Defending Authenticity: Real-Time Signature Verification with Deep Learning Models

Malireddy Charan Kumar Reddy[1], Katragadda Megha Shyam[2], Kandlapalli Aravind Sai[3], Vishwas HN[4]
Dept. of Computer Science and Engineering, Amrita School of Computing, Bengaluru, Amrita Vishwa Vidyapeetham, India.
bl.en.u4cse21121@bl.students.amrita.edu,bl.en.u4cse21097@bl.students.amrita.edu, bl.en.u4cse21087@bl.students.amrita.edu,
hn_vishwas@blr.amrita.edu

Abstract*— Deep learning-based signature verification uses sophisticated neural network topologies to verify signatures. Deep learning models may be trained to discriminate between authentic and falsified signatures with a high degree of accuracy by utilizing methods such as vgg, xceptionmodels *and Transfer Learning. These models are made to examine the complex characteristics and patterns found in signatures, which allows them to identify minute variations that could point to fraud. Deep learning methods provide a reliable and effective way to handle jobs involving the verification of signatures by utilizing large libraries and complex algorithms. The implementation of deep learning techniques for image processing and classification tasks, which can be adapted for signature verification applications. By preprocessing signature images, extracting relevant features, and training deep learning models on labeled data, the system can learn to differentiate between authentic and forged signatures effectively. Using deep learning involves the development and deployment of neural network models that can accurately authenticate signatures by analyzing their unique characteristics and patterns.

***Keywords* - Signature verification, deep learning, vgg model, xception model, data sets**

I. INTRODUCTION

In order to guarantee the confidentiality and integrity of financial and legal transactions, signature verification is a crucial component of document authentication. Conventional methods of verifying signatures frequently depend on labor-intensive, human error-prone rule-based systems and manual feature extraction. Deep learning methods, on the other hand, provide a viable way to automate this procedure by using neural networks' ability to extract significant characteristics from signature photos directly. In order to develop cutting-edge models, this study explores the use of deep learning in signature verification, making use of well-known frameworks like Keras and TensorFlow. Deep learning models may be made more resilient and accurate by utilizing methods like picture data creation and preprocessing, which allow the models to adapt to different settings and signature styles..

Additionally, quantitative insights into the performance of these models are provided by model assessment approaches including confusion matrices, accuracy scores, and classification reports, which facilitate optimization and fine-tuning. This paper intends to clarify the efficacy of deep learning architectures such as Xception, EfficientNetB0, NASNetLarge, InceptionV3, and VGG16 in signature verification tasks on a variety of datasets by means of a thorough investigation.

Deep learning-based signature verification introduces the use of advanced neural network models for signature authentication. Through the utilization of Convolutional Neural Networks (CNNs) and Transfer Learning, deep learning methodologies may be applied to accurately discern between authentic and counterfeit signatures. These models may identify minute variations that can point to possible fabrication since they are made to examine the complex patterns and characteristics found in signatures. By using large datasets and sophisticated algorithms, deep learning techniques provide a reliable and effective way to handle the problem of verifying signatures.

The process of signature verification using deep learning involves several stages, including data collection, data preprocessing, model training, and model evaluation. The data collection stage involves gathering a large dataset of genuine and forged signatures, which are then preprocessed to enhance their quality and normalize them for training. The model training stage involves feeding the preprocessed data into a deep learning model, which learns to extract features and classify the signatures as genuine or forged. The model evaluation stage involves testing the trained model on a separate dataset to evaluate its performance and accuracy

II. RELATED WORK

In this Paper [1] author presents The verification of offline signatures is a challenge since they lack distinctive properties. Therefore, a Deep Learning (DL) based technique utilizing Convolutional Neural Networks (CNN) for verification has been proposed. The current work aims to make a contribution to the area as previous attempts to verify offline signatures using DL have not yielded substantial results. CNN models have demonstrated success in a number of areas, and it is anticipated that they will perform well in offline signature verification given the use of feature extraction techniques. CNNs have been successfully applied in several fields related to feature learning, including autonomous cars, object identification, video processing, medical image processing, and character recognition.With Writer Dependent (WD) attaining 75% success and Writer Independent (WI) reaching 62.5% success, the study demonstrates CNN's effectiveness in offline signature verification. However, it also suggests that new feature extraction techniques might lead to even greater improvement.

