San Jose State University

Department of Computer Science

Devanagari Handwritten Character Recognition

CS 271 Final Project Report

Submitted by:

Devam Sanjay Sheth (017404218)
Komal Sanjaykumar Joshi (017463459)

Submitted to:

Professor Mark Stamp

Date of Submission:
December 4, 2024

Contents

1 Introduction

One of the main challenges in the field of computer vision has been handwriting identification, which has various applications like bank signature verification, postal code recognition, and processing bank cheques. Especially, recognizing handwritten characters in non-English scripts poses additional difficulties due to the intricate shapes of the characters. There has been thorough research going on this with the help of common databases to evaluate and compare techniques based on metrics like accuracy, processing time, and complexity. Indian languages such as Hindi, Gujarati, Sanskrit, Marathi, and Sindhi utilize the Devanagari script, which adds to the complexity. Figure 1 of the Devanagari Handwritten Character Dataset (DHCD) illustrates a few alphabets.

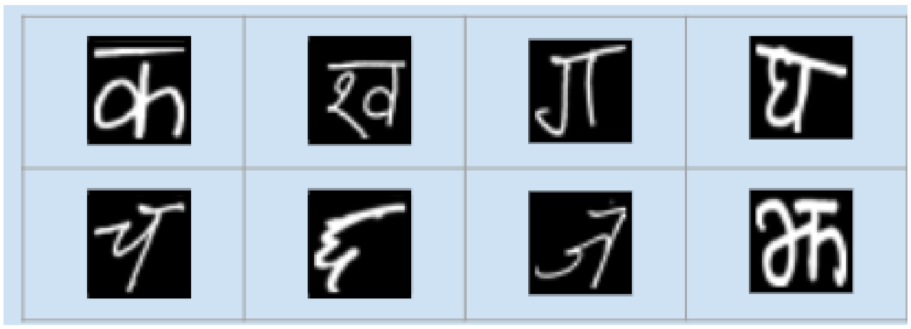
Figure 1: Devanagari Characters

Recognition of Devanagari characters is particularly challenging because many characters share similar shapes. Researchers have developed handcrafted features to analyze strokes for recognition, but creating a reliable feature set for Devanagari remains difficult due to the complex structure of its characters.

Convolutional Neural Networks (CNNs) have gained prominence in computer vision and pattern recognition. CNNs combine convolutional layers for feature extraction, fully connected layers for image representation, and pooling layers for dimensionality reduction, along with activation functions like sigmoid or ReLU for introducing non-linearity. CNNs leverage local receptive fields and shared weights. Local receptive fields capture localized information across an image, with each field representing a neuron, and shared weights enhance the efficiency of the hidden layers.

Techniques like dropout, where some hidden neurons are randomly deactivated, further help to overcome overfitting. Pre-trained CNN models such as EfficientNets or MobileNets are trained on extensive datasets, often with thousands of class labels. Transfer learning enables the application of these pre-trained models to tasks with limited data.

In this paper, we have trained our devanagari character dataset on different machine learning algorithms and carried out a comparative analysis among all of them. Following are the main models trained on:

1. Principle Component Analysis (PCA): For dimensionality reduction.
2. Multi-layer Perceptron (MLP), Naive Bayes Classifier, k-Nearest Neighbors (KNN), : Classical machine learning models for classification on PCA transformed data.
3. Convolutional Neural Network (CNN): Training a CNN network from scratch.
4. EfficientNet: A pre-trained CNN which is known for efficient computation.
5. MobileNet: A pre-trained CNN specially designed for small devices.
6. Deep Belief Networks (DBNs): A type of generative deep learning model composed of multiple layers of Restricted Boltzmann Machines (RBMs), stacked on top of each other.

2 Data Description

The Devanagari Handwritten Character Dataset (DHCD) is an extensive collection of grayscale images that represent handwritten characters from the Devanagari script. This script is used in languages such as Hindi, Marathi, and Sanskrit. The dataset was compiled from handwritten samples provided by diverse participants, including students from Mount Everest Higher Secondary School in Bhaktapur, Nepal. These samples were scanned and manually segmented to isolate each character, a method that ensured a variety of handwriting styles and enhanced the robustness of the dataset. The dataset is publicly available through the UCI Machine Learning Repository and designed to support the development and evaluation of Optical Character Recognition (OCR) systems and machine learning models.

Dataset Link: https://archive.ics.uci.edu/dataset/389/devanagari+handwritten+character+dataset

This dataset consists of 46 distinct character classes, including 36 consonants and 10 numerals. Each class contains 2,000 samples, totaling 92,000 images. The images are formatted in grayscale, each with a resolution of 32x32 pixels and centered within a 28x28 pixel area with a 2-pixel padding on all sides to ensure consistency.

For machine learning applications, the dataset is divided into a training set comprising 85% of the data (78,200 images) and a testing set consisting of the remaining 15% (13,800 images). This partitioning facilitates effective training and evaluation of models.

