INTERNSHIP: PROJECT REPORT

Name of the Student	Christina Sharanyaa Selvakumari S
Internship Project Title	Automate identification and recognition of handwritten text from an image
Name of the Company	TCS iON
Name of the Industry Mentor	Ashutosh Tiwari
Name of the Institute	Panimalar Engineering College Chennai City Campus

Start Date	End Date	Total Effort (hrs.)	Project Environment	Tools used
16-Dec-2024	16-Mar-2025	210	Google Colab – Jupyter Notebook, Anaconda Navigator (Jupyter Notebook).	Kaggle Dataset platform (IAM dataset), Tensorflow 2.0, Keras – 2.3.0, Google Colab Virtual GPU or **NVIDIA Tesla K80** GPU

Project Synopsis:

- This project aims to create a machine learning model capable of detecting and interpreting handwritten data from an uploaded image. The solution enhances the accuracy and reliability of optical character recognition (OCR) systems. OCR technology involves converting images of handwritten, typed, or printed text into machine-readable text, which enables the storage, search, and manipulation of such data.
- This project combines Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) to improve the efficiency and accuracy of OCR. CNN is particularly effective in analyzing image-based patterns, while RNN handles the sequential recognition of characters. The objective is to extract handwritten data and convert it into a digital format for better accessibility and processing.

Goals and Purpose

The primary aim is to develop a robust machine learning model that can accurately extract handwritten characters from images. The model should:

- Accurately detect and recognize characters from uploaded images.
- Convert extracted data into editable and searchable text.
- Improve recognition accuracy through the combination of CNN and RNN.

Solution Strategy

This project addresses the problem of recognizing handwritten data using a Convolutional Recurrent Neural Network (CRNN). The CRNN model combines the strengths of CNN for image analysis and RNN for sequence prediction.

Implementation Steps:

1. Setting Up kaggle
2. Collecting Dataset
3. Preprocessing Data
4. Creating Network Architecture
5. Defining Loss Function
6. Training Model
7. Testing and Prediction

- Dataset **Preparation:**
    - The IAM dataset from Kaggle was used, containing 115,320 images of handwritten text.
    - A subset of 7,850 images was used for training and 876 images for validation.
- Preprocessing**:**
    - Images were converted to grayscale.
    - Image dimensions were adjusted to (128, 32) with padding.
    - Pixel values were normalized to fall between 0 and 1.
    - Text labels were encoded into numeric values and padded for consistency.
- Model **Architecture:**
    - Seven convolutional layers with increasing filter sizes (64 to 512).
    - Two max-pooling layers of size (2, 2) and (2, 1) to reduce spatial dimensions.
    - Batch normalization layers after the fifth and sixth convolution layers.
    - Lambda function to adjust CNN output for RNN compatibility.
    - Two bidirectional LSTM layers (128 units each) for sequence processing.
    - Final output layer using a dense layer for character classification.
- Loss **Function:**
    - The model used the Connectionist Temporal Classification (CTC) loss function.
    - CTC loss was implemented to handle variable-length sequences and align predicted characters with ground truth.
- Training**:**
    - Adam optimizer was used to minimize loss.
    - The model was trained on 7,850 training images and validated on 876 images.
    - Best weights were saved based on validation performance.

- Testing **and Evaluation:**

  - The model was tested on unseen images.
  - Output was decoded using the CTC decoder to generate readable text.
  - Jaro Distance & Ratio method was used to calculate accuracy.

