Available online at www.qu.edu.iq/journalcm
JOURNAL OF AL-QADISIYAH FOR COMPUTER SCIENCE AND MATHEMATICS
ISSN:2521-3504(online) ISSN:2074-0204(print)

Fire Detection by using DenseNet 201 algorithm and Surveillance Cameras images

Satar Shaker Muhammad[a] , Jamal M. Alrikabi [b]*

a Directorate General of Education in Thi Qar Governorate, Iraq. Email: satar.shaker@utq.edu.iq

b Department of Computer Science, College of Education for Pure Science / University of Thi-Qar, Nasiriyah, Iraq. Email: jama113_mah@utq.edu.iq

ARTICLE INFO

Article history:
Received: 18 /1/2024
Rrevised form: 4 /3/2024
Accepted : 20 /3/2024
Available online: 30 /3/2024

Keywords:

deep learning;
convolutional networks;
Fire detection;
CCTV;
DenseNet 201

ABSTRACT

Early warning of fires is very important to reduce loss of life and property. It is a technology that has become one of the most important design elements for smart cities in the future. Most current solutions rely on temperature and smoke sensors, which have limited detection ranges and scenarios and long response times. These are used inside some buildings and cannot be used in the external environment, as they are not a preferred solution. In addition to the above, sensor systems are costly because of the hardware and installation requirements, and they require regular upkeep. With the development of artificial intelligence and learning machines, fire detection has been used based on deep learning. Widely. Therefore, a solution to computer vision technology is proposed, which uses a convolutional neural network (CNN), Reducing false alarms. The use of computer vision techniques is certainly better than sensor-based systems, by utilizing security camera footage at specific periods to detect a fire when it breaks out. Our system achieved excellent results with an average predictive accuracy of 98% on the dataset. This accuracy and low cost make it a good alternative to systems that use sensors. The contribution of this study is to improve the detection process by conducting a classification process instead of detecting an object (fire), which requires complex mathematical calculations, large resources, and a long time.

MSC..

https://doi.org/ 10.29304/jqcsm.2024.16.11437

1. Introduction

The demand for fire detection devices is increasing, especially in urban cities [1], due to the increased convergence of buildings and the fear of the rapid spread of fire and population density, which increases the number of casualties among the population as well as the loss of property [2]. Fire detection methods are a crucial component of making firefighting easier. and reducing the damages resulting from these accidents. It has become one of the key components of future smart city designs [1]. Previously, adults were relied upon to call the fire brigade when they spotted a fire,

∗Corresponding author: Satar Shaker Muhammad , *Jamal M. Alrikabi*

Email addresses: satar.shaker@utq.edu.iq

Communicated by

which could take a long time, or there may not be an adult present who is capable of handling the situation. Thus, the presence of an automatic detection system decreases human effort and improves the capacity to work quickly with high efficiency to prevent wasting time [3]. And helps in reducing losses and speed in putting out the fire. The majority of existing solutions depend on sensors. that are used indoors and cannot be used in the outdoor environment. Because of the hardware and installation requirements, they are pricey and require regular maintenance[4]. Because of the foregoing, the research community has been motivated to research and develop fire detection techniques using cameras [5]. Since computer vision technologies can use footage from surveillance cameras, they are a better option than sensor-based systems. Other advantages include the ability to operate this system in the external environment, and in real time. Regardless of outside limitations like the weather or environmental conditions, the detecting method needs to be dependable. Moreover, the solution is implemented at the lowest cost in terms of installation and maintenance.

2. Related work

Muhammad et al. (2017) In [6] This article proposes an early fire detection framework using accurate convolutional neural networks for CCTV cameras. The model used has a similar architecture to the AlexNet model. which can detect fires in various indoor and outdoor environments. The research paper reported an increase in accuracy from 93.55% to 94.39% and a reduction in false positives from 11.67% to 9.07% compared to other methods.

Khan Muhammad et al. (2018) In [7] The article propose a fire-detection framework using a convolutional neural network architecture with surveillance videos. The model is inspired by GoogleNet architecture. Using a sizable dataset of fire videos that were gathered from the internet and real environments, the classifier was trained and tested on it. Experimental results on standard fire datasets showed an accuracy improvement of 94.43% with fine-tuning.

Jichao Li et al. (2021) In [8] Made improvements to the YOLOv3 micro-algorithm, which aimed for accurate fire detection through multiscale fusion and k-means aggregation. There was only one application situation, and the detection speed was not optimal. As the researchers claim in this paper, this method can accurately locate fires and detect small fires effectively, showing its improved performance compared to the original YOLOv3 mini-model. This paper uses the mean accuracy rate (mAP), which achieved 92%.

Yu et al. (2021) In [5] The article presents research on an improved YOLOv3 fire detection algorithm based on enlarged feature map resolution and cluster analysis. The study focuses on enhancing the detection precision and recall rate of suspected flames, utilizing deep learning techniques and data enhancement methods. The research paper reported an accuracy of 97%.

