YILDIZ TECHNICAL UNIVERSITY
FACULTY OF CHEMISTRY METALLURGICAL
MATHEMATICAL ENGINEERING

MULTIDISCIPLINARY DESIGN PROJECT

SIGNATURE CONTROL APPLICATION

Project Superviser: Prof. Dr. Fatma İnci Albayrak

20058071
Hasan Basri Uzun

İstanbul, 2025

© All rights of this multidisciplinary design project belong to Yildiz Technical University Department of Mathematical Engineering.

TABLE OF CONTENTS **Page**

FIGURES LIST

TABLE LIST

ABBREVIATIONS LIST

SIFT	Scale-Invariant Feature Transform
GNN	Graph Neural Network
DCGANs	Deep Convolutional Generative Adversarial Networks
GNN	Graph Neural Network
CNN	Convolution Neural Network
CEDAR	Center of Excellence for Document Analysis and Recognition
BCE	Binary cross-entropy
MSE	Mean Squared Error
AVG	Average
MAX	Maximum

PREFACE

Artificial intelligence has always been an area of interest for me, and I have developed myself to a certain level in this field. However, I had never conducted my own research before. Combining my interest in image processing with artificial intelligence, I started questioning how I could use these two concepts to solve a real-world problem. This led me to the idea of researching methods for detecting forged signatures. After reading numerous articles and with the guidance of my supervisor, Fatma İnci Albayrak, I have successfully completed this study.

I would like to express my heartfelt gratitude to my supervisor, Fatma İnci Albayrak, for her accurate, structured feedback and continuous support through.

ABSTRACT

In many written documents, signatures are widely used for verification, approval, and validation of transactions. However, the process of verifying signatures presents a distinct challenge. Manual signature verification depends on human expertise and is often time-consuming. At this point, leveraging technological advancements becomes essential. To rapidly verify large numbers of signatures, emerging Deep Learning methods are increasingly utilized today. Among these methods, Convolutional Neural Networks (CNNs) stand out because of their straightforward architecture, fast training process, and the availability of numerous optimization techniques, making it easier to develop efficient models. In this study, the effects of overfitting were examined using CNNs with various epoch numbers, suitable batch sizes were determined through experiments, the UTSig and CEDAR datasets were employed, and a novel pooling method, T-Max-Avg, was tested. Instead of using a pre-designed architecture, a custom architecture was developed specifically for this study. The research demonstrates that the 4x4 pooling method is more efficient for signature verification, and the newly proposed T-Max-Avg method has also proven to be more effective.

ÖZET

Birçok yazılı dökümanda doğrulama, onaylama ve işlemleri geçerli kılmak için günümüzde imza sıkça başvurulan bir yöntemdir. Ancak, imzayı doğrulama süreci başlı başına bir sorun teşkil etmektedir. İnsan gözüyle imza doğrulama işlemi kişinin yetkinliğine bağlı olup oldukça zaman alıcı olabilmektedir. Bu noktada, teknolojik gelişmelerden yararlanmak kaçınılmazdır. Çok sayıda imzayı hızlı bir şekilde doğrulamak için günümüzde gelişmekte olan Derin Öğrenme (Deep Learning) yöntemlerine başvurulmaktadır. Bu yöntemler arasında yer alan Convolutional Neural Network (CNN), basit yapısı ve mevcut optimizasyon teknikleri sayesinde eğitimi hızlı ve model geliştirme sürecini daha verimli bir kılmaktadır. Bu çalışmada, CNN kullanılarak farklı iterasyon sayıları ile aşırı öğrenme (overfitting) etkileri araştırılmış, çeşitli paket sayısı (batch size) değerleri ile uygun paket sayısı belirlenmiş, UTSig ve CEDAR veri setleri kullanılmış, yeni bir pooling yöntemi olan T-Max-Avg denenmiş ve hazır bir mimari kullanmak yerine bu çalışmaya özel bir mimari geliştirilmiştir. Araştırma sonucunda, imza doğrulamada 4x4 boyutundaki pooling yönteminin daha verimli olduğu ayrıca **yeni bir yöntem olan** T-Max-Avg yönteminin daha verimli olduğu gösterilmiştir

1. INTRODUCTION

In today's world, secure authentication and authorization have become essential, with biometric systems widely utilized in fields such as banking, passport control, and legal applications to enhance security measures. Among various biometric options, signatures are particularly noteworthy due to their quick verification process and distinct characteristics that reflect an individual's unique handwriting. A robust signature recognition system is built upon the critical modules of biometric systems [2]: data capture, feature extraction, matching, and secure storage. This integration facilitates efficient identity verification while addressing challenges such as forgery detection. It thereby establishes signatures as a vital component of contemporary biometric applications.

Signature forgery is categorized into three distinct types, each presenting its own level of complexity and similarity: random forgery, simple forgery, and skillful forgery. **Random forgery** refers to signatures that bear no resemblance to the genuine owner's signature, making detection straightforward and efficient. **Simple forgery** arises when a forger tries to imitate the actual owner's signature style but lacks the necessary precision, leading to clear differences between the forged and original signatures that can be easily spotted. On the other hand, **skillful forgery** is executed by individuals who have meticulously studied or practiced the original signature, resulting in a high degree of similarity that makes detection significantly more challenging. As such, skillful forgery stands out as the most difficult type to identify and usually demands advanced signature verification techniques for successful detection.

Despite these challenges, traditional manual verification methods can be inefficient in terms of both time and labor. Over the past years, machine learning and deep learning have proven their potential to automate the signature verification process, enhancing both accuracy and speed. Different machine learing and deep learning techniques have been examined in the literature have addressed distinct challenges. For instance, Ali and Hadieh [25] utilized the SIFT [18] algorithm for detailed feature extraction and analyzed it using a GNN model. Their model, trained exclusively on genuine signatures, achieved accuracies of 99.2% on the UTSig dataset and 100% on the MCYT-75 dataset. These findings demonstrate that forged signature detection is feasible even for small-sized signature images.

In the context of transfer learning using pre-trained models, Hirunyawanakul, Bunrit, Kerdprasop, and Kerdprasop [17] conducted a study employing AlexNet and VGG16 models, followed by their custom CNN architectures. This approach showed that training models with high accuracy are achievable within shorter time frames. Max-pooling layers, commonly used in CNN architectures for image processing, were employed by Yuchen and colleagues to analyze micro-deformations in genuine and forged signatures. Their method focused on feature extraction to detect subtle differences and successfully capture positional features. This approach strengthened the forgery detection process, enabling highly accurate detection of forgeries through CNN-based feature extraction.

While CNN architectures are typically supervised learning models, Zhang et al. [21] presented a multi-phase offline signature verification framework using the DCGAN method for unsupervised feature learning. This approach combined the strengths of writer-independent and writer-dependent classifiers to balance accuracy and convenience, achieving effective verification results on large datasets.

This study aims to review existing literature [23] to analyze previously used models and methods, identifying successful approaches and recording the factors contributing to their effectiveness through re-evaluation. A new method will then be developed by combining these successful techniques with

deep convolutional neural network architectures. The anticipated results aim to provide an enhanced version of current studies and offer guidance for future research.

The organization of this study is planned: In Section 2, the background of the proposed method is provided, along with a review of previous studies related to it. Section 3 introduces the proposed method in detail. Section 4 presents the experimental setup and results, including a comparison with other models in the literature. Finally, Section 5 summarizes the overall findings of the study, discussing the positive and negative aspects of the proposed technique.

2. PRELIMINARIES

CNN is a widely used deep learning method applied in various fields, such as image processing and image classification. The fundamental structure of a CNN consists of layers like Figure 1. These layers are responsible for extracting features from the image, selecting the most important ones, thereby reducing computational load, and making predictions. Some of these layers include convolutional layers, activation layers, pooling layers, and batch normalization layers.

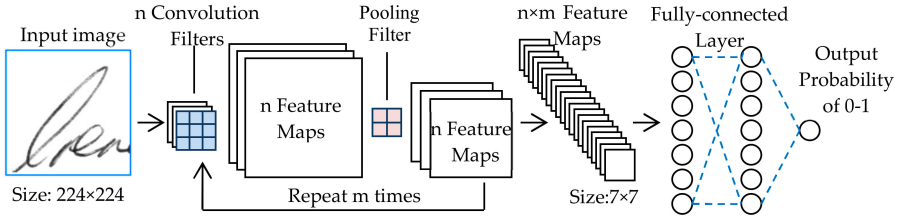


Figure 2.1. Schematic diagram of CNN [19]

2.1 Convolitional layer

The convolutional layer, a core element of CNNs, employs multiple convolution filters to extract high-level features such as shapes, structures, and complex patterns by building upon low-level features like edges, corners, and textures [6]. Generating a feature map emphasizes distinct characteristics within the data. This layer can extract features from various types of data, including written content [25], sound recordings [16], sequential data [14], and more.

