Comparison of Image Processing Techniques on the Effectiveness of Automatic Signature Forgery Detection

Faiz Haseeb
Dhanani School of Science & Engineering
Habib University
Karachi, Pakistan
faiz_haseeb@hotmail.com

Asad Raza
Dhanani School of Science & Engineering
Habib University
Karachi, Pakistan
ar06246@st.habib.edu.pk

***Abstract*—Signatures are one of the simplest and most unique ways of identification. Every person's signature is different and challenging to replicate, which is why they are used for authentication in banks worldwide, for verifying essential documents, and for verifying transactions. Despite being unique and challenging to replicate, signs are often forged to impersonate an individual so that funds or documents can be stolen. With this project, we aim to identify what image processing techniques best suit a neural network's ability to distinguish between genuine and forged signatures.**

***Index Terms*—Signature, Image-Processing, Convolutional Neural Network, Mean, Median, Canny Edge-Detection, Thresholding**

I. Introduction

Every person's signature is unique and difficult to copy, which is why they are used for authentication in banks worldwide and for verifying essential documents and transaction authentication. Although signatures are unique and difficult to copy, they are frequently falsified to impersonate an individual in countless ways. Our goal with this project is to uncover forgeries that may not be visible to the naked eye and automate this process.

Using image processing for signature detection is distinct from prior techniques for various reasons. Image processing may detect even minor changes in the visual appearance of a signature that conventional examination methods may overlook. Furthermore, the system can comprehend an image because image processing can recognise distinctive patterns, forms, and/or colours.

II. Previous Work

A.

Off-line Signature Verification Based on Geometric Feature Extraction & Neural Network Classification.

This paper presents an off-line signature verification system based on geometric feature extraction and neural network classification. The proposed system is designed to improve the accuracy and efficiency of signature verification.

The paper begins with a discussion of existing signature verification systems. Existing systems typically rely on handcrafted feature extraction methods, such as Fourier descriptors, Gabor filters, and local binary patterns, to extract features from a signature. The extracted features are then used in a classification algorithm to determine the signature's authenticity. However, these methods often have difficulty capturing the subtle differences between genuine and forged signatures.

In contrast, the proposed system utilizes a neural network-based classifier for signature verification. The neural network is trained on a data-set of genuine and forged signatures and is used to classify a given signature as either authentic or forged. The neural network is trained using a geometric feature extraction method, which captures the subtle differences between genuine and forged signatures. The extracted features are then fed into the neural network, which is used to classify the signature.

The paper then presents the results of the experiments conducted to evaluate the system's performance. The experiments showed that the proposed system achieved higher accuracy than the existing methods when tested on a large signatures database. The system also improved accuracy when tested on a subset of the database.

The paper presents a novel off-line signature verification system based on geometric feature extraction and neural network classification. The proposed system achieves better accuracy than existing methods and provides an efficient and accurate way to verify signatures.

B.

Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review.

This paper reviews the existing literature on deep convolutional neural networks (CNN) for image classification. The authors review the existing literature on CNNs, from the early days of CNN research to the more recent developments in the field. The paper reviews the applications of CNNs in various domains, such as medical image analysis, object recognition, and scene understanding.

The paper briefly introduces CNNs, describing their origins and development and their key components, such as convolutional layers, pooling layers, and fully connected layers. The paper then discusses the various architectures and models used for CNNs and the various methods used to train them. This includes a discussion of transfer learning, data augmentation, and regularization techniques, as well as the various methods used to evaluate the performance of a CNN.

The paper then reviews the various applications of CNNs in image classification, including medical image analysis, object recognition, scene understanding, and other related tasks. The paper discusses how CNNs have been used to analyze medical images, such as CT, MRI, and X-ray. It also discusses using CNNs for object recognition tasks, such as recognizing objects in images and videos. Finally, the paper reviews the use of CNNs for scene understanding tasks, such as image segmentation, image de-noising, and image generation.

The paper reviews the existing literature on deep convolutional neural networks for image classification. The authors provide a thorough overview of the various architectures, models, and methods used in CNNs, as well as the various applications of CNNs in different domains. The paper is a valuable resource for researchers looking to gain a better understanding of the current state-of-the-art in deep convolutional neural networks for image classification.