3 Algorithms Used

This section offers an overview of the technical descriptions and background of the dimension reduction method, PCA, machine learning model, and deep learning models used in this study. This section also introduces Deep Belief Networks.

3.1 Principal Component Analysis (PCA)

PCA is a statistical technique that reduces the dimensionality of data while preserving as much variability as possible. It transforms a set of possibly correlated variables into a smaller number of uncorrelated variables called principal components. The transformation aims to retain the components that explain a substantial amount of variance, which are often the first few principal components. The primary goal of PCA is to project the original variables into a new coordinate system, where the greatest variance by any projection of the data comes to lie on the first coordinate, the second greatest variance on the second coordinate, and so on.

PCA is particularly effective in situations where the dimensionality of the data is high relative to the number of observations and hence, we will be using it on our dataset. PCA is sensitive to the scaling of the variables and assumes that components with larger variance are more important, potentially overlooking important variables that may have smaller variance.

3.2 Multi-layer Perceptron (MLP)

The Multi-layer Perceptron (MLP) Classifier is a type of feedforward neural network widely used for classification tasks. It works by learning complex relationships between features and target labels through multiple layers of interconnected neurons.

MLP is versatile and can model non-linear decision boundaries effectively, making it a powerful tool for complex datasets. However, it requires a considerable amount of data and computational power, particularly as the number of layers and neurons increases. It is sensitive to hyperparameters like learning rate, number of epochs, and architecture design. Preprocessing steps such as normalization and feature scaling are essential to enhance its performance, and using PCA-transformed data can help reduce computational complexity.

3.3 Naive Bayes Classifier

Naive Bayes is a straightforward and highly effective probabilistic machine learning algorithm. It operates based on Bayes' theorem and assumes that the features are conditionally independent given the class label.

The algorithm is computationally efficient and works well with both small and large datasets. However, its performance can be affected by the presence of highly correlated features. Hence, preprocessing like feature selection or dimensionality reduction is important for enhancing its effectiveness. By applying it to transformed data such as that produced by PCA, the robustness and interpretability of Naive Bayes can be further optimized.

3.4 k-nearest neighbors (KNN)

K-Nearest Neighbors (KNN) is a simple but powerful machine learning algorithm used for both classification and regression tasks. It operates based on the intuitive principle that similar data points are likely to have similar outcomes.

KNN can become computationally expensive as the dataset size increases, and it is sensitive to the scale of data and the presence of noisy features. However, KNN is a baseline method for many classification and regression problems due to its robustness and ease of interpretation. Preprocessing steps such as feature scaling and dimensionality reduction can significantly improve its performance and hence, we are using PCA transformed data for KNN implementaion.

3.5 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a specialized type of deep learning model designed for processing structured data such as images, videos, or other grid-like data types. A CNN applies this principle by using convolutional layers that extract spatial features from the data. At the core of a CNN are three key operations: convolution, pooling, and fully connected layers. Convolution operation preserving the spatial relationship between pixels for tasks where positional information is critical, pooling to reduce the spatial dimensions of feature maps, and fully connected layers to perform classification or regression tasks.

CNNs also use activation functions like ReLU (Rectified Linear Unit) to introduce non-linearities which enables them to learn complex mappings between inputs and outputs. Weight-sharing in convolutional layers reduces the number of parameters significantly compared to traditional fully connected neural networks and hence, CNNs are computationally efficient.

3.6 EfficientNet

EfficientNet is a pre-trained CNN designed to achieve high performance with significantly fewer parameters and computational costs compared to traditional deep learning models. It uses compound scaling, which balances the number of channels, number of layers, and input image size in a systematic way. This allows EfficientNet to scale up or down efficiently for different resource constraints, making it versatile for a wide range of applications.

Scaling CNNs involves increasing one dimension, like depth, leading to diminishing returns or overfitting. EfficientNet instead uses a set of fixed scaling coefficients to expand the network in width, depth, and resolution simultaneously, ensuring balanced growth. Additionally, EfficientNet incorporates features like squeeze-and-excitation (SE) blocks to adaptively recalibrate channel importance, further improving performance.

3.7 MobileNet

MobileNet is a pre-trained CNN designed for mobile and embedded devices, where computational resources and power consumption are limited. MobileNet focuses on reducing the size and complexity of deep learning models without significantly sacrificing accuracy. The key innovation in MobileNet is the use of depthwise separable convolutions, which split the standard convolution operation into two smaller, more efficient steps: a depthwise convolution and a pointwise convolution. This drastically reduces the number of computations and parameters and hence, MobileNet is lightweight and fast.

MobileNet also allows for trade-offs between accuracy and efficiency using two hyperparameters, width multiplier and resolution multiplier. The width multiplier reduces the number of channels in each layer, while the resolution multiplier reduces the input image size. This enables customization of the model for various resource constraints. All these features make MobileNet highly versatile, with applications in real-time tasks like object detection, image classification, and face recognition on mobile devices.

