BANGLA HANDWRITTEN CHARACTERS DETECTION USING CUSTOM LOSS FUNCTION

Md. Sakib Hossain Zisan
Faculty of Science and Technology
American International University-Bangladesh (AIUB)
408/1 (Old KA 66/1), Kuratoli, Khilkhet
Dhaka-1229, Bangladesh
sakibhossainzisan@gmail.com

Sumit Kanti Sarker
Faculty of Science and Technology
American International University-Bangladesh (AIUB)
408/1 (Old KA 66/1), Kuratoli, Khilkhet
Dhaka-1229, Bangladesh
sumitsarker141@gmail.com

Nesar Uddin
Faculty of Science and Technology
American International University-Bangladesh (AIUB)
408/1 (Old KA 66/1), Kuratoli, Khilkhet
Dhaka-1229, Bangladesh
nasar.uddin.56723@gmail.com

Abu Naser Md Abu Sadik
Faculty of Science and Technology
American International University-Bangladesh (AIUB)
408/1 (Old KA 66/1), Kuratoli, Khilkhet
Dhaka-1229, Bangladesh
abusadik88@gmail.com

DR. DEBAJYOTI KARMAKER
Associate Professor , Computer Science
American International University-Bangladesh (AIUB)
408/1 (Old KA 66/1), Kuratoli, Khilkhet
Dhaka-1229, Bangladesh
d.karmaker@aiub.edu

January 14, 2024

ABSTRACT

Bengali, a language with extensive global usage, possesses a distinctive and complex writing system that poses difficulties for systems designed to recognise handwritten text. Transfer learning has been applied to multiple Convolutional Neural Network models to effectively adapt to various Bangla handwritten datasets and acquire more advanced representations. In addition, we implemented a customised dynamic regularisation loss function that adaptively modifies the strength of regularisation during the training process. This approach fosters a refined equilibrium between the precision of the model and its ability to generalise.

1 Introduction

Handwritten character recognition is complicated and used in document analysis and language processing. Bangla, a complex script with many characters, requires accurate and efficient recognition systems. This article uses transfer learning and a custom loss function to improve Bangla handwritten character detection. Machine learning relies on transfer learning, which applies knowledge from one problem to another related problem. Transfer learning is essential to developing a strong model for detecting handwritten Bangla characters. We use pre-trained models for large-scale image classification to improve our character recognition system. Transfer learning is chosen because it can handle data shortages and computational constraints when training deep neural networks from scratch. Transfer learning lets us initialise our models with features from large datasets, speeding convergence and improving generalisation. Understanding Bangla characters requires a subtle understanding. This study examines advanced foundational models, including VGG16, VGG19, MobileNetV2, ResNet50, InceptionV3, DenseNet201, and Xception, pre-trained on large image datasets. These models extract hierarchical visual information as powerful feature extractors. Transfer learning allows us to adapt these models to Bangla characters' complexity and create a resilient handwritten character recognition system. We also use a custom loss function to optimise training and address any issues, taking into account our dataset's unique characteristics. This custom loss function enhances the model's response to subtle handwritten Bangla character variations, enabling more accurate predictions. We examine the experimental results of applying categorical cross entropy and our customised loss function to the foundational models in the following sections. The comparative analysis shows how transfer learning and our customised loss function affect recognition system performance. This study aims to advance Bangla handwritten character detection, which will benefit document analysis, language processing, and other fields.

