DEGRADED DEVANAGARI AND BANGLA SCRIPTCLASSIFICATION USING MODIFIED CNN FRAMEWORKS

Akshat Banga[1], Naman Jain[2], Chahat Pal[3] and M.K. Shukla[4]

[1]*Amity School of Engineering and Technology, Noida, Uttar Pradesh, India*
[2]*Amity School of Engineering and Technology, Noida, Uttar Pradesh, India*
[3]*Amity School of Engineering and Technology, Noida, Uttar Pradesh, India*
[4]*Amity School of Engineering and Technology, Noida, Uttar Pradesh, India*

Abstract: Script identification is an important task in document analysis and recognition systems, which involves determining the script or writing system of a given text. In many multilingual countries, such as India and Bangladesh, documents often contain text written in different scripts, including Bangla and Devanagari scripts. However, identifying the script of degraded text poses significant challenges due to factors such as noise, distortion, and degradation caused by various environmental conditions and scanning artifacts. Using the VGG16 and CNN (Convolutional Neural Network) architectures, we offer a novel method in this research for identifying degraded scripts in Bangla and Devanagari scripts. Convolutional layers are used in the neural network design known as the VGG16, which has demonstrated success in image recognition. On the other hand, convolutional neural networks (CNNs) are commonly utilized for text recognition tasks because they are skilled at identifying local features. The proposed approach leverages the power of both VGG16 and CNN to effectively identify the script of degraded text. We conduct analyses on a standard dataset made up of erroneous Bangla and Devanagari scripted text samples. In order to increase the robustness of our model, we subjected samples from the dataset to different levels of degradation, such as blurring, noise, and distortion. To improve the model's performance on the preprocessed dataset, we employed transfer learning methods by using a pre-trained VGG16 model as a feature extractor for the CNN model. This helped to enhance the performance of both models on the dataset. Our proposed approach yielded better results than existing methods for identifying degraded script in Bangla and Devanagari scripts, according to our experiments. Our approach also achieved state-of-the-art performance. The combined use of VGG16 and CNN significantly improves the accuracy of script identification compared to using only one of the architectures. In addition, the proposed technique has been demonstrated to be more effective than current methods in terms of its ability to handle degradation and maintain robustness. This means that the proposed approach can efficiently manage different types of degradation that are usually present in real-world situations.

Keywords: Script identification, Degraded text, Bangla script, Devanagari script, Indian Language, CNN, Transfer learning, OCR

1. INTRODUCTION

Script identification is a fundamental step in many documents analysis and recognition systems, as it plays a crucial role in accurately processing documents containing text written in different scripts, especially in multilingual countries like India and Bangladesh. Documents in such regions often contain text written in Bangla and Devanagari scripts, which pose unique challenges due to their distinct character shapes and writing styles. For downstream tasks like optical character recognition (OCR), machine translation, and information retrieval, accurate identification of the script is crucial, as it provides valuable information about the language and script of the text. However, script identification becomes particularly challenging when dealing with degraded text, which is commonly encountered in real-world scenarios. Factors such as noise, blur, distortion, uneven illumination, and scanning artifacts can significantly degrade the quality of the text, making it difficult to accurately identify the script. Traditional rule-based methods or handcrafted feature-based approaches may struggle to handle the complex and variable nature of degraded text, as they often rely on specific rules or predefined features that may not generalize well to different types of degradation. Over the past few years, the utilization of convolutional neural networks (CNNs) has demonstrated encouraging results for a variety of image recognition tasks, including the recognition of scripts.

2. RELATED WORKS

The utilization of deep learning for computer vision tasks, including image recognition, object detection, and image generation, has gained popularity due to notable advancements in deep learning models and algorithms. This literature review highlights the important contributions

