Signature Verification using ResNet-50 and Transfer Learning

1st Deepali Narwade1*, Latika Desai2nd, Suresh Mali3rd
1, 2, 3, Department of AI&DS, Dr. D. Y. Patil Institution of Technology, Savitribai Phule Pune University, Pune (MS), India,

Abstract: Recent years have seen a rise in the use of deep convolutional neural networks in a number of computer vision and pattern recognition applications. The offline handwritten signature, which is still a challenging undertaking, is one of the most important biometrics used in banking systems, administrative software, and financial applications. This paper aims to explore existing convolutional neural network-based signature verification methods and evaluate the performance of well-known deep convolutional neural networks as feature extractors in transfer learning-based signature verification. ResNet-50, a pre-trained model, is used for this. It is a Convolutional Neural Network (CNN), a class of deep neural networks that is frequently used to analyze visual data, pre-trained deep learning model for image classification.

In order to keep the network’s information flow moving and keep gradients from dissipating, ResNet-50 adds residual connections to the network. The residual link is a short cut that enables data to pass through one or more network layers and arrive straight at the output. ResNet-50 hence reduces the number of errors while increasing the effectiveness of deep neural networks with more neural layers.

Index Terms- Signature Verification, CNN, ResNet-50, Transfer Learning.

1. INTRODUCTION

Security systems employ biometrics to identify or confirm the validity every day all across the world. One of the most crucial biometrics used in banking systems, administrative, and financial applications is the handwritten signature.

There are two ways that a signature processing system might receive a signature: offline and online. Processing offline handwritten signatures is more challenging than processing them online because there is less information available. Three types of forgeries are dealt with by a Handwritten Signature Verification (HSV) system: simple, random, and skilled forgeries. A Handwritten Signature Verification/Recognition (HSV/R) system may employ either Writer-Dependent (WD, special learning) or Writer-Independent (WI, general learning) learning methods [1]. When it comes to WI, learning is based on a sizable population of signature samples that are connected to every person in the dataset, whereas when it comes to WD, learning is based on the individual signature samples from each person. Although WD learning produces strong results, the complexity and cost of the system rise since a classifier must be run for each user who is added. The three main processes in every HSV/R system are preprocessing, feature extraction, and classification. There are two types of feature extraction methods: learning feature representation and handmade feature extractors. The verification of signatures frequently uses handcrafted features.

A computer will determine which class an image belongs to when doing an image classification task. Prior to deep learning taking off, it is impossible to accomplish tasks like image classification at a human level. This is due to the machine learning model’s inability to learn an image’s neighbor information. Only pixel-level information is provided to the model.
Convolutional Neural Network (CNN) is a model that can execute picture classification tasks at a human level thanks to the strength of deep learning. A deep learning model called CNN uses an image to teach it how to represent something. Without human intervention, this model is capable of learning from low to high level features.

The model picks up knowledge beyond only pixel-level information. The model also picks up neighbor information from an image using a convolutional approach. Convolution will multiply the set of pixels in a region and add them up to a value to aggregate neighborhood information. These characteristics will be used to assign the image to a certain class.

Deep learning requires a lot of data, even though it can perform at a human level. What happens if we lack them? We can make use of the idea of transfer learning.

We employ a model that has been developed on a large-scale data problem called transfer learning. As a result, we exclusively teach them through model optimization. The model will train quickly, which is a plus for us.

Convolutional Neural Network (CNN) models and deep learning have emerged with the development of artificial intelligence, and the outcomes of autonomous processing systems have undergone a revolution. In order to successfully learn a decent feature representation directly from the raw data, learning feature representation uses deep CNNs. Transfer learning, often known as knowledge transfer, is one of the most effective methods in the field of CNN models. Through transfer learning, we can apply the capabilities of a CNN model from the original task to new tasks that are constrained by a large amount of data or a shortage of resources. Residual Network (ResNet-50) is a deep learning model used for computer vision applications. It is a Convolutional Neural Network (CNN) architecture designed to support hundreds or thousands of convolutional layers [1].

