|  |  |
| --- | --- |
| Name of the Student | SAUBHAGYAJEET SAHOO |
| Internship Project Title | AUTOMATE IDENTIFICATION AND RECOGNITION OF HANDWRITTEN TEXT FROM AN IMAGE |
| Name of the Company | TCS iON |
| Name of the Industry Mentor | ANAMIKA CHATTERJEE |
| Name of the Institute | DHANESWAR RATH INSTITUTE OF ENGINEERING AND MANAGEMENT STUDIES (DRIEMS), TANGI, CUTTACK, ODISHA |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Start Date | End Date | Total Effort (hrs.) | Project Environment | Tools used |
| 01/06/2020 | 01/08/2020 | 210 | PYTHON- 3.6, GOOGLE COLAB – Jupyter Notebook, Anaconda Navigator- Jupyter Notebook. | IAM Dataset, TensorFlow 2.2.0, Keras-2.3.0, Google Colab Virtual GPU, numpy-1.18.5. |
| **Project Synopsis:**   * This is TCS iON-RIO 210-Industry Project. This project is basically focused on identification and recognition of handwritten text from an image. It is based on enhancement of optical character recognition system. Optical Character Recognition (OCR) or Optical Character Reader is the electronic or mechanical conversion of images of handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo or from subtitle text superimposed on an image. An optical character recognition problem is basically a type of image-based sequence recognition problem. For an image-based problem most suited are convolution neural networks (CNN) while for sequence recognition problem, most suited neural networks are recurrent neural networks (RNN). To cop up with the OCR problems we need to combine both of this CNN and RNN. The purpose of OCR in text identification and recognition from an image is to extract handwritten data to digital data. So, one can easily handle this digital data by editing, adding new information in that text. OCR is developing field of Computer Vision, Pattern Recognition and Artificial Intelligence. The main objective of this project is to develop machine learning algorithm in order to enable entity and knowledge extraction from documents with handwritten annotations, with an aim to identify handwritten words on an image. | | | | |
| **Solution Approach:**  The solution provided in this project is highly inspired from the method that is described in the paper [DeepWriter: A Multi-Stream Deep CNN for Text-independent Writer Identification](https://arxiv.org/abs/1606.06472) . The main aim of the paper was to use a deep multi-stream CNN in order to predict hand written text. The images used in this project are the scanned images of various writers that are provided in the IAM Database.  The steps that are taken for the solution of the project are  1) Providing the samples of the Image and form that are required for making the database.  2) Creating a dictionary to store form ID and its writer.  3) Selecting the writers ID which will be used for making the dataset.  4) Selecting the form that are associated with the selected writers ID.  5) Creating a temporary directory which will store the writer’s image from the selected writer ID.  6) Creating arrays of the input file and visualize it.  7) Encoding writer’s ID.  8) Splitting the dataset into train, validation and tests sets.  9) Defining constants that will be used throughout the project.  10) Cropping the Inputted Data and converting it into array and visualizing sample from the training set.  11) Creating a generating method that will used to call the trainset while training.  12) Building a network architecture in Keras using CNN.  13) Training the model.  14) Checking the accuracy of the trained model.  The following steps are explained briefly below:  **1) Providing the samples of the image and form that are required for the making of the database**  **Step 1** - Download the data from the IAM Database “[http://www.fki.inf.unibe.ch/databases/iam-handwriting-database](http://www.fki.inf.unibe.ch/databases/iam-handwriting-database%22)”    **Step 2 -** Download the text file of the data you have downloaded from the ascii folder.    **Step 3** - Make a Folder  **Step 4** - Unzip the downloaded data into a folder inside the created folder.  **Step 5** - Move the “text file” to the folder.  The structure of the folder must look like    **2) Creating a dictionary to store form ID and its writer**  This is the first step of the program. The first step include the creation of dictionary which will map each form ID to a writer. This information is available in the “text file” that have been downloaded from the ascii folder in the IAM Database. Each line of the text file (except for the first 16 lines, which are documentation) defines the form ID at index 0, and its writer at index 1.    **3) Selecting the writers ID which will be used for making the dataset**  From the created dictionary we have to select a few writers in order to increase the efficiency of the program. In this program we have used 50 writers from the 221 writers present in the dataset.    **4) Selecting the form that are associated with the selected writers ID**  From the above selected writers, we need to select the images associated with them.    **5) Creating a temporary directory which will store the writer’s image from the selected writer ID**  Creating a temporary directory which will contain the images which are associated with the selected writers ID.    **6) Creating arrays of the input file and visualize it.**  Creating the array of the selected image in the temporary folder and the writer ID associated with it.    Visualizing the image    **7) Encoding writer’s ID.**  Encoding the writes with a value between 0 and n\_classes-1.    **8)** **Splitting the dataset into train, validation and tests sets**  Splitting the dataset into train, validation and test sets so that these sets could use for training and testing purpose.    **9) Defining constants that will be used throughout the project**  Defining some constants they are crop size, number of labels and batch size that will be used in the project.    **10) Cropping the Inputted Data and converting it into array and visualizing sample from training set**  As suggested in the paper, the input to the model are not unique sentences but rather random patches cropped from each sentence. The get\_augmented\_sample method is in charge of doing so by resizing each sentence's height to 113 pixels, and its width such that original aspect ratio is maintained. Finally, from the resized image, patches of 113x113 are randomly cropped.    Visualizing how the above method crops the image    Visualizing how many labels are created after cropping the original text.    Visualizing a sample created by cropping of the original text    Visualizing another sample of the cropped image of the original text    **11) Creating a generating method that will used to call the trainset while training**  The program uses a generator method in order to call sample of image created in the above method when training the model.      Creating training, validation and test generators    Importing Tensor Flow    Resizing the image to size    **12) Building a network architecture in Keras using CNN.**  Building a network in Keras using CNN      **13) Training the model**  The model is trained using GPU acceleration.      **14) Checking the accuracy of the trained model**  Selecting the best model from the trained models and checking the model for the accuracy.    For better accuracy fine tune model hyper parameters on the dataset. | | | | |