Hao Xu et al. (2022) In [2] A Light-YOLOv5 network was employed to detect fire in complex scenarios; It utilizes a separable vision transformer, a light bidirectional feature pyramid network, a global attention mechanism, Mish activation function, and SIoU loss to enhance detection speed and accuracy. This use greatly reduced the number of parameters, which improved the speed of fire detection and increased mean-average accuracy.

Zhang et al. (2022) In [3] In their research, they proposed the T-YOLOX algorithm for fire detection through VIT technology to increase fire detection precision The method includes a light attention module channel shuffle technique and the integration of a light transformer module, without discussing the number of parameters or the computational efforts of this technique. The results of the research paper show that the proposed T-YOLOX model achieved a mean average precision (mAP) of 69.54%, which is a 2.24% increase compared to the original YOLOX algorithm.

Zhao et al. (2022) In [9] proposed an improved Fire-YOLO algorithm whose aim was to detect small forest fire targets to enhance detection and reduce the model size, but like its predecessor, computational efforts and the number of parameters were not discussed. The research paper shows that the Fire-YOLO model outperforms YOLO-V3 and Faster R-CNN in detecting small targets like fire and smoke. It achieved precision 91.50% values better compared to the other models.

Ali Khan et al. (2022) In [10] a paper on fire detection using convolutional neural networks and transfer learning. The authors propose a time-efficient model for fire detection in surveillance videos that uses architecture VGG19 with

reasonable computational time and is feasible for real-time applications. with the proposed approach achieving a mean accuracy of 95%.

3. The proposed System

The proposed system consists of three phases:

1- Preprocessing

2- Convolutional neural network

3- System Application

This system uses a monitoring camera installed, which is preferably at a suitable height, to avoid blockage. Every half minute, the image captured by these cameras is utilized as the system's input, containing the coordinates that divide the image into 6 parts, which are manually set in advance. The system segments images according to those coordinates and feeds them into a convolutional neural network (DenseNet 201), Which has been previously trained on fire images and non-fire images. The output of DenseNet 201 is the segmented image classification. The system colors the image that is classified as having a fire frame red and sounds a warning, and the image that is classified as no fire has a framed green, and it is displayed on a screen in the Incident Follow-up Control Center. The proposed system is shown in the flow chart in Fig. 1.