III. Methodology

In order to compare the effectiveness of different image processing techniques on the effectiveness of our convolutional neural network, we must have a control group in which the data set is not pre-processed at all and is left as is. The model is trained on the unaffected data set, whose outputs are used as a baseline comparison with the outputs of the other models trained on pre-processed data with varying filters.

A. Mean Filter

The first image processing technique used in this paper is the mean filter. This works by taking an average of a pixel's neighbours, including itself. The formula is as given: [2]

$$f(x,y) = \frac{1}{mn} \sum_{(s,t) \in S_{xy}} g(s,t)$$

A kernel, in our case 3x3 is applied where each element is $\frac{1}{n}$ where n is the size of the kernel

B. Median Filter

Like the mean filter, the median filter examines each picture pixel and its neighbours to determine if it is typical of its surroundings. It uses the median of surrounding pixel values instead of the mean. // The median is derived by sorting all the nearby pixel values into numerical order and replacing the pixel being analyzed with the middle value.// In this implementation a 5x5 kernel is used.

C. Thresholding

Thresholding, namely manual thresholding, is the process of segmenting an image by order of the pixel values. If a pixel's value is above or below a specific pre-defined value, it is overwritten with either black or white. This method takes in gray scale images and outputs a binary image.

D. Canny Edge Detection

Canny Edge Detection, developed by John F. Canny in 1986, is a method of extracting a wide range of edges from an image. He detailed the process in his paper, "A Computational Approach to Edge Detection [1]. The steps are as follows:

- To start, noise is filtered out, as it greatly affects the results of the edge detection. To achieve this noise filtering, a Gaussian filter kernel is convolved with the image. The equation for a gaussian filter of size $(2k+1) * (2k+1)$ is:
$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(- \frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2} \right); 1 \leq i,j \leq (2k+1)$$
- Finding the intensity gradient of the image. This is done to find the direction of edges in an image. The algorithm detects horizontal, vertical and diagonal edges. The edge gradient and direction is determined by calculating the first derivative in the horizontal G_x and vertical G_y direction and then using the formulae:
$$G = \sqrt{G_x^2 + G_y^2}$$
$$\theta = atan2(G_y, G_x)$$
where atan2 is the arctangent function of the two arguments.
- Gradient magnitude thresholding or lower bound cut off suppresion, which are edge thinning techniques, used to get rid of false responses to edge detection
- Double threshold is used next to filter out edges caused by noise and colour variation. This is done by excluding pixels with a weak gradient value. If the gradient value is smaller than the pre-determined low threshold value, it is suppressed.
- Finally edge tracking is done using hysteresis. This involves checking all the weak gradient value pixels' 8-neighbourhood. If it is connected to any strong gradient value pixel(s) then it is preserved, else it is excluded.

After the pre-processing stage and training and testing of the models, the accuracy results of each model are compared.

IV. Implementation Details

Firstly, we split the chosen "Hand-written Signatures" data-set opted from Kaggle into two groups precisely. One for training and one for testing. In our project, we are using around 70% for training and 30% for testing. Moving on, we create a new directory for storing the training and test data. We are applying the filters and edge-detection on the specified data-set and overwriting them to have the original directory and a new one. There are multiple layers in our model. To name them specifically, Convolutional 2D Layer, Max Pooling

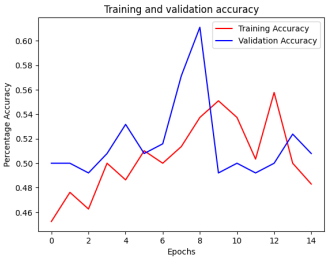
Layer, and the Dense Layer. The Convolutional 2D Layer creates a kernel which is further used to process the training data. When applied to 2-D input, sliding convolutional filters are applied by a 2-D convolutional layer. The layer convolutes the input by moving the filters vertically and horizontally along the input, computing the dot product between the input and the weights, and then adding a bias term. The Max Pooling Layer downsizes the input images with respect to their height and width to focus on the input image's specific features and maintain its result accordingly. A pooling process known as "max pooling" chooses the maximum element from the part of the feature map that was subject to the filter. That's why a max-pooling layer should provide a feature map highlighting the most critical aspects of the prior feature map. Lastly, the dense layer classifies the input from the output. It has a strong connection to the layer that comes before it. This implies that the neurons of the dense layer are connected to each and every neuron of the layer that comes before it.

