**Proceedings of 14th IOE Graduate Conference**
*Peer Reviewed*
**Year: 2023 Month: December Volume: 14**
**ISSN: 2350-8914 (Online), 2350-8906 (Print)**

# Reviving Indigenous Script: Optical Character Recognition for the Prachalit Newa Script

Sushovan Shakya [a], Shovit Nepal [b], Riwaj Neupane [c], Shiv Ranjan Gupta [d], Hasina Shakya [e]

[a, b, c, d, e] *Department of Electronics and Computer Engineering, Thapathali Campus, IOE, Tribuvan University, Nepal*

[a] tha075bei046@tcioe.edu.np, [b] tha075bei039@tcioe.edu.np, [c] tha075bei040@tcioe.edu.np, [d] tha075bei034@tcioe.edu.np

**Abstract**

Many manuscripts and inscriptions were written historically, spanning several centuries in Kathmandu valley, by the indigenous Newar people, who developed their own alphabetic writing system. To make a robust system for effective character recognition of Prachalit Newa script and thus assisting in research and preservation of the language, a dataset was created from scratch and various supervised deep learning models such as VGG-16 and AlexNet were devised, obtaining a significant accuracy of 92% in simple character recognition.

**Keywords**

Dataset, Deep Learning, Image Processing, Inscription, Optical Character Recognition

## 1. Introduction

### 1.1 Background

The Nepal Bhasa, also known as *Newā Bhay* or the *Newā* language, is an endangered language belonging to the Sino-Tibetan family of languages. Nepal Bhasa is a digraphic language, which means it uses more than one script to transcribe the language. Despite being a member of Sino-Tibetan family, the scripts used to write Nepal Bhasa share ancestry with the Indic scripts like Devanagari. Many scripts used to write Nepal Bhasa have become obsolete, while Ranjana and Prachalit scripts have maintained their popularity in modern times.

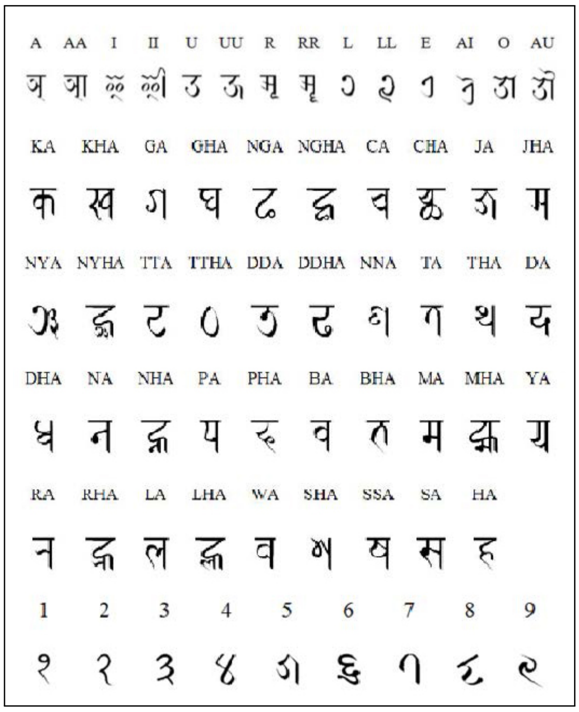
A AA I II U UU R RR L LL E AI O AU
KA KHA GA GHA NGA NGHA CA CHA JA JHA
NYA NYHA TTA TTHA DDA DDHA NNA TA THA DA
DHA NA NHA PA PHA BA BHA MA MHA YA
RA RHA LA LHA WA SHA SSA SA HA
1 2 3 4 5 6 7 8 9

**Figure 1:** Varnamala of Prachalit Lipi

Prachalit Lipi came to prominence in 6th century CE, and was widely used to write Nepal Bhasa till 20th century. The earliest recorded instance of Prachalit script dates to 10th century CE, in a manuscript titled Lankavatara Sutra , which is dated Nepal Samvat 28 (908 AD). With the Nepali language (Khas Bhasa) being the lingua franca of Nepal and eventually the de facto official language starting from the early 20th century, Devanagari script become more prevalent and thus Nepal Bhasa also adopted Devanagari for transcription. Prachalit (lit. "popular") script hitherto have been prevalent and widely used to transcribe the language in tremendous amount, many of which are well-preserved within the valley, and still has maintained its popularity in modern times within the Nepal Bhasa linguasphere. However, extraction of information from the manuscript is often a tedious task, as one is expected to be acquainted with both the script and the language, and reading the text line-by-line becomes quite inefficient in this regard.

