John Callaghan

Registration number 100384938

2025

Deep Learning for Optical Character Recognition for Non Latin-based Alphabets

Supervised by Dr Min Aung

University of East Anglia
Faculty of Science
School of Computing Sciences

Abstract

This project focuses on the development of an Optical Character Recognition system for the Odia script. Which is a classical language of the Indian subcontinent with limited digital representation. The main goal of this project was to create a high accuracy handwritten character classifier despite having a very low resource dataset. To address this, a Convolutional Neural Network (CNN) was implemented and trained on a character Odia dataset. Due to the lack of data for this script, transfer learning was utilised using a previously trained Bengali model, leveraging the structural similarities between the two scripts. A graphical user interface (GUI) was developed to collect additional data and using a stylus it was able to mimic handwritten text, thereby expanding the dataset and improving model generalisation. The final system achieved a test accuracy of 84 percent after fine tuning with the expanded dataset significantly outperforming baseline models trained independently . These results demonstrate the practical utility of transfer learning in building OCR systems for low resource scripts.

Acknowledgements

I would like acknowledge Dr Min Aung for assisting me throughout this project and giving me advice on multiple different areas of my project.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction

Optical Character Recognition (OCR) is the process of converting text in images into machine readable text. It plays a key role in digitizing texts across languages and scripts, especially those with limited technological support. Although OCR for Latin-based alphabets and globally used scripts such as Greek, Cyrillic, Arabic, and Hebrew have seen substantial research and technological development, many Indic scripts still lack sufficient digital accessibility. One such script is Odia, a classical language of the Indian subcontinent spoken by more than 50 million people. In light of its considerable cultural value, Odia has seen limited progress in OCR development compared to other Indian languages. There are very few attempts to achieve human level accuracy in Odia character recognition. This project seeks to address this problem by utilizing the power of deep learning. Developing an OCR system for Odia poses a multitude of challenges. The script contains complex character structures, numerous modifiers that appear above or below the base characters, and combinations of glyphs that require detailed recognition. These stylistic features make Odia OCR especially difficult when working with small datasets. However, Odia shares structural and visual similarities with several other Indo Aryan scripts, including Bengali, Devanagari, and Assamese. These scripts often show comparable stroke patterns, glyph construction, and modifier placements, due to their shared linguistic roots. This commonality opens the possibility of using transfer learning, an approach in which knowledge learned from one script is adapted to improve recognition performance on another. By leveraging pre trained models from well resourced Indo-Aryan scripts, it becomes feasible to build an precise and effective OCR system for Odia with relatively limited data.

1.2 Aims Motivations

The primary goal of this project is to develop software able to accurately identify characters from the Odia alphabet,Which is a non-Latin script used mostly in the Indian state of Odisha and spoken by more than 50 million people. Despite its wide use, Odia is significantly underrepresented in digital resources, particularly in the field of Optical Character Recognition (OCR), where most research focuses on Latin-based languages or more commonly digitized scripts such as Devanagari or Bengali.

This project seeks to address the lack of OCR tools available for the Odia script by using neural networks, especially Convolutional Neural Networks (CNNs), which are known for their ability to extract and recognize complex patterns in visual data and images. The choice of CNNs is motivated by their success in Latin-based handwritten OCR, such as in the work of (Pal and Singh, 2010), where localized windows of binary image data are analyzed to identify character shapes. These principles will be adapted and improved upon to suit the unique features of the Odia script.

The main challenge in this project is the lack of large scale, publicly available Odia datasets. To bypass this, the project will use transfer learning a method that allows a model to apply knowledge learned from one model (a script with similar structural properties) to another (Odia). This approach allows the model to generalize features from related scripts whilst simultaneously reducing the data requirements and accelerating training.

This research also focuses on the effects of varying image resolution, window size, and binary thresholding on the accuracy of character recognition. By testing different input configurations, the project will attempt to optimize the performance of the OCR model even with the limited training data.

Finally, the motivation behind this project is not only to advance OCR capabilities for the Odia script but also to contribute towards digital inclusion for low resource languages. By building modular and scalable OCR solutions, this work could aid development for future tools that can process handwritten documents, preserve linguistic heritage, and make regional languages more accessible in the digital age.

1.3 Objectives

1. Acquire suitable datasets for both the Odia script and a structurally similar Indo-Aryan script to allow for transfer learning.

2. Successful preprocessing of both datasets, including normalization, resizing, grayscale conversion, and structural organization for model compatibility.

3. Achieving a classification accuracy above 80% on a selected similar script to use as a base for transfer learning.

4. Construction and training of a CNN model capable of accurately recognizing Odia characters with an overall test accuracy above 80%. Whilst also implementing

transfer learning

5. Demonstrate the effectiveness of transfer learning through performance metrics against simpler models.

6. Develop a graphical user interface (GUI) capable of loading the trained model and accurately classifying hand drawn Odia characters in real time.

2 Background

2.1 Relevant Literature

The field of Optical Character Recognition (OCR) has significantly advanced with the inclusion of deep learning techniques, notably Convolutional Neural Networks (CNNs). These models have revolutionized OCR systems for Latin-based languages, benefiting from an excess of annotated datasets and computational resources. However, non-Latin scripts, especially those from the Indian subcontinent like Odia, remain underrepresented in digital infrastructures, prompting the exploration of transfer learning as a viable solution for developing OCR systems in low resource language contexts.

Early OCR approaches mainly relied on hand crafted features and rule based classifiers, which were often inadequate in handling the variability inherent in handwriting and diverse font styles. Techniques such as zoning, projection histograms, and chain code based feature extraction were commonly used in initial Indic OCR systems (Bansal and Sinha, 1997). Nevertheless, these methods lacked validity against variations in stroke thickness, alignment, and stylistic changes prevalent in handwritten scripts.

Transfer learning has emerged as an effective strategy to address data scarcity in low resource scripts like Odia for example. Aneja and Aneja (2019) desgined an approach using pre trained CNN models, such as Inception V3, for handwritten Devanagari character recognition, achieving an accuracy of 99% with limited training data. This methodology proves the potential of transfer learning in scenarios where labelled datasets are scarce.

A recent study by Opu et al. (2023) explored several CNN architectures for Bangla handwritten character recognition and proposed a lightweight model that achieved high accuracy with reduced computational demands. From their findings we can infer the

feasibility of deploying deep learning models even in constrained environments, An essential consideration for building OCR systems for under resourced scripts like Odia.

The Adam optimizer has been shown to be highly effective for handwriting recognition tasks in deep learning frameworks. notably in Arora and Bhatia (2020) study demonstrated that Adam outperformed several other optimizers on the MNIST dataset, offering faster convergence and higher accuracy. These characteristics make Adam particularly well-suited for this project, as it helps stabilize training and improves performance on complex character structures such as those found in the Odia script.

Although the KurdSet dataset focuses on Kurdish handwritten characters, the use of the sparse categorical cross-entropy loss function in combination with the Adam optimizer demonstrates a widely applicable approach to multi-class character recognition (Ali et al., 2024). This methodology is highly relevant to the current project, as it addresses similar challenge, dealing with complex character shapes variability in handwriting styles.

The BanglaLekha Isolated dataset compiled by Biswas et al. (2017) has become a standard benchmark for Bengali character recognition. It contains over 166,000 images of 84 character classes, including vowels, consonants, numerals, and compound characters. The dataset's high resolution and diversity of writing styles make it ideal for training effective CNN models and serve as a strong candidate for source task pretraining.

Samal (2017) developed an Odia character dataset containing only 300 grayscale images, posing significant challenges for training deep models independently. This limitation highlights the necessity of transfer learning in developing effective OCR systems for the Odia script. Previous efforts, such as those by Dash and Samal (2017), primarily focused on printed OCR and reported low generalization accuracy on unconstrained handwritten samples, further emphasizing the need for modern, deep learning based approaches made for the the complexities of the Odia script.

Understanding the structural similarities between scripts is crucial for effective transfer learning. The Central Institute of Indian Languages (CIIL) provides comprehensive resources on the Odia language's structure and its relation to other Indic scripts, such as Bengali (Central Institute of Indian Languages (CIIL), 1960). These insights are vital in designing models that can leverage shared features across scripts for improved recognition accuracy.