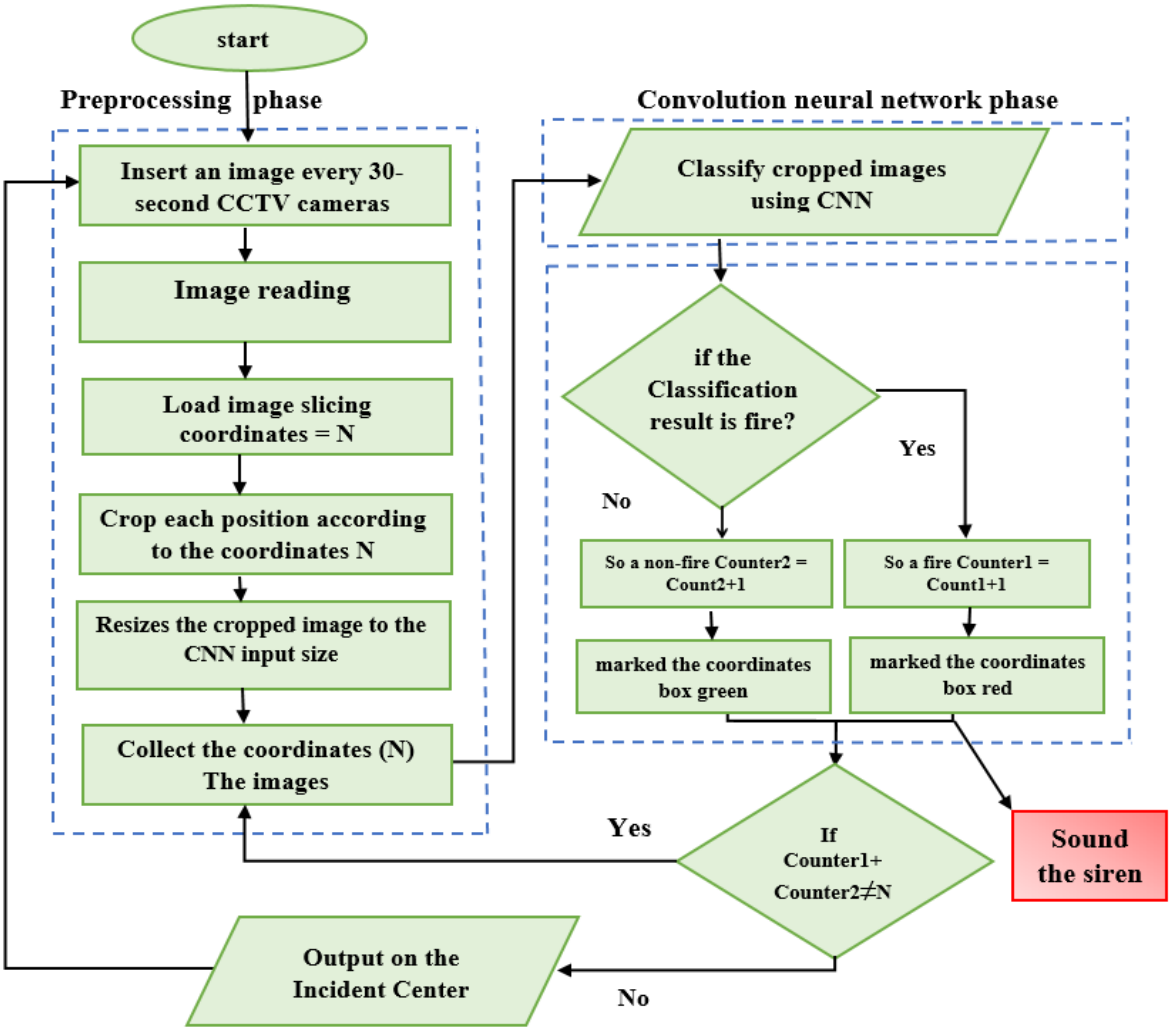

Fig. 1: The Diagram for the proposed framework (Fire Detection)

3.1-Pre Processing Stage:

Before using an image in the proposed system, pre-processing is applied to it as shown in Fig. 2. These steps are as follows:

step 1: From CCTV cameras, an image is obtained.

step 2: Enter camera coordinates that divide the image into 6 parts.

step 3: Crop the coordinates of segmenting the image into 6 segmenting images.

step 4: The image resized is adjusted to suit the algorithm of the system (DenseNet 201).

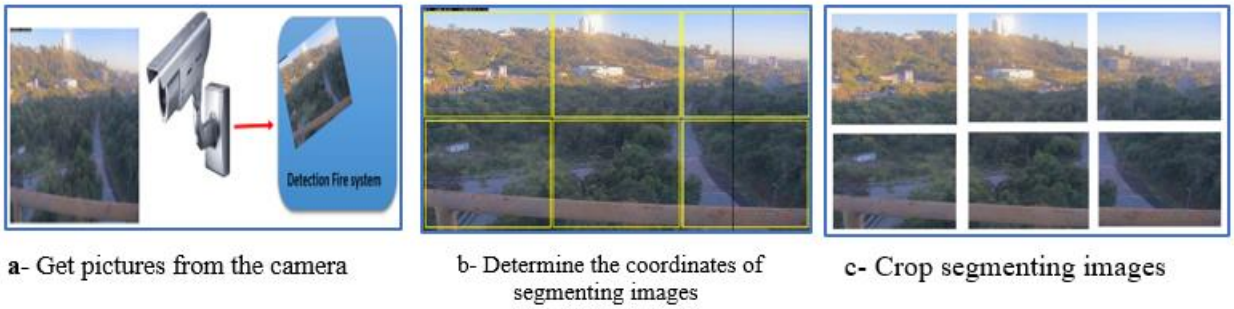

Fig. 2: Clarify Pre Processing Stage

3.2 - Convolution neural network phase:

Convolutional Neural Network CNN is a solution to many computer vision problems in artificial intelligence such as image and video processing, because of its ability to detect and understand patterns. This pattern detection makes CNN useful for image and video analysis, where it should deliver optimum performance in detection, image segmentation, and classification [11], Trained detector DenseNet 201 was used in this study to determine if it is Fire or non-fire.

3.2.1- Convolution neural network layers:

The general architecture of a CNN consists of two main parts: A Feature extractor and a classifier. In feature extraction layers, each layer takes the output from the preceding layer as input and sends its output to the following layer as input [12]. CNN is a series of layers; each layer performs its function. A CNN architecture usually consists of three main combined layers: the convolutional layer, the pooling layer, and the fully connected layers (FC). The function of the first and second layers is to extract features, and the function of the third layer is to assign the extracted features to the final output as classification [13]. The three layers are:

I- The Convolutional layer: In this layer, we perform a convolutional mathematical operation by sliding the kernel matrix over the input matrix. At each location, we perform an element-wise matrix multiplication and then sum the result onto the feature map. This layer creates a feature map for the following layer by passing a matrix named kernel over the input matrix [14]. Convolution, which can be applied over more than one axis, is a specialized type of linear operation that is widely used in many fields, such as image processing, statistics, and physics. The convoluted image is calculated if we have a 2-Dimensional image input, I, and a 2-Dimensional kernel filter, K. [15] as shown in Fig. 3.