2.2 Pooling Layer

The pooling layer is a crucial layer with different computational methods. Its primary function is to decrease the size of the feature map, thereby decreasing the computational load. Consequently, the pooling layer [4] facilitates the extraction of prominent features from the feature map [13]. It achieves this through various methods, such as max-pooling and average-pooling.

2.2.1 Max Pooling

The max-pooling [7] layer selects the highest value in the chosen area and discards the remaining values, thus retaining the most important feature in that region. Max-pooling also helps improve the model's translation invariance, as it ensures that key features are preserved even in the presence of small shifts or transformations. It is typically implemented by specifying filter size and stride values, allowing the data to be sampled at regular intervals.

2.2.2 Avarage Pooling

Average pooling [3] is a commonly used pooling method in convolutional neural networks (CNNs) for feature extraction. The method calculates the average of all values within a specific region and represents that region with a single value. Average pooling is preferred to reduce data dimensions and achieve more abstract representation levels.

2.2.3 Avg-TopK Pooling

The Avg-TopK [22] pooling method is an advanced pooling technique designed to address the limitations of traditional pooling methods like max pooling and average pooling. The process entails choosing the K largest values in a pooling region of the input data and calculating their mean. By doing so, Avg-TopK pooling preserves the most significant features in the data while mitigating the risk of noise or loss of subtle variations often encountered in traditional methods.

2.3 Activation Functions

Activation functions [27] transform the output from the filters using a mathematical operation, enabling the model to learn non-linear relationships. This transformation facilitates the development of more complex neural network architectures and helps uncover deep patterns within the data. Activation functions not only enhance the model's predictive power but also simplify the optimization process. The most popular activation functions include ReLU, sigmoid, tanh, and softmax.

2.4 Adaptive Moment Estimation

Adam (Adaptive Moment Estimation) [26] is a widely used optimization algorithm in training deep learning models. Using the exponential moving averages of the first and second moments of the gradients, this algorithm adapts the learning rates for individual parameters. This approach enables fast and efficient convergence, especially for problems involving large datasets and parameter sets. Additionally, the Adam algorithm is invariant to diagonal rescaling and computationally efficient with limited memory requirements. These features make it highly effective in handling noisy or sparse gradients, delivering robust performance in such scenarios.

2.5 Dropout

Dropout [9] is an effective technique developed for regularizing deep learning models and preventing overfitting. It operates by randomly deactivating selected neurons during training, and this process is repeated with different combinations in each forward pass. This method reduces the model's dependency on specific neurons, enabling it to achieve a more generalizable structure. To balance the variance introduced by the random deactivation of neurons, dropout scales the activations of the entire network during the testing phase, ensuring consistent performance.

2.6 Fully-Connected Layer

A fully-connected (FC) [27] layer connects every neuron in one layer to every neuron in the next, enabling the network to combine extracted features for final predictions. It is essential for tasks like classification and regression, as it maps high-dimensional inputs to outputs. Despite being parameter-intensive, FC layers are key to modeling complex relationships in diverse applications.

3. PROPOSED METHOD

3.1 OTSU

The Otsu algorithm prepares images by thresholding them for use in model learning. The OTSU [1] method analyzes the image, computes preprocessing parameters, and finds optimal thresholding volues (binary or multi-threshold) to separates pixel regions to highlight the image's important features. Given an image with pixels distributed L gray levels $[1,2,3,\dots,L]$, the OTSU thresholding technique determines the optimal threshold k^* to segment the image into two levels gray intensity.

To describe the method, let n_i represent the number of pixels at gray level i, and the total of pixel count $N = n_1 + n_2 + \cdots + n_L$. The simplify the analysis, the gray-level histogram is transformed into a normalized probability distribution, expressed as:

$$p_i = \frac{n_i}{N}, \quad p_i \geq 0, \quad \sum_{i=1}^{L} p_i = 1 \tag{1}$$

Consider partitioning the pixels into two distinct classes, C_0 and C_1 denote the classes separated by the threshold k. Pixels with levels $[1,2,\dots,k]$ belong to C_0, and those in $[k+1,\dots,L]$ from C_1. The probabilities of accurrence for these classes and mean of class levels respecively, are given by

$$\omega_0 = Pr(C_0) = \sum_{i=1}^{k} p_i = \omega(k) \tag{2}$$

$$\omega_1 = Pr(C_1) = \sum_{i=k+1}^{L} p_i = 1 - \omega(k) \tag{3}$$

and

$$\mu_0 = \sum_{i=1}^{k} i\, Pr(i \mid C_0) = \frac{\sum_{i=1}^{k} i\, p_i}{\omega_0} = \frac{\mu(k)}{\omega(k)} \tag{4}$$

$$\mu_1 = \sum_{i=k+1}^{L} i\, Pr(i \mid C_1) = \frac{\sum_{i=k+1}^{L} i\, p_i}{\omega_1} = \frac{\mu_T - \mu(k)}{1 - \omega(k)} \tag{5}$$

where

$$\omega(k) = \sum_{i=1}^{k} p_i, \quad \mu(k) = \sum_{i=1}^{k} i\, p_i \tag{6}$$

They correspond to the zeroth- and first-order cumulative moments of the histogram calculated up to the k-th gray level. Also total of original picture that mean level is

$$\mu_T = \mu(L) = \sum_{i=1}^{L} i\, p_i \tag{7}$$

Formatted: Caption

Formatted: Font: Times New Roman, 12 pt, Not Italic, Font color: Text 1

Formatted: Font: 12 pt, Font color: Text 1

Formatted: Font: Times New Roman, 12 pt, Not Italic, Font color: Text 1

Formatted: Font: 12 pt, Font color: Text 1

The variance of the classes are given by

$$\sigma_0^2 = \sum_{i=1}^{k} (i - \mu_0)^2 \, Pr(i \mid C_0) = \sum_{i=1}^{k} \frac{(i - \mu_0)^2 p_i}{\omega_0} \tag{8}$$

$$\sigma_1^2 = \sum_{i=k+1}^{L} (i - \mu_1)^2 \, Pr(i \mid C_1) = \sum_{i=k+1}^{L} \frac{(i - \mu_1)^2 p_i}{\omega_1} \tag{9}$$

To evaluate the "effectiveness" of the threshold at level k, the following discriminant criterion measures, commonly used in discriminant analysis to assess class separability, are introduced:

$$\lambda = \frac{\sigma_B^2}{\sigma_W^2}, \quad \kappa = \frac{\sigma_T^2}{\sigma_W^2}, \quad \eta = \frac{\sigma_B^2}{\sigma_T^2} \tag{10}$$

Here, the within-class variance σ_W^2 the between-class variance σ_B^2 and the total variance σ_T^2 are defined as follows:

$$\sigma_W^2 = \omega_0 \sigma_0^2 + \omega_1 \sigma_1^2, \tag{11}$$

$$\sigma_B^2 = \omega_0 (\mu_0 - \mu_T)^2 + \omega_1 (\mu_1 - \mu_T)^2 = \omega_0 \omega_1 (\mu_1 - \mu_0)^2, \tag{12}$$

$$\sigma_T^2 = \sum_{i=1}^{L} (i - \mu_T)^2 p_i \, . \tag{13}$$

These definitions reduce the problem to an optimization task, where the goal is to find the threshold k that maximizes one of the given criterion measures λ, κ, η.

The optimal threshold k^* , which maximizes η or equivalently σ_B^2, can be determined using a sequential search. This process utilizes the simple cumulative quantities defined in (6) and (7), or explicitly based on (2)-(5). The discriminant criterion $\eta(k)$ is defined as:

$$\eta(k) = \frac{\sigma_B^2(k)}{\sigma_T^2} \tag{14}$$

The between-class variance, $\sigma_B^2(k)$ is computed as:

$$\sigma_B^2(k) = \frac{[\mu_T \omega(k) - \mu(k)]^2}{\omega(k)[1 - \omega(k)]} \tag{15}$$

The optimal threshold k^* then given by:

$$\sigma_B^2(k^*) = \max_{1 \le k < L} \sigma_B^2(k) \tag{16}$$

From this definition, the range of k values over which the maximum is sought can be restricted to the set:

$$S^* = \{k; \omega_0\omega_1 = \omega(k)[1 - \omega(k)] > 0, \text{or } 0 < \omega(k) < 1\} \quad (17)$$

This process ensures that the optimal threshold k^* maximizes the between-class variance $\sigma_B^2(k)$, achieving the best separation between classes in the histogram.

