Bitlis Eren Üniversitesi Fen Bilimleri Dergisi
BİTLİS EREN UNIVERSITY JOURNAL OF SCIENCE
ISSN: 2147-3129/e-ISSN: 2147-3188
VOLUME: 13 NO: 3 PAGE: 871-884 YEAR: 2024
DOI:10.17798/bitlisfen.1527670

Deep Learning Based Offline Handwritten Signature Recognition

Bahar ÇİFTÇİ[1*], Ramazan TEKİN[2]

[1]*Siirt University, Distance Education Application and Research Center*
[2]*Batman University, Faculty of Engineering and Architecture, Department of Computer Engineering*
(ORCID: 0000-0001-5976-6236) (ORCID: 0000-0003-4325-6922)

Keywords: Deep Learning, Signature Recognition, Image Classification, CNN Architectures

Abstract

In our digitalized world, the need for reliable authentication methods is steadily increasing. Biometric authentication methods are divided into two main categories: physiological and behavioral. While physiological biometrics include features such as face, iris, and fingerprint, behavioral biometrics encompass dynamics such as gait, speech, and signature. Most of these methods require specialized equipment, whereas signatures can be easily obtained without additional tools, making them ideal for verifying the legality of documents. Although manual signature recognition is effective, it is resource-intensive, slow, and susceptible to errors. With advancements in technology, the need to automate the signature recognition process to enhance accuracy and efficiency has become increasingly important. Based on this need, in this study, five different DL techniques (GoogLeNet, MobileNet-V3 Large, Inception-V3, ResNet50 and EfficientNet-B0) are used to classify signature images with detailed analyses. DL methods have outperformed traditional techniques by leveraging the power of CNNs to automatically learn and extract complex features from signature data. The dataset used consists of a total of 12,600 images belonging to 420 individuals, each contributing 30 original signatures. The dataset is divided into training, validation, and test sets in different proportions to analyze classification performance. The pre-trained DL models were fine-tuned to optimize their parameters for the signature dataset. The results demonstrate that DL models achieve high accuracy in signature classification, with the GoogLeNet and Inception-V3 models reaching an accuracy of 98.77% at a 20% test rate. The study also highlights the impact of different test rates on model performance.

1. Introduction

In our digitalized world, the need for reliable authentication methods is increasing. Biometric authentication methods are divided into two main categories: physiological and behavioral. Physiological biometrics include features such as face, iris, and fingerprint, while behavioral biometrics encompass dynamics such as gait, speech, signature, and keystroke dynamics [1], [2]. Although other techniques, such as fingerprint, iris/retina scanning, facial recognition, and voice recognition, are more precise, they require specialized equipment. In contrast, signatures can be easily obtained without additional tools, making them indispensable for daily transactions and verifying the legality of documents such as certificates, checks, letters, approvals, visas, and passports. Traditional signature verification has been a manual process involving the comparison of a sample signature with previously obtained genuine signatures. With advancements in technology, there has been an increasing need to automate the signature verification process to enhance accuracy and efficiency. Handwritten signature recognition is divided into two primary categories according to the method of signature acquisition: online and offline. Online signature recognition involves capturing dynamic features such as ink pressure and writing

*Corresponding author: bahar.ciftci@siirt.edu.tr
Received: 03.08.2024, Accepted: 12.09.2024

speed during the signing process using devices like electronic handwriting pads or touch screens [3], [4]. Offline signature recognition, on the other hand, involves writing the signature on paper, scanning it, and digitally processing the image [5]. Additionally, signature analysis encompasses two methods: signature verification and signature recognition. Signature recognition is a classification process where the signatory is identified from among many signatures. Signature verification, however, involves comparing two signatures to determine their similarity and conclude whether a signature is genuine or forged. Signature recognition studies require datasets with multiple signatures from different individuals, whereas signature verification necessitates the presence of both genuine and forged signatures for each individual.

Studies on offline signature recognition and verification date back to the early 1990s [6]. In their work, Sabourin and Drouhard Jean-Pierre classified offline signature images using Feedforward Neural Networks (FFNN) and the Probability Density Function (PDF) [6]. Ismail and Gad used various preprocessing techniques to classify 220 genuine and 110 forged Arabic signature images using different algorithms based on fuzzy concepts [7]. Deore and Handore's study utilized Discrete Wavelet Transform (DWT) and Principal Component Analysis (PCA) for extracting features, followed by Artificial Neural Networks (ANN) to implement an offline signature recognition system [8]. As technology advances, the methods we use to verify signatures must also evolve to remain robust against ever-progressing forgery techniques. The integration of ML and DL techniques into signature recognition systems has proven to be highly effective, enhancing accuracy and efficiency while preserving the authenticity of documents and the integrity of transactions. Calik and colleagues used the SUSIG-Visual dataset for classification with CNNs [9]. Gumusbas and Yildirim achieved high accuracy with CapsNet using the CEDAR [10] dataset, which contains genuine and forged signatures Gumusbas & Yildirim, 2019). In another study, the same authors used CapsNet for signature verification with different datasets (GPDS-100 and MCYT) [12]. Tarek and Atia used various CNN models, including VGG-16, ResNet50, Inception-v3, and Xception, to classify genuine and forged signatures from two separate datasets [13].

For the recognition and verification of offline signatures, various feature extraction processes are performed before proceeding to the classification stage with ML and DL. In their study, Jain et al. used GoogLenet for feature extraction and then used SVM method for classification [14]. Foroozandeh et al., in their study, used SigNet, SigNet-F, VGG16, VGG19, InceptionV3, and ResNet50 for feature extraction, and then preferred the Support Vector Machine (SVM) method for classification [15]. Özyurt et al., in their study, performed feature extraction using the MobileNetV2 method on the large dataset they obtained. During the feature selection phase, they used Neighborhood Component Analysis (NCA), Chi-Square (Chi2), and Mutual Information (MI) methods for classification. Their evaluation employed various ML classifiers, including Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Tree (DT), Linear Discriminant Analysis (LDA), and Naïve Bayes. The study results showed that the highest accuracy of 97.7% was achieved using SVM with NCA feature selection [16]. Pokharel et al., in their signature recognition application, used GoogLeNet for both feature extraction and classification [17].

The drive to develop and improve signature classification systems arises from the need to ensure the security of identities and transactions in an environment where the boundaries between the physical and digital worlds are increasingly blurred. The main motivation of this work is to develop efficient and reliable authentication systems that can respond to the growing demands of modern society, while ensuring that signatures remain a reliable means of authentication

2. Material and Method

2.1. Dataset

The dataset used in this study consists of 12,600 images from 420 different individuals, with each participant contributing 30 genuine signatures (Özyurt et al., 2024). The signatures were obtained from students and faculty members at Raparin University in Ranya, Iraq, and Firat University in Elazig, Turkey. Collected within two months, the signatures were cropped using MATLAB and then resized to 500x600 pixels.