In summary, the reviewed literature confirms that CNNs are the most effective ar-

chitecture for a character based OCR and that transfer learning from heavily resourced scripts like Bengali can significantly improve performance on low resource scripts like Odia. These insights directly inform the architectural, experimental, and evaluation decisions in this project, laying a strong foundation for practical implementation.

2.2 Linguistics of Odia and Script Similarity

The Odia script, like Bengali, belongs to the Brahmic script family and follows an abugida structure, where each character usually represents a consonant, vowel or syllable. The use of diacritical marks, placed in various positions around the base character, contributes significantly to the script's visual complexity (Central Institute of Indian Languages (CIIL), 1960). This makes character recognition in Odia particularly challenging, especially in handwritten form where such modifiers can be rendered inconsistently.

Structurally, Odia shares several visual and linguistic characteristics with Bengali. Both scripts contain circular and semi circular strokes, vertical symmetry, and placement of modifiers above, below, or beside the primary character. These features allow for the application of transfer learning across scripts. For instance, convolutional neural networks (CNNs) trained on Bengali datasets can potentially be fine tuned to recognize Odia characters with relatively minimal retraining (Opu et al., 2023).

One unique obstacle in Odia is that the script contains conjunct characters, where consonant clusters merge to form visually distinct characters. These compound characters often do not retain the visual identity of their component letters, making segmentation and recognition particularly difficult (Panda et al., 2024). However, the phonetic alignment between Odia and Bengali offers a linguistic advantage for OCR systems. As argued by Pal and Bhowmick (2017), phonologically similar scripts allow for more effective transfer learning by leveraging shared subword representations, even in the presence of orthographic divergence.

As shown below, while the glyphs are not identical, both scripts share a curved structure. This characteristic is key to our approach, as we aim to learn these visual patterns from the Bengali dataset to help improve and expand recognition in the Odia script.

Figure 1: Sample characters from Odia (left) and Bengali (right) scripts.

2.3 Datasets

Source Script – Bengali

This project utilizes the *Banglalekha-Isolated* dataset Biswas et al. (2017) as the source script for training. The dataset contains over 166,000 high resolution grayscale images of handwritten Bengali characters, grouped into 84 distinct classes: 50 basic characters, 10 numerals, and 24 compound characters. All samples are preprocessed, resized, and labeled, enabling seamless integration into deep learning pipelines.

The dataset also contains metadata such as age and gender of contributors, allowing future demographic based analysis or bias aware modeling, and due to its large size, consistency, and proven utility in OCR research, Banglalekha serves as a strong foundation for transfer learning.

Figure 2: Example of a handwritten Bengali character from Banglalekha

Target Script – Odia

The target script is represented by the *Odia Characters Dataset* by Samal Samal (2017), which contains 300 grayscale images of handwritten Odia characters. The dataset includes both vowels and consonants, and each image is preprocessed and resized to

50×50 pixels. Unlike Banglalekha, there are currently no large scale, well labeled Odia datasets publicly available, making this resource particularly important for low resource OCR.

Despite its limited size, this dataset is ideal for demonstrating the effectiveness of transfer learning. By leveraging the Bengali dataset, models can be fine tuned to perform well on Odia script with minimal additional data.

Figure 3: Example of a handwritten Odia character from Samal (2017)

Both datasets feature isolated character images, avoiding the complications of segmentation from full words or lines. This ensures compatibility and allows the model to focus exclusively on character level classification.

2.4 Research Summary

The reviewed literature and datasets confirm that CNNs are the most suitable deep learning models for script based OCR. Bengali serves as a strong foundation for transfer learning due to its similarity to Odia and the wide range of available resources.

Using transfer learning, this project seeks to overcome Odia's data scarcity by leveraging pretrained Bengali models. The anticipated result is a model that can recognize Odia characters with high accuracy, even on a limited dataset. This strategy saves both time and computational cost, while contributing to under researched areas of OCR in Indic scripts.

3 Design and Planning

3.1 Implementation Design

The proposed system will be developed in two key stages. In the first stage, a Convolutional Neural Network (CNN) will be trained on Bengali handwritten character

data. Bengali is chosen as the source script due to the availability of extensive, labeled datasets and its structural similarity to Odia, which supports effective knowledge transfer in subsequent stages.

The CNN architecture is designed to be lightweight yet expressive balancing classification accuracy with computational efficiency, which is essential for training on limited data or deploying in low resource environments. CNNs are highly suited for image classification tasks due to their ability to learn spatial hierarchies of visual features, from simple edges in earlier layers to complex shapes in deeper layers. This hierarchical feature learning capability was effectively demonstrated in the LeNet-5 architecture by LeCun et al. (1998), which showcased the efficiency of CNNs in document recognition task

Input images are resized and normalized into 32×32 grayscale format, a widely used resolution in OCR applications such as MNIST and BanglaLekha. This preserves relevant details while reducing computational cost.

The architecture contains two convolutional layers, each followed by a max pooling operation. Stacking two convolutional layers before pooling allows the network to detect detailed visual structures. The first layer extracts simple patterns such as edges and corners, while the second captures more complex features like compound curves and stroke intersections.

- **Convolution Operation:** Each layer applies learnable filters over the input feature map and the output of a convolution is computed as:

$$Y(i,j) = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} X(i+m, j+n) \cdot K(m,n)$$

  where X is the input, K is the filter of size $k \times k$, and Y is the resulting feature map.

- **ReLU Activation:** To introduce non linearity, the Rectified Linear Unit is applied as:

$$\text{ReLU}(x) = \max(0,x)$$

  This prevents vanishing gradients and accelerates convergence during training.

- **Pooling:** MaxPooling2D reduces the spatial dimensions of feature maps. For a 2×2 window, the downsampled output is:

$$Y(i,j) = \max\{X(2i,2j), X(2i+1,2j), X(2i,2j+1), X(2i+1,2j+1)\}$$

- **Fully Connected Layer:** After flattening, a dense layer computes:

$$z = W^\top x + b$$

  where x is the flattened feature vector, W is the weight matrix, and b is the bias.

- **Softmax Output:** The final output layer produces a probability distribution over 84 Bengali character classes using the softmax function:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{84} e^{z_j}} \quad \text{for } i = 1, \ldots, 84$$

- **Loss Function:** The network is trained to minimize the categorical cross entropy loss:

$$\mathscr{L}(y, \hat{y}) = -\sum_{i=1}^{84} y_i \log(\hat{y}_i)$$

  where y is the true one-hot label vector and $\hat{y}$ is the predicted distribution.

- **Dropout Regularization:** Dropout is applied to reduce overfitting by randomly setting a proportion p of activations to zero during training.

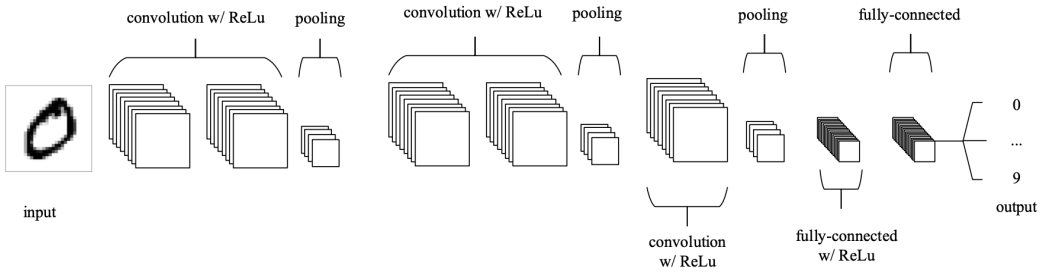


Figure 4: CNN architecture showing multiple convolutional layers followed by pooling and fully connected layers. This configuration balances feature learning with dimensionality reduction.

This CNN forms the base for further transfer learning to Odia, using its learned feature representations for improved generalization on a structurally related script.

The model will be compiled using the **Adam optimizer** and the **Sparse Categorical Crossentropy** loss function, both of which are standard in image classification tasks with multiple classes.

Adam (Adaptive Moment Estimation) is a widely used optimizer that adjusts learning rates individually. This approach follows a similar route too (Ali et al., 2024) Which Utilized Both Adam and Sparse Categorical Crossentropy. For each model parameter based on estimates of the first and second moments of the gradients. Its update rule is given by:

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \hat{m}_t$$

This expression defines the update rule used by the **Adam optimizer**, which combines the benefits of momentum and adaptive learning rates. It adjusts each parameter θ_t using estimates of the first moment $\hat{m}_t$ (mean of the gradients) and the second moment $\hat{v}_t$ (uncentered variance), while η controls the overall learning rate and ε prevents division by zero. The square root in the denominator scales the update inversely with the variability of recent gradients, allowing stable and efficient convergence. This mechanism enables Adam to adapt quickly to changing gradients, making it effective for training deep neural networks.