### 1.2 Problem Statement

Understanding the script used in historical manuscripts is necessary to comprehend the information they contain. Similarly, it is necessary to learn the script for transcribing the language in modern times. The major issue here lies in storing the information from historical sources in a digital format. Even if photographs of the manuscripts or inscriptions are taken and saved as digital images, extracting the essential information can be time consuming as one needs to first learn the script, then read the text line-by-line and transliterate into a different script before typing them in a textual format for storage. This process is inefficient and consumes a lot of time. To improve this process, a dataset of characters specific to the writing system can be created, and deep learning algorithms can be applied to accurately recognize characters and extract information more efficiently.

### 1.3 Objectives

- To apply deep learning model to effectively classify Prachalit Newa Lipi characters
- To convert the recognized characters into Romanized form, enabling their digitization

### 1.4 Scope of System

The research will be of great assistance to various professional of related field such as archaeologists, paleographers, and linguists. Given the vast number of historical manuscripts of Kathmandu Valley archived in museums and historical places, the project's output will help scholars extract valuable information from these manuscripts accurately and efficiently. The digitized information can then be stored and archived for future use.

## 2. Related Works

Recently, deep learning-based methods have drawn increasing attention in handwritten character recognition.

Acharya et al. suggested a deep learning-based approach for character recognition in languages like Hindi, Nepali, and Marathi that employ the Devanagari script [1]. Their recommended design made use of deep belief networks (DBNs) and convolutional neural networks (CNNs) to achieve great accuracy in character recognition. The proposed deep learning architecture achieved an impressive accuracy of 98.47% on the Devanagari character recognition task. The 92 thousand pictures of 46 distinct classes of Devanagari characters that were carefully separated from handwritten texts made up the dataset utilized in this study. The authors used dataset increment and dropout as two strategies to increase the precision of their model, by using transformations like rotation, scaling, and translation on existing training samples, thus creating new training examples from them.

Sonawane et al. [2] have employed 16870 pictures of 22 often used Devanagari consonants to train a strong CNN namely AlexNet, achieving a validation accuracy of 94.49% and test accuracy of 95.46%. The study concluded transfer learning as a better alternative due to fast training and only requiring few samples. Das, et al. [3] have proposed a system for recognition of handwritten Bangla characters using Extended Convolutional Neural Network. The Bangla alphabet consists of total 84 classes of letters- 39 consonants, 11 vowels, 10 numerals and 24 combined characters. The model's accuracy rates for Bangla numerals were 99.50%, vowels were 93.18%, consonants were 90%, and mixed characters were 92.25%.

Prashanth et. al [4] have proposed a system for recognizing handwritten Devanagari characters, using modified LeNet and AlexNet CNN architectures. The goal of the experiment was to create a dataset of 38,750 pictures of Devanagari vowels and numerals, which was made available to other academics working in this field. The data was gathered from more than 3000 people of various ages. The segmentation method here was used to extract each character, and the dataset was used for the tests. Three different CNN architectures were experimented on; CNN, modified Lenet CNN (MLCNN) and Alexnet CNN (ACNN). Using CNN, accuracy was 96% on training data and 94% on unobserved data; MLCNN achieved these accuracy rates at a lower cost while ACNN achieved them at a higher rate. A minimal loss of 0.001% was discovered after a series of studies on the data using various combination splits of the data.

Mhapsekar, et. al [5] proposed a system for recognizing handwritten Devanagari characters using Residual Neural Network (ResNet) model. The ResNet 34 and ResNet 50 designs were employed, and the outcomes were compared with the most advanced convolutional neural network architecture, which has four and eight layers, respectively. Without segmenting handwritten line text image into words or characters, S. Gautam [6] has utilized CRNN to recognize it. The RNN model is trained using LSTM with Alex Graves CTC loss to solve the alignment issue in handwritten data.