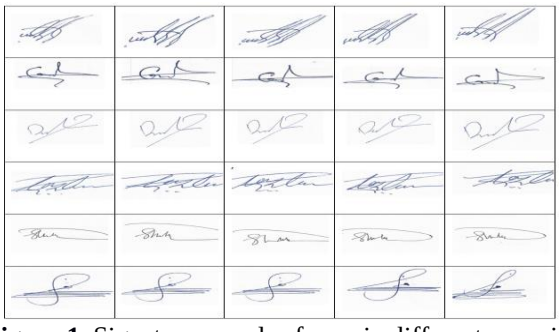

Figure 1. Signature samples from six different users in the dataset

2.2. Proposed Method

The classification of the signature images in this dataset was performed using five DL techniques, considering the remarkable success of DL in various fields.

In the approach proposed for the study, the dataset was divided into different proportions to analyze the performance of classification accuracy. The dataset was split into 20% test and 80% (train + validation) sets. The validation data was then split from the training data as 20%. Thus, in the first scenario, the dataset was divided into 20% test, 16% validation, and 64% train. In the second scenario, the dataset was split into 30% test and 70% (train + validation) sets. The validation data was then split from the training data as 30%. Thus, the second scenario consisted of 30% test, 21% validation, and 49% train. In the third scenario, the dataset was split into 40% test and 60% (train + validation) sets. The validation data was then split from the training data as 40%. Thus, the third scenario consisted of 40% test, 24% validation, and 36% train. Various preprocessing methods were then applied to the train-validation and test datasets. The DL models used were pretrained DL models. The pretrained weights were applied to the dataset, and parameters were optimized for new models. Subsequently, each of these models was fine-tuned individually to enhance their performance. For each classification scenario, train-validation, and test success, loss, and weights were recorded for all epochs. The flowchart of the proposed method is shown in detail in Figure 2.

2.3. Preprocessing

Preprocessing is essential in classification tasks, including signature classification, as it cleans and standardizes raw data, which often contains inconsistencies and irrelevant information. This step enhances DL classifier models' effectiveness and helps prevent overfitting, leading to more accurate results. (James & Koresh, 2023). The preprocessing methods used in this study include Resize, CenterCrop, ColorJitter, GaussianBlur, Grayscale, and Normalize. Resize is used to resize images to a fixed size. CenterCrop; makes a crop of a certain size from the center of the image. For the Inception-V3 model used in the study, Resize 299 and CentreCrop value 299 were used. For EfficientNet-B0, MobileNet-V3-Large, GoogLeNet and ResNet50 models, Resize 299 and CenterCrop 299 were used. Normalization is a pre-processing step in which the pixel values of the images are scaled to a certain range or distribution. Since the DL models used in the study are Pretrained models previously trained with ImageNet; Normalization values are used as [0.485, 0.456, 0.406] and [0.229, 0.224, 0.225].

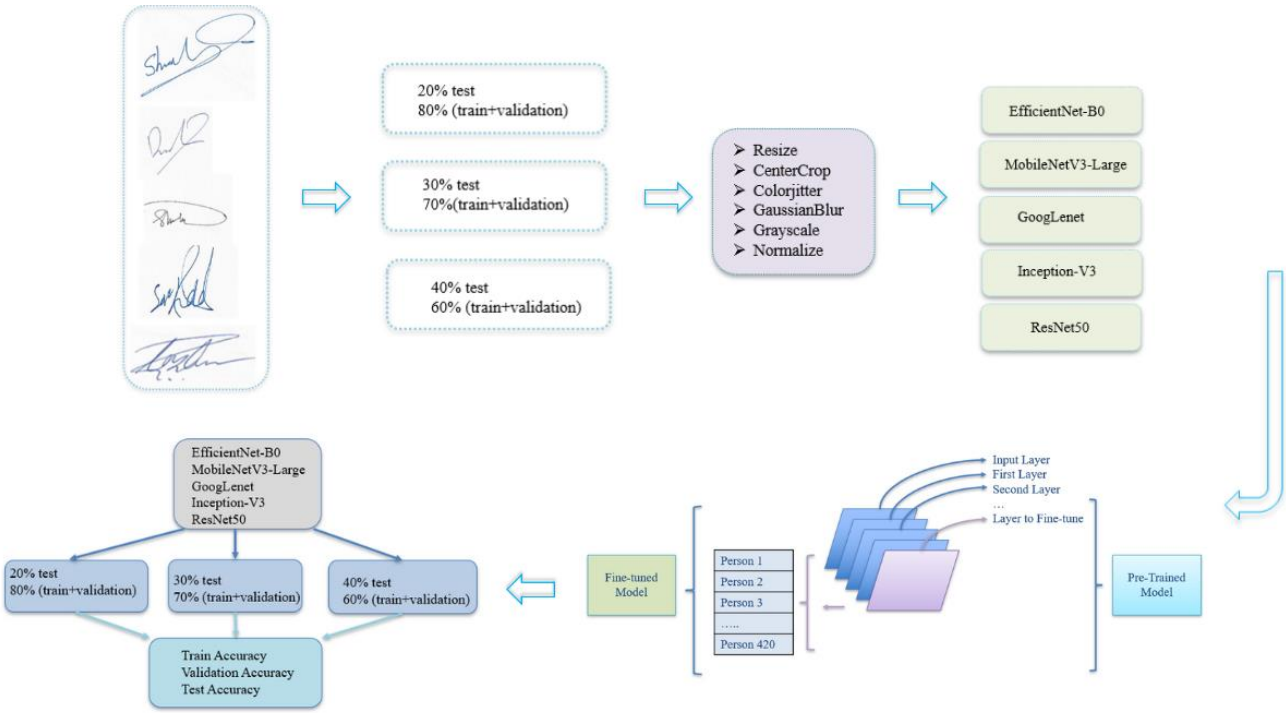

Figure 2. Flowchart of the Implemented Signature Recognition Application

Colourjitter is a preprocess that adjusts the brightness, contrast, saturation and hue of the image. GaussianBlur applies a Gaussian blur to the image. For this study, p=0.1 was chosen for the Colourjitter and GaussianBlur methods, and the transformations were applied with a 10% probability. In the GaussianBlur method, a Gaussian blur with a kernel size of 3x3 was applied. The GrayScale method converts the image to gray scale. Below in Figure 3, the appearance of several different signature samples from the dataset is shown after applying each of the preprocessing steps used in the study individually.

