

Siamese Network with Gabor Filter for Recognizing Handwritten Digits

Rauzan Sumara[1(✉)] and Ivan Luthfi Ihwani[2,3]

[1] Faculty of Mathematics and Information Science, Warsaw University of Technology, Warsaw 00661, Poland
rauzan.sumara.dokt@pw.edu.pl
[2] Department of Mathematics, Universitas Gadjah Mada, Yogyakarta 55281, Indonesia
[3] Department of Mathematics, National Central University, Taoyuan City 32001, Taiwan

Abstract. Even though significant progress has been made in the field of pattern recognition in recent years, handwritten digit recognition remains an exciting challenge. Due to its widely possible application, this issue has attracted much attention. Nowadays, different methods are available for handwritten digit recognition. The current demand is for researchers to find new techniques for better performing handwritten digit recognition problems. Therefore, this paper aims to propose a new approach, the Siamese network with Gabor filter, for handwritten digit recognition. Inspired by several studies on the Siamese network and Gabor filter separately, which have already achieved superb performances, this research will bring out the best qualities of their fusion. The computational experiments have been conducted on a handwritten digit image of the MNIST dataset. Empirically, the results implied that the proposed Siamese network with the Gabor filter algorithm outperformed the classical Siamese network and other existing methods in terms of accuracy.

Keywords: Gabor Filter · Handwritten Digit · Recognition · Siamese Network

1 Introduction

Recently, handwritten digit recognition has been one of the most challenging issues. Many researchers have looked into this extensively in recent years. In empirical practice, high recognition accuracy and reliability are of the essence in handwritten digit recognition systems. The recognition system is even more complex because each person's writing pattern and style are unique. Handwritten digit recognition remains an ongoing issue. The current demand is for how researchers can find new techniques for handwritten digit recognition problems. The methods are to improve the performance in terms of accuracy.

Multiple algorithms have been developed for handwritten recognitions. Several of them used various techniques, including k-nearest neighbor (kNN) [1],

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023, LNCS 13957, pp. 32–47, 2023.
https://doi.org/10.1007/978-3-031-36808-0_3

support vector machines (SVM) [2], neural networks (NN) [3], and convolutional neural networks (CNN) [4]. In [5], a new approach to identifying handwritten digits was presented. The writer suggested an ensemble classification strategy that employs bagging to enhance accuracy. A hybrid system incorporating bagged-radial basis function (RBF), bagged-SVM, and RBF-SVM was developed and assessed using various standard datasets. The authors in [6] suggested a NN and extreme learning machine (ELM) based method for handwritten digit recognition. The convolutional recurrent neural network (CRNN) introduced by the authors in [7] effectively merged the benefits of deep convolutional neural networks (CNN) and recurrent neural networks (RNN). Moreover, this approach proved to be valuable for the task of categorizing text within scenes.

Numerous classification algorithms were introduced in [8], such as naive Bayes, multilayer perception (MLP), Bayes-net, J48, SVM, random trees, and random forest (RF). The results showed that MLP exceeds other classifiers in handwritten recognition performance. A unique method for feature computation, as described in reference [9], employed structural and statistical approaches. The initial step involved preprocessing the input data through binarization, cropping, and normalization. Subsequently, four distinct sets of features were computed using methods such as a cavity, zoning, profile projection, and Freeman chain coding (FFC). Finally, kNN was utilized for key point classification. However, it is important to note that all these previous techniques require extensive computations.

In this study, we formulated a new approach using a Siamese network with a Gabor filter for handwritten digit recognition to overcome the challenge. Several studies on the Gabor filter, such as that conducted by [10], showed that using the Gabor filter can improve the accuracy rate of the CNN architecture in the case of vision-based vehicle recognition. Other evidence by [11] and [12] combine existing CNN architectures, e.g., ResNet, AlexNet, VGG16, and InceptionV3, with the Gabor filter to get high accuracy. Because of their complexity, performing those existing CNN architectures is tedious and time-consuming. Therefore, the Siamese network has a strategic role here. This network is not only able to reduce training time but also can be trained from small data [13]. Because of that, the Siamese network is more popular nowadays. The architecture has been successfully applied to recognizing handwritten signatures by [14]. With the same analogy, this architecture can be used for other handwritten data, such as the handwritten digits on the MNIST dataset. Inspired by those particular ideas, our research aims to fill the gap. We concentrate our research on their fusion of the Siamese network with the Gabor filter as a kernel on the network's first convolution layer. After the embedding process, it is forwarded to a classifier using kNN.

We also organize the rest of the chapter as follows: After this introduction, we describe the related work in Sect. 2 and the proposed network architecture in Sect. 3, then specify empirical settings consisting of three aspects: data set descriptions, a baseline model, and hyperparameters in Sect. 4. Mean-

while, Sect. 5 describes the results and discussion, and finally, we outline some conclusions in Sect. 6.

2 The Related Work

Multiple algorithms have been developed for handwritten recognitions on the comprehensive investigation of several research papers regarding the dataset used, accuracy performance, and limitations. In [6], the authors proposed a technique for classifying MNIST handwritten digits using a combination of NN and ELM. The findings indicated that ELM achieved superior classification accuracy and required less processing time compared to NN. It achieved an accuracy of 98.40%. Additionally, in [7], the authors introduced a CRNN that merged the strengths of CNN and RNN. This approach in [7] was also employed for classifying MNIST handwritten digits and achieved an accuracy of 98.70%. However, it should be noted that this approach entails computational complexity.