![Figure 1: Machine Learning and Transfer Learning.](image)

ResNet-50, short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks. This model was the winner of ImageNet challenge in 2015. The fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+ layers successfully. ResNet-50 is a pretrained Deep Learning model for image classification of the Convolutional Neural Network (CNN, or ConvNet), which is a class of deep neural networks, most commonly applied to analyzing visual imagery. In ResNet-50 models, all convolutional layers apply the same convolutional window of size \(3 \times 3\), the number of filters increases following the depth of networks, from 64 to 512 (for ResNet-18 and ResNet-34), from 64 to 2048 (for ResNet-50, ResNet-101, and ResNet-152). Residual Network (ResNet-50) architecture is a type of artificial neural network that allows the model to skip layers without affecting performance.

- **ResNet-50**

  Convolutional Neural Networks (CNN, or ConvNet), a kind of deep neural networks that is most frequently used to analyze visual vision, have a pretrained deep learning model called ResNet-50 for classifying images. ResNet-50, which has 50 layers, was trained using a million photos from the ImageNet database in 1000 different categories.

  The model also boasts over 23 million trainable parameters, indicating a deep architecture that improves image identification. In comparison to starting from scratch, where you would need to gather a lot of data and train the model yourself, using a pretrained model is a highly effective strategy. There are alternative pretrained deep models to utilize, such as AlexNet, GoogleNet, or VGG19,
of course, but the ResNet-50 is renowned for having outstanding generalization performance and lower error rates on recognition tasks, making it a handy tool to be aware of.

- **Architecture of ResNet-50**

  ResNet-50 stands for *Residual Network* and more specifically it is of a Residual Neural Network architecture. What characterizes a residual network is its identity connections. Identity connections takes the input directly to the end of each residual block, as shown below with the curved arrow:


Specifically, the ResNet-50 model consists of 5 stages each with a residual block. Each residual block has 3 layers with both 1*1 and 3*3 convolutions. The concept of residual blocks is quite simple. In traditional neural networks, each layer feeds into the next layer. In a network with residual blocks, each layer feeds into the next layer and directly into the layers about 2–3 hops away, called identity connections.

- **Transfer Learning**

  A deep learning technique called transfer learning involves training a neural network model on a problem that is comparable to the one that has to be solved. Reduced generalization error and a shortened training period are two advantages of transfer learning. The process of using a model that has been trained on one problem in some capacity on another related problem is known as transfer learning. For instance, while attempting to identify mangos, the skills used when learning to recognize oranges may be useful. Transfer learning with ResNet50 entails "fine-tuning" the pre-trained model on a fresh dataset.

  With ResNet50, transfer learning entails "fine-tuning" the pre-trained model on a fresh dataset. The pre-trained model's weights are changed during fine-tuning in order to better match the fresh data. The pre-trained weights from the ImageNet dataset are loaded into the ResNet50 model.

  Transfer learning is typically used to avoid having to train numerous machine learning models from scratch to fulfill comparable tasks, saving time and resources. As a means of increasing efficiency in machine learning tasks that demand a lot of resources, including image classification or natural language processing, to negate a lack of labelled training data held by an organisation, by using pre-trained models.

**II. LITERATURE REVIEW**

Traditional machine learning models were trained with handcrafted attributes, and categorisation accuracy is closely connected with these qualities. The main flaw in conventional models is thought to be this reliance [2]. Recently, academics have become increasingly interested in Deep Learning (DL) and CNNs, notably in the area of bioinformatics. The claimed results of the signature verification methods based on DL and CNN have greatly improved as compared to the handcrafted features. Traditional neural networks have an input layer, a hidden layer, and an output layer, as shown in Fig. 3(a).