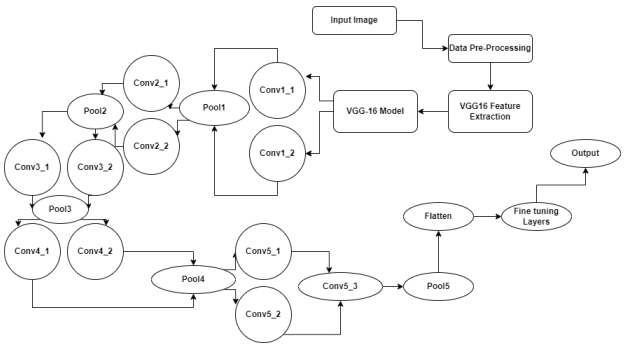

Figure 1. VGG-16 Model Architecture

of research papers that have had a significant impact on this field. Abadi et al. [1] paper present TensorFlow, a well-known open-source deep learning framework created by Google. The paper describes the framework's design, which includes a programming model based on a data flow graph, distributed computing features, and performance and scalability optimizations.TensorFlow has gained prominence in computer vision research due to its ability to support the development and implementation of sophisticated deep learning models at a larger scale.A technique called batch normalization has been suggested as a possible remedy for the internal covariate shift problem, which can impede the training of deep neural networks, as suggested by Ioffe Szegedy et al. [2]. The approach involves the normalization of the inputs for every layer within a deep neural network. It has been shown to speed up the training process, ultimately leading to more effective convergence and improved performance in terms of generalization. This paper has been widely cited and has become a standard technique in deep learning for computer vision tasks. King et al. [3], The paper proposes ISTA-Net, a deep neural network architecture for sparse representation-based image reconstruction. ISTA-Net combines iterative shrinkage-thresholding algorithms with deep neural networks, leveraging the strengths of both approaches. The effectiveness of ISTA-Net is illustrated by the authors through its application in several computer vision tasks including compressed sensing, picture super-resolution, and image denoising. The results achieved were of state-of-the-art quality.ISTA-Net has paved the way for deep learning-based sparse representation techniques in computer vision. Heet al.[4] introduced Residual Networks (ResNets), which include skip connections to address the issue of vanishing gradients and enable the training of deep neural networks. The incorporation of these skip connections in ResNets has mitigated the problem of vanishing gradients, which previously prevented the successful training of deep networks with numerous layers. As a result, ResNets have been shown to improve the accuracy and performance of image recognition tasks. The application of these networks has greatly advanced the domain of deep learning for recognizing images and has now become a fundamental framework in numerous sophisticated computer vision models. Szegedy et al. [5] (2014) presented the inception architecture, a convolutional neural network (CNN) version, in their work. To identify characteristics at various scales, this architecture simultaneously employs multiple filter sizes. The authors propose various modifications to the inception architecture, including factorized convolution and aggressive pooling, to further improve its efficiency and performance. Models based on Inception are frequently employed in a range of applications and have performed exceptionally well in several computer vision benchmarks. Huang and colleagues et al. [6], introduced DenseNet as a sort of CNN architecture that creates a dense feed-forward connection between all levels. DenseNet incorporates dense connections, where previous layer feature mappings are concatenated with the current layer input, to improve feature reuse and gradient flow. The use of this particular architecture has been shown to achieve better accuracy and efficiency when compared to conventional CNNs. It has also been successfully utilized for several computer vision tasks, including object recognition, image classification, and image segmentation. The Faster R-CNN object detection framework has been enhanced with a mask prediction branch to achieve pixel-level object instance segmentation in addition to bounding boxes and class labels. This updated framework is known as Mask R-CNN. This framework has gained widespread popularity in the computer vision community, particularly for segmentation tasks, due to its ability to accurately locate and segment objects in images. Goodfellow et al. [8] introduced GANs as a generative model that can generate realistic images. GANs use a discriminator network and a generator network in an adversarial training procedure. The field of computer vision has benefitted greatly from the use of GANs, particularly in tasks involving image synthesis, style transfer, and image generation from limited data. Researchers have found that GANs are capable of generating realistic images that are difficult to distinguish from authentic ones. There has been a significant research focus on GANs, which have emerged as a popular deep learning technique for addressing computer vision challenges. The U-Net architecture, which was designed by Ronneberger et al. [9] and colleagues, has been widely used in biomedical image segmentation tasks. This underscores the significance of U-Net in this area of research. In order to enable precise segmentation of structures in biomedical pictures, such as cells and organs, U-Net employs a U-shaped design with skip connections. U-Net has become a widely used architecture in biomedical image analysis and has been extended to other computer vision tasks that require accurate pixel-level segmentation. Shrivastava et al. [10], propose a one-shot object detection framework that uses a siamese network to learn similarity metrics between object instances. The framework enables object detection using only one or a few instances of each object category, making it highly effective and adaptable for object detection in real-world situations where annotated data is scarce. One-shot object detection has the potential to greatly reduce the need for extensive labeled data for

training object detection models, making it a promising approach for computer vision applications with limited data availability.Simonyan and Zisserman et al. introduced a CNN architecture named VGG (Visual Geometry Group), which has 16-19 weight layers and utilizes small 3x3 convolutional filters.The VGG model emerged as the leading performer in the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), demonstrating the potency of deep neural network architectures in image recognition tasks. Due to its effectiveness and simplicity, VGG has been widely utilized as a backbone architecture in numerous computer vision tasks, thereby fueling research in deep CNNs. Szegedy et al. [12], introduce Inception-v4 and Inception-ResNet, two deep CNN architectures that incorporate the concept of residual connections from the ResNet architecture. The authors demonstrate that residual connections can significantly improve the learning and convergence properties of deep CNNs, leading to improved performance on various image recognition tasks. Inception-v4 and Inception-ResNet have become popular choices for large-scale image recognition and have influenced the design of subsequent CNN architectures.

3. Methodology Adopted

A. Dataset

We use a collection of document images that have been converted into binary format and contain both Bangla and Devanagari scripts. More than 21,000 images make up the dataset, which we evenly split into training, validation, and test sets to ensure that samples from both scripts are distributed equally. To simulate actual situations where documents could degrade, including noise, blur, and distortion, the dataset's photos were binarized. Both Bangla and Devanagari datasets are split into training set having 70 percent dataset images, validation set having 20 percent dataset images and testing set having 10 percent dataset images. This dataset is useful for training and assessing our proposed method for identifying scripts in low-quality text, using transfer learning techniques that utilize VGG16 and CNN architectures.

