**HANDWRITTEN TEXT TO DIGITAL TEXT**

**Hithiesh V K, Ajay K P, Aneesh Kulkarni, Giridhar D U**

**ABSTRACT**

In sectors such as digitization of historic documents, facilitating seamless communication in the digital age, transformation of written text into digital images has considerable implications. The need for effective and accurate communication is becoming increasingly important, as handwriting continues to retain the primary form of communication. Therefore, we are using advanced image processing techniques and the power convolutional neural networks (CNNs) to address the complexities that come with handwritten text, which includes variations in the style and size of letters as well as their orientation. Our research and experimentation contribute to the development of handwriting recognition technology, In the digital age, bridging the gap between analogue and digital worlds will lead to improved communication and knowledge preservation.

1. **Introduction**

1.1 Problem Statement

Imagine a situation where our method is applied by educational institutions trying to modernize their teaching methodologies. Imagine a classroom where, for better comprehension and revision, handwritten notes on the blackboard or blackboard need to be captured and shared digitally with students. Our approach therefore consists of two steps. Firstly, accurate recognition and digitalization of handwritten text on a whiteboard is part of the capture process. Secondly, in order to ensure clarity and legibility for students to access them remotely, the conversion step will convert the digitized text into digital images. The ability to execute these steps quickly and precisely is a fundamental requirement.

1.2 Plan

In order to achieve its objectives effectively, the project plan is organized in a number of main sections. First, in order to develop a new approach for real-time processing, the focus is on identifying areas of handwritten text in photographs. After that, the project moves to the classification phase, where the content of the identified text regions is decoded. In order to assess the accuracy and speed of detection and classification procedures, strict testing shall be carried out so as to ensure timely feasibility. Finally, efforts are being made to fine tune the method to further improve accuracy. It is considered essential to improve the algorithm's performance, despite the potential time-consuming nature of this phase. Overall, the plan sets out a structured approach to the objectives of the project and provides a clear roadmap for its implementation.

1. **Related Works**

Using classical and deep learning techniques in computer vision, handwritten text recognition has been extensively explored. The landscape for handwritten text recognition has been transformed by the advent of deep learning, in particular neural networks, and it is aligning itself with advances in object classification, localization and detection.

2.1 Sliding Window

Initial approaches to handwritten text recognition treated it as a repetitive object classification task, employing sliding windows alongside neural networks for image analysis. Vaillant et al. pioneered the use of convolutional neural networks (CNNs) to detect human presence within sliding window areas. Although this method is efficient, due to repetitive CNN computations in local regions it has resulted in substantial computational costs.

Other researchers such as LeCun et al. and Bengio et al. contributed significantly to the development of CNN architectures for handwriting recognition, in addition to Vaillant et al. In order to enhance the accuracy and effectiveness of recognition, their research focused on optimizing network architecture and training strategies.

2.2 Detection with Proposals

In order to limit the number of candidate locations for analysis, researchers have been using regional proposal methods in order to address computational problems associated with window switching. The groundbreaking RCNN, introduced by Girshick et al., used regional proposals for the creation of regions of interest, which were subsequently analyzed by CNNs for classification and localization. RCNN faced challenges such as slow training and testing times, which led to further innovation in spite of its breakthrough.

In addition to R-CNN, researchers like He et al. and Ren et al. proposed advancements such as Fast R-CNN and Faster R-CNN. These methods optimized training and testing times through techniques like RoI pooling and integrated Region Proposal Networks (RPNs), achieving significant reductions in runtime compared to R-CNN.

2.3 Detection without Proposals

Instead of using regional approaches, methods such as SSDSingle Shot MultiBox Detector by Liu et al. are directly processing input images through convolutional neural networks without relying on region proposals. These methods divide the input image into grid cells and determine classification scores and bounding box coordinates in each cell. These approaches may sacrifice accuracy compared to proposal methods based on regions, although they offer faster training and test times.

In addition, to further improve the efficiency and accuracy of handwritten text recognition tasks, researchers such as Liu et al. have proposed variants of their methods, such as SSD with MobileNet. In addition, techniques to improve the accuracy and effectiveness of recognition with no reliance on region proposals are being explored in this area for making a balance between speed and precision when it comes to handwriting applications.