As seen in Fig. 3(b), deep learning is learning with multiple hidden layers. A subset of deep learning models called CNNs was created for the first time in 1980. They contain several secret layers. The hidden layers commonly contain convolutional, pooling, Rectified Linear Unit (ReLU), and Fully Connected (FC) layers, as seen in this image [3].
The effectiveness of CNNs in computer vision and machine learning tasks motivated the researchers to utilize CNN in the HSV/R systems they demonstrated. Khalajzadeh et al. proposed the HSV approach using CNN; however, expert forgeries were not considered; only random forgeries were. This study used a Multilayer Perceptron (MLP) to classify the data. Using a private dataset of Persian signatures gathered by 22 writers, the verification rate of 99.86% was made public. Cozzens et al. obtained an 83% rate on the SigComp2011 signature dataset using a CNN model for HSV. Deep Multitask Metric Learning (DMML), a classification method for offline HSV, was introduced by Soleimani et al.

A deep CNN-based writer-independent feature learning method for offline HSV was presented by Hafemann et al. in 2016. This study found false rejection rates of 19.81% and false acceptance rates of 5.99% on the GPDS-160 signature dataset. Deep CNNs were also used in Hafemann et al.'s [4] analysis of characteristics found in offline HSV.

Deep CNNs are used in the Hafemann et al. [9] technique to directly learn representations from signature images. The following four datasets were used to evaluate the method's efficacy: GPDS-960, CEDAR, MCYT-75, and Brazilian (PUC-PR). The error rate for the GPDS-160 signature dataset was 1.72%.

Two pre-trained models were proposed in this work: SigNet and SigNet-F. As far as we know, these models are basically CNN models that have already been trained on certain datasets, and the research community can get their weights without charge. A select number of well-known offline signature datasets were used to create these models. The SigNet model was trained using only genuine signatures. The SigNet-F model was trained using both real signatures and a small number of well fabricated signatures.

The authors recently introduced some CNN models in their systems. Therefore, their results should be compared to the performance of certain benchmark CNN models. The authors are unable to compare their findings to those of the most recent CNN models because there isn't a study in the field of signature processing that does this yet. In order to overcome this issue, some well-known CNN models have been used, including SigNet, SigNet-F (previously trained on signature datasets), VGG16, VGG19, InceptionV3, and ResNet50 (trained on ImageNet). In this work, transfer learning has been used to evaluate the performance of these models. It is believed that the CNN model's structure is fixed, and forward propagation has been carried out from the first layer to the layer immediately preceding the last layer.

The feature layer's output is regarded as the feature vector for the source signature image. The pre-trained model performs feature extraction based on this procedure. To determine if the input signature is authentic or fake, the recovered feature vectors are utilized to train a support vector machine (SVM) with a Radial Basis Function (RBF) kernel as the classifier [10]. Tables I and II list the pre-trained models that are being considered, along with their depth, number of layers, feature layer, and feature vector size.
TABLE I: The pre-trained models, its depth, the number of layers, and image input size [1].

<table>
<thead>
<tr>
<th>Pre-trained model</th>
<th>Depth</th>
<th>Number of layers</th>
<th>Image input size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SigNet</td>
<td>7</td>
<td>13</td>
<td>150*220</td>
</tr>
<tr>
<td>SigNet-F</td>
<td>7</td>
<td>13</td>
<td>150*220</td>
</tr>
<tr>
<td>VGG16</td>
<td>16</td>
<td>41</td>
<td>256<em>256 (200</em>200 for FUM-PHSD)</td>
</tr>
<tr>
<td>VGG19</td>
<td>19</td>
<td>47</td>
<td>256<em>256 (200</em>200 for FUM-PHSD)</td>
</tr>
<tr>
<td>ResNet50</td>
<td>50</td>
<td>177</td>
<td>256<em>256 (200</em>200 for FUM-PHSD)</td>
</tr>
<tr>
<td>InceptionV3</td>
<td>48</td>
<td>316</td>
<td>256<em>256 (200</em>200 for FUM-PHSD)</td>
</tr>
</tbody>
</table>

TABLE II: The pre-trained models, the considered feature layer and the size of feature vector [1].