In the paper[2] the author explains A variety of strategies, including as holistic, regional, and multiple regional matching techniques, are used to compare signatures. The following techniques are often used in signature analysis: neural networks, neural matching, split and merge, relaxation matching, tree matching, elastic matching, neural correlation, Hidden Markov Models (HMM), and Support Vector Machines (SVM). Effective analysis and comparison of signatures using classifiers has been the subject of research. Prominent research works have condensed the application of classifiers in signature verification, improving knowledge of signature analysis methods. Researchers have been creating synthetic signatures to expand the dataset for training and

testing signature verification algorithms in an effort to overcome the problem of scarce sample availability.

In Paper [3] the author Due to the time-variant nature of signatures, reliable offline signature verification requires the use of sophisticated technologies like deep learning. Offline signature verification systems typically employ two main approaches: Writer Dependent (WD) and Writer Independent (WI). While WI concentrates on a more broad strategy, WD trains the classifier individually for each person. Prior studies have demonstrated that although deep learning techniques have been applied to signature verification, their effectiveness has not always been great. By showcasing the efficacy of a CNN-based signature verification method, this research seeks to advance the discipline

In the paper [4] the author Tells The study emphasizes the significance of this technology in a number of domains, including administration, finance, and security. It focuses on offline handwritten signature verification using deep learning techniques. The two primary facets of handwritten signature analysis that are covered are verification and recognition. Finding a user's signature's validity is the process of verification; for reliable verification, the False Acceptance Rate (FAR) and False Rejection Rate (FRR) should be as low as possible. The study highlights the application of neural networks, in particular convolutional neural networks (CNN), for tasks related to signature verification, demonstrating the efficacy and efficiency of these networks in this field.

In the Paper[5] The Author Tarek Atia classified binary images with a custom CNN model, ResNet-50, VGG-16, Inception-v3, and Xception, all of which produced encouraging results. CNNs with ADAM and RMSprop were employed by Kancharla et al. for offline signature recognition; they observed that the accuracy of ADAM varied depending on the dataset. Noor et al. used CNNs to obtain excellent accuracy with preprocessed signature photos.For online signature verification, Miaba et al. integrated the DFT, DCT, and DWT transforms.

In the Paper [6] the author The use of handwritten signatures in biometric technology is essential for personal verification. Scholars have delved deeply into the field of signature verification, synthesizing developments both prior to and beyond 1989. Signature verification has been accomplished by a variety of techniques, including the contourlet transform, co-occurrence matrix, and SVM algorithms. Systems for verifying signatures frequently employ deep learning models for categorization, such as neural networks and SVM.

In the Paper [7] the Author Emphasizing the significance of signature verification, researchers are investigating several techniques to discern between authentic and counterfeit signatures. Using TensorFlow for deep learning and a Convolutional Neural Network (CNN) for precise verification of distinct signature traits, the research attempts to create a dependable system for identifying signatures on bank checks and legal documents. False Rejection Rates (FRR) and False Acceptance Rates (FAR) are used to assess the efficacy of the proposed system. During testing, both rates were reported to have values of 5%, while the system's total accuracy was 90%.

In the Paper [8] the author Verifying handwritten signatures is essential for confirming identification and authenticating documents. It uses both dynamic and static verification techniques. User acceptability, security level, accuracy, cost, and implementation are examples of critical parameters. For Handwritten Signature Verification (HSV), the study assesses deep convolutional neural networks (CNNs) such as VGG16, VGG19, and ResNet50. Activation functions are utilized in transfer learning. The best accuracy, 97.83%, was displayed by VGG19.The handwritten signatures in the dataset, which was gathered with friends' assistance, is from SigComp 2009.

In the Paper [9] the author explains In order to differentiate between real and fake signatures, the research article focuses on signature verification utilizing TensorFlow, a well-liked deep learning framework. The use of Convolutional Neural Networks (CNNs), which are popular in image classification applications because of their capacity to learn spatial hierarchies of information, is mentioned for the verification of signatures. In order to assess the efficacy of the signature verification system, the study addresses the significance of False Rejection Rates (FRR) and False Acceptance Rates (FAR) as critical metrics. According to reports, the system has an overall accuracy of 90%. During testing, the FAR and FRR values were 5% each, demonstrating the system's resilience in identifying counterfeit and authentic signatures.