3.2 Convoloution Layer

In our project, we input an image, which can also be viewed as a multidimensional array. We will refer to this array as I. A filter will be applied to the image, which can similarly be considered as a multidimensional array. We will denote this filter as K. This operation is represented using the following notation [15]:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \quad (18)$$

The convolution operation mentioned here is illustrated in Figure 2. In Figure 2, the section labeled as a, b, c, d, …, l in the Input corresponds to the variable I described in (18). The variable denoted as K in (18) refers to the section named Kernel in Figure 2. The output from (18) is represented as the Output section in Figure 3.1.

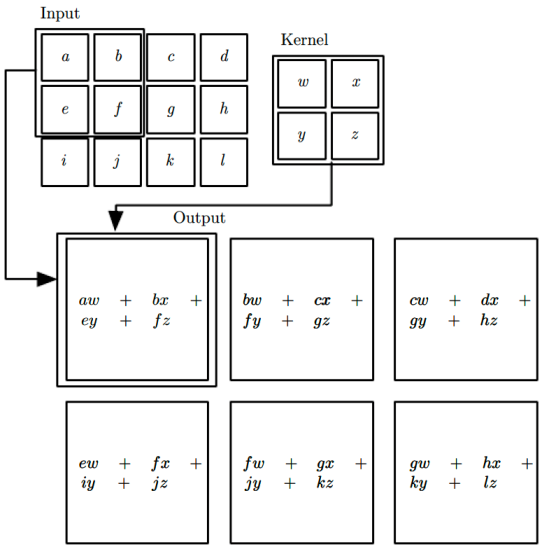


Figure 3.1. 2-D Convolution [15]

The matrix obtained after the convolution operation can be referred to as the feature map.

3.3 Activation Function

In order to capture nonlinear relationships within the data, an activation function [5] layer is employed. The ReLU activation function works by eliminating negative values and passing positive values unchanged. The corresponding formula is expressed as:

$$f(x) = \max(0, x) \tag{19}$$

3.4 Pooling Layer

The pooling layer [10] is essential for extracting key features from the input image. By applying pooling, we reduce the spatial dimensions of the data, which helps minimize computational resources and processing time.

3.4.1 Max pooling

The Max pooling [22] method is a widely used technique in CNN models for image processing. When applied, it not only reduces the spatial dimensions but also emphasizes the most significant features. Its primary function is to select the highest value within a specific region and pass it on to the next layer. The mathematical formulation for this operation is presented in (20).

$$F_{\max}(x) = \max\{x_i\}_{i=0}^{N} \tag{20}$$

The schematic representation of this operation is shown in Figure 15. In this figure, the highest value from the 3x3 pooling region, indicated by the blue area, is selected and passed on to the next stage.

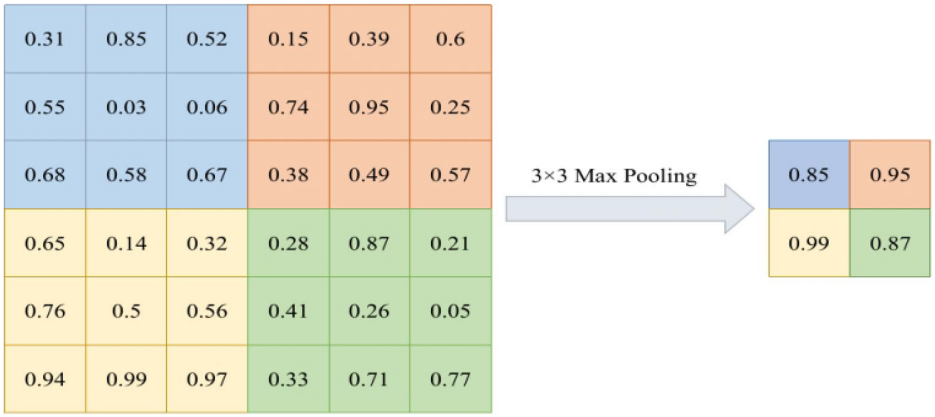


Figure 3.2 Max Pooling Layer [22]

3.4.2 Average Pooling

Average pooling is a widely used layer in CNNs. This layer computes the average of all values within a specified region (4) and passes the result to the next layer. The purpose of this layer is to reduce the dimensionality of the features while retaining the important information. As shown in Figure 4, the average of the 3x3 region within the defined area is calculated and transferred to the subsequent layer. The mathematical formulation for this operation is given in (21).

$$F_{\text{average}}(x) = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{21}$$

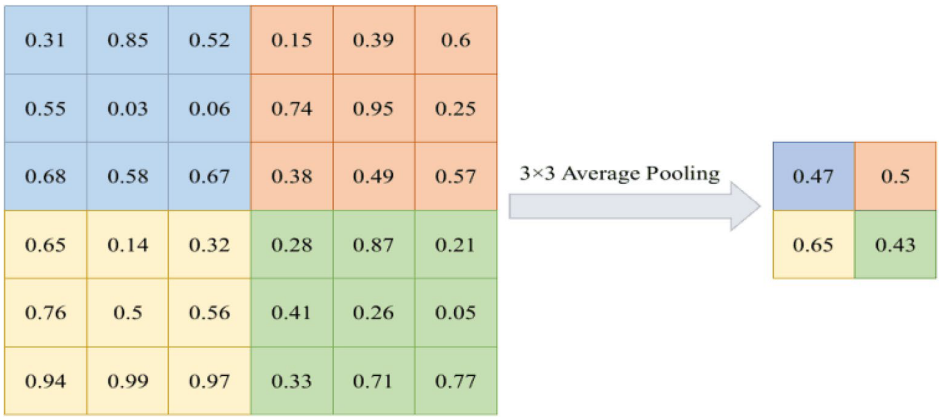


Figure 3.3. Average pooling [22]

3.4.3 Avg-TopK pooling

The purpose of the Avg-TopK [15]method is to address the shortcomings of the Max Pooling and Average Pooling methods. This method calculates the average of the top k highest values within the selected region. (22) provides the mathematical formulation of this method.

$$F_{\text{Avg-TopK}}(X) = \frac{1}{k}\sum_{i=1}^{k} Y_i \tag{22}$$

It is stated in Figure 3.4 that the average is calculated after selecting the top 3 highest values within the 3x3 region.

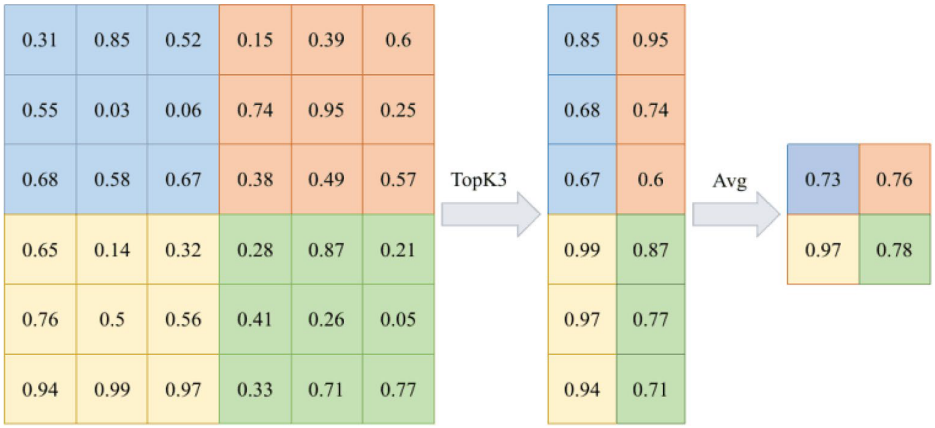


Figure 3.4. 3x3 Avg-TopK pooling. [22]

3.4.4 T-Max-Avg pooling

In this method of T-Max-Avg [22], a value T is first determined. Subsequently, the top k values are selected from the defined region. If all the selected values are greater than T, the method uses Max Pooling under specific conditions; if not, it switches to Average Pooling, effectively mitigating the drawbacks of both approaches.The formulation is provided in (23).

$$F_{T\_Max\_Avg}(X) = \begin{cases} max\ \{Y_i\}_{i=0}^{K} & , \quad Y_i \geq T \\ \frac{1}{K}\sum_{i=1}^{K} Y_i & , \quad Y_i < T \end{cases} \tag{23}$$

