Hindawi
Journal of Healthcare Engineering
Volume 2023, Article ID 9856361, 1 page
https://doi.org/10.1155/2023/9856361

Retraction

Retracted: Efficient Algorithms for E-Healthcare to Solve Multiobject Fuse Detection Problem

Journal of Healthcare Engineering

Received 29 August 2023; Accepted 29 August 2023; Published 30 August 2023

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Peer-review manipulation

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] I. Ahmad, I. Ullah, W. U. Khan et al., "Efficient Algorithms for E-Healthcare to Solve Multiobject Fuse Detection Problem," *Journal of Healthcare Engineering*, vol. 2021, Article ID 9500304, 16 pages, 2021.

Hindawi
Journal of Healthcare Engineering
Volume 2021, Article ID 9500304, 16 pages
https://doi.org/10.1155/2021/9500304

Research Article

Efficient Algorithms for E-Healthcare to Solve Multiobject Fuse Detection Problem

Ijaz Ahmad,[1] **Inam Ullah**,[2] **Wali Ullah Khan**,[3] **Ateeq Ur Rehman**,[2,4] **Mohmmed S. Adrees**,[5] **Muhammad Qaiser Saleem**,[5] **Omar Cheikhrouhou**,[6] **Habib Hamam**,[7,8] **and Muhammad Shafiq**[9]

[1]*School of Pattern Recognition and Intelligent System, Shenzhen Institute of Advance Technology (Chinese Academy of Science), Shenzhen, China*
[2]*College of Internet of Things (IoT) Engineering, Hohai University (HHU), Changzhou Campus, Changzhou 213022, China*
[3]*School of Information Science and Engineering, Shandong University, Qingdao 266071, China*
[4]*Department of Electrical Engineering, Government College University, Lahore 54000, Pakistan*
[5]*College of Computer Science and Information Technology, Al Baha University, Al Baha, Saudi Arabia*
[6]*College of CIT, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia*
[7]*Faculty of Engineering, Université de Moncton, Moncton NB E1A3E9, Canada*
[8]*International Institute of Technologie (IIT), Sfax, Tunisia*
[9]*Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, Republic of Korea*

Correspondence should be addressed to Muhammad Shafiq; shafiq@ynu.ac.kr

Received 7 April 2021; Revised 23 April 2021; Accepted 13 May 2021; Published 27 May 2021

Academic Editor: Han Wang

Object detection plays a vital role in the fields of computer vision, machine learning, and artificial intelligence applications (such as FUSE-AI (E-healthcare MRI scan), face detection, people counting, and vehicle detection) to identify good and defective food products. In the field of artificial intelligence, target detection has been at its peak, but when it comes to detecting multiple targets in a single image or video file, there are indeed challenges. This article focuses on the improved K-nearest neighbor (MK-NN) algorithm for electronic medical care to realize intelligent medical services and applications. We introduced modifications to improve the efficiency of MK-NN, and a comparative analysis was performed to determine the best fuse target detection algorithm based on robustness, accuracy, and computational time. The comparative analysis is performed using four algorithms, namely, MK-NN, traditional K-NN, convolutional neural network, and backpropagation. Experimental results show that the improved K-NN algorithm is the best model in terms of robustness, accuracy, and computational time.

1. Introduction

E-healthcare is a broad term consisting of improvements in medical services detected by digital technology. In E-healthcare, there are various diseases including diabetes, cancer, and stroke, as well as machine learning systems for diagnosing these diseases and integrating AI and so forth. Recently, deep learning classifiers have had excellent target detection performance in various electronic healthcare applications such as diagnosis of heart disease based on heart image, detection of cancer, and various EEG data sets classification including chest X-ray, diabetic retinopathy, and skin cancer. FUSE-AI, a startup company based in Hamburg, has developed a system that can detect and classify tumors in MRI scans. Hamburg has developed the FUSE-AI system, which can classify tumors based on MRI scans using machine learning classifiers.

Object detection is an artificial intelligence technology related to image processing and computer vision, which can detect various objects (vehicles, buildings, and people) in specific categories in digital videos and images. In-depth study of object detection areas includes pedestrian detection,

face detection, and traffic signal detection [1–7]. Use cases ranging from personal safety to work efficiency subdivide object detection into a wide range of areas [8–13].

Although huge innovations are taking place in the field of healthcare, many problems must still be solved, especially heterogeneous data fusion, mobile data transmission, and analysis. Facial recognition is a form of face detection that can be used as a high security measure that allows only certain people to enter, such as highly sensitive areas in government buildings. Many applications of multitarget detection and classification exist, such as face detection [14], people counting [15], vehicle detection [16], and identification of good and bad food [17].

The combination of image processing and artificial neural network is a huge combination that can be used endlessly for various purposes. In the past few years, a lot of work has been done in these two fields. Therefore, this technology has become the focus of any artificial intelligence company, and it is also very necessary for the governments of many countries/regions in the world [18]. In addition, researchers are also focusing on the modernization of other scientific fields such as artificial intelligence and information technology [19–31].

Object detection methods are usually divided into two methods, machine learning and deep learning. Machine learning methods include support vector machine (SVM) classification strategies and deep learning classification methods used by various neural networks such as convolutional neural networks (CNN), neural backpropagation networks (or backscatter communication networks), and K-nearest neighbor network (KNN) [32]. The object detection model can be divided into two parts, extraction function and classification [33–35].

In the visualization of objects, feature extraction needs to extract various visual features to provide a reliable representation [18, 33, 34, 36, 37]. In fact, these features represent similarities with the human brain and complex cells [12]. It is difficult to find a powerful feature extractor to extract all the features of an object and construct it manually. However, in classification, the classifier distinguishes the target object from other representative features or categories for visual recognition. Generally, support vector machine (SVM) [38], deformable part-based model (DPM) [39], and AdaBoost [40] are good classifiers.

This article compares three different neural networks to conduct CNN, BP, and K-NN experiments to find out which is the best in fusion detection and can also evaluate training and output time and preprocessing time. Most of the researches in the field of target detection and surveillance in wireless sensor networks (WSN) focus on single or multiple target detection. However, there are few studies aimed at monitoring and detecting fuse objects. The purpose of this research is to design a robust algorithm to effectively classify fuse objects in a single image. First, preprocess the data and prepare it for each algorithm, because each algorithm uses the data differently, especially the backpropagation network. Then train the neural network to correctly classify each fuse. We used deep learning in our research work.

The following are the key contributions of this work:

(i) We apply the new method by adding robust neighbor to the training samples to modify the K-NN model.

(ii) A comparison study is accomplished by using modified K-NN with classical K-NN, B-PN, and CNN models based on accuracy and time complexity in E-healthcare object (fuse) detection. Our study proved that, compared with other models, the modified K-NN model has higher accuracy and less time complexity.

(iii) We propose using the modified K-NN model for future research in E-healthcare object detection applications.

The rest of this article is structured as follows. Section 2 introduces related work. Section 3 introduces the methodology. Section 4 introduces the modified K-nearest neighbor (M-KNN) algorithm. Section 5 discusses the experimental results, and Section 6 discusses the comparison results. Finally, Section 7 summarizes the paper and provides prospects for future research work.

2. Related Work

2.1. CNN. CNN is an artificial neural network with similar architectures to ordinary neural networks [41]. CNNs have three sections of architectures, input neurons, learnable weights, and biases. CNN has been widely applied in the area of handwritten character recognition. CNN is an artificial neural network with an architecture similar to ordinary neural networks [41]. CNN has a three-part structure, namely, input neurons, learnable weights, and deviations, and is widely used in the field of handwritten character recognition [42–46] and face recognition [47]. The explicit assumption that the image is input is made by the CNN architecture, which allows us to encode certain attributes into the architecture. Make the forwarding function more effective and greatly reduce the number of network parameters. The general structure has special benefits for using neural networks to solve different problems. CNN has a clear biological structure; the early work of Hubel and Wiesel is based on the ANN model of cat's visual cortex application [42, 45, 48].

CNN has been successfully applied to many classification applications such as traffic signal detection to detect various traffic signals. CNN uses supervised learning algorithms, so it can predict better classification results. The CNN architecture is shown in Figure 1.

2.2. BP. Backpropagation neural network (BPN) is a neural network used as a neural forwarding network [49–51]. The BP algorithm has a multilayer architecture mapping, which can transmit information between the forwarding and output layers. Therein, it transmits the signal through the hidden layer and adjusts the weight based on the delta rule between the actual output of the ANN and the predicted output, thereby minimizing the error rate. Any nonlinear function with arbitrary precision can usually be

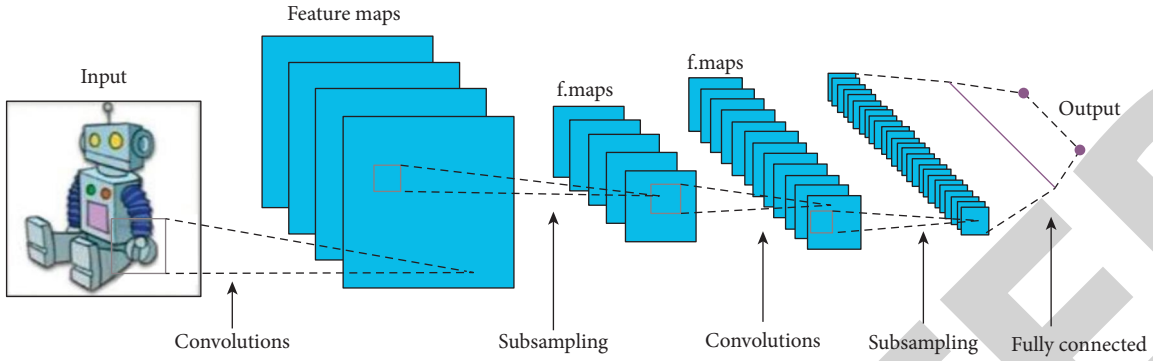

Figure 1: CNN architecture.

approximated by a hidden BPN layer [52]. For predicting complex nonlinear systems, this feature makes BPN famous. BP algorithm is a classification model with two gradient descent methods and mean square error. When the minimum error is reached, the square number and connection weight will be adjusted. In the process of BP algorithm, the input signal is provided to the network first, and then the sample training process is carried out. Then calculate the gradient of the error signal value [53].

