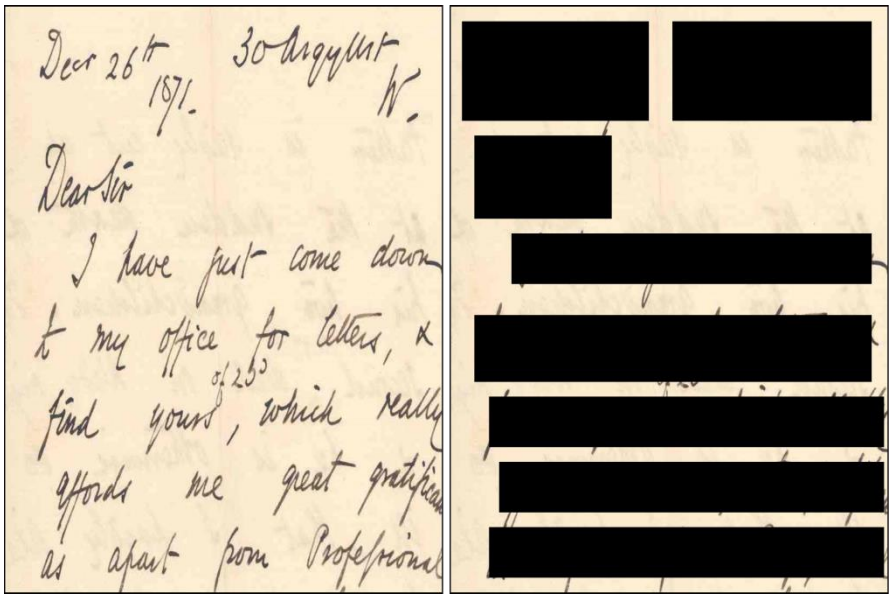
The creation of a good distribution of forms to suffice the requirement of 800+ images was incredibly difficult, the physical act of first creating and then hand-filling out these forms was unwise. Instead, the IAM Dataset, an open-source data repository containing thousands of individual handwritten sentence images put together by The Research Group

on Computer Vision and Artificial Intelligence INF, University of Bern, was used. This dataset provided a source of handwritten text that did not have to be self-made and could be leveraged to create a dataset of forms.

The dataset was to contain hand-filled forms, and upon condensing the concept of hand-filled forms, it was discovered that they could be represented as a sporadic distribution of handwritten sentences across a sheet of paper. With this point at the forefront, an algorithm was written to randomly pick 4–5 sentences from the IAM Dataset and append them to a blank image in a random fashion to mimic the fact that hand-written data is distributed across the page like in forms. The size of the images was decided to be kept (3, 512, 512) pixels throughout the dataset.

Semantic segmentation also requires the existence of a parallel mask for every image in the dataset. A mask is a binary image, not unlike a 'label' in typical machine learning datasets, wherein the pixels that are of the class to be segmented are highlighted in black, and the rest of the image is kept white. Generating these masks for every image required writing an algorithm that would append a box of black pixels precisely the same dimensions of a sentence to another blank image at the exact location that the sentence is being appended to in its image.

This method allowed us to generate 900+ images and masks for training seamlessly. Figure 5 shows one such image and its corresponding mask.



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

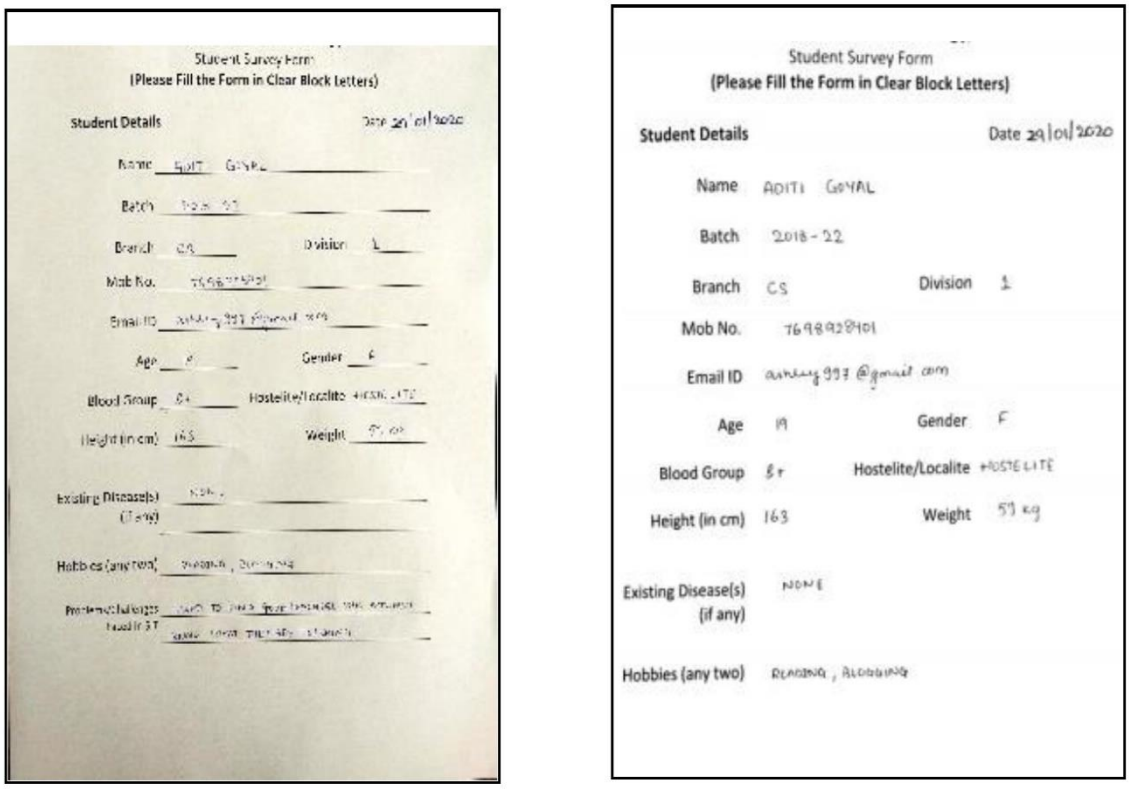
Figure 5. (**a**) Input Form, (**b**) Corresponding Mask.