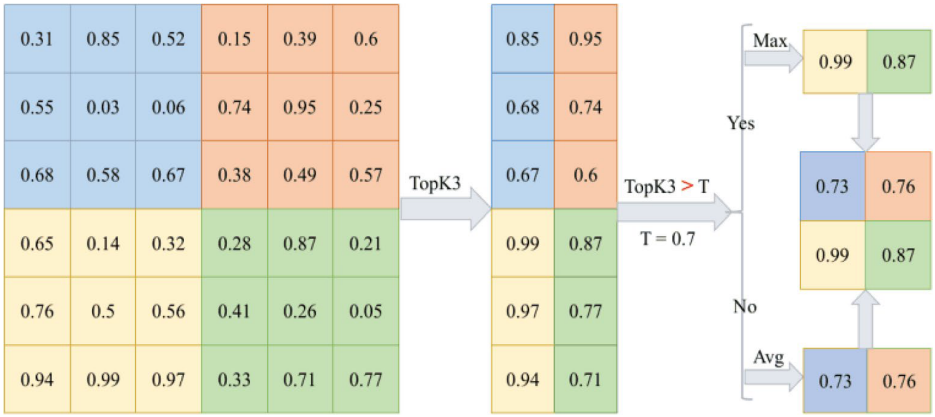


Figure 3.5. T-Max-Avg pooling [22]

3.5 Dropout Layer

A standard neural network consists of layers, with the fundamental ones being the input, hidden, and output layers. However, utilizing only these layers can lead to overfitting, where the model memorizes the data instead of learning from it. In such cases, the inclusion of a dropout layer becomes functional. The dropout layer randomly deactivates a certain portion of neurons, helping to prevent overfitting. The mathematical representation of this method is provided as (24) – (27).

$$r_j^{(l)} \sim \text{Bernoulli}(p) \tag{24}$$

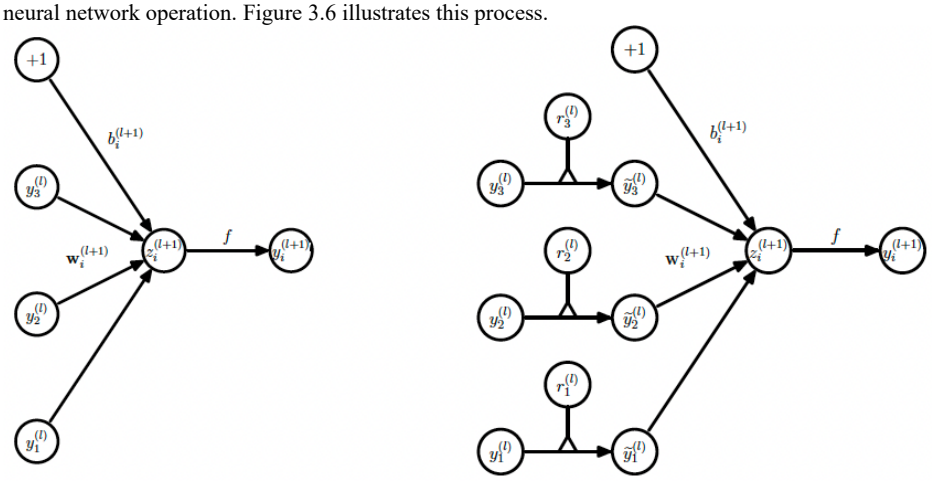
$$\widetilde{y^{(l)}} = r^{(l)} * y^{(l)} \tag{25}$$

$$z_i^{(l+1)} = w_i^{(l+1)}\widetilde{y^{(l)}} + b_i^{(l+1)} \tag{26}$$

$$y_i^{(l+1)} = f\left(z_i^{(l+1)}\right) \tag{27}$$

where, $r_j^{(l)}$ represents a random variable for the j-th neuron in the l-th layer. Since this variable follows a Bernoulli distribution, its value will be either 1 or 0. The term $\widetilde{y^{(l)}}$, due to the $r^{(l)}$ variable, will either

be 0 or will have the same degree as $y^{(l)}$. Subsequently, the computations proceed as in a standard neural network operation. Figure 3.6 illustrates this process.



(a) Standard network (b) Dropout network

Figure 3.6 . Comparison of the fundamental operations between a standard neural network and a dropout-enhanced network [11]

3.6 Feedorward Neural Network

Data flows in a single, uninterrupted path from the input layer to the output layer in a feedforward Neural Network, with no feedback mechanisms. In this architecture, every neuron in a layer is connected to all neurons in the subsequent layer. The mathematical representation of a single-hidden-layer architecture is given as follows:

$$N_i = \sum_{j=1}^{R} I_{w_{i,j}} x_j + Hb_i \quad \forall\, i = 1,2,\dots,n. \tag{28}$$

$$H_{o_i} = f\left(\frac{N_i}{n}\right) \quad \forall\, i = 1,2,\dots,n. \tag{29}$$

$$y = \sum_{i=1} H_{w_i} H_{o_i} + Ob. \tag{30}$$

where, N_i represents the i-th neuron in the hidden layer. This neuron is calculated as shown in (28). The term $w_{i,j}$ represents the weight between the j-th input x_j and the i-th hidden neuron. Hb_i denotes the bias of the i-th hidden neuron. The output H_{o_i} is the result of activating N_i using the activation function f. In the output layer, the final output y is calculated by summing the product of the hidden neuron activations H_{o_i} and their corresponding weights H_{w_i}, along with the bias Ob.

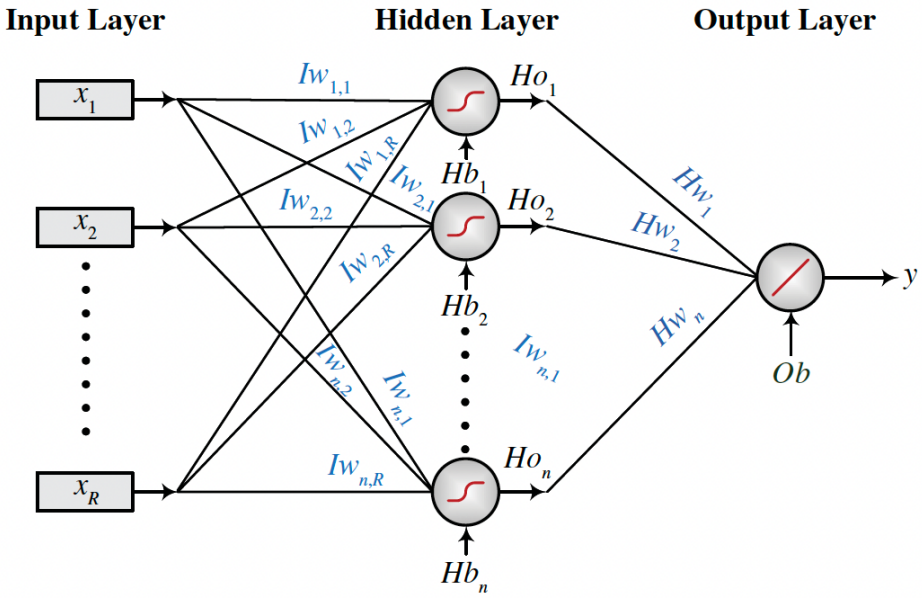


Figure 3.7. Feedforward neural network with a single hidden layer and a single output [11]

3.7 Backpropagation Algorithms

The backpropagation algorithm [24] is a keystone of learning in deep neural networks (DNNs). This algorithm calculates the error between the model's prediction during the feedforward phase and the actual outcome, then optimizes the model's biases and weights to reduce this error. The algorithm uses the gradient of the loss function to update the parameters, typically through gradient descent.

In the forward propagation phase, the input is processed layer by layer to compute the predicted value z. This value is calculated as shown in (31), where the weighted input is summed with the bias. The resulting z value is then sent through an activation function, transforming it into the activation output a, which is propagated to the next layer.

$$z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]} \quad (31)$$

$$a^{[l]} = f\left(z^{[l]}\right) \quad (32)$$

Here, $z^{[l]}$ is the linear combination in the l-th layer, and $a^{[l]}$ is the output of the activation function. The performance of the model is evaluated using a loss function $\mathrm{l}\left(y, \hat{y}(\theta)\right)$, which measures the variation between the actual value y and the predicted value $\hat{y}(\theta)$. The loss function, defined in Equation 7, serves as the foundation for the learning process:

$$\mathrm{l}\left(y, \hat{y}(\theta)\right) = \mathrm{l}\left(y, f^{[L]}\left(z^{[L]}(\theta)\right)\right) \quad (33)$$

There are various types of loss functions utilized in neural networks, each tailored to specific tasks. Among the most commonly used are the following:

3.7.1 Binary cross-entropy (BCE)

This loss function [20] is designed for binary classification tasks, measuring the difference between predicted probabilities $\hat{y}$ and actual binary labels y. It evaluates The measure of how effectively the model separates two classes is defined as:

$$l_{BCE}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \quad (34)$$

where C denotes the number of classes.