<table>
<thead>
<tr>
<th>Pre-trained model</th>
<th>Feature layer</th>
<th>Feature vector size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SigNet</td>
<td>Fully connected (FC7)</td>
<td>2048</td>
</tr>
<tr>
<td>SigNet-F</td>
<td>Fully connected (FC7)</td>
<td>2048</td>
</tr>
<tr>
<td>VGG16</td>
<td>Max Pooling2D</td>
<td>8<em>8</em>512 (6<em>6</em>512 for FUM-PHSD)</td>
</tr>
<tr>
<td>VGG19</td>
<td>Max Pooling2D</td>
<td>8<em>8</em>512 (6<em>6</em>512 for FUM-PHSD)</td>
</tr>
<tr>
<td>ResNet50</td>
<td>Average Pooling2D</td>
<td>2048</td>
</tr>
<tr>
<td>InceptionV3</td>
<td>Mixed10 (before Global Average Pooling)</td>
<td>6<em>6</em>2048 (4<em>4</em>512 for FUM-PHSD)</td>
</tr>
</tbody>
</table>

TABLE III: The statistics of four datasets used in this work[1].

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Users</th>
<th>Genuine</th>
<th>Forgery</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPDS synthetic</td>
<td>4000</td>
<td>24</td>
<td>30 (Simple)</td>
</tr>
<tr>
<td>MCYT-75</td>
<td>75</td>
<td>15</td>
<td>15 (Skilled)</td>
</tr>
<tr>
<td>UTSig</td>
<td>115</td>
<td>27</td>
<td>6 (Skilled)</td>
</tr>
<tr>
<td>FUM-PHSD</td>
<td>20</td>
<td>20</td>
<td>10 (Skilled)</td>
</tr>
</tbody>
</table>

It should be emphasized that a CNN model’s depth is determined by how many consecutive convolutional or fully connected layers there are between the input and output layers. Datasets and preprocessing, in first Four datasets have been taken into consideration in order to assess the effectiveness of transfer learning on the pre-trained models: the GPDS synthetic signature dataset [11], the MCYT-75 dataset [12], two benchmark Latin signature datasets, the UTSig (University of Tehran Persian Signature) dataset [13], a benchmark Persian signature dataset, and the FUM (Ferdowsi University of Mashhad) dataset [14]. In Table III, the statistics of these datasets are compared, along with several signature examples from the used datasets.

The following preprocessing tasks were carried out to improve the quality of the signature images: (i) converting the signature image into binary format using Otsu’s algorithm [1], (ii) inverting the image so that the background is zero-valued, (iii) removing the salt and pepper noise produced after binarization using a Gaussian filter, (iv) removing the white space surrounding the signature image and cropping the image, and (v) normalizing the size to All of the signature photos from the four datasets in Table III have undergone the preprocessing operations. Due to the poor quality of the signature photos in the FUM-PHSD dataset, the Gaussian filter did not work in this situation because it removed some of the signature image.

Fig.3 shows the output of the preprocessing tasks on a signature image from MCYT-75 dataset.
1. **RESEARCH METHODOLOGY**

   1. **Import Libraries**

The first step that we need to do is to import libraries. We need TensorFlow, NumPy, os, and pandas. If you don’t install the package yet, you can use the pip command to install the libraries.

   ```
   # ! pip install tensorflow==2.4.1
   # ! pip install pandas
   # ! pip install numpy
   
   import os
   import numpy as np
   import pandas as pd
   import tensorflow as tf
   ```

   2. **Preparing the Data**

After you load the libraries, the next step is to prepare our dataset. In this case, we will use a dataset called Food-5K. This dataset consists of 5000 images with two classes where the classes are food and non-food. Also, the data is already divided into training, validation, and a test set of data. The folder structure of our dataset looks like this,

   ![Food-5K Folder Structure](image)

As you can see above, each folder consists of images, where each image filename contains the class and the identifier of it. The identifier is divided by an underscore.

With that folder structure, we need to generate the dataframe with columns are the image filename and the label. The code for preparing the dataset.

The next step is to prepare an object to put the images into the model. We will use the ImageDataGenerator object from tf.keras.preprocessing.image library first.