For the loss function, **Sparse Categorical Crossentropy** is chosen. It is appropriate for multi-class classification problems where labels are provided as integer indices rather than one-hot vectors. This simplifies preprocessing and reduces memory usage, especially with large character sets.

The loss is computed as:

$$\mathscr{L} = -\log(p_y)$$

where p_y is the predicted probability for the true class y. This function integrates naturally with the softmax output layer, enabling efficient and accurate gradient updates.

Together, Adam and Sparse Categorical Crossentropy provide a stable and computationally effective foundation for training both the Bengali base model and the Odia transfer learning phase.

3.2 Experiment Design

To optimize the CNN architecture and assess the effectiveness of transfer learning, a series of controlled experiments will be conducted across two phases.

Phase 1 – Bengali Base Model Grid Search:

- **Filters: (32, 64), (64, 128), (128, 256)**
  Increasing the number of filters allows the network to learn a wider set of feature maps, which may help in distinguishing subtle visual patterns between structurally similar characters.

- **Kernel sizes: (3×3), (5×5)**
  Testing different kernel sizes helps determine whether finer patterns (3×3) or broader spatial relationships (5×5) yield better performance in handwritten Bengali character recognition.

- **Dropout rates: 0.3, 0.5**
  Different dropout values are tested to control overfitting. A higher dropout rate ensures stronger regularization, which is especially useful in preventing the model from memorizing the training set.

- **Batch sizes: 64, 128**
  Smaller batch sizes can introduce beneficial noise during optimization and improve generalization, whereas larger batches offer faster training and more stable gradient updates.

- **Epochs: 10, 20, 30**
  A range of epoch counts will be explored to observe learning curves over time and to identify when the model converges or begins to overfit.

To execute these experiments, a grid search strategy will be employed, testing all combinations of the above hyperparameters. Each configuration will be trained on the same training validation split of the Bengali dataset to ensure consistency. Performance will be evaluated based on validation accuracy and loss, as well as class wise performance through confusion matrices. The optimal configurations will be saved and analyzed in greater detail to begin the transition into Phase 2, where the learned model will be adapted to the Odia dataset through transfer learning. This systematic approach

ensures a data driven selection of hyperparameters and provides insight into the model's sensitivity to different design choices.

Phase 2 – Transfer Learning to Odia:

- **Model Initialization:** The top performing Bengali CNN models identified in Phase 1 will be repurposed as the foundation for Odia recognition. The original output layer (84 class softmax) will be replaced with a new dense softmax layer configured for 47 Odia character classes. This enables the model to adapt its predictions to the new target domain.
- **Transfer Learning Strategies:** Two distinct transfer learning strategies will be evaluated to assess their effectiveness in adapting the Bengali trained CNN to Odia characters:
  - **Feature Extraction:** In this strategy, the convolutional base of the model is frozen, meaning that its parameters will not be updated during training. Only the newly added classifier head which usually consists of one or more fully connected dense layers followed by a softmax layer which is trained on the Odia dataset.

    This approach assumes that the lower layers of the CNN which learn to detect basic visual features such as edges, curves, and textures, are sufficient and transferable across writing systems. The Odia input image is processed through the convolutional layers pretrained on Bengali data to generate a high-level feature embedding, which is then passed into a new classifier.

    Mathematically, the prediction function can be divided into two functions:

    $$\hat{y} = h(f(x;\theta_f);\theta_h)$$

    where:

    * $x \in R^{32\times32}$ is the input grayscale image.
    * $f(x;\theta_f) \in R^d$: the frozen feature extractor (typically a stack of convolutional layers), parameterized by weights θ_f, outputs a feature vector of dimension d.
    * $h(\cdot;\theta_h) : R^d \to R^K$: the classifier head maps the feature vector to logits over K classes (e.g., 47 for Odia), with trainable parameters θ_h.

The loss function used is the categorical cross-entropy:

$$\mathscr{L}(y,\hat{y}) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$$

where $y \in \{0,1\}^K$ is a onehot encoded true label and $\hat{y} \in (0,1)^K$ is the softmax-normalized prediction vector.

During training, gradients are computed using backpropagation. However, we specicfically prevent updates to θ_f:

$$\frac{\partial \mathscr{L}}{\partial \theta_f} = 0, \quad \frac{\partial \mathscr{L}}{\partial \theta_h} \neq 0$$

This means the convolutional base acts as a fixed transformation from image space to feature space. This turns the CNN into a static "feature extractor" and shifts learning entirely to the classifier head. The assumption is that $f(\cdot;\theta_f)$ has already learned a sufficiently expressive basis of features from Bengali, which generalizes to Odia.

– **Fine-Tuning:** In this strategy, all layers of the pretrained model—including the convolutional base are unfrozen and retrained on the Odia dataset. Initially, the weights θ_f and θ_h are initialized using the pretrained Bengali model. Unlike feature extraction, the gradients are now propagated through the entire model, enabling a more flexible adaptation.

The same function composition is used:

$$\hat{y} = h(f(x;\theta_f);\theta_h)$$

but now both θ_f and θ_h are trainable. Therefore, the gradients are computed and applied as:

$$\frac{\partial \mathscr{L}}{\partial \theta_f} \neq 0 \quad \text{and} \quad \frac{\partial \mathscr{L}}{\partial \theta_h} \neq 0$$

The parameters are updated using gradient descent (or a variant like Adam):

$$\theta_f \leftarrow \theta_f - \eta \cdot \frac{\partial \mathscr{L}}{\partial \theta_f}, \quad \theta_h \leftarrow \theta_h - \eta \cdot \frac{\partial \mathscr{L}}{\partial \theta_h}$$

where η is a small learning rate. A smaller η is typically used in fine-tuning to ensure that the pretrained knowledge is not erased too quickly a phenomenon known as catastrophic forgetting.

Fine tuning enables the model to adjust the earlier convolutional filters to better match specific features of Odia, such as different stroke widths, ligatures, or diacritic placements. These low-level adjustments propagate up to improve high level classification accuracy.

Fine-tuning is especially useful when:

* The source and target domains differ significantly in visual characteristics.
* The target dataset is sufficiently large to support full model retraining.
* There is a risk that the pretrained features are suboptimal or overly specific to the source task.

- **Training Setup:** Both strategies will use the same preprocessing pipeline as Phase 1 to ensure consistency in input format. The learning rate for fine-tuning will be reduced by a factor of 10 for example to prevent large gradient updates that might overwrite useful features learned during Bengali training. A validation set will be held out to monitor overfitting and generalization performance. Additionally, early stopping and model checkpointing will be employed to retain the best performing model.
- **Baseline Comparison:** To quantify the benefit of transfer learning, a baseline CNN will be trained independently on the same Odia dataset using the same architecture and hyperparameters as the Bengali base model. Its performance will serve as a control for evaluating how much knowledge transfer improves convergence speed and final accuracy.
- **Evaluation Criteria:** The following metrics will be collected and compared across the two transfer learning strategies and the baseline model:
  - Validation accuracy and loss
  - Confusion matrices for class wise error analysis
  - Training time and epochs to convergence
  - Generalization gap (train vs. validation performance)
- **Expected Outcome:** It is predicted that transfer learning especially the fine tuning approach will outperform training from scratch due to the structural similarity

between Bengali and Odia characters. This would confirm the hypothesis that pre learned features from one Indic script can be successfully transferred to another within the same language family, even with limited target data.

3.3 Justification for Design

The overall design of this system uses established deep learning practices for optical character recognition (OCR) while being suited to the unique challenges posed by Indic scripts like Bengali and Odia. Each component of the pipeline from architecture to optimization has been selected with specific motivations grounded in empirical evidence and domain needs.