2 Literature Review

Handwritten character recognition for a variety of languages, including Roman scripts, English, numerous European languages, and several Asian languages including Chinese, Korean, and Japanese, has been the subject of much research. Nonetheless, the Bangla language has seen comparatively little work done on it. Rahman et al.'s multistage approach is one of the significant study achievements about the handwritten Bangla characters. Bhowmik et al.'s MLP classifier is another intriguing piece of work. The upper portion of the character, the vertical line, and the double vertical line are the main characteristics of the multistage method. The human expert created alphabetic characteristics for the MLP classifier [3]. Recently, a different method for identifying English handwritten numerals that does not require feature engineering by a human expert was devised and is being used: the DBN [5]. In this case, the features are retrieved unsupervised—that is, without taking the labels into account—from the character image's pixel data. Convolutional neural networks are another method for extracting local characteristics [13]..[7]. It should be noted that learning distributed representation is the common goal of these techniques [11]. which led to a rebirth of neural network research. Nevertheless, it was discovered that several layered structures are incompatible with the gradient-based BP algorithm. Convolutional neural networks and DBNs avoid these problems while maintaining the original objective of learning distributed representation. We pointed out that the handwritten Bangla character identification task has not yet seen the application of such techniques. In order to assess the DBN's recognition performance on the Bangla handwritten character recognition challenge, we examine one of the previously suggested approaches in this study. While extensive study has been conducted on the recognition of handwritten English text, relatively little work has been done on the recognition of handwritten Bangla text. However, computer scientists in Bangladesh have carried out some noteworthy research. Rahman et al. [9] . developed a CNN-based system for handwritten character recognition in Bangla in 2015. In this case, the accuracy of the CNN approach was 85.96% over 50 classes. A bespoke dataset including 20,000 total samples and 400 sample photos per class was created. The experiment used a 28x28 image resolution. A deep learning approach was applied on Bangla numeric digits by Zahangir et al. [1] .. On the ten classes, this method achieved 98.8% accuracy. In this experiment, 6000 photos for Bangla numerals [10 numeric digit classes] from the CMATERdb3.1.1 dataset were used. In this experiment, the image resolution was 32x32. A research on offline handwritten numeral recognition was conducted by Bhattacharya et al. [6]. and the method employed in this instance was an ensemble of Multi Layer Perceptrons (MLPs) [2]..[12], which Adaboost integrated. This study compares the Multi-Layer Perceptron method of numeric digit recognition with handwritten Bangla, Oria, and Devnagari in detail. Chowdhury et al.[4] presented a study on the use of fuzzy logic for handwritten word identification in Bangla. Since the data was gathered as a time-ordered sequence of coordinates, no particular dataset was used. At different stages, the mouse-downs and mouse-ups were gathered in a sequential manner. This technique is incompatible with image data and only functions with mouse input.

Another online Bangla handwriting recognition system was developed by K. Roy et al. [10]. This method gathers online handwriting data via mouse and touch screen input; the primary goal of this work is to leverage sequential information from the pen's movements on the touch screen surfaces to train a quadratic classifier for recognition. 12500 Bangla characters and 2500 numeric Bangla digits were used to test the system. This approach achieved 91.13% accuracy on 50 classes of character data and 98.42% accuracy on 10 numeric data classes.

Bishwajit et al.[8]. used Deep Convolutional Neural Network to provide an offline or image-based system for recognizing handwritten characters in Bengali. This system was developed in 2017. Using 50 alphabets, this experiment's categorization accuracy was 91.23%. The Banglalekhaisolated dataset, which includes binary pictures of isolated Bangla alphabets, was the dataset used in this experiment. In this instance, a 28x28 image resolution was employed, and only 5% of the dataset was used for validation.

3 Methodology

Our methodology for Bangla handwritten character detection relies on a cornerstone in the field of deep learning which is the powerful technique of transfer learning .We combined that with state-of-the art base model and lastly and most importantly with the custom loss function we introduced .

We explored a variety of pre-trained models suitable for image classification tasks, including VGG16, VGG19, MobileNetV2, ResNet50, InceptionV3, DenseNet201, and Xception.The choice of these models is driven by their proven effectiveness in capturing complex visual features, which is crucial for recognizing the diverse and intricate characters present in Bangla script.

3.1 Layer Freezing

We utilize the layer freezing method to capitalize on the knowledge embedded in the pre-trained models. This involves preserving the weights of the initial layers of the base model while allowing subsequent layers to be fine-tuned for the specific task of Bangla handwritten character detection. By transferring generalizable characteristics from the pre-trained models to our particular recognition issue, layer freezing helps overcome the limitations caused by a lack of annotated Bangla character data.

3.2 Customization of Architecture

Additional layers are introduced to the base models to adapt them to the intricacies of Bangla characters. These added layers are kept minimal to avoid overfitting and to allow the model to capture relevant features without compromising computational efficiency.The input shape for the models is set to (128, 128, 3), aligning with the characteristics of our dataset.

3.3 CUSTOM LOSS FUNCTION

In this section, we present a novel approach to dynamic regularization within the context of training neural networks. The Dynamic Regularization Loss Function aims to adaptively adjust the regularization strength during the training process, fostering a balance between model precision and generalization.

In traditional neural network training, regularization is often static, set prior to training and remaining constant throughout the process. This approach may lead to suboptimal performance, especially when faced with dynamic or evolving datasets. Dynamic regularization strategies aim to address this limitation by modulating the regularization strength during training, allowing the model to adapt to changing patterns in the data.

3.4 Dynamic Regularization Loss Function

3.4.1 Background

In the realm of neural network training, the conventional practice involves using fixed regularization throughout the training process. However, this static approach may fall short when confronted with dynamic or evolving datasets. To address this limitation, our research introduces the Dynamic Regularization Loss Function—an innovative technique that dynamically adjusts regularization strength during training, promoting a nuanced balance between model precision and generalization.