$$S(i,j) = \sum_{m}\sum_{n} I(m,n)k(i-m,j-n) \qquad (1)$$

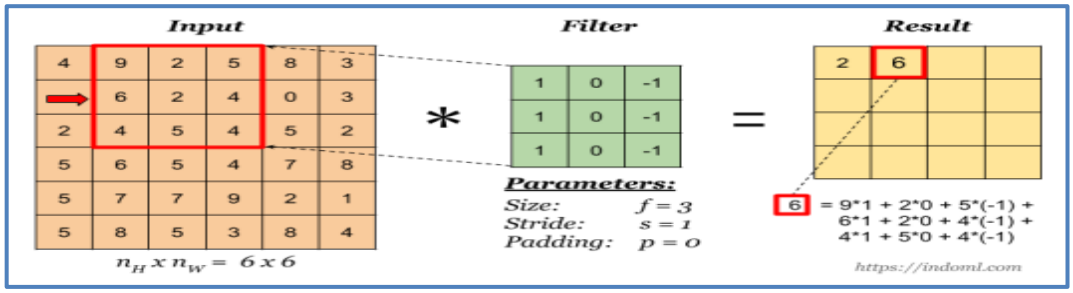

Fig. 3: element-wise matrix multiplication in the convolutional layer [15]

II – The Pooling layer: also known as the downsampling layer, is a crucial layer that creates new feature maps with a condensed resolution by down-sampling the feature maps from the previous layer. The pooling operation applies a filter throughout the entire input, just like the convolutional layer does, thereby reducing the spatial dimension of the input. The distinction is that this filter has no weights; instead, the kernel applies an aggregation function to the values in the receptive field, filling the output array [16].

It accomplishes two basic goals. Firstly, it lowers the computational cost by reducing the number of parameters or weights. The second is managing the network's overfitting. An optimal pooling strategy should extract only important data and eliminate unnecessary features.

There are a lot of methods for the implementation of pooling operations, However, there are two primary forms of pooling [17]:

1. Max pooling: The filter picks the pixel with the maximum value to send to the output matrix as it passes through the input. as shown in Fig. 4.A. It is generally more common to employ this strategy than average pooling.

2. Average pooling: As the filter moves through the input, the average value in the future field is computed and sent to the output array, as shown in Fig. 4.B.

$$x_j^l = \text{down}(x_j^{l-1}) \quad (2)$$

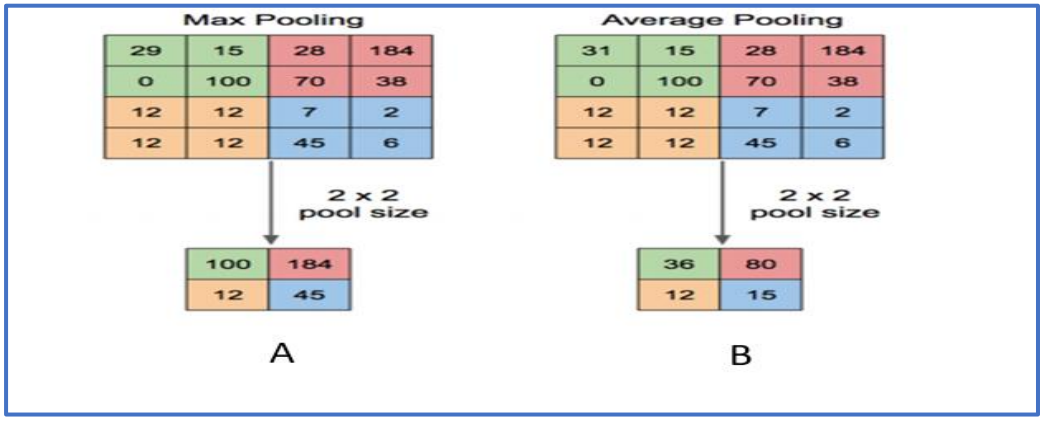

Fig. 4: Types of pooling [15].

III –Fully connected layers: A completely linked network called the classification layer receives its input from the output of the CNN's final layer or the output of the final assembly layer [18]. The fully connected layer's name effectively sums up what it is. The output layer node in the Fully Connected Layer (FC), on the other hand, is directly connected to every node in the partially linked layers, where the pixel values of the input image are not directly related to those of the output layer. This layer uses features taken from earlier layers and their different filters to complete a classification task [19].

ReLU functions are typically used by convolutional and aggregate layers, whereas SoftMax (SM) activation functions are typically used by FC layers for proper classification, which results in a probability from 0 to 1. The FC layer goes through a transformation process from a map of 2D features to a 1-D feature vector. Furthermore, the resulting vector is either categorized as classes for classification or the feature vector is further processed [20].

3.2.2 DenseNet 201

It is nothing but a convolutional neural network (CNN) architecture that has shown promising results in various computer vision tasks, including image classification and object detection. It is designed to solve the gradient fading problem. In a DenseNet architecture, each layer takes input from the previous layer, and then features from the previous layer pass to the next layer. Each layer is densely connected to all subsequent layers[21]. The conceptual density graph is illustrated in Fig. 5. Therefore, it is formed by dense communication between layers, this effective concept of feature reuse greatly reduces network parameters. DenseNet is made up of multiple dense blocks and transition blocks positioned in between two neighboring dense blocks. Every layer receives as input all of the prior feature maps.