## 3. Methodology

### 3.1 Dataset Creation and Acquisition

Creation of dataset involved collecting data from primary and secondary sources. Primary sources of data involved collecting handwritten data of school students, and collection of manuscripts and inscriptions.

In the first approach, different public schools in Kathmandu Metropolitan City were visited and forms as shown in fig. 2 were distributed to them. After collecting the forms, the forms were scanned and cropped for collecting the characters for dataset creation.

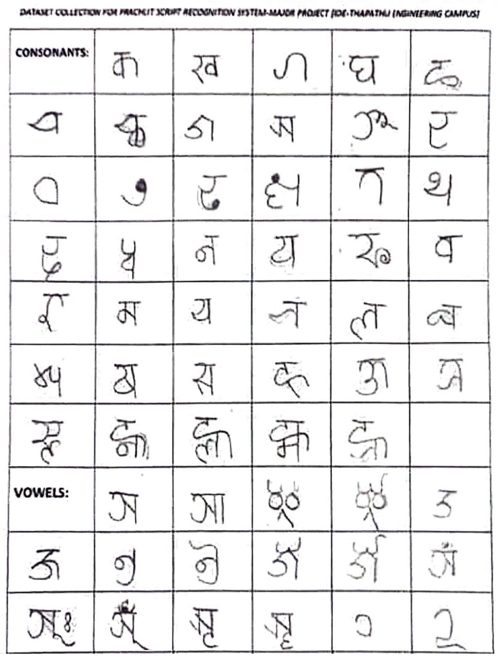
DATASET COLLECTION FOR PRACHLIT SCRIPT RECOGNITION SYSTEM-MAJOR PROJECT (IOE-THAPATHALI ENGINEERING CAMPUS)
CONSONANTS:
VOWELS:

**Figure 2:** Handwriting data collection

In the second approach, manuscripts were scanned to extract for individual characters. Primary sources of manuscript were the actual physical manuscripts available to us, of which we took photographs and scanned the image. Secondary sources of manuscript were the pre-scanned manuscripts available on the internet, which was simply downloaded and cropped as shown in fig. 3.

Other primary sources of dataset included stone and metal inscriptions available to us in various religious and heritage sites across the Kathmandu Valley. The sites were visited and photographs of inscriptions were taken, and akin to manuscripts, the images were cropped and characters were collected.

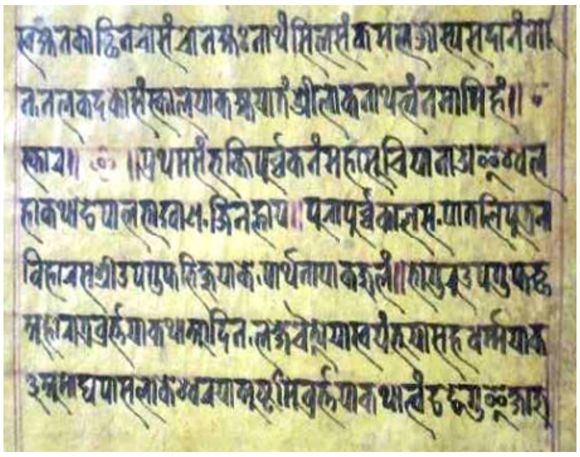

**Figure 3:** Manuscript data collection

## 3.2 Data augmentation

Data augmentation refers to the process of increasing the dataset for a machine learning model. Data augmentation involves various image transformation techniques, such as rotation, shearing, adding noise, changing contrast, among others. Augmenting the existing dataset, especially if the dataset is constructed from scratch, becomes beneficial for the machine learning model as it manually increases the dataset size and introduces additional noise which helps in regularizing the deep learning model and increasing the accuracy of recognition.