Moving on further after creating the model, we will train it by specifying the number of runs (epochs). In our case, we chose to set its value for 15 runs. After that, we get the accuracy data and the validation data. The validation data is separated from the testing data to get an unbiased result of the program. After that, we plot all the values for accuracy loss and the validation loss across the number of epochs on the x-axis and the loss on the y-axis, shown further in the Experimental Results section.

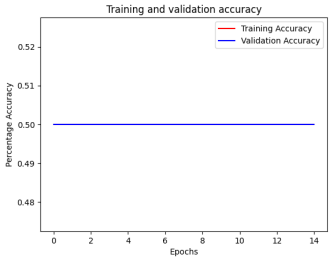
V. Experimental Results

Below is the training and validation accuracy and training and validation loss for each model

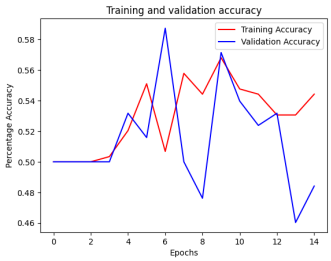
A. *Control Group (no pre-processing)*



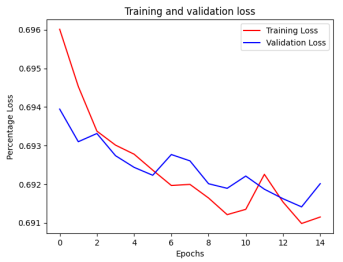
B. *Only Mean Filter Applied*



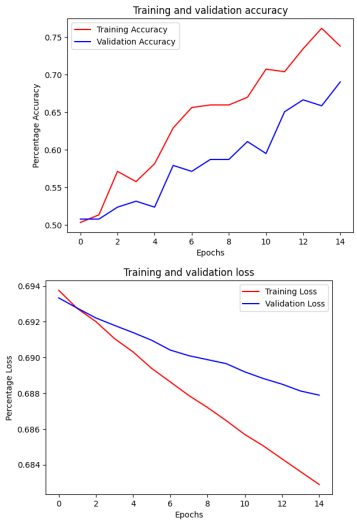
C. *Only Median Filter Applied*



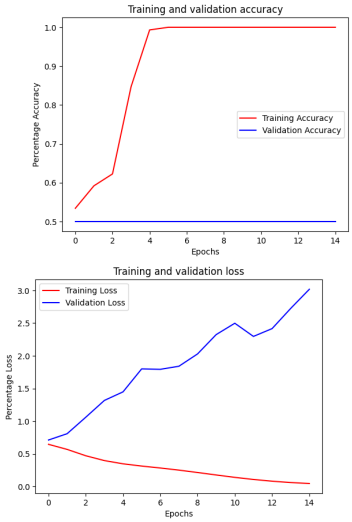
D. *Only Manual Threshold Applied*



E. Only Edge Detection Applied



F. All Pre-Processing Techniques besides Threshold Applied



VI. Conclusion

From the results, we can clearly see that the model which is trained on images that only have Canny edge detection applied on them has the highest validation accuracy, while the model which is trained on images with all techniques applied on them has the best training accuracy and lowest loss, but bad validation accuracy. This means that it wont be very effective in identifying forged and real signatures outside of the given data set, while the canny edge detection model will be, by about 70%. The results from the other models are not really worth mentioning as the training and validation accuracy either stay oscillating around 50% or drop below. From this we can conclude that canny edge detection is the best image processing technique to use when trying to train a neural network to distinguish between forged and real signatures.

Acknowledgment

Thanks to Dr. Muhammad Mobeen Movania, for giving us the means to tackle any problem relating to digital image processing.

References

[1] J. Canny, ”A Computational Approach to Edge Detection,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 6, pp. 679-698, Nov. 1986, doi: 10.1109/TPAMI.1986.4767851.

[2] A. Krzisnik ”How To Use Arithmetic Mean Filter On Images – C# Guide” https://epochabuse.com/arithmetic-mean-filter/ (Accessed 27-11-2022)