The layers in DenseNet201 are very narrow, i.e. 12 filters, which adds a lower set of new feature maps [22]. It is 201 layers deep, and DenseNet201 connects all layers in a feed-forward fashion (inside each dense block); for every layer, all previous layers' feature maps are used as input, and all following layers' feature maps are used as input to all prior layers. DenseNet201 provides direct connections between any two layers of the same size as a feature map [21], and the input image size for the network is 224 x 224, DenseNet201 has several compelling advantages: it mitigates the vanishing scaling problem, strengthens feature spread, encourages feature reuse, greatly reduces parameter counts, and requires much lower computation requirements and a much smaller number of parameters than the latest offerings [23]. Fig. 6 shows the structures of DenseNet-201.

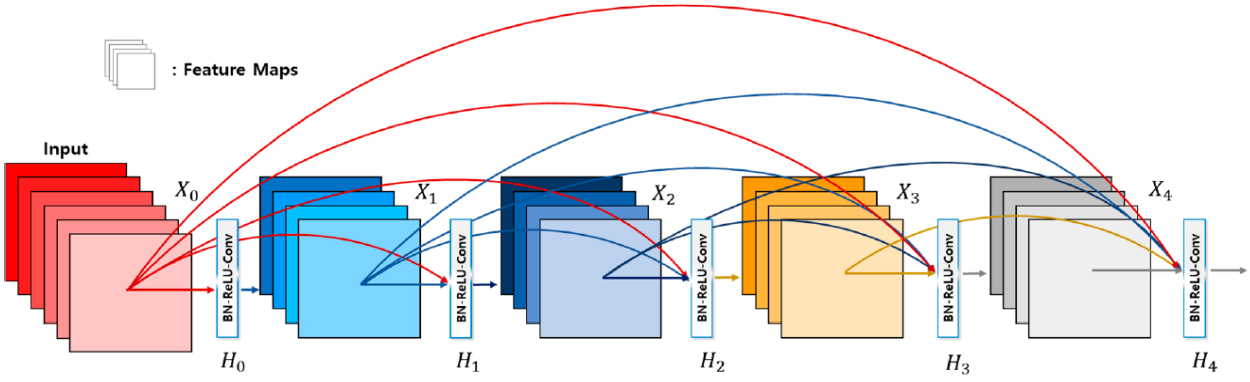

Fig. 5: Illustration of Dense Communication Between Layers[24]

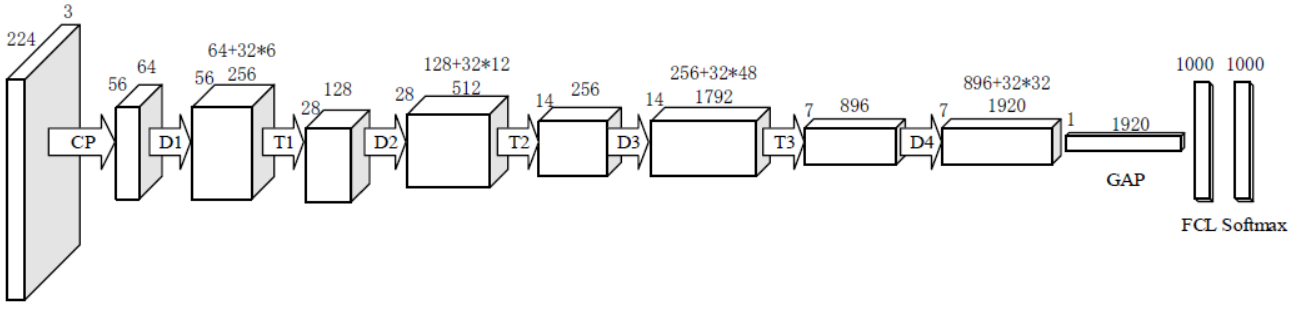

Fig. 6: Illustration of DenseNet201 Architecture[25]

3.2.3Training of DenseNet201

Training a DenseNet 201 network involves using the backpropagation algorithm, which is a neural network training technique that reduces discrepancies between output predictions and supplied ground-truth labels on a training dataset by identifying kernels in convolutional layers and weights in FC layers. RMSProp (Root Mean Square Propagation) also maintains per-parameter learning, where the Learnable parameters, like kernels and weights, are changed by the loss value through backpropagation and gradient descent. A loss function, obtained through forwarding propagation on a training dataset, determines a model's performance under certain kernels and weights. The loss function and gradient descent optimization algorithm play crucial roles. as shown in Table 1. The DenseNet201 model contains 201 layers on dense block 1, transition layer 1, dense block 2, transition layer 2, dense block 3, transition layer 3, dense block 4, and classification layer processes that produce the output model. While kernels are the only parameters automatically learned during the training process in the convolutional layer, other hyper-parameters that must be set before the training process begins include the size of the kernels, the number of kernels, padding, and stride. The process of training a CNN model with regard to the convolutional layer involves identifying the kernels that work best for a given task based on a given training dataset [26]. During the training phase, the data is divided into training data and test data according to a predetermined percentage, where all training and test samples are chosen at random to carry out the CNN training procedure on the dataset. The proportion of training data used in this research was adopted (80%), while the percentage of test data was (20%).