III. METHODOLOGY

The objective is inn order to effectively represent real-world settings, deep learning signature verification starts with the acquisition of a varied dataset that includes both authentic and forged signature photos. This ensures that there is heterogeneity in style, size, and quality. The quality and diversity of the dataset are then improved by applying data preprocessing techniques, such as scaling, normalization, and augmentation.

Following dataset preparation, appropriate deep learning architectures, such as Xception, EfficientNetB0, NASNetLarge, InceptionV3, and VGG16, are selected based on factors like computational efficiency and complexity. These models are then trained on the prepared dataset, with careful division into training, validation, and test sets to prevent overfitting, using optimization techniques like stochastic gradient descent (SGD) or Adam optimizer.

Model evaluation is conducted on the test set, employing metrics like accuracy, precision, recall, F1-score, and confusion matrices to gauge performance in distinguishing between genuine and forged signatures. Further optimization involves fine-tuning hyperparameters and exploring transfer learning and ensemble methods to enhance generalization. The optimized model is then deployed in real-world scenarios, where its performance is validated on unseen data, with ongoing monitoring and refinement based on feedback and new data.

A. VGG16 model:

Signature verification benefits greatly from the use of the VGG16 model, a convolutional neural network architecture known for its ease of use and efficiency in picture classification applications. VGG16 is able to extract discriminative features from signature photos by utilizing transfer learning from pretrained weights on large-scale datasets such as ImageNet. To improve the model's capacity

to discriminate between real and fake signatures, fully linked layers are usually customized and adjusted to the particular task domain in signature verification jobs.

B. Xception model:

The Xception model is a very efficient convolutional neural network architecture that was first created for image classification. It is renowned for its tremendous depthwise separable convolutions. Xception is a useful tool in signature verification jobs because of its ability to capture complex patterns while retaining computing speed. Xception uses pretrained weights and transfer learning to extract discriminative features from signature photos on datasets like ImageNet. Xception can reliably and robustly discriminate between authentic and forged signatures by optimizing using techniques such as stochastic gradient descent and fine-tuning the model's top layers to the particular task area.

C. NASNet model

A breakthrough in neural network architectural design is the Neural architectural Search Network, or NASNet model. NASNet, created by researchers at Google AI, uses techniques from reinforcement learning to automatically find effective neural network topologies. With the least amount of processing resources, NASNet architectures may achieve state-of-the-art performance on a wide range of computer vision applications because to their excellent scalability and versatility. Because of its flexibility in handling various signature variants and styles, NASNet presents a viable option in the area of signature verification. By leveraging transfer learning from pretraining on large-scale datasets, NASNet can effectively extract relevant features from signature images. Fine-tuning the model's parameters and optimizing its performance using techniques such as stochastic gradient descent further enhances its effectiveness in distinguishing between genuine and forged signatures.

D. Inception model

Convolutional neural network (CNN) researchers at Google created the Inception model, sometimes referred to as GoogLeNet. It is distinguished by the creative way in which it uses inception modules, which use several simultaneous convolutional procedures of varying filter sizes to effectively collect information at different scales. With minimal computing costs thanks to this architecture, the model can extract hierarchical and complicated information from photos.

E. Efficientnet model

Convolutional neural network (CNN) models of the EfficientNet family were developed by Google researchers with the goal of maximizing performance while consuming the least amount of computing power. EfficientNet performs remarkably well under a range of resource limitations by dynamically adjusting the network's depth, breadth, and resolution. EfficientNet's design is flexible and useful in signature verification, since it can extract unique characteristics from signature pictures quickly.

