HANDWRITTEN CHARACTER RECOGNITION(HCR)

A DISSERTATION submitted

by

THAKKAR ADITYA M.

THAKKAR SAGARKUMAR D.

VALA SAGAR

For the Software Development Project for Semester –VI of

BACHELOR OF TECHNOLOGY
(COMPUTER ENGINEERING)

Under the guidance of

Prof. Ashish K. Gor

Department of Computer Engineering
Faculty of Technology
DHARMSINH DESAI UNIVERSITY
NADIAD-387001

CERTIFICATE

This is to certify that the project entitled as “HANDWRITTEN CHARACTER RECOGNITION(HCR)” is a

Bonafede report of the work carried out by:

1] Aditya Thakkar, Roll No: CE-158, Id: 19CEUOS131

2] Sagar Thakkar, Roll No: CE-159, Id: 19CEUOG026

3] Sagar Vala, Roll No: CE-174, Id: 19CEUOG081

of Department of Computer Engineering, semester VI, under the guidance and supervision of Prof. Ashish K Gor for the subject Software Development Project during the academic year 2021-2022.

Prof. Ashish K. Gor
Guide,
Dept. of Computer Engg.,
Faculty of Technology,
Dharmsinh Desai
University,
Nadiad.

Dr. C. K. Bhensdadia
Head of Department,
Dept. of Computer Engg.,
Faculty of Technology,
Dharmsinh Desai
University,
Nadiad.

Table of Contents

1. Introuction

Handwritten character recognition (HCR) is the electronic conversion of images of handwritten text into machine-encoded text from a scanned document. It transcribes handwritten text written in English language contained in scanned images into digital text.

User interface is created where user need to upload scanned handwritten text document written in English handwriting and the system will generate the predicted digital text of the same.

HCR consists of several parts, which are responsible for processing pages with full text (scanned or photographed) dividing them into lines, splitting the resulting lines into words, segmenting the resulting words into characters and following recognition of characters from them.

The segmented characters are passed into a deep learning model which predicts the character and then each character of word is again assembled to form a word from where it got segmented. These assembled words are then assembled to form a line from where it got segmented. And finally these lines are assembled to form a digital documented text. Thus, digital documented text is shown as output beside the handwritten scanned document on the user interface.

There are many applications of Handwritten character recognition like:

- Quick digital searches : HCR system converts scanned text into word processing file so the search for specific documents using a keywords or phrase is simple.
- Save space: Free up storage space by scanning paper documents and hauling the originals off to storage.
- HCR is widely used in many other fields, including education, finance and government agencies.
- HCR has made countless texts available online, saving money for students and allowing knowledge to be shared.

Scope of this project is to detect simple handwritten English characters from segmented characters with maximum accuracy achievable.

2. Literature

Handwritten digit recognition is an important problem in optical character recognition and it has been used as a test case for theories of pattern recognition and machine learning algorithms for many years.

It can be classified into two categories: online recognition and offline recognition. The on-line recognition technology, which emerges in recent years, uses the geometry and temporal dynamics information of the users' input. The methods for online recognition relatively pose low resource and processing requirement, and may effectively use many kinds of clues to capture users' input customs. They are effective with good user adaptation.

Inversely, Offline Recognition mainly processes and recognizes the user input handwritten digit based on images (the scanned images of handwritten digit, or the digital images transformed from the real time handwritten). A lot of methods have been proposed to solve offline recognition.

In this paper, we focus on Offline Recognition. Many methods has been used for HCR like Neuro-Fuzzy Systems (NFS), Artificial Neural Network (ANN), Support Vector Machine (SVM) and deep learning-based classifiers. All the classifiers are providing good accuracy but still there is a lot to explore in the field of HCR to further improve the performance. The performance parameters used to find the performance of classifiers are accuracy, running time and computational complexity. The important factor in CNN model is that it fully use the topological information as well as it is invariant to basic transformations like rotation, translation etc.

HCR Models

The popular models which are used for HCR are KNN (K nearest Neighbors), SVM (Support Vector Machine), NN (Neural Networks).

A. KNN (K nearest Neighbors)

KNN is used to solve regression problems and also used as a classifier. In KNN classifier, since computations are calculates up to the end stage that's why it is also called as late learning classification algorithm. And

since all the computation occur locally; it is also called as instance-based classification algorithms. There is no training required earlier in KNN classifier, as well as there is no generalization is performed on training data. KNN algorithm describes categorical value by making use of majority of votes of K - nearest neighbors, the K value used to differ here. It is found that votes value changes with change in K value.

B. SVM (Support Vector Machine)

Support Vector Machine is a kind of supervised learning. It can be used for both regression problems and for classification purpose. SVM make use of optimal hyper plane that can be utilized to divide it into multiple categories. For 2D spaces, independent variable data points are plotted which are corresponds to dependent variables. After it, classification is started to find hyper plane or linear/nonlinear plane which is used to categorize class.

C. NN (Neural Networks)

The Neural Networks methodology is inspired from working of brain. It became popular in the field of computational power. NN is termed as Deep learning where multilayers are connected together to form a network. Nodes are formed by using these layers. Each node is subjected to execute some computation. This then input into node's activation function, in order to show context signal progress into the network for classification purpose.

i. CNN(Convolutional Neural Network)

The CNN classifier is very popular to perform HDR. CNN is a multi-layer convolutional network where there is one input layer and more than one hidden layers and at last one output layer.