3.7.2 Mean Squared Error (MSE)

Typically employed in regression tasks, MSE [12] assesses the mean squared variation between actual and predicted outcomes, quantifying the error in continuous predictions:

$$l_{MSE}(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \quad (35)$$

These loss functions forms the basis for the backpropagation process. During the backward propagation phase, the gradient of the loss function in (35) is computed and used to optimize the network's parameters. This process begins at the output layer, where the error is calculated as the gradient of the loss function with respect to the output. The error is then propagated backward through the network, being multiplied by the Jacobian matrix $J_\theta z^{[L]}(\theta)$ and the derivatives of the activation functions. This recursive propagation through the layers enables the adjustment of weights and biases to minimize the overall loss function, thereby improving the model's predictions.

$$\nabla_\theta l\left(y, \hat{y}(\theta)\right) = \left(J_\theta z^{[L]}(\theta)\right)^{\top} \nabla_{z^{[L]}} l\left(y, f^{[L]}(z^{[L]})\right) \quad (36)$$

This backward propagation process leads directly to the update of the network's parameters weights and biases through gradient descent. Once the gradients of the loss function with respect to the parameters are computed, the weights $W^{[l]}$ and biases $b^{[l]}$ are iteratively updated to minimize the loss. The updates are mathematically respectively defined as:

$$W^{[l]} \leftarrow W^{[l]} - \eta \nabla_{W^{[l]}} l\left(y, \hat{y}(\theta)\right) \quad (37)$$

$$b^{[l]} \leftarrow b^{[l]} - \eta \nabla_{b^{[l]}} l\left(y, \hat{y}(\theta)\right) \quad (38)$$

where, η represents the learning rate, which controls the magnitude of each step taken in the parameter space. The gradients, $\nabla_{W^{[l]}} l$ and $\nabla_{b^{[l]}} l$, provide the direction of steepest descent, ensuring that the weights and biases are adjusted, respectively, to progressively reduce the error.

This sequential parameter optimization [8] cycle is repeated for each training iteration, allowing the model to iteratively refine its understanding of the data. Over successive iterations, the parameters converge toward values that minimize the loss, thereby improving the network's predictive capabilities.

3.8 Performance Metrices for Classification Models

In binary classification problems, the evaluation of model performance is critical for understanding its accuracy, precision, and reliability. Several metrics are used to assess different aspects of the model's ability to classify data points correctly.

True Positive (TP) represents situations where positive instances are accurately identified, whereas

True Negative (TN) corresponds to cases correctly classified as negative. On the other hand, False Positive (FP) occurs when negative instances are mistakenly predicted as positive, and False Negative (FN) arises when positive cases are incorrectly categorized as negative. These terms form the foundation for deriving metrics like accuracy, precision, recall, and F1-score, which are essential for assessing a classification model's effectiveness.

3.8.1 Accuracy

Accuracy is the proportion of correctly classified instances (both positive and negative) among all instances.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (39)$$

3.8.2 Precision

Precision measures the proportion of correctly predicted positive instances out of all predicted positive instances. It is calculated as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (40)$$

3.8.3 Recall (Sensivitiy)

Recall measures the proportion of actual positive instances that are correctly identified:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (41)$$

3.8.4 F1-Score

The F1-score is the harmonic mean of precision and recall. It balances the trade-off between these two metrics, especially in imbalanced datasets. The formula is:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (42)$$

4. APPLICATION AND RESULT

4.1 Data

4.1.1 Cedar (Center of Excellence for Document Analysis and Recognition) Dataset

The CEDAR dataset is a prominent dataset primarily used for handwriting and signature verification tasks. It consists of a wide variety of handwritten signatures collected from multiple individuals. The dataset includes both genuine and forged signatures, making it ideal for training and evaluating models in biometric authentication and fraud detection applications.

Formatted: Font: 12 pt, Not Bold, Font color: Text 1

4.1.2 UTsig (A Persian Offline Signature Dataset)

UTSig serves as a key dataset for offline signature verification studies, offering a rich collection of authentic and forged signatures that reflect real-world variations in handwriting. Its comprehensive design facilitates the benchmarking of algorithms, ensuring that models are evaluated for both accuracy and generalization capabilities in diverse application scenarios.

4.2 Created CNN network

To evaluate the effectiveness of pooling methods and pooling sizes, optimization methods, and batch size, a new architecture is created in Table 4.1. This architecture consists of a total of 11 layers, including 4 convolution layers, 3 pooling layers, 2 dropout layers, and 2 fully connected layers.

Table 4.1. Architecture 1

Layer	Layer Type	Size of Feature Map	Kernel Size	Parameter	Activation Function
Input Layer	Image	224 x 224 x 1	-		-
Conv1 Layer	Convolution	219 x 219 x 32	6 x 6	1,184	ReLU
Pool1 Layer	CustomPooling	73 x 73 x 32	3 x 3	0	-
Dropout1 Layer	Dropout (0.35)	73 x 73 x 32	-	0	-
Conv2 Layer	Convolution	69 x 69 x 64	5 x 5	51.264	ReLU
Pool2 Layer	CustomPooling	23 x 23 x 64	3 x 3	0	-
Dropout2 Layer	Dropout (0.30)	23 x 23 x 64	-	0	-
Conv3 Layer	Convolution	19 x 19 x 120	5 x 5	192.120	ReLU
Conv4 Layer	Convolution	15 x 15 x 240	5 x 5	720,240	ReLU
Pool3 Layer	CustomPooling	5 x 5 x 240	3 x 3	0	
Flatten Layer	Flatten	6,000	-	0	-
Dropout4 Layer	Dropout (0.50)	6000	-	3,072,512	-
FC1	Fully Connected	512	-	0	ReLu
Dropout5 Layer	Dropout (0.30)	512	-	131,328	-
FC2	Fully Connected	256	-	514	ReLU
FC3	Fully Connected	2	-		Softmax

4.3 Cedar (Center of Excellence for Document Analysis and Recognition) Dataset

The CEDAR dataset is a prominent dataset primarily used for handwriting and signature verification tasks. It consists of a wide variety of handwritten signatures collected from multiple individuals. The dataset includes both genuine and forged signatures, making it ideal for training and evaluating models in biometric authentication and fraud detection applications.

Formatted: Font: 12 pt, Not Bold, Font color: Text 1

4.4 UTsig (A Persian Offline Signature Dataset)

UTSig serves as a key dataset for offline signature verification studies, offering a rich collection of authentic and forged signatures that reflect real-world variations in handwriting. Its comprehensive design facilitates the benchmarking of algorithms, ensuring that models are evaluated for both accuracy and generalization capabilities in diverse application scenarios.

4.5 Network parameters

Formatted: Heading 2

The experiments in this section were conducted using gradient-based optimization algorithms. The pixel values in the images range between 0 and 255; however, they were normalized to fall within the range of 0 to 1.

4.6 Experimental results

In Section 4, the k value in the T-max-Avg-k term is described as progressing in the sequence $k = 1, 2, 3, 4, \ldots, n$ as defined in Equation (22).

Table 4.2. Comparison of experimental results for varying batch sizes and epochs when the pooling size = 3.

CEDAR	Batch size = 16	Batch size = 32	Batch size = 64	Batch size = 128
Epoch = 30	0.910985	0.920455	0.895833	0.892045
Epoch = 50	0.922348	0.939394	0.914773	0.918561
Epoch = 100	0.893939	0.929924	0.935606	0.933712

Figure 4.4.1 Traning and validation accuracy graph; traning and validation loss graph and ROC Curve for Epochs: 50, Batch Size: 32, Pooling size: (3, 3)

Figure 4.4.2 Traning and validation accuracy graph; traning and validation loss graph and ROC Curve for Epochs: 100, Batch Size: 32, Pooling size: (3, 3)

- As observed in Table 4.2, the highest performance is achieved with 50 epochs and a batch size of 32. However, as the number of epochs increases, overfitting occurs, as illustrated in Figure 4.4.2, adversely affecting the model's performance. In subsequent studies, all models will be trained using a batch size of 32 and 50 epochs.