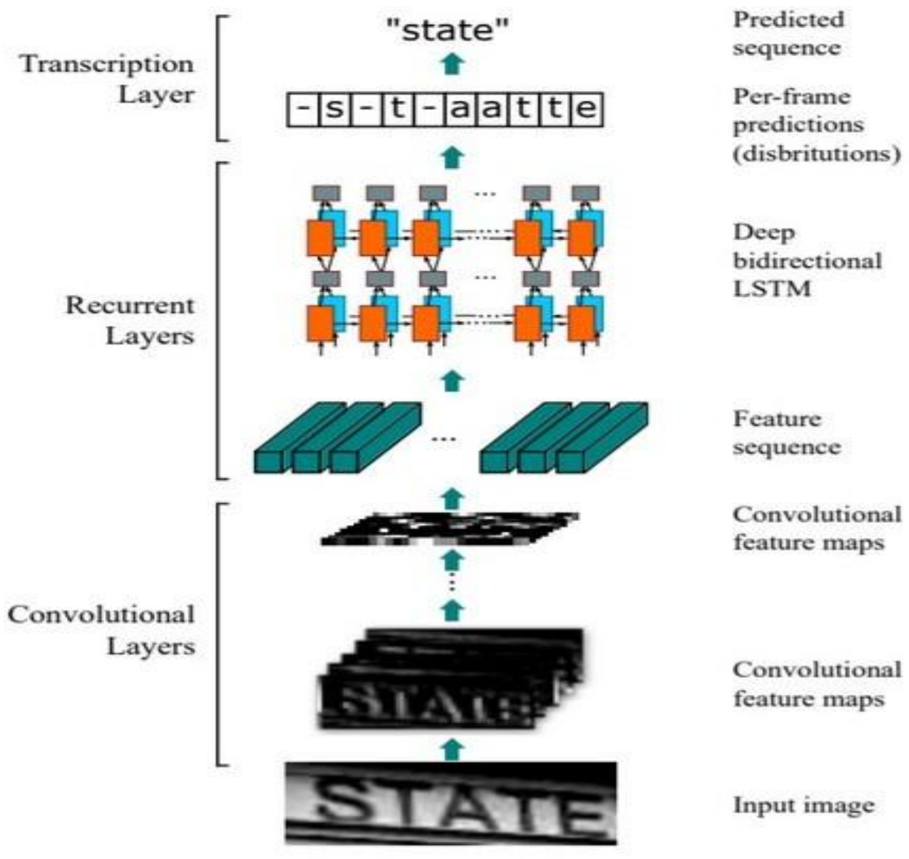
Assumptions

- **The handwritten content in the image should be in English.**
- **The image should not be rotated or distorted.**
- **The model expects input images only (no external data).**
- **All dependencies and libraries should be installed correctly.**

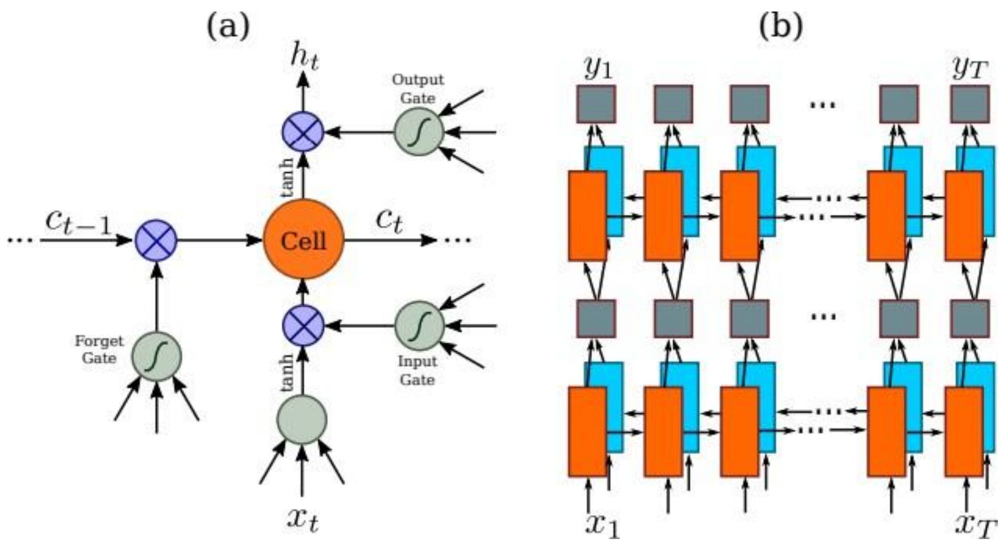
Model Design

The CRNN model follows a structured architecture with three main components:

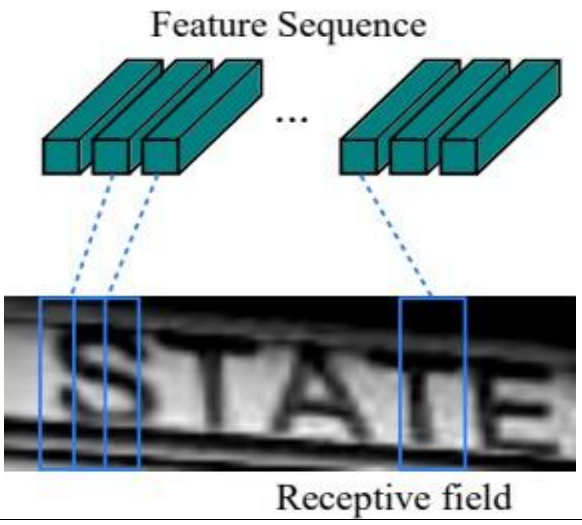
1. **Convolutional Layer:**
   - Extracts feature maps from the input image using seven convolutional layers.
   - Filter sizes increase progressively to enhance feature extraction.



2. **Recurrent Layer:**
   - Two bidirectional LSTM layers predict the sequence of characters.
   - Handles varying input sizes and complex character patterns.



3. **Transcription Layer:**
   - Converts the sequence predictions into readable text using CTC decoding.



Algorithm of CRNN Model:

1. Input shape for our architecture having an input image of height 32 and width 128.
2. Here we used seven convolution layers of which 6 are having kernel size (3, 3) and the last one is of size (2.2). And the number of filters is increased from 64 to 512 layer by layer.
3. Two max-pooling layers are added with size (2, 2) and then two max-pooling layers of size (2, 1) are added to extract features with a larger width to predict long texts.
4. Also, we used batch normalization layers after fifth and sixth convolution layers which accelerates the training process.
5. Then we used a lambda function to squeeze the output from conv layer and make it compatible with LSTM layer.
6. Then used two Bidirectional LSTM layers each of which has 128 units. This RNN layer gives the output of size (batch_size, 31, 63). Where 63 is the total number of output classes including blank character.

Outcome

The model achieved an efficiency rate of 91.72% on validation data. The model was able to accurately extract and recognize text from diverse handwritten samples.

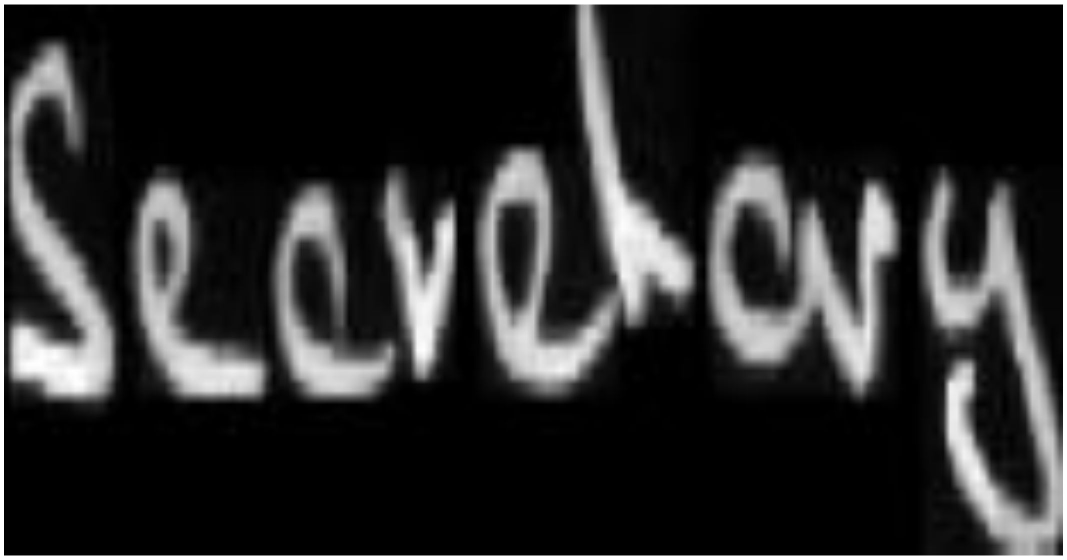
Example:

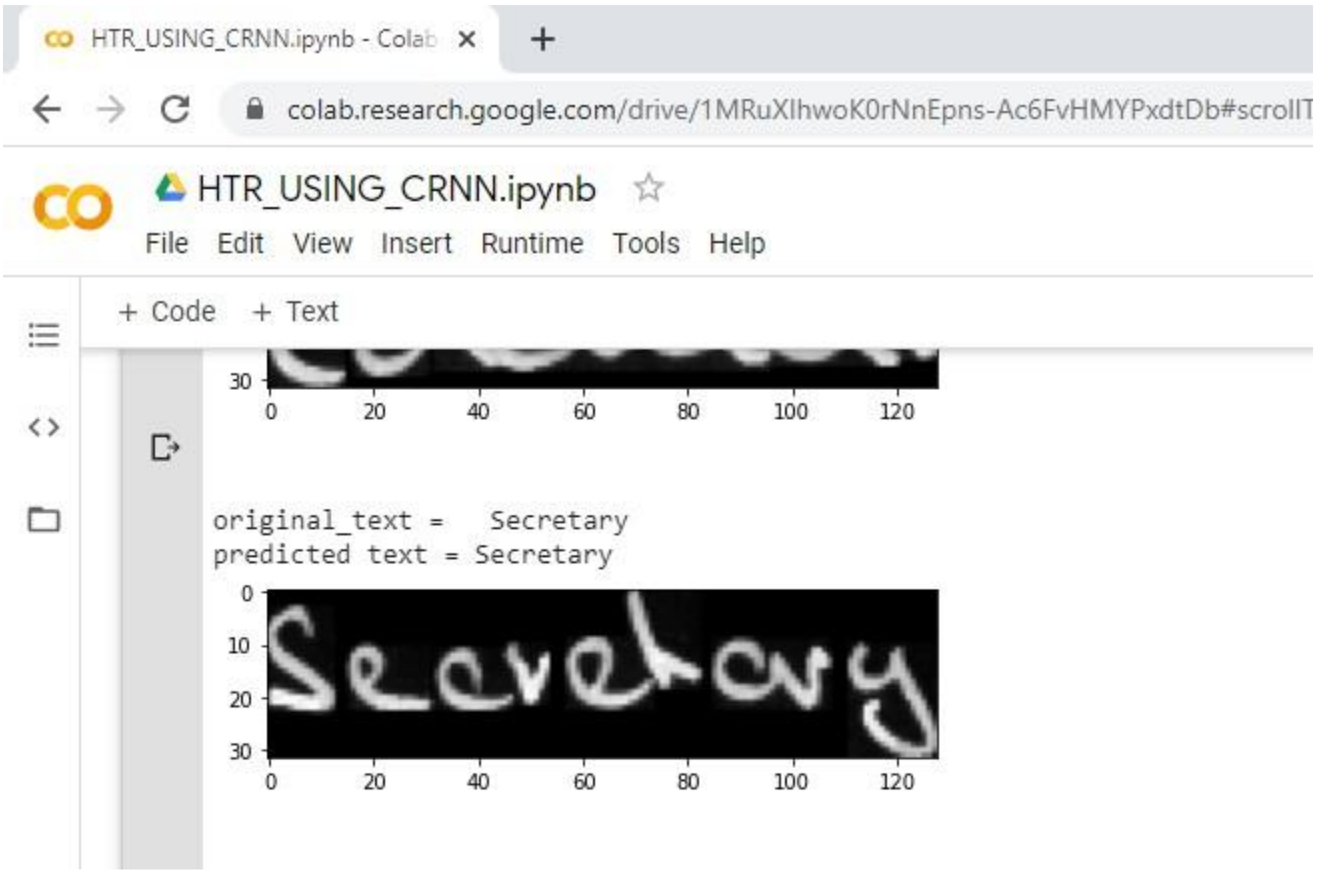
- **Input: Handwritten text image**
- **Output: Successfully transcribed text**

Sample Prediction:

Example 1:

- **Input: Handwritten image – "Secretary"**
- **Output: "Secretary"**

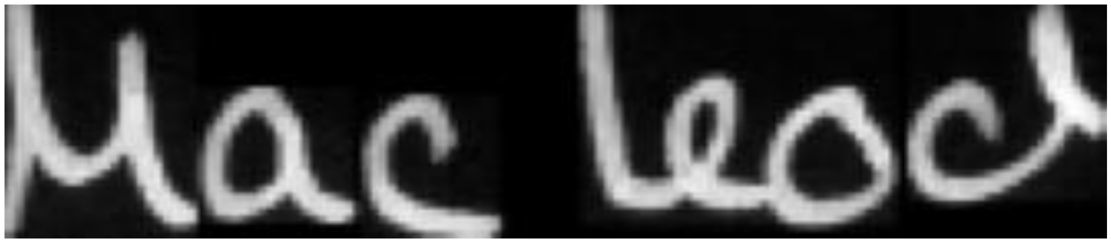


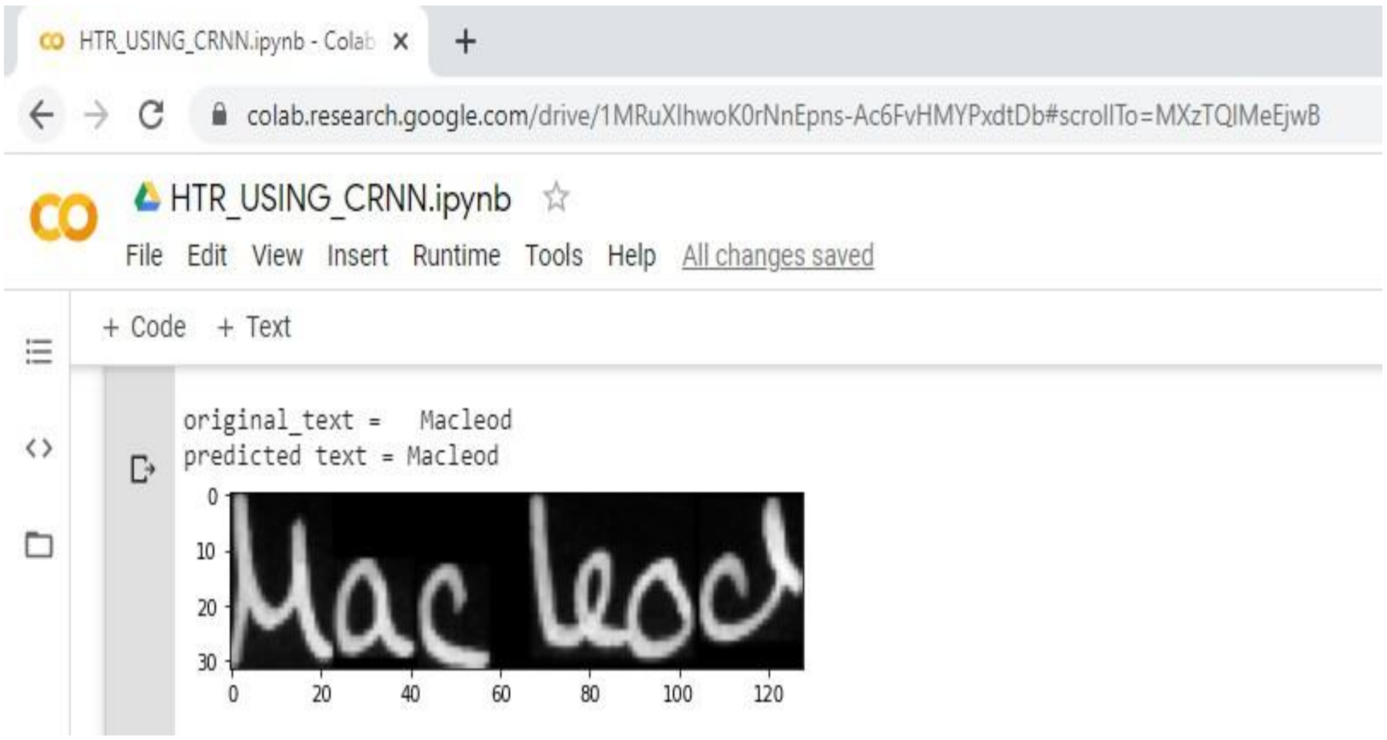

HTR_USING_CRNN.ipynb - Colab
colab.research.google.com/drive/1MRuXlhwoK0rNnEpns-Ac6FvHMYPxdtDb#scrollT
HTR_USING_CRNN.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text

```
original_text =   Secretary
predicted text = Secretary
```