CNN is kind of ANN with feed forward method. Connectivity in CNN is inspired by the organization of the animal visual cortex. CNN has many neurons that carry two parameters which are learnable weights & biases. Some input is provided to each neuron and it performs operations like dot product and performs it with non-linearity.

A. Layers of CNN

There are many layers in CNN. Deep leaning is a use of all these layers in iteration. The 3 category of CNN layers can be explained as:

1) Input Layer: the raw pixel values of digital image are carried by input layer.

2) Convolutional Layer: A result of input block neuron layer is supplied into convolution layer. There are many filters defines for this layer by user. The filter with some window size is used for input pixels and it gives out the highest intensity pixels in output.

3) Rectified Linear Unit Layer: the function of this layer to pass image pixel into activation function element wise. Back propagation is applied in CNN, this leads to change in values of image pixel, in order to rectify it ReLU function is applied.

4) Pooling Layer: down sampling is performed on spatial dimensions like width and height by this layer. The output comes in the form of volume.

5) Fully Connected Layer: computation of score of classes is performed at this layer. It computes maximum of score secured by input.

3. Algorithms used for each process:-

1) ***Segmentation of line*s**:
   Projection Profile method

2) ***Segmentation of words and characters***:
   Implementation of Scale-space technique for word segmentation as proposed by R. Manmatha and N. Srimal. Even though the paper is from 1999, the method still achieves good results, is fast, and is easy to implement. The algorithm takes an image of a line as input and outputs the segmented words.

3) ***Deep Machine Learning Model***:
   For recognition of handwritten characters, it was decided to use Convolutional Neural Network (CNN).

4. Datasets, Tools & Technologies Used

The tool and technologies used for Handwritten Character Recognition(HCR) project is:

Tools:

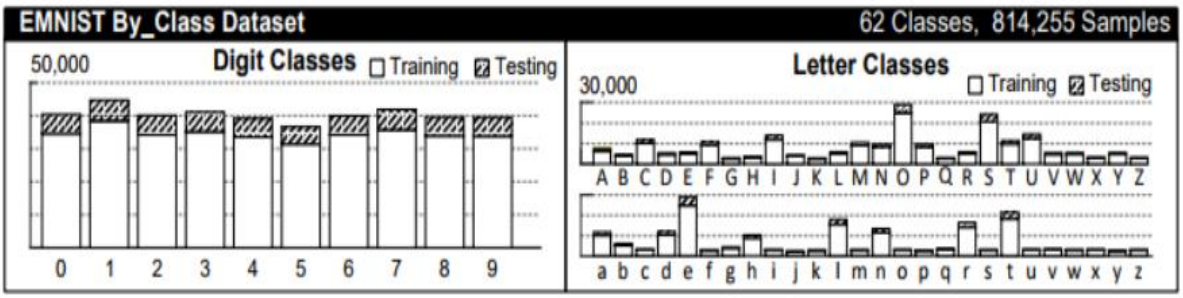
- Graphical User Interface: Django
- Programming Language: Python, OpenCV Python

Technologies:

- Google Colab
- Jupyter Notebook
- Visual Studio Code

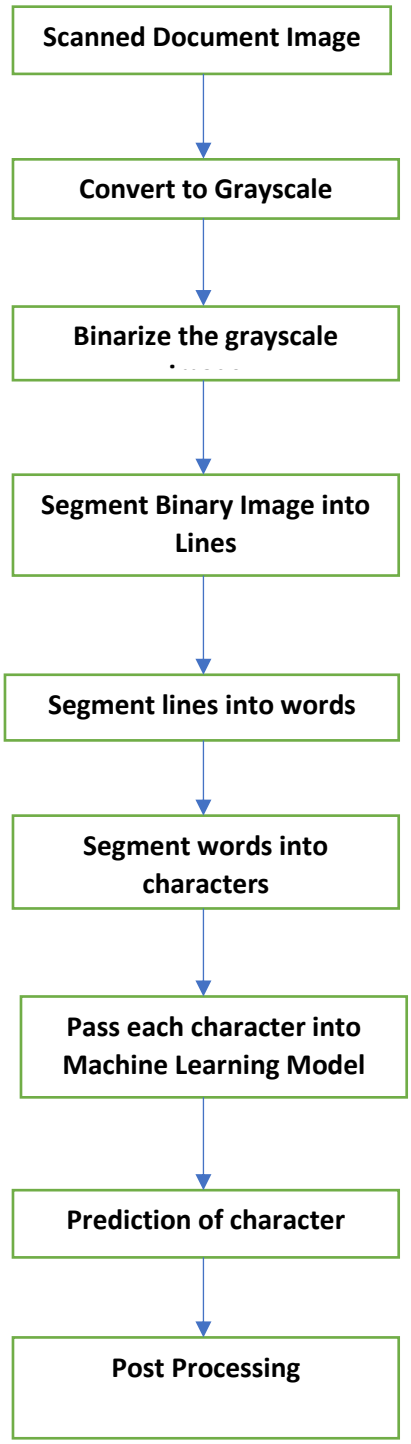
Dataset:

- ***EMNIST By-Class*** Dataset is used for training CNN model.
- It has 62 number of classes i.e.
  0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z
  a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z
- Number of samples in this dataset are: 8,14,255



5. Implementation Details

Workflow:-



Following is pseudo code for Handwritten Character Recognition:

I. **LineSegmentation(image):**

Input: Scanned Document Image

Output: Images of Segmented Lines

Procedure:

1) Convert image to grayscale image
2) Create Anisotropic diffusion Kernel filter.
3) Filter the image by convolving Kernel with grayscale image.
4) Transpose the image.
5) Find Vertical Projection Profile of the image for distribution of pixels.
6) Find local minima.
7) Line is formed by two consecutive local minima.
8) Save all the lines.