| **Assumptions:** The assumptions considered as follows:  1) The handwritten text across the image must be in English.  2) The image should not be tilted.  3) Only image is provided for text recognition.  4) All machine dependencies must be installed properly | | | | |
| **Project Diagrams:**   1. **Block Diagram**   CNN Model  Start  Output and Stop   1. **CNN Model**     **A Simple CNN Model**   1. **CNN model used in the project**     **CNN Used**    **A multi-stream CNN network** | | | | |
| **Algorithms:**  **Model = Image + CNN + CTC loss**  Our model consists of three parts  1) Image set  2) The Convolution Neural Network for extraction of Image  3) CTC loss function which is transcription layer used to predict output for each time step.  **Model Architecture**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Layer (type) Output Shape Param #  =================================================================  zero\_padding2d\_1 (ZeroPaddin (None, 115, 115, 1) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  lambda\_1 (Lambda) (None, 56, 56, 1) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  conv1 (Conv2D) (None, 28, 28, 32) 832  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  activation\_1 (Activation) (None, 28, 28, 32) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  pool1 (MaxPooling2D) (None, 14, 14, 32) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  conv2 (Conv2D) (None, 14, 14, 64) 18496  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  activation\_2 (Activation) (None, 14, 14, 64) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  pool2 (MaxPooling2D) (None, 7, 7, 64) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  conv3 (Conv2D) (None, 7, 7, 128) 73856  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  activation\_3 (Activation) (None, 7, 7, 128) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  pool3 (MaxPooling2D) (None, 3, 3, 128) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  flatten\_1 (Flatten) (None, 1152) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dropout\_1 (Dropout) (None, 1152) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dense1 (Dense) (None, 512) 590336  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  activation\_4 (Activation) (None, 512) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dropout\_2 (Dropout) (None, 512) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dense2 (Dense) (None, 256) 131328  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  activation\_5 (Activation) (None, 256) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dropout\_3 (Dropout) (None, 256) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  output (Dense) (None, 50) 12850  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  activation\_6 (Activation) (None, 50) 0  =================================================================  Total params: 827,698  Trainable params: 827,698  Non-trainable params: 0  **Strategy used in the program**  1) Design and optimize multi-stream structure for writer identification task.  2) Introducing data augmentation learning to enhance the performance of the program.  3) Introducing a patch scanning strategy to handle text image with different lengths. | | | | |
| **Outcome:**  The algorithm is able to detect and segment handwritten text from an image. The model successfully able to detect maximum words in a given line of sentence or words, which makes it about 90% accurate while implementation and testing. | | | | |
| **Exceptions considered:** The exceptions considered are as follows:  1) The text across the input image must be of the same color not multicolor handwritten text.  2) The image doesn’t have too aggressive multicolor backgrounds across the text of the image.  3) The image doesn’t have any kind’s objects in the background across the text of the image.  4) The image should not be tilted or rotated. | | | | |
| **Enhancement Scope:** The enhancement scope of this project are follows:  1) The accuracy of the model can increase 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) If we increase the number of epochs, we can get a better accuracy.  4) This Model can be used in paragraph extraction if we increase the CNN layers and RNN layers and preprocess the data well.  5) This Model can be used in extraction of text from video if we can join CNN and OpenCV concepts together. | | | | |
| **References:**  1) <https://arxiv.org/pdf/1606.06472.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>  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](%20http:/www.fki.inf.unibe.ch/databases/iam-handwriting-database/download-the-iam-handwriting-database)  14) <https://towardsdatascience.com/a-gentle-introduction-to-ocr-ee1469a201aa>  15) <https://link.springer.com/article/10.1007/s11036-019-01243-5>  16) <https://medium.com/@tomhoag/opencv-text-detection-548950e3494c>  17) <https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_ml/py_knn/py_knn_opencv/py_knn_opencv.html>  18) <https://github.com/RiteshKH/Cursive_handwriting_recognition>  19) <https://www.pyimagesearch.com/2014/09/22/getting-started-deep-learning-python/>  20) <http://hanzratech.in/2015/02/24/handwritten-digit-recognition-using-opencv-sklearn-and-python.html>  21) <https://www.learnopencv.com/deep-learning-based-text-recognition-ocr-using-tesseract-and-opencv/> | | | | |
| **Link to Code and executable file:**  **Link to GitHub -** <https://github.com/SaubhagyajeetSahoo/TCSiON-Hand-Writting-Extraction-from-an-Image>  **GitHub Main Project File -** <https://github.com/SaubhagyajeetSahoo/TCSiON-Hand-Writting-Extraction-from-an-Image/tree/master/Main%20Project%20file>  **Google Colab Link -** <https://colab.research.google.com/drive/1wrFa5CAGEfQH_J-yBh0KjVrUm5PRI_Ga?usp=sharing> | | | | |