Formatted: Font: Bold

Formatted: Font: Bold

Deleted: 16

Table 4.3. Batch size: 32, Learing rate: 0.0001, T: 0.8, Pooling size: 3, Epoch: 50

CEDAR	Exp-1 (Ac)	Exp-2 (Ac)	Exp-3 (Ac)	Exp-4 (Ac)	Exp-5 (Ac)	Exp-6 (Ac)	Average (Ac)
Avg-Pooling	0.9394	0.9318	0.9356	0.9356	0.9299	0.9318	0.9340
Max-Pooling	0.9375	0.9337	0.9432	0.9299	0.9299	0.9394	0.9356
T-Max-Avg-1	0.922348	0.916667	0.905303	0.909091	0.907197	0.914773	0.912563
T-Max-Avg-2	0.920455	0.907197	0.914773	0.922348	0.910985	0.890152	0.910985
T-Max-Avg-3	0.893939	0.905303	0.910985	0.909091	0.910985	0.909091	0.906566
T-Max-Avg-4	0.914773	0.907197	0.918561	0.912879	0.905303	0.895833	0.909091
T-Max-Avg-5	0.890152	0.912879	0.903409	0.897727	0.909091	0.907197	0.903409
T-Max-Avg-6	0.897727	0.907197	0.899621	0.909091	0.910985	0.905303	0.904987
T-Max-Avg-7	0.910985	0.903409	0.893939	0.893939	0.912879	0.909091	0.904040

Table 4.4. Batch size: 32, Learing rate: 0.0001, T: 0.7, Pooling size: 3

CEDAR	Exp-1 (Ac)	Exp-2 (Ac)	Exp-3 (Ac)	Exp-4 (Ac)	Exp-5 (Ac)	Exp-6 (Ac)	Average (Ac)
T-Max-Avg-1	0.924242	0.912879	0.907197	0.914773	0.922348	0.909091	0.915088
T-Max-Avg-2	0.907197	0.907197	0.909091	0.914773	0.918561	0.909091	0.910985
T-Max-Avg-3	0.920455	0.905303	0.901515	0.909091	0.903409	0.910985	0.908460
T-Max-Avg-4	0.905303	0.912879	0.899621	0.907197	0.892045	0.901515	0.903093
T-Max-Avg-5	0.901515	0.901515	0.893939	0.907197	0.899621	0.905303	0.901515
T-Max-Avg-6	0.893939	0.905303	0.907197	0.892045	0.903409	0.897727	0.899937
T-Max-Avg-7	0.892045	0.888258	0.893939	0.893939	0.901515	0.905303	0.895833

Table 4.5. Batch size: 32, Learing rate: 0.0001, T: 0.6, Pooling size: 3

CEDAR	Exp-1 (Ac)	Exp-2 (Ac)	Exp-3 (Ac)	Exp-4 (Ac)	Exp-5 (Ac)	Exp-6 (Ac)	Average (Ac)
T-Max-Avg-1	0.922348	0.912879	0.914773	0.918561	0.914773	0.905303	0.914773
T-Max-Avg-2	0.910985	0.909091	0.909091	0.910985	0.907197	0.905303	0.908775
T-Max-Avg-3	0.905303	0.897727	0.905303	0.907197	0.895833	0.909091	0.903409
T-Max-Avg-4	0.912879	0.910985	0.901515	0.912879	0.905303	0.914773	0.909722
T-Max-Avg-5	0.901515	0.892045	0.890152	0.903409	0.901515	0.893939	0.897096
T-Max-Avg-6	0.905303	0.895833	0.909091	0.912879	0.895833	0.899621	0.903093
T-Max-Avg-7	0.892045	0.888258	0.897727	0.893939	0.899621	0.905303	0.896149

- In Table 3, the Max Pooling, Avg Pooling, and T-Max-Avg Pooling methods were tested. In Tables 4 and 5, the values of T were modified, and six different models were trained again, followed by calculating their averages. As seen in the results, the highest performance was achieved using the Max Pooling method. For the T-Max-Pooling method, the best results were obtained when the T parameter was set to 0.7 and the K parameter to 1.

Table 4.6. Batch size: 32, Learing rate: 0.0001, T: 0.8, Pooling size: 4

CEDAR	Exp-1 (Ac)	Exp-2 (Ac)	Exp-3 (Ac)	Exp-4 (Ac)	Exp-5 (Ac)	Exp-6 (Ac)	Average (Ac)
Max	0.9337	0.9223	0.9299	0.9413	0.9318	0.9318	0.9318
Avg	0.9337	0.9280	0.9318	0.9299	0.9318	0.9318	0.9311
T-Max-Avg-1	0.945076	0.950758	0.939394	0.945076	0.946970	0.937500	0.944129
T-Max-Avg-2	0.946970	0.948864	0.945076	0.935606	0.943182	0.939394	0.943182
T-Max-Avg-3	0.935606	0.945076	0.935606	0.943182	0.952652	0.939394	0.941919
T-Max-Avg-4	0.943182	0.935606	0.933712	0.946970	0.945076	0.960227	0.944129
T-Max-Avg-5	0.948864	0.937500	0.945076	0.945076	0.935606	0.946970	0.943182
T-Max-Avg-6	0.933712	0.935606	0.943182	0.937500	0.943182	0.939394	0.938763
T-Max-Avg-7	0.928030	0.937500	0.937500	0.933712	0.939394	0.933712	0.934975
T-Max-Avg-8	0.939394	0.939394	0.941288	0.939394	0.937500	0.939394	0.939394

Table 4.7. Batch size: 32, Learing rate: 0.0001, T: 0.7, Pooling size: 4

CEDAR	Exp-1 (Ac)	Exp-2 (Ac)	Exp-3 (Ac)	Exp-4 (Ac)	Exp-5 (Ac)	Exp-6 (Ac)	Average (Ac)
T-Max-Avg-1	0,9508	0,9375	0,9394	0,9432	0,9527	0,9470	0,9451
T-Max-Avg-2	0,9413	0,9470	0,9432	0,9489	0,9413	0,9489	0,9451
T-Max-Avg-3	0,9470	0,9470	0,9545	0,9508	0,9470	0,9489	0,9492
T-Max-Avg-4	0,9375	0,9432	0,9413	0,9470	0,9413	0,9451	0,9426
T-Max-Avg-5	0,9470	0,9432	0,9356	0,9527	0,9432	0,9432	0,9441
T-Max-Avg-6	0,9489	0,9451	0,9527	0,9451	0,9470	0,9375	0,9460
T-Max-Avg-7	0,9375	0,9451	0,9337	0,9451	0,9432	0,9375	0,9403
T-Max-Avg-8	0,9394	0,9432	0,9451	0,9451	0,9413	0,9413	0,9426

Table 4.8. Batch size: 32, Learing rate: 0.0001, T: 0.6, Pooling size: 4

CEDAR	Exp-1 (Ac)	Exp-2 (Ac)	Exp-3 (Ac)	Exp-4 (Ac)	Exp-5 (Ac)	Exp-6 (Ac)	Average (Ac)
T-Max-Avg-1	0,9432	0,9470	0,9451	0,9413	0,9413	0,9413	0,9432
T-Max-Avg-2	0,9564	0,9508	0,9470	0,9545	0,9489	0,9451	0,9504
T-Max-Avg-3	0,9451	0,9413	0,9261	0,9451	0,9508	0,9432	0,9419
T-Max-Avg-4	0,9356	0,9451	0,9527	0,9489	0,9451	0,9508	0,9463
T-Max-Avg-5	0,9564	0,9413	0,9489	0,9432	0,9394	0,9356	0,9441
T-Max-Avg-6	0,9432	0,9413	0,9527	0,9470	0,9413	0,9470	0,9454
T-Max-Avg-7	0,9470	0,9489	0,9356	0,9470	0,9413	0,9337	0,9422
T-Max-Avg-8	0,9470	0,9545	0,9337	0,9394	0,9451	0,9337	0,9422

- In Table 4.6, it is evident that a pooling size of 4 demonstrates better performance compared to a pooling size of 3. Furthermore, the k values of 1 and 4 yield the same average accuracy, highlighting their comparable effectiveness under these conditions.