Explanation of Pseudo Code:-

An anisotropic diffusion Kernel is created to reduce image noise without removing significant parts of the image content, typically edges, lines or other details that are important for the interpretation of the image.

So, this kernel is convolved with the converted grayscale image which is achieved by *cv2.filter2D( )* function for filtering the image.

After this, below set of functions are executed in order:

Filtered image is transposed to apply Vertical Projection profile where sum of each column is taken to know the distribution of pixels. Wherever, the sum is low it indicates the space between lines or no text which is the parameter by which line is segmented. The local minima are found by the function *argrelmin( )* function and each line is segmented by: starting of a line is one local minimum and ending of line is another local minimum and so on for all the lines.

II. WordSegmentation(line_image):

Input: Scanned Document Image

Output: Images of Segmented Words

Procedure:

1) Convert line image to grayscale image
2) Create Anisotropic diffusion Kernel filter.
3) Filter the image by convolving Kernel with grayscale image.
4) Apply thresholding to image.
5) Find connected components by detecting the contours.
6) Form the words by iterating over all the contours found.
7) Save all the words.

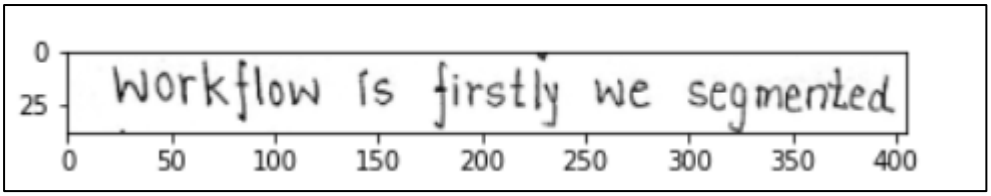
Explanation of Pseudo Code:-

Each segmented line is again applied filter kernel. Filtered image of line is then applied thresholding. What is thresholding? Thresholding is a very popular segmentation technique, used for separating an object considered as a foreground from its background. A threshold is a value which has two regions on its either side i.e. below the threshold or above the threshold.

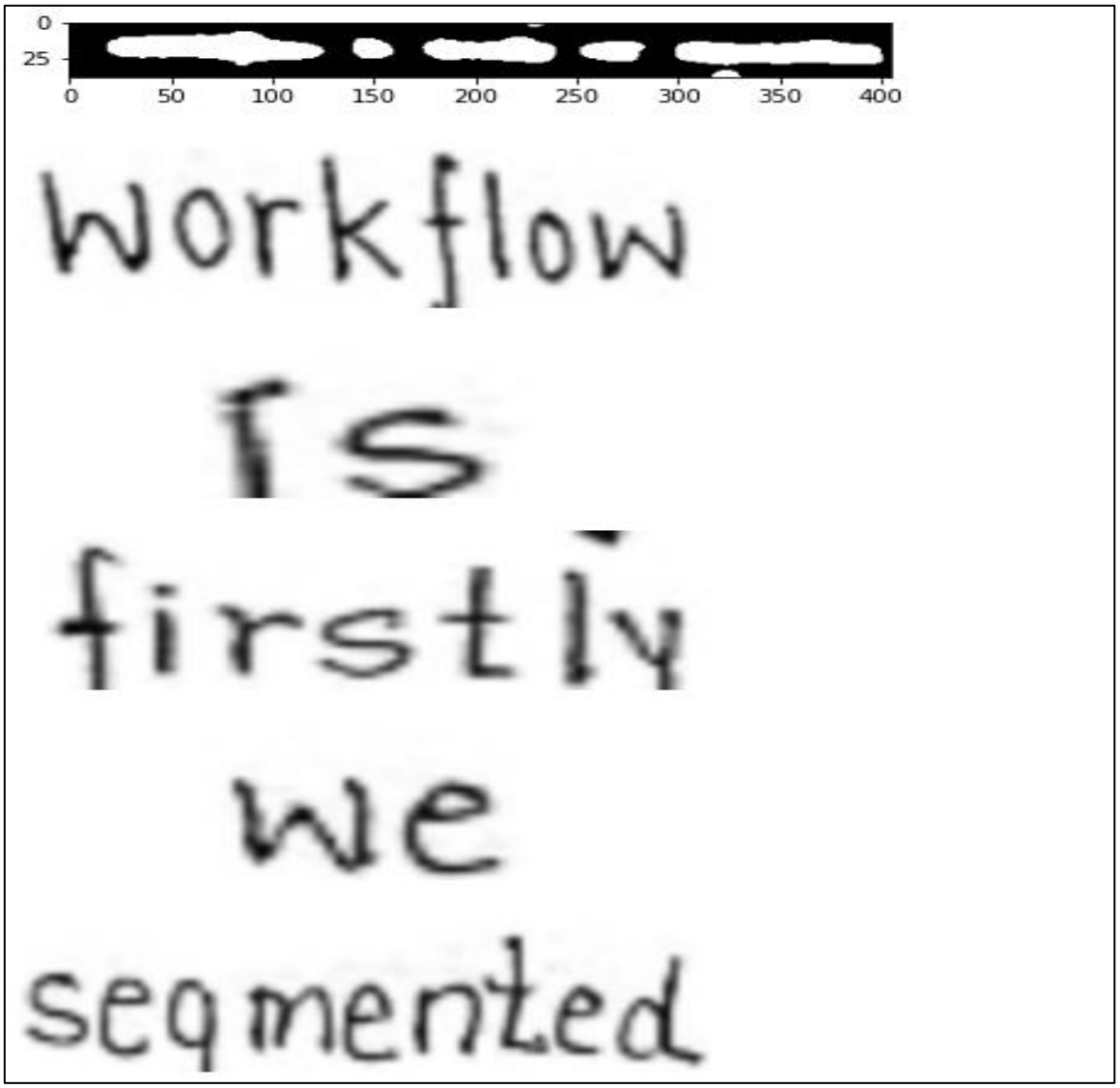
In OpenCV with Python, the function used for thresholding is:
cv2.threshold(imgFiltered, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
After thresholding, the characters which are near to each other and form a word gets same pixel value and if there is space between them will get 0-pixel value.

