# DESIGN OF ODIA HANDWRITTEN ALPHABET RECOGNITION SYSTEM USING

# CONVOLUTIONAL NEURAL NETWORK

**Ashpo Ghosh**

**&**

**MD Sajid Sahanawaz**



Computer science & Engineering

Veer Surendra Sai University Of Technology, Burla

2019

# DESIGN OF ODIA HANDWRITTEN ALPHABET RECOGNITION SYSTEM USING

# CONVOLUTIONAL NEURAL NETWORK

A major project Submitted in partial fulfillment of the Requirements for the Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

by

Ashpo Ghosh

Regd.no.1602031098

Md Sajid Sahanawaz

Regd. No. 1602081028

Under The Guidance

of

**Dr. Santosh kumar Majhi**

**Asst. professor**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGG.**

VEER SURENDRA SAI UNIVERSITY OF TECHNOLOGY,

BURLA

## VEER SURENDRA SAI UNIVERSITY OF TECHNOLGY

## BURLA, ODISHA



**Declaration**

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included; I have adequately cited and referenced the original sources. I also declare that i have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the university and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

DATE: 20.07.2020 Ashpo Ghosh

Regd.no.1602031098

Md Sajid Sahanawaz

Regd.no.1602031028

**Department of Computer Science & Engineering,**

**Veer Surendra Sai University of Technology, Burla, Odisha**



**Certificate**

This is to certify that the dissertation entitled "**DESIGN OF ODIA HANDWRITTEN ALPHABET RECOGNITION SYSTEM USING CONVOLUTIONAL NEURAL NETWORK**" submitted by Ashpo Ghosh & MD Sajid Sahanawaz is approved for the degree of bachelor of technology in **Computer Science and Engineering** , is a record of an original research work carried out by his under my supervision and guidance.

Dr. Manas Ranjan Kabat Dr. Santosh Kumar Majhi Head of Department Supervisor

**Acknowledgement**

I would like to express my sincere gratitude to my supervisor, Dr. Santosh Kumar Majhi, for his invaluable help during the course work towards this dissertation. He was a source of constant ideas and encouragement and provided a friendly atmosphere to work in. I am really very thankful to his for everything.

I am also thankful to Dr. Manas Ranjan Kabat, Head of the Department and to all the faculties of Department of Computer Science and Engineering for having supported me to carry out this dissertation and for their constant advice. I would like to thank all my friends for their encouragement and understanding. I would like to express my heart-felt gratitude to

them.

Ashpo Ghosh

Regd.no.1602031098

Md Sajid Sahanawaz

Regd.no.1602031028

# Approval Sheet

This dissertation entitled "**DESIGN OF ODIA HANDWRITTEN ALPHABET RECOGNITION SYSTEM USING CONVOLUTIONAL NEURAL NETWORK**" by **Ashpo Ghosh & MD Sajid Sahanawaz** is approved for the degree of bachelor of technology in "**Computer Science and Engineering**", department of computer science and engineering.

## Date: 20.07.2020 Examiner

## Place: Burla Dr. Santosh Ku. Majhi Supervisor

|  |  |  |
| --- | --- | --- |
|  | **Contents** |  |
| 1. Abstract |  | 8 |
| 2. Introduction |  | 9 |
| 3. Literature Review |  | 11 |
| 4. Overview of CNN |  | 13 |
| 5. Data Description |  | 20 |
| 6. Proposed Methodology |  | 22 |
| 6.1 Back End Phase |  | 22 |
| 6.2 Front End Phase |  | 25 |
| 5. Result and Discussion |  | 27 |
| 6. Conclusion |  | 35 |
|  |  |  |

## Chapter 1

**Abstract**

Odia is one of the famous languages spoken in India. It is the official language of Odisha and its speaker make 3.10% of total population of India. Therefore, a system needs to be developed for the recognition of Odia characters that would help many illiterate and physically handicapped people. Each character in Odia comprises of more than one connected symbols.

Character recognition is one of the difficult problems of Machine learning. The recognition rate for the offline recognition system is highly affected by various writing style, size and shape. In this paper work, we have developed a two phase system for the recognition of Odia alphabets. In the first phase, backend for the mentioned system is developed. This includes the development of CNN model for the classification of Odia alphabets in python environment and training the model for deployment. In the second phase, we have developed a front end for taking user input. This includes deployment of trained model and designing a website. The front end is built using Flask which is a web framework of python. Our model exploits the inherit characteristics of Odia character images. We have also developed a VSSUT Odia handwritten character dataset which comprises of 3576 images belonging to 37 classes. The proposed convolution neural network has given us an accuracy of 89%.

## Chapter 2

**Introduction**

Handwritten character recognition has gained interest of research in the field of pattern recognition. An immense effort has been spent on developing automatic document analysis and processing systems for the correct interpretation of characters as it has wide commercial applications. Character recognition of various scripts written in English, French, Dutch and many more languages have been carried out by different researchers. Character recognition process of handwritten characters is harder than the recognition of printed characters. In India, OCR is used for the character recognition and is the active area of research. It uses the conventional method of document image analysis where inputs are converted into digital text format. Odia is an official language of state Odisha. Odia is the sixth language of India to get classical language status. In this paper, we have attempted to produce a Machine Learning model that uses Convolutional Neural Networks for the classification of Odia characters. While developing the model we faced lot of difficulties due to the presence of similar orientation in shape and size. We also faced difficulties due to various writing skills of different users. Along with these difficulties there are many more which has made the evolution of an accurate handwritten character recognition system is still considered as on open research problem. The reason for using CNN is that in the field of Image processing, Neural Networks is one of the essential methods for classifying images. It solves many issues of handwritten character recognition to achieve a good recognition rate. A high recognition rate can be achieved by having a good feature extraction technique along with an eminent classifier. In recent times, CNN based models have got more attention because these models have achieved tremendous success in several fields like in image Net (Krizhev sky et al. 2012), Object Detection (He et al. 2015), Facial expression Recognition (Zhao et al. 2015), etc. Many researchers [1-3] from IIT Bhubaneswar, NIT Rourkela, Ravenshaw University, Utkal University and IIIT Bhubaneswar took interest and did pioneering work in the field of Odia Character and numerals recognition. They have used dataset from ISI, Kolkata and NIT, RKL. Sufian et al. [4] in the year 2020 produced a CNN based model named BDNet for the recognition of Bengal characters. They have also proposed a new dataset of 1000 images of Bengali handwritten numerals. These works show that automation of interpreting Odia scripts (Characters and Numerals) has enormous significance in social and commercial fields.

The rest of the paper is divided into six sections. Section 2 describes the previous works done on Odia handwritten alphabet recognition. The motivation for proposing a model and various developed techniques related to classification of Odia characters is discussed in this section. The concepts related to convolutinal neural network and its internals are discussed in Section 3. The dataset developed for our proposed model is described the Section 4. Section 5 explains the process of developing the Odia handwritten alphabet recognition. This includes the development of dataset, classification model and user interface for taking input. The result obtained of classification and hyper parameter tuning along with recognition rate is reported in Section 6. Section 7 provides the ending of the report with a conclusion.

## Chapter 3

**Literature Review**

The recognition of Odia characters is a difficult task as it contains compound and similar characters. These compound characters are made up of amalgamation of basic Odia characters leading to confusion in recognition. The intrinsic cursive nature of Odia characters make them look alike which forms the major challenge for recognition. This necessitates the study of research contributions relating to Odia handwritten alphabet recognition.

Sethy and Patra [5] developed offline Odia handwritten numeral recognizer using neural networks in the year 2016. There work was based on Binarization and Discrete Cosine Transform (DCT) scheme. There extensive simulation results in recognition rate of 80.2% for Binarization and 90% for DCT. In the same year, Rushiraj et al. [6] presented character recognition of handwritten Odia scripts. They used Euclidean Distance method that used 48 geometric features to classify 36 Odia characters.  The overall recognition rate is 87.6%. Sethy et al. [7] proposed automation of handwritten character recognition using a set of symmetry axes in the year 2017. They used standard database of NIT RKL Odia handwritten character and ISI Kolkata Odia numeral database. They used J48 classifier that gave 96.2% accuracy upon Odia numeral database and 95.6% accuracy upon Odia character database.