3.8 Deep Belief Networks (DBNs)

Deep Belief Networks (DBNs) are a class of generative models in machine learning, primarily utilized for unsupervised learning and feature extraction. They are structured as layered stacks of Restricted Boltzmann Machines (RBMs), with each layer learning to capture increasingly abstract representations of the data.

DBNs operate by modeling the joint probability distribution of observed data and their hidden representations. The bottom layer receives raw data as input, while the top layer encodes a probabilistic representation of the data. This approach makes DBNs especially effective for dimensionality reduction, feature learning, and pre-training deep neural networks. It often improves model performance and convergence in scenarios with limited labeled data. DBNs are valued for their ability to uncover latent structures within complex datasets.

4 Methodology - Comparative Analysis of Algorithms

As our data was already in black-and-white format, there was least requirement of any kind of pre-processing. Though, we augmented the images for better training of the models on and reduce bias. We assumed this would help models to understand the pattern of each character to recognize it.

Data augmentation is a critical preprocessing step used to artificially expand the size and diversity of a dataset by applying various transformations to the input data. Common augmentation techniques include random flipping, rotation, zooming, cropping, translation, and color adjustments. These transformations simulate real-world variations in the data, such as changes in orientation, scale, or lighting conditions, without altering the underlying labels.

We initially performed PCA over the image dataset, reducing the dimension of dataset for easier training on required components. Based on the components, we performed classical machine learning technique KNN, MLP, and Naive Bayes for classification of characters and numbers.

A CNN model was designed to train it from scratch. It contains 3 layers of convolutions, along with max-pooling layers to extract features from the images. Convolution layers are then followed by Dense layers for computation. At each convolution layer, we are normalizing data for consistency. Figure 2. below shows the CNN architecture used to train.

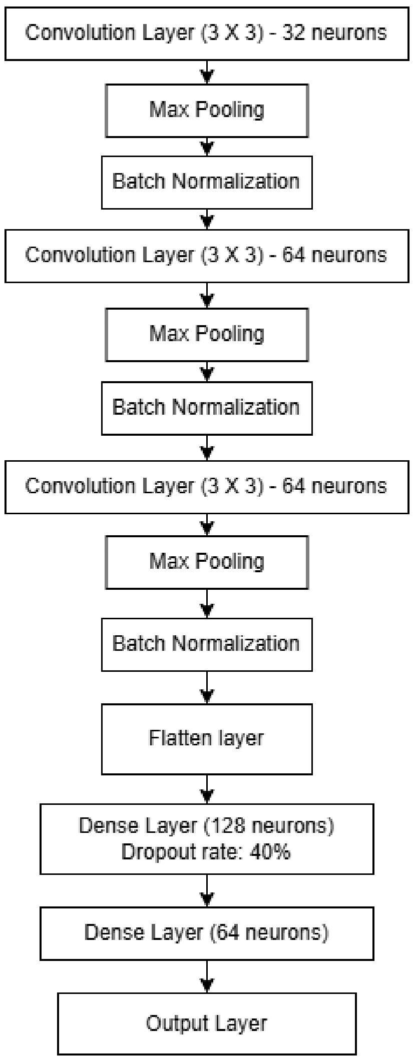

Figure 2: CNN Architecture

Two pre-trained models, EfficientNet and MobileNet were fine-tuned with the dataset as well. They were considered based on their capabilities and their base architecture being CNNs. Apparently, they converged earlier than the original CNN model with better accuracies.

We considered Python Tensorflow Library to build the models and train them. For each epoch, a few parameters were observed to monitor the training of models, such as training accuracy, validation accuracy, training loss, validation loss, and f1-score.

Apart from CNN based models, we also worked with Deep Belief Networks (DBNs). As it was difficult to find a library implementing DBNs, we build it from scratch with the help of TensorFlow. We developed two classes: RBM and DBN. RBM, corresponding to Restricted Boltzmann Machine, is the base class containing the all the functions required to model the dense layers including forward and backward pass. DBN are built by stacking up RBMs.

Initially, DBNs are pre-trained with images, just with forward pass, to initialize the weights. After pre-training, a dense layer per RBM layer is added with initialized weights. Post which fine-tuning is carried out, which is similar to training the whole model.