The architecture of BPN consists of two parts called forward propagation (or output layer) and backpropagation (or error signal). In forward propagation, the input layer is propagated to the input signal and forwarded to the hidden layer through the output layer. The weight and offset values remain constant throughout the process. The next layer is affected by the operation signal. If the predicted output layer and the output layer do not match, the operation signal will shift to the backpropagation of the error signal. On the other hand, backpropagation is defined as the difference between the expected output value and the actual output layer value. The weight and offset value will be constantly changed and added to reduce the difference between the actual output layer value and the predicted output layer value. In this step, the error signal is propagated and used as the input signal. The BP architecture is shown in Figures 2 and 3, respectively.

2.3. K-NN. In object detection, the K-NN algorithm has been successfully used for classification and regression, which is an unassisted method. In the K-NN algorithm, the input depends on the value of $K = 0$ and $K = n$ in the training process room. Due to its high accuracy and simplicity and being easy to understand and easy to implement, the K-NN algorithm has been successfully used in many data analysis applications, including pattern recognition databases, information retrieval, and machine learning [54, 55]. Therefore, this is why the top ten algorithms for data mining in recent decades have been K-NN algorithms [56]. This algorithm is used for data processing, data classification, and clustering. D. Vijayalaksmi applies the K-NN algorithm to the classification and clustering of diabetes in the medical application of diabetes. Based on the accuracy and purity of the sample, the solution was compared between the K-NN and K-means methods. The implementation results of these models show that the K-NN solution is more effective and reliable than K-means. Satheesh and Patel's dynamic K-NN is proposed as a data classifier, combined with the principle of object-oriented programming [57]. This question is classified in a single consolidated form during the training phase. Compared with traditional K-NN, it implements and classifies solutions more effectively. The K-NN architecture is shown in Figure 4.

All object detection methods must consider the inherent uncertainty of the size, position, and structure of the target in the natural scene image. Viola Jones detector [58], histogram of oriented gradients (HOD) detector [34], and deformable part-based model (DPM) [59] are just some traditional methods of object detection in the natural image scenes.

These methods mainly rely on manually extracted object features to determine the parameters of the algorithm. However, due to the rapid development of deep learning in recent years, many object detection methods based on this advanced technology have been developed [60]. By properly training their network architecture, these methods have proven that they can accurately locate object regions in natural image scenes. The object detection method based on R-CNN, the object detection method based on SSD, and the object detection method based on YOLO are three types of object detection methods.

Researchers have proposed a variety of target detection algorithms. For example, an algorithm based on multiscale deformable CNN is proposed in [61]. The authors compare and study mainstream object detection algorithms to solve the problems faced by current methods. This study confirmed results that are comparable to or better than the latest methods. Deep convolutional networks usually are used to obtain multiscale features and solve geometric transformations by integrating deformable convolutional structures. These networks fuse multiscale features through sampling to introduce the final object recognition and region regression.

In [62], the authors introduced another object detection through flowing and fusion. The authors propose an end-to-end deep neural network (DNN) and flow fuse tracker (FFT) tracking technology, which solves two methods of tracking problems, such as target fusion and target flow. To be more precise, the FlowTracker DNN module obtains an unlimited number of target directional motions from the pixel-level optical flow in the target stream. On the other hand, the FuseTracker DNN module refines and tracks the target

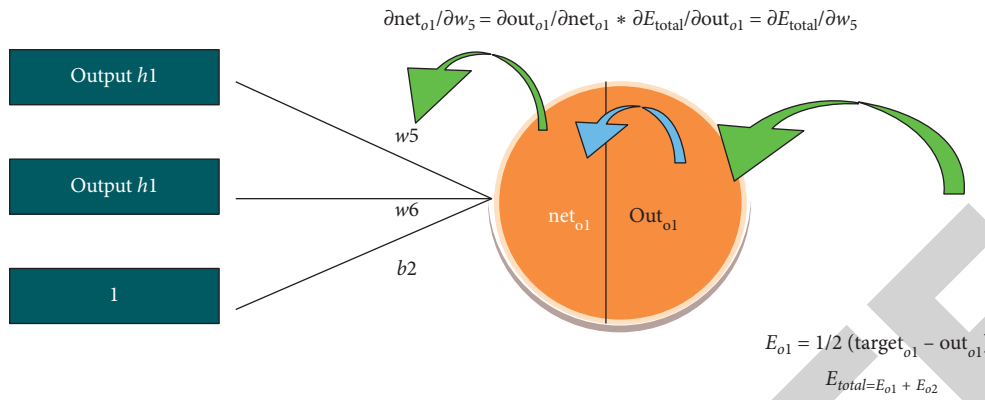

Figure 2: BP architecture (output layer).

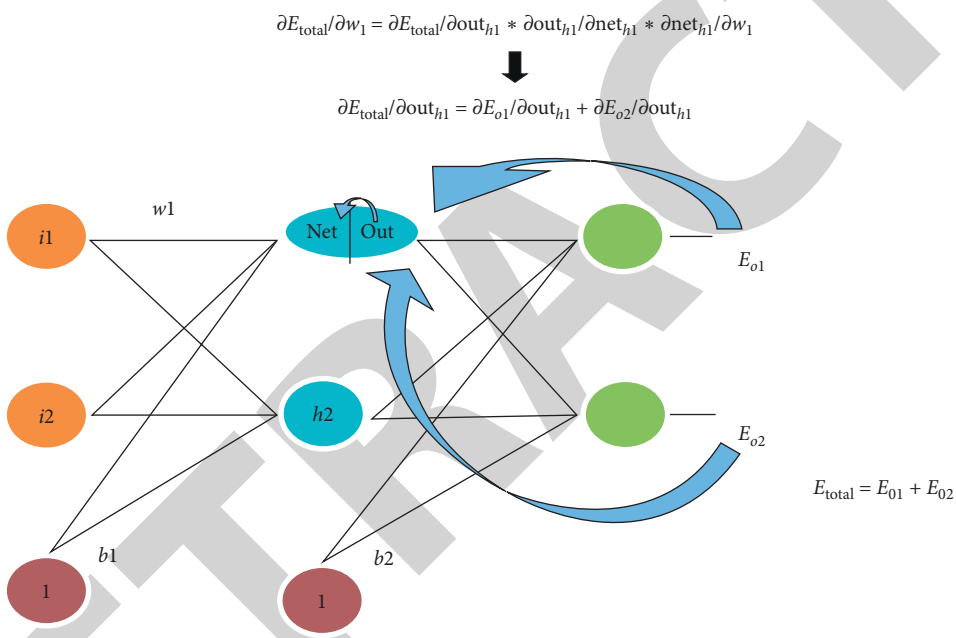

Figure 3: BP architecture (hidden layer).

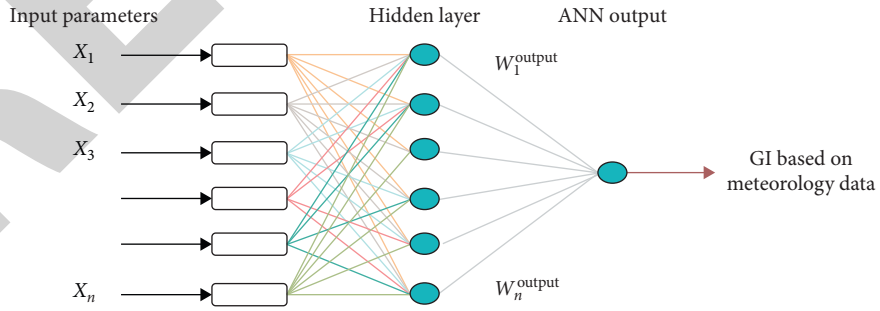

Figure 4: Typical K-NN architecture.

proposed by FlowTracker and integrates frame-by-frame object detection in target fusion, instead of trusting one of the two incorrect sources of target recommendations. Because FlowTracker can detect complex target motion patterns, FuseTracker can improve and integrate the targets in FlowTracker and the detector.

3. ANN Models Evaluation Methodology

In this section, three different neural networks are evaluated for CNN, BP, and K-NN experiments to find which is better in fusion detection and to determine training time, execution time, or preprocessing time. The processing process of

some networks is not simple, requiring a lot of preprocessing to adjust our data and then put it on the network. The whole process is shown in Figure 5. The data is preprocessed first, and then the demand for the network is evaluated, because each network can be processed differently in different ways. For example, the input signal required by CNN may be simple and straightforward, but processing BP may not be the case.