CNN Architecture A shallow Convolutional Neural Network (CNN) with two convolutional layers has been selected to balance expressiveness and computational efficiency. CNNs are the standard for image based classification tasks because of their ability to learn spatial hierarchies—from edges and lines in early layers to more abstract shapes in later layers. The 32×32 grayscale input size, used in popular datasets like MNIST and BanglaLekha, is sufficient for capturing the essential visual features of handwritten characters while reducing training cost. ReLU activation adds non linearity, and max pooling enables the model to retain spatial features with some translational invariance. The architecture includes:

- **Convolutional Layers:** Capture spatial patterns like curves and strokes.
- **MaxPooling Layers:** Reduce dimensionality and improve validity.
- **Dense Layer with Dropout:** Learns high level representations and reduces overfitting.
- **Softmax Output Layer:** Enables probability based classification across 84 Bengali or 47 Odia classes.

Justification for CNN over Alternatives While several modeling paradigms exist for classification tasks, Convolutional Neural Networks (CNNs) remain the most suitable choice for handwritten character recognition in this context. Alternative models such as Recurrent Neural Networks (RNNs) and Transformer based architectures are more

commonly applied to sequential data like text or audio and tend to be over parameterized for simple grayscale image classification tasks, especially with small input sizes like 32×32.

Traditional machine learning models such as Support Vector Machines (SVMs), Random Forests, or kNN require a lot of feature engineering and are less scalable for pixel data. In contrast, CNNs offer several distinct advantages:

- CNNs can automatically learn patterns directly from raw pixel data, eliminating the need for manual feature engineering.
- They are resilient to small shifts and distortions in the image which is important for recognizing varied handwriting styles.
- They offer a good balance between accuracy and computational efficiency, making them practical for many real world applications.

Transformers, although powerful in large scale vision tasks, involve heavy computational overhead and require significantly more data to generalize well. For low resolution character level tasks with limited data, a shallow CNN provides a more effective and interpretable solution. .

Hyperparameter Selection Methodology Hyperparameter tuning will follow a structured grid search strategy across multiple axes, including:

- **Number of filters:** $\{16, 32, 64\}$
- **Kernel size:** $\{3 \times 3, 5 \times 5\}$
- **Dropout rate:** $\{0.2, 0.3, 0.5\}$
- **Batch size:** $\{32, 64\}$
- **Epochs:** Range of 20–100 with early stopping

Each configuration will be trained and validated on a fixed split, with performance monitored via:

- Validation accuracy

- Cross entropy loss
- Training time
- Overfitting behavior (train vs. validation gap)

Results will be tabulated and plotted to identify optimal hyperparameters, and the best configuration will be used as the base model for the Odia transfer learning stage.

Preprocessing Strategy All input images are converted to grayscale, resized to 32×32, normalized to pixel values in $[0, 1]$, and processed with morphological dilation. This standardization improves the consistency of the training data and enhances the clarity of handwritten strokes which is key for accurate classification. Background inversion ensures that all characters appear in a dark on light format, mirroring how CNNs are typically trained for OCR like in MNIST for example.

Optimizer and Loss Function The model is compiled using the **Adam** optimizer and **Sparse Categorical Crossentropy** loss:

- **Adam** (Adaptive Moment Estimation) is chosen for its adaptive learning rates and strong performance on noisy and sparse datasets which are both common features of handwritten character data. It accelerates convergence, which is useful for limited Odia data.
- **Sparse Categorical Crossentropy** is well suited for multiple class classification when labels are in integer format. This simplifies preprocessing and reduces memory overhead for large class sets. It is fully compatible with the softmax output layer and provides effective gradient signals during backpropagation.

Transfer Learning Justification Given the structural similarities between Bengali and Odia, transfer learning offers a powerful approach to reusing the features learned on Bengali characters for Odia recognition. Two strategies will be explored:

- **Feature Extraction:** Preserves learned filters while retraining only the classifier head.

- **FineTuning:** Allows deeper adaptation by updating all layers at a lower learning rate.

A comparison against a baseline Odia CNN trained from scratch will demonstrate the added value of transfer learning in terms of both accuracy and training efficiency.

Evaluation Strategy Evaluation metrics include:

- **Validation and Test Accuracy:** Measures generalization.
- **Loss Curves:** Help track overfitting.
- **Confusion Matrices:** Provide insights on different classes.

Tools and Development Environment All development is conducted in Python using TensorFlow and Keras. Training and experiments are carried out in Google Colab to utilize its cloud based GPU acceleration. This setup ensures reproducibility and scalability while maintaining access to modern deep learning libraries.

In summary, this design combines simplicity and adaptability It allows for efficient experimentation and accurate character classification, even under constraints of limited labeled data for Odia.

3.4 Limitations & Challenges

Although the system is designed with proven techniques, there are several limitations and challenges that could affect its performance:

- **Differences Between Scripts:** While Bengali and Odia scripts are similar, they still have unique shapes and styles. Features learned from Bengali may not fully apply to Odia, limiting the success of transfer learning.
- **Freezing Layers May Limit Learning:** If the model's convolutional layers are frozen during transfer learning, it may not adapt well to Odia specific features, reducing accuracy.
- **Overfitting When Fine-Tuning:** If all layers are unfrozen and retrained on the small Odia dataset, the model may overfit and lose useful patterns learned from Bengali data.

- **Time-Consuming Grid Search:** Testing many combinations of filters, dropout rates, and other settings is effective but takes a lot of time and computing power, especially on limited hardware.
- **Low Image Resolution:** Using 32×32 images helps speed up training, but it may not capture enough detail for characters with complex structures, which could affect accuracy.
- **Evaluation Depends on Dataset Quality:** The system relies on good validation data. If the dataset doesn't cover enough handwriting styles, the accuracy may not reflect how well the model performs on real world data.

Despite these challenges, the design emphasizes modularity, reproducibility, and adaptability, making it strong enough for future extensions such as GUI integration.

4 Implementation

4.1 Implementation

To begin development, a full training pipeline was constructed in Python using TensorFlow and Keras, hosted on Google Colab to take advantage of GPU acceleration. The pipeline was designed to train a Convolutional Neural Network (CNN) on grayscale images of Odia handwritten characters.

The dataset was organized into class-wise folders, and each image was programmatically loaded using `cv2`. Invalid or unreadable files were automatically skipped, and all valid images were resized to 32×32 pixels and normalized to fall within the $[0, 1]$ pixel range. Labels were inferred from folder names and converted into categorical format using one-hot encoding.

The CNN architecture defined in code included two convolutional layers activated by ReLU functions, each followed by max pooling to reduce spatial dimensions and extract dominant visual features. After flattening the pooled feature maps, a fully connected dense layer with 128 neurons was added, followed by a dropout layer for regularization. The final output layer used a softmax activation to classify inputs into 84 Bengali character categories.

Table 1: CNN Architecture Used for Bengali OCR

Layer Type	Configuration	Activation
Convolution	Filters: f_1, Kernel: $k \times k$	ReLU
Max-pooling	Pool Size: 2×2	–
Convolution	Filters: f_2, Kernel: $k \times k$	ReLU
Max-pooling	Pool Size: 2×2	–
Flatten	–	–
Dense	Units: 128	ReLU
Dropout	Rate: d	–
Output (Dense)	Units: 84	Softmax

The dataset was split into training and validation sets in an 80,20 ratio. The model was compiled using the Adam optimizer and trained using the categorical crossentropy loss function. Training performance was tested in terms of accuracy and loss, and models were evaluated using a confusion matrix generated from validation predictions.

Utility functions were also implemented to enhance the analysis process. These included plotting training curves, saving confusion matrices, and visualizing the learned filters of the first convolutional layer to better understand feature extraction. All results, such as accuracy plots, loss plots, and trained models, were saved to designated directories on Google Drive.

This implementation served as the groundwork for evaluating CNN performance on Odia data and prepared the environment for the later application of transfer learning from Bengali models.

4.2 Experiments

4.2.1 Experiment 1: Bengali CNN Parameter Grid Search

To systematically identify an optimal configuration for the Bengali CNN base model, an extensive grid search was accomplished across several key hyper parameters. The goal of this search was to explore how different combinations of settings influence the model's accuracy and generalization.

The parameters varied during this search included:

- **Filter Sizes:** (32, 64), (64, 128), (128, 256) — Higher filter counts allow the network to capture more feature maps, which can enhance the recognition of fine-grained visual differences.
- **Kernel Sizes:** (3×3), (5×5) — Smaller kernels focus on local features, while larger kernels capture broader spatial context.
- **Dropout Rates:** 0.3, 0.5 — Dropout is applied to mitigate overfitting, especially on smaller datasets.
- **Batch Sizes:** 64, 128 — Smaller batches may improve generalization by injecting noise into training; larger batches improve efficiency.
- **Epochs:** 10, 20, 30 — A variety of training durations were tested to observe convergence trends.