Pal et al. [8] used curvature feature of Odia characters to develop a classifier in the year 2007. This classifier recognized the Odia characters at a rate of 94.6%. Bhowmik et al. [9] proposed a novel hidden Markov model (HMM) for classification of handwritten Odia numerals in the year 2006. They have constructed one HMM for each numeral for training the dataset. There proposed model figured class restrictive likelihood for each HMM to recognize an unknown numeral image. They tested there classification scheme on an enormous transcribed Odia numeral database which came up with a recognition rate of 95.89% and 90.50% for training and test sets respectively. Mishra et al. [10] proposed two recognizers for Odia handwriting in the year 2013. There recognizer was based on Discrete Cosine Transformation (DCT) and Discrete Wavelet Transformation (DWT). There proposed paper also compared pros and cons of the two transformation schemes.  The two recognizers exploit the inherent characteristics of the Odia numeral images to produce a vector that is given input to a Back Propagation Neural Network (BPNN). The general precision rate got in there experimentation on manually written numerals utilizing DCT and DWT features are seen as 92% and 87.50% respectively. Pujari and Majhi [11] proposed an ensemble model for the recognition of Odia handwritten character in the year 2015. This ensemble model was built using four base classifier: Artificial Neural Network (ANN), Support Vector Machine (SVM), C5.0 Decision Tree and Discriminant Analysis (DA). They performed simulation using a combination of gradient and curvature based features extracted from the numeral dataset. They also performed feature reduction using Principal Component Analysis before simulation. The classification results obtained shows that the ensemble of SVM + DA and SVM + C5.0 + DA performed best with 97.5% accuracy.  Same year Ray et al. [12] proposed a Deep Bidirectional Long Short Term Memory (LSTM) based Recurrent Neural Network architecture for text recognition. The results of Deep Neural Networks for isolated numeral recognition and improved speech recognition using Deep BLSTM based approaches motivated them to develop this architecture which uses Connectionist Temporal Classification for training to recognize the labels of united words of printed Odia text.

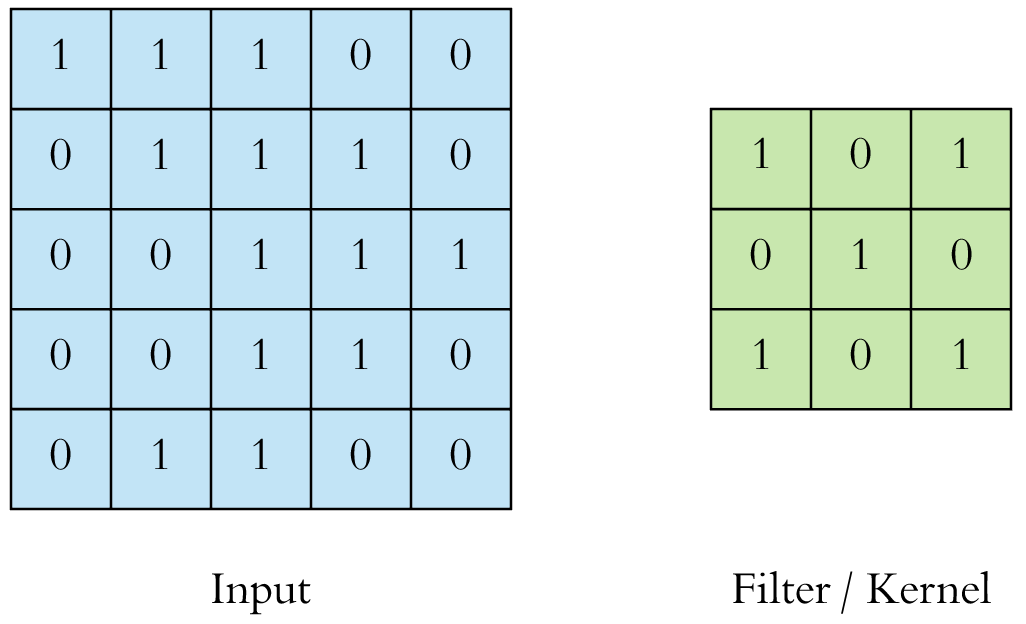
## Chapter 4

**Overview of CNN**

CNN is a specialized kind of neural network for processing data that has a known, grid-like topology, such as time- series (1D grid), image data (2D grid), etc. It is a supervised deep learning algorithm; it is used in various fields like speech recognition, image retrieval and face recognition.

**1. Convolution Layer**

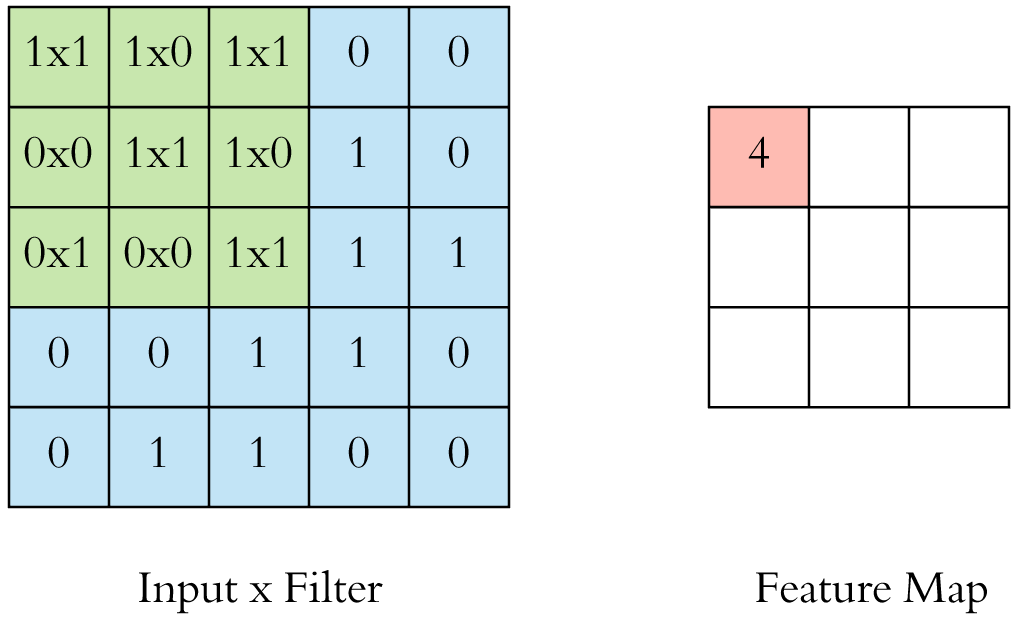
The main building block of CNN is the Convolutional layer. The primary focus of this layer is to extract high level spatial features. The convolution is applied on the input data using a convolutionfilter or kernel to produce a featuremap.



Input (5 x 5) Data and (3 x 3) Kernel

Fig - 1

The convolution operation is performed by sliding the filter over the input image. At every cell of the matrix, matrix multiplication is performed and the result goes into the feature map and this is continued to aggregate the convolution results in the feature map.

