HIGH SCHOOL EDITION
Journal of Student Research
Volume 12 Issue 4 (2023)

The Application of CNN-Based Image Classification to Wildfire Early Detection

Yanming Huang[1] and Alfred Renaud[#]

[1]Charter School of Wilmington
[#]Advisor

ABSTRACT

Wildfires have become increasingly common and devastating in recent years, yet many regions still rely on traditional methods of using human spotters atop lookout towers to detect fires. These methods are slow and inefficient, and as fires become more frequent demand for new detection methods has grown along with it. This study aims to utilize artificial intelligence along with stationary camera systems to perform early detection of wildfires. An open-source dataset of 1900 wildfire pictures was obtained and processed, and a convolutional neural network model was trained on the dataset. The final model achieved an accuracy of 95.59% on the validation dataset after being trained for 37 epochs.

Introduction

An Introduction to Wildfires and Detection Methods

Wildfires are one of the most devastating natural disasters in the modern era, alongside hurricanes, earthquakes, floods, etc. In the past few years, wildfires have become larger, more common, and more devastating as an indirect result of the changing global climate. From 2013 to 2022, wildfires burned "an average of 7.2 million acres" annually, substantially more than "the average annual acreage burned in the 1990s (3.3 million acres)" (Hoover & Hanson, 2023). Following these trends, wildfires in the future are projected to occur more frequently and become more devastating as well.

Traditionally, wildfires were detected by fire lookouts on lookout towers scattered throughout forests. However, this method is inefficient, being prone to human errors and fatigue. Thus, the demand for new, more efficient fire detection systems has been rising rapidly. The installation of sensors in recent years has helped with the issue, but these sensors have limited ranges and only activate when the wildfire has spread and reached the sensing area (Barmpoutis et al., 2020). This not only reduces the effectiveness of sensors but also adds a measurable delay between the ignition and detection of fires. Thus, many sensors would be required to reliably detect wildfires in time, leading to significant costs in installation and maintenance.

Other approaches to fire detection have attempted to circumvent the limits of conventional sensors with the use of other monitoring techniques, ranging from satellites and aerial vehicles to stationary networks. Satellites orbit high above the Earth and can cover large regions. However, there are currently no satellites designed for rapid fire detection, and most cannot also maintain a view of the same area for extended periods. Stationary camera systems, often positioned at vantage points, offer coverage of large areas and long ranges, and require minimal maintenance (Govil et al., 2020). Normally, these camera systems are operated and monitored by human watchers and are often used to detect fires through manual observation, confirm the existence of fires reported through other methods, or track and monitor active fires. This approach improves on the traditional method by allowing a single watcher to observe multiple distant locations at once but presents similar drawbacks due to human nature. In attempts to reduce

these inefficiencies, computer programs were utilized to monitor the cameras in place of humans. In recent years, the development of artificial intelligence technologies has led to the common utilization of deep learning-based Convolutional Neural Networks in fire detection (Jindal et al., 2021). This research aims to evaluate the effectiveness of simple convolutional neural networks in detecting wildfires when combined with remote camera systems.

An Introduction to CNNs and Goals of the Research

In this research, a CNN (convolutional neural network) is utilized to classify the images into their respective categories. The Convolutional Neural Network is a popular machine-learning method for the identification and classification of images. A typical convolutional neural network includes several layers: a convolutional layer, a pooling layer, a fully connected layer, and an output layer (Figure 1) (Albawi et al. 2017).