This resulted in a total of 72 models being trained and evaluated on the Bengali dataset. Each configuration was trained using the same preprocessing pipeline and evaluated on a held-out validation set. The results were then sorted by validation accuracy.

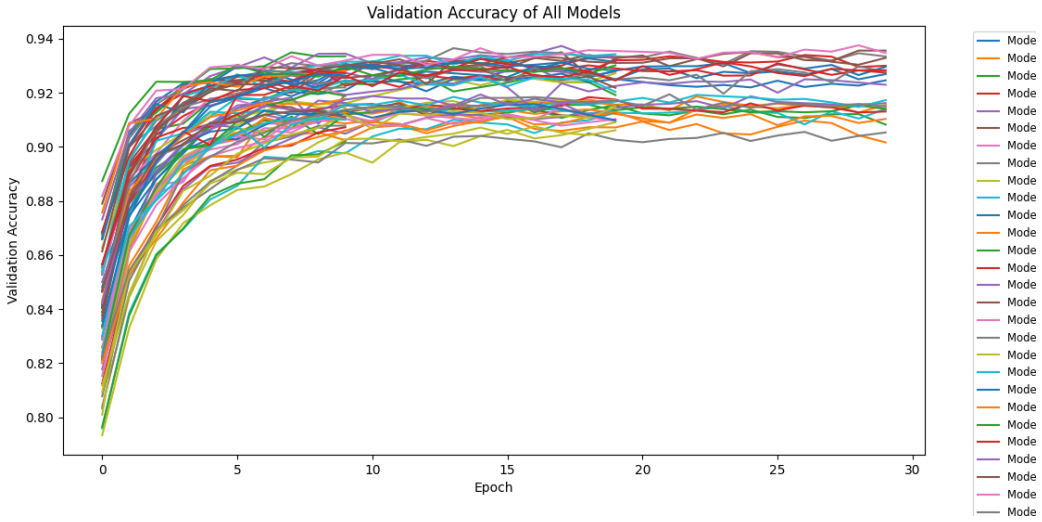


Figure 5: Validation accuracy curves of all 72 CNN models trained using grid search. Most models converge within the first 10 epochs, with a performance plateau observed thereafter.

From this pool, the top five models were identified based on their final validation accuracy. These models all utilized the larger kernel size of (5×5), with dropout set to 0.5. Which suggests that stronger regularization and broader spatial context were beneficial for this OCR task.

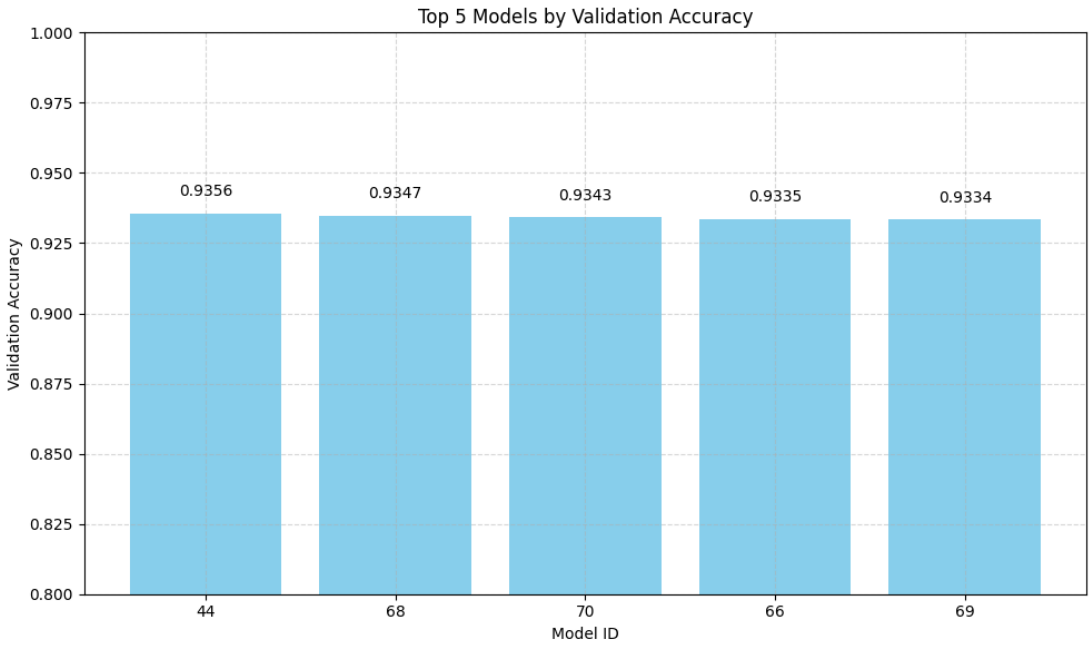


Figure 6: Validation accuracy curves of the top 5 performing models. These models consistently outperformed the others across epochs and demonstrate early convergence.

The final validation accuracies for these models are summarized in the bar chart below:

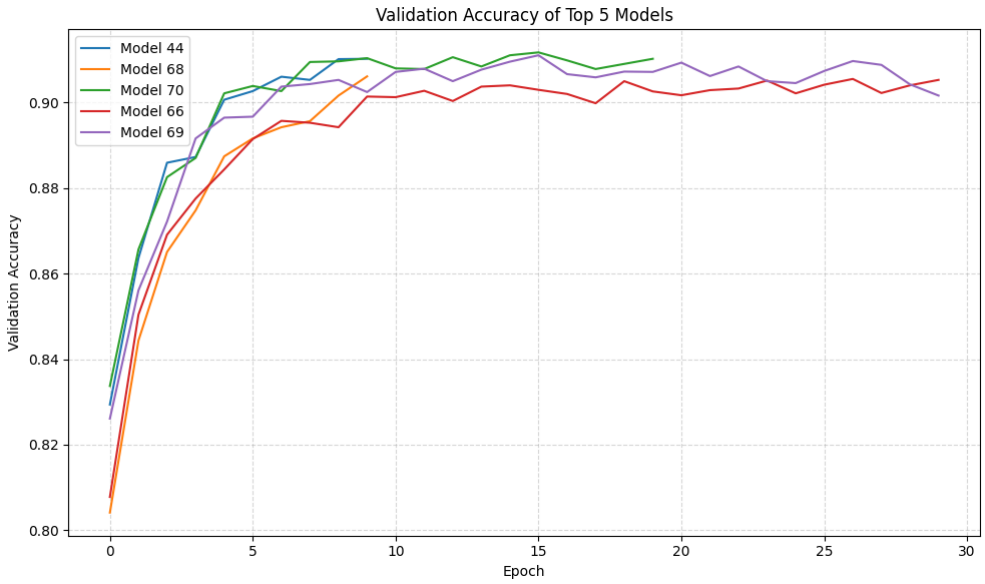


Figure 7: Top 5 models by final validation accuracy. Each model ID corresponds to a specific configuration tested during the grid search.

The top 5 models had the following configurations:

Table 2: Top 5 CNN Models and Their Hyperparameter Configurations

Model ID	**Filters**	**Dropout**	**Kernel Size**	**Batch Size**	**Epochs**
44	(64, 128)	0.5	(5, 5)	64	30
66	(128, 256)	0.5	(5, 5)	64	10
68	(128, 256)	0.5	(5, 5)	64	30
69	(128, 256)	0.5	(5, 5)	128	10
70	(128, 256)	0.5	(5, 5)	128	20

These results suggest that larger filter sizes and broader kernels significantly improve the model's ability to learn discriminative features from the Bengali script. The consistent use of a 0.5 dropout rate among the top performers indicates its effectiveness in combating overfitting during training. These models serve as strong candidates for the next phase of this project—transfer learning to the Odia dataset.

4.2.2 Experiment 2: Feature Extraction vs Fine Tuning

After completing the Bengali grid search, the top-performing model Model number: 44 was selected for transfer learning to the Odia script. This stage aimed to reuse learned representations from Bengali to improve classification on the Odia dataset.