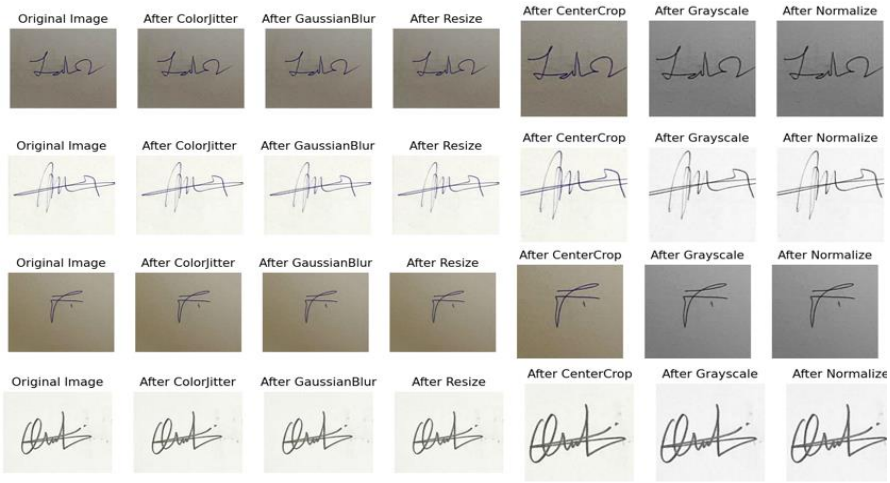

Figure 3. Preprocessing Steps Applied to Several Different Signature Samples from the Dataset

2.4. Classification

In this study, a signature classification application was implemented using advanced DL architectures. CNN architectures were selected for this task because of their superior capabilities in image processing. The CNN architectures used are GoogLeNet [18], MobileNet-V3 Large [19], Inception-V3 [20], ResNet50 [21] and EfficientNet-B0 [22].

In the approach proposed for the study, the data set was divided into train, validation and test. By using a validation set, the possibility of overfitting a model that performs well on the training data and poorly on the unseen data is avoided. The separation of train, validation and test provides an unbiased assessment of the performance of the final model, ensuring that the success of the model is not due to overfitting in the validation set. The CNN architectures used are pre-trained on the ImageNet dataset. ImageNet is a massive dataset ctable3ontaining over 14 million images (Deng et al., 2010). These models can be repurposed for other tasks without requiring training from scratch, as they have learned a rich set of features from millions of images (Zhuang et al., 2019). This significantly shortens the training time, provides higher accuracy and performance, reduces the computational resources required for training, and offers better generalization capability, making them extremely valuable for practical applications (Zhuang et al., 2019). By fine-tuning the final layers of the models, they were better adapted to the signature classification task, resulting in higher accuracy and performance during the training process. Below in Table 2, the original and modified states of the final layer of each model and their total number of parameters are shown. In the implementation, the highest number of parameters is found in the Inception-V3 model, with 27,481,128 parameters, while the MobileNet-V3-Large model has the lowest number of parameters, with 4,714,196. Detailed information about the layers was accessed using the 'torchscan' and 'torchstat' libraries provided by PyTorch.

Table 1. Modifications and total number of parameters applied to the final layers of the classification models before and after fine-tuning

	The original state of the last layer	**Modified last layer**	**Total parameters**
EfficientNet- B0	[0]: Linear (in =1280, out = 1000)	[0]: Dropout(p=0.2) [1]: Linear (in =1280, out =1024 [2]: ReLU () [3]: Linear (in =1024, out =420)	6,185,248

GoogLeNet	[0]: Linear (in =2048, out =1000	[0]: Dropout(p=0.2) [1]: Linear (in =2048, out =1024 [2]: ReLU () [3]: Linear (in =1024, out =420)	5,288,548
MobileNet-V3-Large	[0]: First Linear (in = 960, out = 1280) [1]: Hardswish [2]: Dropout (p = 0.2) [3]: Last Linear (in = 1280, out = 1000)	[0]: Dropout(p=0.2) [1]: Linear (in =1280, out =1024 [2]: ReLU () [3]: Linear (in =1024, out =420)	4,714,196
Inception-V3	[0]: Linear (in =2048, out =1000)	[0]: Dropout(p=0.2) [1]: Linear (in =2048, out =1024 [2]: ReLU () [3]: Linear (in =1024, out =420)	27,481,128
ResNet50	[0]: Linear (in =2048, out =1000)	[0]: Dropout(p=0.2) [1]: Linear (in =2048, out =1024 [2]: ReLU () [3]: Linear (in =1024, out =420)	26,036,708

EfficientNet-B0

EfficientNet-B0 is a member of the EfficientNet family of architectures (B0, B1, B2, ..., B7, B8), developed to provide a more fluid and effective approach to scaling DL models (Tan & Le, 2019). This model is distinguished by its use of a systematic method called Compound Scaling, which scales CNNs in a balanced way across all dimensions. It emerged from the idea that balancing the depth, width, and input resolution of a network can provide better efficiency and effectiveness. The core components of the EfficientNet-B0 model are MBConv Blocks, which are previously known as bottlenecks in MobileNetV2. These blocks include expansion and squeeze layers that recalibrate feature maps, further enhancing performance (Sandler et al., 2018). In this study, the use of the EfficientNet-B0 model was preferred based on its balance of time and performance efficiency. The schematic diagram of the fine-tuned EfficientNet-B0 model architecture used in this study is shown in Figure 4. a.

GoogLeNet

Inception V1, also known as GoogLeNet, is a deep convolutional neural network developed by Google (Szegedy et al., 2014). The distinguishing feature of GoogLeNet is the Inception module. This module includes multiple convolutions of different sizes (1x1, 3x3, and 5x5) and a 3x3 max pooling operation, all performed in parallel. The outputs from these operations are concatenated to form the final output of the module. In this study, GoogLeNet achieved effective classification with an average of only 5.2 million parameters. For this study, the GoogLeNet model, previously trained on the ImageNet dataset, was fine-tuned with the modifications shown in Table 1. The schematic diagram of the fine-tuned GoogLeNet model architecture used in this study is shown in Figure 4. b.