With that object, we will generate image batches. Also, we can augment our image to largen the number of the dataset. Because we also augment those images, we also set parameters for the image augmentations method.

Also, because we use a dataframe as the information about the dataset, we will use the flow_from_dataframe method to generate batches and augment the images.

3. **Train the model**

After we generate the batches, now we can train the model with the transfer learning method. Because we use that method, we don’t need to implement CNN architecture from scratch. Instead, we will use the existing and already pretrained architecture. We will use ResNet-50 as the backbone for our new model. We will create the input and change the final linear layer of ResNet-50 with the new one based on the number of classes. Now let’s train the model. We will use the fit method for training it.

4. **Test the Model**

After we train the model, now let’s test the model on the test data. In addition, we need to add pillow library to load and resize the image and scikit-learn for calculating the model performance. We will use classification_report from the scikit-learn library to generate a report about model performance. Also, we will visualize the confusion matrix from it.
5. Save the Model

If you want to use the model for later use or deployment, you can save the model using the save method. If you want to load to load the model, you can use the load_model function.

Now you know how to implement transfer learning using TensorFlow.

II. RESULTS AND DISCUSSION

According to Table IV, the pre-trained models SigNet, VGG16, SigNet-F, ResNet50, VGG19, and InceptionV3 performed the best overall across all datasets. It should be noted that the models with similar architectures (Tables I and II) SigNet and ResNet50, SigNet-F and VGG16, VGG19 and InceptionV3 all exhibit comparable performance on datasets. It is only reasonable that the features obtained from the CNN models SigNet and SigNet-F, which have already been trained on signature datasets, are more discriminative and outperform those from other models.

TABLE IV: Obtained signature verification results in terms of EER (%) using transfer learning on the pre-trained models [1].

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Users</th>
<th>Genuine</th>
<th>Forgery</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPDS synthetic</td>
<td>4000</td>
<td>24</td>
<td>30 (Simple)</td>
</tr>
<tr>
<td>MCYT-75</td>
<td>75</td>
<td>15</td>
<td>15 (Skilled)</td>
</tr>
<tr>
<td>UTSig</td>
<td>115</td>
<td>27</td>
<td>6 (Skilled)</td>
</tr>
<tr>
<td>FUM-PHSD</td>
<td>20</td>
<td>20</td>
<td>10 (Skilled)</td>
</tr>
</tbody>
</table>

The lowest performance in Table IV, on the other hand, corresponds to VGG19 and InceptionV3, demonstrating that shallow learning models like SigNet and VGG16 can produce acceptable HSV outcomes without the requirement for highly deep neural networks like VGG19 and InceptionV3.

Using the Sony HD camera dataset, the SVM, KNN, and K-Means have accuracy ratings of 91%, 87%, and 89%, respectively. In both datasets, SVM produces good results [15]. The proposed order structure is made up of the pre-processing, segmentation, feature extraction, and classification steps. Each stage is completed utilizing a variety of machine learning and digital image processing approaches, with a Support Vector Machine [16] reaching 90 to 95% accuracy.
IV. CONCLUSION

The efficiency of transfer learning on the trained CNN models for signature verification has been evaluated in this study. The pre-trained models SigNet, SigNet-F, VGG16, VGG19, InceptionV3, and ResNet50 were used in Signature Verification systems to extract features, and the classifier SVM with RBF kernel was used to determine the query signature's identity in the end. Experimental results supported the performance of the pretrained models VGG16 and SigNet in HSV and HSR.

Future research will concentrate on optimizing CNN models to enhance HSV and HSR outcomes by tweaking the first, middle, and final layers of a pre-trained model's learning rates, adding or removing layers, or changing the type of layers. The results of data augmentation with exact modification in the future will also be assessed.

V. ACKNOWLEDGMENT

We are thankful to Department of AI & DS, Dr. D. Y. Patil Institute of Technology, Varale Talegaon. Savitribai Phule Pune University, Pune(MS) for encouragement and Pune.

REFERENCES