(2) Image Processing Module

Image Processing Algorithms—This module houses various algorithms that are essential to pre-processing an input image of a form. The pre-processing techniques used are:

- Line Removal
- Gray Scale Conversion
- Gaussian Blurring
- Thresholding
- 3-Channel Re-Conversion

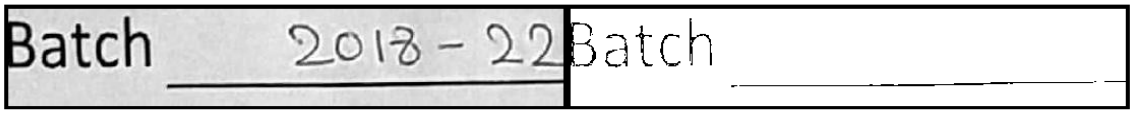
These techniques were used consecutively to enhance the image. Figure 6 depicts a sample input image and the image resulting from the application of these pre-processing methods. Label Isolation Component.



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

Figure 6. (**a**) Original Form, (**b**) Processed Input Form.

Another component included in the image processing module is the label isolation component. This is a simple thresholding algorithm that accepts the input image as a Tensor variable, replaces each pixel with a Boolean value checking whether or not that pixel's value is equal to 0.0 or not, and then converts each pixel back to a float value (wherein the floating equivalent of "True" is 1.0 and "False" is 0.0). This algorithm hinges on the assumption that all machine-printed text that appears in images of forms contains black pixels, while all other pixels are not relevant. This allows us to retain only those black pixels in color, thereby coloring any other white pixels. In effect, the machine-printed text has been retained while all other data in the image as been removed (Figure 7).



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

Figure 7. (**a**) Original Raw Image (**b**) Isolated Printed Text.

(3) Relevant Block Identification and Isolation Module

This module is by far the most important in the entire pipeline. It is here that the algorithms that crop relevant sections of the input image have been written. This module contains two main components:

(a) Image Segmentation using U-Net:

For the application of the pipeline, the segmentation task was to identify which areas of a form pertain to relevant text. To achieve this, it was decided that the U-Net algorithm is a suitable option, as it produces better outputs compared to another segmentation algorithm with a scarcity of data, and also that the output of the net is the same size as the size of

the input. The U-Net was developed by Olaf Ronneberger et al. for Bio-Medical Image Segmentation [9]. The architecture contains two paths; the encoder path and the decoder path in parallel. The left side is known as the contraction path (Encoder), where we apply regular convolutions and max-pooling layers. Here, the size of the image gradually reduces while the depth gradually increases. The encoder path (also known as the contraction path) is used to capture the semantic context and features in the image.

The right side is the expansion path (Decoder), where we apply transposed convolutions along with regular convolutions. In the decoder, the size of the image gradually increases, and the depth gradually decreases. The parallel decoder path (also known as the symmetric expanding path) enables precise localization using transposed convolutions. Intuitively, the Decoder recovers the "WHERE" information (precise localization) by gradually applying up-sampling or transposed convolutions [22,23]. To get better, precise locations at every step of the decoder, skip connections are used by concatenating the output of the transposed convolution layers with the feature maps from the Encoder at the same level. After every concatenation, we again apply two consecutive regular convolutions so that the model can learn to assemble a more precise output. This is what gives the architecture a symmetric U-shape, hence the name U-Net. The architecture is essentially an end-to-end fully convolutional network (FCN), and in essence, it only contains convolutional layers and no dense layers, enabling it to be used with images of varying sizes [24]. In Figure 8 given input of an image of a hand-filled form, a trained semantic segmentation algorithm can output a segmentation map, a blank image wherein only the areas of interest (in this case, relevant text) are highlighted in black [25].

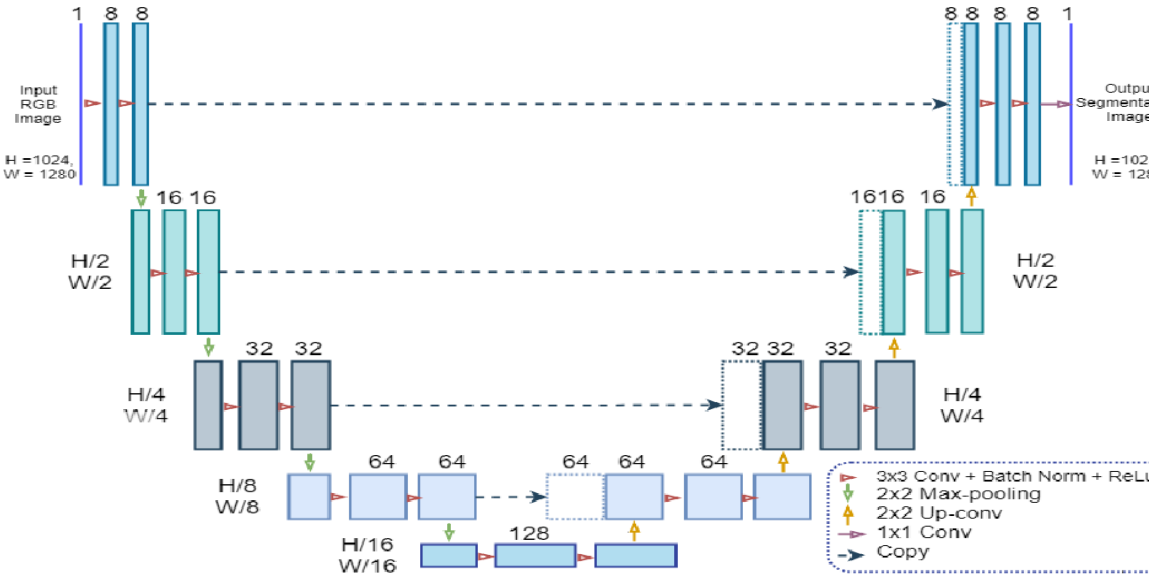


Figure 8. U-Net Architecture [26].