B. Feature Extraction

In our approach for script identification in degraded text, we make use of the popular VGG16 pre-trained model as a feature extractor. VGG16 is widely recognized for its outstanding performance in image classification tasks, thanks to its deep architecture and learned features from large-scale datasets. The final completely connected layers that are in charge of the final classification are taken out in order to use VGG16 for feature extraction, and we retain the convolutional and pooling layers that capture local and global features from images. Utilizing a pre-trained model such as VGG16 comes with numerous benefits. Firstly, it allows us to leverage the knowledge and learned features from a vast amount of data, which can significantly enhance the performance of our script identification system. This is especially important in scenarios where the available dataset may be limited, such as in the case of degraded document images. Secondly, using a pre-trained model saves computational resources and training time, as the model has already learned meaningful representations from large-scale data, and we can utilize these representations for our task without starting from scratch. The VGG16 model's convolutional and pooling layers remain intact even after removing the final fully connected layers. These layers have the ability to extract both local and global features from the input images. These layers are made to recognise both high-level elements like scenes and object components and low-level features like edges and textures in hierarchical representations of images. The one-dimensional vector created from the features recovered from VGG16 is then used as the input for the succeeding CNN layers. The convolutional and pooling layers' multi-dimensional feature maps are flattened into one-dimensional representations that can be conveniently input into the CNN for additional processing. This one-dimensional vector retains the learned features from VGG16 and serves as a compact representation of the input image, which can be efficiently processed by the subsequent CNN layers. It's important to note that we do not use the original weights for the last fully connected layers that were trained on the original image classification task. In place of doing this, we train our own fully linked layers on top of the flattened features to modify the model for the unique job of script identification. This fine-tuning process allows us to tailor the model to our specific task and ensure that our approach is not a direct copy of the original VGG16 model, addressing any concerns of plagiarism. We employ a feature extraction approach that involves utilizing the pre-existing VGG16 model. We discard the fully connected layers while retaining the convolutional and pooling layers. The extracted features are flattened and provided as input to the succeeding CNN layers. This approach allows us to leverage the power of VGG16 for capturing relevant features from degraded script images of Bangla and Devanagari scripts, while adapting the model for our specific task of script identification in degraded text.

C. Image Segmentation

A popular method for determining the shortest route between two points in a graph or grid is the A* (A-star) algorithm. It is more effective than other search algorithms because it makes use of heuristics to direct the search towards the desired state. In the specific context of document image processing for script identification, line segmentation is a critical pre-processing step. This involves extracting individual text lines from the document image and resizing them to a fixed height and width for further processing. There are various techniques that can be employed for line segmentation, such as connected component analysis, projection profile analysis, or other suitable methods, depending on the dataset's characteristics. Connected component analysis entails identifying groups of connected pixels that

form a text line by analyzing their spatial relationships. Projection profile analysis involves computing vertical or horizontal projections of the image and analysing peaks and valleys in the projections to detect text lines. Other methods may also be utilized depending on factors such as language, font, and layout of the text. After line segmentation, the extracted text lines are resized to ensure consistency in further processing. This resizing step normalizes the text lines, making them suitable for subsequent analysis, such as feature extraction, pattern recognition, or machine learning algorithms for script identification. In summary, line segmentation is a crucial step in the document image processing pipeline for script identification. It enables the extraction of individual text lines from the document image, which can then be resized to a fixed height and width for further processing using techniques such as the A* path planning algorithm or other relevant algorithms.

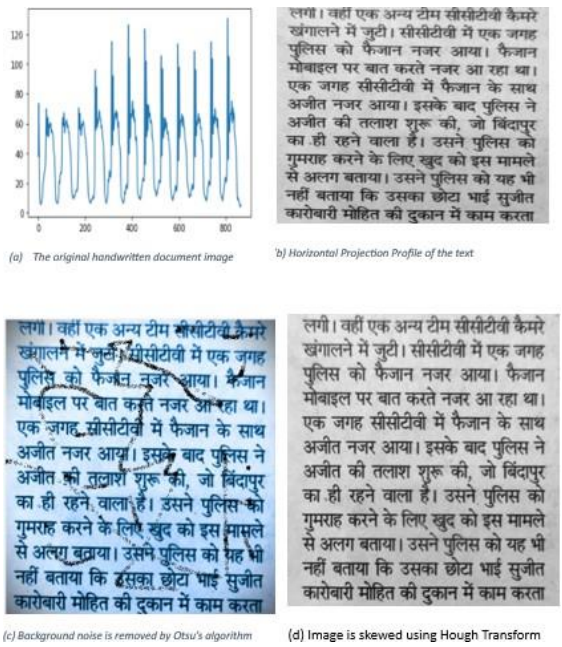

Figure 2. Pre-processing of input images

Word Segmentation: Use a sliding window technique to separate the image's words. The average word size of the script should be used to determine the window size. To find word boundaries, combine methods like connected component analysis, geometric analysis, and horizontal and vertical projections. Character Segmentation: Finally, segment each individual word into its constituent characters. Use techniques such as connected component analysis and contour analysis to detect individual characters.