So, in a sense we get words as a set of connected components. Now, to find words we ne ed to find connected components in the line image which is found by contour detection method i.e. cv2.findContours(). All The words are formed by the x co-ordinate, y co-ordinate, width ,and height of contour found.

Filtered image of line:-



After Thresholding:-



III. CharacterSegmentation(word_image):

Input: Scanned Document Image

Output: Images of Segmented Characters

Procedure:

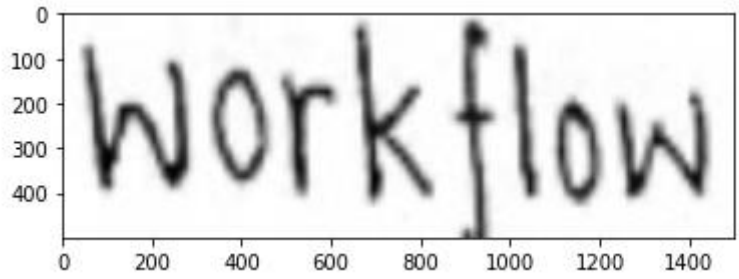
1) Convert line image to grayscale image
2) Resize the image to a greater height and width.
3) Create Anisotropic diffusion Kernel filter.
4) Filter the image by convolving Kernel with grayscale image.
5) Apply thresholding to image.
6) Find connected components by detecting the contours.
7) Form the characters by iterating over all the contours found.
8) Save all the characters.

Explanation of Pseudo Code:-

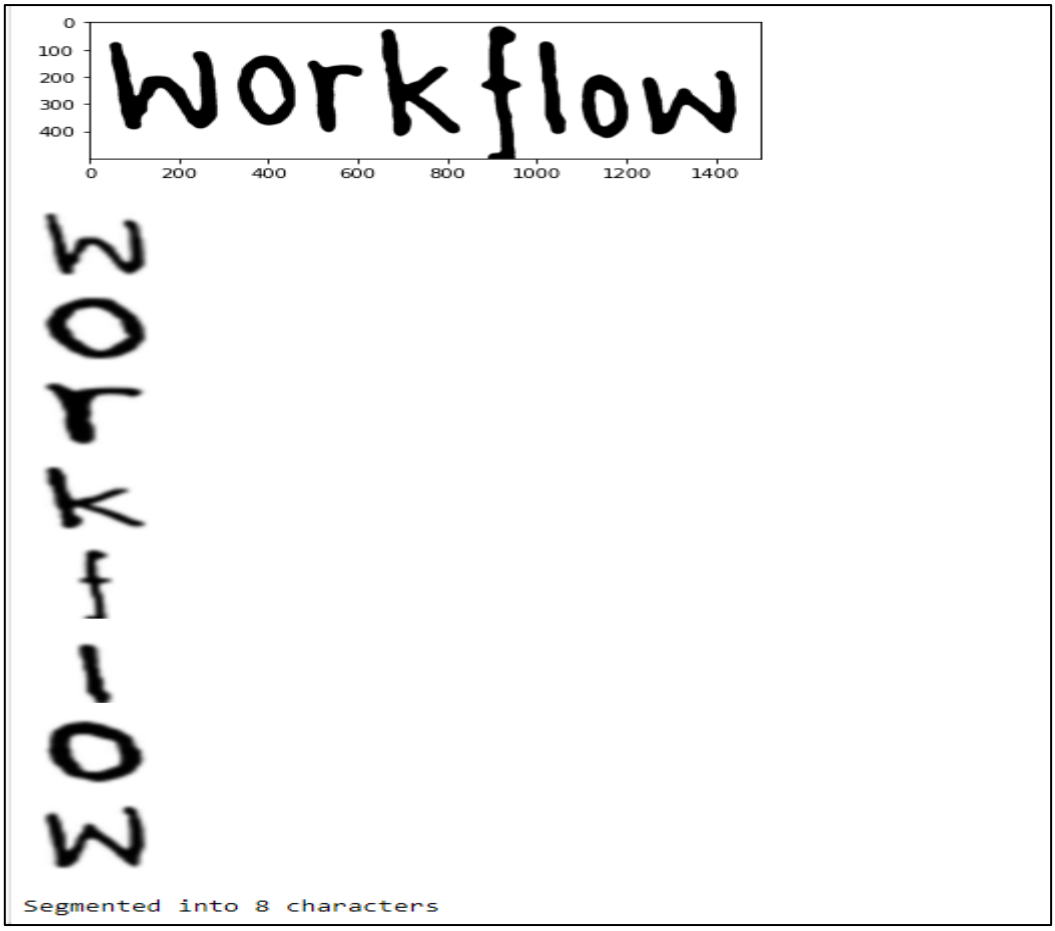
Character segmentation is done exactly in same manner as of word segmentation but only pre-processing which is done is the word image is resized to a greater height and width so after thresholding each character remains non-connected to other character and remain as single connected component. And after that process remains same of detecting each character by contour detection explained in above method.