This segmentation map is crucial in understanding which areas of an image pertain to the major class in consideration. Figure 9 showcases the details of the architecture; the types of layers, their order and sequence, the shape of the output after each step, and the number of parameters at that step (trainable and non-trainable).

Figure 10 depicts the convergence of the loss in the range of 80–90 epochs. It shows the network performance loss curve during the model training process. Figure 11 showcases the segmentation map that was output for a given test image.

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 512, 512]	896
ReLU-2	[-1, 32, 512, 512]	0
Conv2d-3	[-1, 32, 512, 512]	9,248
ReLU-4	[-1, 32, 512, 512]	0
MaxPool2d-5	[-1, 32, 256, 256]	0
Dropout-6	[-1, 32, 256, 256]	0
Down-7	[[-1, 32, 256, 256], [-1, 32, 512, 512]]	
Conv2d-8	[-1, 64, 256, 256]	18,496
ReLU-9	[-1, 64, 256, 256]	0
Conv2d-10	[-1, 64, 256, 256]	36,928
ReLU-11	[-1, 64, 256, 256]	0
MaxPool2d-12	[-1, 64, 128, 128]	0
Dropout-13	[-1, 64, 128, 128]	0
Down-14	[[-1, 64, 128, 128], [-1, 64, 256, 256]]	
Conv2d-15	[-1, 128, 128, 128]	73,856
ReLU-16	[-1, 128, 128, 128]	0
Conv2d-17	[-1, 128, 128, 128]	147,584
ReLU-18	[-1, 128, 128, 128]	0
MaxPool2d-19	[-1, 128, 64, 64]	0
Dropout-20	[-1, 128, 64, 64]	0
Down-21	[[-1, 128, 64, 64], [-1, 128, 128, 128]]	
Conv2d-22	[-1, 256, 64, 64]	295,168
ReLU-23	[-1, 256, 64, 64]	0
Conv2d-24	[-1, 256, 64, 64]	590,080
ReLU-25	[-1, 256, 64, 64]	0
MaxPool2d-26	[-1, 256, 32, 32]	0
Dropout-27	[-1, 256, 32, 32]	0
Down-28	[[-1, 256, 32, 32], [-1, 256, 64, 64]]	
Conv2d-29	[-1, 512, 32, 32]	1,180,160
ReLU-30	[-1, 512, 32, 32]	0
Conv2d-31	[-1, 512, 32, 32]	2,359,808
ReLU-32	[-1, 512, 32, 32]	0
ConvTranspose2d-33	[-1, 256, 64, 64]	524,544
Dropout-34	[-1, 256, 64, 64]	0
Conv2d-35	[-1, 256, 64, 64]	1,179,904
ReLU-36	[-1, 256, 64, 64]	0
Conv2d-37	[-1, 256, 64, 64]	590,080
ReLU-38	[-1, 256, 64, 64]	0
Up-39	[-1, 256, 64, 64]	0
ConvTranspose2d-40	[-1, 128, 128, 128]	131,200
Dropout-41	[-1, 128, 128, 128]	0
Conv2d-42	[-1, 128, 128, 128]	295,040
ReLU-43	[-1, 128, 128, 128]	0
Conv2d-44	[-1, 128, 128, 128]	147,584
ReLU-45	[-1, 128, 128, 128]	0
Up-46	[-1, 128, 128, 128]	0
ConvTranspose2d-47	[-1, 64, 256, 256]	32,832
Dropout-48	[-1, 64, 256, 256]	0
Conv2d-49	[-1, 64, 256, 256]	73,792
ReLU-50	[-1, 64, 256, 256]	0
Conv2d-51	[-1, 64, 256, 256]	36,928
ReLU-52	[-1, 64, 256, 256]	0
Up-53	[-1, 64, 256, 256]	0

Figure 9. Details about the architecture and Layer.