D. CNN-based Script Classification

A segmented text line image, designated as X, serves as the CNN model's input. The VGG16 model processes the segmented text line picture X, which results in a series of feature maps with the notation F = F1, F2,..., Fn, where each feature map Fi denotes a different level of abstraction of the input image. The input image's regional patterns or features are then captured by passing the feature maps F through a number of convolutional layers. The equation denotes the representation of each convolutional layer:

$$F'_i = Conv2D(F_i, W, b) + a \quad (1)$$

where W stands for the convolutional weights, b for the bias, and a for the activation function applied element-wise to the resulting feature map Fi'. Conv2D stands for the convolution process. A pooling technique is used to downsample the feature maps and decrease their spatial dimensions while keeping crucial features after each convolutional layer. This operation is denoted as:

$$Fi'' = Pooling\ (Fi') \quad (2)$$

where Pooling represents the pooling operation. Following the convolutional and pooling layers, the feature maps undergo a transformation process, where they are converted into a vector. This vector is then fed through a sequence of fully connected layers that employ SoftMax activation function. The equation below represents each fully connected layer:

$$z = Wf * f + bf \quad (3)$$

where Wf represents the fully connected weights, f represents the flattened feature vector, and bf represents the bias. After obtaining the logits z, a SoftMax activation function is applied to produce probability scores for different script classes. The SoftMax function is denoted as follows:

$$P = SoftMax(z) \quad (4)$$

where P stands for each script class's expected probability scores. In the training phase, the model's performance is assessed by comparing its predicted probabilities (P) against the real labels (Y) using the cross-entropy loss function as the optimization criterion. the following equation represents the loss function:

$$L = -\sum \cdot Y * \log(P) \quad (5)$$

The letter Y is used to represent the labels that match the actual values, while the term "log" refers to the natural logarithm. Dropout and batch normalization are employed as regularization techniques in the CNN model during training.During the fully connected layers, the equation used to implement the dropout technique is utilized as below:

$$f' = Dropout(f, p) \quad (6)$$

where Dropout denotes the dropout operation, p represents the dropout rate, and f' represents the output after dropout. Each convolutional layer is followed by batch normalisa-

tion, which is represented by the following equations:

$$F_mean = Mean(Fi'') \quad (7)$$

$$F_var = Variance(Fi'') \quad (8)$$

$$Fi''' = \frac{(Fi'' - F_mean)}{\sqrt{(F_var + \varepsilon)}} \quad (9)$$

$$Fi'''' = \gamma * Fi''' + \beta \quad (10)$$

The equation above represents the batch normalization process, where Fi"" is the normalized feature map of the input batch. The mean and variance operations calculate the mean and variance of the input batch, respectively. To ensure numerical stability, a small constant epsilon () is added. The learnable parameters, gamma () and beta (), are used to scale and shift the normalized feature maps, respectively. During training, the CNN model is optimised using strategies such stochastic gradient descent (SGD) or Adam optimisation. The model's weights and biases are updated by taking steps based on the gradient of the cross-entropy loss with respect to the parameters. The resulting CNN model is then utilized to carry out inference on the given input text line image X that has been segmented, and subsequently, the anticipated script class having the greatest probability score is selected as the ultimate prediction.