Table 4.9. Batch size: 32, Learing rate: 0.0001, T: 0.8, Pooling size: 3

UTSig	Exp-1 (Ac)	Exp-2 (Ac)	Exp-3 (Ac)	Exp-4 (Ac)	Exp-5 (Ac)	Exp-6 (Ac)	Average (Ac)
Avg	0.8466	0.8382	0.8484	0.8388	0.8412	0.8412	0.8424
Max	0.8496	0.8430	0.8514	0.8502	0.8551	0.8466	0.8493
T-Max-Avg-1	0.8176	0.8080	0.8134	0.8170	0.8213	0.8037	0.8135
T-Max-Avg-2	0.8309	0.8134	0.8152	0.8158	0.8146	0.8152	0.8175
T-Max-Avg-3	0.8188	0.8188	0.8080	0.8146	0.8237	0.8086	0.8154
T-Max-Avg-4	0,8104	0,8110	0,8170	0,8122	0,8128	0,8170	0,8134
T-Max-Avg-5	0,8219	0,8080	0,8225	0,8019	0,8080	0,8194	0,8136
T-Max-Avg-6	0,8164	0,8104	0,8068	0,8050	0,7971	0,8050	0,8068
T-Max-Avg-7	0,8037	0,8122	0,8098	0,8158	0,8068	0,8025	0,8085
T-Max-Avg-8	0,8134	0,8140	0,8031	0,8092	0,8176	0,8219	0,8132

Table 4.10. Batch size: 32, Learing rate: 0.0001, T: 0.7, Pooling size: 3

UTSig	Exp-1 (Ac)	Exp-2 (Ac)	Exp-3 (Ac)	Exp-4 (Ac)	Exp-5 (Ac)	Exp-6 (Ac)	Average (Ac)
T-Max-Avg-1	0.8019	0.8128	0.8134	0.8068	0.8104	0.8098	0.8091
T-Max-Avg-2	0.8098	0.8122	0.8170	0.8164	0.8182	0.8207	0.8157
T-Max-Avg-3	0.8068	0.8213	0.8062	0.8068	0.8170	0.8243	0.8137
T-Max-Avg-4	0.8194	0.8200	0.8098	0.8194	0.8152	0.7965	0.8133
T-Max-Avg-5	0.8056	0.8104	0.8098	0.7965	0.8188	0.8032	0.8073
T-Max-Avg-6	0,8104	0,8140	0,8104	0,8092	0,8037	0,8037	0,8086
T-Max-Avg-7	0,8110	0,8170	0,8056	0,8104	0,8037	0,8019	0,8083
T-Max-Avg-8	0,8001	0,8098	0,8092	0,8074	0,8080	0,8104	0,8075

Table 4.11. Batch size: 32, Learing rate: 0.0001, T: 0.6, Pooling size: 3

UTSig	Exp-1 (Ac)	Exp-2 (Ac)	Exp-3 (Ac)	Exp-4 (Ac)	Exp-5 (Ac)	Exp-6 (Ac)	Average (Ac)
T-Max-Avg-1	0.8116	0.8152	0.8188	0.8056	0.8200	0.8019	0.8121
T-Max-Avg-2	0.8080	0.8134	0.8056	0.8194	0.8188	0.8134	0.8131
T-Max-Avg-3	0.8182	0.8074	0.8050	0.8182	0.8098	0.8164	0.8125
T-Max-Avg-4	0.8098	0.8110	0.8080	0.8104	0.8104	0.8140	0.8106
T-Max-Avg-5	0.8007	0.8164	0.8116	0.8116	0.8019	0.8188	0.8101
T-Max-Avg-6	0,8164	0,8104	0,8068	0,8050	0,7971	0,8050	0,8068
T-Max-Avg-7	0,8037	0,8122	0,8098	0,8158	0,8068	0,8025	0,8085
T-Max-Avg-8	0,8134	0,8140	0,8031	0,8092	0,8176	0,8219	0,8132

Table 4.12. Batch size: 32, Learing rate: 0.0001, T: 0.8, Pooling size: 4

UTSig	Exp-1 (Ac)	Exp-2 (Ac)	Exp-3 (Ac)	Exp-4 (Ac)	Exp-5 (Ac)	Exp-6 (Ac)	Average (Ac)
Max	0.8273	0.8309	0.8261	0.8345	0.8273	0.8207	0.8278
Avg	0.8533	0.8569	0.8514	0.8623	0.8502	0.8575	
T-Max-Avg-1	0,8460	0,8448	0,8412	0,8569	0,8521	0,8406	0,8469
T-Max-Avg-2	0,8478	0,8478	0,8496	0,8502	0,8388	0,8527	0,8478
T-Max-Avg-3	0,8466	0,8430	0,8569	0,8442	0,8394	0,8351	0,8442
T-Max-Avg-4	0,8448	0,8382	0,8424	0,8430	0,8412	0,8357	0,8409
T-Max-Avg-5	0,8448	0,8406	0,8376	0,8357	0,8382	0,8490	0,8410
T-Max-Avg-6	0,8484	0,8382	0,8460	0,8496	0,8339	0,8442	0,8434
T-Max-Avg-7	0,8412	0,8514	0,8412	0,8412	0,8267	0,8261	0,8380

Table 4.13. Batch size: 32, Learing rate: 0.0001, T: 0.7, Pooling size: 4

UTSig	Exp-1 (Ac)	Exp-2 (Ac)	Exp-3 (Ac)	Exp-4 (Ac)	Exp-5 (Ac)	Exp-6 (Ac)	Average (Ac)
T-Max-Avg-1	0.847826	0.838164	0.850242	0.838768	0.854469	0.858696	0.848027
T-Max-Avg-2	0.847222	0.856280	0.849034	0.846618	0.856884	0.852657	0.851449
T-Max-Avg-3	0.853261	0.850242	0.847222	0.839372	0.854469	0.844203	0.848128
T-Max-Avg-4	0.842391	0.848430	0.836957	0.849638	0.839976	0.852053	0.844907
T-Max-Avg-5	0.840580	0.832729	0.838164	0.837560	0.832126	0.842391	0.837259
T-Max-Avg-6	0.836353	0.845411	0.846618	0.849638	0.846014	0.843599	0.844605
T-Max-Avg-7	0.849638	0.844203	0.852053	0.841184	0.852053	0.836957	0.846014

Table 4.14. Batch size: 32, Learing rate: 0.0001, T: 0.6, Pooling size: 4

UTSig	Exp-1 (Ac)	Exp-2 (Ac)	Exp-3 (Ac)	Exp-4 (Ac)	Exp-5 (Ac)	Exp-6 (Ac)	Average (Ac)
T-Max-Avg-1	0.8521	0.8472	0.8521	0.8436	0.8454	0.8424	0.8471
T-Max-Avg-2	0.8442	0.8502	0.8382	0.8533	0.8460	0.8575	0.8482
T-Max-Avg-3	0.8394	0.8466	0.8521	0.8545	0.8484	0.8569	0.8496
T-Max-Avg-4	0.8430	0.8382	0.8454	0.8466	0.8436	0.8454	0.8437
T-Max-Avg-5	0.8430	0.8514	0.8442	0.8430	0.8406	0.8508	0.8455
T-Max-Avg-6	0.8563	0.8394	0.8382	0.8412	0.8430	0.8454	0.8439
T-Max-Avg-7	0.8399	0.8472	0.8375	0.8393	0.8454	0.8405	0.8416

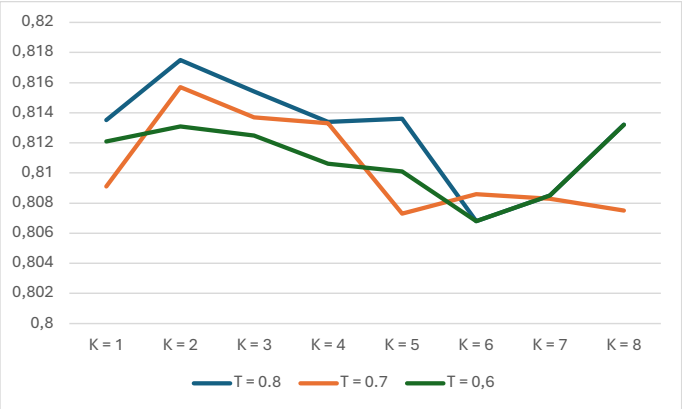


Figure 4.4.1 Vizilation parameter when pooling-size = 3 and dataset UTSig

- When the pooling size is set to 3, the UTSig dataset achieves its highest performance at K = 2 across all T values.

Figure 4.4.1 Vizilation parameter when pooling-size = 4 and dataset UTSig

- The outputs obtained in Tables 4.9 - 4.11 are illustrated in Figure 4.4.1. The highest accuracy for the T-Max-Avg pooling method was achieved when T = 0.7 and k = 2.
- The max pooling method achieved an average result of 0.8278, as shown in Figure 4.10. The T-Max-Avg pooling method demonstrated superior performance on the UTSig dataset.