A different approach to recognizing handwritten digits was presented in [5]. The author suggested an ensemble classification technique that utilizes bagging to enhance accuracy. A hybrid system consisting of bagged-RBF, bagged-SVM, and RBF-SVM was developed and assessed using various real-world and benchmark datasets. The performance evaluations illustrated that the proposed hybrid RBF-SVM classifier delivers favorable classification outcomes. It achieved an accuracy of 98.00%. However, it should be mentioned that it is time-consuming to implement this approach because of its complexity.

Numerous classification algorithms were introduced in [8], such as naive Bayes, MLP, Bayes-net, J48, SVM, random trees, and RF. The results showed that MLP outperforms other classifiers in recognition performance. A novel technique for computing features was introduced in [9], utilizing structural and statistical approaches. The initial stage involved preprocessing the input data to binarize, crop, and normalize it. Subsequently, four distinct feature sets were generated using cavity, zoning, FFC, and profile projection methods. The classification of key points was then carried out using the kNN algorithm. Notably, this approach [9] achieved an impressive recognition accuracy of 95.00% on the MNIST database when using FFC. However, its performance on more challenging databases may be limited.

In addition, still in cases of MNIST handwritten digits, [2] proposed a hybrid approach, which utilized a CNN for feature extraction and a hybrid classifier for classification. The resulting system achieved an accuracy of 94.40%. [15] used the sparse group lasso (SG-L1) for feature selection and a deep NN for classification, achieving a classification accuracy of 97.00%. [16] applied spiking neural networks (SNN) with normalized approximate descent (NormAD) as the learning algorithm and achieved 98.40% classification accuracy. Similarly, [17] used the spike timing dependent plasticity (STDP) based deep SNN training and achieved a classification accuracy of 98.40%. Finally, [18] utilized fractional order backpropagation (FOBP) neural networks and population extremal optimization (PEO) for network optimization, resulting in an overall classification accuracy

of 96.54%. Nevertheless, from all references presented earlier, their accuracy and computing complexity could still be improved.

3 The Siamese Network with Gabor Filter System

Our approach was intended to integrate the Siamese network and Gabor filter. The Gabor filter theory will be introduced in Subsect. 3.1, followed by the Siamese network in Subsect. 3.2. Then, their combination will be discussed in Subsect. 3.3 at the end of this section.

3.1 Gabor Filters

A Gabor function is defined as the product of a sinusoid with a Gaussian distribution in mathematical form as follows,

$$g_{\mathbf{\Theta}}(x,y) = \exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right)\exp\left(i\left(2\pi\frac{x'}{\lambda}+\phi\right)\right) \tag{1}$$

and

$$\begin{aligned} x' &= x\cos(\theta)+y\sin(\theta) \\ y' &= -x\sin(\theta)+y\cos(\theta) \end{aligned} \tag{2}$$

while $\mathbf{\Theta} = \{\theta, \lambda, \phi, \gamma, \sigma\}$ serves as the parameter and i represents the imaginary integer. Notice thus the equation (1) could be written in two forms, namely the real form (3)

$$g_{\mathbf{\Theta}}^{\text{real}}(x,y) = \exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda}+\phi\right) \tag{3}$$

and imaginary form (4)

$$g_{\mathbf{\Theta}}^{\text{img}}(x,y) = \exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right)\sin\left(2\pi\frac{x'}{\lambda}+\phi\right). \tag{4}$$

Both components can be utilized as a filter in a convolution layer, but we restrict our experiment by using the real form only. The real form (3) contains five parameters, and every parameter has a different purpose and can take numerical values within a given interval. Additionally, we summarize the example of interval values for every parameter below,

Orientation $\boldsymbol{\theta}$. The direction of the Gabor filter is specified by the parameter θ. The acceptable values are ranging from 0 and 2π. Gabor filters with several values of θ which are 0, $\pi/8$, $\pi/4$, $3\pi/8$, $\pi/2$, $5\pi/8$, $3\pi/8$ and $7\pi/8$ are displayed in Fig. 1(a). It showed the bands of Gabor filters spin in response to a change in orientation parameter.

Wavelength λ. In general, acceptable values of λ should larger than or equal to 2. Considering that the Gabor function should be zero at $\lambda = 2$ and $\phi = \pi/2$, we will ensure in this experiment that λ is greater than 2. In addition, to decrease the possibility of undesirable effects at the picture boundaries, it is necessary to set the wavelength λ will be not greater than the pixels of the image. Therefore, to determine the maximum value of λ, we simply define $C_\lambda = \min(I_{\text{width}}, I_{\text{height}})$, where I_{width} and I_{height} are the width and height of the input picture. Different values of λ which are $2, 4, 6, 8, 10, 12, 14$, and 16 showed in Fig. 1(b). The bandwidth increases as the increasing λ.

Phase offset ϕ. The Gabor function has a parameter of ϕ in which it represents the argument for the sine or cosine factor. It has to be real numbers between $-\pi$ and π. If the value of ϕ is 0, it is in accordance with center-symmetric 'center-on' and if the value is π, it is in accordance with non-center-symmetric 'center-off' functions. In addition, the Gabor function will be in accordance with anti-symmetric functions while the parameters of ϕ are $-\pi/2$ and $\pi/2$; otherwise, it is similar to asymmetric functions. Different values of ϕ, from 0 to $\frac{7\pi}{8}$, are illustrated in Fig. 1(c).