3.4.2 Dynamic Regularization Callback

Central to our approach is the `Dynamic Regularization`, a custom component seamlessly integrated into the training process. This callback plays a pivotal role by dynamically modifying the regularization strength after each training epoch. The adjustment is governed by a linear decay function:

$$\text{Dynamic Strength}(t) = \text{Initial Strength} - \frac{t}{\text{Decay Epochs}} \times (\text{Initial Strength} - \text{Final Strength})$$

In this expression, t signifies the current epoch, while parameters like `initial_strength` and `final_strength` determine the range of the dynamic strength.

Custom Loss Function

The essence of our dynamic regularization is encapsulated in the custom loss function. This loss function combines both cross-entropy loss and dynamically adjusted L2 regularization. Mathematically, it can be expressed as:

$$\text{Total Loss}(y_{\text{true}}, y_{\text{pred}}, t) = \text{CrossEntropy}(y_{\text{true}}, y_{\text{pred}}) + \text{Dynamic Strength}(t) \times \text{L2 Regularization}$$

$$\text{Total Loss}(y_{\text{true}}, y_{\text{pred}}, t) = -\sum_i y_{\text{true},i} \cdot \log(y_{\text{pred},i}) + \text{Dynamic Strength}(t) \times \lambda \sum_i \sum_j w_{i,j}^2$$

Here:

- $\text{CrossEntropy}(y_{\text{true}}, y_{\text{pred}})$ signifies the cross-entropy loss between predicted and true distributions.
- $\text{Dynamic Strength}(t)$ is the dynamically adjusted regularization strength.
- λ represents the regularization strength parameter.
- $w_{i,j}$ denotes the weights of the model.

Our approach can be implemented across various frameworks using a dynamic strength variable. This variable is adjusted during training and integrated into the loss function.

The Dynamic Regularization Loss Function offers a versatile and adaptive framework for training neural networks. By dynamically adjusting regularization strength, our approach aims to fortify the model's capability to capture evolving patterns in the data. Comprehensive experimental assessments are warranted to evaluate the effectiveness of this dynamic regularization strategy across diverse datasets and model architectures.

Dynamic Regularization Algorithm Workflow

Step 1: Initialization

- Set the initial strength of regularization (`initial_strength`), the final strength (`final_strength`), and the number of decay epochs (`decay_epochs`).

Step 2: Training Loop

- Iterate through multiple training sessions (epochs).

Step 3: Adjust Strength Dynamically

- Calculate the dynamically adjusted regularization strength using a linear decay function:

  $$\text{Dynamic Strength}(t) = \text{Initial Strength} - \frac{t}{\text{Decay Epochs}} \times (\text{Initial Strength} - \text{Final Strength})$$

  where t is the current epoch.

Step 4: Forward Pass

- Perform a forward pass to obtain predictions from the model.

Step 5: Loss Computation

- Compute the cross-entropy loss, measuring the dissimilarity between predicted and true distributions.
- Calculate the L2 regularization term, penalizing large weights in the model.

Step 6: Total Loss Calculation

- Combine the cross-entropy loss and dynamically adjusted L2 regularization to form the total loss.

Step 7: Backward Pass and Updates

- Conduct a backward pass to compute gradients.
- Update the model parameters using an optimization algorithm.

Step 8: Convergence Evaluation

- Check convergence criteria to determine whether to continue training.

Step 9: Iterative Training

- Repeat the training loop until convergence.

Step 10: Conclusion and Evaluation

- Evaluate the trained model on a validation dataset.
- Analyze convergence, performance, and adaptability.

Step 11: Fine-Tuning

- Optionally fine-tune hyperparameters such as `initial_strength`, `final_strength`, and `decay_epochs` based on model performance.

Embarking on the journey of training a neural network is akin to navigating a complex map, aiming to unravel its intricate details for accurate predictions.

Starting Point (Cross-Entropy Loss): In this journey, we begin with a basic understanding, much like trying to predict locations accurately on the map. The cross-entropy loss acts as our guide, measuring how well our predictions align with the actual locations.

Avoiding Overthinking (L2 Regularization): As our understanding deepens, it's essential to avoid overthinking certain details and focus on the essential landmarks. L2 regularization serves as a guide, preventing the model from fixating too much on specific details and avoiding unnecessary complexity.

The Adaptive Path (Dynamic Regularization Strength): Picture adjusting your pace during the journey—starting with brisk steps and gradually slowing down to adapt to the terrain's complexity. The dynamic regularization strength mirrors this adaptive learning pace, allowing the model to be agile initially, capturing crucial patterns, and then gradually refining its understanding.

Balancing Act (Total Loss): Imagine walking a tightrope between precision and generalization. The total loss serves as our measure of balance, evaluating how well we navigate accuracy on the known parts of the map (training data) while staying flexible enough to handle new, unseen areas (validation data).