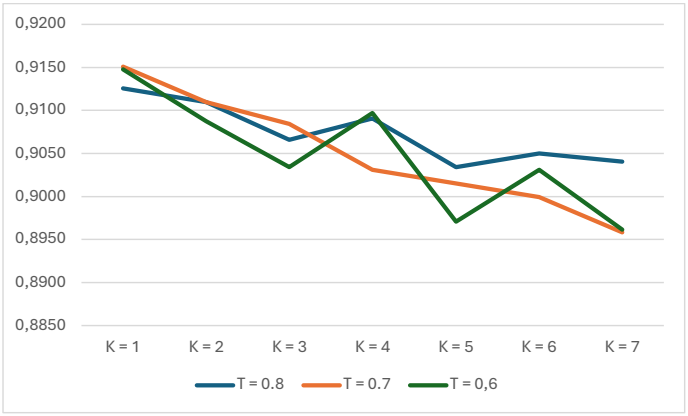


Figure 4.4.2 Vizilation parameter when pooling-size = 3 and dataset CEDAR

- When training the model on the CEDAR dataset with a pooling size of 3, it is observed that an increase in the k value results in a decline in model performance. However, the highest performance on this dataset was also achieved at T = 0.7.

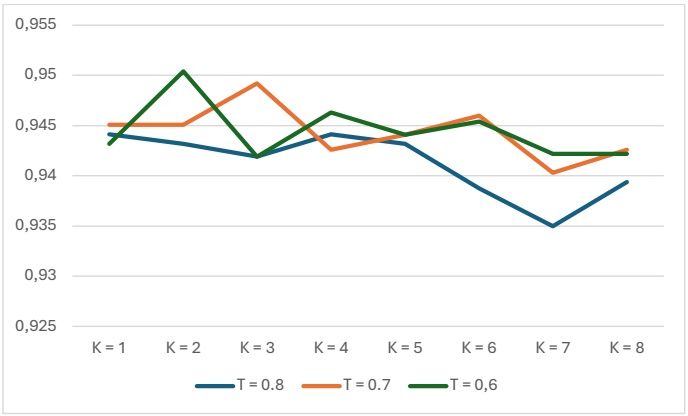


Figure 4.4.3 Vizilation parameter when pooling-size = 4 and dataset CEDAR

- It can be observed from this graph that the highest accuracy was achieved at T=0.6 and k=2. Accuracy values varied across different T values for distinct k values.
- While the Max Pooling method achieved an average accuracy of =0.9318, the T-Max-Avg method increased this accuracy to Acc=0.9504.

5. CONCULATION AND DISCUSSION

This study underscores the importance of accurate offline signature verification systems, addressing the limitations of traditional manual methods. These systems are pivotal in ensuring authentication and minimizing error risks, particularly in high-security applications. The research presented herein has introduced a novel convolutional neural network (CNN) architecture, specifically designed to enhance signature verification accuracy and efficiency.

A key contribution of this work is the introduction and evaluation of the T-Max-Avg pooling method. This approach outperformed traditional pooling techniques in terms of accuracy, demonstrating its effectiveness in retaining critical features while mitigating noise. Experimental results obtained from the UTSig and CEDAR datasets validate the superior performance of the proposed method. Notably, a pooling size of 4 combined with specific parameter configurations, such as T=0.7 and K=2, achieved optimal results. Additionally, the results emphasize that the custom architecture tailored for this task outperforms conventional predefined models.

The findings also highlight that while increasing the number of epochs may improve training accuracy, it can lead to overfitting, as observed in specific cases. This underscores the necessity of carefully balancing model complexity and training parameters.

Future research could explore the integration of hybrid methods, combining the strengths of traditional machine learning and deep learning approaches. This may further enhance feature extraction and reduce computational complexity. Additionally, incorporating multimodal biometric systems that integrate signature verification with other identification methods could bolster overall security.

In conclusion, this study successfully demonstrates the potential of innovative pooling techniques and custom CNN architectures in advancing offline signature verification systems. The results not only contribute to the field but also provide a foundation for future advancements in secure and efficient authentication technologies.

REFERENCES

[1] Otsu, N. (1975). A threshold selection method from gray-level histograms. Automatica, 11(285-296), 23-27.

[2] Jain, A. K., Ross, A., & Prabhakar, S. (2004). An introduction to biometric recognition. IEEE Transactions on circuits and systems for video technology, 14(1), 4-20.

[3] Gholamalinezhad, H., & Khosravi, H. (2009). Pooling methods in deep neural networks, a review. arXiv 2020. arXiv preprint arXiv:2009.07485.

[4] Scherer, D., Müller, A., & Behnke, S. (2010, September). Evaluation of pooling operations in convolutional architectures for object recognition. In International conference on artificial neural networks (pp. 92-101). Berlin, Heidelberg: Springer Berlin Heidelberg.

[5] Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10) (pp. 807-814).

[6] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).

[7] Graham, B. (2014). Fractional max-pooling. arXiv preprint arXiv:1412.6071.

[8] Diederik, P. K. (2014). Adam: A method for stochastic optimization.

[9] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

[10] Yu, D., Wang, H., Chen, P., & Wei, Z. (2014). Mixed pooling for convolutional neural networks. In Rough Sets and Knowledge Technology: 9th International Conference, RSKT 2014, Shanghai, China, October 24-26, 2014, Proceedings 9 (pp. 364-375). Springer International Publishing.

[11] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

[12] Kartheeswaran, S., & Durairaj, D. D. C. (2015, November). Minimize the mean square error by data segregation approach for back-propagation artificial neural network with adaptive learning based image reconstruction in electron magnetic resonance imaging tomography. In 2015 Online International Conference on Green Engineering and Technologies (IC-GET)(pp. 1-5). IEEE.

[13] Zhang, Z., Liu, X., & Cui, Y. (2016, December). Multi-phase offline signature verification system using deep convolutional generative adversarial networks. In 2016 9th international Symposium on Computational Intelligence and Design (ISCID)(Vol. 2, pp. 103-107). IEEE.

[14] Cui, Z., Chen, W., & Chen, Y. (2016). Multi-scale convolutional neural networks for time series classification. arXiv preprint arXiv:1603.06995.

[15] Goodfellow, I. (2016). Deep learning.

[16] Hershey, S., Chaudhuri, S., Ellis, D. P., Gemmeke, J. F., Jansen, A., Moore, R. C., ... & Wilson, K. (2017, March). CNN architectures for large-scale audio classification. In 2017 ieee international conference on acoustics, speech and signal processing (icassp) (pp. 131-135). IEEE.

[17] Hirunyawanakul, A., Bunrit, S., Kerdprasop, N., & Kerdprasop, K. (2019). Deep learning technique for improving the recognition of handwritten signature. Int. J. Inform. Electron. Eng, 9(4), 72-78.

[18] Noor, D. F., Li, Y., Li, Z., Bhattacharyya, S., & York, G. (2019). Multi-scale gradient image super-resolution for preserving SIFT key points in low-resolution images. Signal Processing: Image Communication, 78, 236-245.

[19] Kao, H. H., & Wen, C. Y. (2020). An offline signature verification and forgery detection method based on a single known sample and an explainable deep learning approach. Applied Sciences, 10(11), 3716.

[20] Ruby, U., & Yendapalli, V. (2020). Binary cross entropy with deep learning technique for image classification. Int. J. Adv. Trends Comput. Sci. Eng, 9(10).

[21] Zheng, Y., Iwana, B. K., Malik, M. I., Ahmed, S., Ohyama, W., & Uchida, S. (2021). Learning the micro deformations by max-pooling for offline signature verification. Pattern Recognition, 118, 108008.

[22] Biswas, S., & Wong, B. M. (2022). High-temperature decomposition of diisopropyl methylphosphonate (dimp) on alumina: Mechanistic predictions from ab initio molecular dynamics. arXiv preprint arXiv:2203.08035.

[23] Kaur, H., & Kumar, M. (2023). Signature identification and verification techniques: state-of-the-art work. Journal of Ambient Intelligence and Humanized Computing, 14(2), 1027-1045.

[24] Damadi, S., Moharrer, G., Cham, M., & Shen, J. (2023, June). The Backpropagation algorithm for a math student. In 2023 International Joint Conference on Neural Networks (IJCNN) (pp. 01-09). IEEE.

[25] Badie, A., & Sajedi, H. (2024). Offline handwritten signature authentication using Graph Neural Network methods. International Journal of Information Technology, 1-11.

[26] Zhao, L., & Zhang, Z. (2024). A improved pooling method for convolutional neural networks. Scientific Reports, 14(1), 1589.

[27] Zhao, X., Wang, L., Zhang, Y., Han, X., Deveci, M., & Parmar, M. (2024). A review of convolutional neural networks in computer vision. Artificial Intelligence Review, 57(4), 99.

CURRİCULUM VİTAE

Name Surname	Hasan Basri Uzun
Birth	21.10.2002
High School	2015 – 2020 Tekirdağ Anadolu İmam-Hatip High School
Internship	Ziraat Teknoloji – İstanbul – Data Science