Since the dataset was constructed de novo, the dataset was augmented to eventually aid the deep learning model we implemented in the project. The necessity for data augmentation is that it primarily increases the size of dataset, as image augmentation can help to create more sets of data from a preexisting dataset. One image can be augmented several times allowing to capture minute variations in the data. Augmenting the dataset thus increases the robustness of dataset. Having augmented data points make model more robust as the model becomes less sensitive to small variations in the dataset.

The images collected during the dataset were divided into a total of 63 different classes, and grouped under three categories: digits, vowels and consonants. The 63 different classes included 10 vowels, 16 vowels and 37 consonants. While creating the dataset, we observed that some characters were used much more frequently than others, thus causing class imbalance. For this very reason, augmentation was necessary before training the model. Data augmentation considerably increased the dataset size and thus we maintained the dataset size at 800 characters per class. The prime reason for data augmentation is to prevent overfitting of the model. Overfitting occurs when the model is not able to generalize well to unseen data. Augmentation prevents overfitting by providing the model more diversed dataset to learn from allowing detection of more features.

The distribution of the classes in the dataset before and after augmentation is illustrated in fig. 5, fig. 4 and fig. 6.

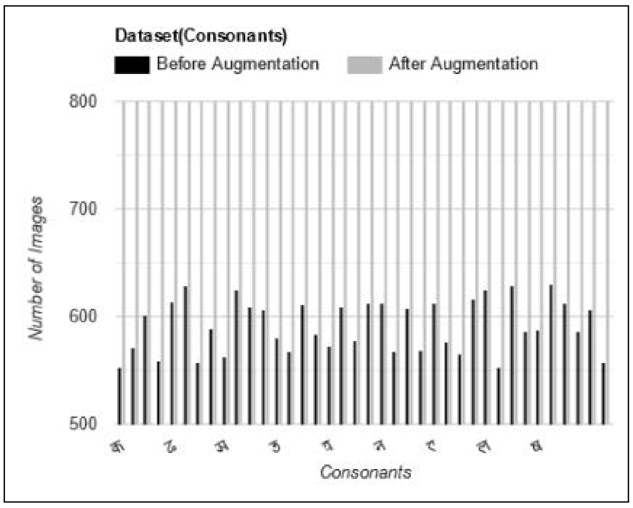

**Figure 4:** Distribution of Consonants

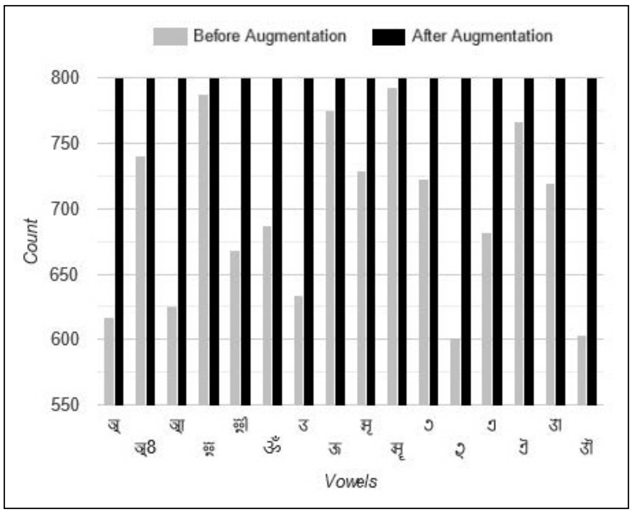

**Figure 5:** Distribution of Vowels

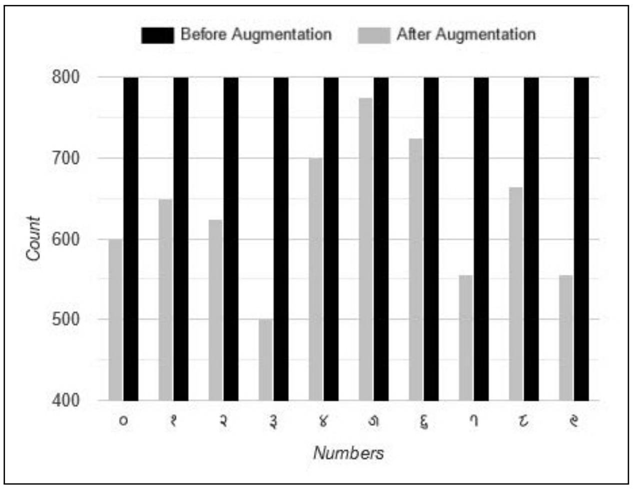

**Figure 6:** Distribution of Digits

## 3.3 Image preprocessing

Preprocessing is an essential process during the project implementation. This is because the images scanned from the sources were not uniform in nature, while a dataset consisting of uniform images were required for the model to be trained. Image preprocessing was carried out using the OpenCV cv2 library, where the images were converted to grayscale, followed by dimension reduction and resizing into 32×32 pixels resolution.

### 3.3.1 Grayscale Conversion

The function for converting an image to Grayscale is available in the OpenCV cv2 package and turns a 3-channel RGB image into a single grayscale image. Grayscale conversion also increases the computational efficiency of the model as it requires less computational resources to process. Grayscale conversion also increases the contrast of the images, color images have higher noise and distortion. Grayscale conversion is carried out using eq. 1 where 3-channel image is converted to a grayscale image.

$$\text{Gray} = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \tag{1}$$

Where R, G and B are the red, green and blue colour values of the image, each of which is in a range of [0,255].