Our journey offers adaptability, a gradual refinement of understanding, and a delicate balance between accuracy and readiness for new challenges. The journey requires tuning, is sensitive to sudden changes, and involves some additional mental and computational effort for adaptability.

4 Advantages of the Dynamic Regularization Loss Function:

4.1 Adaptive Regularization:

The dynamic nature of the regularization strength allows the model to adapt its regularization during training. This adaptability is beneficial in scenarios where the optimal level of regularization may change as the model learns from the data.

4.2 Gradual Refinement:

The linear decay of the regularization strength facilitates a gradual refinement of the model. By starting with a higher regularization and gradually reducing it, the model is encouraged to focus on capturing essential patterns in the early stages and fine-tune its parameters later, potentially preventing overfitting.

4.3 Balancing Accuracy and Generalization:

The combination of cross-entropy loss and L2 regularization in a dynamic framework seeks to strike a delicate balance between accuracy on the training data and generalization to unseen data. This can contribute to the development of models that perform well on both training and validation datasets.

4.4 Fine-Tuning Control:

The ability to control the rate of regularization decay through the parameter ‘decay_epochs‘ provides practitioners with a fine-tuning mechanism. This flexibility allows for experimentation with different rates of regularization reduction to find the optimal balance for a specific task.

5 Limitations of the Dynamic Regularization Loss Function:

5.1 Dependency on Hyperparameters:

The effectiveness of the loss function depends on appropriately tuning hyperparameters such as ‘initial_strength‘, ‘final_strength‘, and ‘decay_epochs‘. Improper setting of these hyperparameters may lead to suboptimal performance or unintended effects on the model’s training dynamics.

5.2 Sensitivity to Data Characteristics:

The dynamic regularization may exhibit sensitivity to the characteristics of the dataset. In situations where the dataset has abrupt changes or noise, the linear decay of the regularization strength may not be the most suitable strategy, and other adaptive techniques may be more appropriate.

5.3 Increased Computational Cost:

The inclusion of dynamic regularization introduces additional computations during training, particularly in the calculation of the dynamic strength at the end of each epoch. While this cost might be negligible for smaller models, it could be a concern for larger and more complex architectures.

5.4 Not Universally Applicable:

The dynamic regularization strategy might not be universally applicable to all types of neural network architectures or learning tasks. Its efficacy may vary depending on the specifics of the problem at hand, and alternative regularization methods may be more suitable in certain scenarios.

5.5 Potential for Overshooting:

The linear decay of the regularization strength assumes a consistent and linear change in the complexity of the data over epochs. In cases where the data distribution undergoes abrupt changes, the linear decay may lead to overshooting, causing the model to adjust its regularization too quickly.

6 Conclusion:

The Dynamic Regularization Loss Function presents a promising approach to enhancing the adaptability of neural networks during training. While it offers several advantages in terms of adaptability and balance between accuracy and generalization, careful consideration of hyper-parameters and an understanding of its limitations are crucial for its successful application in various machine learning tasks. As with any novel technique, empirical testing and validation on specific datasets and models are essential to ascertain its effectiveness and uncover potential areas of improvement.

7 Used Cnn Architecturs

7.1 Resnet 50

ResNet-50 belongs to the ResNet family of convolutional neural network architectures. At the 2016 Conference on Computer Vision and Pattern Recognition (CVPR), Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun presented

"Deep Residual Learning for Image Recognition" to introduce ResNet, a Microsoft Research neural network architecture. ResNet relies on residual blocks, which contain skip connections (shortcuts) to learn residual functions. This helps solve the vanishing gradient problem and trains multi-layer complex networks. ResNet-50 has 50 layers—convolutional, pooling, fully connected, and skip connections. The architecture uses bottleneck blocks with 1x1, 3x3, and 1x1 convolutions to reduce computational cost while expressing complex features. Connections that bypass neural network layers or blocks, allowing information to flow between them. Skip connections, also known as shortcut connections or identity mappings, bypass one or more layers to allow direct gradient backpropagation without diminishing.Connections are added before residual block non-linearities. Global Average Pooling (GAP) calculates the average value of each neural network feature map across all spatial locations. Most ResNet-50 networks end with global average pooling rather than fully connected layers. This reduces parameters and aids generalisation. Batch normalisation before residual block activation functions aids deeper network training and generalisation. ResNet-50 and other ResNet variants were pre-trained on ImageNet.