Example:

Filtered image



After Thresholding:-



IV. Deep Learning Model – CNN:

Convolutional Neural Networks:

Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. They have three main types of layers, which are:

- Convolutional layer
- Pooling layer
- Fully-connected (FC) layer

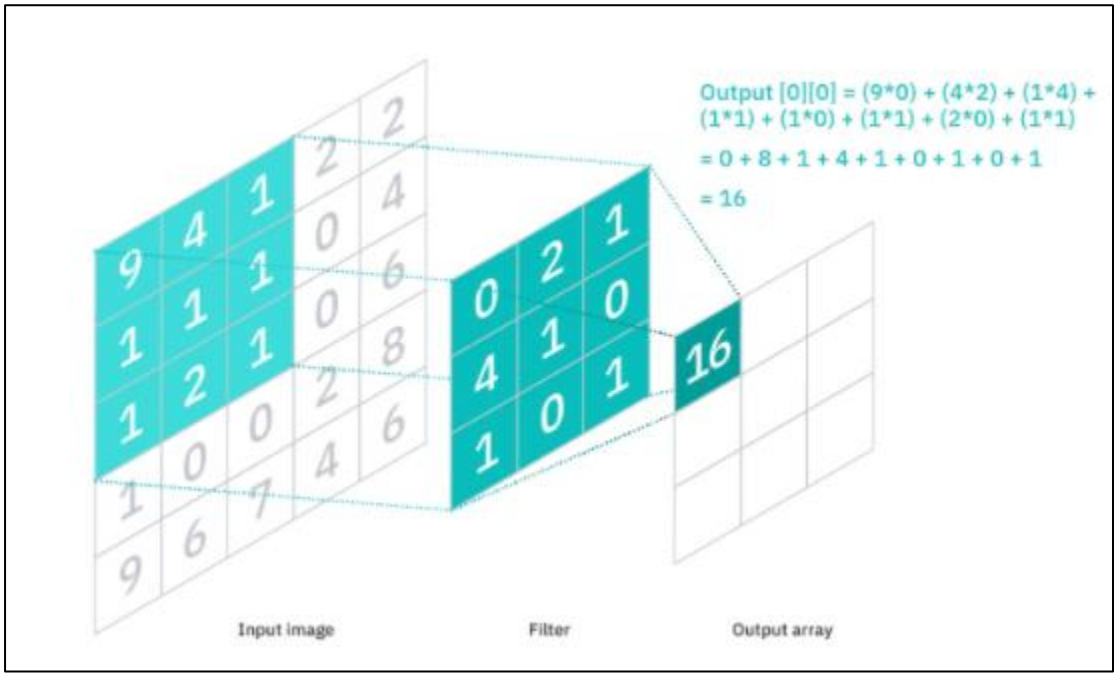
Input layer size is of (28 , 28 , 1) for EMNIST dataset .

Convolutional Layer :

- The convolutional layer is the first layer of a convolutional network. While convolutional layers can be followed by additional convolutional layers or pooling layers, the fully-connected layer (dense layer) is the final layer. With each layer, the CNN increases in its complexity, identifying greater portions of the image. Earlier layers focus on simple features, such as colours and edges. As the image data progresses through the layers of the CNN, it starts to recognize larger elements or shapes of the object until it finally identifies the intended object.

- The convolutional layer is the core building block of a CNN, and it is where the majority of computation occurs. It requires a few components, which are input data, a filter, and a feature map. Let's assume that the input will be a colour image, which is made up of a matrix of pixels in 3D. This means that the input will have three dimensions—a height, width, and depth—which correspond to RGB in an image. We also have a feature detector, also known as a kernel or a filter, which will move across the receptive fields of the image, checking if the feature is present. This process is known as a convolution.

- The feature detector is a two-dimensional (2-D) array of weights, which represents part of the image. While they can vary in size, the filter size is typically a 3x3 matrix; this also determines the size of the receptive field. The filter is then applied to an area of the image, and a dot product is calculated between the input pixels and the filter. This dot product is then fed into an output array. Afterwards, the filter shifts by a stride, repeating the process until the kernel has swept across the entire image. The final output from the series of dot products from the input and the filter is known as a feature map, activation map, or a convolved feature.



<u>Pooling Layer</u> :

- Pooling layers, also known as down-sampling, conducts dimensionality reduction, reducing the number of parameters in the input. Similar to the convolutional layer, the pooling operation sweeps a filter across the entire input, but the difference is that this filter does not have any weights. Instead, the kernel applies an aggregation function to the values within the receptive field, populating the output array. There are two main types of pooling: Max Pooling and Average Pooling. We have used Max Pooling.