### 3.3.2 Resizing the Image

All the images in the dataset were resized into 32×32 pixel resolution to maintain a uniform size input array. Resizing of an image is carried out by using a scaling factor to reduce an image of certain resolution to 32×32 while maintaining the image's aspect ratio. The images in dataset was bulk resized using OpenCV library and a consistent 32×32 image size was maintained for all images in the dataset. This operation is performed on both the training dataset and input images to ensure compatibility with the model's input requirements and reduce computational complexity. By standardizing image sizes we simplify the preprocessing pipeline, making the model more efficient, and potentially improving its generalization ability for better performance on diverse data.

### 3.3.3 Normalization

Normalization in image processing typically involves scaling the intensity values of the pixels in an image such that they fall within a certain range, usually between 0 and 255 (for 8-bit images) or 0 and 1 (for floating-point images).

Image normalization is a crucial step in image processing, aimed at reducing variations among input images. This process is essential for machine learning models to extract meaningful features and improve their ability to handle diverse data. By standardizing images and scaling them to a specific pixel value range, we enable models to recognize images regardless of their original pixel values. Four different CNN models were implemented for the project. LeNet, AlexNet, VGG and ResNet. This was carried out to test the accuracy for character recognition in different networks and to choose the best network based on accuracy, loss, precision and F1 score.

## 3.4 Supervised Deep Learning Model

A supervised Convolutional Neural Network (CNN) is trained using labeled data, where each input image has a corresponding label. The goal is to establish a mapping between inputs and outputs that generalizes well to new data.

In supervised CNN training, a large labeled dataset is used to adjust the model's weights based on the error between predicted and true labels. During training, the network learns to recognize various features in input images and combines them to make accurate predictions. Once trained, the network can predict labels for new, unseen images. Evaluation typically involves measuring accuracy on a validation or test dataset.

Convolutional Neural Networks are apt choice for character recognition system for several reasons. CNN are translation invariant, which means CNNs can recognize characters regardless of their position in an image, making them robust to variable character placement. Moreover, CNNs are efficient in feature extraction, and extract relevant image features like lines, edges, and shapes for predictions, while efficiently handling noise and degradation in images, outperforming traditional methods. With deep networks and techniques like transfer learning, CNNs achieve high accuracy in character recognition.

A deep research into different models used for optical character recognition, decisions was made for trying out 4 different models namely: VGG, Resnet, AlexNet, Lenet. Upon implementing the models on the dataset, the results obtained would be compared and contrasted, and the best performing model would be picked for further improvements on the project.