Inception-V3

Developed by researchers at Google, Inception-V3 is designed to provide high accuracy with efficient computation by optimizing the structure of convolutional networks (Szegedy et al., 2015). Inception-V3 introduces the idea of factorizing convolutions into smaller operations to reduce computational load while maintaining or enhancing the network's ability to capture critical features. Inception-V3 also employs Auxiliary Classifiers placed at intermediate points in the network during training. These classifiers add additional regularization, helping to prevent overfitting. While the outputs of these classifiers are not used during inference, they play a crucial role during the training phase. A schematic diagram of the fine-tuned Inception-V3 model architecture used in this study is shown in Figure 4. c.

MobileNet-V3-Large

MobileNetV3-Large is part of the MobileNet family of neural network models. The main purpose of MobileNets is to design a lightweight CNN for efficient performance on mobile and edge devices with lower capacity than computers (Sandler et al., 2018). MobileNet V1 has a structure that uses width and resolution multipliers. MobileNet V2, in addition to the structure of the MobileNet V1 model, used inverted residual structures that provide more efficient memory usage (Sandler et al., 2018).

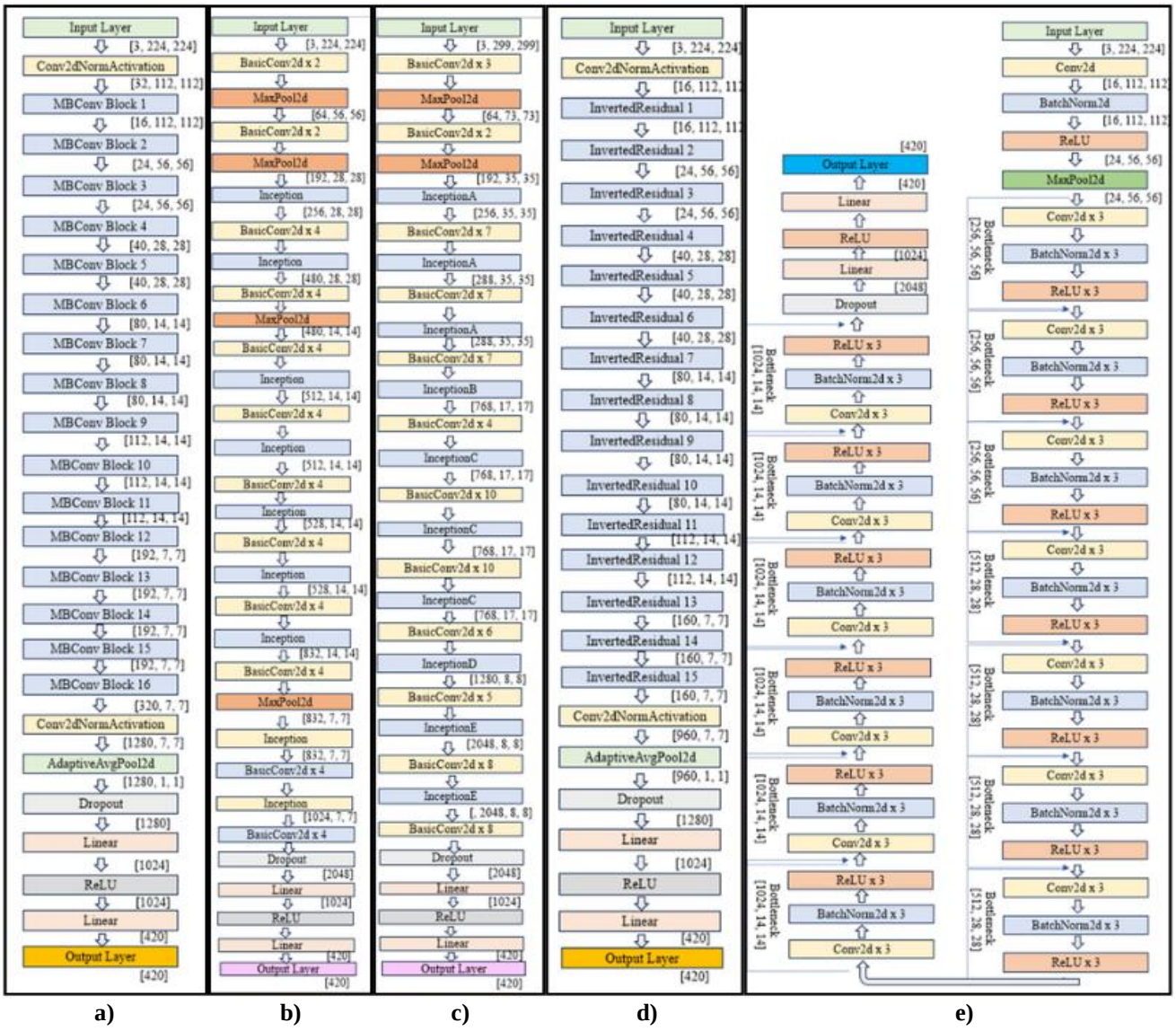

Figure 4. Schematic diagram of the fine-tuned a) EfficientNet-B0, b) GoogLeNet, c) Inception-V3, d) MobileNetV3-Large, e) ResNet50 model architecture used in this study

MobileNet V3 is a more optimized and improved version of the architectures (Howard et al., 2019). The efficiency of the network is increased by removing complex layers. With approximately 4.7 million parameters, MobileNetV3-Large is both lightweight and fast. A schematic diagram of the fine-tuned MobileNetV3-Large model architecture used in this study is shown in Figure 4.d.

ResNet50

The Residual Network with 50 layers, abbreviated as ResNet-50, is a CNN architecture widely used in image processing tasks, including signature recognition. Developed by Microsoft Research, ResNet-50 has significantly impacted the field of computer vision due to its innovative design that addresses the vanishing gradient problem often encountered in deep networks (He, Zhang, & Ren, 2015). The core idea behind ResNet is the introduction of residual learning. The convolutional layers of ResNet-50 effectively extract hierarchical features from signature images, ranging from simple edges and textures in the initial layers to more complex patterns and shapes in the deeper layers.

The ResNet50 architecture, previously trained on the ImageNet dataset, was used for this study. It accepts input images of 224x224 pixels. A schematic diagram of the fine-tuned ResNet50 model architecture used in this study is shown in Figure 4.e.

2.5. Performance Metrics

Performance metrics provide a comprehensive evaluation of the effectiveness of a classification model [23]. The performance metrics used in this study are Accuracy, Precision, Recall, F1-Score.