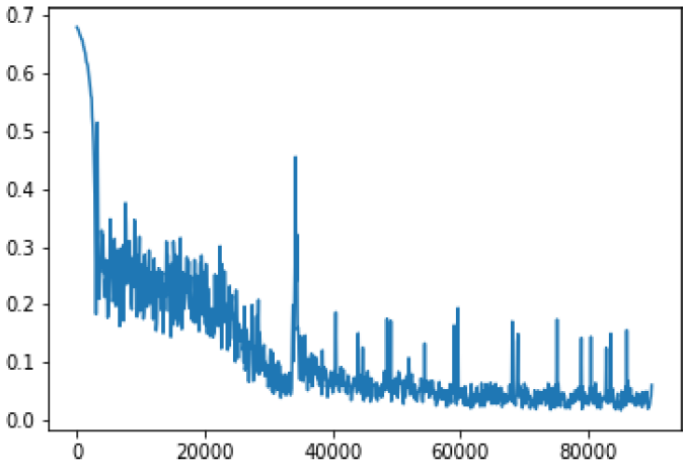
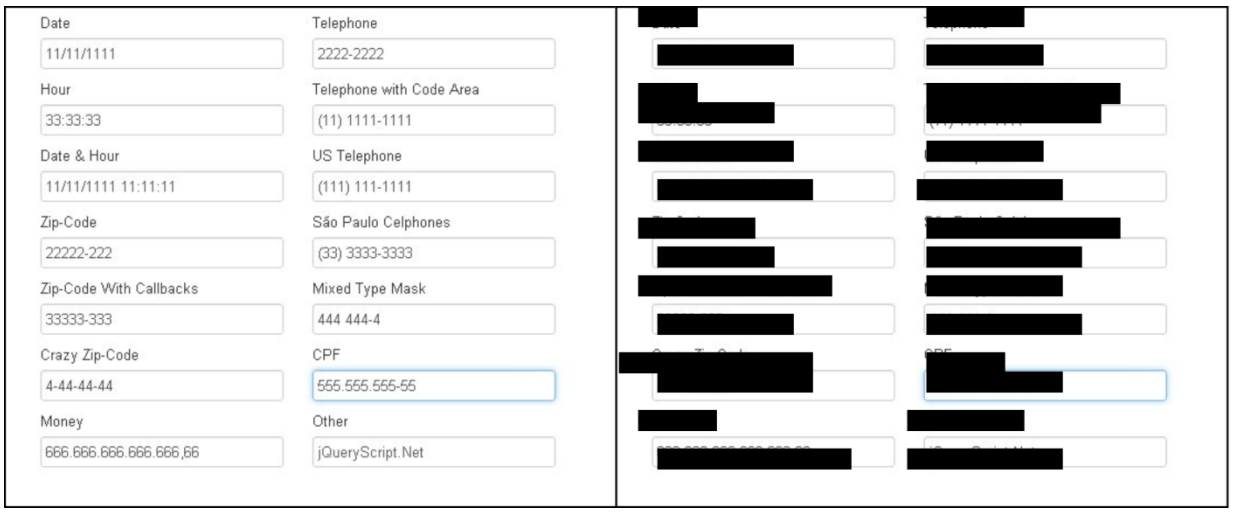


Figure 10. Loss Curve During Training.



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

Figure 11. (**a**) Input Sample Form, (**b**) Predicted Segmentation Map.

Table 4 shows the details about the parameters used in U-Net Architecture in terms of total parameters used, trainable parameters, Non-trainable parameters, input parameters, parameters size, estimated total size used. An interesting observation was made during the intermittent testing of the model prior to the finalization of the model for pipeline production. Upon experimenting with different types of training data, the model seemed to behave in unexpected manners. Namely, when trained with a training set consisting of only machine-printed text, the model understandably could not recognize and segment hand-written text, as it had not learned the specific features of hand-written text to be able to converge well enough. When a fresh model was trained on a training set consisting of images that contained a mixture of machine-printed and handwritten text (i.e., mixed text types), the model could not properly converge and accurately segment handwritten and machine-printed text. The deduction made at this point was that because now the model faced twice the number of features, it required a larger number of epochs to reach satisfactory convergence. However, when the model was presented with a training set that consisted exclusively of handwritten text images, convergence was met quite a sizeable number of epochs before the full 100. A fitting explanation of this phenomenon was that while it is difficult to converge using data with twice the number of separate learnable features, it is simpler to converge on data that includes a single type of text, but whose features are similar to the other type as well. This means that the features of the handwritten text are strikingly similar to machine printed text, which is why convergence was possible.

Table 4. Details about the Parameters of the U-Net Architecture.

Total Params	7,760,353
Trainable Params	7,760,353
Non-Trainable Params	0
Input Size (MB)	3.00
Forward/Backward Pass Size (MB)	178,256,520.00
Params Size (MB)	29.60
Estimated Total Size (MB)	178,256,552.60

The algorithm is a simple one:

1. Create an active flag and set it as False. This flag is used to indicate the first time a row containing black pixels has been encountered and indicate when the parsing of a single block is finished before starting the next one.
2. For every row in the segmentation map image, check whether that row contains one or more black pixels.
3. If it contains one or more black pixels, it means that that row has a part of the segmented class in it. Append the values of the entire row of the segmentation map to one list and the same respective row of the original image to another list. These lists represent a single continuous box. Set the active flag as True.
4. If the row does not contain any black pixels, and if active is True, append the above lists to two other lists that house entire boxes. Empty out the lists that represent single boxes. Set active to False and continue.
5. If the row does not contain any black pixels, and if active is False, continue.
6. End parsing when all rows have been parsed.
7. For every box within the lists that represent entire boxes, perform steps 1–6 with transposed boxes as input. Figure 12 depicts some sample results of this algorithm on the given segmentation map.
8. Re-transpose all the boxes obtained from 7.
9. End.

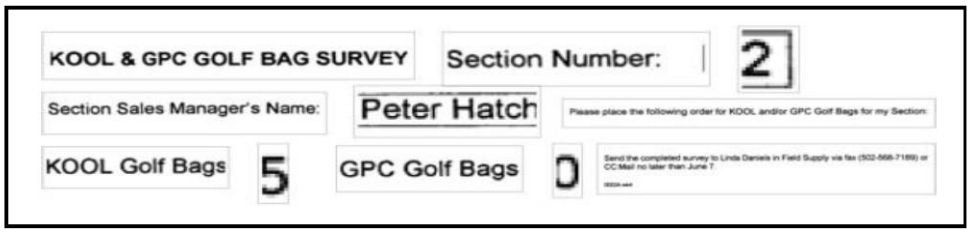


Figure 12. Depicts some results of this algorithm on the given segmentation map.

(4) Text Processing Module

(a) Optical Character Recognition Engine

The function of this methodology hinges on having a conventional OCR engine whose results are to be optimized using the D-Rex pipeline. A thorough internet search reveals that, as mentioned previously, OCR engines dedicated to recognizing mixed text types are not widely available. Therefore, online OCR engines were considered as the resort.