Aspect ratio γ. This parameter defines the ellipticity of the Gabor function's support. This support will be round when the value of γ equals 1, but this support is stretched in the position of the perpendicular lines of the function when the value of γ is less than 1. Acceptable values of γ are real values within interval $0 < \gamma \leq 1$. As an illustration, we use several values of λ, which are $0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 0.8$, and 1.0 to visualize Gabor filters shown in Fig. 1(d) We may observe that when the aspect ratio increases, the band length decreases.

Generate σ. The parameter σ is well known as a deviation in a Gaussian distribution. In the Gabor filters, the value of σ determines the number of bands. As we can see from Fig. 1(e), it depicts the Gabor filters when the values of σ are set to be $0.1, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0$, and 3.5.

Further details about parametrization can be found in [19]. Traditionally, we must create a collection of Gabor filters with various combinations of their parameters in order to use them. In applying Gabor filters, the values of the parameters of the Gabor function need to be properly decided. We explained those in Subsect. 4.3 about the hyperparameters of the Gabor filters. Because this type of pipeline utilizes only hand-designed unique parameters, it may be suboptimal for particular inputs. The Gabor filter banks are derived from a comprehensible Gabor function. In contrast to classical convolution, its filters are created at random, and they are independent of one another. This is what makes a significant difference between them. In addition, all parameters in the Gabor filters have distinct meanings, whereas those in a classical convolution do not.

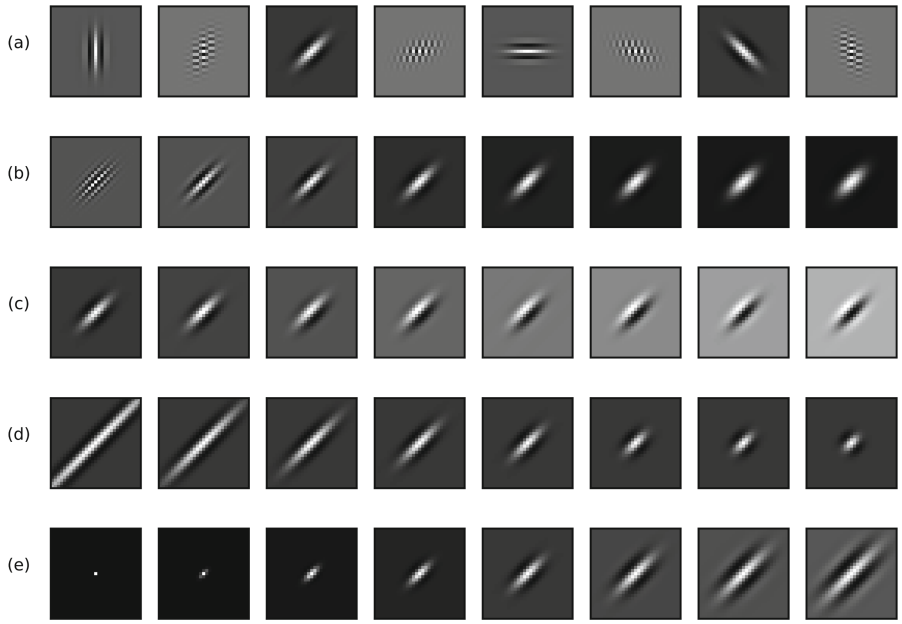


Fig. 1. Images with varying Gabor filter settings: (a) orientation θ, (b) wavelength λ, (c) phase offset ϕ, (d) aspect ratio γ, and (e) various values of σ.

3.2 Siamese Networks with Triplet Loss

The Siamese neural network is a category of network designs that typically consists of two or more identical subnetworks. The configuration, parameters, and weights of each subnetwork are identical because the parameter update is replicated on all subnetworks. [20] has reported applications of the Siamese networks not only in speech signal processing and text mining but also in dimensionality reduction [21] and face verification [22] in which the framework has shown promising results. Generally, Triplet loss is the loss function operated in Siamese networks [23]. As first introduced in [22], it was used for recognizing faces by integrating triplet loss into the deep convolutional network. To date, the triplet loss has been extensively used in a variety of image processing, such as image retrieval [24–26] as well as person re-identification [27,28].

Figure 2 shows the framework of the Siamese networks with Triplet loss. By defining three embeddings (subnetworks), there must be three input sample images, namely an anchor (x_a), a positive (x_p), and a negative (x_n) sample image. The idea behind the Triplet loss is to ensure that an anchor image has a closer distance to a positive image in the embedding space (the positive has the same label as the anchor) than a negative image with a different label. According