7.2 Densenet 201

DenseNet-201 is a densely connected convolutional neural network. The proposed DenseNet-201 architecture increases the model's depth and expressiveness. The main innovation is dense connectivity patterns within dense blocks. DenseNet-201 has dense blocks like other DenseNet variants. Every layer in these blocks receives feature maps from previous layers and sends feature maps to subsequent layers. This high level of connectivity improves feature reuse and information flow. Each dense block has bottleneck layers to reduce input channels before 3x3 convolutions. This reduces computational cost without sacrificing expressiveness. The growth rate hyperparameter controls how many feature maps each layer in a dense block generates. DenseNet-201 generates more feature maps per dense block than DenseNet-121 due to its faster growth rate.Transition layers control feature map spatial dimensions in dense blocks. The model uses 1x1 convolutions and average pooling to reduce channel count and spatial resolution. Global Average Pooling (GAP) is a deep learning method that averages all feature map values to summarise its spatial information. Similar to ResNet, DenseNet-201 uses global average pooling instead of fully connected layers at the end. This reduces parameters and improves model generalisation. DenseNet architectures are known for parameter optimisation. Compared to skip-connection architectures, the network's high connectivity allows for more concise and expressive representations. DenseNet-201 is trained using data augmentation to improve its generalisation across input data types. Batch normalisation before dense block activation functions aids deeper network training and convergence.

7.3 Mobilenet v2

MobileNetV2, a convolutional neural network architecture, is optimised for mobile and edge devices. MobileNetV2 improves previous models. Innovative design balances model accuracy and computational efficiency. MobileNetV2 uses "inverted residual with linear bottlenecks." This block captures non-linearities with lightweight depthwise separable convolution and a linear bottleneck. The linear bottleneck causes depthwise separable convolution after a 1x1 convolutional layer without a non-linear activation function. Reduces computational costs while accurately representing data. MobileNetV2, like ResNet, uses skip connections to improve gradient flow during training. Unlike residual networks, MobileNetV2 uses different connection mechanisms. The inverted residual block uses skip connections, linear bottlenecks, and light depthwise separable convolutions. This design lets the network capture low- and high-level features. Bottlenecks reduce feature map, parameter, and computation dimensionality. Layers expand, increasing size or capacity. The expansion layer improves channel dimensionality before depthwise separable convolution in MobileNetV2. The model depicts traits better. Model size and computational resources are optimised by MobileNetV2. Mobile phones and low-resource edge devices process data instantly. MobileNetV2 width and resolution multipliers can be adjusted to balance model size and accuracy. Change deployment scale and impact factors.

7.4 VGG16

The Visual Geometry Group's VGG-16 convolutional neural network (CNN) is known for its simplicity and high performance in image classification tasks. Oxford's Visual Geometry Group created VGG-16, a 16-layer composite with different weights. Image size is usually 224x224 pixels for the VGG-16 architecture. The network is divided into convolutional blocks with two or more 3x3 convolutional layers and a ReLU activation function. These convolutional layers are followed by a 2x2 max-pooling layer to reduce spatial dimensions.The VGG-16 architecture's 13 convolutional layers enable hierarchical image feature acquisition. The network has three dense layers after the convolutional blocks. Fully connected layers use acquired features to reason sophisticatedly. ReLU activation functions introduce non-linearity after each convolutional layer across the network. In multi-class classification tasks, the last layer uses a softmax activation function to convert network output into class probabilities. Dropout randomly deactivates neurons during training. This method prevents overfitting and improves model generalisation. Batch normalisation standardises layer inputs, stabilising and convergent training. VGG-16's convolutional layers have many filters, allowing it to capture a variety of features at different scales.

7.5 VGG19

Visual Geometry Group 19 enhances VGG-16. The Oxford Visual Geometry Group made both models. CNN VGG-19 classifies images. There are many architectural similarities to its predecessor. The 19 weight layers—16 convolutional and 3 fully connected—and increased depth distinguish VGG-19. With consecutive 3x3 convolutional layers and max-pooling layers to reduce spatial dimensions, VGG-19 is simple and consistent like VGG-16. Regular 224x224-pixel images are needed for VGG-19. Two or more 3x3 ReLU-activated convolutional layers make up convolutional blocks. A 2x2 max-pooling layer after each convolutional block downsamples spatially.Block-based VGG-19 has 16 convolutional layers. Deeper than VGG-16, complex feature acquisition is possible. Three dense, fully connected layers use acquired features to perform sophisticated reasoning after convolutional blocks. ReLU activation functions after each convolutional layer create non-linearity, enabling intricate mapping. In multi-class classification tasks, the final layer converts network output into class probabilities using a softmax activation function. Dropout randomly deactivates neurons during training to prevent overfitting and improve model generalisation. Layer inputs are batch normalised to improve training stability and convergence. VGG-19 uses many filters in its convolutional layers like VGG-16. This lets the model capture diverse features at different scales. Compact 3x3 convolutional filters throughout VGG-19's architecture make it simple and consistent. VGG-19's deep structure simplifies hierarchical feature collection, making it a popular image classifier.