The selection of a reliable and accurate OCR engine was a tough one. Table 1 depicts a comparison of different online OCR engines that were taken into consideration. Google Cloud Vision API outperformed the rest [23]; however, the reasons for selecting this engine extend beyond accuracy. It was found that Google Cloud Vision API offers a dedicated handwriting recognition service that was observed to perform to the same caliber on machine-printed text simultaneously [27,28]. This ensured that any disparity between handwritten text recognition from machine printed text was gapped as the same OCR engine would be applied in both instances.

(b) Label/Info Separation Component

The Label/Info Separation component was a mandatory implementation to distinguish between the labels of a field and the actual information that was hand-written into them. When two instances of strings, one about just the label names and the second about the label names and the info, are passed to this component, it uses a simple word-level substring replacement algorithm to remove the presence of labels in the information strings. This has the effect of separating the labels and the respective information from each other. It outputs two series of strings; labels recognized and respective information. Before these strings can be used as recognized data from hand-filled forms, the final step is running them through the post-processing algorithms. Several bugs were detected upon observing the output of the Label/Info Separation component, namely;

(1) Instances of printed text that did not correspond to any handwritten data. This included machine-printed texts like titles and serial numbers, etc.
(2) Instances where there are actual labels, but there was no corresponding handwritten data text (the missing text was usually in the line below).
(3) Substring replacement had to be done on a word-by-word basis and not on an entire sentence basis to avoid stray words staying behind.
(4) Instances where the removal of binary option labels (e.g., "Yes/No") removed the actual data filled in for that label correspondingly.
(5) Instances where random blank spaces appeared.

The post-processing algorithms are then just a series of simple algorithm and code changes made to the OCR module to ensure that the bugs listed above are not encountered.

4. Results, Observations and Discussions

4.1. Defining Instances for Evaluation

As shown in Figures 13 and 14, the D-Rex pipeline was able to accept a form image as input and output labels and their respective information relatively well. However, to properly assess the functionality of the pipeline, two cases must be recognized wherein opportunities for result and performance observations were possible:

CASE I—Post Semantic Segmentation Step (assessment of the performance of the segmentation algorithm)

CASE II—Post Complete Text Extraction (assessment of text extracted using OCR). Performed against two sub-cases:

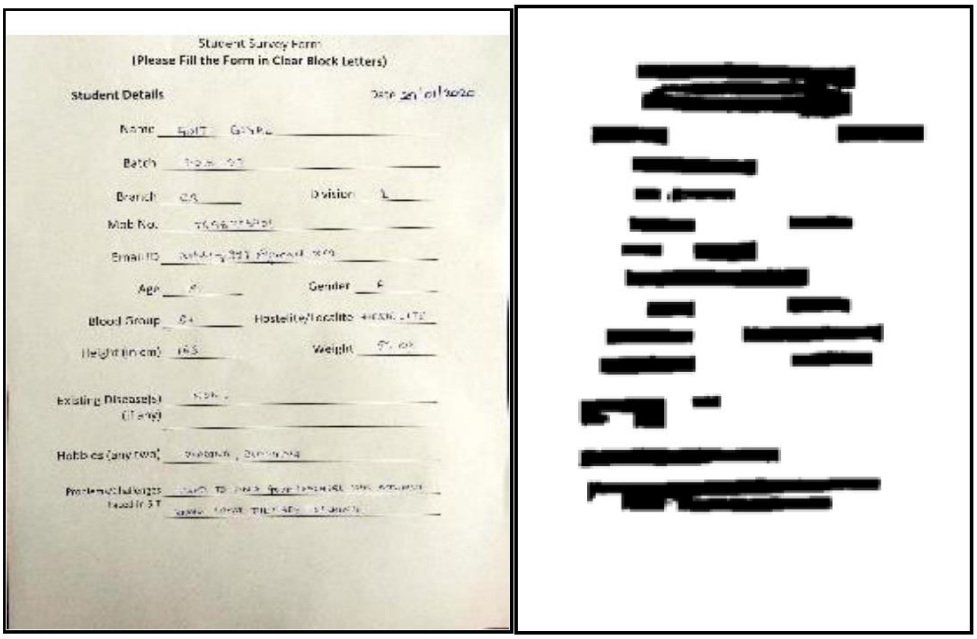
(a) the actual handwritten text translated manually by a human being, and
(b) text that is extracted by Google OCR without using our pipeline for the same input.

4.2. Results

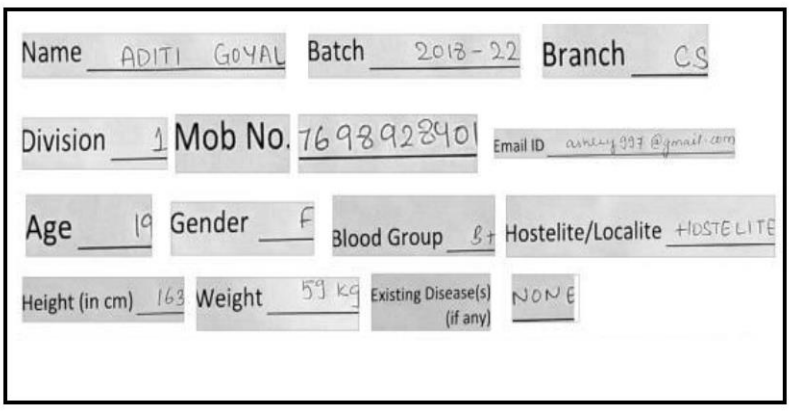
All results and evaluations performed in this section and at any phase of the pipeline were performed using the same set of testing images to ensure fairness and uniformity. Although this, some alterations to the raw output of CASE—II, part b had to be made to clean the output to make it suitable for result generation. These alterations have been outlined in the respective sub-section and were made sure not to compromise the integrity of the tests in any way.