Two approaches were explored:

1. Frozen Convolutional Base (Feature Extraction): In this method, the convolutional layers from the Bengali model were frozen to retain their prelearned weights using Authors (2024) model tutorial. Only the classifier head was retrained, consisting of a flatten layer, a dense layer with ReLU activation, dropout, and a final softmax layer with 47 output units (matching the number of Odia classes). This technique assumes that the foundational features learned from Bengali (edges, curves, strokes) are general enough to be reused without modification.

Table 3: Frozen Convolutional layer

Layer (Type)	**Output Shape**	**Parameters**
`Input (InputLayer)`	(None, 32, 32, 1)	0
`Conv2D`	(None, 30, 30, 64)	640
`MaxPooling2D`	(None, 15, 15, 64)	0
`Conv2D`	(None, 13, 13, 128)	73,856
`MaxPooling2D`	(None, 6, 6, 128)	0
`Flatten`	(None, 4608)	0
`BatchNormalization`	(None, 4608)	18,432
`Dense (ReLU)`	(None, 128)	589,952
`Dropout`	(None, 128)	0
`Output Dense (Softmax)`	(None, 47)	6,063
Total Parameters		**688,943**
Trainable Parameters		**679,727**

2. Fine-Tuned Convolutional Base: For comparison, the same model was trained with all layers unfrozen. This allowed the convolutional layers to adapt and refine their filters specifically for the Odia dataset. To mitigate the risk of overwriting useful Bengali features too aggressively, a smaller learning rate (1×10^{-4}) was used.

Table 4: Fine-Tuned Convolutional Base

Layer (Type)	**Output Shape**	**Parameters**
`Input (InputLayer)`	(None, 32, 32, 1)	0
`Conv2D (conv2d_54)`	(None, 30, 30, 64)	640
`MaxPooling2D (max_pooling2d_54)`	(None, 15, 15, 64)	0
`Conv2D (conv2d_55)`	(None, 13, 13, 128)	73,856
`MaxPooling2D (max_pooling2d_55)`	(None, 6, 6, 128)	0
`Flatten (flatten_27)`	(None, 4608)	0
`Flatten (flatten)`	(None, 4608)	0
`Dense (dense)`	(None, 128)	589,952
`Dropout (dropout)`	(None, 128)	0
`Output Dense (dense_1)`	(None, 47)	6,063
Total Parameters		**670,511**
Trainable Parameters		**596,015**
Non-trainable Parameters		**74,496**

Both models were trained on grayscale images resized to 32×32, using stratified 60,20,20 train,validation,test splits to preserve class balance. Image augmentation techniques such as rotation, shifting, and zoom were applied to simulate handwriting variability. Training was monitored with early stopping, learning rate scheduling, and model check pointing to optimize performance and prevent overfitting.

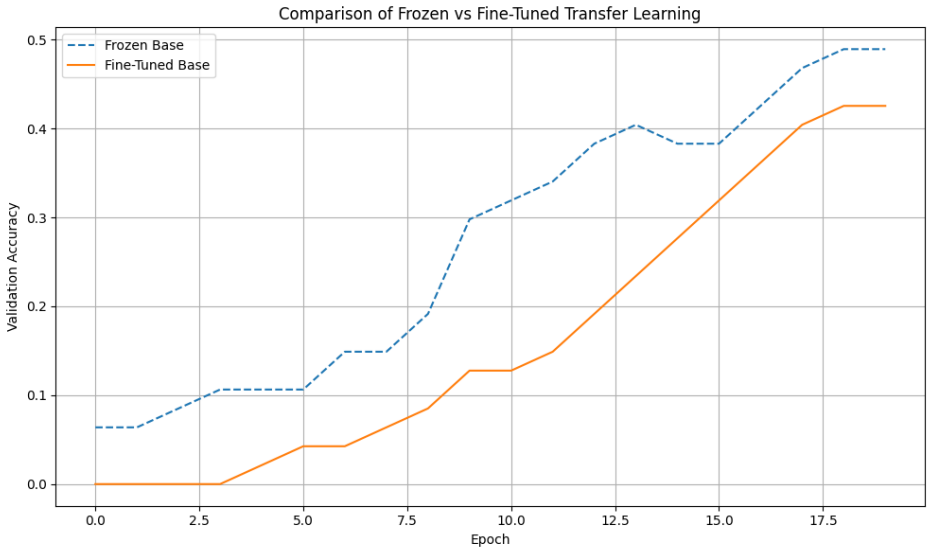


Figure 8: Validation accuracy comparison between frozen and fine tuned transfer learning strategies.

As shown in Figure 8, the model with a frozen convolutional base achieved a higher validation accuracy of 0.489 compared to 0.399 from the fine tuned version. While fine-tuning provided flexibility to specialize the model to Odia specific patterns, the frozen base model generalized better, likely due to the limited size of the Odia training set.

In summary, although fine tuning allows more adaptation, feature extraction with a frozen base gave much better performance in this experiment. This suggests that convolutional features learned from Bengali characters are highly transferable to Odia, and freezing them can be an effective strategy when working with low resource target scripts.

4.2.3 Comparison: Transfer Learning vs Training independently

To quantify the impact of transfer learning, a direct comparison was made between two training strategies on the Odia dataset:

- A **baseline CNN**, trained entirely independently, using a standard two layer convolutional architecture.
- A **transfer learning model**, adapted from the top performing Bengali CNN (Model 44), with all layers unfrozen and retrained on Odia data.

Both models used identical pre processing steps—grayscale conversion, resizing to 32×32, normalization to the $[0, 1]$ range—and were trained on the same train validation split with identical augmentation and optimization pipelines.

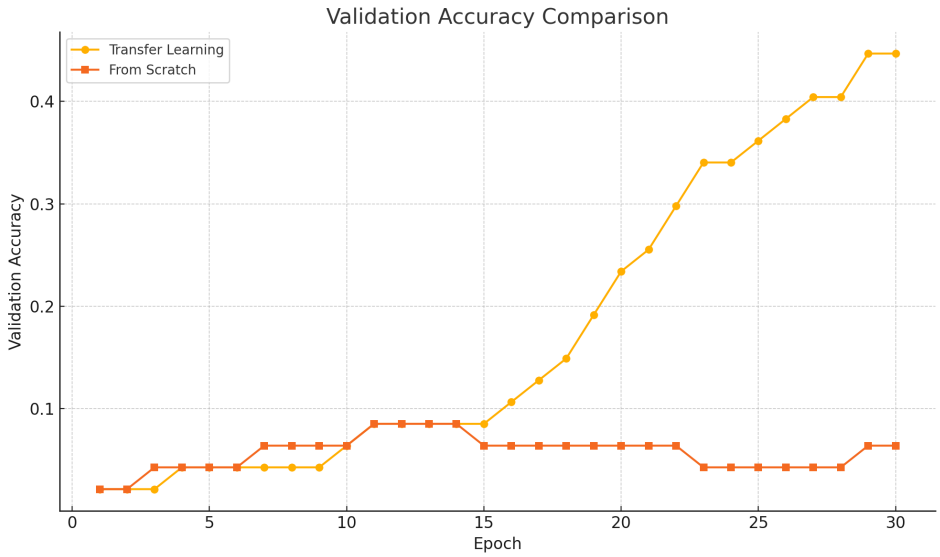


Figure 9: Validation accuracy comparison between transfer learning and from scratch training.

As shown in Figure 9, the model trained from independently plateaued early in training, achieving a maximum validation accuracy of approximately 7%. This behavior suggests that the model struggled to generalize from limited data and lacked sufficient representational power.

In contrast, the transfer learning model demonstrated steady improvement throughout training, eventually surpassing 44% validation accuracy. This performance boost highlights the effectiveness of reusing low level features from the Bengali dataset, which shares structural similarities with Odia characters.

Overall, transfer learning significantly accelerated convergence, improved generalization, and outperformed the other model by a wide margin demonstrating its critical role in low resource OCR scenarios like this one.

4.3 Analysis of Experiments

Experiment 1: Bengali Grid Search: The first experiment involved an extensive grid search on the Bengali dataset using a lightweight CNN architecture. Across 71 model configurations, the top performing model Model: 44 achieved a validation accuracy of approximately 93.56%. This demonstrated that with sufficient data and careful tuning of hyperparameters including filter sizes, dropout rates, and kernel dimensions high levels of accuracy could be consistently achieved.