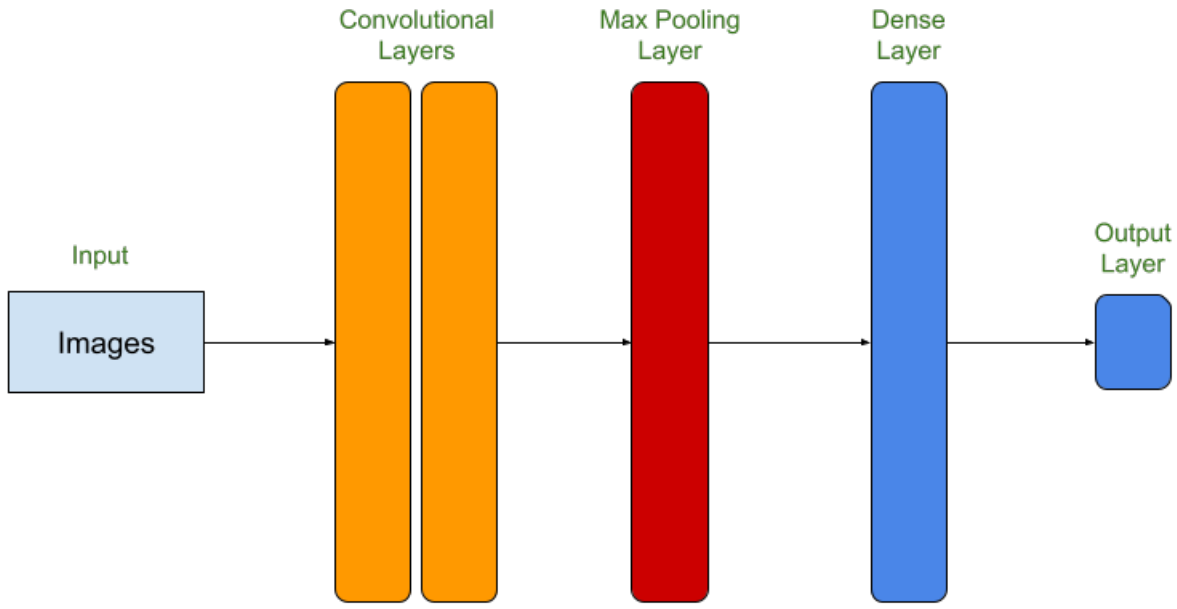

Figure 1. Illustration of a typical CNN with two convolutional layers, one pooling layer, one dense layer, and one output layer.

When an input is given to the network, it propagates through the layers sequentially and is outputted as a number or multiple numbers. The ultimate goal of the network is to find the relationship between the input data and output. If a network has more layers, it has more capability to find the relationship but is also more likely to "memorize" the dataset. The goal of this research is to create and train a model that is capable of identifying the presence of wildfires by analyzing images.

Methods

Data Collection

The data used in this research was retrieved from a Kaggle dataset (Dincer, 2021). The dataset consisted of 1900 files classified into two classes, "fire" and "nofire". Each file was an image in the .jpg format with a size of 250 by 250 pixels and pictured a natural environment either with fire burning or without. An example of the images is shown below (Figure 2): the left image is from the "fire" class, while the right is from the "nofire" class.

Figure 2. Two sample images from the Kaggle Dataset. One is from the "fire" class (left), and one is from the "nofire" class (right).

The dataset contained 950 pictures of each class, further split into "Testing" and "Training and Validation" subfolders. The "testing" subfolder contained 68 images, of which 22 were of the "fire" class and 46 were of the "nofire" class. The "Training and Validation" subfolder contained 1832 images, of which 928 were of the "fire" class and 904 were of the "nofire" class. In this research, the files from the "Training and Validation" folder were used for the training process of the model, while files from the "Testing" subfolder were later used to verify the accuracy of the model.

Data Preprocessing

Before a neural network can be trained, the data collected must first be preprocessed into a format the network can understand. For the convolutional neural network, the images needed to be resized, converted into arrays of values, labeled, and normalized. In this research, the images are first resized to 128 by 128 pixels using the *PIL* library *image.resize()* function. They are then converted into the Numpy Array format using the function *np.asarray()*. Images from the "Training and Validation" subfolder were appended to a larger Numpy array named *images*, while images from the "Testing" subfolder were appended to a Numpy Array named *images_t*. Simultaneously, the labels for the images were appended to the Numpy Arrays *labels* and *labels_t* respectively. The values of the images are then normalized for better performance of the model by dividing by 255. This reduces the range of values from integers between 0 and 255 to a decimal between 0 and 1. Finally, the *images* and *labels* arrays are fed into the *train_test_split()* function from the *Scikit Learn* library, and split into training and validation datasets named *x_train, x_val, y_train,* and *y_val* respectively. The validation datasets contain 15% of the images from the "Training and Validation" dataset. The *images_t* and *labels_t* arrays were stored for later use.

Model Creation

The artificial intelligence model used in this research was created using the *Tensorflow* Python library. The model contained a total of 12 layers: Three pairs of 2D Convolutional layers and 2D Max Pool layers, one Batch Normalization layer, two Dropout layers, one Flattening layer, and two Dense layers. The 2D Convolutional layers were identical, each with 64 filters, a kernel size of (3,3), and utilized *relu* as the activation function; The max pooling layers were identical as well, each with a pool size of (2,2); The dropout layers had a drop rate of 0.5; The batch normalization

layer followed default settings; The first dense layer had 128 nodes and activation function *relu*, while the second dense layer was the output and had 1 node and activation function *sigmoid* (Figure 3). In total, the model contains 1,681,793 parameters, of which 1,681,665 were trainable, and 128 were not.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 64)	1792
max_pooling2d (MaxPooling 2D)	(None, 63, 63, 64)	0
batch_normalization (BatchNormalization)	(None, 63, 63, 64)	256
dropout (Dropout)	(None, 63, 63, 64)	0
conv2d_2 (Conv2D)	(None, 61, 61, 64)	36928
max_pooling2d_1 (MaxPooling 2D)	(None, 30, 30, 64)	0
dropout_1 (Dropout)	(None, 30, 30, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 64)	36928
max_pooling2d_2 (MaxPooling 2D)	(None, 14, 14, 64)	0
flatten (Flatten)	(None, 12544)	0
dense (Dense)	(None, 128)	1605760
dense_1 (Dense)	(None, 1)	129