## 3.5 Implementation of Deep Learning Model

### 3.5.1 LeNet Architecture

The LeNet architecture was proposed by Lecun, et al [7] in 1998, which was a groundbreaking research in the field of optical character recognition. While the original LeNet architecture constituted 7 layers, the model we implemented is a modified LeNet consisting of 11 layers - 1 input layer, 2 convolutional layers, 2 max pooling layers, 3 dropout layers, 1 flatten layer and 2 dense layers. Sparse Categorical Crossentropy Loss was used, ReLU was used as the activation function in hidden layers and Softmax on the output layer, and Adam as the choice of optimizer. The model thus has 484,864 total parameters.

### 3.5.2 AlexNet Architecture

The AlexNet architecture is a groundbreaking deep learning architecture proposed by Krizhevsky, et al. [8] in 2012, which optimized the performance of neural networks in character recognition with a significantly powerful performance and reduced error rate. The AlexNet implemented involved 16 layers: 1 input layer, 6 convolutional layers, 3 max pooling layers, 2 dropout layers, 1 flatten layer and 3 dense layers. ReLU was used as an activation function with Softmax in the output layer. The loss function implemented was Sparse Categorical Crossentropy Loss, and Adam was used as the optimizer. The model thus has a total of 1,483,872 parameters.

**Table 1:** Specifications of different CNN models used

| Parameters | LeNet | AlexNet | ResNet | VGG |
|---|---|---|---|---|
| Batch size | 64 | 64 | 64 | 64 |
| Epochs | 50 | 50 | 50 | 50 |
| Image shape | (32, 32, 1) | (32, 32, 1) | (32, 32, 1) | (32, 32, 1) |
| Classification classes | 64 | 64 | 64 | 64 |
| Learning rate | 0.001 | 0.001 | 0.001 | 0.001 |
| Loss function | Sparse Categorical Crossentropy | Sparse Categorical Crossentropy | Sparse Categorical Crossentropy | Sparse Categorical Crossentropy |
| Optimizer | Adam | Adam | Adam | Adam |
| Filters | 64, 128 | 128 | 64, 128 | 32, 64, 128 |
| Kernel size | (3, 3) | (3, 3) | (3, 3) | (3, 3) |
| MaxPool size | (2, 2) | (2, 2) | (2, 2) | (2, 2) |
| Dropouts | 0.25, 0.25 | 0.25, 0.25 | 0.25, 0.50 | 0.25, 0.25, 0.25, 0.50 |
| Activation function | ReLU, Softmax | ReLU, Softmax | ReLU, Softmax | ReLU, Softmax |
| Total parameters | 484,864 | 1,483,872 | 9,390,784 | 1,368,928 |
| Trainable parameters | 484,864 | 1,483,872 | 9,390,784 | 1,368,928 |
| Nontrainable parameters | 0 | 0 | 0 | 0 |

### 3.5.3 ResNet Architecture

The ResNet (Residual Network) model was introduced in a paper titled “Deep Residual Learning for Image Recognition” by Kaiming He et al [9] in 2016. ResNet in particular is known for skipping layers during the learning process. In ResNet, the weight layers learn from residual functions with reference to input layers. The skip connections perform identity mappings and are merged with the layer outputs by addition, enabling the deep learning models with tens or hundreds of layers to train easily and aproach better accuracy when going deeper.

In our implementation of ResNet, 15 layers were used to define the model. The original Resnet Architecture included 12 layers, consisting of 1 input layer, 6 convolutional layers, 2 max pooling layers, 2 dropout layers, 1 flatten layer and a dense layer. Like previous models, the loss function used was Sparse Categorical Crossentry, ReLU and Softmax were used as activation functions and Adam was used as the optimizer. The model has 9,390,784 total parameters involved.

### 3.5.4 VGG Architecture

VGG, which stands for Visual Geometry Group, is a very deep convolutional neural network originally proposed by Zisserman and Simonyan [10] in 2014. A deep convolutional neural network several multiple layers, the number of layers in VGG vary depending on different architectures of the model. VGG replaces the larger kernel sizes with several layers of small kernel sizes, usually (3×3), one after another, thus making significant improvements over feature extraction compared to other CNN models.