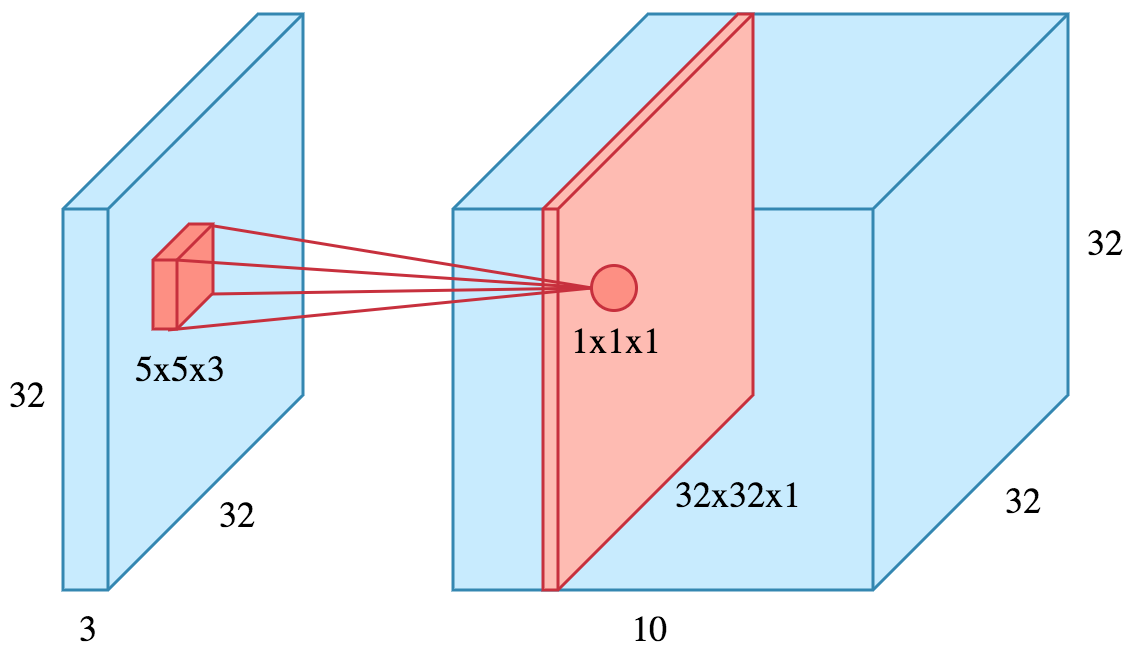


Conversion of Input x filter into Feature Map

Fig 2

An image is represented by a 3D matrix with height, width and depth, where depth corresponds to color channels (RGB).

More often, multiple convolutions are applied on an input, each having different filter and which results a distinct feature map. Finally, all these feature maps are stacked together which becomes the final output of the convolution layer.



Stacking of Feature Maps

Fig 3

**2. Non- Linearity or Activation Layer**

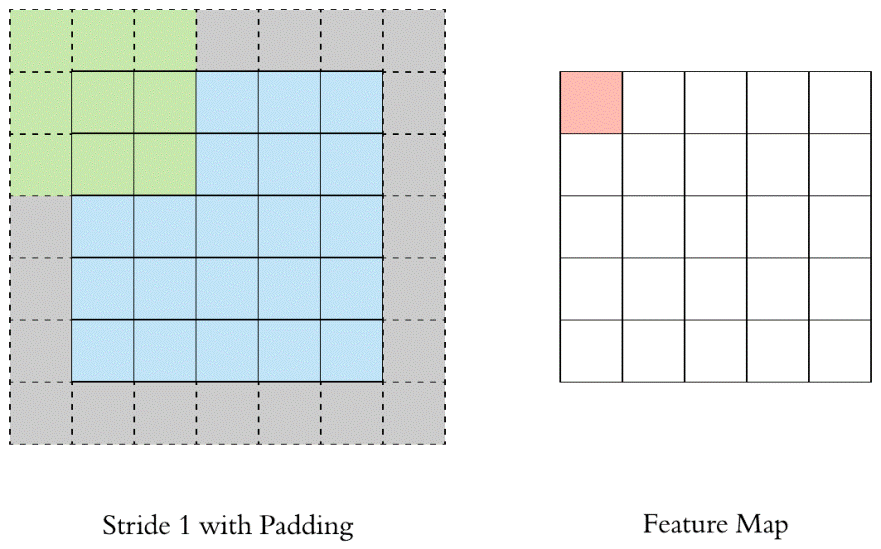
 The result of the convolution operation is passed through an activation function. The Activation functions are used so that the values in the final feature maps are not actually the sums, but some non linearity is function applied to them. One such activation functions are Relu (Rectified Linear Unit). It introduces non-linearity to our ConvNet. Other non- linear functions such as tanh or sigmoid can also be used instead of ReLU, but ReLU has been found to perform better in most situations

**3. Stride and Padding**

Stride specifies how much we move the convolution filter at each step and by default the value is 1. The stride can be manipulated as per the need; bigger strides will result less overlap between the receptive fields. This also makes the resulting feature map smaller since some of the potential locations are skipped.

In order to maintain same dimensionality, padding is used which surrounds the input with zeroes.

Padding is commonly used in CNN to preserve the size of the feature maps; otherwise they would shrink at each layer, which is not desirable.



Stride 1 with Padding and the Feature Map

Fig 4

**4. Pooling**

After a convolution operation, the pooling layer is applied so that there is reduction in dimensionality. Itreduces the number of parameters, which both shortens the training time and combats overfitting. Pooling layers downsample each feature map independently, reducing the height and width, keeping the depth intact.In all cases, pooling helps to make the representation become approximately invariant to small translations of the input.The local translation can be a very useful property in order to find some specific features.

There are basically three types of Pooling:-

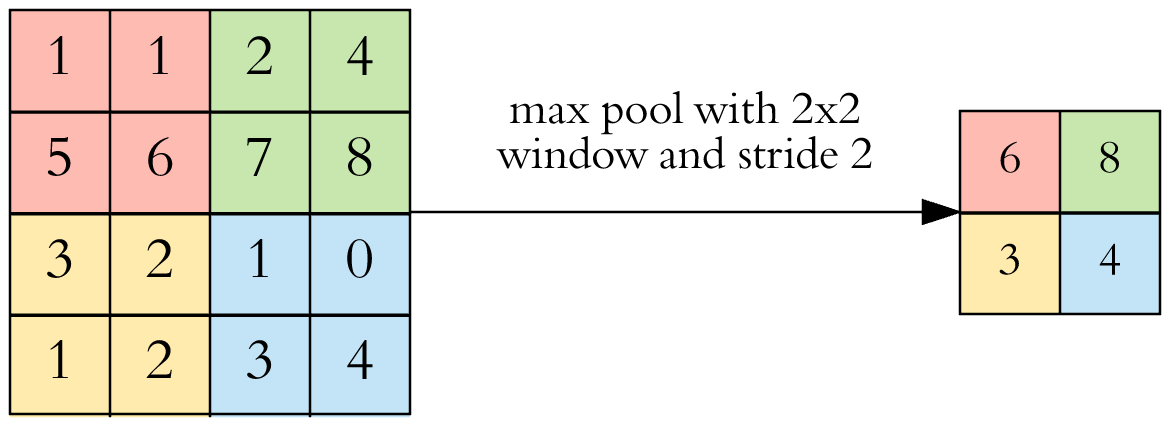
• Max Pooling (works better)

• Average Pooling

• Sum Pooling

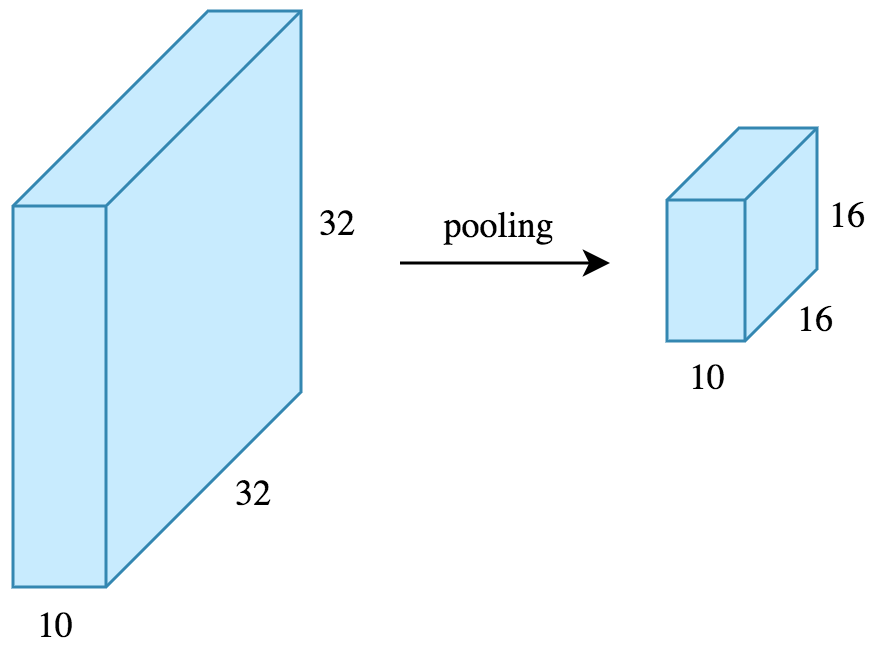
The most common type of pooling is maxpooling which just takes the max value in the pooling window. Contrary to the convolution operation, pooling has no parameters. It slides a window over its input, and simply takes the max value in the window. Similar to a convolution, we specify the window size and stride.