Accuracy is a fundamental metric for evaluating classification models and refers to the proportion of correctly predicted samples (both true positives and true negatives) in the total samples. It is calculated as shown in Equation 1.

$$Accuracy(\%) = 100 * \frac{\sum Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions} \quad (1)$$

Precision is a metric that quantifies the number of accurate positive predictions made by the model relative to the total number of positive predictions. It is calculated as shown in Equation 2.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (2)$$

Recall is also known as precision or true positive rate. It measures the proportion of true positive samples that the model correctly identifies. It is calculated as shown in Equation 3.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (3)$$

The F1-Score is a widely used metric in ML and statistics that combines precision and recall into a single measure. It is particularly useful when you need to balance the trade-off between these two metrics. It is calculated as shown in Equation 4 below:

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (4)$$

3. Results and Discussion

In this section, the signature dataset has been classified in several experiments and the results are presented. All tests were performed with Pytorch. PyTorch is an open-source ML library written in Python and developed by Facebook's Artificial Intelligence Research Lab (FAIR). It is widely used to develop and train DL models due to its flexibility, ease of use and dynamic computational graph feature. The experiments were conducted on a computer with a 13th Gen Intel(R) Core (TM) i9-13900HX processor, 64 GB RAM, and an Nvidia GeForce RTX 4060 graphics card. In the implemented application, the model was evaluated with three different train-validation and test ratios. In the first scenario, the dataset was divided into 20% test and 80% (train + validation). In the second scenario, the dataset was divided into 30% test and 70% (train + validation). In the third scenario, the dataset was divided into 40% test and 60% (train + validation). The dataset was evaluated using five different CNN models: EfficientNet-B0, ResNet50, MobileNet-V3-Large, Inception-V3, and GoogLeNet. The parameters and methods determined for the classification process were optimized in detail. Firstly, the Batch Size, which determines the amount of data to be used in each epoch, was selected as 32. This value ensures that the model receives sufficient information during training while balancing memory usage. For optimization, the AdamW algorithm was chosen. AdamW effectively regularizes weight decay, enhancing the model's overall performance. AdamW improves the overall performance of the model by regulating the weight decay more efficiently. Learning Rate was initially set as 0.001. However, in order for the model to have a more stable and efficient learning process, it was considered to decrease the learning rate gradually. For this purpose, Step_size was set to 3 and Gamma to 0.8. These parameters ensure that the learning rate is reduced every 3 epochs by multiplying the Learning Rate by a certain coefficient, Gamma. Thus, the Learning Rate is updated as shown in equation 5:

$$LR_{new} = LR_{old} \times (gamma)^{floor(epoch/step\_size)} \quad (5)$$

This equation allows the model to learn rapidly at first, but then to progress in smaller steps over time and to get closer to the result. All tests performed were run for 50 epochs. This allows the model to complete the learning process on the data with a sufficient number of iterations. Cross-entropy was chosen as the loss function used in the study. A Label Smoothing value of 0.11 was chosen. Label Smoothing is a technique used to prevent overconfident predictions in classification problems. After each train and validation step, the tests performed using the test dataset and the corresponding model parameters were copied to memory. This is vital for monitoring the generalization ability and actual performance of the model. In Table 2 and Table 3, it is seen that Inception-V3 and GoogLeNet models have the highest test accuracy (98.77%) at 20% test rate. The result values of GoogLeNet for Precision, Recall and F1-Masure were found to be 0.99 for all three criteria.

However, in Inception-V3, Precision and F1-Score results were 0.98 and Recall result was 0.99 as in GoogLeNet.

Table 2. Test results of Inception-V3 at different Train-Validation-Test ratios

Train-Test Ratio	Test Accuracy (%)	Precision	Recall	F1-Score
20% Testing, 80% (Train +Validation)	98.77	0.98	0.99	0.98
30% Testing, 70% (Train +Validation)	98.17	0.98	0.98	0.98
40% Testing, 60% (Train +Validation)	97.64	0.98	0.98	0.98

Table 3. Test results of GoogLeNet at different Train-Validation-Test ratios

Train-Test Ratio	Test Accuracy (%)	Precision	Recall	F1-Score
20% Testing, 80% (Train +Validation)	98.77	0.99	0.99	0.99
30% Testing, 70% (Train +Validation)	98.36	0.99	0.98	0.98
40% Testing, 60% (Train +Validation)	97.30	0.97	0.97	0.97

Figure 5 presents a comparative graphical representation of the accuracy performance of the five different models used in the study at varying test ratios. The accuracy of GoogLeNet at a 30% test ratio dropped from 98.77% to 98.36%. In contrast, Inception-V3 showed a more significant decrease to 98.17% compared to GoogLeNet at the same 30% test ratio. At a 40% test ratio, GoogLeNet' s accuracy fell to 97.30%, whereas Inception-V3 experienced a lesser decline to 97.64%. As the test ratio increased, a performance drop was observed in both models. The second most successful model, EfficientNet-B0, achieved the highest test accuracy of 98.65% at a 20% test ratio. For EfficientNet-B0, the Precision and Recall values were both determined to be 0.99, while the F1-Score was 0.98. Performance declined slightly in the other two scenarios, with an accuracy of 98.25% at a 30% test ratio and 97.76% at a 40% test ratio.

Table 4. Test results of EfficientNet-B0 at different Train-Validation-Test ratios

Train-Test Ratio	Test Accuracy (%)	Precision	Recall	F1-Score
20% Testing, 80% (Train +Validation)	98.65	0.99	0.99	0.98
30% Testing, 70% (Train +Validation)	98.25	0.98	0.98	0.98
40% Testing, 60% (Train +Validation)	97.76	0.98	0.98	0.98

The third most successful model, ResNet50, had the highest test accuracy (98.10%) at 20% test rate (Table 5). The result values of ResNet50 for Precision, Recall and F1-Score were found to be 0.98 for all three criteria. In the other two scenarios, with 30% and 40% test ratios, a slight decline in performance was observed. At the 30% test rate, the accuracy value was 97.96%, while at the 40% test rate it showed a greater decrease (2.30%) compared to the other four models and was obtained as 95.26%. It achieved the highest test accuracy of 97.14% at a 20% test ratio.

Table 5. Test results of ResNet50 at different Train-Validation-Test ratios

Train-Test Ratio	Test Accuracy (%)	Precision	Recall	F1-Score
20% Testing, 80% (Train +Validation)	98.10	0.98	0.98	0.98
30% Testing, 70% (Train +Validation)	97.96	0.98	0.98	0.98
40% Testing, 60% (Train +Validation)	95.26	0.95	0.95	0.95

The result values of MobileNet-V3-Large for Precision, Recall and F1-Score were 0.97 for all three criteria (Table 6). In the other two scenarios, i.e. 30% and 40% test rates, it was observed that the performance decreased slightly. While the accuracy value was 96.61% at 30% test rate, the accuracy value decreased to 95.14% at 40% test rate.