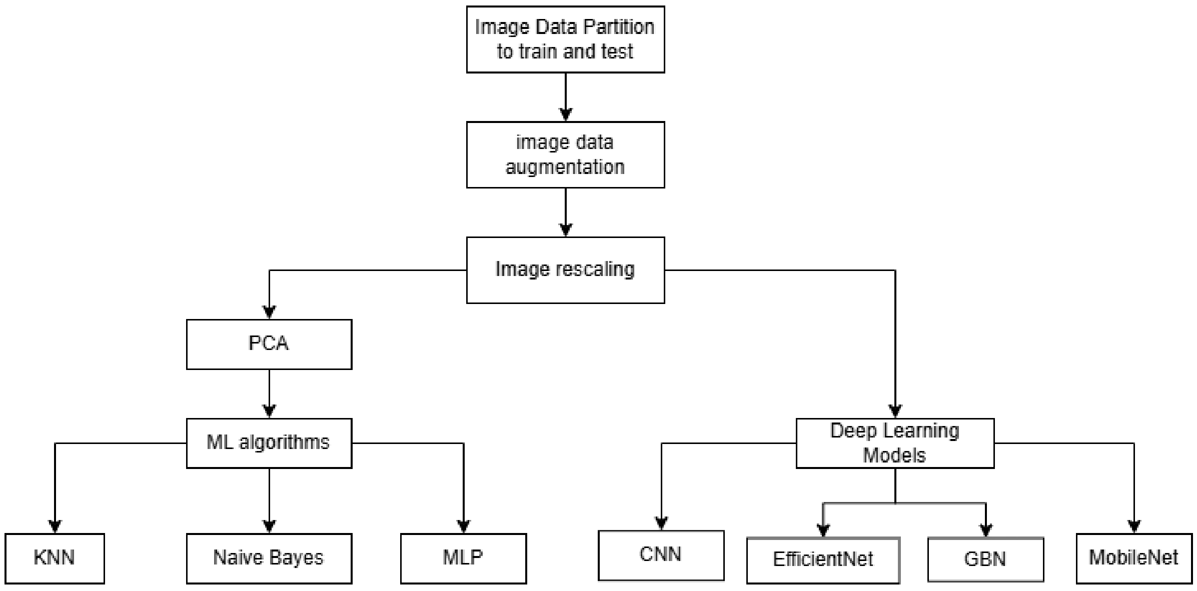

Figure 3: Workflow

5 Applications and Results

5.1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is used here for dimensionality reduction. We started by converting TensorFlow's dataset into NumPy arrays to simplify handling and merging all batches into a single large array. Each image was flattened into a one-dimensional vector Next, we mean-centered the data to facilitate PCA in identifying directions of maximum variance and calculated the covariance matrix to discern correlations among pixel features. Eigen decomposition was performed to find eigenvalues and their corresponding eigenvectors which indicate the directions and magnitude of maximum variance. Then, we decided to select the top 127 eigenvectors related to the largest eigenvalues to retain the most informative features (95% variance), reducing dimensionality from 3072 to 127. The goal was to simplify the dataset while preserving essential patterns.

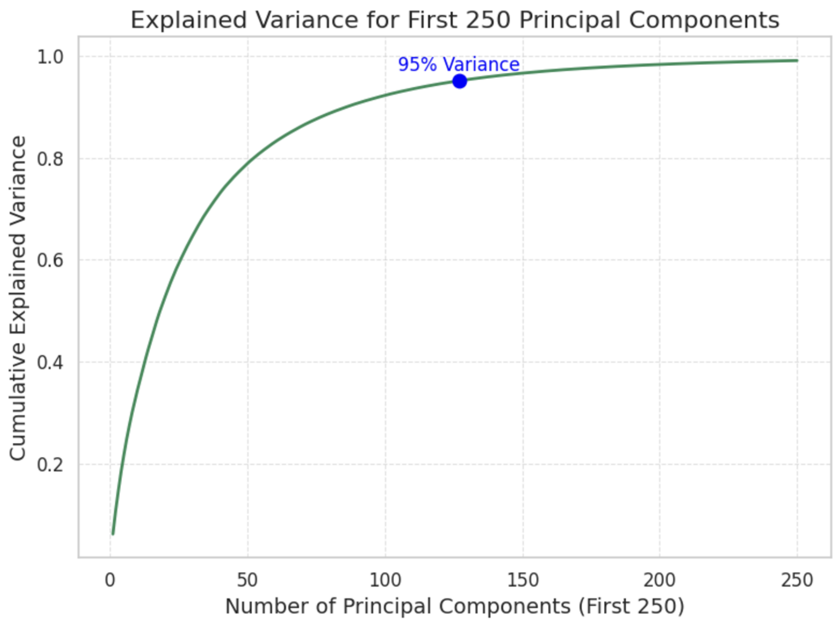

Figure 4: PCA Result

The primary strengths of PCA are:

1. Ability to reduce dimensionality, which simplifies the model's complexity and mitigates overfitting issue.

2. Removes less significant components and hence decreases computational load as well as reduces memory usage, boosting efficiency.

However, PCA has limitations, which include:

1. The potential loss of important information when reducing dimensionality.

2. Has an assumption of linearity, which may not capture complex, nonlinear relationships effectively. Inappropriate scaling can lead to misleading results.

3. Effectiveness depends on the assumption that components with the highest variance are the most informative, which is not always the case.

5.2 Multi-layer Perceptron (MLP)

Models	MLP	Naive Bayes	KNN
Validation Accuracy	66.6%	54.1%	89.6%
F1-Score	76.44%	55.44%	89.66%

Table 1: Classical Model Results

Figure 5: KNN Result

MLP classifier achieved 66.6% accuracy and 76.44% F1-score, indicating moderate performance. The relatively higher F1-score suggests it captures class distinctions well, though further tuning could enhance results.