Example 2:

- Input: Handwritten image – "Macleod"
- Output: "Macleod"

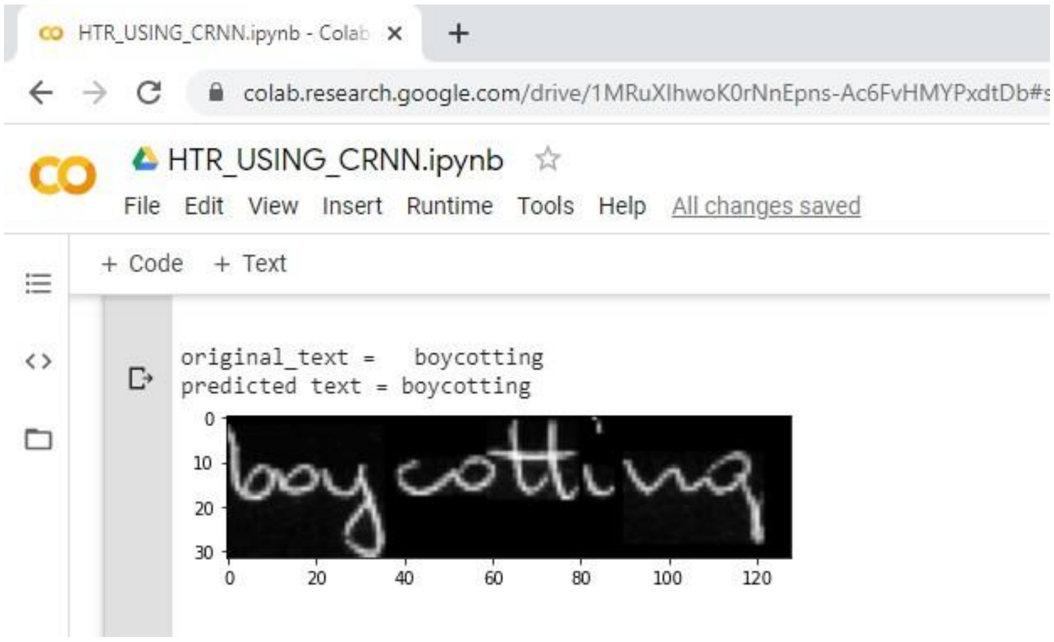



HTR_USING_CRNN.ipynb - Colab
colab.research.google.com/drive/1MRuXIhwoK0rNnEpns-Ac6FvHMYPxdtDb#scrollTo=MXzTQIMeEjwB
HTR_USING_CRNN.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
original_text =   Macleod
predicted text = Macleod
```

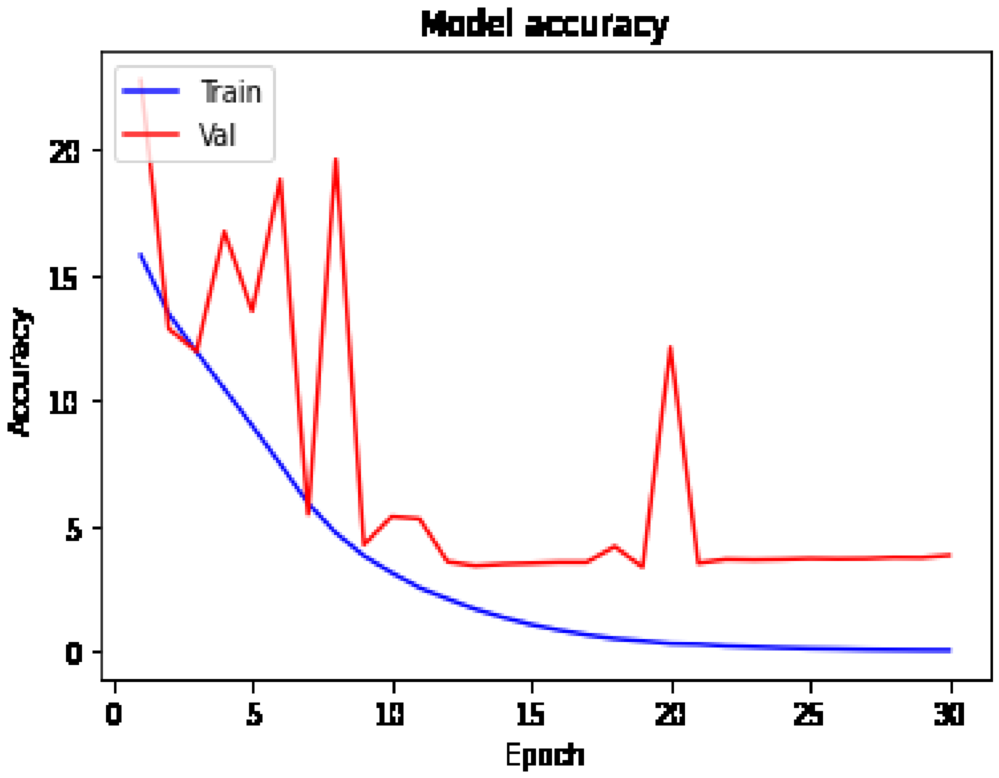
Example 3:

- Input: Handwritten image – "boycotting"
- Output: "boycotting"
-


HTR_USING_CRNN.ipynb - Colab
colab.research.google.com/drive/1MRuXIhwoK0rNnEpns-Ac6FvHMYPxdtDb#s
HTR_USING_CRNN.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
original_text =   boycotting
predicted text = boycotting
```

1) **Loss:**



Challenges and Solutions

Challenges:

- **Noisy and low-quality images affected accuracy.**
- **Varying handwriting styles required additional preprocessing.**
- **Alignment issues due to skewed text.**

Solutions:

- **Enhanced preprocessing to improve clarity and contrast.**
- **Used padding and normalization to handle varying input sizes.**
- **Data augmentation techniques improved generalization.**

Enhancement Scope: The enhancement scope of this project are follows:

1) The accuracy of the model can increased with predefined models and powerful machine learning GPU processors can be used to attain a good percentage of accuracy.
2) In future we can use this algorithm with more than one particular language.
3) This Model can be used in paragraph extraction if we increase the CNN layers and RNN layers and preprocess the data well.