CASE—I (Post Semantic Segmentation):

Figure 13 shows one pass of the segmentation and block extraction algorithms obtains conclusive results for the two most important fundamental issues addressed by this methodology:



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



(**c**)

Figure 13. (**a**) Input Form, (**b**) Predicted Segmentation Map (**c**) Extracted Blocks of Relevant Text.

Evaluating the performance of the U-Net was undoubtedly amongst the most important evaluations that had to be made, as a large portion of the pipeline depends on the accurate output of the net. Quite a few metrics were chosen to ensure that the segmentation net was working well. Table 5 shows the performance results of the network.

Table 5. (**a**) Confusion Matrix Results, (**b**) Complex Metrics Results.

(a)		**Predicted Positive**	**Predicted Negative**
True Positive		True Positive: 245194.68888888	False Negative: 296.6111111111111
True Negative		False Positive: 198.422222222222	True Negative: 16454.277777777777
(b)	Sr. No.	Evaluation Metric	Score
	01.	Average Accuracy	0.9981115976969401
	02.	Precision	0.9991914107884944
	03.	Recall	0.9987917652841012
	04.	F1	0.9991914107884944
	05.	Jaccard Index	0.9980515790200687

CASE—II (Post Complete Text Extraction):

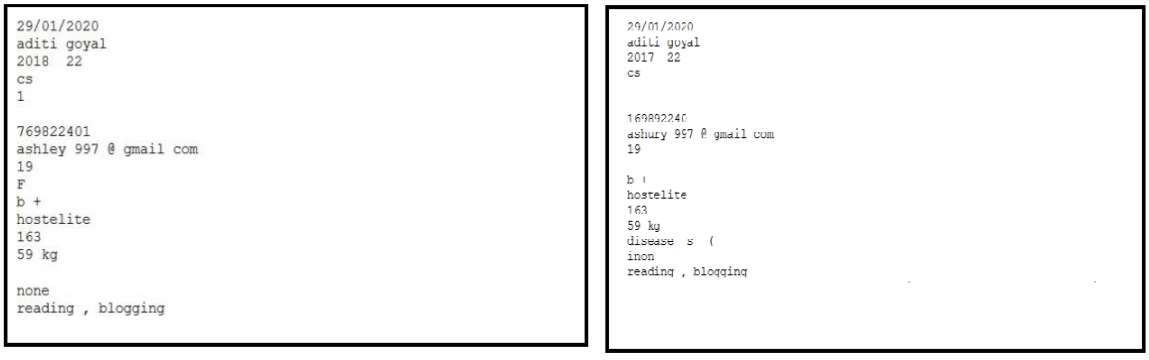
An evaluation of the results of this phase essentially pertains to a general evaluation of the entire pipeline, as the output of this phase is the final output of the pipeline. Two types of evaluations had to be made to

(a) Are the D-Rex pipeline output results of a good enough level to be used for production? and,
(b) Did the D-Rex pipeline enhance the results of a conventional OCR Engine on mixed-type data?

The first sub-case was performed due to the D-Rex pipeline measured against the true strings (as donated and verified by a human being). The second sub-case was performed with the slightly processed output of running the raw image through the same Google Cloud Vision API OCR engine without first running it through the D-Rex pipeline. The result of this evaluation is indicative of the success of the methodology provided in this paper.

A. D-Rex Pipeline Results measured against True Strings

Figure 14 denotes the results of the D-Rex pipeline for the input form image from Figure 13a. Table 6 contains the Jaccard Index for String Similarity and the Jaro-Winkler Score, both calculated by applying these metrics to compare the output and the True Strings.



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

Figure 14. (**a**) True Strings for given Input Form, (**b**) Extracted Information for same Form using D-Rex Pipeline.

Table 6. Metric Scores for Case II—A.

Sr. No.	Metric	Result
01.	Jaccard Index for String Similarity	0.6892857142857143
02.	Jaro-Winkler Score	0.8248173731109226

B. **Google Cloud Vision API's OCR Engine Results measured against True Strings**. Upon giving the same OCR engine the same image without first running it through the D-Rex Pipeline, the results are displayed in Table 7.

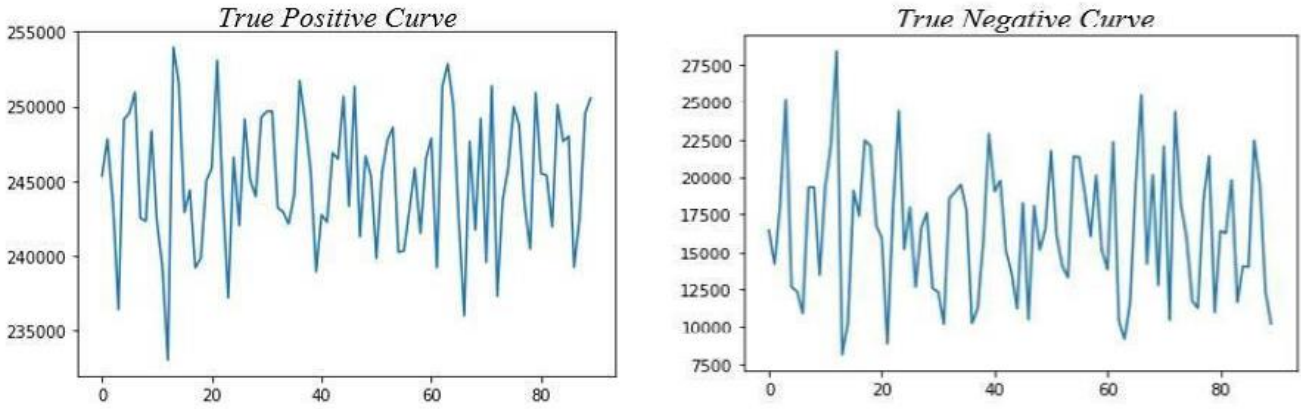
Table 7. Metric Scores for Case II—B.