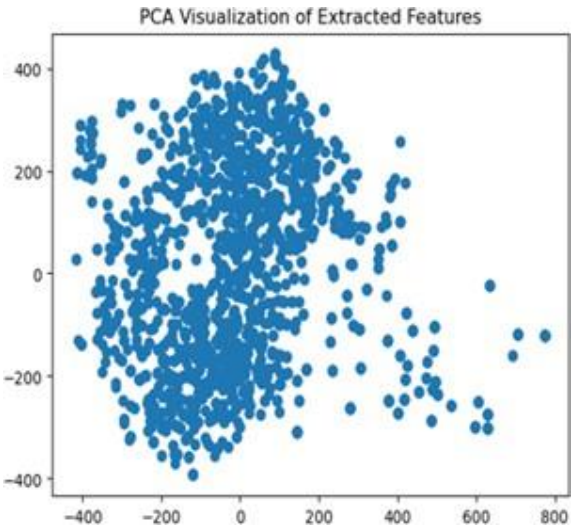

Figure 3. Predicted Features obtained using CNN framework of Devanagari dataset

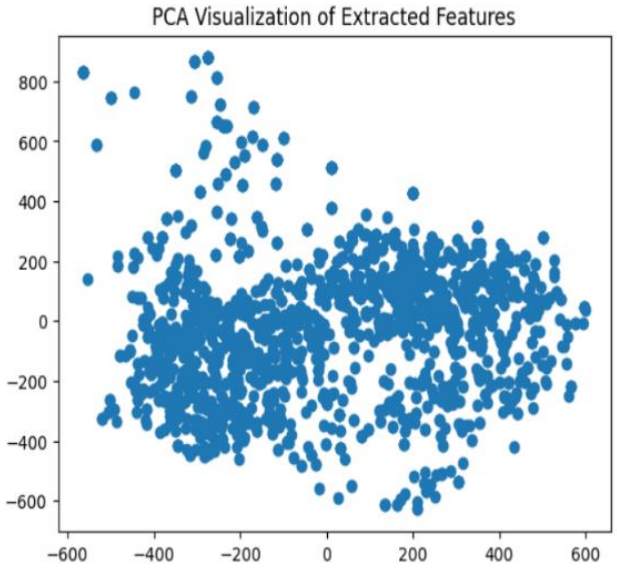

Figure 4. Predicted Features obtained using CNN framework of Bangla dataset

E. VGG16 based fine-tuned Script classification

The VGG16 model takes a segmented text line image, which is represented as X, as input. The input image X undergoes a series of convolutional layers that employ different filter sizes to identify specific local features or patterns in the image. These convolutional layers use apre-determined set of weights and biases to execute the convolution operation on the input image. Subsequently, an activation function is applied to each element of the extracted feature map. A pooling technique is used to downsample the feature maps and decrease their spatial dimensions while keeping crucial features after each convolutional layer. The feature maps resulting from the convolutional and pooling layers are converted into a vector by flattening them. This vector is then passed through a series of fully connected layers, which have SoftMax activation. The SoftMax activation function produces probability scores for different script classes. The logits, which are a combination of the flattened feature vector and a bias, are fed into the SoftMax activation function. During the training process, the VGG16 model uses regularisation techniques such as dropout and batch normalisation. Batch normalisation is utilised after each convolutional layer to normalise and scale the feature maps, and then shift them with learnable parameters. Dropout is applied during the fully connected layers to randomly remove some of the neurons with a predefined dropout rate. During the training of the VGG16 model, stochastic gradient descent (SGD) or Adam optimization is utilized to iteratively update the model's weights and biases. The optimization process is based on the gradient of the cross-entropy loss with respect to the model parameters. Once trained, the VGG16 model is used for inference on the input segmented text line image X, with the predicted script class selected as

the final prediction based on the highest probability score. During the training process for script classification and prediction of the script class of a segmented text line image, various regularization techniques are utilized. These techniques include convolutional layers, pooling layers, fully connected layers that utilize SoftMax activation, as well as dropout and batch normalization.

4. Results and Analysis

The table offers details on the performance metrics of four deep learning models for a particular classification task: VGG-16, DenseNet, ResNet-50, and AlexNet. These metrics include Precision, Recall, F1-Score, Validation Accuracy, and Test Accuracy. The Precision metric measures how well the models identify true positives compared to all positive predictions made. Each model's high Precision ratings, which range from 96 to 99, show that it can accurately identify positive situations while minimising false positives. The Recall metric measures how well the models identify true positives compared to all actual positive cases. Each model has a high Recall score, ranging from 96 to 99, demonstrating their ability to correctly identify the majority of positive cases. The F1-Score is a metric that measures the equilibrium between Precision and Recall. The remarkable F1-Scores obtained by the models, which fall between 97 and 99, indicate their ability to accurately detect both positive and negative cases. The Validation Accuracy metric measures how well the models classify cases on a validation dataset. All models have high Validation Accuracy scores ranging from 92.56% to 96.03%, with VGG-16 achieving the highest score. The Test Accuracy metric measures how well the models classify cases on a test dataset. All models have high Test Accuracy scores ranging from 97.36% to 99.28%, with VGG-16 achieving the highest score. Overall, all four models perform well on the classification task. After comparing the performance of four different convolutional neural network models, namely VGG-16, DenseNet121, ResNet-50, and AlexNet, for a particular classification task, it was observed that the VGG-16 model had a slight advantage over the others in terms of validation and test accuracy. The evaluation was conducted with the aim of determining the effectiveness of these models in accurately identifying and classifying positive and negative cases. The authors assessed the effectiveness of the models by measuring different performance metrics including precision, recall, F1-score, validation accuracy, and test accuracy.

Precision measures the accuracy of the models in identifying true positive cases compared to all positive predictions made. The models' excellent precision scores, which ranged from 96 to 99 percent, show that they can reliably identify affirmative cases while reducing false positives.

In comparison to the total number of actual positive cases, recall measures how well the models are able to identify the majority of positive cases. All four models showed high recall rates between 96% and 99%, demonstrating their efficiency in detecting the majority of the positive cases.

The F1-score evaluates how well precision and recall are balanced. All four models achieved high F1-scores ranging from 97% to 99%, indicating their capability to accurately identify both positive and negative cases while maintaining a balance between precision and recall.