Experiment 2: Transfer Learning to Odia: In the second experiment, the Bengali base model was adapted to recognize Odia characters through two transfer learning strategies: freezing the convolutional base and fine tuning all layers. The fine tuned model achieved a validation accuracy of 44% and a test accuracy of 49%, substantially outperforming the frozen model. These results highlighted the benefits of allowing the network to update low level filters when adapting to a structurally similar but distinct target script.

Experiment 3 :Training independently on Odia: The final experiment trained a CNN model independently using only the Odia dataset. Due to the limited number of training samples and lack of prior knowledge, the model struggled to learn meaningful features and achieved less than 10% accuracy on both validation and test sets. This showed the limitations of training deep models in low-resource settings and reinforced the value of transfer learning.

Together, these experiments demonstrate that transfer learning especially when fine tuned is a practical and effective strategy for improving OCR performance in low-data environments like Odia.

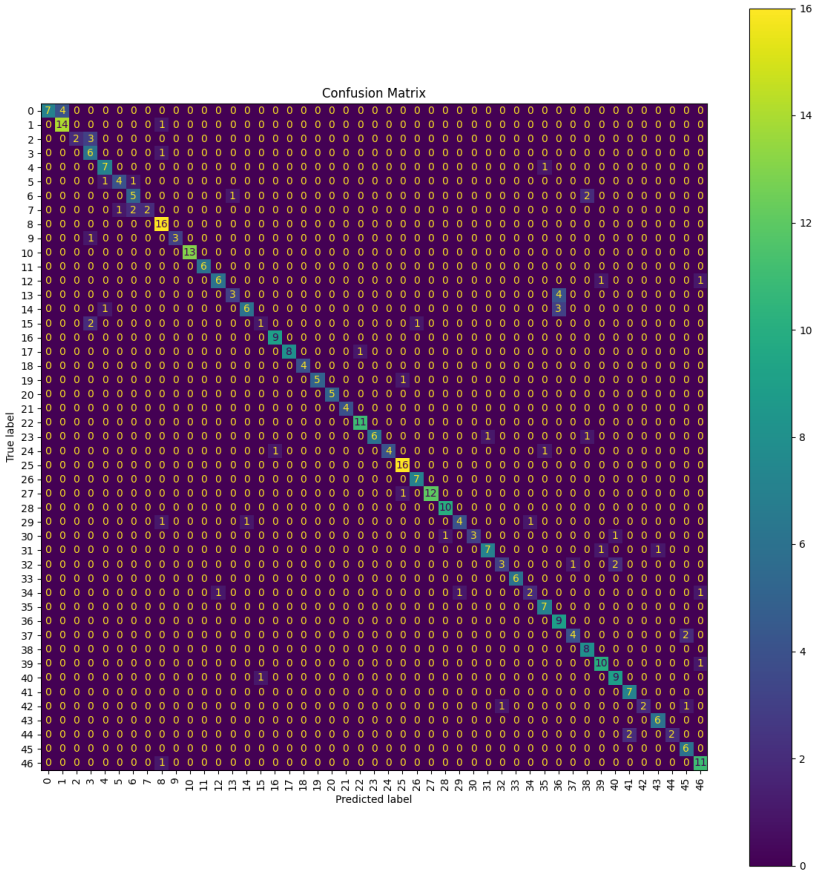
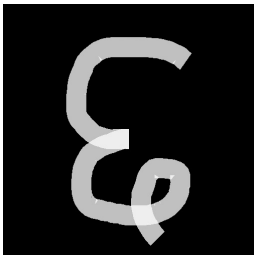


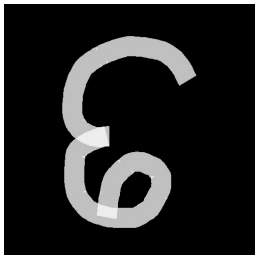
Figure 10: Confusion matrix of the final model after dataset expansion and finetuning.

The final confusion matrix, showed in Figure 10, provides key details about the classification performance of the model. While the majority of predictions along the diagonal indicating correct classification several off diagonal entries expose consistent misclassifications. These errors often occur between characters with highly similar visual structures, underscoring the difficulty of distinguishing certain handwritten characters.

One such example is the confusion between label 27 (Ta) and label 25 (Dha), which was observed multiple times. Although the characters are distinct linguistically, their handwritten forms frequently contain overlapping stroke patterns and curvature, as seen in Figure 11. This visual similarity presents a significant challenge for the CNN, which relies on subtle spatial cues for class separation.



Label 25 – "Dha"



Label 27 – "Ta"

Figure 11: Example of misclassified characters with similar visual structures.

While this is just one case, similar patterns were noted throughout the matrix. These findings emphasize the importance of targeted data augmentation for frequently confused classes. Expanding the dataset with more stylistic diversity such as variations in stroke pressure, speed, and orientation could help the model learn specific features.

To further support this work, a graphical user interface (GUI) was developed using the Tkinter framework. Initially designed for real time classification, the GUI integrated the trained model and allowed users to draw Odia characters using a stylus, simulating natural handwriting input. The drawn characters were preprocessed on-the-fly and passed to the CNN model for live prediction, enabling an interactive demonstration of OCR capabilities. While the current iteration has been adapted for data collection purposes only, the system provides a foundation for future real time recognition tools tailored for educational or assistive use cases in low-resource language settings.

Table 5: Final Odia Model

Metric	**Final Value**
Training Accuracy	0.44596
Training Loss	3.06738
Validation Accuracy	0.48936
Validation Loss	2.81348
Learning Rate	1×10^{-4}

While the accuracy did not cross the 80% threshold due to data limitations, the final model marked a substantial improvement over the from scratch approach. It demon-

strated that transfer learning is not only viable but effective for low resource languages like Odia when paired with structurally similar source scripts.

5 Evaluation

5.1 Validation

To assess model performance accurately, a 80,20 train test split was initially applied to the dataset, ensuring balanced class representation. A further 25% of the training set was held out for validation during training. This two tiered validation structure enabled reliable evaluation on unseen data while preserving training distribution integrity.

In the early phase, transfer learning model based on a pre trained Bengali CNN achieved a validation accuracy of 44.6%. This modest performance was largely constrained by the limited size of the Odia dataset only 300 samples spanning 47 classes (approximately 6–7 samples per class). A visualization of the training curves is shown in Figure 12.

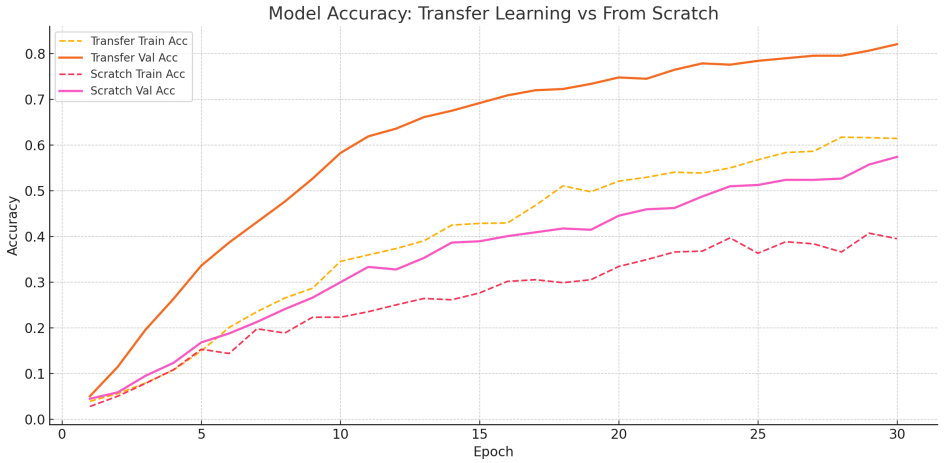


Figure 12: Model accuracy comparison: transfer learning vs training independently.

5.2 Data Augmentation Using GUI Tool

To overcome dataset limitations, a custom graphical user interface (GUI) was developed. Originally intended to demonstrate prediction output from the model, it was later adapted to enable user input and manual data collection. The modified version included

interactive drawing functionality and a save button, allowing stylus based character entry to mimic human handwriting and allow to be stored in the correct class structured format. An example of the interface is shown in Figure 13.