- **Max pooling:** As the filter moves across the input, it selects the pixel with the maximum value to send to the output array. As an aside, this approach tends to be used more often compared to average pooling.

- While a lot of information is lost in the pooling layer, it also has a number of benefits to the CNN. They help to reduce complexity, improve efficiency, and limit risk of overfitting.

Fully-Connected Layer :

- The name of the full-connected layer aptly describes itself. As mentioned earlier, the pixel values of the input image are not directly connected to the output layer in partially connected layers. However, in the fully-connected layer, each node in the output layer connects directly to a node in the previous layer.

- This layer performs the task of classification based on the features extracted through the previous layers and their different filters. While convolutional and pooling layers tend to use ReLu functions, FC layers usually leverage a softmax activation function to classify inputs appropriately, producing a probability from 0 to 1.

Dropout Layer **:**

- It is used to **fix the over-fitting issue**. Input data may have some of the unwanted data, usually called as Noise. Dropout will try to remove the noise data and thus prevent the model from over-fitting.

```
keras.layers.Dropout(rate, noise_shape = None, seed = None)
```

- **rate** − represent the fraction of the input unit to be dropped. It will be from 0 to 1. We have used rate=0.2.

- **noise_shape** represent the dimension of the shape in which the dropout to be applied. For example, the input shape is **(batch_size, timesteps, features).** Then, to apply dropout in the timesteps, **(batch_size, 1, features)** need to be specified as **noise_shape**
- **seed** − random seed.

Flatten layer:

- Flatten Layer As its name suggests, Flatten Layers is used for **flattening of the input**. For example, We have an input shape as (batch_size, 28,28), after applying the flatten layer, the output shape is changed to (batch_size,784).

RELU Activation Function :

- RELU is more well-known activation function which is used in the deep learning networks. RELU is less computational expensive than the other non-linear activation functions.
- RELU returns 0 if the x (input) is less than 0
- RELU returns x if the x (input) is greater than 0

```
def RELU(x):
    ''' It returns zero if the input is less than zero otherwise it returns the given
input. '''
    x1= []
    for i in x:
        if i<0:
            x1.append(0)
        else:
            x1.append(i)

    return x1
```

Softmax Activation Function :

- Softmax turns logits, the numeric output of the last linear layer of a multi-class classification neural network into probabilities.

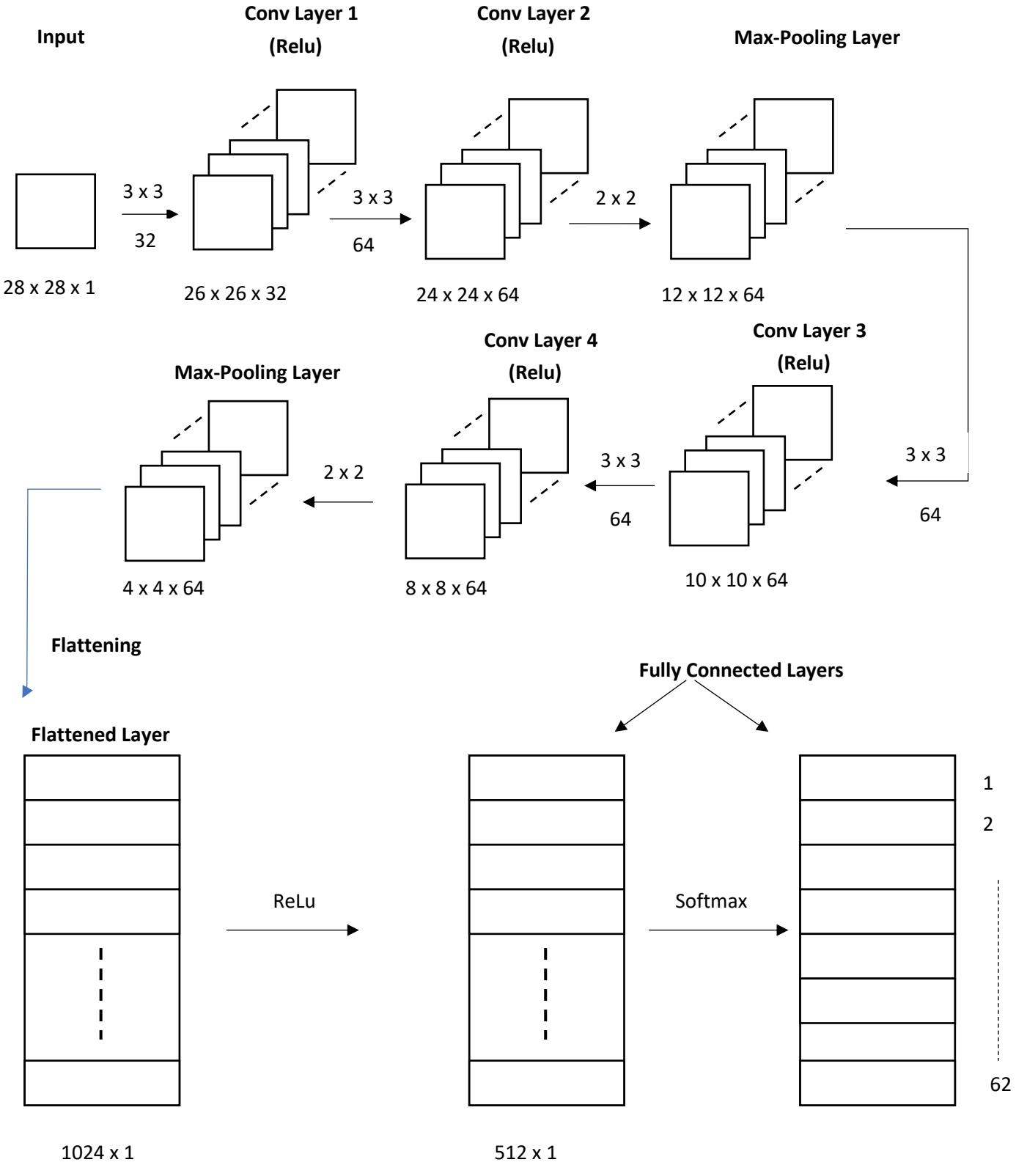
```
def softmax(x):
    ''' Compute softmax values for each sets of scores in x. '''
    return np.exp(x) / np.sum(np.exp(x), axis=0)
```