With the latest developments in computer vision models that focus on deep learning, object detection applications are easier to create than ever. In addition to significant performance improvements, these methods also utilize massive image data sets to reduce the need for large data sets. The current method focuses on a complete end-to-end pipeline and allows real-time use, so performance has been greatly improved. If you prepare the data in vector form, provide the same data set for each network and start the training process. After the training is completed, give the network a record from the main vector data set, and let the network predict the performance. First check the single fuse detection, and if the result is good, provide the entire test data set to the network to predict the fuse and the empty fuse slot, and allow the network difference between the two items. Finally, evaluate the accuracy and timing of each network.

The following describes in detail the experimental process from preprocessing to output detection for fuse detection.

3.1. Preprocessing. In the preprocessing process of fuse target detection, it is important to understand the basis of the specific neural network which our experiment can effectively perform. In this study, the overall results of these three models are evaluated for the first time based on 126 images of a small data set. It contains a fuse artifact and an empty fuse slot. There are several methods of detecting objects. The experiment tested three models of CNN, BP, and K-NN [9]. There are more models that can also be tested, but these three models were selected for the research work because these models have been widely used in many major applications around the world. In order to deal with these models first, the processing steps must be divided into the following questions.

Training the model with more and more images and determining the best accuracy is a difficult task. As the number of images increases from 500 to 1,000, the accuracy and time complexity of the model also increase. The model takes some time to load the data set. As a result, training time will increase. To balance the two, we use 500 images to train the model.

(1) Good data set of images: in order to train any neural network model, good experimental results require a large number of images to provide the model. The "good number" has no specific number, only the model can be trained and tested, and the number of images continues to increase until the model produces more reliable results and fewer errors. But increasing the number of images is not free. If the computer is not powerful enough, it will cost us more training time. Therefore, the collection of image data sets is a difficult task.

(2) Basic knowledge of each model: it is necessary to understand the basic knowledge of each model, because this will allow us to choose which model is the most effective to complete our work, thus spending relatively less training time and better performance. The size of the image needs to be adjusted, and usually the entire size of a single image or the entire data set needs to be changed, so it is very important to have basic knowledge of image processing; and, as mentioned above, each network takes data as input in a different format or size, in order to understand the internal data of the image and how to use it.

Before continuing to compare results, you must have basic knowledge of how neural networks work, especially how convolutional neural networks and deep neural networks work and how to train these levels. What are activation and loss functions? How do they work and how to get different results for different functions? Another important parameter in a neural network is the learning rate, which directly affects the model learning process in any neural network. During training, the average learning rate and the average failure rate should be opposite to each other. For example, if the loss is less than 0.05 and the learning rate is close to 1, it is assumed that the model is well trained and the training process can be stopped.

3.2. Data Set (Input Data). The fuse object data comes from the Natural Science Foundation of Jilin Province. This data set includes 3000 images of fuses. In order to train a custom model, the experiment needs to prepare the data set of the required target and in some cases also needs to prepare other classes and categories, so that the network can easily identify the required target and nontarget objects. Each network requires different forms and sizes of data to be fed into the network first. It is necessary to understand how this particular network works, and then do some data preprocessing and prepare for the network.

Usually, the problem of object detection related to the X problem is raised. There is no need to implement the model. The important view is how many images are needed in the training phase. Each type of representative image must have a large number (e.g., >100 and possibly >1000). The special context is required for the image perspective in pattern recognition. In traffic signal detection, it will be required to meet the image requirements under each condition, such as camera conditions and different weather conditions. The training process may be affected by these contexts, causing the model to fail to train correctly, if we have a small amount of data.

The following is a fuse board containing at least 39 fuse objects, as shown in Figure 6; the fuse board extracted fuse objects from different images and prepared our small data set, which mixed an empty fuse slot.

First, extract each fuse object, convert it to grayscale, and then adjust it to 200×200 in a small image to save training time.

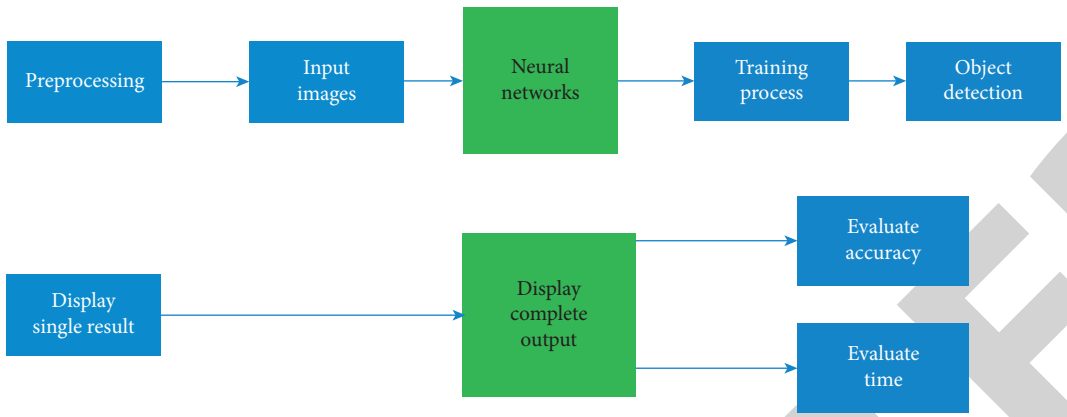

Figure 5: The complete overview of the system.

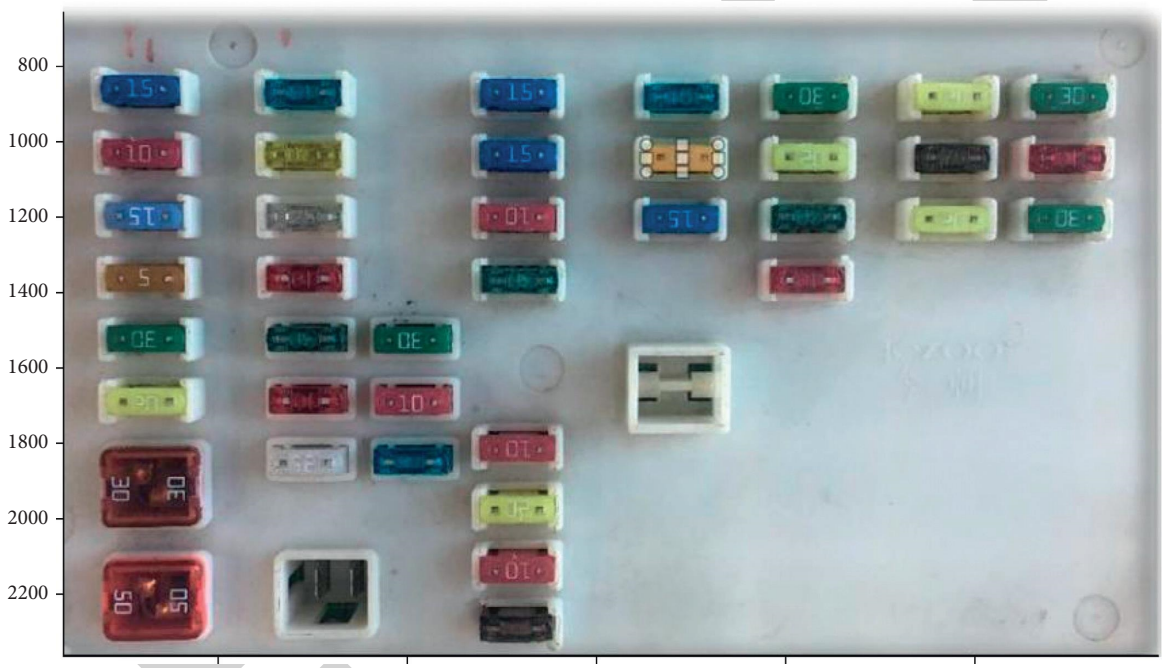

Figure 6: The fuse board.

3.3. Training Process. Training a neural network is usually simple, but sometimes it can be tricky, which can take a lot of time to find and solve difficult problems. For example, in machine learning and neural networks, it is really difficult to figure out why the learning rate is not so high and why the loss is not so low. At this stage, check the entire circle from the beginning to find the parameters that may cause the problem. These are three different python fuse detection algorithms. The training phase should include the following:

(i) Preparing images

(ii) Extracting images

(iii) Resize and grayscale

(iv) Feeding network with images

(v) Start training and monitor the learning rate

3.4. Actual Training. If the image is ready, the actual training and monitoring will be carried out in the next stage of target detection. In this process, the algorithm needs enough data to train the model. The best training phase requires more epochs and secret layers. The learning rate must be selected to improve the accuracy of the model. If adjustment is required, a very low value must be activated.

4. Modified K-Nearest Neighbour (M-KNN) Algorithm

This section contains the concept of weight features and makes an excellent improvement on distance measurement by adding suggested process weight features. The workflow of the proposed method is shown below. The idea of proposing the proposed method is to assign appropriate labels of K data training points to the instance. The first point of

starting MKNN adds a strong neighbor to the training and calculation of the sample. Then define the value after assigning the weights. Figure 7 shows the pseudocode of the K-NN modified algorithm. Validation error rate and training error rate are two parameters, and their values are between $k = 1$ and $k = 7$. If we increase the value of $k = 1$ to $k = 7$, the error rate and verification error decrease, so we use $k = 7$ to get the best results.