Below are its advantages:

1. Capable of learning complex patterns in data due to its multi-layer architecture and non-linear activation functions.
2. Versatile for both classification and regression tasks, adaptable to diverse applications.
3. Effective at handling high-dimensional datasets, leveraging backpropagation for efficient parameter optimization.

MLP has the following disadvantages:

1. Computationally expensive, especially for large datasets and deep networks.
2. Susceptible to overfitting if not properly regularized or if the dataset size is small.
3. Sensitive to the choice of hyperparameters like the learning rate, number of layers, and nodes in hidden layers.

5.3 Naive Bayes Classifier

Naive Bayes classifiers are probabilistic models based on Bayes' Theorem. The Naive Bayes Classifier had an accuracy of 54.1% and an F1-score of 55.44%, showing it struggled with the complexity of the dataset. Its assumption of feature independence makes it less suited for this image data.

Its advantages are:

1. Handles large datasets effectively and processes them quickly.
2. Works well even if some features are irrelevant.
3. Performs exceptionally well in natural language processing tasks like spam detection and sentiment analysis.

Its disadvantages are as below:

1. Assumes independence between features, which might not hold in real-world scenarios.
2. Assigns zero probability to a class if a feature has not been observed, though this can be mitigated using techniques like Laplace smoothing.
3. Struggles with complex relationships and highly interdependent features in datasets.

5.4 k-nearest neighbors (KNN)

KNN's simplicity and ease of implementation make it a good choice for many classification tasks, particularly when paired with PCA for preprocessing. With $k = 7$, it outperformed the other models. It achieved 89.6% accuracy and 89.66% F1-score. The similarity-based approach works effectively here, benefiting from PCA's preservation of essential feature relationships.

1. Allows us to choose different distance metrics (Euclidean, Manhattan, Minkowski) based on the specific nature of the data.
2. Straightforward to implement and doesn't require complex algorithms.
3. The classification can immediately proceed based on the proximity of nearest neighbors without a need for a training phase.

KNN has below disadvantages:

1. As the size of the dataset increases, KNN's performance may degrade due to the need to compute distances to every training sample.
2. Remains sensitive to noise in the data, even after dimensionality reduction.
3. Choosing the correct number of neighbors is important as too less neighbors can make algorithm too sensitive to noise and too many can obscure meaningful class boundaries.

5.5 Convolutional Neural Networks (CNNs)

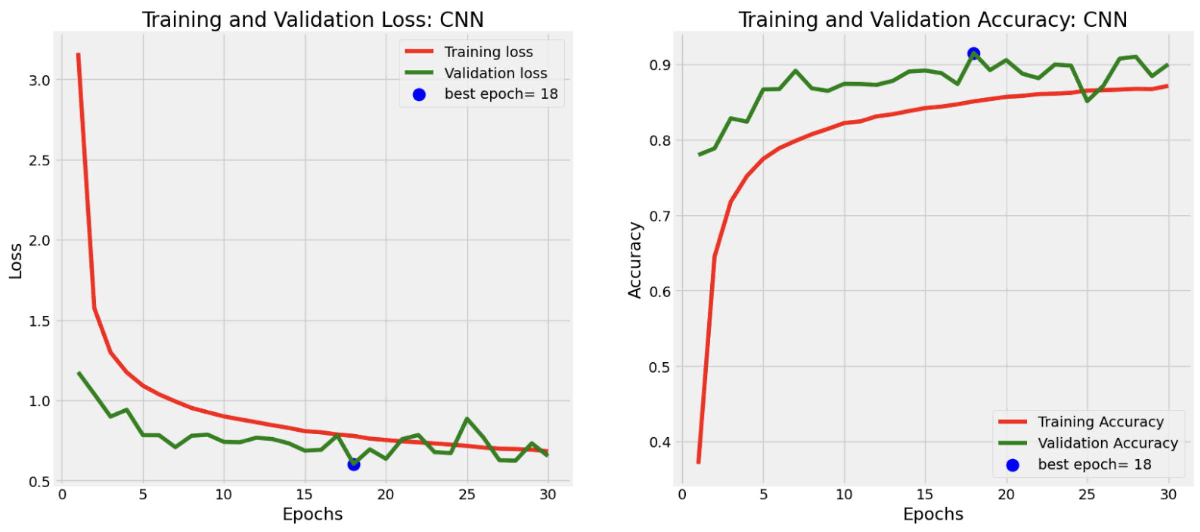

Figure 6: CNN Result Plots

The Convolutional Neural Network (CNN) demonstrated a consistent learning pattern over 30 epochs, with training accuracy peaking at about 92.17% and validation accuracy at around 90.8%. This minimal discrepancy between the training and validation metrics indicates effective generalization with limited overfitting, highlighted by the best epoch at 18 where validation loss was minimized—a strategy to ensure the model performs well on new data.