We implemented a modified VGG architecture which includes 17 layers. The layers of the model include 1 Input layer, 6 Convolutional layers, 3 Max Pooling layers, 4 Dropout layers, 1 Flatten layer and 2 Dense layers. The loss function used was Sparse Categorical Crossentropy Loss. ReLU was used as the activation function in hidden layers and Softmax as the activation on the output layer. Adam was used as the optimizer. The model has 1,368,928 total parameters involved.

## 4. Results

A total of 64 characters, has been taken for the dataset creation: 38 consonants with some combined characters, 16 vowels and 10 numerals. Since there was no dataset available for the Prachalit script, dataset was built from scratch, through collection from various sources, such as manuscripts and hand-written samples. Eventually, a total of 51200 images, of 64 classes, containing 800 images in each class, were trained on 4 different CNN Architectures.

Five different CNN models were trained in Google Colab notebook each for 50 epochs. The subsequent results obtained were as follows.

**Table 2:** Observations using Different CNN models

| Model | Early Stopped | Precision | Recall | F-1 |
|---|---|---|---|---|
| **AlexNet** | 22 | 0.91 | 0.91 | 0.91 |
| **LeNet** | 50 | 0.86 | 0.86 | 0.86 |
| **ResNet** | 17 | 0.92 | 0.92 | 0.91 |
| **VGG** | 34 | 0.96 | 0.96 | 0.96 |

**Table 3:** Training, Validation, and Testing Parameters for Different CNN Models

| Model | Training Parameters | | Validation Parameters | | Testing Parameters | |
|---|---|---|---|---|---|---|
| | Loss | Accuracy | Loss | Accuracy | Loss | Accuracy |
| **AlexNet** | 0.11 | 0.96 | 0.29 | 0.93 | 0.30 | 0.92 |
| **LeNet** | 0.44 | 0.86 | 0.49 | 0.88 | 0.49 | 0.88 |
| **ResNet** | 0.11 | 0.96 | 0.31 | 0.93 | 0.26 | 0.93 |
| **VGG** | 0.20 | 0.93 | 0.18 | 0.95 | 0.15 | 0.95 |

Upon comparison it was found that the LeNet model was not able to work properly for image classification task as it has higher value of testing loss and lower value of accu racy on unseen data. Whereas, VGG and ResNet and AlexNet had better performance than the LeNet model. But, VGG model was selected for the reason for it having a better performance in the test(unseen) data set. The training loss is lowest among the models tried which indicates that the model is being able to generalize well on unseen data and is not overfitting which is also supported by the highest value of its testing accuracy. The VGG model has precision, recall and F1 score of 0.96 , 0.96 and 0,96 respectively indicate that the model is able to classify images with a higher level of confidence. A greater recall number demonstrates the classifier's ability to accurately identify more positive cases, while a higher precision value shows that the classifier is generating fewer incorrect positive predictions. A classifier that performs well in accuracy and recall has a higher F1 score. On comparison, the precision, recall and F1 score of the VGG model is found to be highest among the models tried showing that the model is able to identify more positive cases, and fewer incorrect positive predictions.

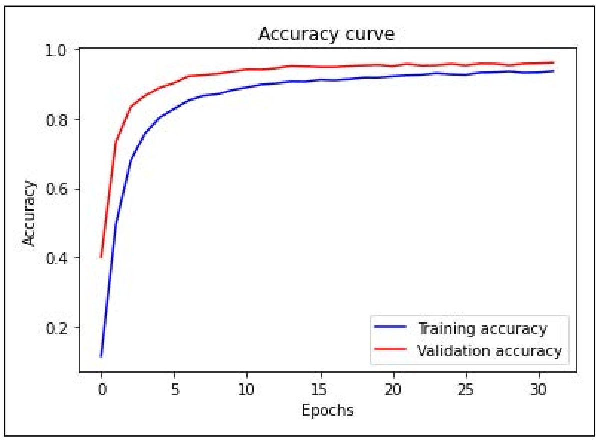

**Figure 7:** Accuracy Curve of the best performing model

The training accuracy is 0.93, validation accuracy is 0.95 and training accuracy is 0.95. The small gap between the training and validation accuracy also shows that the model is not over fit to the training data. The higher value of validation and testing accuracy compared to training accuracy shows that the model is being able to recognize and generalize to the unseen data.