Table 6. Test results of MobileNet-V3-Large at different Train-Validation-Test ratios

Train-Test Ratio	Test Accuracy (%)	Precision	Recall	F1-Score
20% Testing, 80% (Train +Validation)	97.14	0.97	0.97	0.97
30% Testing, 70% (Train +Validation)	96.61	0.97	0.97	0.96
40% Testing, 60% (Train +Validation)	95.14	0.96	0.96	0.95

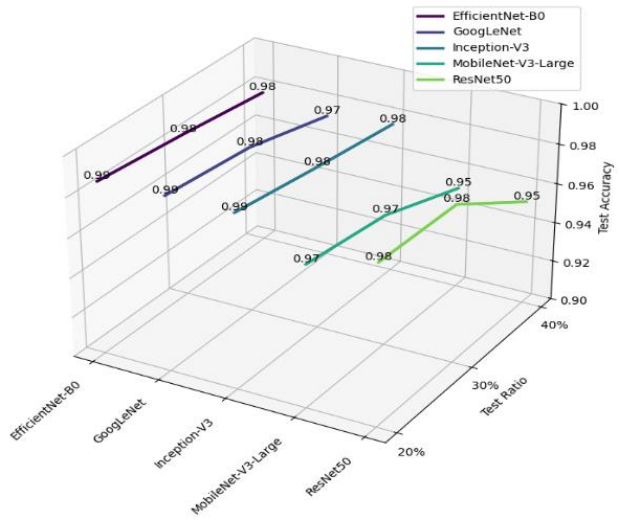

Figure 5. Classifier test accuracies at different Test Ratios for dataset

Figure 6 shows the Training and Validation accuracy graphs of the models over 50 epochs on the dataset divided with a 20% test ratio. We observe a rapid increase in accuracy during the early epochs for all models, indicating efficient learning. Inception-V3 and GoogLeNet show the best performance with minimal gaps between training and validation accuracy, suggesting good generalization and minimal overfitting, highlighting their stable and robust performance. This observation is further supported by their highest test performance as shown in Table 2 and Table 3. EfficientNet-B0 and ResNet50 also show no significant gaps between training and validation accuracy, indicating the absence of significant overfitting. Conversely, the MobileNet-V3-Large model, which is less successful compared to the other four models, shows a relatively larger gap between training and validation accuracy, suggesting potential overfitting.

The dataset used in our study was previously has only been utilized in a single study in the literature [16]. In their signature recognition application, they employed Transfer Learning for feature extraction, followed by classification using various ML methods. The researchers primarily focused on the feature extraction phase. As seen in Table 3, the highest accuracy achieved in their application was 97.7% using the SVM method. The other ML methods used included KNN, Decision Tree (DT), Linear Discriminant Analysis (LDA), and Naïve Bayes. They obtained accuracy rates of 92.54% with LDA, 90.28% with KNN and DT, and 82.50% with Naïve Bayes. In this study, instead of ML techniques, we fine-tuned five different pre-trained DL models, saving time and cost while enhancing performance. The results of the EfficientNet-B0, ResNet50, Inception-V3, and GoogLeNet models surpassed those obtained in the study by Özyurt et al. Only the MobileNet-V3-Large model used in this study performed slightly less successfully than the SVM result from their research but outperformed the other models (KNN, DT, LDA, Naïve Bayes) in terms of accuracy Compared to our study, the direct use of a pre-trained DL model, as well as fine-tuning the model, provided both time and cost savings and higher accuracy values. The data preprocessing methods used in our study are largely similar to those in other studies in the literature. However, additional methods such as ColorJitter and GaussianBlur were used to enrich the data. The use of these methods, in addition to commonly used techniques like resizing, grayscale, and normalization, contributed to the improved performance of the model. Table 3 shows that DL methods, especially CNN models, have been widely used in signature recognition applications in the literature in recent years. Researchers such as Calik et al.[9], Jampour et al. [24], Kancharla et al. [25], Noor et al. [26], Pokharel et al. [17] and Tarek and Atia [13] have achieved high accuracy values using CNN models in their studies. With the use of DL models, the need for the feature extraction stage has decreased and accuracy rates have increased. For example, Kancharla et al., integrated CNN and Jampour et al. integrated CapsNet and ResNet models and achieved very high accuracy values such as 100 %.

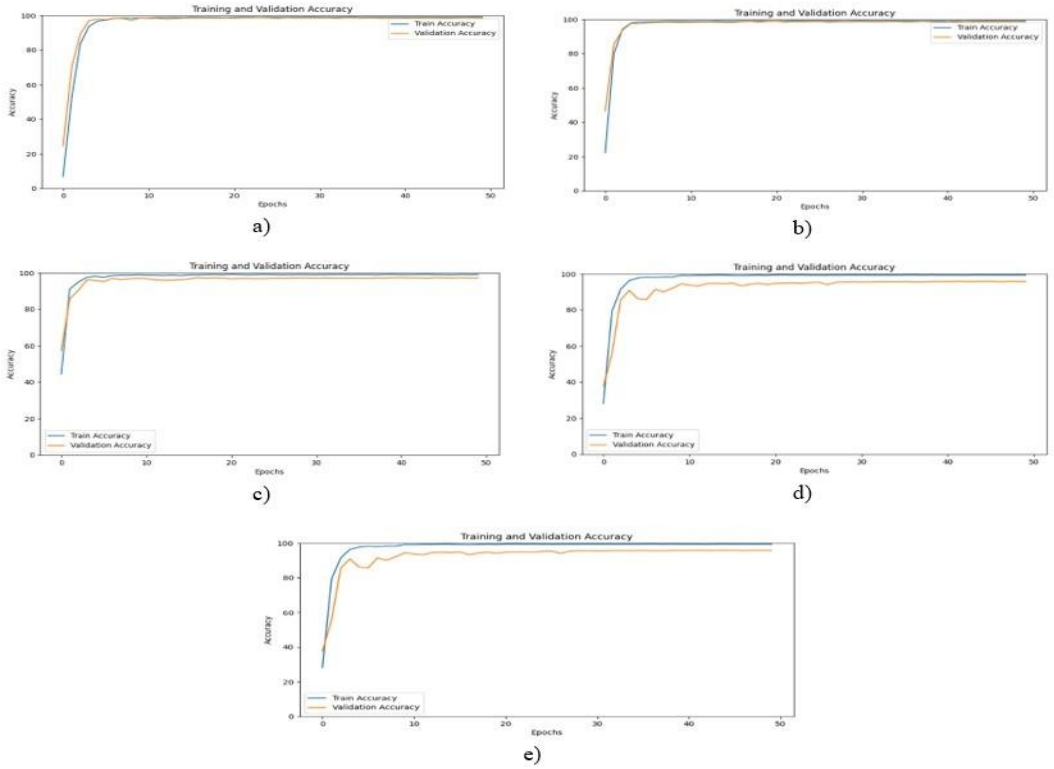

Figure 6. Training and validation accuracy on the signature dataset for a) Inception-V3, b) GoogleNet, c) EfficientNet-B0 and d) MobileNet-V3-Large e) ResNet50

The dataset used in this study has a considerably larger data volume compared to other studies in the literature. Large datasets improve the overall performance of DL models by allowing them to learn more features (Lecun et al., 2015). This is a crucial factor behind the high accuracy values obtained in our study. In the literature, datasets created by researchers, which are not widely available, typically consist of a relatively smaller number of signature images [13], [27], [28]. The extensive size of the dataset in our study allowed the DL models to learn more features and improve their performance [29].