5. Experimental Results

This section introduces the experimental methods used to detect fuses, how to develop the data set, and how to transform the data before feeding it to each network and then discusses the training process and finally see the results. The training process is gradually repeated in CNN, BP, and KNN to detect objects. In addition, it also explains how the training process is carried out, how much time is spent on each network, and the percentage of the fuse recognition rate, which means whether the network is a fuse or an empty fuse slot determined by the percentage.

5.1. Classification of K-Nearest Neighbour Network. MK-NN is a very-easy-to-implement and simple architecture. It determines the closest K neighbor based on the minimum distance between the query text and the training sample. After collecting K-nearest neighbors, most of these K-nearest neighbors will be used to predict problem events. The classifier MK-NN performs the two following steps. First, the entire computational data set is run between each training observation. Then, in the training data, name the K point closest to the range. Note that K is usually an odd number to prevent this from happening. Second, estimate the conditional likelihood of each category, that is, the score of the AA midpoint with the category name (note that $I(x)$ is an indicator function; when the statement xx is true, its value is 1; otherwise it is 0). Finally, our input xx is assigned to the most probable class. Another way to look at K-NN is to treat it as a measure of the decision boundary (i.e., the boundary of two or more categories) and then divide the data into training, testing, and verification, as shown in Table 1.

(1) Training process (K-NN): before starting training, the process needs to have various K values to see the value of K at which the network performs well. For each value of K, train the model and return the accuracy results, as shown in Table 2. If $K = 1$, the accuracy is as high as 98%, while, for $K = 7$ and $K = 9$, it remains at 90% and so on, as shown in Table 3.

It is difficult to find the value of k in MK-NN. A smaller value means that noise will have a greater impact on efficiency, while a higher value will make it computationally expensive. If the number of categories is 2, the data scientist will usually choose an odd number, and another easy way to choose k is to set $k = \text{sqrt}(n)$. The nearest neighbor algorithm uses a very basic classification method. When comparing with the new example, look at the training data to find the k training example that is the closest to the new example. Thereafter, the most common class tags (in these K training examples) are assigned to test cases. Therefore, Ki is just the number of "voting" neighbors of the test example class. If $k = 1$, the test example has the label as the closest example in the training set. If $k = 3$, it will check whether there are three closest categories in the label and will assign the most common label (i.e., at least twice) and so on to get larger Ks.

(2) Preparing data (MK-NN): preparing data for MK-NN is the same as the two networks listed above, but the only difference is that MK-NN obtains data in a two-dimensional (2D) array format. For CNN, the data needs a tensor format, which can have a larger size, including color scheme parameters. But MK-NN will get the data in (sample number, dimension) format. This is why once the data size changes, the process needs to take additional steps to convert the entire training and test data set to this format.

(3) Predictions (MK-NN): if the training process for MK-NN has been completed, the process needs to verify the network data set and determine how many accurate predictions the network makes. The sample results of Y prediction are shown in Table 4. The following prediction shows a perfect fit with the test results. The first value array shows the test results, and the second value array shows the values expected by the model.

(4) Complete predictions on test data set (MK-NN): we evaluate the entire data set on the web and view the overall results. If the model is wrong, or if it is uncertain at a certain stage whether the model is the desired target, the vertical line will reflect the trustworthiness of the model. The following is the result of the complete test data set shown in Figure 8. The predictions of the test data set are perfect, and none of the predictions are lower than 98%. Overall, the model works well, and there is no single expected result with a confidence level of less than 98%.

The accuracy is $\text{tp} = (\text{tp} + \text{FP})$ ratio, where the number of true positives is tp and the number of false positives is fp. The ability of the classifier to intuitively not mark negative samples as positive is what we called accuracy. Recall the ratio of $\text{tp} = (\text{tp} + \text{fn})$, where the true positive number is tp and the wrong negative number is fn. Intuitively, recall is the ability of the classifier to identify all positive samples. The $F1$ score can be defined as the weighted average of precision and recall, where the highest score of $F1$ is 1, and the worst is 0. The relative accuracy and recall rate of the $F1$ grades are comparable. The formula for $F1$ score is as follows: $F1 = 2$ (precision recall) = (precision + recall).

(5) Precision and recall (K-NN): the precision and recall results based on the above perfect results are listed in Table 5.

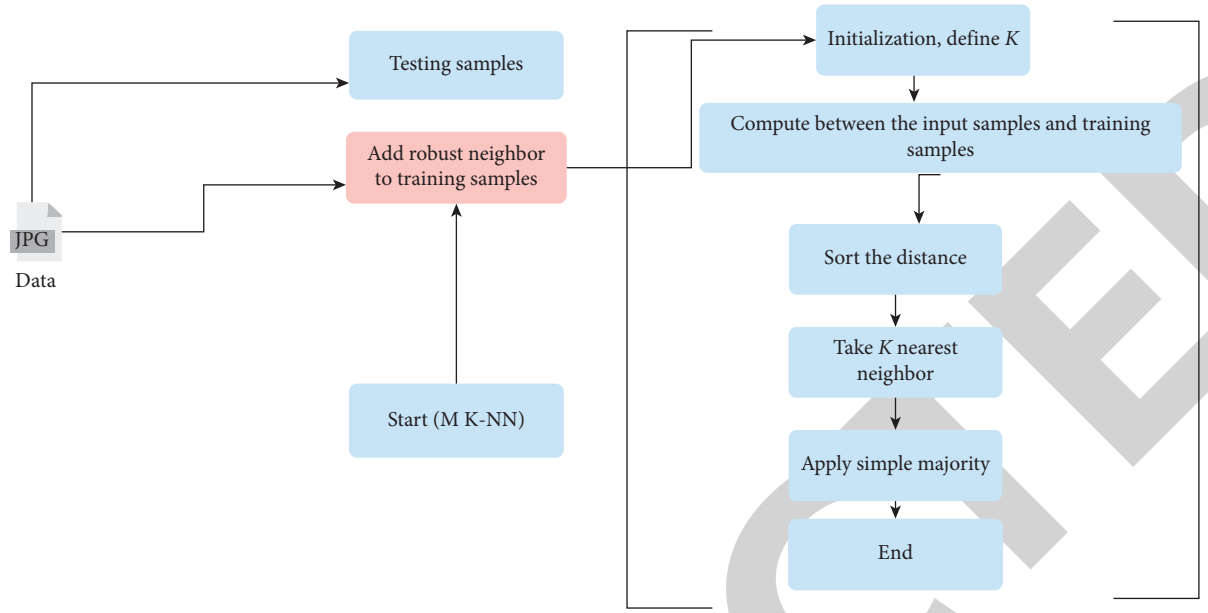

Figure 7: Proposed work model (modified K-NN).

Table 1: Training test and validation split.

Training data points	Training data labels	Validation data points	Test data labels
90	90	10	10

Table 2: Predicted labels.

True label	Predicted label	Predicted items	Detected class
[0 1]	[0.0087723 0.991227]	0.0 1.0	Not target
[0 1]	[$9.99775e-01$ $2.244791e-04$]	1.0 0.0	Target
[0 1]	[$9.99931e-01$ $6.88999e-05$]	1.0 0.0	Target
[0 1]	[$9.998918e-01$ $1.081199e-04$]	1.0 0.0	Target
[0 1]	[$9.991754e-01$ $8.245995e-04$]	1.0 0.0	Target

Table 3: Training process (K-NN).

K-value	Accuracy
$K=1$	Accuracy = 98%
$K=1$	Achieved highest accuracy of 98% on validation data
$K=3$	Accuracy = 98%
$K=1$	Achieved highest accuracy of 98% on validation data
$K=5$	Accuracy = 98%
$K=1$	Achieved highest accuracy of 98% on validation data
$K=7$	Accuracy = 98%
$K=1$	Achieved highest accuracy of 98% on validation data

Table 4: Predictions accuracy on test data (K-NN).

Test data set items	Correctly predicted	Incorrectly predicted	Accuracy (%)
26	25	1	98

5.2. Classification of Convolutional Neural Network. CNN is the simplest network to implement all image processing models. Therefore, CNN is always the programmer's first choice in terms of basic object recognition. In the process of preprocessing, training, and displaying the results, when a simple item is found, there are fewer problems with the K value. Each process will be discussed in more detail below. For the size of the output tensor (image) of the regular layer, O is the size or width of the output image. I represents entering the size or width of the image. K is the kernel size or width used in the traditional layer. N is kernel number. S is stride of the convolution operation. P is padding. The size of the output image (O) is determined by