7.6 Inception V3

Significant educational accomplishment InceptionV3 improves the performance of convolutional neural networks (CNNs). Google InceptionV3 employs image classification and object detection techniques. The redesigned InceptionV3 enhances its performance. Inception modules utilise pooling and parallel convolutions of various sizes to capture intricate hierarchical features. Network performance and stability are enhanced by the implementation of batch normalisation and factored convolutions. InceptionV3 utilises classifiers during training to mitigate the issue of gradient vanishing. Network-end global average pooling reduces the number of parameters. InceptionV3, a powerful deep learning tool, exhibits versatility and high efficacy across various computer vision applications. The stem network of Inception v3 is composed of convolutional and max-pooling layers. The network utilises image analysis techniques to extract distinctive characteristics.Inception v3 is composed of modules. The modules consist of parallel convolutional filters and pooling operations with different sizes, such as 1x1, 3x3, and 5x5. The network acquires multiscale characteristics through parallel pathways.Inception v3 convolutions are simplified by factoring them, which helps to simplify complex convolutions. The parameters and computation are streamlined.Batch normalisation accelerates the training of post-convolutional layers.Blocks are formed by arranging stacks of Inception modules. The blocks symbolise intricate hierarchical data.Inception v3 employs convolutional and max-pooling reduction blocks in between its main blocks. Blocks restrict the available feature map space.Inception v3 utilises auxiliary classifiers to train the intermediate layers. Expert classifiers provide guidance for training on the issue of vanishing gradients.Inception v3 replaces fully connected layers with global average pooling in its final stage. Computing the average value of each feature map decreases the spatial dimensions to one per channel.Inception v3 utilises softmax activation to derive class probabilities for image classification.Inception v3 is well-suited for numerous computer vision tasks due to its ability to effectively capture intricate and diverse features across multiple levels. The model's accuracy and computational efficiency are optimised. Transfer learning and feature extraction employ widely-used architecture.

7.7 Xception

The groundbreaking CNN structure "Extreme Inception," or Xception, revolutionises deep learning for image classification and computer vision. Keras deep learning library creator François Chollet introduced Xception, a novel convolutional design. Use depthwise separable convolutions to split the conventional convolutional layer into pointwise and depthwise convolutions. This method captures complex patterns with less computational effort. Novel architecture lets Xception capture intricate features efficiently. The architecture's name, "exceptional" and "inception," reflects its deep learning innovation. Xception has helped researchers solve many computer vision problems. An important neural network architecture development.Xception's main improvement is depthwise separable convolutions, which split the conventional convolutional layer into depthwise and pointwise convolutions. Pointwise convolutions combine results using 1x1 convolutions, while depthwise convolutions use one filter per input channel. The division of this process reduces parameters and calculations, improving efficiency over conventional convolutions. Xception's entry flow extracts unique features from the input image using conventional convolutional and max-pooling layers. Residual modules with depthwise separable convolutions are in the intermediate stream. The network records local and global data interactions with these modules.The exit flow eases high-dimensional feature map-to-class prediction transition. The classification model uses depthwise separable convolutions, global average pooling, and softmax-activated fully connected layers. Xception bypasses convolutional blocks with skip connections like residual networks. These connections improve gradient propagation during network training, allowing deeper training. Batch normalisation stabilises and speeds training after convolutional layers.Xception ends with a softmax activation function and fully connected layer for multi-class classification.

8 Experimental Results

8.1 Results Using categorical cross-entropy

In the pursuit of making computers understand our handwriting as well as we do, we've put seven different brainy models—VGG16, VGG19, MobileNetV2, ResNet50, InceptionV3, DenseNet201, and Xception—to the test. Think of them as unique detects, each with its own set of tricks and skills, trying to crack the code of our handwritten messages. Our goal? To figure out which of these brainy models excels at recognizing handwritten characters accurately and quickly. We threw all sorts of handwriting styles their way—fancy loops, blocky print, even messy scribbles—to see how well they could untangle the characters.Here's the results we found:

Table 1: Transfer Learning with Categorical Cross Entropy - Training and Validation Accuracies and Losses

Base Model	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
VGG16	76.90%	79.96%	0.7482	0.6248
VGG19	72.63%	78.04%	0.8968	0.7483
MobileNetV2	39.27%	45.88%	2.1135	1.9358
InceptionV3	41.61%	55.17%	1.9889	1.5365
ResNet50	39.27%	45.88%	2.1135	1.9358
DenseNet201	78.98%	81.67%	0.6602	0.5690
Xception	71.54%	73.17%	0.9046	0.9365

8.2 Custom Loss Function with Dynamic Regularization

To further improve training, we introduced a custom loss function with dynamic L2 regularization for selected base models. This dynamic regularization was adjusted during training epochs to strike a balance between model complexity and generalization.