The pre-trained DenseNet 201 contains 1000 categories and is currently not suitable for predictions. Therefore, we need to replace the current output layer with the corresponding two-class layer to make it “Fire” or “Non-Fire”. In this work, a single neuron was used with a sigmoid activation function. This function outputs a probability score ranging between 0-1, which leads to the possibility that the input belongs to one of the two categories. This requires adjusting the loss function, as an appropriate function was used as mentioned previously to ensure that the model training process is consistent with the new target. It is then retrained through fine-tuning through our dataset, which helps the model adjust the weights it has learned to improve performance under the new CNN configuration.

Table 1: DenseNet 201 network parameters

Parameter Name	Parameter Value
Learning_rate	0.00001
Max Epochs	2
MiniBatchSize	8
InitialLearnRate	1e-4

Fig. 7: The result of training the DenseNet201 model

3.2.4 Test of DenseNet201

After the training is finished, the image of the new unsorted (unclassified) is tested using the CNN classifiers that were developed during the training phase, and the image is classified as either fire or non-fire.

3.3 System Application:

An image is entered from the monitoring camera into the convolutional neural network (trained classifier) to get the classification result. The classification stage is the final stage of the convolutional neural network, in which the image is obtained from the surveillance cameras every 30 seconds. We loaded the coordinates of the cameras, which were previously selected, and the image was divided into 6 bounding boxes, where these selections are in the form of bounding boxes The bounding box is defined by [x, y, w, h], where [x, y] represents one corner of the square and w and h denote the width and height of the squares. It will cut 6 segmented images from each image based on those coordinates. After cropping, the images are resized to" match the CNN input size (DenseNet 201). Then, these cropped images will be passed to the CNN network for classification as either fire or non-fire.

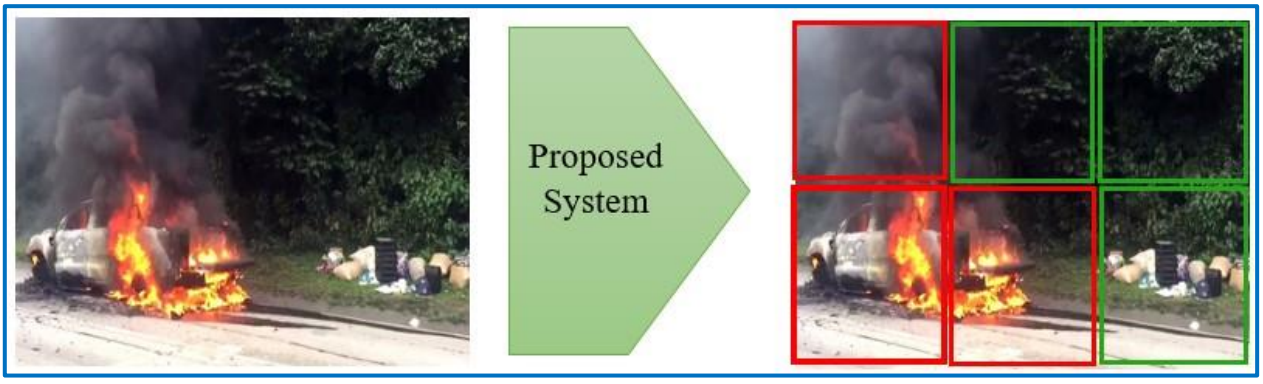

Fig. 8: Classification by Fire Detection System.

4. Datasets:

Due to the lack of reliable fire detection datasets, the dataset used in this paper is derived from the public dataset (D-Fire-001) [27], which contains 4306 test images and 17221 training images, and the public dataset (fire_dataset) [28]. was also used, which contains 755 fire_images and 244 non_fire_images. 155 images were selected from the public data sets (D-Fire- 001), and 195 images from (fire_dataset), for a total of 350 images. Each image was cut according to Coordinates into 24 segmented images, equivalent to 14,583 segmented images that were sorted into 2,507 fire images and 12,076 non-fire images. These images were used for training. As for examining the operation of the system, 195 images from the public data set (fire_dataset) were used. This dataset that we used contains different scenes, including forest fires, urban fires, indoor fires, car fires, and so on. A sample of the data set is shown in Fig. 7.

a- A sample of a forest fire dataset

b- A sample from a city fire dataset

Fig. 9: A sample of the study's data set [27]