In case of Average Pooling, the average value for each patch on the feature map is calculated. Thus, while max pooling gives the most prominent feature in a particular patch of the feature map, average pooling gives the average of features present in a patch. In case of Sum Pooling, the sum for each patch on the feature map is calculated.



Feature Map with a 2 x 2 Max pool

Fig 5



Pooling Operation on Feature Map

Fig 6

**5. Fully Connected Layer**

After the convolution + pooling layers, couple of fully connected layers to wrap up the CNN architecture is applied. In Fully Connected (FC) layer, every neuron of previous layer is connected to every neuron of the next layer in the model like normal MLP. This layer normally uses Softmax activation function in the output layer. The output of both convolution and pooling layers are 3D volumes, but a fully connected layer expects a 1D vector of numbers. So, the output of the final pooling layer is flattened to a vector and that becomes the input to the fully connected layer. Flattening is simply arranging the 3D volume of numbers into a 1D vector, Each value of the output vector represents the class probability.

**6. Loss Function**

Loss function is an important statistical parameter to measure the performance of the model. Loss is defined as the difference between predicted values and expected values. It can be used to estimate the loss of the model so that the weights can be updated to reduce the loss on the next evaluation.

Several loss functions are used to measure the performance of the network:

* Regression Loss Functions
  + Mean Squared Error Loss
  + Mean Squared Logarithmic Error Loss
  + Mean Absolute Error Loss
* Binary Classification Loss Functions
  + Binary Cross-Entropy
  + Hinge Loss
  + Squared Hinge Loss
* Multi-Class Classification Loss Functions
  + Multi-Class Cross-Entropy Loss
  + Sparse Multiclass Cross-Entropy Loss
  + Kullback Leibler Divergence Los

**7. Regularization**

Weight regularization provides an approach to reduce the overfitting of a deep learning neural network model on the training data and improve the performance of the model on new data.

There are multiple types of weight regularization, such as L1 and L2 vector norms, and each requires a hyperparameter that must be configured. Other types of regularization like Drop out and Data Augmentation can be used.

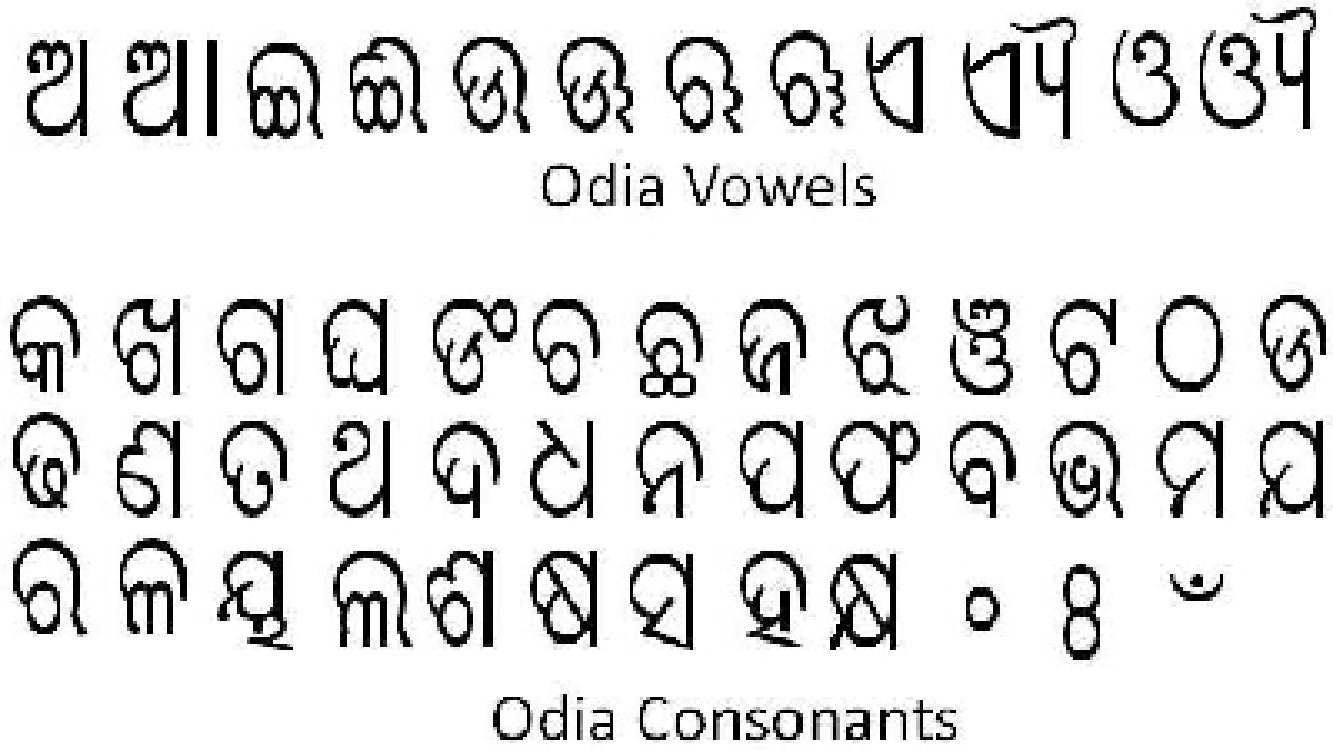
**8. Hyper Parameters**

* **Filter size**: Typically, 3x3 filters are used, but 5x5 or 7x7 are also used depending on the application. There are also 1x1 filters which we will explore in another article, at first sight it might look strange but they have interesting applications. Remember that these filters are 3D and have a depth dimension as well, but since the depth of a filter at a given layer is equal to the depth of its input, we omit that.
* **Filter count**: this is the most variable parameter; it’s a power of two anywhere between 32 and 1024. Using more filters results in a more powerful model, but we risk overfitting due to increased parameter count. Usually we start with a small number of filters at the initial layers, and progressively increase the count as we go deeper into the network.
* **Stride**: we keep it at the default value 1 so that maximum pixel is used for computation.
* **Padding**: It refers to the amount of pixels added to an image when it is being processed by the kernel of a CNN.

## Chapter 5

**Data Description**

Odia is an Indo-Aryan language spoken by 33 million peoples around the globe. Odia language comprises of 12 vowels and 48 consonants. This language doesn't comprise the idea of upper and lower case letters.  Here we have developed our own dataset for processing our model. In the process of creating dataset, first we have collected the handwriting copies of student from various schools, colleges and universities across the Odisha. This helps to learn the different writing styles of Odia alphabets across the state. These copies were scanned and converted into images. These scanned copies constituted many alphabets on the single page. Adobe Photoshop was used to crop the images of each alphabet. Each of the cropped images was saved in the .jpg format.  Slice operator of Adobe Photoshop was used for cropping. Our dataset comprises of 37 classes. The table 1 describes the dataset distribution of each class.