$$O = \frac{1 - K + 2px^2}{S} + 1. \quad (1)$$

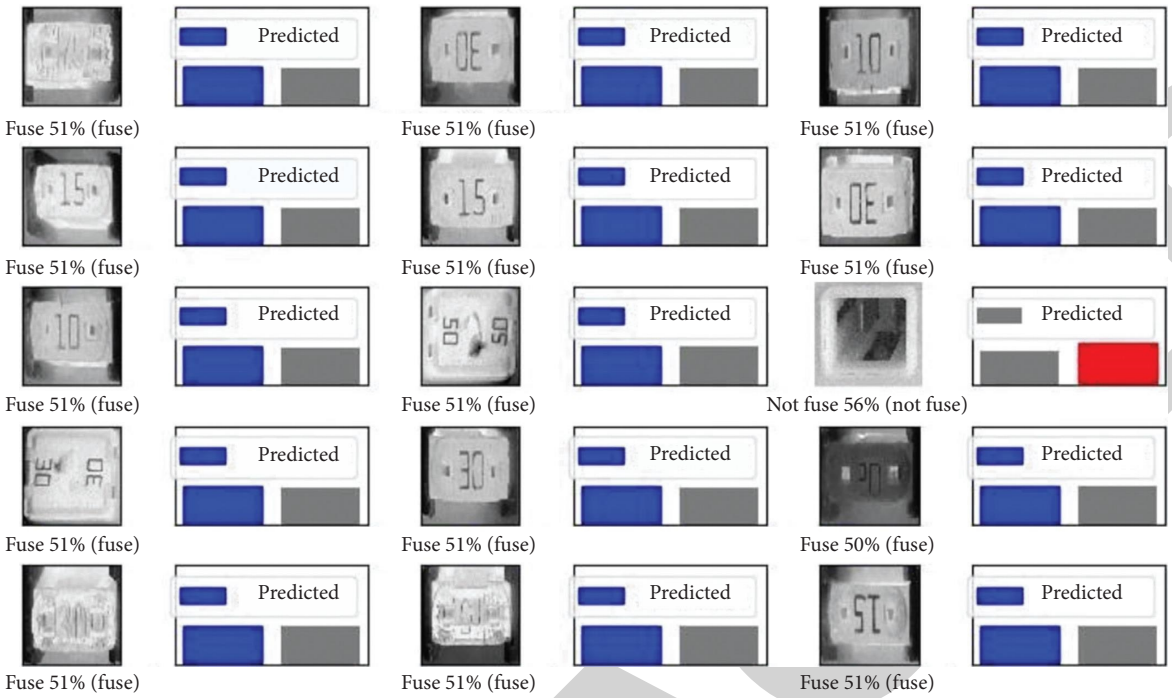

Figure 8: Complete predictions on test data set (BP).

Table 5: Precision and recall.

Label	Precision	Recall	*F*1-source	Support
0	1.00	1.00	1.00	3
1	1.00	1.00	1.00	23
Av/total	1.00	1.00	1.00	26

The number of channels in the output image is equal to the number of cores.

The input image is $227 \times 227 \times 3$, and the 96 kernels in the first convolutional layer are $11 \times 11 \times 3$. The path is 4 and the padding is 0. Therefore, the ratio of the output image is just after the first convolutional layer.

$$O = \frac{277 - 11 + 2x^0}{44} + 1 = 55. \quad (2)$$

So, the output image size is $55 \times 55 \times 96$ (one channel for each kernel).

(1) Preparing data set (CNN): in this step, prepare a common data set for all models, because you will not be able to compare our results because you want to change the size of the data set or image. First, delete each fuse object from the key image, and then save it to the data set. The next step is to resize it to 200×200 in the small image, and then convert the RGB to grayscale to save training time, as shown in Figure 9. First load the data set and build the marker. But, before creating these systems, we need to create labels first.

A sample of the image vector is shown in Figure 10. Next, make a paste image; for example, from the first image to 104 images, the target object must be marked as 0 or 1 or as needed. Mark the remaining images from 104 to 126 as 1 or 0 as untargeted images. Select the first group of images as 0; the remaining images must be labeled as 1. There is a simple data division and a clear boundary between the target object and no target object. The division of the training test is 20% of the data used for testing and 80% of the data used for preparation. Figure 11 shows the pseudocode of the modified K-NN.

Once the data set preparation process is completed using data tags, the data and labels will be changed to change where the fuse objects are located and the order in which no data set objects are merged. It has greater significance to the network and contributes to stronger and more effective preparations. This method will repeat these steps to ensure that each of our models remain straight so that the results can be easily matched. Finally, the data set is divided into training data set and test data set. The training data set will be provided to the network for training purposes, and the test data set will be provided to the network to verify the quality of the data on the network.

5.3. Training Process (CNN). The provided data can be used during the training process to train and execute the model using the fit () function. Before the training process, we should understand the number of layers required to enable this feature, as shown in Figure 12. Then compile and call the function.

On the other hand, the accuracy value should be equal to 1, and the error value should be changed to 0. During the training process, failure and accuracy must be reversed. The same is true for the lack of legitimacy and accuracy of

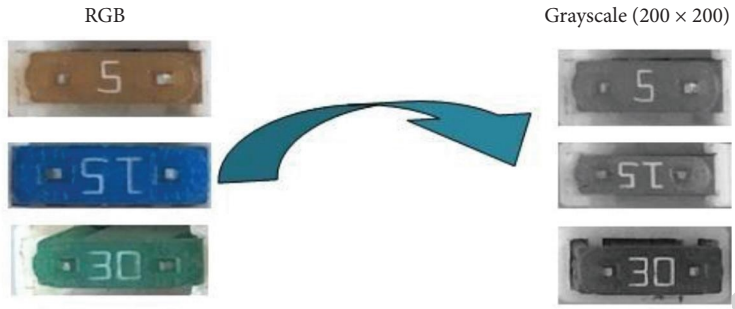

Figure 9: Preparing data for CNN.

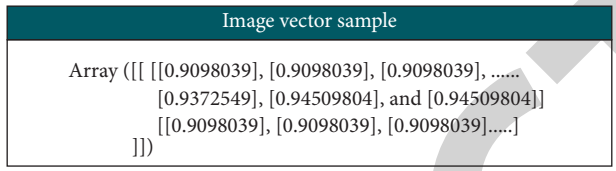

Figure 10: Image vectors and labels.

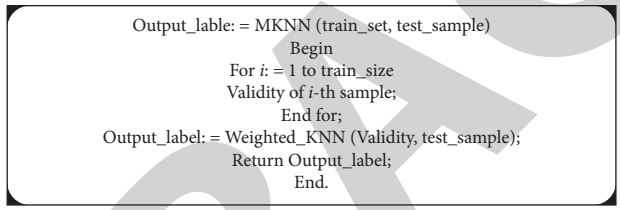

Figure 11: Pseudocode (modified K-NN).

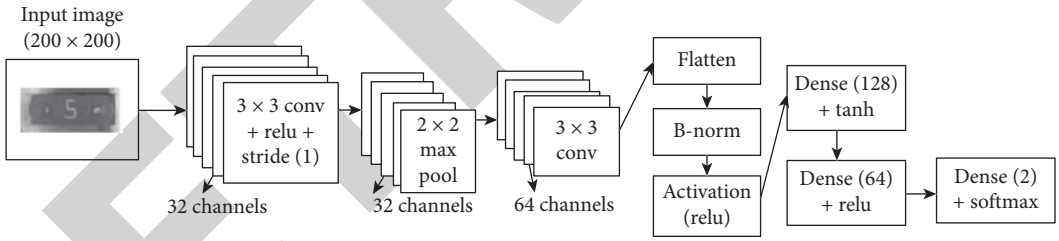

Figure 12: Preparing for training and defining the network.

verification. As long as the loss is reduced during the training phase, the checkpoint/weight will be stored as CNN.hdf5. Next time use these weights to prevent the lengthy training phase and start using the new weights saved locally to make predictions on our artifacts.

5.4. Predictions (CNN). If the CNN training is complete, provide the test data set to the network and determine how many accurate predictions the network has made. An example prediction screen is shown in Table 2. The following table shows the real label [0 1], for example, the predicted label [0.008 0.991] and the predicted object [0 1], and finally shows the detected category [not target]. Repeat this process for each item in the test data set and evaluate the results as shown below. As shown in Table 2, the time used for each period/prediction should be 1.23 seconds.

5.5. Complete Predictions on the Test Data Set (CNN). If a prediction is made through a single image test, then it is time to test the entire data set on the network and see the overall results. If the model is wrong, or it is uncertain whether the model is the desired goal at a certain stage, the vertical line will reflect the trustworthiness of the model. For example, if the model determines that 96% is the required target, the first bar will be almost complete, and the second bar will be almost invisible in the forecast chart. However, if the model predicts that the object accounts for 51% of the appropriate object, the height of the two bars will be almost the same. The following is the result of the complete test data set shown in Figure 13. The

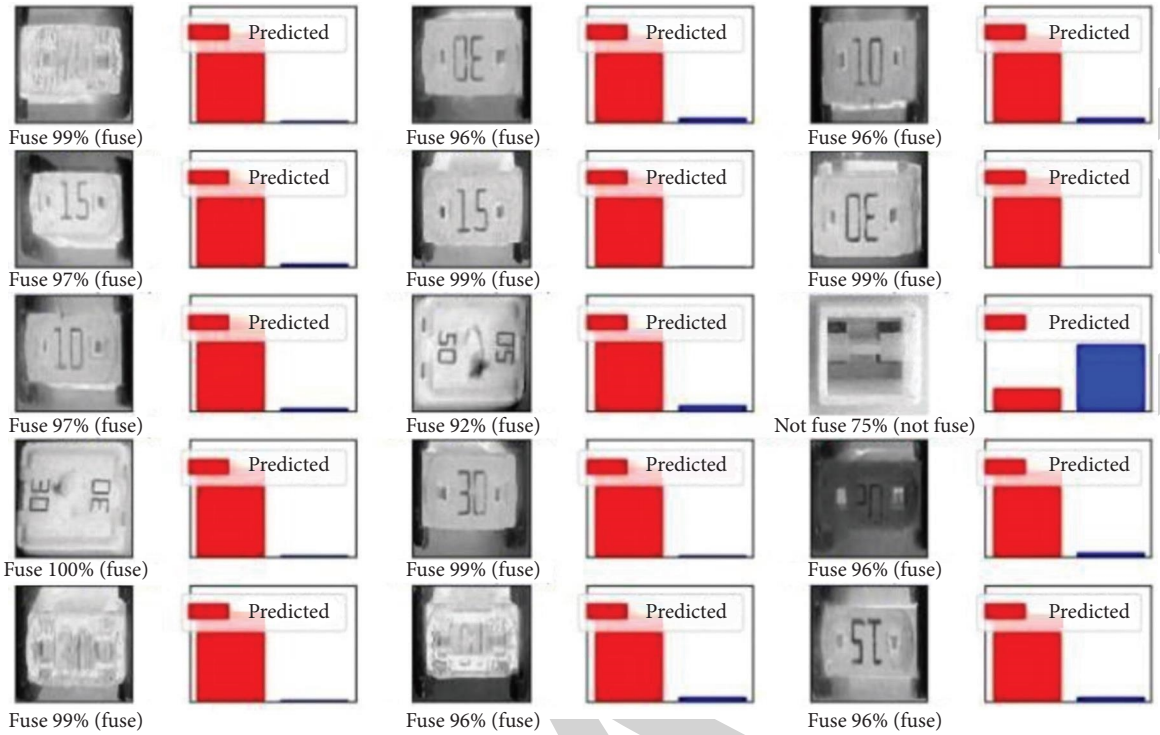

Figure 13: Complete predictions on the test data set (CNN).

prediction of the test data set is satisfactory because, on average, all artifacts required have a confidence level of more than 96%.