CONVOLTUTIONAL NEURAL NETWORK

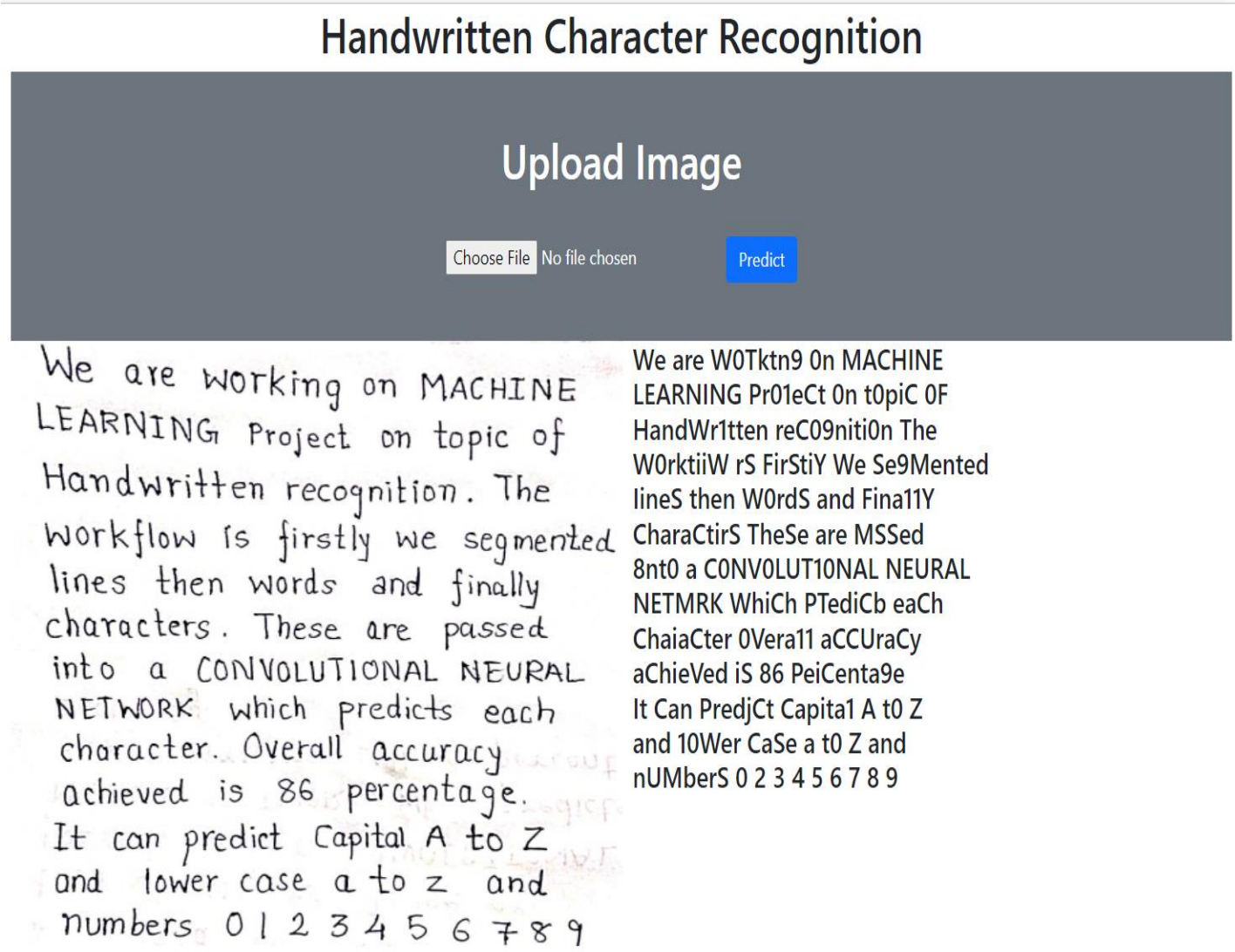
```
number_of_classes = 62

from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

model = Sequential()

model.add(Conv2D(filters = 32, kernel_size = (3, 3), activation = 'relu',
input_shape = (28, 28, 1)))
model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu'))
model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Dropout(.2))

model.add(Flatten())

model.add(Dense(units = 512, activation = 'relu'))
model.add(Dropout(.2))
model.add(Dense(units = 62, activation = 'softmax'))

model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accur
acy'])
```

Accuracy achieved : 87% on EMNIST dataset on training and testing both.