8.2.1 VGG16 with Custom Loss:

We implemented the custom loss function in the VGG16-based model and trained it for 200 epochs. The model attained a training accuracy of 65.07% and a validation accuracy of 71.62% following 108 epochs. The efficacy of the custom loss function in mitigating overfitting was proven.

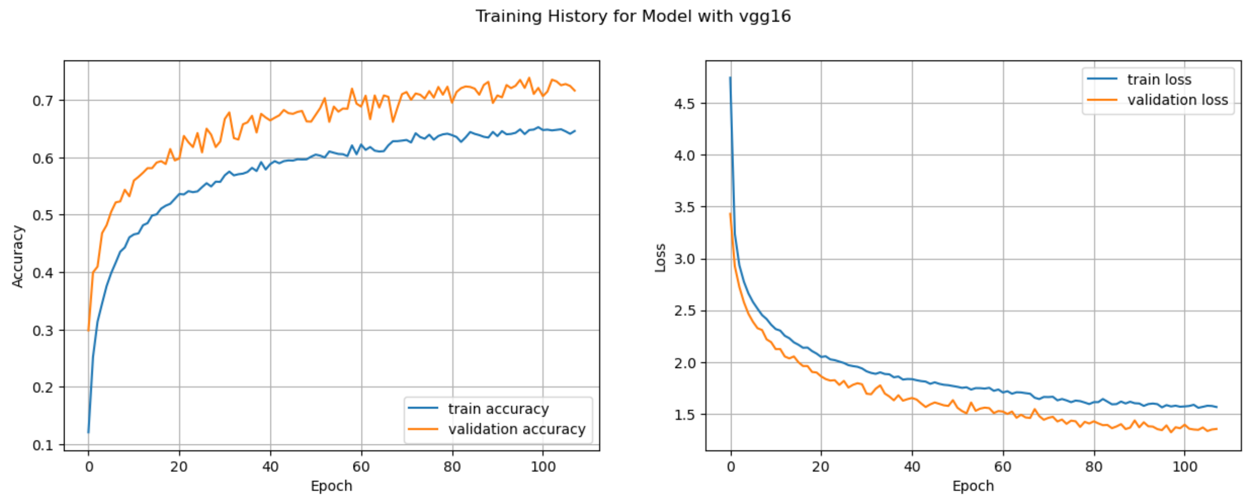

Figure 1: vgg16 model performance with Custom Loss function

8.2.2 MobileNetV2 with Custom Loss:

Similarly, the custom loss function was employed with the MobileNetV2-based model. After 34 epochs, the model achieved a training accuracy of 66.67% and a validation accuracy of 69.23%. The results indicated that the dynamic regularization contributed to improved generalization.

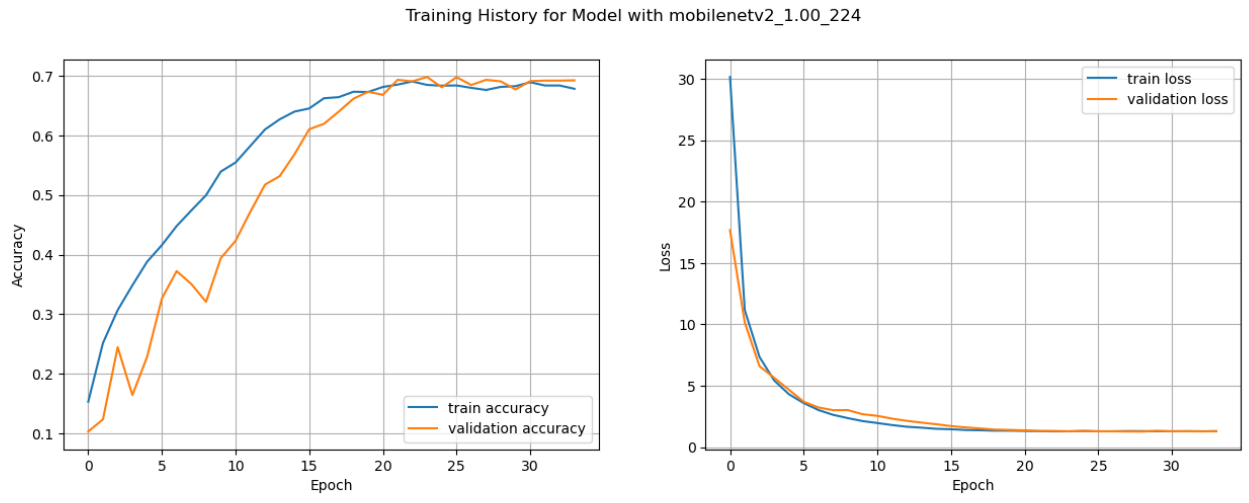

Figure 2: MobileNetV2 model performance with Custom Loss function

8.3 Comparison between categorical crosse-entropy and custom loss function

In order to evaluate the efficacy of the customized loss function, we conducted a performance comparison between the VGG16 and MobileNetV2 models trained using both categorical cross entropy and the customized loss function.