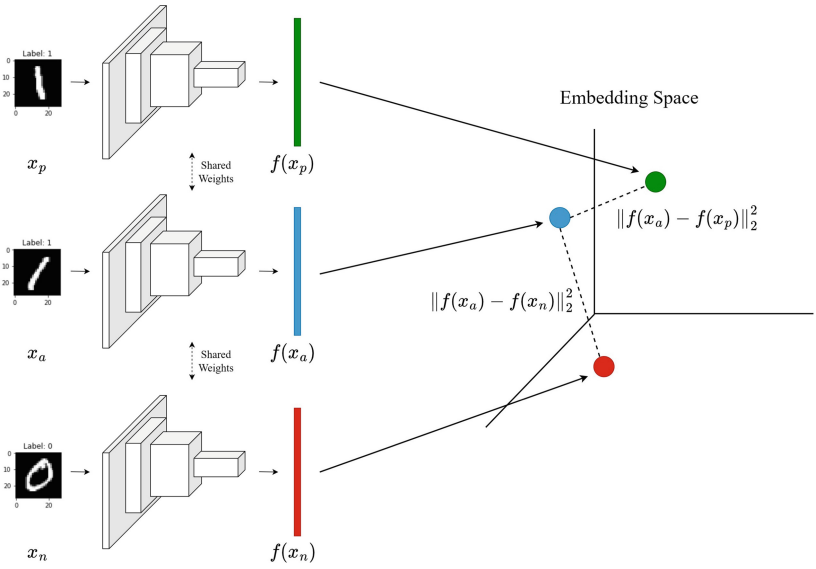


Fig. 2. Siamese network architecture with Triplet loss.

to [22], we can write a formula for triplet (a, p, n) loss as follow,

$$L_{triplet} = max\left(d_p - d_n + \alpha, 0\right) \tag{5}$$

where $d_p = \|f\left(x_a\right) - f\left(x_p\right)\|_2^2$, $d_n = \|f\left(x_a\right) - f\left(x_n\right)\|_2^2$, and α represents a required margin to maintain a large distance between the positive and negative image. Based on the definition of the loss, there are three categories of triplets: first, easy triplets, which have a loss of 0 since $d_p + \alpha < d_n$. Second, hard triplets where the positive is further from the anchor than the negative, $d_n < d_p$, and third, semi-hard triplets where the positive is closer to the anchor than the negative but still hard because the distance is still inside the margin, $d_p < d_n < d_p + \alpha$. Selecting the right triplets is essential to achieving fast convergence. Therefore, we will use semi-hard triplets, known as the best results in [22]. In order to efficiently identify these triplets within each batch, we apply an online strategy and train from the semi-hard samples only.

We employed the architecture influenced by [29] to solve this challenge. In order to facilitate the understanding of the network, Table 1 contains a comprehensive listing of the parameters used to create the CNN layers. Mainly, we will have 2 convolution layers and 2 pooling layers with dropout regularization in the networks. Every layer creates an output with the size of N × H × W, where N, H, and W are the number of filters, the height, and the width of the output, respectively. The stride parameter is the number of movements over an image, and the size parameter refers to the pooling operation.

We use the Rectified Linear Units (ReLU) activation function for all convolutional layers throughout the network but not on the last dense layer. The first

Table 1. Overview of the Siamese network architecture.

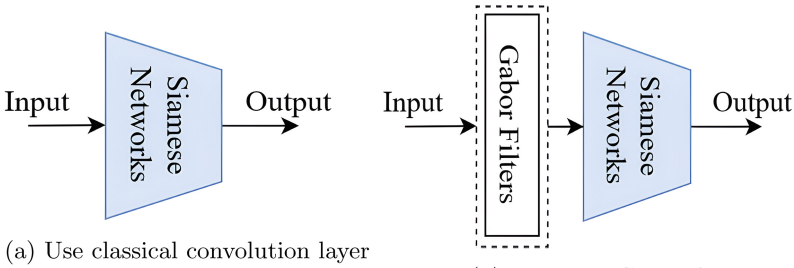
Layers	An output size	Parameters
Convolution 1	64 × 28 × 28	Size = 2, stride = 1, padding = 'same'
MaxPooling + Dropout	64 × 14 × 14	Size = 2, p = 0.3
Convolution 2	32 × 14 × 14	Size = 2, stride = 1, padding = 'same'
MaxPooling + Dropout	32 × 7 × 7	size = 2, p = 0.3
Flattening	1568	-
Fully Connected	256	-

convolutional layer creates the 28 × 28 output size with 64 filters, adding a stride of 1 pixel and pooling layer along with a Dropout rate equal to 0.3. Next, the second convolutional layer is performed from the output of the first convolutional layer. It creates 14 × 14 output with 32 filters, adds a stride of 1 pixel and pooling layer along with the Dropout rate equal to 0.3 again. In conclusion, there are 1568 connected units (nodes) in the flattened layer, while there are 256 nodes in the fully connected layer. This suggests that the latest learned feature vector from either side of the Siamese Network has a size of 256. This indicates that the latest embeddings on each subnetwork of Siamese networks are 256 in size.

3.3 The Architecture of the Proposed Model

Hereinafter in this topic, we will illustrate how Siamese networks may benefit from the Gabor filter when used for handwritten recognition. Our suggested architecture is constructed by inserting the first convolutional layer of the Siamese networks with the Gabor filter. We will also perform the Siamese networks without employing the Gabor filter as a basis for comparison. Therefore, there will be two variants of architecture, with and without the Gabor filter, based on the Siamese networks. Figure 3 gives a clear idea of two variant structures. As mentioned in the previous subsection, our Siamese network consists of two convolutional layers and two pooling layers with dropout regularization in the networks.

Firstly, we will use the original architecture of Siamese networks, let it be known as "network 1". Secondly, we insert the Gabor filter into the first convolution layer in the Siamese network; let it be known as "network 2". Lastly, Both network 1 and network 2 are similar, whereas the only difference is in the first convolution layer, where network 1 will use default filters from the 2D convolution layer, while network 2 will use Gabor filters. Again, for a detailed understanding of the network, read Table 1. According to the previous section, we mentioned triplet loss in order to train those networks. We should notice that there are several types of triplets in accordance with its definition. However, we will only use the semi-hard triplet loss function. In conclusion, the embedding output, after completing the training process, can be taken as a new feature vector for training a baseline classifier such as a k-NN. Finally, the baseline classifier performs the recognition task to predict the labels on the testing images.