5.6. Accuracy and Loss Check (CNN). Figures 14(a) and 14(b) show the accuracy and failure of the test and training data check. The accuracy of the training data is almost 0.8, and the accuracy of the test data is 0.9. Of course, the loss is initially high and gradually decreases for training and test results. If the number of periods increases and the size of the data set increases, the results will of course be updated. Or at least increase the number of periods to further improve efficiency. In the training process, more data and more opportunities, direct loss, and accuracy provide more conditions for the model. In this way, the network can learn and create more spaces between the target and the nontarget.

5.7. Classification of Backpropagation Network. Backpropagation may be the second choice of most python programmers, because python does not have a built-in framework or library to implement it easily. However, python scripts must be run to implement the backpropagation model. In other words, BP is not a simple network. Most of the steps taken are the same as CNN's image preparation, training process, and evaluation process. The difference is that the size of the input node during the network feed is 200×200 to accommodate the input size data of each image. The key steps taken to train the network are as follows. The overall BP equation for this problem is given as follows:

$$\begin{aligned}
\delta^{Lj} &= \frac{\partial C}{\partial a^{Lj}}\sigma'\left(Z^{Lj}\right),\\
\delta^{1} &= \left(\left(w^{1}+1\right)T\delta^{1}+1\right)^{*}\sigma'\left(Z^{1}\right),\\
\frac{\partial C}{\partial b^{1j}} &= \partial^{1j},\\
\frac{\partial C}{\partial w^{1jk}} &= a^{1-1}k^{\partial^{1j}},
\end{aligned} \tag{3}$$

where k represents overall nodes in the layers above node j, dj is the expected output of node j, and yj is the actual output. In addition, w is the weight between the inputs, and δL is the only data required to calculate the weight gradient of layer L, and then we can calculate the previous ∂^{L-1} layer and replicate it recursively, and σ is the error term of node j.

(1) Preparing data (BP): just like preparing data for CNN, the same measures are taken to prepare data for BP network. Take the same data set and the same dimensions; nothing has changed. The division of the training test is 20% of the data used for testing and 80% of the data used for preparation. First, you need to extract each fuse object from the main photo and save it to our data set. The next step is to resize it to a small image of 200×200. Therein, we convert RGB to grayscale to save training time.

If the command is executed, python will display the progress of the training process. Each epoch will show the learning rate and the elapsed time (in

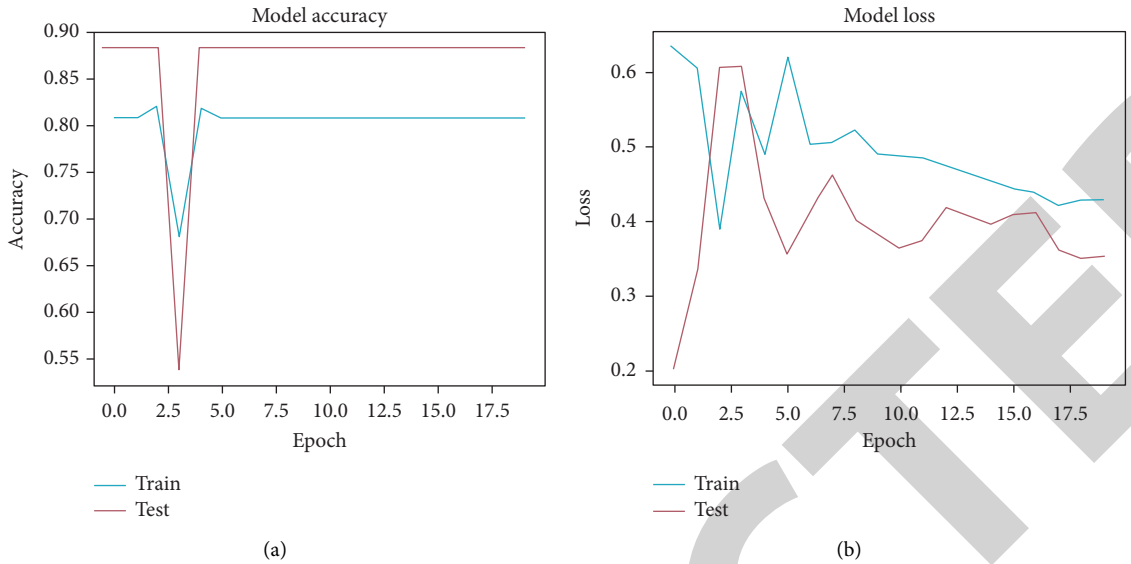

Figure 14: (a) Accuracy of the model. (b) Loss of the model.

seconds). The BP network must automatically change the learning rate in each period because it must fit the input vector and output, and then measure the error. Based on the error, the learning rate will be adjusted and the network will be tested in the next period. The training phase is shown in Table 6.

(2) Predictions (BP): if the training process is completed, BP will use the network test data set to test how many accurate predictions the network produces. The results of the predicted samples are shown in Table 7. The following prediction shows that these two objects are not fusion objects, and the rest are fuse objects on the network.

(3) Complete predictions on test data set (BP): once the identification of a single image test is completed, it is now possible to test the entire data set on the network and view the overall results. If the model is wrong, or at a certain stage it is uncertain whether the model is the desired goal, the vertical line will reflect the trustworthiness of the model. For example, if the model determines that 96% is the required target, the first bar will be almost complete, and the second bar will be almost invisible in the forecast chart. However, if the model predicts that the object accounts for 51% of the appropriate object, the height of the two bars will be almost the same. The entire result of the test data set is shown in Figure 8. The prediction on the test data set is good because the confidence of all necessary items is not too high, and the average is only slightly higher than 51%.

(4) MSE and learning rate (BP): Figures 15(a) and 15(b) show the relationship between MSE (mean square error), learning rate, and accuracy of training results. When the learning rate is maintained at 0.5, the MSE decreases rapidly and then becomes unstable but still remains high and low. In addition, since no single object is detected, the accuracy is 0%. When the learning rate drops to 0.01, the MSE drops rapidly and remains consistent throughout the rest of the period. At the same time, because the model can detect target and nontarget objects at the same time,

Table 6: Sample epochs from training (BP).

Epoch	Learning rate	Elapse time
1/60	0.000683	207.932893
3/60	0.000676	213.214195
7/60	0.000669	218.458495
10/60	0.000662	223.785800
13/60	0.000656	229.071102
16/60	0.000649	234.325403
18/60	0.000643	239.591704
20/60	0.000636	244.842004
25/60	0.000630	250.073303
30/60	0.000624	255.384607

Table 7: Predictions on the test data set.

True label	Predicted label	Predicted items	Detected class
[1 0]	[0.50812526 0.49187474]	[1.0 0.0]	Fuse
[1 0]	[0.50812513 0.49187487]	[1.0 0.0]	Fuse
[1 0]	[0.50812734 0.49187266]	[1.0 0.0]	Fuse
[1 0]	[*h*0.50801862 0.49198138]	[1.0 0.0]	Fuse
[1 0]	[0.50801227 0.49198773]	[1.0 0.0]	Fuse
[1 0]	[0.50800363 0.49199637]	[1.0 0.0]	Fuse
[1 0]	[0.50801598 0.49198402]	[1.0 0.0]	Fuse
[1 0]	[0.5080938 0.4919062]	[1.0 0.0]	Fuse
[1 0]	[0.44388368 0.55611632]	[0.0 1.0]	Not fuse
[1 0]	[0.50800662 0.49199338]	[1.0 0.0]	Fuse
[1 0]	[0.50801372 0.49198628]	[1.0 0.0]	Fuse