IV. IMPLEMENTATION

On using Convolutional Neural Networks (CNNs), a type of deep learning approach, to image classification tasks associated with signature verification. The required packages are built up, datasets like CIFAR-10 are imported, model architectures are defined using pre-trained models like ResNet50, ResNet152V2, and DenseNet121, and layers for convolution, pooling, normalization, and activation functions are incorporated. Furthermore, the implementation entails training the model with optimization algorithms like Adam, preparing the data using methods like image augmentation, and evaluating the model using metrics like the confusion matrix, accuracy score, and classification report. The code also handles loading, resizing, and converting images to numpy arrays for additional processing, showing how to use deep learning for jobs involving signature verification in a thorough manner.

The libraries used in the implementation for signature verification using deep learning include Keras, TensorFlow, scikit-learn, mlxtend, imutils, matplotlib, pandas, numpy, OpenCV (cv2), PIL, and Google Colab specific libraries like google.colab, google.colab.patches, and google.colab.drive. These libraries are essential for tasks such as model building, data preprocessing, image manipulation, evaluation metrics calculation, and visualization within the deep learning framework.

Fig 1. Sample Images for Forged and Genuine

A. Data Labellingl:

The data labeling process in the provided implementation involves organizing images into categories such as genuine and forged signatures. In our datasets there consists of 887 genuine images and 785 forged images

B. Train-Test-Validation Split:

The train-test-validation split in the provided implementation involves dividing the dataset into three subsets: training, testing, and validation sets. We obtained Train set Size is 1002, Validation set size is 335, Test set size is 335.

C. Model Training:

The training of a deep learning model using the VGG16 architecture for image classification tasks related to signature verification. The model is trained on a dataset split into

training, validation, and test sets, with images resized to match the input shape of the VGG16 model. The training process involves freezing the pre-trained layers of the VGG16 model, adding custom classification layers, compiling the model with the Adam optimizer, and iteratively training the model over multiple epochs while monitoring its performance on the validation set to prevent overfitting.

V. Results and Discussion

The trained VGG16 model achieves high accuracy on both the training and validation sets, with the training accuracy reaching 100% and the validation accuracy exceeding 94% after 40 epochs of training. Techniques like data augmentation and regularization may have been employed to further mitigate overfitting and improve the model's generalization ability. The implementation compares the performance of the VGG16 model with other pre-trained models like ResNet50, ResNet152V2, and DenseNet121. The results show that the VGG16 model outperforms the other models in terms of accuracy and loss on the signature verification dataset.

The model was trained for 40 epochs on a dataset, with the training loss and accuracy, as well as validation loss and accuracy, recorded at the end of each epoch. The model achieved a final training accuracy of 100% and a validation accuracy of 94.93%.

The validation loss decreased from 0.3450 in the first epoch to 0.1515 by epoch 23, which was the lowest validation loss achieved. This indicates that the model was able to learn meaningful features from the training data and generalize well to the validation set.

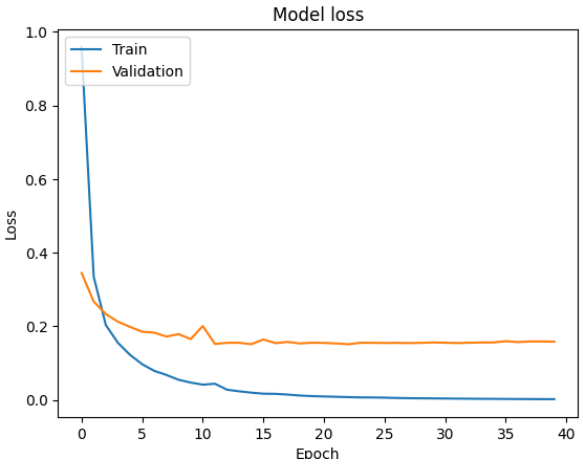

Fig 2. Plot training & validation loss values

The training loss decreased from 0.9586 in the first epoch to 0.0028 by the final epoch, showing that the model was able to fit the training data very well. However, the validation loss started increasing slightly after epoch 23, suggesting that the model may have started overfitting to the training data.