The model tends to have a good amount of advantages for image classification:

1. Ability to learn hierarchical feature representations from image data.
2. Learn efficiently over time.
3. Handles spatial data hierarchies that is essential for interpreting spatial correlations in images.

However, CNNs do have drawbacks:

1. Need extensive data to prevent overfitting.
2. Computationally intensive, requires powerful hardware to manage the training process.
3. Sensitive to initial weights and learning rate.

5.6 EfficientNet

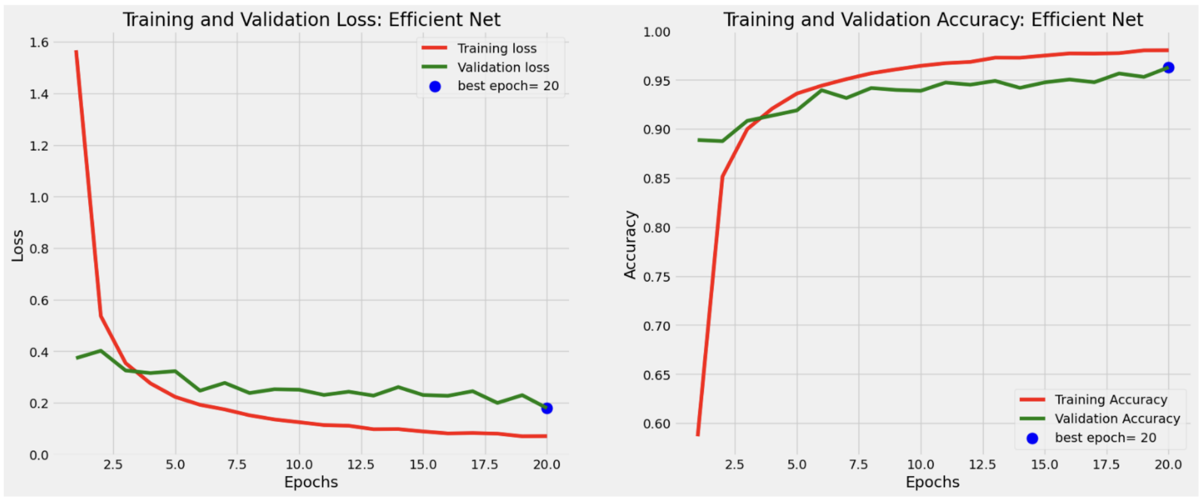

Figure 7: EfficientNet Result Plots

The EfficientNet model displayed robust performance on the dataset, achieving high accuracy and low loss across 20 epochs, with training accuracy peaking at about 98.2% and validation accuracy at 95.9%. The model's rapid convergence and minimal divergence between training and validation indicate strong generalization capabilities, selecting the best model at epoch 20 to balance performance without overfitting.

EfficientNet's architecture provides its own benefits:

1. Allows uniform scaling of model dimensions—depth, width, and resolution.
2. Scaling training and validation metrics beneficial for managing diverse and complex image datasets efficiently without compromising accuracy.

Despite its advantages, EfficientNet's has a few drawback:

1. Demands significant computational resources, especially for scaling up in larger datasets.
2. Dependency on scaling could also limit its adaptability to datasets.
3. Continuing training beyond best epoch could result in model over-fitting.