5. Evaluation criteria

To evaluate the proposed system, Accuracy, precision, sensitivity, specificity, and F1-Score—five widely used metrics—were computed to assess the suggested method [29]. The most logical performance metric is accuracy, which is calculated as the ratio of correctly predicted observations to all observations. The precision measures how many positive observations were successfully predicted out of all the positive observations that were expected. Sensitivity can be defined as the ratio of true observations to accurately predicted positive observations. In terms of statistics, specificity is the ratio of accurately predicted negative observations to all expected negative observations. An accurate way to assess a model's performance is to look at its F1 score, which is the harmonic average of precision and sensitivity. Regarding symbols in equations True positive (TP) refers to the number of sub-images that are categorized as fire and are in fact fire; True negative (TN) refers to the number of non-fire sub-images that are classified as non-fire, and false positive (FP) refers to the number of sub-images that are classified as fire but are not fire. The number of sub-images that are classified as non-fire but are actually fire is known a (FN) false negative. Here are the expressed values for the assessment metrics. The expressions for the evaluation metrics are presented as follows.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad (2)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (3)$$

$$\text{F-score} = \frac{2 \times precision \times \text{recall}}{\text{precision} + \text{recall}} \quad (4)$$

6. Training Environment and Details

An MSI personal computer running Windows 11 Home with an Intel(R) Core (TM) i7-10750H@ 2.60 GHz CPU, 16 GB RAM, 64-bit operating system, and GPU (RTX3060) was used to run the suggested setup. DenseNet201 was easily tested and trained using the MATLAB R2021a language.

7. Evaluation results

By employing the suggested regime, we were able to attain a 98% accuracy rate with 195 images from the dataset. These photographs featured a variety of situations, such as car fires, urban flames, indoor fires, and forest fires. Additionally, our suggested approach achieved a High accuracy When applying the primary classification criterion using photos from the dataset, a DenseNet201 network trained on the dataset is used to transfer learning on previously unseen images Table 2.

Table 2: Results of the classification criteria of model.

Evaluation criteria	Evaluation
Accuracy	98 %
precision	98 %
Recall	95 %
F-measure	96 %

8. Advantages of Fire Detection using DenseNet 201

The use of DenseNet 201 algorithm for fire detection has several advantages, including:

1. Improved Accuracy: DenseNet 201 algorithm has been shown to achieve high accuracy on image classification tasks, which makes it suitable for fire detection.

2. Real-time Detection: The algorithm can detect fires in real-time, which is crucial in fire safety and prevention.

3. Cost-effective: The use of DenseNet 201 algorithm is cost-effective compared to other machine learning algorithms, which makes it suitable for deployment in real-world scenarios.

9. Compared with other models

The comparison between our proposed system and other fire detection systems that use deep learning is made in terms of accuracy only, because the execution time was not mentioned in other works, while in our proposed system the average detection time was half a second. Table 3 shows the results.

Table 3: Results of comparing the proposed system with other models.

References	Model	Recognition Accuracy
Muhammad et al. (2017) In [6]	Alexnet	94.39%
Khan Muhammad et al. (2018) In [7]	GoogleNet	94.44%
Yu et al. (2021) In [5]	YOLOv3	97 %
Ali Khan et al. (2022) In [10]	VGG19	95%
The Proposed System (2024)	DenseNet201	98 %

10. Conclusion

In conclusion, DenseNet 201 algorithm is a powerful tool for fire detection using surveillance cameras. Its ability to learn complex and abstract features makes it suitable for detecting fires in real-time. In this paper, an image-based framework is created for fire detection in the external and internal environment using a pre-trained Convolution neural network. The system attained a high degree of accuracy of 98% with an average execution time of half a second, on the data set used, suggesting its applicability where the framework can provide a reliable and cheap solution for detecting fire systems in indoor and outdoor environments. However, there are some performance-limiting challenges, for example, red and fire-colored objects, fire-like sunlight scenarios, and fire-colored lighting in different buildings, by biasing the training data used.

References

[1] R. Giffinger, "Smart cities ranking: an effective instrument for the positioning of the cities?," *ACE Archit. City Environ.*, pp. 703–714, 2010, doi: 10.5821/ace.v4i12.2483.

[2] H. Xu, B. Li, and F. Zhong, "Light-YOLOv5: A Lightweight Algorithm for Improved YOLOv5 in Complex Fire Scenarios," *Appl. Sci.*, vol. 12, no. 23, 2022, doi: 10.3390/app122312312.

[3] J. Zhang and S. Ke, "Improved YOLOX Fire Scenario Detection Method," *Wirel. Commun. Mob. Comput.*, vol. 2022, 2022, doi: 10.1155/2022/9666265.

[4] D. M. M. J. H. J. Hayawi and S. S. Muhammad, "Detection parking Spaces by using the ResNet50 Algorithm," *J. Al-Qadisiyah Comput. Sci. Math.*, vol. 14, no. 2, pp. 1–10, 2022, doi: 10.29304/jqcm.2022.14.2.932.

[5] C. Yue and J. Ye, "Research on Improved YOLOv3 Fire Detection Based on Enlarged Feature Map Resolution and Cluster Analysis," *J. Phys. Conf. Ser.*, vol. 1757, no. 1, 2021, doi: 10.1088/1742-6596/1757/1/012094.