Sr. No.	Metric	Result
01.	Jaccard Index for String Similarity	0.254951690821256
02.	Jaro-Winkler Score	0.8248173731109226

4.3. Observations and Discussions

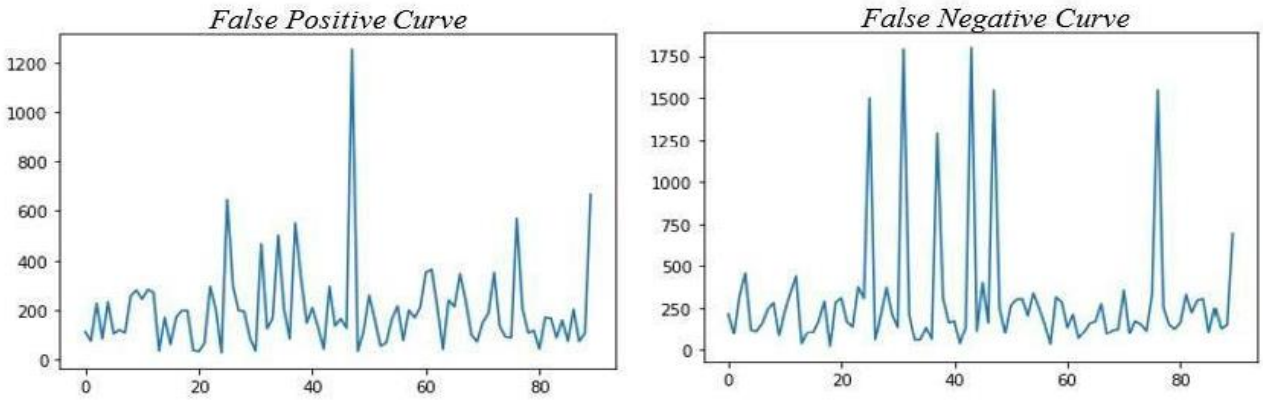
As can be observed from Table 5a, the number of pixels that were predicted true and turned out to be positive (i.e., true positive) and those that were predicted to be false and turned out negative (true negative) far outweigh the other features. This means that the net performs well in predicting whether a given pixel belongs to a class or not, as compared to it making a mistake and misclassifying the pixel.

Figures 15 and 16 show the variation of all components of the confusion matrix over all the samples included in the testing dataset. This is an important factor to consider, as visualizations such as this one reveal important observations that may otherwise be overlooked. For example, while depicting what seems to be rapid ascents and descents in all of the graphs, these graphs still show that the general range for the number of pixels identified is within an acceptable range with respect to the conditions portrayed in the last paragraph. However, the most important observation with regards to the topic of this paper is the vast difference in Jaccard Index for String Similarity scores between using raw OCR and using the D-Rex pipeline to pre-process the image. This is a clear indication of the fact that given an image, a conventional OCR engine would benefit by a magnitude of almost 2.7 times by segmenting and cropping the relevant areas of text beforehand. This proves that the use of D-Rex produces reliable and much better results, but this also portrays the fact that the improvement of the results of an OCR need not require incredibly complex image pre-processing concepts and custom OCR designs.



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

Figure 15. (**a**) True Positive Curve, (**b**) True Negative Curve.



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

Figure 16. (**a**) False Positive Curve, (**b**) False Negative Curve.

5. Conclusions

In an effort to enhance the quality of the output, authors have presented a pixel-wise classification technique to precisely identify areas of an image containing pertinent text in this paper. The suggested methodology also facilitates the improved digitization of mixed-typed text manuscripts. An experimental investigation demonstrates that the suggested pipeline design gives conventional OCR dependable and high-quality inputs through intricate picture preprocessing, improving performance and accuracy.

The main takeaway from this methodology should not be that cropped images can help OCRs produce better results; rather, it should be understood that the key to improving OCR is not a complex methodology involving cutting-edge technology, but rather a clear understanding of the factors that make it simpler to recognize text in images. It will be helpful for researchers to work in this domain of research.

6. Limitations and Future Scope

Perhaps the most crucial limitation of the D-Rex pipeline was the need to re-initialize the model every time the data filled in the form has to be extracted. Model initializations are heavy in terms of computation, and amongst other things, frequently redundant. Another important shortcoming is the quality of the input form image. Testing with images that were deliberately made to be noisy or with images that were intentionally taken at the wrong angles revealed that the quality of the output dropped. To remedy this, the consideration of using image pre-processing techniques such as skew correction was made; however, it was decidedly out of the scope of this discussion.

However, a key takeaway from this methodology should not be the fact that OCRs may output improved results if fed in cropped images, on the contrary, it is that the solution for the hole in the heart of OCR lies not in complicated methodology that employs impressive technology, but in the simple understanding of what allows for the easier recognition of text in images.