2.4 Comparison

A spectrum of tradeoffs between computational efficiency and accuracy is presented in the landscape of handwritten text recognition. The techniques of sliding windows offer simplicity, but their processing times are slow. The use of regional proposal methods increases efficiency and accuracy, but may incur computational costs. Proposal Free methods prioritize speed, but can compromise accuracy. The key challenge in handwriting recognition remains the balance between efficiency and accuracy, which is a prerequisite for continued research efforts to come up with novel solutions.

1. **Model**

3.1 CRNN

Our approach to handwritten text recognition revolves around treating it as a sequence recognition problem, where the CRNN (Convolutional Recurrent Neural Network) architecture plays a pivotal role. Convolutional neural networks are used in this architecture to extract features from input images and to capture spatial information efficiently. These convolutional layers are then followed by Recurrent Neural Networks (RNNs), such as LSTM (Long Short-Term Memory) which specialize in processing sequential data.

We perform preprocessing steps to prepare the input data before feeding the images into the CRNN model for training. In the first place, the input image is loaded from the dataset using OpenCV, which reads the grayscale image. Then the images are adjusted to a standardised size that can be processed by your neural network. Images are adjusted to a width of 256 mm and height of 64 mm with our implementation. This resizing is intended to ensure uniform input dimensions for each image.

After resizing, the pixels of the resized images are normalized to a range of 0 to 1. Dividing the pixel values by 255.0 is used for this normalization step. Finally, the preprocessed images are converted to NumPy arrays and reshaped to the required input shape to be expected from the CRNN model. In our case, images are reshaped to a shape of (-1, 256, 64, 1), where -1 denotes the batch size dimension.

Convolutional layers are performing feature extraction in our CRNN model, while repeat layers process the sequence. Max-pooling layers are utilized for downsampling the feature maps obtained from convolution, and dropout layers are inserted for regularization to prevent overfitting. In order to capture the temporal dependencies arising from the Sequential character of written text, it is important to have recurring layers in particular LSTM.

To classify each element of the output sequence into characters, a fully connected layer with softmax activation is used at the end of the CRNN model. This step essentially transposes the input image into a set of characters, including alphanumeric characters and special symbols.

We use the (Connectionist Temporal Classificationual) CTC loss function during training to allow the model to learn the alignment between the input images and the corresponding text labels without the need for explicit alignment annotations. In addition, to enhance the generalization capability of the model and to avoid overfitting during the training process, we use techniques such as data augmentation and regularization.

3.2 Loss function

The loss function is Connectionist Temporal ClassificationsCTC Loss. In cases where the alignment between input sequences and output sequences is not explicitly known, this loss function is commonly used in sequence-to-sequence tasks, such as handwriting recognition.

Without requiring explicit alignment annotations, the CTC loss allows a model to determine whether an input image is aligned with its corresponding text labels. The probability of any alignment is calculated, based on all possible alignments between the input sequence image features and the output sequence recognizable text.

During training, the CTC penalizes deviations from the predicted sequence and the ground truth labels and encourages the model to learn meaningful representations of handwritten text. The model learns to accurately transcribe input images into corresponding text sequences, thereby facilitating effective handwritten text recognition, by optimising the CTC loss.

3.3 Training

In order to normalize the pixels, we divide the range between 0 and 1 by 255, then transform the samples into dimensions 1, 256, 64, 1. A CTC loss function has been selected. We have used batch normalization, along with relu activation function with 2D maxpooling of pool size 2X2 and dropout with input parameter 0.3. In addition, our RNN was defined using LSTM. There is a softmax activation function in the last output layer. Adam optimizer with a learning rate of 0.0001 is included in this model.

1. **Process**

4.1 Setup

Our implementation is developed in Python, compatible with both Python 2 and Python 3 environments. We're using TensorFlow 2.16 to build a CRNn model, define loss functions and optimize the solver so that we can streamline our development processes. In addition, we use OpenCVcv2 to load and write pictures, do different image processing tasks such as resizing or normalization. Using the Jupyter Lab's code execution and experimentation capabilities, training was conducted in a standard laptop environment without GPU acceleration.

4.2 Dataset

We need a fairly large dataset to train the CRNN. To avoid potential overlap, the dataset must have enough images.

With a resolution of 284 x 50 pixels, this dataset consists of two categories of clear and unclear images. The model was tested using a total of 413,70 handwritten images. For the training of the model and for validation of the model and the Pixel values represent the grayscale or color intensity of each pixel in the image, 70000 handwritten images are used and 7000 handwritten images are used for validation.