Odia Vowels and Consonants

Fig 7

There are in total 3576 images out of which 2523 images belong to training class and 1053 images belong to test class.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CLASS** | **TRAIN** | **TEST** | **CLASS** | **TRAIN** | **TEST** |
| ଅ | 126 | 54 | ନ | 52 | 22 |
| ଆ | 133 | 56 | ପ | 52 | 22 |
| ଇ | 62 | 26 | ଫ | 52 | 22 |
| ଔ | 100 | 44 | ବ | 52 | 22 |
| କ | 225 | 71 | ଭ | 52 | 22 |
| ଖ | 198 | 70 | ମ | 52 | 22 |
| ଚ | 52 | 22 | ଜ | 52 | 22 |
| ଛ | 52 | 22 | ୟ | 52 | 22 |
| ଜ | 119 | 51 | ର | 52 | 22 |
| ଝ | 52 | 22 | ଲ | 52 | 22 |
| ଟ | 52 | 22 | ଳ | 52 | 22 |
| ଡ | 52 | 22 | ଶ | 52 | 22 |
| ଢ | 52 | 22 | ଷ | 52 | 22 |
| ଣ | 52 | 22 | ସ | 52 | 22 |
| ତ | 52 | 22 | ହ | 52 | 22 |
| ଥ | 52 | 22 | ଠ | 52 | 22 |
| ଦ | 52 | 22 |  |  |  |
| ଧ | 52 | 22 |  |  |  |
|  |  |  |  |  |  |

Train- Test Split of dataset

Table 1

## Chapter 6

## Proposed Methodology

The whole Recognition System has been divided into 2 phases:

1. The Back End Phase
2. The Front End Phase
3. **The Back End Phase**

This is the first phase of our recognition system which incorporates preprocessing of the whole image dataset and then employs the development of the Model.

* 1. **Preprocessing**

Data preparation is required when working with neural network and deep learning models. Increasingly data augmentation is also required on more complex object recognition tasks.

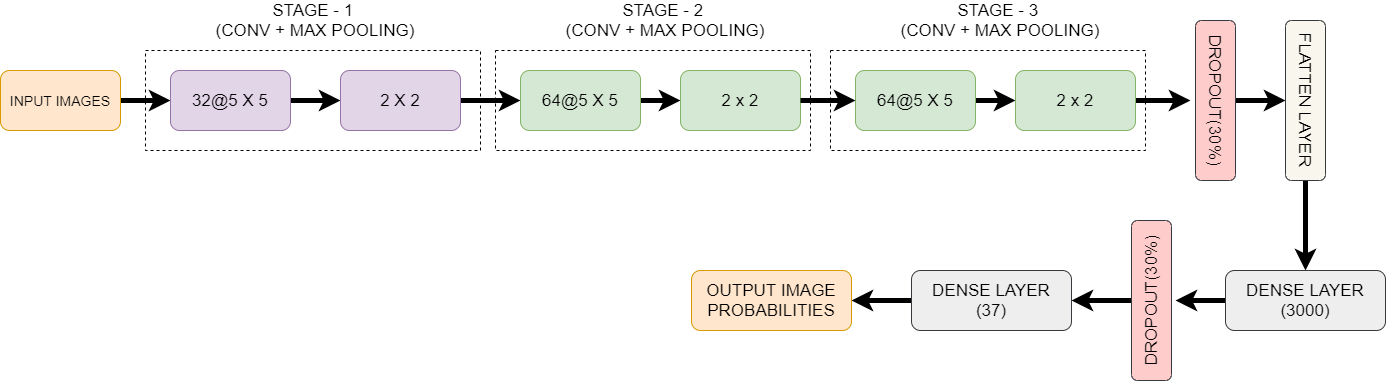
Dataaugmentation is a strategy used to increasetheamountofdata by using techniques like cropping, padding, flipping, etc. Data augmentation makes the model more robust to slight variations, and hence prevents the model from overfitting. It is neither practical nor efficient to store the augmented data in memory, and hence the ImageDataGenerator class from Keras can be used.  ImageDataGenerator generates batchesof tensor image data with real-time data augmentation. The output images generated by the generator will have the same output dimensions as the input images

Although, there are various parameters in the ImageDataGenerator but we have restricted to only some of the few important parameters. For Training data, we have set the rescale parameter to 1./255 , shear range and zoom range to 0.2. The horizontal flip property is set to true which randomly flipping half of the images horizontally relevant when there are no assumptions of horizontal asymmetry.

After the preprocessing phase, all the images are fixed to particular standardized dimensions with proper tuning of attributes of ImageDataGenerator. Next, we have split the whole dataset into train\_set and test\_set with a 70 and 30 split respectively. The train\_set contains 2456 images whereas test\_set contains 1002 images. The batch size for both of them is fixed to 16 and also the shuffle parameter is set to true. Shuffling data serves the purpose of reducing variance and making sure that models remain general and overfit less. We have used the shuffle parameter because the data used is sorted by their class/target. Shuffling will make sure that the training/test sets are representative of the overall distribution of the data.

* 1. **Model Development**

We have used the deep learning model based on CNN for the purpose of Odia handwritten recognition. A layer-wise detailed architecture of proposed CNN model for current work is shown in Fig 10. In this convolutional layer, we have taken 32 learnable convolutional filters each with a kernel size of 5 × 5 pixels and also stride size of 1×1 so that maximum pixel areas are consumed and this can lead to better efficiency. This window spatially slides over all the images of size 64 × 64 pixels and computes the respective dot product. Each filter produces a 3-dimensional activation map and these activation maps are stacked along the depth dimension to produce the output volume. We have used ‘zero padding’ so that the size of the output images of convolution layers remains same. The resultant output is then passed through ReLU activation function in order to avoid the negative values.



Architecture of the Model

Fig 10

To reduce computational cost and to extract higher level features, we have then added pooling layers. We have used max pooling operation with a window of size 2 × 2 that slides over all the images of 64 × 64 pixels and the highest pixel value of every sliding window represents the window. In the second convolutional layer, we have taken 64 learnable filters each with a kernel size of 5 × 5 pixels, followed by ReLU activation and a max pooling layer of window size 2 × 2. Similarly, In the third convolutional layer, we have taken 64 learnable filters each with a kernel size of 5 × 5 pixels, followed by ReLU activation and a max pooling layer of window size 2 × 2. The next layer is the fattened layer where we have converted the resultant 3-dimensional output into a 1-dimensional vector in order to fit it into a FC layer. In the next phase, we have added two FC or dense layers, one with 1000 neurons and other with 37, each having full connection to all the activations in the previous layers. In CNN model, in order to overcome overfitting problem a dropout of 30% after the third pooling layer and 30% after the first fully connected layer (dense) are applied. In general, forming an ensemble of neural network model with different architectures on the same dataset and averaging the predicted output from all the architectures finally reduce the overfitting problem to some extent. We have used different combinations of architecture in order to conclude for a better and efficient model so that maximum accuracy is obtained. One of the regularization technique used is Dropout. It randomly selects neurons and drops it during training so that different architecture for same CNN model is generated. It is observed experimentally, that dropout increases the accuracy by a suitable amount as all the neurons do not actively contribute to the final result and unnecessary neurons cause variations in the output. So, dropping dormant neurons improves the accuracy of the model. Numbers of neurons are less in feature maps generated after max pooling layers; thus, dropout probability should be less in order to increase the efficiency of model. In this regard, it is experimentally observed that, applying dropout of 30% after max pooling layer is the best choice. Finally, the dense layer containing 37 neurons for the 37 classes is added followed by Softmax function, which returns a list of probabilities corresponding to each of the 37 classes. The class with the highest probability value is taken as the final output.

1. **The Front End Phase**

In the second phase, we have developed a website for user input. Our backend is based on python, so our front end is also developed using a web application framework of python called Flask. The development of front end needs presentation logic and business logic. The presentation logic of the recognition system is written using HTML, CSS and Bootstrap. The business logic of the recognition system is written using Flask framework that includes code for the deployment of trained machine learning model. Flask helps to build a light weight web application using its pre defined libraries. The user input is taken in form of image and the most three likely outcomes is displayed on the web page in tabular format. The Screenshots of the developed web application are depicted in the following Fig 8 & Fig 9.