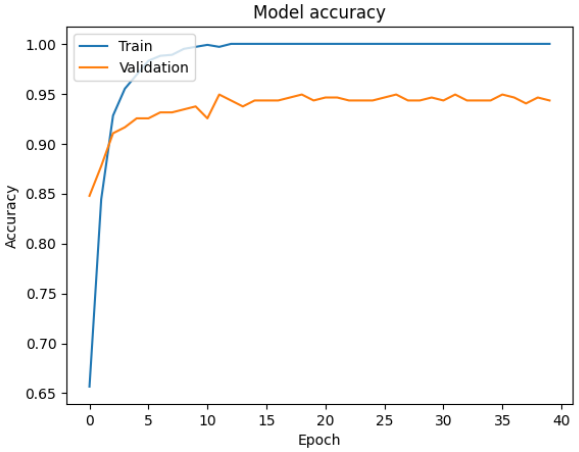

Fig 3. Plot training & validation accuracy values

A ROC curve, you would typically need to calculate the true positive rate and false positive rate at different thresholds for the model's predictions. This is often done using the roc_curve function from scikit-learn in Python. However, the provided results do not include this information.

To analyze the performance of a classification model using a ROC curve, you would need to calculate the true positive rate and false positive rate at different thresholds and then plot these values. The ROC curve provides a visual representation of the model's performance, with the area under the curve (AUC) being a common metric used to evaluate the model's performance.

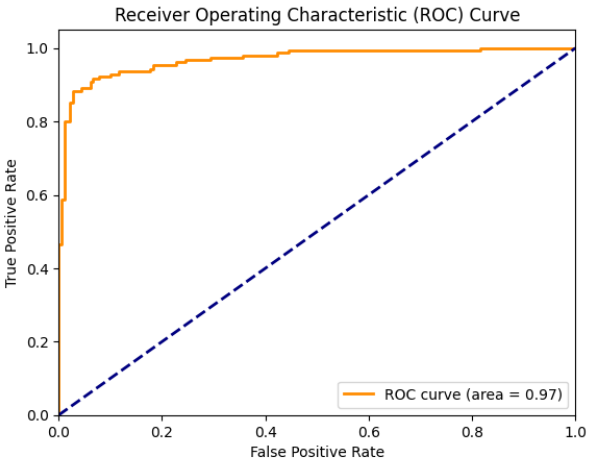

Fig 4. ROC curve VGG16 MODEL

The training loss decreased from 0.9586 in the first epoch to 0.0028 by the final epoch, showing that the model was able to fit the training data very well. However, the validation loss started increasing slightly after epoch 23, suggesting that the model may have started overfitting to the training data.

The model achieved a high validation accuracy of nearly 95%, indicating that it can classify the images in the dataset with a high degree of accuracy. The training and validation metrics demonstrate that the model was able to learn effectively from the data and generalize well to unseen examples.

Fig 5. Accuracy of Train, Validation, Test

S.NO	Model Name	Train Accuracy	Validation Accuracy	Test Accuracy
1	VGG16	100%	94%	92%
2	XCEPTION	91.4%	91.5%	89%
3	Efficient Net	98.1%	92.8%	93.7%
4	NASNetLarge	87%	88%	87%
5	InceptionV3	93%	87%	97%

To generate a classification report including precision, recall, and F1-score for a classification model, you typically use the classification report function from scikit-learn in Python. This report provides a summary of various metrics for each class in the classification task. In our project we got high classification report for VGG16.

S.NO	MODEL NAME	PRECISION	RECALL	F1-SCORE
1	VGG16	92.2%	92%	92.5%
2	XCEPTION	67.1%	67.4%	66.8%
3	Efficient Net	93.7%	93.3%	93.7%
4	NASNetLarge	77%	87%	92.4%
5	Inception	78.3%	88.2%	92.5%

Fig 6. Classification Report

For VGG16 the results obtained for precision is 92.2%, for Recall 92% and for F1-score is 92.5%. we got less classification report results for xception model for precision is 67.1%, for recall is 67.4% and for F1-score is 66.8%.

VI. CONCLUSION

The outlining a deep learning model's training procedure for a binary image classification job that involves actual and fraudulent signatures. The model in question is probably a VGG16 model. Training and validation metrics were recorded at each epoch during the 40 epochs that the model was trained on. At 100% for final training accuracy and 94.93% for validation, the model performed flawlessly. After gradually declining, the validation loss peaked at epoch 23 at 0.1515. The validation loss did, however, begin to somewhat rise after epoch 23, suggesting that overfitting may have occurred. Promising outcomes for signature verification tasks were shown by the model's performance, which efficiently distinguished between genuine and fraudulent signatures. Improved generalization of the model might be achieved with additional improvements including data augmentation and model architectural modifications.