Architecture of CNN Model:-



6. Testing Results

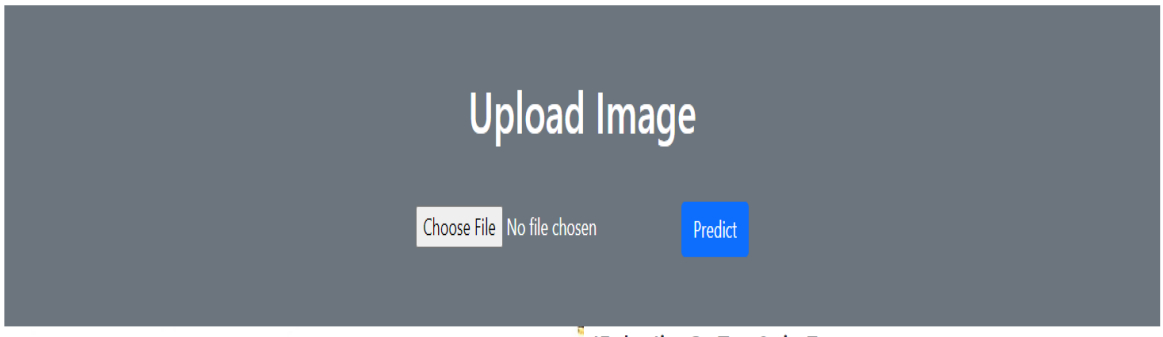


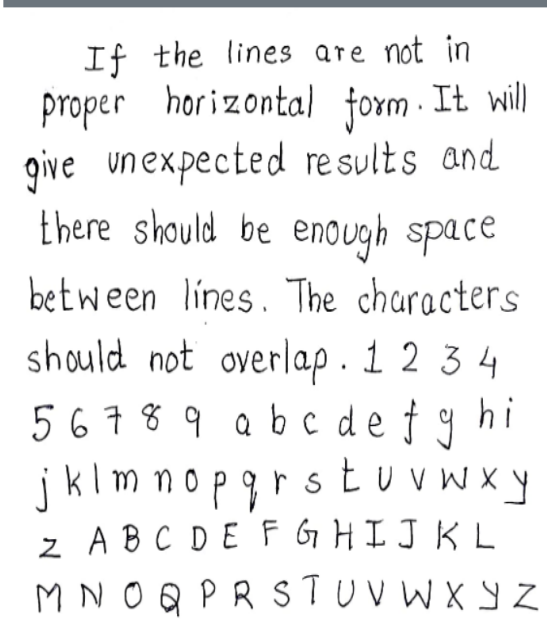
Correctly Predicted Characters:-267

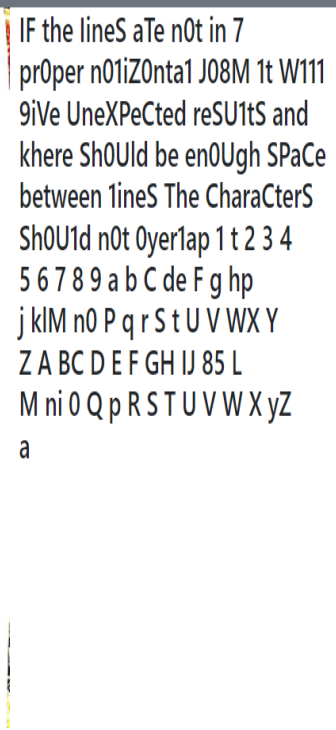
Total Characeters:299

Accuracy=267/299=89%

Handwritten Character Recognition







Correctly Predicted Characters:-180

Total Characeters:186

Accuracy=267/299=96%

7. Conclusion

Functionalities that are successfully implemented in the system are:

- Segmentation of lines, words and characters is accurately achieved if scanned image document is clearly visible and less noisy and if there is proper space between lines, words and characters.
- Overall Prediction of characters achieved is 87%.

8. Limitations and Future Enhancements

Limitations of the system are mentioned as below:-

- Scanned document should be as less noisy as possible.
- User need to upload scanned handwritten text document written in simple English handwriting (cursive or other styles/fonts are not accepted).
- Lines must not be so much crooked and so should be as horizontal as possible.
- If some characters overlap(as in cursive style), accuracy will decrease and result in false predictions of character.
- If image is noisy some dummy characters may be generated.
- Characters which are similar like I(capital I) and 1(one) and small l(el), small s and capital S, zero 0 and capital O, small c and capital C can be predicted interchangeably.

The system can be extended to support not just simple English font style but also cursive style and new model can be developed for training cursive handwritten characters in addition to simple handwritten characters.
For now, model is trained on EMNIST dataset but can be trained by creating new dataset which is developed by taking images of handwritten characters of different persons so that variety of handwritten styles can be trained and accuracy can be increased.
Characters mentioned above which are predicted interchangeably can be predicted correctly by developing a model where characters are predicted based on the context i.e. seeing past characters and then deciding what could be the character.

9. Bibliography

Following links and websites were referred during the development of this project:

- https://www.kaggle.com
- https://towardsdatascience.com