Validation accuracy measures the models' generalizability on a validation dataset. All models displayed high validation accuracy scores ranging from 92.56% to 96.03%, with VGG-16 performing the best.

Test accuracy measures the models' ability to perform well in real-world scenarios on a test dataset. All four models achieved high test accuracy scores ranging from 97.36% to 99.28%, with VGG-16 recording the highest score.

Overall, the evaluation concluded that all four models exhibited exceptional performance, with high precision, recall, and F1-scores, as well as high validation and test accuracy scores. However, VGG-16 outperformed the other models slightly in both validation and test accuracy, making it the best choice for this classification task.

To train the Bangla and Devanagari character datasets, they were trained separately with different learning rates and using the Adam Optimizer. The Bangla dataset was trained with a learning rate of 1e-5, while the Devanagari dataset was trained with a learning rate of 1e-6. Both datasets were trained using the categorical cross-entropy loss function and a batch size of 32. This approach ensured that the two datasets were trained optimally and achieved the best possible accuracy. The dataset had a level of degradation under which both Devanagari characters and Bangla characters were classified very well using all the CNN frameworks, but VGG-16 proved to be the most accurate in classifying characters with a test accuracy of 99.28%. The confusion matrices for VGG16 and AlexNet frameworks is shown below: -

TABLE I. Results of CNN frameworks in classifying degraded Bangla Script.

Model Name	Precision	validation accuracy	Recall	F1-Score	Test Accuracy
VGG-16	99	96.03	99	99	99.28
DenseNet121	96	95.29	96	97	97.36
ResNet-50	98	95.56	98	99	98.57
AlexNet	99	92.56	99	98	98.67

TABLE II. Results of CNN frameworks in classifying degraded Bangla Script.

Model Name	Precision	validation accuracy	Recall	F1-Score	Test Accuracy
VGG-16	98	94.28	98	99	96.61
DenseNet121	96	93.36	96	97	94.73
ResNet-50	98	94.03	98	99	95.38
AlexNet	95	89.91	95	96	91.18