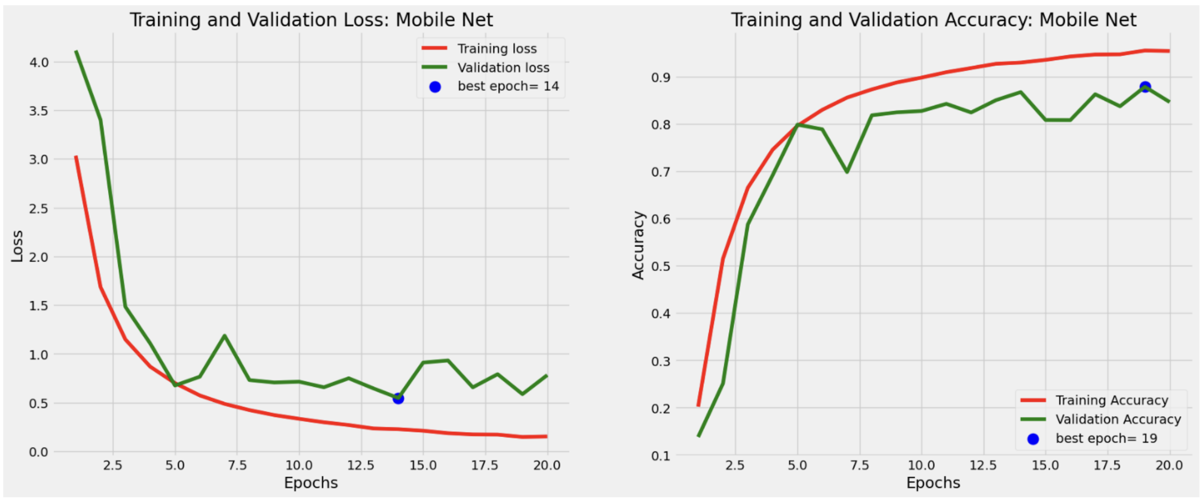

Figure 8: MobileNet Result Plots

5.7 MobileNet

The MobileNet model showed promising results over 20 epochs, with training accuracy rising sharply to above 90.72% and validation accuracy stabilizing at around 86.98%. This indicates a modest gap between training and testing performances, suggesting slight overfitting but within acceptable limits. The optimal stopping point was identified at epoch 19 to best balance validation performance and minimize overfitting.

MobileNet is renowned for its efficiency on mobile devices and provides similar advantages over other models:

1. Offers a balance between speed and accuracy.
2. Uses depthwise separable convolutions which significantly reduce the number of parameters compared to traditional convolutional layers.
3. Performs well across a wide range of image sizes and quality which makes it versatile for various applications.

Alongwith its advantages, MobileNet comeswith its disadvantages:

1. Lag slightly in accuracy compared to more complex models, especially with complex image datasets which need detailed feature detection.
2. May struggle with complex image recognition tasks that require detecting fine-grained features.
3. Hyperparameter tuning, particularly the depth and resolution multipliers, is challenging.

5.8 Deep Belief Networks (DBNs)

We went over 40 epochs for fine-tuning DBN to enhance model's performance on training and validation dataset. Before fine-tuning, there were 10 epochs for pre-training the model to initialize weights as per the images. The training loss consistently decreases over the epochs, indicating that the model is effectively learning from the training data. However, the validation loss plateaus after epoch 37, suggesting the model has reached its optimal performance on the validation set. Similarly, training

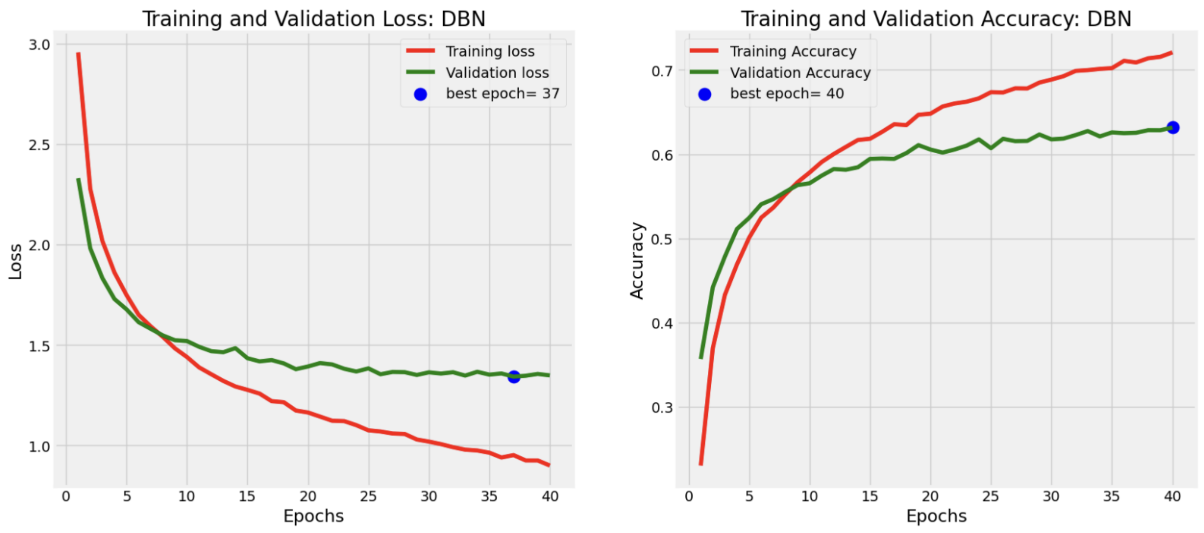

Figure 9: DBN Result Plots

accuracy continues to improve, while validation accuracy stabilizes, with the best validation performance observed around epoch 40.

Advantages for Deep Belief Networks are:

1. Cab be trained one layer at a time, which allows each layer to stabilize before moving on to the next, leading to efficient training process.
2. Have strong generative capabilities which makes them useful for applications such as image reconstruction, enhancement, etc.

Despite these advantages, it has below limitations:

1. Can struggle with scalability when dealing with extremely high-dimensional data.
2. Training DBNs can be computationally expensive due to their deep architecture and the iterative nature of the training process.
3. Performance is very sensitive to the settings of various hyperparameters, including the number of layers, the number of units in each layer, and the learning rates.

5.9 Summarised Results

On comparing performances of all the models, we found that EfficientNet outperformed all the models, based on training accuracy, validation accuracy, and f1-score. Plain CNN is near to EfficientNet. DBN was found to have an average performance.