(a) Use classical convolution layer

(b) With the Gabor filters

Fig. 3. Proposed Siamese network.

4 Empirical Settings

The presented experiment evaluates the Siamese networks with the Gabor filter for MNIST handwritten digits. The experimental framework is presented in three aspects: dataset description, a baseline classifier, and hyperparameters.

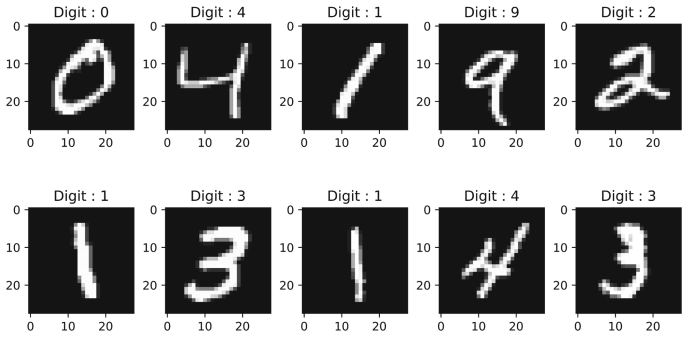


Fig. 4. Example of MNIST Dataset.

4.1 Data Set Description

In the current study, the suggested classification system has been trained and tested using MNIST, a well-known benchmark dataset of handwritten digits. The 10 digits are depicted in handwritten form in the dataset (0–9). The MNIST dataset is a portion of a larger dataset made available by the National Institute of Standards and Technology (NIST). The dataset has a total of 70000 instances, which includes 60000 instances for training and 10000 instances for testing. Initially, a 20×20 pixel matrix was used to represent the images that were contained in the NSIT database. However, these handwritten images were normalized into 28×28 pixel images [30]. We have presented a few examples of handwritten digits from the MNIST dataset in Fig. 4. Furthermore, it should be highlighted that

we used the MNIST dataset because it is a complex and challenging one due to its inherent variations in digit size, scale, and angle. The images in the dataset are also affected by issues such as blurring and variations in intensity, making it still interesting for classification study.

4.2 A Baseline Classifier

We use the kNN for a baseline classification model. This algorithm can categorize the data based on training features. Regardless of the learning methodologies, the kNN algorithm uses basic computations, which makes it the easiest categorization method. Objects are categorized depending on the votes of their neighbors, who are symbolized by the letter k. In kNN, a sample is categorized into the class with the most votes. K-NN calculates the distance, specifically using Euclidian distance, between the feature vector values of each training set and the feature vector values of the testing set. The training vector is an array with several dimensions. The training pictures' class labels and feature values are each included in a row of an array, but the test vector only includes feature values. Each row in the test set receives a classification label according to Euclidian distance measurements and the number of neighbors (k) taken into account. The elbow method, one of the most popular methods, can help us select the optimal number of k.

4.3 Hyperparameters

In this subsection, we will specify how the hyperparameters will be ranged in our model. It consists of hyperparameters for Gabor filters and hyperparameters for learning setup.

Gabor Filters. We used real form (3) in order to create the number of Gabor filter banks. There are five parameters in the Gabor function such as θ, λ, ϕ, γ, and σ. In experiments, we take the value of θ be $0, \pi/4, \pi/2, 3\pi/4, \pi, 5\pi/4, 3\pi/2$, and 2π. Parameter λ is set to be 7, 11, 16, and 22. Thus, the parameter of phase offset ϕ is specified as 0 and $\pi/2$. Finally, we also set the aspect ratio $\gamma = 0.25$ and the values of $\sigma = 1.75$ in the whole setting. Through all possible combinations of elements in θ, λ, ϕ, γ, and σ, we will obtain 64 filters. These filters are then operated to the first convolutional layer of the Siamise networks.

Learning and Experimental Setup. Here is the common setup we followed to conduct experiments: The entire experiment was coded with Python 3.8.5. We also used fundamental libraries such as pandas, numpy, scikit-learn, keras and tensorflow. The training was done using standard PCs with a specification of 8 Core CPUs, 16 GB RAM, and GeForce GTX 1660 Ti and Windows 11. Next, because the datasets are already split into train and test sets, the train set was used to fit the model. We trained the networks using adaptive moment

estimation (Adam) to optimize parameters to get the smallest loss and back-propagation, the method to calculate the gradient for each layer. These are widely used optimization algorithms for deep neural networks. By applying the Adam optimizer and the back-propagation algorithm together, we can update the weights of a neural network to minimize the loss function 5 by alternatively doing forward and back-propagation processes. We initiate the learning rate (LR) equal to 0.001 and the hyperparameter of $\varepsilon = 10^{-7}$. We also set the number of epochs equal to 15. Other setups are set, such as the first exponential decay rate (1st) equal to 0.9, the second exponential decay rate (2nd) equal to 0.999, and the batch size equal to 32. Next, the test set was then used to evaluate our proposed method's performance. Therefore, the achieved results concern test sets only. After that, comparisons with other existing methods were prepared with the use of resources from Sect. 2, and finally, we would like to emphasize that there is no training validation test split process again or any other augmentation techniques used.