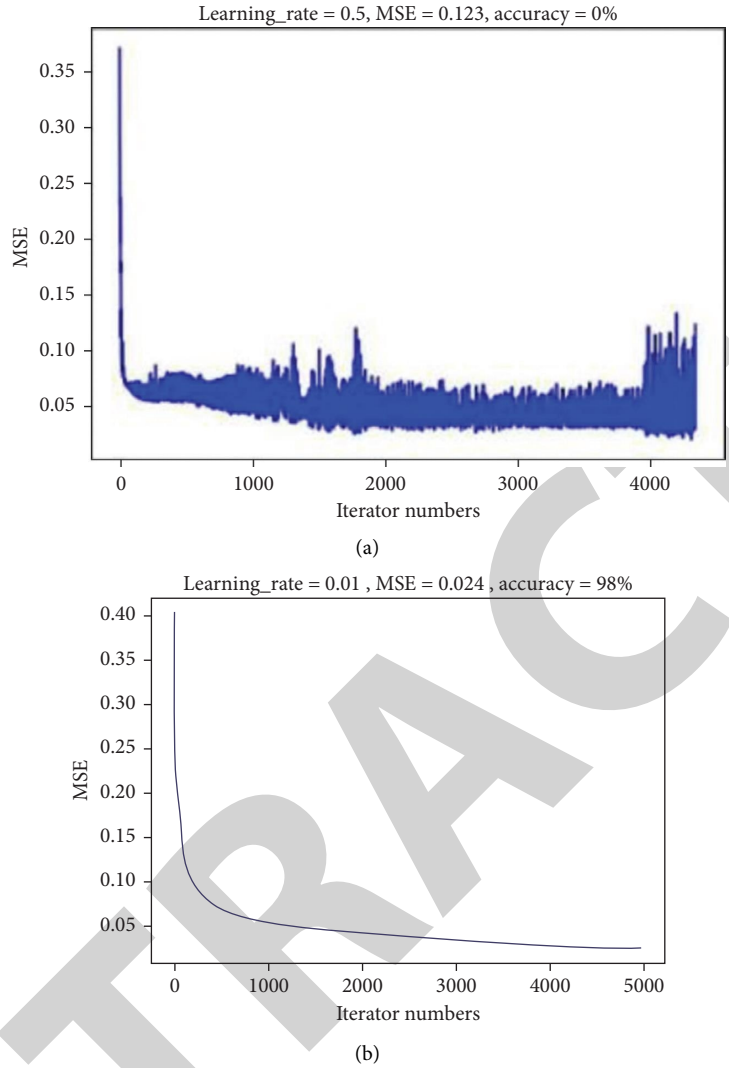

Figure 15: (a) MSE versus L-Rate = 0.5. (b) MSE versus L-Rate = 0.01.

the accuracy is increased to 98%. The accuracy of the training data is almost 0.61, and the test data is about 0.51. Initially, of course, for both the training and the test results, the loss is very high and gradually decreases, but, by increasing the number of periods, the performance will be further improved.

6. Results and Discussion

Based on the above data, it is not difficult to conclude that K-NN has the best model among the three models. According to training time, prediction time, and prediction accuracy, KNN performed well and provided a result of 98%. As you can see, each model in the table below has several parameters. The first is whether the model needs to preprocess the image, and the second parameter is the training time, which is one of the most important parameters in neural network training. The third key parameter is estimated time. This parameter is very important for real-time systems, because the prediction must be made in real time, so delayed prediction may cause a big problem; and combine all the parameters in the final parameters with the percentage accuracy. Therefore, looking at all the data below, it is easy to assume that the performance of K-NN is very effective and that the predicted result is much higher than the planned performance. In addition, from an application point of view, processing time, memory, and resource requirements play a key role in selecting a classifier, and K-NN can provide results comparable to results that are computationally inexpensive and easy to classify.

For the computationally demanding CNN and BP, when selecting a classifier, applications where processing time and resource requirements are key considerations should also consider hardware capabilities. Table 8 and Figure 16 show the overall comparative analysis of these three models. Throughout the research process, we came to the conclusion that, compared with traditional K-NN, CNN, and BP, in terms of accuracy (%), time complexity (min), and processing high dimensionality, performance-based modified K-NN is a more effective model data set.

Table 8: Overall comparisons of the three algorithms with modified K-NN in fuse detection.

Deep learning models	Accuracy (%)	Time complexity (mins)	Required data processing	Handle high-dimensional data set
Modified K-NN	98–100	15	Yes	Yes
Classical K-NN	96–98	20	Yes	Yes
CNN	96	30	No	No
BP	51	25	No	No

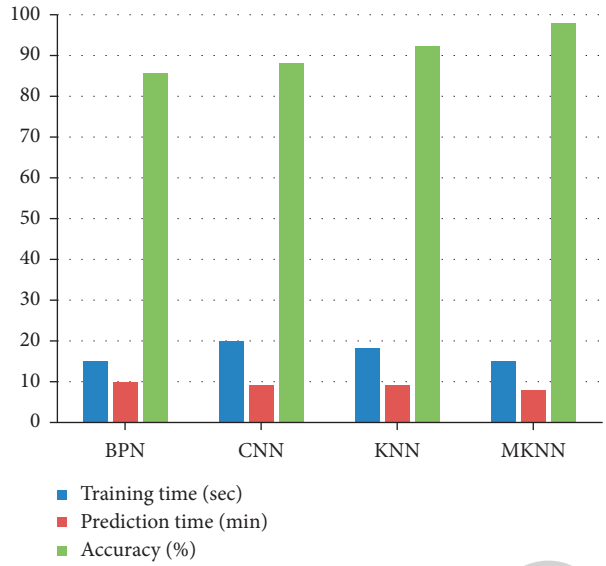

Figure 16: Overall comparisons of CNN, BP, and K-NN.

7. Conclusion

This paper proposes an improved K-NN (MK-NN) model to improve object detection in E-healthcare applications. We analyzed the proposed MK-NN model through CNN, BP, and traditional K-NN models to find out which model is effective after multiple training and testing rounds. Our analysis results show that, based on the key parameters of the network, training time, prediction time, and prediction accuracy, MK-NN is the best model among the four models. MK-NN performs well in training time and prediction time (20 seconds, 20 minutes) and provides 98% accuracy. Future research should discover and incorporate the proposed Model Architects (MK-NN) model and compare the performance in other computer vision applications. Similarly, an interesting implementation is to use multilayer output to fully connect the layers to increase the vector size of the function to improve performance.

Data Availability

Since the funding project is not closed and related patents have been evaluated, the simulation data used to support the findings of this study are currently under embargo, while the research findings are commercialized. Requests for data, upon the approval of patents after project closure, will be considered by the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

Omar Cheikhrouhou acknowledges Taif University for its support under the project Taif University Researchers Supporting Project (TURSP-2020/55), Taif University, Taif, Saudi Arabia.

References

[1] J. Zhang, M. Huang, X. Jin, and X. Li, "A real-time Chinese traffic sign detection algorithm based on modified yolov2," *Algorithms*, vol. 10, no. 4, p. 127, 2017.

[2] A. Mogelmose, M. M. Trivedi, and T. B. Moeslund, "Vision-based traffic sign detection and analysis for intelligent driver assistance systems: perspectives and survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1484–1497, 2012.

[3] I. Ullah, Y. Liu, X. Su, and P. Kim, "Efficient and accurate target localization in underwater environment," *IEEE Access*, vol. 7, pp. 101415–101426, 2019.

[4] J. Zhang, Q. Huang, H. Wu, and Y. Liu, "A shallow network with combined pooling for fast traffic sign recognition," *Information*, vol. 8, no. 2, p. 45, 2017.

[5] B. Moustapha, "The effect of propagation delay on the dynamic evolution of the bitcoin blockchain," *Digital Communications and Networks*, vol. 6, no. 2, pp. 157–166, 2020.

[6] Y. Wu, Y. Liu, J. Li, H. Liu, and X. Hu, "Traffic sign detection based on convolutional neural networks," in *Proceedings of the 2013 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Dallas, TX, USA, 2013.

[7] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German traffic sign recognition benchmark: a multi-class classification competition," in *Proceedings of the 2011 International Joint Conference on Neural Networks*, IEEE, San Jose, CA, USA, 2011.

[8] M. T. Sadiq, X. Yu, Z. Yuan et al., "Motor imagery EEG signals decoding by multivariate empirical wavelet transform-based framework for robust brain–computer interfaces," *IEEE Access*, vol. 7, pp. 171431–171451, 2019.

[9] H. H. Aghdam, E. J. Heravi, and D. Puig, "A practical approach for detection and classification of traffic signs using convolutional neural networks," *Robotics and Autonomous Systems*, vol. 84, pp. 97–112, 2016.

[10] I. Ullah, S. Qian, Z. Deng, and J.-H. Lee, "Extended Kalman filter-based localization algorithm by edge computing in wireless sensor networks," *Digital Communications and Networks*, 2020, In press.

[11] X. Yao, F. Farha, R. Li, I. Psychoula, L. Chen, and H. Ning, "Security and privacy issues of physical objects in the IOT: challenges and opportunities," *Digital Communications and Networks*, 2020, In press.

[12] L. Bourdev and J. Malik, "Poselets: body part detectors trained using 3D human pose annotations," in *Proceedings of the 2009*

IEEE 12th International Conference on Computer Vision, IEEE, Kyoto, Japan, 2009.

[13] P. Ren, W. Fang, and S. Djahel, "A novel Yolo-based real-time people counting approach," in *Proceedings of the 2017 International Smart Cities Conference (ISC2)*, IEEE, Wuxi, China, 2017.

[14] X. Zhao, Y. Ni, and H. Jia, "Modified object detection method based on Yolo," in *Proceedings of the CCF Chinese Conference on Computer Vision*, 2017.