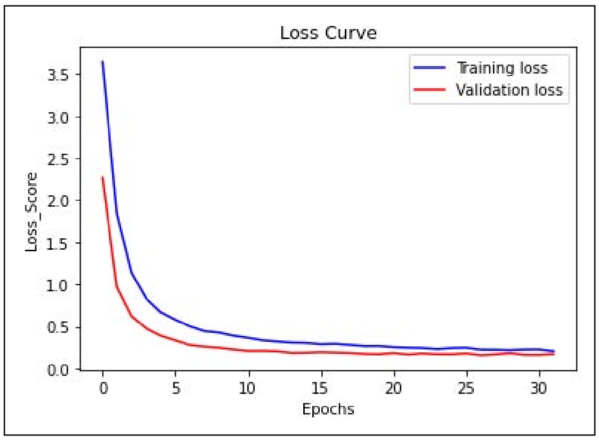

**Figure 8:** Loss curve of the best performing model

The training loss is 0.20, validation loss is 0.18 and testing loss is 0.15. The lower value of the loss in the training data shows that the model is able to adjust its weights effectively. The validation loss of 0.18 is slightly less than the training loss indicating that the model is able to generalize well to data in validation set. The testing loss of 0.15 is slightly lower than the validation loss which shows that the model is performing well on unseen data.

## 5. Conclusion

The project involved the creation of a Prachalit Script dataset comprising 51,200 images distributed across 64 classes, including consonants, vowels, and numerals, with 800 images per class. These images were meticulously sourced from old manuscripts and handwritten documents. Augmentation techniques, such as rotation, skewing, and shearing, were applied to expound the dataset. The dataset serves as a valuable resource for training machine learning models, with a particular focus on Convolutional Neural Networks (CNNs) due to their data invariance and noise resistance properties.

Four CNN architectures, namely LeNet, AlexNet, ResNet, and VGG16, were trained on the dataset. Among these models, VGG16 demonstrated the highest accuracy and better generalization to unseen data. However, the presence of characters that closely resembled each other posed challenges in categorization and feature extraction, leading to occasional errors in predictions.

To mitigate overfitting, various techniques were employed, including early stopping, dropout layers for regularization, and hyperparameter tuning. The incorporation of dropout layers proved effective in classifying complex and similar characters within the dataset.

While the project is an initiation on character recognition of ancient and medieval writing systems and yet a lot of work is to be done for much more robust and efficient development of the character recognition, these results demonstrate the potential of supervised machine learning in supporting conservation efforts to preserve language and culture.

## Acknowledgments

The authors express their deepest gratitude to the Department of Electronics and Computer Engineering, Thapathali Campus, Institute of Engineering, for all the inputs provided to carry out this study.

## References

[1] S. Acharya, A. K. Pant, and P. K. Gyawali. Deep learning based large scale handwritten devanagari character recognition. In *2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, pages 1–6. IEEE, 2015.

[2] P. K. Sonawane and S. Shelke. Handwritten devanagari character classification using deep learning. In *2018 International Conference on Information, Communication, Engineering and Technology (ICICET)*, pages 1–4. IEEE, 2018.

[3] T. R. Das, S. Hasan, M. R. Jani, F. Tabassum, and M. I. Islam. Bangla handwritten character recognition using extended convolutional neural network. *Journal of Computer and Communications*, 9(3):158–171, 2021.

[4] D. S. Prashanth, R. V. K. Mehta, K. Ramana, and V. Bhaskar. Handwritten devanagari character recognition using modified lenet and alexnet convolution neural networks. *Wireless Personal Communications*, 122:349–378, 2022.

[5] M. Mhapsekar, P. Mhapsekar, A. Mhatre, and V. Sawant. Implementation of residual network (resnet) for devanagari handwritten character recognition. In *Advanced Computing Technologies and Applications: Proceedings of 2nd International Conference on Advanced Computing Technologies and Applications—ICACTA 2020*, pages 137–148. Springer, 2020.

[6] S. Gautam. Devnagari handwritten word recognition with deep learning. 2019.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.