5 Result and Discussion

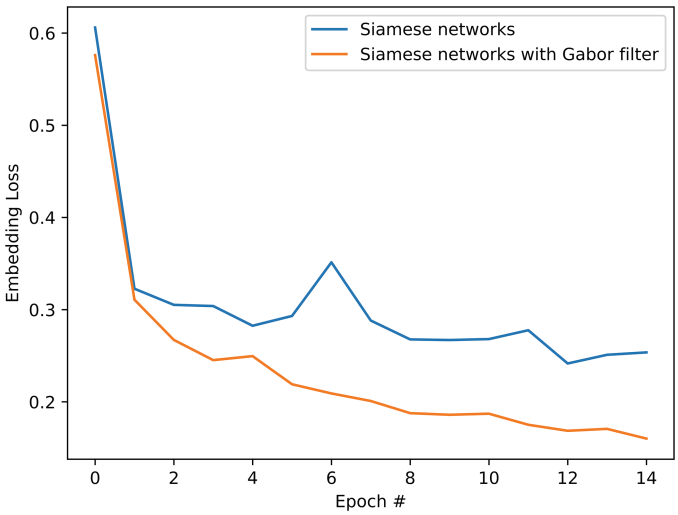
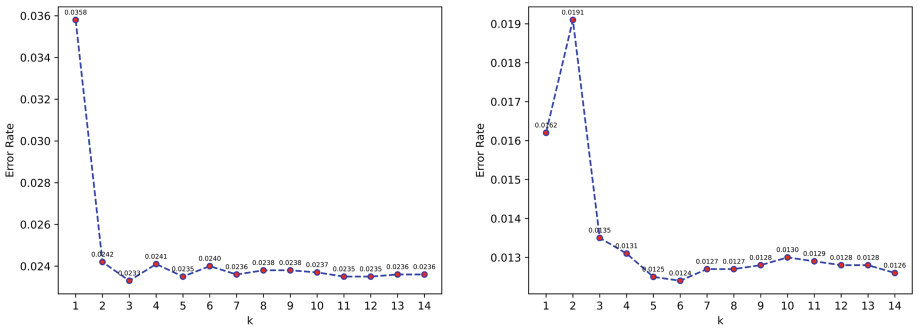


Fig. 5. Embedding loss.

A minimum loss of training result means that the classification was well performed. The loss of the embedding process on the training data is shown in the following Fig. 5. The training data contains 60000 images used to build the model. The network optimized the error at every iteration until 15 epochs. This comparison between those two networks shows that network 2, the Siamese network with Gabor filters, generated a better learning curve and was able to obtain

better performance with a lower loss of 0.1601 rather than network 1, the original Siamese network, with a loss of 0.2536.

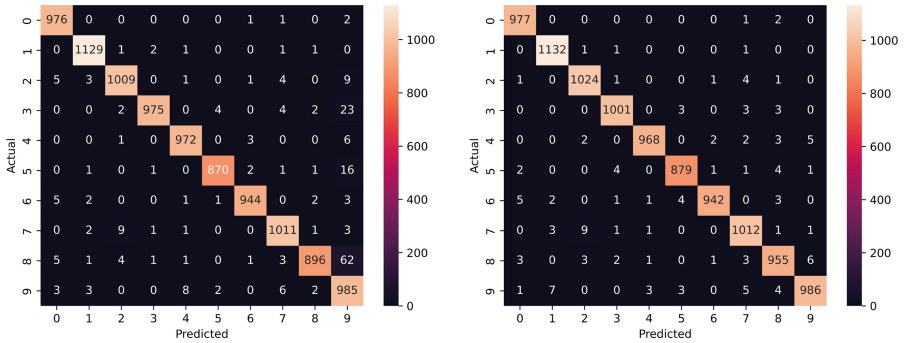
According to Fig. 5, the following are our conclusions: Firstly, we noticed that the Gabor filter is effective in the Siamese networks with the given parameters. As we can see from the figure during the training process, the Siamese network combined with the Gabor filter works well for the MNIST dataset. Secondly, the Gabor filters produced by the real part of the Gabor function are suitable for this network. Finally, we contended that this property is advantageous for the Siamese network. The explanation behind this is that typically, a Gabor layer relates to textural features that are effective in the lower network layers. Therefore, this suggests that we can utilize Gabor filters for replacing a classical one in the lower layers of a network to really get excellent results.



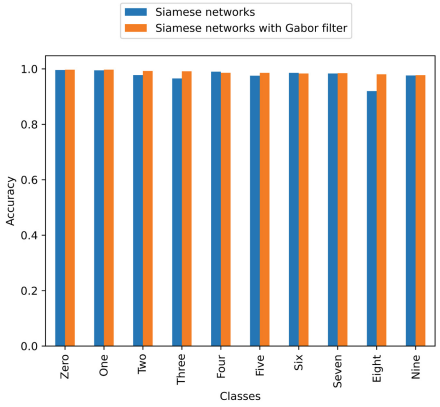
(a) Without the Gabor filters (b) With the Gabor filters

Fig. 6. The optimal value of k using elbow method.