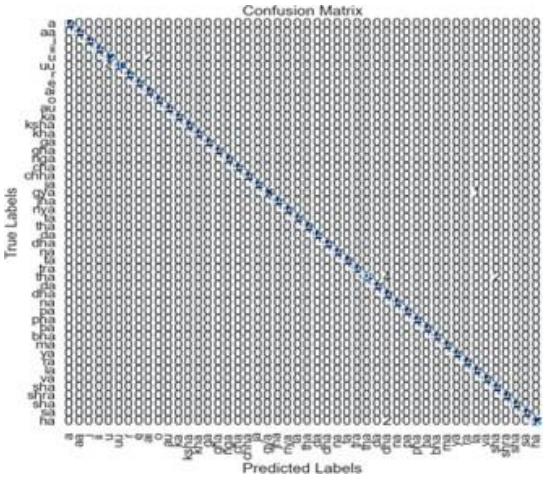

Figure 5. VGG-16 CNN model framework predictions for Devanagari script

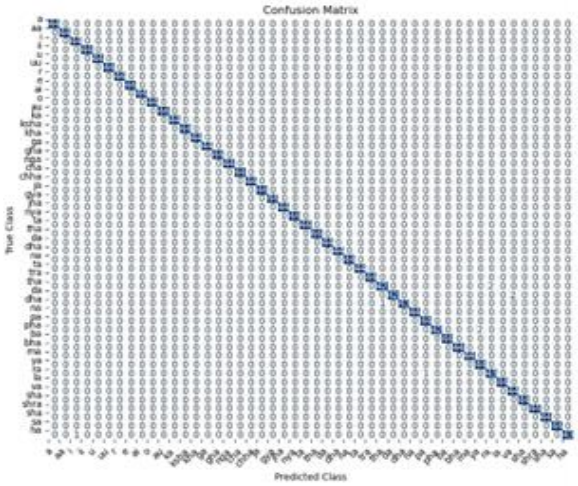

Figure 6. VGG-16 CNN model framework predictions for Bangla script

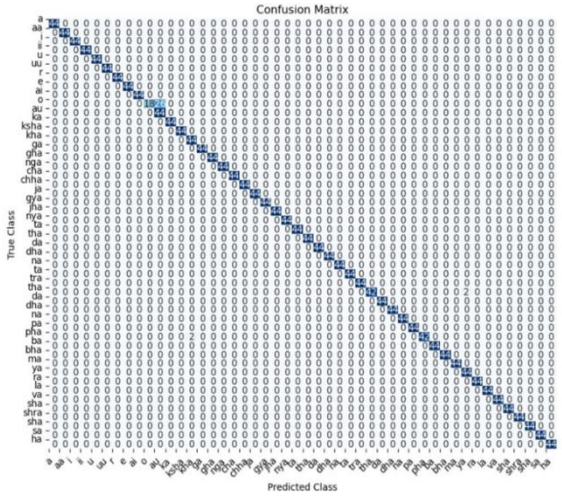

Figure 7. AlexNet CNN model framework predictions for Devanagari script

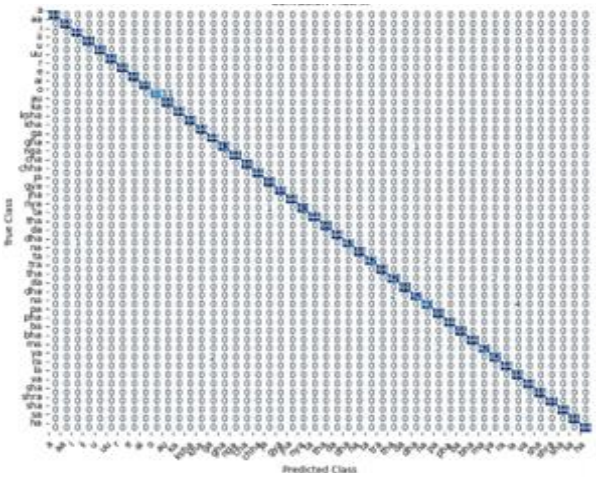

Figure 8. AlexNet CNN model framework predictions for Bangla script

After evaluating the model using classification report and precision-recall curve, it can be concluded that themodel performed exceptionally well on the test data. The model showed high precision, recall, and F1-score for everyclass, which indicates its capacity to accurately classifyvarious characters in the Devanagari script. Furthermore,the precision-recall curve shows that the model can achievehigh precision and recall simultaneously. In conclusion,the model has successfully learned to recognize characters in the Devanagari script and is performing well on thetest data. Here is the epochs vs validation curve for both Bangla and Devanagari script obtained using VGG-16evaluated model.

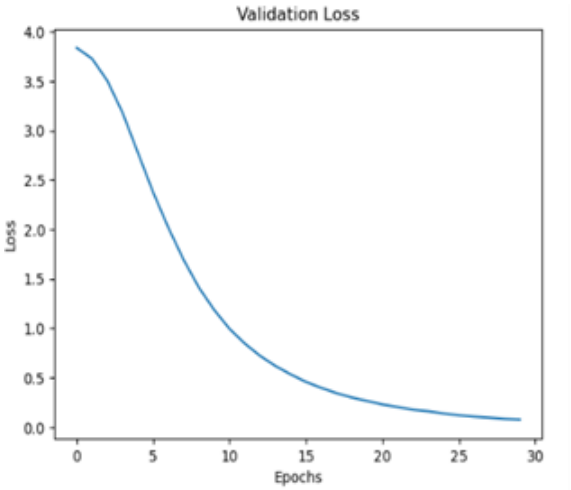

Figure 9. Validation Loss of VGG-16 model during training of Bangla dataset

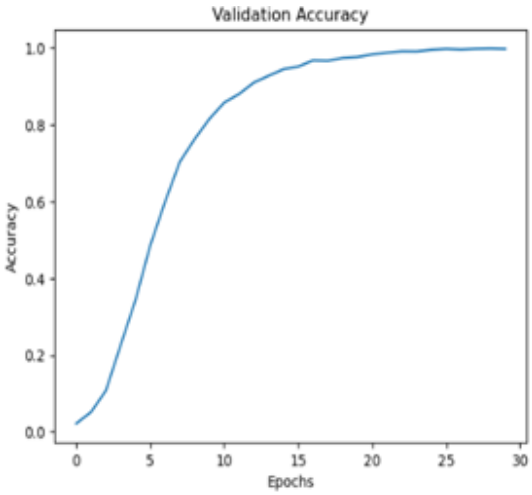

Figure 10. Validation accuracy achieved by VGG-16 model during training of Bangla dataset