Table 7. Literature summary of offline signature recognition applications using ML and DL

Authors (year)	**Database/Number of people / Number of samples**	**Data pre-processing**	**Feature extraction**	**ML techniques**	**Signature application**	**Accuracy**
[30]	GPDS/2106 signature	Boundary extraction	Modified Direction Feature (MDF) centroid, surface area, length, skew	RBP Neural Network (Resilient Backpropagation), RBF Network (Radial Basis Function)	Classification	91.21%, 88%
[31]	Self-built/100/3000 signature	Conversion to Binary Images, Normalization, Reshaping	Global Features Extraction (Signature Height, Image Area, Pure Width and Pure Height)	ANN, MLP	Classification	75.20%
[9]	SUSIG-Visual/100/2000 signature	Resizing, Grayscale conversion	-	CNN	Classification	92%

[25]	SigComp 2011 Dutch/10/240 signature, SigComp 2011 Chinese/10/240 signature, UTsig Farsça/10/240 signature	Grayscale Conversion, Finalization, Erosion, Dilation, Image Resizing	-	CNN	Classification	100%, 85.11%, 96.29%
[11]	CEDAR/55/2640 signature	Image Resizing, Otsu thresholding, binarizing	-	CapsNet	Verification	98.8%
[12]	CEDAR/55/2640 signature, GPDS-100/400/216,000 signature, MCYT/75/2250	Image Resizing, Otsu thresholding, binarizing	-	CapsNet	Verification	97.96%, 94.89%, 91%
[26]	Self-built/4/10 signature Self-built/10/240 signature Self-built/54/648 signature Self-built/33/165 signature	Grayscale Conversion, Normalization, Inverting, Image Resizing	-	CNN	Classification	99%, 89.1%, 80.8%, 72.7%
[15]	GPDS Senthtetic/4000/216000 signatures, MCYT-75/75/2250 signatures, UTSig/115/7935 signatures, FUM-PHSD/20/600 signatures	Binarization (Otsu's algorithm), Noise removing, Gussian filter, Normalization	SigNet, SigNet-F, VGG16, VGG19, InceptionV3, ResNet50	SVM	Classification	99.7%, 100%, 98.71%, 100%
[17]	Self-built/25/2,500 signature	Cropping image, Scaling dimension,	GoogLeNet	GoogLeNet	Classification	95,2%
[27]	Self-built/30/450 signature	Decoding, Resizing, Padding	CNN	Siamese Network	Classification	84%
[24]	CEDAR/55/2640 signature, MCYT/75/2250 signature UTSig/115/8280 signature	Centralized padding, Image Resizing	-	CapsNet, ResNet	Classification	100%, 99.66%, 99.38%
[14]	MCYT-75/75/2250 signatures, BHSig260(Bengali)/100/5400 signatures, BHSig260(Hindi)/160/8640 signatures, MCYT-75/75/2250 signatures	Image Resizing	GoogLeNet	SVM	Classification	96.5%, 95.7%, 93%
[32]	4NSigComp2012/5/300, SigComp2011/10/576, BHSig260(Bengali)/100/5400 signatures, BHSig260(Hindi)/160/8640 signatures,	Bounding Box Method, Resizing, Binary Thresholding, Inversion, Normalization	-	Siyam Ağlar	Verification	93.23%, 92.11%, 89.78%, 91.3%
[28]	Self-built/20/200 signature	Image acquisition, Grayscaling, Segmentation, Area filtering, Object bounding areas.	Local Binary Pattern (LBP), Uniform LBP, LBP 8 rotation,	Learning Vector Quantization (LVQ)	Classification	90%

			uniform LBP 8 rotation			
[13]	Self-built/30/300 signature Self-built/140 images	Grayscale Conversion, Resizing	-	VGG-16, ResNet50, Inception-v3, Xception, modified CNN	Classification	75%, 86%, 77%, 70%, 75%
[16]	Self-built/420/12,600 signature	Cropping, Resizing	MobileNetV2, Neighborhood Component Analysis (NCA), Chi-Square (Chi2), Mutual Information (MI)	SVM, KNN, Decision Tree (DT), Linear Discriminant Analysis (LDA), Naïve Bayes	Classification	97.7%, 90.28%, 90.28%, 92.54%, 82.50%
This Study	Built by [16] /420/12,600	Resize, CenterCrop, ColorJittter, GaussianBlur, Grayscale, Normalize	-	EfficientNet-B0, ResNet50, Inception-V3, GoogLeNet, MobileNet-V3-Large	Classification	98.65%, 98.10%, 98.77%, 98.77%, 97.14%

4. Conclusion and Suggestions

The proposed methodology offers a robust and efficient approach for signature recognition, with potential applications in various fields such as banking, law, and security systems. However, since the focus of this study is on offline handwritten signatures, further research is needed for other types of signatures, such as digital signatures. For future studies, the application of larger and more diverse datasets, as well as the use of next-generation DL techniques like transformers and attention mechanisms, could allow for the capture of more complex and variable signature features.

Additionally, incorporating dynamic information from the signing process (e.g., pressure, speed) along with signature images could lead to the development of multimodal learning models. This would further enhance the reliability of signature verification systems.

In conclusion, this study demonstrates the potential of DL in the field of signature recognition. Future research, in parallel with technological advancements, will contribute to the emergence of more advanced and reliable signature recognition systems.

Conflict of Interest Statement

There is no conflict of interest between the authors.

Statement of Research and Publication Ethics

The study is complied with research and publication ethics

References

[1] A. K. Jain, K. Nandakumar, and A. Ross, "50 years of biometric research: Accomplishments, challenges, and opportunities," *Pattern Recognit Lett*, vol. 79, pp. 80–105, Aug. 2016, doi: 10.1016/J.PATREC.2015.12.013.