After the embedding process steps were over, we employed the kNN classifier in order to classify handwritten digit images. The elbow method was executed with successive values of k from 1 to 15. Figure 6 depicts a visualization of the classification error for various k values. Evidently, the ideal number of k equals 3 for classifying MNIST numerical digits using Siamese networks without Gabor filters (network 1). Next, the ideal number of k for classifying MNIST numerical digits using Siamese networks with Gabor filters (network 1) equals 6. We get the error rates on the testing dataset on network 1 and network 2 are 0.0233 and 0.124, respectively. We also provided the differences between network 1 and network 2 in terms of the confusion matrix and the accuracy of each class shown in Fig. 7. In accuracy, our network 2 outperformed network 1 in almost all classes. Especially in class "eight", the accuracy obtained by network 2 clearly higher than network 1. Regarding the difference in accuracy, network 2 is 1.09% more accurate than network 1.



(a) Without the Gabor filters. (b) With the Gabor filters.



(c) Class-wise accuracy values.

Fig. 7. Confusion matrix and the accuracy of each class.

The accuracies of our proposed network and other existing methods on the same dataset are shown in Table 2. Noteworthy is the fact that our suggested network outperforms existing approaches on MNIST datasets. A possible reason for the better results is based on the feature extraction method and the classifier used for classification. Our approach used Siamese networks with Gabor filters to classify MNIST digits and achieved an overall classification accuracy of 98.76%. On the other hand, [2] introduced a hybrid method for recognizing the MNIST dataset. They used CNN for feature extraction and fed the results to a hybrid classifier for classification. Their CNN and SVM-based hybrid classifier achieved 94.40% classification accuracy. [15] carried out the handwritten digit recognition using SG-L1 for feature selection. They obtained 97.00% classification accuracy with a deep NN as the classifier. [16] also investigated this classification task, which used SNN with NormAD as its learning algorithm and achieved 98.40% classification accuracy. [17] performed handwritten digit recognition using deep SNN training with STDP algorithm and obtained a classification accuracy of 98.40%. [18] used FOBP neural

networks and PEO optimization to optimize the network and achieved an overall classification accuracy of 96.54%. Another method introduced by [31] proposed a combination of LeNet-5 for feature extraction and multi-level fusion of kNN and RF, which achieved an accuracy of 98.10%. Additionally, [32] introduced a CNN approach for classifying MNIST digits, achieving a 98.00% accuracy rate. However, we would like to emphasize that this work contained several limitations. First, cross-validation experiments were not conducted to assess the model's generalization capabilities, and second, cross-dataset evaluation for robust handwritten digit classification method also was not addressed. Consequently, this approach could be incapable of recognizing the handwritten digits from unseen samples.

Table 2. Evaluate the performance between proposed and other existing methods.

References	Models	Accuracy
[2]	CNN + SVM	94.40%
[15]	SG-L1-DNN	97.00%
[16]	SNN + NormAD	98.17%
[17]	Deep SNN + STDP	98.40%
[18]	PEO-FOBP	96.54%
[31]	LeNet-5 + fusion of kNN and RF	98.10%
[32]	CNN + kNN + SVM	98.00%
Network 1	Siamese networks + kNN	97.67%
Network 2	Siamese networks + Gabor filter + kNN	98.76%

6 Conclusion and Future Work

In this paper, we have shown a good fusion of the Siamese network with the Gabor filter for handwritten digit recognition. The model combines the advantages of the Siamese network, the Gabor filter, and the kNN classifier in recognizing handwritten digits. The experimental study revealed that our suggested strategy has a classification accuracy 98.76% on the MNIST dataset. It is important to note the research on the Siamese networks with the Gabor filter is in its early stages and can be further improved. In upcoming projects, we will enhance the network's performance in the following areas. Firstly, we will improve the performance using different Gabor filter banks with learnable parameters and utilize the Gabor filters to deeper layers of the networks, as well as adapt this algorithm as a preprocessing tool. Next, we will also focus on improving some limitations of the proposed method, such as cross-validation experiments for generalization and cross-dataset evaluation for the robustness of classifying unseen samples. Finally, will try to design new configurations of Siamese networks based on the best real and imaginary forms of Gabor filters.