8.3.1 VGG16 Comparison:

The VGG16 model utilizing categorical cross entropy achieved a validation accuracy of 79.96%, whilst the model employing the custom loss function attained 71.62% accuracy. Despite the somewhat reduced accuracy, the utilization of the custom loss function showcased superior regulation of overfitting, as indicated by the prolonged training epochs.

8.3.2 Comparison of MobileNetV2:

The MobileNetV2 model attained a validation accuracy of 45.88% when trained using categorical cross entropy, but the use of a bespoke loss function resulted in an accuracy of 69.23%. The unique loss function demonstrated its capacity to improve generalization, resulting in a substantial enhancement in validation accuracy.

The custom loss function was expressly devised to mitigate overfitting, and its effectiveness is particularly notable in situations where overfitting is a prominent issue, as shown in the cases of VGG16 and MobileNetV2.

This comprehensive examination emphasizes the efficacy of the customized loss function, particularly in situations where the conventional categorical cross entropy may encounter difficulties with overfitting. The custom loss function's ability to lengthen training periods showcases its potential to improve both model generalization and performance
.

9 Conclusion and Future Work

Our approach to Bangla handwritten character detection utilises transfer learning, incorporating robust base models such as VGG16, MobileNetV2, and ResNet50. Our methodology achieves a balance between efficiency and capturing intricate features by freezing layers, adding minimal additional layers. Future efforts will concentrate on enhancing the customised loss function, examining dynamic regularisation strategies, and researching ensemble models to enhance performance. The data augmentation techniques will be optimised, and the practical usefulness will be evaluated through real-world deployment. This study establishes the foundation for enhancing precision and flexibility in the detection of Bangla characters.

References

[1] Md Zahangir Alom, Paheding Sidike, Tarek M Taha, and Vijayan K Asari. Handwritten bangla digit recognition using deep learning. *arXiv preprint arXiv:1705.02680*, 2017.

[2] Ujjwal Bhattacharya, Malayappan Shridhar, Swapan K Parui, PK Sen, and BB Chaudhuri. Offline recognition of handwritten bangla characters: an efficient two-stage approach. *Pattern Analysis and Applications*, 15:445–458, 2012.

[3] Tapan Bhowmik, Ujjwal Bhattacharya, and Swapan Parui. Recognition of bangla handwritten characters using an mlp classifier based on stroke features. pages 814–819, 11 2004.

[4] Kanchan Chowdhury, Lamia Alam, Shyla Sarmin, Safayet Arefin, and Mohammed Moshiul Hoque. A fuzzy features based online handwritten bangla word recognition framework. In *2015 18th International Conference on Computer and Information Technology (ICCIT)*, pages 484–489. IEEE, 2015.

[5] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[6] Tarun Jindal and Ujjwal Bhattacharya. Recognition of offline handwritten numerals using an ensemble of mlps combined by adaboost. In *Proceedings of the 4th International Workshop on Multilingual OCR*, pages 1–5, 2013.

[7] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*, pages 253–256. IEEE, 2010.

[8] Bishwajit Purkaystha, Tapos Datta, and Md Saiful Islam. Bengali handwritten character recognition using deep convolutional neural network. In *2017 20th International conference of computer and information technology (ICCIT)*, pages 1–5. IEEE, 2017.

[9] Md Mahbubar Rahman, MAH Akhand, Shahidul Islam, Pintu Chandra Shill, MH Rahman, et al. Bangla handwritten character recognition using convolutional neural network. *International Journal of Image, Graphics and Signal Processing*, 7(8):42–49, 2015.

[10] K Roy, N Sharma, T Pal, and U Pal. Online bangla handwriting recognition system. In *Advances In Pattern Recognition*, pages 117–122. World Scientific, 2007.

[11] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[12] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning internal representations by error propagation, 1985.

[13] Aiquan Yuan, Gang Bai, Po Yang, Yanni Guo, and Xinting Zhao. Handwritten english word recognition based on convolutional neural networks. In *2012 international conference on frontiers in handwriting recognition*, pages 207–212. IEEE, 2012.