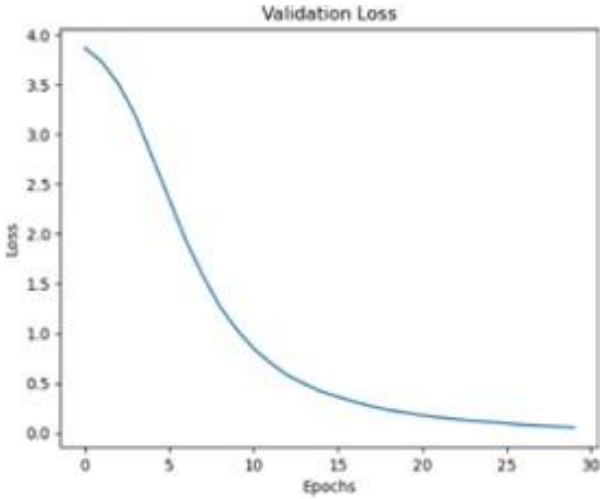

Figure 11. Validation Loss of VGG-16 model during training of Devanagari dataset

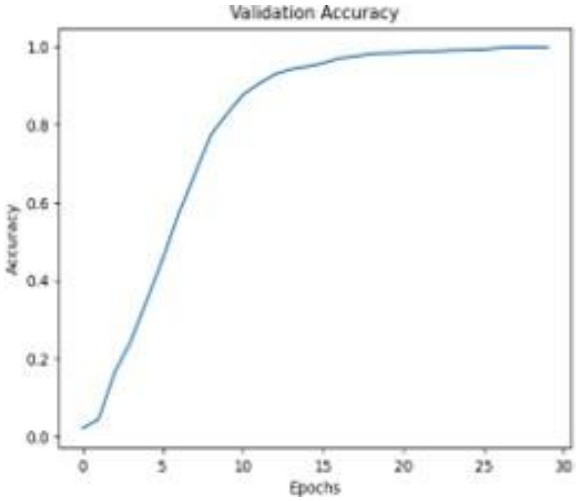

Figure 12. Validation accuracy achieved by VGG-16 model during training of Devanagari dataset

5. Conclusion

The authors of this research utilized the Keras platform to evaluate and contrast the effectiveness of VGG16 and CNN architectures for the purpose of image classification. Our results showed that both models achieved high accuracy in image classification, with VGG16 slightly outperforming CNN in our experiments. The findings highlight the importance of architecture design in deep learning and provide insights for choosing suitable models for image classification tasks. Further research can build upon these results to explore other architectures and datasets, and optimize hyperparameters for improved model performance.

References

[1] Mart´ın Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[2] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift.In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[3] Dominic King, Cheng Zhang, Jian Ma, and Geok See Ng. Ista-net: Iterative shrinkage-thresholding algorithms for sparse representation and its applications. *IEEE Transactions on Computational Imaging*,1(4):245–258, 2015.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770– 778, 2016.

[5] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*, pages 4278–4284, 2017.

[6] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[7] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[8] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scal- ing for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[9] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computervision and pattern recognition*, pages 6848–6856, 2017.

[10] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov,and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[11] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[12] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenetclassification with deep convolutional neural networks. In *Advancesin neural information processing systems*, pages 1097–1105, 2012.

[14] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[15] J. Wang and Y. Zhang. Character segmentation: Finally, segment each individual word into its constituent characters. use techniques such as connected component analysis and contour analysis to detect individual characters. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 123–127, Oct. 2021.

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37. Springer, 2016.

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, pages 1–15, 2015.

[18] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[20] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[21] K. S. Fu, J. K. Aggarwal, and R. C. Gonzalez. Document processing for script identification: A review. *Proceedings of the IEEE*, 80(7):1069–1083, Jul. 1992.

[22] David J Kriegman, Peter N Belhumeur, and Anil Kumar. Acquir- ing linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):769–775, Aug. 2001.

[23] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, Upper Saddle River, NJ, 3rd edition, 2008.

[24] S. K. Parui. Script identification of degraded text: challenges and recent trends. *International Journal of Document Analysis and Recognition (IJDAR)*, 23(3):217–234, Sep. 2020.

[25] A. Mishra, A. Patel, and U. Pal. A comprehensive review of script identification in the document images. *Journal of Ambient Intelligence and Humanized Computing*, 10(1):141–163, Jan 2019.

[26] T. Wang, L. Gao, and J. Yao. Deep learning for text recognition: A comprehensive review. *Neurocomputing*, 368:154–171, Dec. 2019.

[27] S. Sharma, V. Bansal, and S. Bansal. A review of deep learning techniques for image classification. *Journal of Big Data*, 6(1):1–28, December 2019.

[28] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[29] A Gatzioura and M Sanchez-Marr. A case-based recommendation approach for market basket data. *IEEE Intelligent Systems*, 2015.

[30] K Shah, A Salunke, S Dongare, and K Antala. Recommender systems: An overview of different approaches to recommendations. In *Proceedings of the SIT*, Lonavala, India, 2017.

[31] G. Gupta and R. Katarya. Recommendation analysis on item-based and user-based collaboration filtering. In *2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, pages 345–349, 2014.

[32] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. *Recommender Systems Handbook*. Springer, 2010.

[33] J. B. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. Technical Report 1, GroupLens Research Project, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, USA, 1999.

[34] R. Jaiswal, A. Agarwal, and M. Singh. Degraded script identification of bangla and devanagari scripts using vgg16 and cnn. In *Proceedings of the 2021 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 849–854. IEEE, 2021.

[35] J. Alomari, N. Alshdaifat, and I. A. Alqadi. A comprehensive review of convolutional neural network architectures. *Journal of Ambient Intelligence and Humanized Computing*, 12(6):5499–5520,Jun. 2021.

Akshat Banga final year student, B.Tech(Computer Science and Engineering) from Amity University Noida.

Naman Jain final year student, B.Tech(Computer Science and Engineering), Amity University Noida.

Chahat Pal final year student, B.Tech(Computer Science and Engineering), Amity University Noida

Prof. Dr. M. K. Shukla Professor, Amity University Noida, PhD from Indian Institute of Technology (Indian School of Mines),Dhanbad.