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

Funding: The APC was funded by Symbiosis International (Deemed) University.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Ranjan, A.; Behera, V.N.J.; Reza, M. OCR Using Computer Vision and Machine Learning. In *Machine Learning Algorithms for Industrial Applications*; Springer: Cham, Switzerland, 2021; pp. 83–105.
2. Available online: http://www.capturedocs.com/thread/handwritten-invoices/ (accessed on 5 January 2022).
3. Rabby, A.K.M.; Islam, M.; Hasan, N.; Nahar, J.; Rahman, F. A Deep Learning Solution to Detect Text-Types Using a Convolutional Neural Network. In *Proceedings of the International Conference on Machine Intelligence and Data Science Applications*; Springer: Singapore, 2021; pp. 727–736.
4. Zheng, Y.; Li, H.; Doermann, D. Machine printed text and handwriting identification in noisy document images. *IEEE Trans. Pattern Anal. Mach. Intell.* **2004**, *26*, 337–353. [CrossRef] [PubMed]
5. Patil, S.; Joshi, S. Demystifying User Data Privacy in the World of IOT. *Int. J. Innov. Technol. Explor. Eng.* **2019**, *8*, 4412–4418. [CrossRef]
6. Bidwe, R.V.; Mishra, S.; Patil, S.; Shaw, K.; Vora, D.R.; Kotecha, K.; Zope, B. Deep Learning Approaches for Video Compression: A Bibliometric Analysis. *Big Data Cogn. Comput.* **2022**, *6*, 44. [CrossRef]
7. Baviskar, D.; Ahirrao, S.; Potdar, V.; Kotecha, K. Efficient Automated Processing of the Unstructured Documents Using Artificial Intelligence: A Systematic Literature Review and Future Directions. *IEEE Access* **2021**, *9*, 72894–72936. [CrossRef]
8. Sayyad, S.; Kumar, S.; Bongale, A.; Bongale, A.M.; Patil, S. Estimating Remaining Useful Life in Machines Using Artificial Intelligence: A Scoping Review. *Libr. Philos. Pract.* **2021**, *2021*, 4798.
9. Chaudhuri, A.; Mandaviya, K.; Badelia, P.; Ghosh, S.K. Optical Character Recognition Systems. In *Optical Character Recognition Systems for Different Languages with Soft Computing*; Springer: Cham, Switzerland, 2016; pp. 9–41. [CrossRef]
10. Chen, X.; Jin, L.; Zhu, Y.; Luo, C.; Wang, T. Text recognition in the wild: A survey. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–35.
11. Memon, J.; Sami, M.; Khan, R.A.; Uddin, M. Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR). *IEEE Access* **2020**, *8*, 142642–142668. [CrossRef]
12. Guo, Y.; Liu, Y.; Georgiou, T.; Lew, M.S. A review of semantic segmentation using deep neural networks. *Int. J. Multimed. Inf. Retr.* **2017**, *7*, 87–93. [CrossRef]
13. Yang, J.; Zhu, J.; Wang, H.; Yang, X. Dilated MultiResUNet: Dilated multiresidual blocks network based on U-Net for biomedical image segmentation. *Biomed. Signal Process. Control* **2021**, *68*, 102643. [CrossRef]
14. Shelhamer, E.; Long, J.; Darrell, T. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 640–651. [CrossRef]
15. Kaur, S.; Mann, P.; Khurana, S. Page Segmentation in OCR System-A Review. *Int. J. Comput. Sci. Inf. Technol.* **2013**, *4*, 420–422.
16. Reisswig, C.; Katti, A.; Spinaci, M.; Höhne, J. Chargrid-OCR: End-to-end trainable Optical Character Recognition through Semantic Segmentation and Object Detection. In Proceedings of the Workshop on Document Intelligence at NeurIPS 2019, Vancouver, BC, Canada, 14 December 2019.
17. Shubh, M.A.; Hassan, F.; Pandey, G.; Ghosh, S. Handwriting Recognition Using Deep Learning. *Emerg. Trends Data Driven Comput. Commun. Proc.* **2021**, *2021*, 67.
18. Boualam, M.; Elfakir, Y.; Khaissidi, G.; Mrabti, M. Arabic Handwriting Word Recognition Based on Convolutional Recurrent Neural Network. In *WITS 2020*; Springer: Singapore, 2021; pp. 877–885. [CrossRef]
19. Huo, Q. Underline Detection and Removal in a Document Image Usingmultiple Strategies. 2001. Available online: https://www.researchgate.net/publication/4090302_Underline_detection_and_removal_in_a_document_image_using_multiple_strategies (accessed on 12 February 2022).
20. Abuhaiba, I.S. Skew Correction of Textural Documents. *J. King Saud Univ.-Comput. Inf. Sci.* **2003**, *15*, 73–93. [CrossRef]
21. Patrick, J. *Handprinted Forms and Character Database, NIST Special Database 19*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 1995.
22. Google Cloud Vision API Documentation. Available online: https://cloud.google.com/vision/docs/drag-and-drop (accessed on 20 February 2022).
23. Dataturks.com. Image Text Recognition APIs Showdown. Google Vision vs Microsoft Cognitive Services vs AWS Rekognition. Available online: https://dataturks.com/blog/compare-image-text-recognition-apis.php (accessed on 1 March 2022).
24. Li, X.; Qian, W.; Xu, D.; Liu, C. Image Segmentation Based on Improved Unet. In *Journal of Physics: Conference Series*; IOP Publishing: Bristol, UK, 2021; Volume 1815, p. 12018.
25. Nasir, T.; Malik, M.K.; Shahzad, K. MMU-OCR-21: Towards End-to-End Urdu Text Recognition Using Deep Learning. *IEEE Access* **2021**, *9*, 124945–124962. [CrossRef]
26. U-Net Architecture Image, 2011, LMB, University of Freiburg Department of Computer Science Faculty of Engineering. Available online: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/ (accessed on 4 April 2022).

27. Hwang, S.-M.; Yeom, H.-G. An Implementation of a System for Video Translation Using OCR. In *Software Engineering in IoT, Big Data, Cloud and Mobile Computing*; Springer: Cham, Switzerland, 2021; pp. 49–57.
28. Edupuganti, S.A.; Koganti, V.D.; Lakshmi, C.S.; Kumar, R.N.; Paruchuri, R. Text and Speech Recognition for Visually Impaired People using Google Vision. In Proceedings of the 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), Tiruchirappalli, India, 7–9 October 2021; pp. 1325–1330.