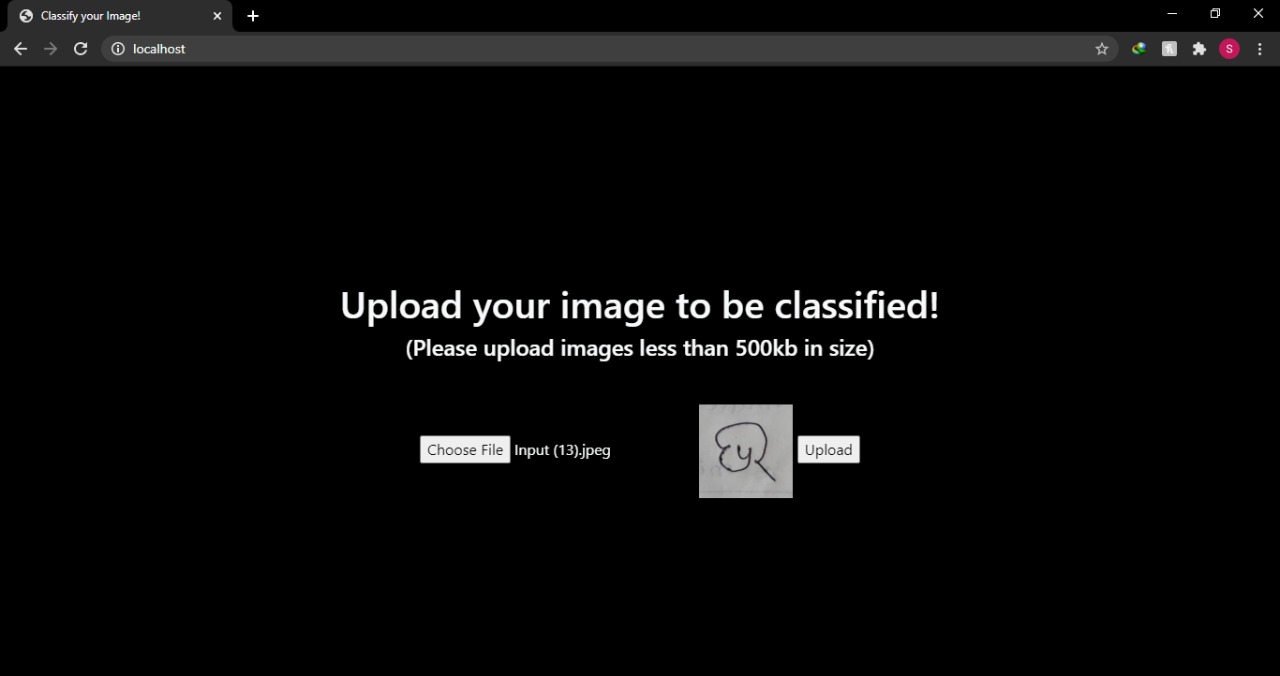
****

Fig 8

Homepage of the Application

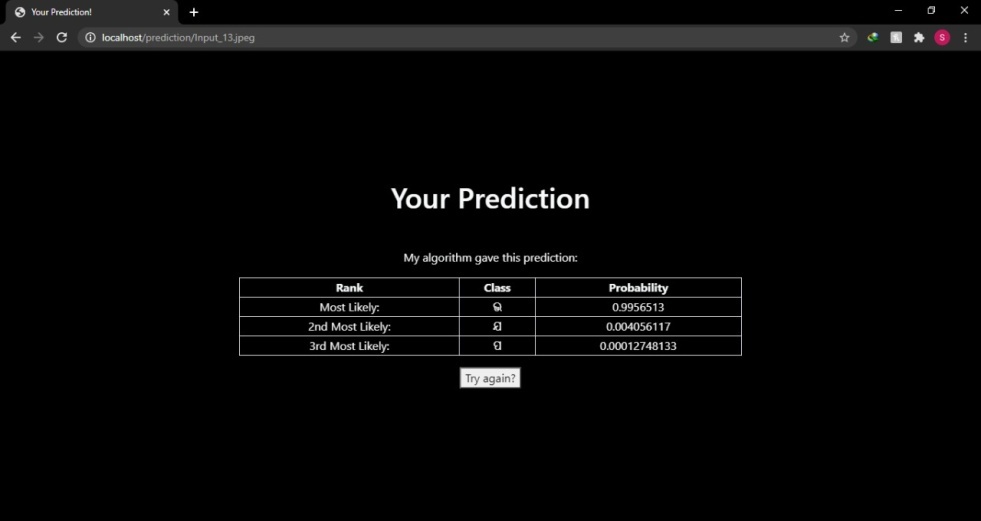
****

Fig 9

Result page of the Application

## Chapter 6

**Results and Discussion**

The experiments are performed on a laptop powered by Intel quad-core i5- CPU @ 3.20 GHz, 8 GB RAM and AMD Raedon as a graphics card for GPU support. As far as the software requirement is concerned, Python framework’ and ‘Keras’ libraries are used. Program is written in Jupyter notebook interface.

Our Proposed CNN Model has secured an accuracy of 88.75% over the whole dataset.

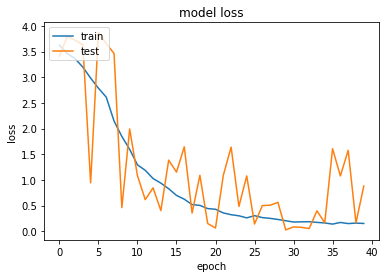
The plot in Fig 11 denotes the relationship between number of epochs and the loss for both train and test.

* When the validation loss starts to increase from the training loss, then the training should be stopped. When the gap between the losses starts widening, then it indicates **overfitting** in the model. This can be prevented by decreasing the network size or increasing the dropout.

* If your training/validation losses are about equal then the model is **underfitting**. It can be prevented by increasing the size of the model. It means increasing the number of neurons

in the layer.

Hence, the plot shows that as the number of epochs increases the gap between training and test loss starts decreasing. Hence, it is fixed to 40. The losses for training and validation during the last epochs are 0.15 and 0.16 which is acceptable

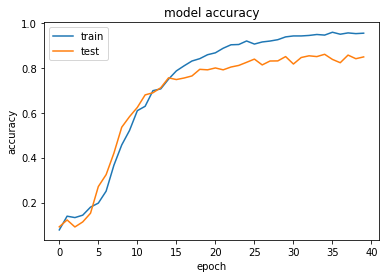


Plot showing loss vs no. of epochs

Fig 11

.

The plot in Fig 12 denotes the relationship between number of epochs and the accuracy. As the number of epochs starts increasing the accuracy also increase. Training of the model should be stopped when the gap between train and test accuracy starts widening. Hence, the number of epochs is fixed to 40 as after these the lines starts diverging which can lead to overfitting in the model. The training and test accuracy during the last few epochs are around 95% and 85% respectively.



Plot showing accuracy vs no. of epochs

Fig 12

To get the best possible performance for any architecture, tuning of hyper-parameters is an essential task. In order to do that, Grid search method has been performed for choosing the best optimizer. Although, there are various hyperparameters for tuning, we have tuned few essential parameters like no of epochs, batch size, number of neurons in the fully connected layer, optimizing algorithm and drop out. The rest tuning parameters are empirically chosen which normally performs best in that range. This also helps us to determine whether there is overfitting in the model or not.