On comparing classic Machine Learning models, which followed PCA, performed relatively poor than deep learning models. Among ML models, KNN outperformed the rest of the two models. KNN had even performed better than DBN as well.

Models	MLP	Naive Bayes	KNN
Validation Accuracy	66.6%	54.1%	89.6%
F1-Score	76.44%	55.44%	89.66%

Table 1: Classical Model Results

Models	CNN	EfficientNet	MobileNet	DBN
Training Accuracy	92.17%	98.2%	90.72%	83.80%
Validation Accuracy	90.8%	95.9%	86.98%	75.26%
F1-Score	86.88%	96.56%	87.46%	74.74%

Table 2: Deep Learning Model Results

6 Conclusion and Future Work

In this study, we explored the task of handwritten character recognition for the Devanagari script by evaluating the performance of multiple machine learning and deep learning models. The dataset posed unique challenges due to the complex shapes of characters and the presence of visually similar groups, requiring robust feature extraction and classification techniques.

We applied dimensionality reduction through Principal Component Analysis (PCA) to preprocess the data for K-Nearest Neighbors (KNN). For deep learning models such as Convolutional Neural Networks (CNNs), EfficientNet, MobileNet, and Generative Belief Networks (GBNs), raw image data was directly used to leverage their feature extraction capabilities.

The results revealed that while traditional ML models like KNN, Naive Bayes, and MLP, coupled with PCA, can achieve reasonable performance, deep learning models significantly outperformed them, particularly in terms of accuracy and generalization. Among these models, EfficientNet exhibited high accuracy and computational efficiency, showcasing suitability for resource-constrained environments. The use of CNNs demonstrated the effectiveness of deep convolutional architectures in capturing intricate patterns, while the GBN highlighted the utility of generative models for hierarchical feature learning.

CNN's training and validation outcomes affirm its capability to manage complex image classification effectively, achieving high accuracy with minimal overfitting. Yet, it is crucial to carefully manage the model's configuration and training length to overcome common issues like overfitting and ensure optimal generalization to unseen data.

EfficientNet's ability to generalize well, as demonstrated by the close alignment of training and validation metrics, underlines its suitability for complex image classification tasks. Nonetheless, careful management of training epochs and scaling parameters is crucial to maximize the model's potential without excessive computational costs or overfitting risks.

The primary limitation of the DBN in this task lies in its inability to handle the complex features of the characters as efficiently as CNN's. Additionally, while DBNs excel in unsupervised feature learning, the lack of advanced optimization techniques such as transfer learning and specialized convolutional layers hindered its performance.

These findings underscore the importance of selecting appropriate dimensionality reduction techniques and leveraging state-of-the-art deep learning architectures for challenging recognition tasks such as handwritten character identification in non-English scripts.

Future work could focus on enhancing recognition performance by experimenting with ensemble methods or hybrid architectures that combine the strengths of multiple models. Additionally, applying advanced techniques like transfer learning or fine-tuning pre-trained models on larger, domain-specific datasets may further improve accuracy.

Dataset misses a few characters from Devanagari script, especially vowels that help in making words. We need to work on searching and training models on full scripts to use them for multiple applications like translation and assistive technologies like as text-to-speech systems.

References

[1] M. Stamp, *Introduction to Machine Learning with Applications in Information Security*, 2nd ed. Boca Raton, Florida: CRC Press, 2023.

[2] Y. Gurav, P. Bhagat, R. Jadhav, and S. Sinha, ”Devanagari Handwritten Character Recognition using Convolutional Neural Networks,” in *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, Istanbul, Turkey, 2020, pp. 1-6, doi: 10.1109/ICECCE49384.2020.9179193.

[3] N. Aneja and S. Aneja, ”Transfer Learning using CNN for Handwritten Devanagari Character Recognition,” accepted for publication in *IEEE International Conference on Advances in Information Technology (ICAIT)*, 2019. Available: arxiv.org/abs/1909.08774.

[4] G. E. Hinton and R. R. Salakhutdinov, ”Reducing the Dimensionality of Data with Neural Networks,” *Science*, vol. 313, no. 5786, pp. 504-507, 2006, doi: 10.1126/science.1127647.

[5] 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)*, Kathmandu, Nepal, 2015, pp. 1-6, doi: 10.1109/SKIMA.2015.7400041.

[6] R. Dey, P. Gawade, R. Sigtia, S. Naikare, A. Gadre, and D. Chikmurge, ”A Comparative Study of Handwritten Devanagari Script Character Recognition Techniques,” in *2022 IEEE World Conference on Applied Intelligence and Computing (AIC)*, Sonbhadra, India, 2022, pp. 431-436, doi: 10.1109/AIC55036.2022.9848911.

[7] A. Krizhevsky, “One weird trick for parallelizing convolutional neural networks,” CoRR, vol. abs/1404.5997, 2014. [Online]. Available: http://arxiv.org/abs/1404.5997.