Deep learning models also have the potential to adjust to varied writing styles and languages, which increases their relevance in a variety of cultural and demographic situations. By incorporating these models into digital security systems, identity verification procedures become more dependable and efficient overall. This is achieved by offering a scalable and automated solution that lessens the need for human knowledge and manual scrutiny.

Deep learning models for signature verification rely on large, high-quality datasets being available, as well as meticulous model parameter optimization. In addition, for these models to be used in practical applications, problems like adversarial assaults must be resolved and their interpretability must be guaranteed. It is anticipated that continued research and development in this field will improve these models even more, increasing their accuracy, dependability, and security.

Finally, with significant gains in accuracy, scalability, and flexibility, deep learning models constitute a revolutionary development in signature verification. These models will probably play a crucial role in safe authentication systems as technology develops, giving rise to increased security in a number of applications including access control, finance, and legal paperwork.

REFERENCES

[1] Yapici, Muhammed Mutlu, Adem Tekerek, and Nurettin Topaloglu. "Convolutional neural network based offline signature verification application." 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT). IEEE, 2018.

[2] Tamrakar, Prasann, and Abhishek Badholia. "Handwritten signature verification technology using deep learning–a review." 2022 3rd International Conference on Electronics and Sustainable Communication Systems (ICESC). IEEE, 2022.

[3] Rokade, ShubhangiS, et al. "An Offline Signature Verification Using Deep Convolutional Neural Networks." 2023 Third International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT). IEEE, 2023.

[4] Muhtar, Yusnur, et al. "A Survey of Offline Handwritten Signature Verification Based on Deep Learning." 2022 3rd International Conference on Pattern Recognition and Machine Learning (PRML). IEEE, 2022.

[5] Mustafa, Basmala, et al. "Handwritten Signature Recognition using Deep Learning." 2023 International Conference on Microelectronics (ICM). IEEE, 2023.

[6] Blessy, Prasanna, K. Kathiresan, and N. Yuvaraj. "Deep Learning Approach to Offline Signature Forgery Prevention." 2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS). Vol. 1. IEEE, 2023.

[7] Shenoy, Ashwin, Saritha Suvarna, and K. T. Rajgopal. "Online digital cheque signature verification using deep learning approach." 2023 2nd International Conference on Edge Computing and Applications (ICECAA). IEEE, 2023.

[8] Rai, Rahul D., and J. S. Lather. "Handwritten signature verification using TensorFlow." 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT). IEEE, 2018.

[9] Sudharshan, Duth P., and R. N. Vismaya. "Handwritten signature verification system using deep learning." 2022 IEEE International Conference on Data Science and Information System (ICDSIS). IEEE, 2022.

[10] Krishna, Chetlapalli Amritha, and R. Bhuvaneswari. "Offline Signature Forgery Detection using Multi-Layer Perceptron." 2023 3rd Asian Conference on Innovation in Technology (ASIANCON). IEEE, 2023.

[11] Madhav, Kagolanu Venkata Chandra, et al. "Handwritten Devanagari Numeral Recognition Using Deep Learning." 2024 3rd International Conference for Innovation in Technology (INOCON). IEEE, 2024.

[12] Jayakrishnan, Hariharan, and Ritwik Murali. "A simple and robust end-to-end encryption architecture for anonymous and secure whistleblowing." 2019 Twelfth International Conference on Contemporary Computing (IC3). IEEE, 2019.

[13] Asmitha, M., and Shinu M. Rajgopal. "Exploring Unique Techniques to Preserve Confidentiality and Authentication." 2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT). IEEE, 2024.

[14] Aravind, G., et al. "Development of biometric security system using CBIR and EER." 2015 International Conference on Communications and Signal Processing (ICCSP). IEEE, 2015.

[15] Sudar, Chandramohan, S. K. Arjun, and L. R. Deepthi. "Time-based one-time password for Wi-Fi authentication and security." 2017 international conference on advances in computing, communications and informatics (ICACCI). IEEE, 2017.