[15] Y. LeCun, B. Boser, J. S. Denker et al., "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[16] S. Yu, J. Liu, J. Wang, and I. Ullah, "Adaptive double-threshold cooperative spectrum sensing algorithm based on history energy detection," *Wireless Communications and Mobile Computing*, vol. 2020, Article ID 4794136, 12 pages, 2020.

[17] X. Su, I. Ullah, X. Liu, and D. Choi, "A review of underwater localization techniques, algorithms, and challenges," *Journal of Sensors*, vol. 2020, Article ID 6403161, 24 pages, 2020.

[18] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[19] F. Jameel, W. U. Khan, M. A. Jamshed, H. Pervaiz, Q. Abbasi, and R. Jantti, "Reinforcement learning for scalable and reliable power allocation in SDN-based backscatter heterogeneous network," in *Proceedings of the 2020 EEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, Toronto, Canada, 2020.

[20] W. U. Khan, X. Li, M. Zeng, and O. A. Dobre, "Backscatter-enabled NOMA for future 6G systems: a new optimization framework under imperfect SIC," *IEEE Communications Letters*, vol. 25, no. 5, pp. 1669–1672, 2021.

[21] F. Lin, J. Liu, Q. Wu et al., "FMNet: feature mining networks for brain tumor segmentation," in *Proceedings of the 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, Portland, OR, USA, 2019.

[22] W. U. Khan, X. Li, A. Ihsan, M. A. Khan, V. G. Menon, and M. Ahmed, "NOMA-enabled optimization framework for next-generation small-cell IoV networks under imperfect SIC decoding," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[23] X. Li, Y. Zheng, W. U. Khan et al., "Physical layer security of cognitive ambient backscatter communications for green internet-of-things," *IEEE Transactions on Green Communications and Networking*, p. 1, 2021, In press.

[24] W. U. Khan, F. Jameel, N. Kumar, R. Jantti, and M. Guizani, "Backscatter-enabled efficient V2X communication with non-orthogonal multiple access," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1724–1735, 2021.

[25] F. Jameel, W. U. Khan, N. Kumar, and R. Jantti, "Efficient power-splitting and resource allocation for cellular V2X communications," *IEEE Transactions on Intelligent Transportation Systems*, p. 1, 2020, In press.

[26] W. U. Khan, J. Liu, F. Jameel, V. Sharma, R. Jantti, and Z. Han, "Spectral efficiency optimization for next generation NOMA-enabled IoT networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 15284–15297, 2020.

[27] F. Jameel, S. Zeb, W. U. Khan, S. A. Hassan, Z. Chang, and J. Liu, "NOMA-enabled backscatter communications: toward battery-free IoT networks," *IEEE Internet of Things Magazine*, vol. 3, no. 4, pp. 95–101, 2020.

[28] W. U. Khan, F. Jameel, M. A. Jamshed, H. Pervaiz, S. Khan, and J. Liu, "Efficient power allocation for NOMA-enabled IoT networks in 6G era," *Physical Communication*, vol. 39, Article ID 101043, 2020.

[29] Z. Ali, W. Farooq, W. U. Khan, M. Qureshi, and G. A. S. Sidhu, "Artificial intelligence techniques for rate maximization in interference channels," *Physical Communication*, vol. 47, Article ID 101294, 2021.

[30] W. U. Khan, F. Jameel, T. Ristaniemi, S. Khan, G. A. S. Sidhu, and J. Liu, "Joint spectral and energy efficiency optimization for downlink NOMA networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 2, pp. 645–656, 2019.

[31] S. Yu, W. U. Khan, X. Zhang, and J. Liu, "Optimal power allocation for NOMA-enabled D2D communication with imperfect SIC decoding," *Physical Communication*, vol. 46, Article ID 101296, 2021.

[32] W. U. Khan, N. Imtiaz, and I. Ullah, "Joint optimization of NOMA -enabled backscatter communications for beyond 5G IoT networks," *Internet Technology Letters*, vol. 4, no. 2, p. e265, 2021.

[33] W. Li, Y. Wang, Z. Jin, K. Yu, J. Li, and Y. Xiang, "Challenge-based collaborative intrusion detection in software defined networking: an evaluation," *Digital Communications and Networks*, 2020, In press.

[34] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, IEEE, San Diego, CA, USA, 2005.

[35] R. A. Naqvi, M. Arsalan, A. Rehman, A. U. Rehman, W.-K. Loh, and A. Paul, "Deep learning-based drivers emotion classification system in time series data for remote applications," *Remote Sensing*, vol. 12, no. 3, p. 587, 2020.

[36] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *Proceedings of the 2002 International Conference on Image Processing*, vol. 1, IEEE, Rochester, NY, USA, 2002.

[37] I. Ullah, X. Su, J. Zhu, X. Zhang, D. Choi, and Z. Hou, "Evaluation of localization by extended Kalman filter, unscented Kalman filter, and particle filter-based techniques," *Wireless Communications and Mobile Computing*, vol. 2020, Article ID 8898672, 15 pages, 2020.

[38] C. Cortes and V. Vapnik, "Support vector machine," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[39] Y. Freund and R. Schapire, "A decision-theoretic generalization of online learning and an application to boosting," in *Proceedings of the 1995 European Conference on Computational Learning Theory (EuroCOLT)*, Barcelona, Spain, 1995.

[40] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models. Sion multimodal latent dirichlet allocation for image annotation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, 2010.

[41] E. Sackinger, B. E. Boser, J. Bromley, Y. LeCun, and L. D. Jackel, "Application of the ANNA neural network chip to high-speed character recognition," *IEEE Transactions on Neural Networks*, vol. 3, no. 3, pp. 498–505, 1992.

[42] Y. Bengio, Y. LeCun, and D. Henderson, "Globally trained handwritten word recognizer using spatial representation, convolutional neural networks, and hidden Markov models," in *Advances in Neural Information Processing Systems*, pp. 937–944, MIT Press, Cambridge, MA, USA, 1994.

[43] M. M. Kamal, I. Ullah, A. Ashraf, and N. Ullah, "Designing band notch features in ultra-wideband antenna," in *Proceedings of the 2017 7th IEEE International Symposium on Microwave, Antenna, Propagation, and EMC Technologies (MAPE)*, IEEE, Xi'an, China, 2017.

[44] R. R. Swain, P. M. Khilar, and T. Dash, "Multifault diagnosis in WSN using a hybrid metaheuristic trained neural network," *Digital Communications and Networks*, vol. 6, no. 1, pp. 86–100, 2020.

[45] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: a convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997.

[46] B. Fasel, "Robust face analysis using convolutional neural networks," in *Object Recognition Supported by User Interaction for Service Robots,* vol. 2, pp. 40–43, IEEE, 2002.

[47] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of Physiology*, vol. 148, no. 3, pp. 574–591, 1959.

[48] I. Khan, Y.-b. Tian, I. Ullah, M. M. Kamal, H. Ullah, and A. Khan, "Designing of E-shaped microstrip antenna using artificial neural network," *International Journal of Computing, Communications and Instrumentation Engineering*, vol. 5, no. 1, 2018.

[49] S. K. Haider, A.-M. Jiang, S. Ashraf, and I. Ullah, "Study on detection of p300 ERP component in EEG signals and algorithms," in *Proceedings of the 3rd Annual International Conference on Electronics, Electrical Engineering and Information Science (EEEIS 2017)*, Atlantis Press, Gangzhou, China, 2017.

[50] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale convolutional networks," in *Proceedings of the 2011 International Joint Conference on Neural Networks*, IEEE, San Jose, CA, USA, 2011.

[51] I. Khan, Y.-B. Tian, Inamullah, H. Vllah, S. U. Rahman, and M. M. Kamal, "Design annular ring microstrip antenna based on artificial neural network," in *Proceedings of the 2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, IEEE, Xi'an, China, 2018.

[52] I. Ullah, Y. Shen, X. Su, C. Esposito, and C. Choi, "A localization based on unscented Kalman filter and particle filter localization algorithms," *IEEE Access*, vol. 8, pp. 2233–2246, 2019.

[53] A. Aslanargun, M. Mammadov, B. Yazici, and S. Yolacan, "Comparison of arima, neural networks and hybrid models in time series: tourist arrival forecasting," *Journal of Statistical Computation and Simulation*, vol. 77, no. 1, pp. 29–53, 2007.

[54] K. Zhou and Y. Kang, *Neural Network Model and Matlab Simulation Program Design*, Publishing House of Tsinghua University, Beijing, China, 2005.

[55] E. Blanzieri and F. Melgani, "Nearest neighbor classification of remote sensing images with the maximal margin principle," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 6, pp. 1804–1811, 2008.

[56] J. Chen and J. Shao, "Jackknife variance estimation for nearest-neighbor imputation," *Journal of the American Statistical Association*, vol. 96, no. 453, pp. 260–269, 2001.

[57] A. Satheesh and R. Patel, "Dynamic nearest neighbours classifier for integrated data using object oriented concept generalization," *International Journal of Simulation-Systems, Science & Technology*, vol. 11, no. 1, 2010.

[58] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[59] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Anchorage, AK, USA, June 2008.

[60] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, 2014.

[61] D. Cao, Z. Chen, and L. Gao, "An improved object detection algorithm based on multi-scaled and deformable convolutional neural networks," *Human-Centric Computing and Information Sciences*, vol. 10, p. 14, 2020.

[62] J. Zhang, S. Zhou, X. Chang et al., "Multiple object tracking by flowing and fusing," 2020, https://arxiv.org/abs/2001.11180.