[2] M. A. Ferrer, J. F. Vargas, A. Morales, and A. Ordóñez, "Robustness of offline signature verification based on gray level features," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 966–977, 2012, doi: 10.1109/TIFS.2012.2190281.

[3] N. S. Kamel, S. Sayeed, and G. A. Ellis, "Glove-Based Approach to Online Signature Verification," *IEEE Trans Pattern Anal Mach Intell*, vol. 30, no. 6, pp. 1109–1113, Jun. 2008, doi: 10.1109/TPAMI.2008.32.

[4] Z. Wang, M. Muhammat, N. Yadikar, A. Aysa, and K. Ubul, "Advances in Offline Handwritten Signature Recognition Research: A Review," *IEEE Access*, vol. 11, pp. 120222–120236, 2023, doi: 10.1109/ACCESS.2023.3326471.

[5] A. Rıza Yılmaz et al., "İmza Tanıma Uygulaması için Çok Katmanlı Algılayıcıların Diferansiyel Gelişim Algoritması ile Eğitimi Training Multilayer Perceptron Using Differential Evolution Algorithm for Signature Recognition Application," 2013.

[6] R. Sabourin and Drouhard Jean-Pierre, "Offline signature verification using directional pdf and neural networks," *International Conference on Pattern Recognition*, 1994.

[7] M. A. Ismail and S. Gad, "Off-line arabic signature recognition and verification," *Pattern Recognit*, vol. 33, no. 10, pp. 1727–1740, Oct. 2000, doi: 10.1016/S0031-3203(99)00047-3.

[8] M. R. Deore and S. M. Handore, "Offline signature recognition: Artificial neural network approach," in *2015 International Conference on Communications and Signal Processing (ICCSP)*, 2015, vol. 2, pp. 1708–1712..

[9] NN. Calik, O. C. Kurban, A. R. Yilmaz, L. D. Ata, and T. Yildirim, "Signature recognition application based on deep learning," *in 2017 25th Signal Processing and Communications Applications Conference (SIU)*, 2017, vol. 20, pp. 1–4..

[10] MM. K. Kalera, S. Srihari, and A. Xu, "Offline signature verification and identification using distance statistics," *Intern. J. Pattern Recognit. Artif. Intell*., vol. 18, no. 07, pp. 1339–1360, 2004.

[11] D. Gumusbas and T. Yildirim, "Offline signature identification and verification using capsule network," in *2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*, 2019.

[12] D. Gumusbas and T. Yildirim, "Offline signature identification and verification based on capsule representations," *Cybernetics and Information Technologies*, vol. 20, no. 5, pp. 60–67, Dec. 2020, doi: 10.2478/CAIT-2020-0040.

[13] OO. Tarek and A. Atia, "Forensic handwritten signature identification using deep learning," in *2022 IEEE 9th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)*, 2022, vol. 39, pp. 185–190..

[14] A. Jain, S. K. Singh, and K. Pratap Singh, "Multi-task learning using GNet features and SVM classifier for signature identification," *IET Biom*, vol. 10, no. 2, pp. 117–126, Mar. 2021, doi: 10.1049/BME2.12007.

[15] A. Foroozandeh, A. Askari Hemmat, and H. Rabbani, "Offline handwritten signature verification and recognition based on deep transfer learning," in *2020 International Conference on Machine Vision and Image Processing (MVIP)*, 2020.

[16] F. Özyurt, J. Majidpour, T. A. Rashid, and C. Koç, "Offline Handwriting Signature Verification: A Transfer Learning and Feature Selection Approach," *Traitement du Signal* vol.40, no.6, 2024.

[17] S. Pokharel, S. Giri, and S. Shakya, "Deep Learning Based Handwritten Signature Recognition," *NCE Journal of Scince and Engineering*, vol.1, no.1, pp. 21-24, 2020.

[18] C. Szegedy et al., "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[19] A. Howard et al., "Searching for MobileNetV3," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[20] C.C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR*), 2016.

[21] K.K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[22] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," 36th International Conference on Machine Learning, ICML 2019, vol. 2019-June, pp. 10691–10700, May 2019, Accessed: Jul. 25, 2024. [Online]. Available: https://arxiv.org/abs/1905.11946v5

[23] OO. El Melhaoui, S. Said, A. Benlghazi, and S. Elouaham, "Improved signature recognition system based on statistical features and fuzzy logic," *e-Prime - Advances in Electrical Engineering, Electronics and Energy*, vol. 8, no. 100505, p. 100505, 2024.

[24] M. Jampour, S. Abbaasi, and M. Javidi, "CapsNet regularization and its conjugation with ResNet for signature identification," *Pattern Recognit*, vol. 120, p. 107851, Dec. 2021, doi: 10.1016/J.PATCOG.2021.107851.

[25] K. Kancharla, V. Kamble, and M. Kapoor, "Handwritten signature recognition: A convolutional neural network approach," in *2018 International Conference on Advanced Computation and Telecommunication (ICACAT)*, 2018, vol. 2709, pp. 1–5..

[26] F. Noor, A. E. Mohamed, F. A. S. Ahmed, and S. K. Taha, "Offline handwritten signature recognition using convolutional neural network approach," *in 2020 International Conference on Computing, Networking, Telecommunications & Engineering Sciences Applications (CoNTESA)*, 2020, pp. 51–57.

[27] S. Mshir and M. Kaya, "Signature Recognition Using Machine Learning," in *2020 8th International Symposium on Digital Forensics and Security (ISDFS)*, 2020.

[28] E. K. D. Kette, D. R. Sina, and B. S. Djahi, "Digital image processing: Offline handwritten signature identification using local binary pattern and rotational invariance local binary pattern with learning vector quantization," *J. Phys. Conf. Ser.*, vol. 2017, no. 1, p. 012011, 2021.

[29] YY. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[30] S. Armand, M. Blumenstein, and V. Muthukkumarasamy, "Off-line signature verification using the enhanced modified direction feature and neural-based classification," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, 2006, pp. 684–691.

[31] A.A. A. M. Abushariah, T. S. Gunawan, J. Chebil, and M. A. M. Abushariah, "Automatic person identification system using handwritten signatures," in *2012 International Conference on Computer and Communication Engineering (ICCCE)*, 2012, pp. 560–565.

[32] M. V. Arısoy, "Signature verification using siamese neural network one-shot learning," *International Journal of Engineering and Innovative Research*, vol. 3, no. 3, pp. 248–260, Sep. 2021, doi: 10.47933/IJEIR.972796.