[6] K. Muhammad, J. Ahmad, and S. W. Baik, "Early fire detection using convolutional neural networks during surveillance for effective disaster management," *Neurocomputing*, vol. 288, pp. 30–42, 2018, doi: 10.1016/j.neucom.2017.04.083.

[7] K. Muhammad, J. Ahmad, I. Mehmood, S. Rho, and S. W. Baik, "Convolutional Neural Networks Based Fire Detection in Surveillance Videos," *IEEE Access*, vol. 6, no. c, pp. 18174–18183, 2018, doi: 10.1109/ACCESS.2018.2812835.

[8] J. Li, S. Guo, L. Kong, S. Tan, and Y. Yuan, "An improved YOLOv3-tiny method for fire detection in the construction industry," *E3S Web Conf.*, vol. 253, pp. 2–5, 2021, doi: 10.1051/e3sconf/202125303069.

[9] L. Zhao, L. Zhi, C. Zhao, and W. Zheng, "Fire-YOLO: A Small Target Object Detection Method for Fire Inspection," *Sustain.*, vol. 14, no. 9, pp. 1–14, 2022, doi: 10.3390/su14094930.

[10] A. Khan, B. Hassan, S. Khan, R. Ahmed, and A. Abuassba, "DeepFire: A Novel Dataset and Deep Transfer Learning Benchmark for Forest Fire Detection," *Mob. Inf. Syst.*, vol. 2022, 2022, doi: 10.1155/2022/5358359.

[11] L. Taylor and G. Nitschke, "Improving Deep Learning using Generic Data Augmentation," 2017, [Online]. Available: http://arxiv.org/abs/1708.06020.

[12] M. Z. Alom *et al.*, "The history began from alexnet: A comprehensive survey on deep learning approaches," *arXiv Prepr. arXiv1803.01164*, 2018.

[13] A. Patil and M. Rane, "rn RecoConvolutional Neural Networks: An Overview and Its Applications in Pattegnition," *Smart Innov. Syst. Technol.*, vol. 195, pp. 21–30, 2021, doi: 10.1007/978-981-15-7078-0_3.

[14] P. Kim, "Matlab deep learning," *With Mach. Learn. neural networks Artif. Intell.*, vol. 130, no. 21, 2017.

[15] S. Tammina, "Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images," *Int. J. Sci. Res. Publ.*, vol. 9, no. 10, p. p9420, 2019, doi: 10.29322/ijsrp.9.10.2019.p9420.

[16] H. Gholamalinezhad and H. Khosravi, "Pooling Methods in Deep Neural Networks, a Review," no. October, 2020, [Online]. Available: http://arxiv.org/abs/2009.07485.

[17] Y. L. Boureau, J. Ponce, and Y. Lecun, "A theoretical analysis of feature pooling in visual recognition," *ICML 2010 - Proceedings, 27th Int. Conf. Mach. Learn.*, no. November, pp. 111–118, 2010.

[18] S. Palit, "Studies on Ozone-oxidation of Dye in a Bubble Column Reactor at Different pH and Different Oxidation-reduction Potential.," *Int. J. Environ. Sci. Dev.*, vol. 1554, pp. 341–346, 2010, doi: 10.7763/ijesd.2010.v1.67.

[19] M. Thoma, "Analysis and Optimization of Convolutional Neural Network Architectures," no. August, 2017, [Online]. Available: http://arxiv.org/abs/1707.09725.

[20] I. Sutskever, J. Martens, and G. Hinton, "Generating text with recurrent neural networks," *Proc. 28th Int. Conf. Mach. Learn. ICML 2011*, pp. 1017–1024, 2011.

[21] 2 Xiang Yu1, Nianyin Zeng3,∗, Shuai Liu4,∗, Yu-Dong Zhang1, "Utilization of DenseNet201 for diagnosis of breast," pp. 1–13.

[22] T. Rahman *et al.*, "Transfer learning with deep convolutional neural network (CNN) for pneumonia detection using chest X-ray," *Appl. Sci.*, vol. 10, no. 9, p. 3233, 2020.

[23] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 2261–2269, 2017, doi: 10.1109/CVPR.2017.243.

[24] P. Singh and Avinash Manure, *Learn TensorFlow 2.0*. 2020.

[25] S. H. Wang and Y. D. Zhang, "DenseNet-201-Based Deep Neural Network with Composite Learning Factor and Precomputation for Multiple Sclerosis Classification," *ACM Trans. Multimed. Comput. Commun. Appl.*, vol. 16, no. 2s, pp. 1–18, 2020, doi: 10.1145/3341095.

[26] J. Vojt, "Deep Neural Networks and Their Implementation," p. 96, 2016, [Online]. Available: http://hdl.handle.net/20.500.11956/75827.

[27] "GitHub - cair/Fire-Detection-Image-Dataset."

[28] "FIRE Dataset (kaggle.com)," p. https://www.kaggle.com/datasets/phylake1337/fire-d.

[29] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.