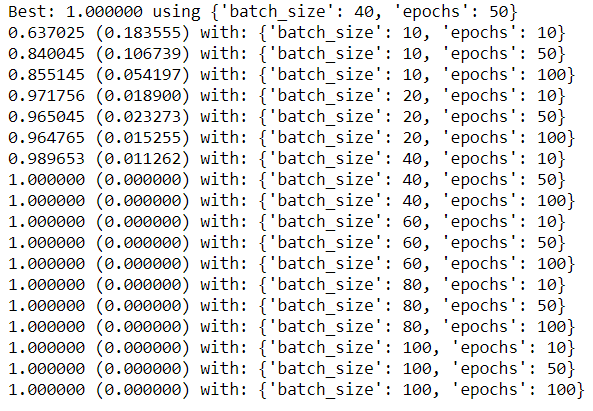
The following are the hyper parameter tuning performed in the project:

1. **Batch Size and No of epochs**

The batch size is the number of patterns shown to the network before the weights are updated. It is done to optimize the training of neural network by, defining how many patterns to read at a time.

The number of epochs is the number of times that the entire training dataset is shown to the network during training.

Fig 13 gives the mean and standard deviation for different epochs and batch size. The best result is observed with an accuracy of near 100% for a batch size of 40 and the number of epochs as 50.

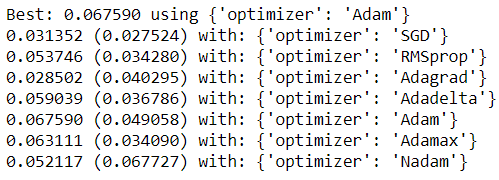


Mean and Standard deviation for different epochs and batch size

Fig 13

1. **Optimization Algorithm**

The optimal setting of an algorithm is very crucial for getting the best performance. As, the Adam optimizer is one of the most popular optimizer and even after hypertuning over the dataset, the Adam Optimizer comes with an accuracy of 6% among others. The comparison of various optimizers with their mean and standard deviation is depicted in Fig 14.



Comparison of various Optimizers with their mean and standard deviation

Fig 14

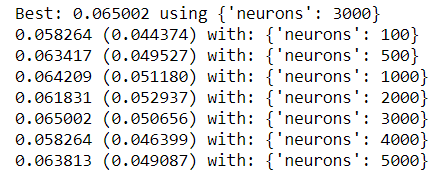
1. **Number of Neurons**

The number of neurons in a layer is an important parameter for tuning any neural network. Generally the number of neurons in a layer controls the representational capacity of the network,

In this example, tuning is done in the second last dense layer or (fully connected layer) for the number of neurons.

When the network size increases, along with the batch size and number of epochs, number of neurons also plays a crucial role.

Fig-15 gives the mean and standard deviation for different number of neurons. The best result is observed with an accuracy of near 6.5% for 3000 neurons. Moreover, the accuracy for 1000 neurons is 6.4%. So, it is not worth for making the architecture more complex as there is no significant gain in accuracy and hence can be set to 1000.



Mean and Standard deviation for different number of neurons

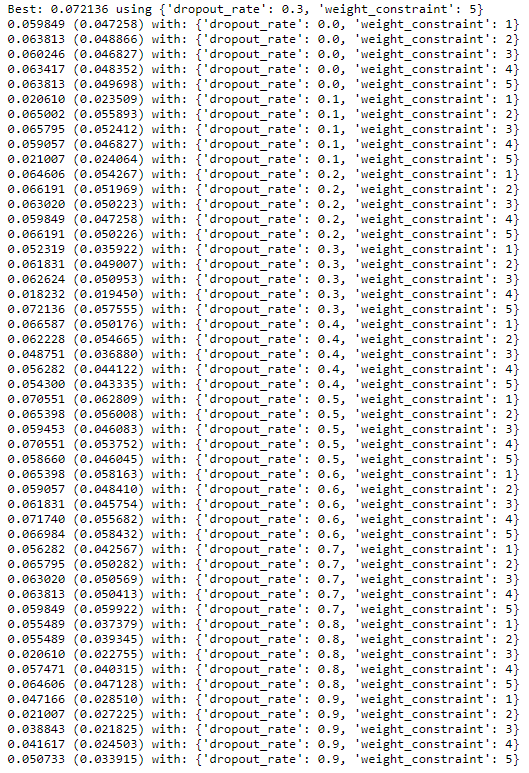
Fig 15

1. **Dropout Reularization**

In order to limit overfitting and improve the model’s ability to generalize, dropout regularization is performed on the model. The percentage for dropout is maintained between 0.0 and 0.9. The

dropout is added after the third stage of the conv+max pool layer and after the first dense layer.

Fig 16 gives the mean and standard deviation for different number of neurons. The best result is observed with a dropout of 0.3 with an accuracy of near 7.2% .



Mean and Standard deviation for different number of neurons and its Dropout rate

## Fig 16

## Accuracy Comparison

## In order to validate the proposed machine learning model, we have analyzed many research papers based on this topic. Many researchers have done lots of research on handwritten character recognition but very less research is done in the field of Odia character recognition.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Method | Features | Classifier | Accuracy (%) | |
| ISI kolkata | IITBBS |
| Bhowmilk et al.[25] | Scalar | HMM | 90.50 | 88.74 |
| Dash et al. [26] | Gestalt | Complexity metric | 92.82 | 91.40 |
| Roy et al. [27] | Directional | QDF | 94.81 | 94.12 |
| Pal et al. [28] | Directional | MQC | 98.40 | 96.30 |
| Dash et al. [29] | Hybrid | DLQDF | 98.50 | 98.28 |
| Dash et al. [30] | AZNRST | K-NN | 99.10 | 98.60 |
| Dash et al. [31] | BESAC | Random forest | 98.44 | 97.30 |
|  |  | SVM | 99.02 | 98.56 |
|  |  | Nearest neighbor | 99.35 | 98.90 |

## Comparison of various Classifiers with their Accuracy

## Table 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Authors | Techniques for Feature Extraction | Classifier used | Recognition Rate (%) | |
| NIT, RKL(Odia Character) | ISI, Kolkata(Odia numeral) |
| Pal et al. in [5] | Curvature | Quadratic | 94.6 |  |
| Sethy et al. in [32] | DWT | BPNN | 94.8 |  |
| Mishar et al. in [10] | DCT+DWT | SVM |  | 92 & 87.5 |
| Sethy et al. in [33] | DCT+Binarization | ANN |  | 90 & 80.2 |
| Sethy et al. [7] | Symmetry axes | J48(Decision Tree) | 95.6 | 96.2 |

## Comparison of various Classifiers with their Accuracy

## Table 3

## Table 2 and Table 3 compare the accuracy of different research works done on Odia handwritten character recognition. The datasets used in these researches are ISI, Kolkata (Odia numerals and character), IIT, BBSR (Odia numerals and character) and NIT, RKL (Odia character). Our machine learning model produces accuracy of 88.75% over our dataset. The study of these research paper shows that recognition rate is highly dataset dependent. Each dataset has its own features that need to be extracted using various techniques for training the model. The researchers have proposed various feature extraction techniques that have been applied to the dataset and trained the model.

## 

## Chapter 7

**Conclusion**

Deep Convolutional neural networks are achieving high success in computer vision problems. So, in the proposed work, we have implemented a CNN model for the recognition of Odia handwritten alphabet with an accuracy of 88.75%. The proposed architecture pipeline consists of 3 stages conv-pool layer and 2 dense connected layers. The optimized framework is more promising than using conventional Artificial Neural Network. The model also generalizes well with not much overfiiting and hence the detection performance can be improved by fine tuning the network with some more number of high resolution images with clear background. The problem arises due to the constraint on the limited dataset. The proposed model is trained on a substantially large dataset which has been developed from scratch. Along with empirical estimation of some hyper parameters, the other hyper parameters are also appropriately tuned. At last, a web application has been developed using flask framework so that it can be tested on real data. The result of the application shows the probability of similarity with the closest character along with 2 other similar characters.