References

1. Babu, U.R., Venkateswarlu, Y., Chintha, A.K.: Handwritten digit recognition using k-nearest neighbour classifier. In: Proceedings - 2014 World Congress on Computing and Communication Technologies, WCCCT 2014, pp. 60–65. IEEE Computer Society (2014)
2. Niu, X.X., Suen, C.Y.: A novel hybrid CNN-SVM classifier for recognizing handwritten digits. Pattern Recogn. **45**, 1318–1325 (2012)
3. Goltsev, A., Gritsenko, V.: Investigation of efficient features for image recognition by neural networks. Neural Netw. **28**, 15–23 (2012)
4. Hossain, A., Ali, M.: Recognition of handwritten digit using convolutional neural network (CNN). In: Proceedings of 2020 International Conference on Computing and Data Science CDS 2020, vol. 19, pp. 183–190 (2020)
5. Govindarajan, M.: Evaluation of ensemble classifiers for handwriting recognition. Int. J. Mod. Educ. Comput. Sci. **5**, 11–20 (2013). https://doi.org/10.5815/ijmecs.2013.11.02
6. Malik, H., Roy, N.: Extreme learning machine-based image classification model using handwritten digit database. In: Advances in Intelligent Systems and Computing, pp. 607–618 (2018). https://doi.org/10.1007/978-981-13-1822-1_57
7. Shi, B., Bai, X., Yao, C.: An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE Trans. Pattern Anal. Mach. Intell. **39**, 2298–2304 (2017). https://doi.org/10.1109/tpami.2016.2646371
8. Shamim, S.M., Miah, M.B.A., Sarker, A., Rana, M., Jobair, A.A.: handwritten digit recognition using machine learning algorithms. Indonesian J. Sci. Technol. **3**, 29 (2018). https://doi.org/10.17509/ijost.v3i1.10795
9. Dine, K.Z., Nasri, M., Moussaoui, M., Benchaou, S., Aouinti, F.: Digit recognition using different features extraction methods. In: Advances in Intelligent Systems and Computing, pp. 167–175 (2016). https://doi.org/10.1007/978-3-319-46568-5_17
10. Ji, P., Jin, L., Li, X.: Vision-based vehicle type classification using partial Gabor filter bank. In: Proceedings of the IEEE International Conference on Automation and Logistics, ICAL 2007, pp. 1037–1040 (2007)
11. Sarwar, S.S., Panda, P., Roy, K.: Gabor filter assisted energy efficient fast learning convolutional neural networks. In: Proceedings of International Symposium on Low Power Electronics and Designs, pp. 1–6 (2017)
12. Alekseev, A., Bobe, A.: GaborNet: Gabor filters with learnable parameters in deep convolutional neural networks. arXiv. (2019)
13. Koch, G., Zemel, R., workshop, R.S.-I. deep learning, 2015, undefined: Siamese neural networks for one-shot image recognition. cs.toronto.edu. http://www.cs.toronto.edu/~gkoch/files/msc-thesis.pdf. Accessed 09 Apr 2021
14. Bromley, J., et al.: Signature verification using a "Siamese" time delay neural network. Int. J. Pattern Recognit. Artif. Intell. **07**, 669–688 (1993)
15. Scardapane, S., Comminiello, D., Hussain, A., Uncini, A.: Group sparse regularization for deep neural networks. Neurocomputing. **241**, 81–89 (2017)
16. 23 Kulkarni, S.R., Rajendran, B.: Spiking neural networks for handwritten digit recognition–Supervised learning and network optimization. Neural Netw. **103**, 118–127 (2018)
17. Kheradpisheh, S.R., Ganjtabesh, M., Thorpe, S.J., Masquelier, T.: STDP-based spiking deep convolutional neural networks for object recognition. Neural Netw. **99**, 56–67 (2018)

18. Chen, M.R., Chen, B.P., Zeng, G.Q., Lu, K.D., Chu, P.: An adaptive fractional-order BP neural network based on extremal optimization for handwritten digits recognition. Neurocomputing **391**, 260–272 (2020)
19. Yuan, Y., Zhang, J., Wang, Q.: Deep Gabor convolution network for person re-identification. Neurocomputing **378**, 387–398 (2020)
20. Chicco, D.: Siamese neural networks: an overview. Methods Mol. Biol. **2190**, 73–94 (2021)
21. Chopra, S., Hadsell, R., LeCun, Y.: Learning a similarity metric discriminatively, with application to face verification. In: Proceedings of 2005 IEEE Computer and Social Conference on Computer Vision and Pattern Recognition, CVPR 2005, I, pp. 539–546 (2005)
22. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: A unified embedding for face recognition and clustering. In: Proceedings of IEEE Computer and Social Conference on Computer Vision and Pattern Recognition 07–12-June-2015, pp. 815–823 (2015)
23. Dong, X., Shen, J.: Triplet loss in Siamese network for object tracking. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11217, pp. 472–488. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01261-8_28
24. Hoffer, E., Ailon, N.: Deep metric learning using triplet network. In: Feragen, A., Pelillo, M., Loog, M. (eds.) SIMBAD 2015. LNCS, vol. 9370, pp. 84–92. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24261-3_7
25. Zhuang, B., Lin, G., Shen, C., Reid, I.: Fast training of triplet-based deep binary embedding networks. In: Proceedings of IEEE Computer and Social Conference on Computer Vision and Pattern Recognition, 2016-December, pp. 5955–5964 (2016)
26. Song, H.O., Xiang, Y., Jegelka, S., Savarese, S.: Deep metric learning via lifted structured feature embedding. In: Proceedings of IEEE Computer and Social Conference on Computer Vision and Pattern Recognition 2016-December, pp. 4004–4012 (2015)
27. Cheng, D., Gong, Y., Zhou, S., Wang, J., Zheng, N.: Person re-identification by multi-channel parts-based CNN with improved triplet loss function. In: Proceedings of IEEE Computer and Social Conference on Computer Vision and Pattern Recognition, 2016-December, pp. 1335–1344 (2016)
28. Hermans, A., Beyer, L., Leibe, B.: In Defense of the Triplet Loss for Person Re-Identification (2017)
29. TensorFlow Addons Losses: TripletSemiHardLoss. https://www.tensorflow.org/addons/tutorials/losses_triplet. Accessed 07 July 2022
30. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010). http://yann.lecun.com/exdb/mnist/
31. Zhao, H., Liu, H.: Multiple classifiers fusion and CNN feature extraction for handwritten digits recognition. Granular Comput. **5**, 411–418 (2019). https://doi.org/10.1007/s41066-019-00158-6
32. Enriquez, E.A., Gordillo, N., Bergasa, L.M., Romera, E., Huélamo, C.G.: Convolutional neural network vs traditional methods for offline recognition of handwritten digits. In: Advances in Intelligent Systems and Computing, pp. 87–99 (2018). https://doi.org/10.1007/978-3-319-99885-5_7