4) This Model can be used in extraction of text from video if we can join CRNN and OpenCV concepts together.

References:

[1] https://arxiv.org/pdf/1507.05717.pdf
[2] https://software.intel.com/content/www/us/en/develop/training/course-artificial-intelligence.html
[3] https://software.intel.com/content/www/us/en/develop/training/course-machine-learning.html
[4] https://www.python-course.eu/machine_learning.php
[5] https://numpy.org/doc/
[6] https://software.intel.com/en-us/ai/courses/deep-learning
[7] https://www.tensorflow.org/tutorials/images/classification
[8] https://www.tensorflow.org/tutorials/text/text_classification_rnn
[9] https://www.tensorflow.org/tutorials/images/cnn
[10] https://www.tensorflow.org/tutorials/keras/classification
[11] https://www.tensorflow.org/tutorials
[12] https://pandas.pydata.org/pandas-docs/version/0.15/tutorials.html
[13] http://www.fki.inf.unibe.ch/databases/iam-handwriting-database/download-the-iam-handwriting-database
[14] https://towardsdatascience.com/setting-up-kaggle-in-google-colab-ebb281b61463
[15] http://www.fki.inf.unibe.ch/DBs/iamDB/data/words/
[16] https://towardsdatascience.com/a-gentle-introduction-to-ocr-ee1469a201aa
[17] http://ufldl.stanford.edu/housenumbers/
[18] http://yann.lecun.com/exdb/mnist/
[19] https://www.coursera.org/learn/neural-networks-deep-learning/home
[20] https://medium.com/@arthurflor23/handwritten-text-recognition-using-tensorflow-2-0-f4352b7afe16

Link to Code and executable file:

1) **GitHub link -** https://github.com/chrissharansel/Handwritten_text