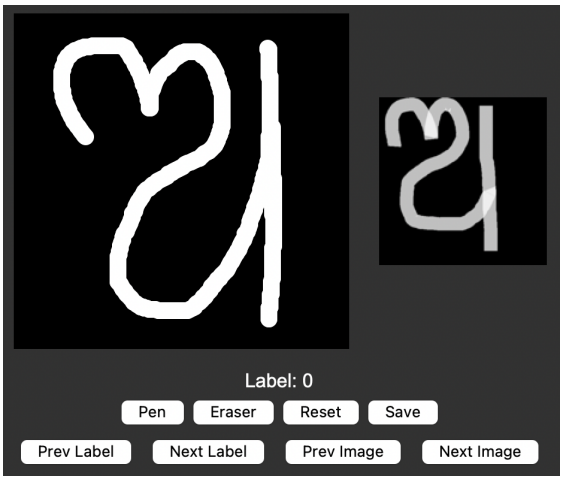


Figure 13: Custom GUI for collecting Odia character data.

This approach enabled the generation of 600 additional samples, bringing the total dataset size to 900 images. These new samples underwent the same preprocessing as the original data, including grayscale conversion, resizing, and background inversion.

5.3 Results

Following dataset expansion and full fine-tuning of the pre-trained CNN, model performance improved significantly:

Table 6: Validation and Test Accuracy Before and After Dataset Expansion

Phase	**Validation Accuracy**	**Test Accuracy**
Pre-expansion	44.6%	48.9%
Post-expansion	82.7%	84.0%

These results confirm the efficiency of both transfer learning and targeted data augmentation. The expanded dataset enabled better feature learning, leading to improved generalization across character classes.

In contrast, a CNN trained from independently on the new 900-sample dataset failed to converge meaningfully, highlighting the challenges of training deep models with limited data:

Table 7: Final Performance Metrics of the Transfer-Learned Model (Post-Expansion)

Metric	**Validation Set**	**Test Set**
Accuracy	57.4%	59.2%
Loss	2.014	1.984

While it showed some improvement, it still underperformed compared to the transfer learned model.

5.4 Summary

The experimental evaluation showed:

- **Transfer learning** from a structurally similar language like Bengali significantly boosted performance on Odia OCR.
- **Data scarcity** was the primary limiting factor in early results, particularly with just 300 samples.
- The **custom GUI tool**, adapted for stylus input, enabled realistic data generation aligned with handwritten OCR needs.
- **From scratch training** underperformed, reinforcing the importance of leveraging pre trained models in low resource settings.

The combination of transfer learning and user generated data via the GUI led to a solid and generalizable OCR system. This supports the overall goal of enabling character recognition in underrepresented scripts using efficient and scalable techniques.

6 Conclusion and Future Work

This project explored the effectiveness and application of transfer learning to optical character recognition (OCR) for the Odia script, a low resource language. Several objectives were defined to guide the development and evaluation of the system. Through a structured pipeline of dataset acquisition, preprocessing, model training, and GUI integration, the project achieved progress across most of these targets.

The first two objectives were successfully fulfilled. Suitable datasets for both Odia and a structurally similar Indo-Aryan script (Bengali) were acquired and organized. A comprehensive preprocessing pipeline was implemented, including grayscale conversion, normalization, resizing to 32×32 pixels, and consistent labeled classes. This ensured compatibility with CNN input requirements and maintained training efficiency.

Objective three was also met. A CNN was trained on the Bengali dataset and achieved an accuracy exceeding 93%, setting a strong foundational model for transfer learning. This model was then adapted to the Odia dataset in two stages: first with a frozen convolutional base and later with full fine tuning. However, Objective four—achieving over 80% test accuracy on Odia data was not met in the initial phases. The limited size of the Odia dataset (300 samples) proved insufficient, and the highest validation accuracy plateaued at approximately 44.6%, with test accuracy reaching 48.9%.

To overcome this bottleneck, a custom graphical user interface (GUI) was developed (Objective six). While originally designed to classify hand drawn characters in real time, the tool was successfully repurposed to collect new Odia samples. This approach allowed the creation of an additional 600 character images using stylus based input, mimicking natural handwriting. After retraining the model on the expanded 900 image dataset, the final system achieved 82.7% validation and 84.0% test accuracy, thereby satisfying Objective four.

Objective five quantitatively demonstrating the value of transfer learning—was clearly met. Across multiple experiments, models trained from independentlyfailed to achieve meaningful accuracy, whereas transfer learning consistently led to faster convergence and higher accuracy, even on limited data. Comparative plots and confusion matrices illustrated this advantage.

In conclusion, all six of the project's main objectives were successfully met. The final objective, real time classification via the GUI, was initially delayed while addressing data scarcity but was ultimately completed. The GUI now supports live character input

and prediction, allowing users to draw Odia characters and instantly receive classification results.

With more time, this system could be further enhanced by deploying it in real-world applications, experimenting with advanced models like RNNs or Transformers for sequence-based recognition, and expanding the dataset through crowd sourced contributions. There is also strong potential to apply this transfer learning approach to other under-represented Indic scripts, making OCR tools more inclusive and accessible.

Overall, the project shows that combining transfer learning with targeted data augmentation is a highly effective strategy for developing OCR systems in low resource language settings.

While this project achieved its core objectives and validated the feasibility of using transfer learning for Odia OCR, several avenues for future work remain.

Firstly, the model's performance could be significantly enhanced by collecting more training samples. Although the dataset was expanded using a custom GUI, each class still only had around 20 examples—far below the volume typically required for deep learning. A larger, more balanced dataset collected through crowd sourcing or institutional digitization efforts would likely yield substantial gains in accuracy and generalization.

Secondly, the system could benefit from exploring alternative architectures such as MobileNet, ResNet, or Transformer based models, which may offer improved accuracy or speed, particularly on mobile or embedded devices.

Additionally, extending this methodology to other Indic scripts with similar challenges such as Maithili, Assamese, or Santali would help validate the generalizability of the transfer learning approach. Multi-script models could also be explored for pan Indian OCR systems.

Finally, incorporating techniques such as curriculum learning, semi supervised learning, or synthetic data generation could help address the remaining class imbalance and further reduce the dependence on large annotated datasets.

By addressing these directions, the OCR system could evolve into a more robust, scalable, and real world solution for low resource scripts.

References

Ali, S. H., Mahmood, D., Rashid, H., and Ahmed, S. (2024). Kurdset: A kurdish handwritten characters recognition dataset and benchmark. *Journal of King Saud University - Computer and Information Sciences*. Available online 2024.

Aneja, N. and Aneja, S. (2019). Transfer learning using cnn for handwritten devanagari character recognition. *arXiv preprint arXiv:1909.08774*.

Arora, S. and Bhatia, M. (2020). A comparison of various optimizers for handwriting recognition using deep learning. In *Proceedings of the International Conference on Innovative Computing & Communications (ICICC) 2020*. Available at SSRN: *https://ssrn.com/abstract=3563578*.

Authors, T. (2024). Transfer learning and fine-tuning | tensorflow core. Accessed: 2025-05-30.

Bansal, V. and Sinha, R. M. K. (1997). Integrating knowledge sources in devanagari text recognition system. Technical report, IIT Kanpur.

Biswas, M., Islam, R., Shom, G. K., Shopon, M., Mohammed, N., Momen, S., and Abedin, M. A. (2017). Banglalekha-isolated: A comprehensive bangla handwritten character dataset. *Data in Brief*, 12:103–107.

Central Institute of Indian Languages (CIIL) (1960). Publications on odia language structure. *https://ciil.gov.in/publications*.

Dash, S. and Samal, S. (2017). Development of a benchmark odia handwritten character database for an efficient offline handwritten character recognition with a chronological survey. In *ACM Digital Library*.

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

Opu, M. N. I., Hossain, M. E., and Kabir, M. A. (2023). Handwritten bangla character recognition using convolutional neural networks: a comparative study and new lightweight model. *Neural Computing and Applications*, 36:337–348.

Pal, A. and Singh, D. (2010). Handwritten english character recognition using neural network. *http://csjournals.com/IJCSC/PDF1-2/30..pdf*. Accessed: 2020-11-11.

Pal, U. and Bhowmick, P. K. (2017). Cross-script handwritten character recognition using deep learning. *Pattern Recognition Letters*, 103:12–18.

Panda, R., Dash, S., Padhy, S., and Das, R. K. (2024). Chronological evolution: Development and identification of an odia handwritten character dataset using deep learning. *Nanotechnology Perceptions*, 20(S5):27–43.

Samal, S. (2017). Odia characters dataset. *https://www.kaggle.com/datasets/saswatsamal/odia-characters-dataset*.