## 

**References**

1. Jena, Om Prakash, et al. "Odia Characters and Numerals Recognition using Hopfield Neural Network based on Zoning Features." *International Journal of Recent Technology and Engineering* 8.2 (2019): 4928-4937.
2. Dash, Kalyan S., Niladri B. Puhan, and Ganapati Panda. "BESAC: Binary External Symmetry Axis Constellation for unconstrained handwritten character recognition." *Pattern Recognition Letters* 83 (2016): 413-422.
3. Dash, Kalyan S., Niladri B. Puhan, and Ganapati Panda. "Sparse Concept Coded Tetrolet Transform for Unconstrained Odia Character Recognition." *arXiv preprint arXiv:2004.01551* (2020).
4. Sufian, Abu, et al. "Bdnet: bengali handwritten numeral digit recognition based on densely connected convolutional neural networks." *Journal of King Saud University-Computer and Information Sciences* (2020).
5. U. Pal, T. Wakabayashi and F. Kimura, "A System for Off-Line Oriya Handwritten Character Recognition Using Curvature Feature," 10th International Conference on Information Technology (ICIT 2007), Orissa, 2007, pp. 227-229, doi: 10.1109/ICIT.2007.63.
6. I. Rushiraj, S. Kundu and B. Ray, "Handwritten character recognition of Odia script," 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), Paralakhemundi, 2016, pp. 764-767, doi: 10.1109/SCOPES.2016.7955542.
7. A. Sethy, P. K. Patra, S. Nayak and P. M. Jena, "Symmetric Axis Based Off-Line Odia Handwritten Character and Numeral Recognition," 2017 3rd International Conference on Computational Intelligence and Networks (CINE), Odisha, 2017, pp. 83-87, doi: 10.1109/CINE.2017.27.
8. U. Pal, N. Sharma, T. Wakabayashi and F. Kimura, "Off-Line Handwritten Character Recognition of Devnagari Script," Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Parana, 2007, pp. 496-500, doi: 10.1109/ICDAR.2007.4378759.
9. T. K. Bhowmik, S. K. Parui, U. Bhattacharya and B. Shaw, "An HMM Based Recognition Scheme for Handwritten Oriya Numerals," 9th International Conference on Information Technology (ICIT'06), Bhubaneswar, 2006, pp. 105-110, doi: 10.1109/ICIT.2006.29.
10. T. K. Mishra, B. Majhi and S. Panda, "A comparative analysis of image transformations for handwritten Odia numeral recognition," 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Mysore, 2013, pp. 790-793, doi: 10.1109/ICACCI.2013.6637276.
11. P. Pujari and B. Majhi, "Performance comparison of ensemble models for recognition of offline handwritten Odia numerals," 2015 IEEE Power, Communication and Information Technology Conference (PCITC), Bhubaneswar, 2015, pp. 29-35, doi: 10.1109/PCITC.2015.7438187.
12. A. Ray, S. Rajeswar and S. Chaudhury, "Text recognition using deep BLSTM networks," 2015 Eighth International Conference on Advances in Pattern Recognition (ICAPR), Kolkata, 2015, pp. 1-6, doi: 10.1109/ICAPR.2015.7050699.
13. T. Wakabayashi, U. Pal, F. Kimura and Y. Miyake, "F-ratio Based Weighted Feature Extraction for Similar Shape Character Recognition," 2009 10th International Conference on Document Analysis and Recognition, Barcelona, 2009, pp. 196-200, doi: 10.1109/ICDAR.2009.197.
14. S. Chanda, K. Franke and U. Pal, "Text Independent Writer Identification for Oriya Script," 2012 10th IAPR International Workshop on Document Analysis Systems, Gold Cost, QLD, 2012, pp. 369-373, doi: 10.1109/DAS.2012.86.
15. T. M. Bernard and A. Manzanera, "Improved low complexity fully parallel thinning algorithm," Proceedings 10th International Conference on Image Analysis and Processing, Venice, Italy, 1999, pp. 215-220, doi: 10.1109/ICIAP.1999.797597.
16. Mishra, T. K., Majhi, B., Sa, P. K., & Panda, S. (2014). Model based odia numeral recognition using fuzzy aggregated features. *Frontiers of Computer Science*, *8*(6), 916-922.
17. Xu, B., Ye, Y., & Nie, L. (2012, June). An improved random forest classifier for image classification. In *2012 IEEE International Conference on Information and Automation* (pp. 795-800). IEEE.
18. Swethalakshmi, H., Jayaraman, A., Chakravarthy, V. S., & Sekhar, C. C. (2006, October). Online handwritten character recognition of Devanagari and Telugu Characters using support vector machines.
19. Sarangi, Pradeepta K., P. Ahmed, and Kiran K. Ravulakollu. "Naïve bayes classifier with lu factorization for recognition of handwritten Odia numerals." *Indian Journal of Science and Technology* 7.1 (2014): 35-38.
20. Impedovo, Sebastiano, Francesco Maurizio Mangini, and Donato Barbuzzi. "A novel prototype generation technique for handwriting digit recognition." *Pattern Recognition* 47.3 (2014): 1002-1010.
21. Liu, Cheng-Lin, and Ching Y. Suen. "A new benchmark on the recognition of handwritten Bangla and Farsi numeral characters." *Pattern Recognition* 42.12 (2009): 3287-3295.
22. Pal, Umapada, Ramachandran Jayadevan, and Nabin Sharma. "Handwriting recognition in indian regional scripts: a survey of offline techniques." *ACM Transactions on Asian Language Information Processing (TALIP)* 11.1 (2012): 1-35.
23. Jindal, M. K., R. K. Sharma, and G. S. Lehal. "Segmentation of horizontally overlapping lines in printed Indian scripts." *International Journal of Computational Intelligence Research* 3.4 (2007): 277-286.
24. Hamamura, Tomoyuki, Takuma Akagi, and Bunpei Irie. "An analytic word recognition algorithm using a posteriori probability." *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Vol. 2. IEEE, 2007.
25. Mishra, T. K., Majhi, B., Sa, P. K., & Panda, S. (2014). Model based odia numeral recognition using fuzzy aggregated features. *Frontiers of Computer Science*, *8*(6), 916-922.
26. K.S. Dash , N.B. Puhan , G. Panda , Handwritten Odia numeral recognition: Com- bining gestalt configural superiority effect and inner distance based complexity metric, in: Proceedings of IEEE 14th International Symposium on Signal Pro- cessing and Information Technology, 2014, pp. 1–6 .
27. K. Roy , T. Pal , U. Pal , F. Kimura , Oriya handwritten numeral recognition system, in: Proceedings of IEEE 8th International Conference on Document Analysis and Recognition, 2005, pp. 770–774 .
28. U. Pal , N. Sharma , T. Wakabayashi , F. Kimura , Handwritten numeral recognition of six popular Indian scripts, in: Proceedings of IEEE 9th International Confer- ence on Document Analysis and Recognition, 2007, pp. 749–753 .
29. K.S. Dash , N.B. Puhan , G. Panda , A hybrid feature and discriminant classi- fier for high accuracy Odia handwritten numeral recognition, in: *Proceed- ings of IEEE Region 10 Technical Symposium* , *TENSYMP’14* , 2014, pp. 531–535 .
30. K.S. Dash , N.B. Puhan , G. Panda , Handwritten numeral recognition using non-redundant Stockwell transform and bio-inspired optimal zoning, IET Im- age Process. 9 (10) (2015) 874–882 .
31. Dash, Kalyan S., Niladri B. Puhan, and Ganapati Panda. "BESAC: Binary External Symmetry Axis Constellation for unconstrained handwritten character recognition." *Pattern Recognition Letters* 83 (2016): 413-422.
32. Sethy, Abhisek, Prashanta Kumar Patra, and Deepak Ranjan Nayak. "Off-line handwritten Odia character recognition using DWT and PCA." *Progress in Advanced Computing and Intelligent Engineering*. Springer, Singapore, 2018. 187-195.
33. Sethy, Abhisek, and Prashanta Kumar Patra. "Off-line Odia handwritten numeral recognition using neural network: a comparative analysis." *2016 International Conference on Computing, Communication and Automation (ICCCA)*. IEEE, 2016.