Total params: 1,681,793
Trainable params: 1,681,665
Non-trainable params: 128

Figure 3. Summary of the model, listing the layers and parameters per layer in sequence.

Results

After training, the model reached an accuracy of 97.09% with a minimum loss of 0.1184 on the validation set (Table 1). Meanwhile, it reached an accuracy of 98.97% and a loss of 0.0272 on the training set (Table 2).

Table 1. Accuracy and loss of model predictions for validation dataset.

	Accuracy	Loss
Validation Dataset	97.09%	0.1184

Table 2. Accuracy and loss of model predictions for training dataset.

	Accuracy	Loss
Training Dataset	98.97%	0.0272

The accuracy and loss of the model for both training and validation datasets are listed below (Figures 4, 5).

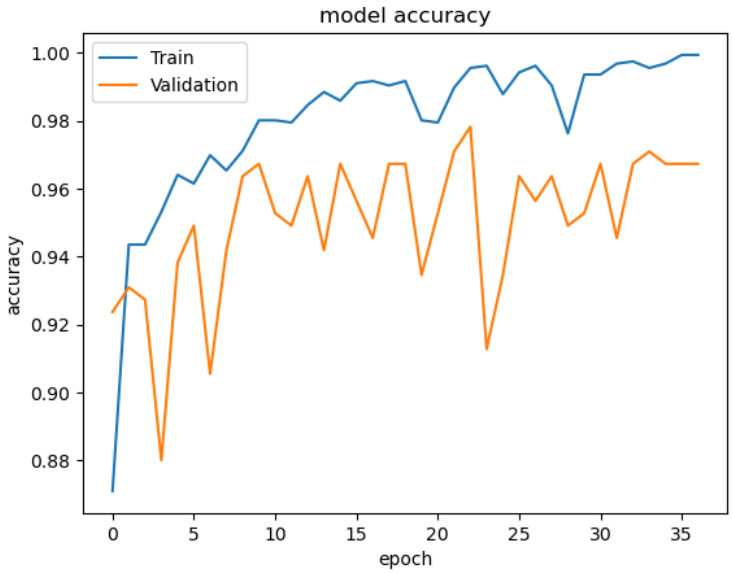

Figure 4. Accuracy of the model in training and validation datasets with respect to epochs.

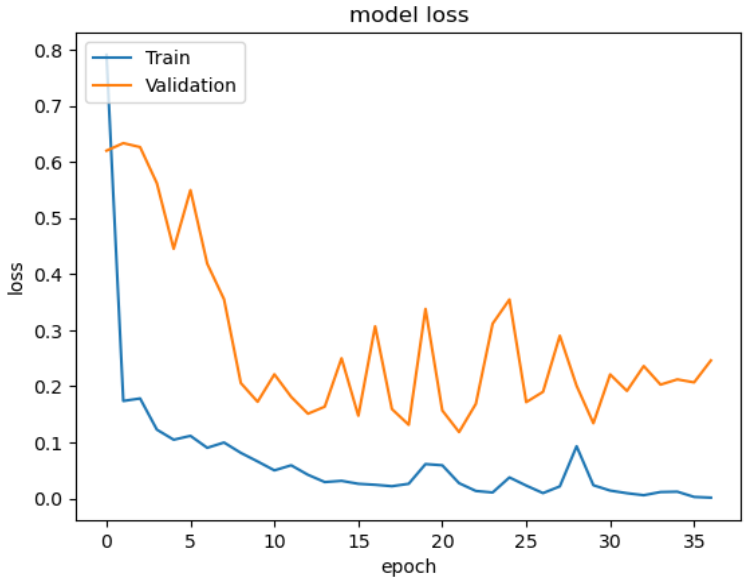

Figure 5. Loss of the model in training and validation datasets with respect to epochs.

After the training was complete, images stored in the array *images_t* were fed to the model, and predicted values were compared against the *label_t* dataset. In total, the model predicted correctly for 65 out of the 68 images (Figure 6).

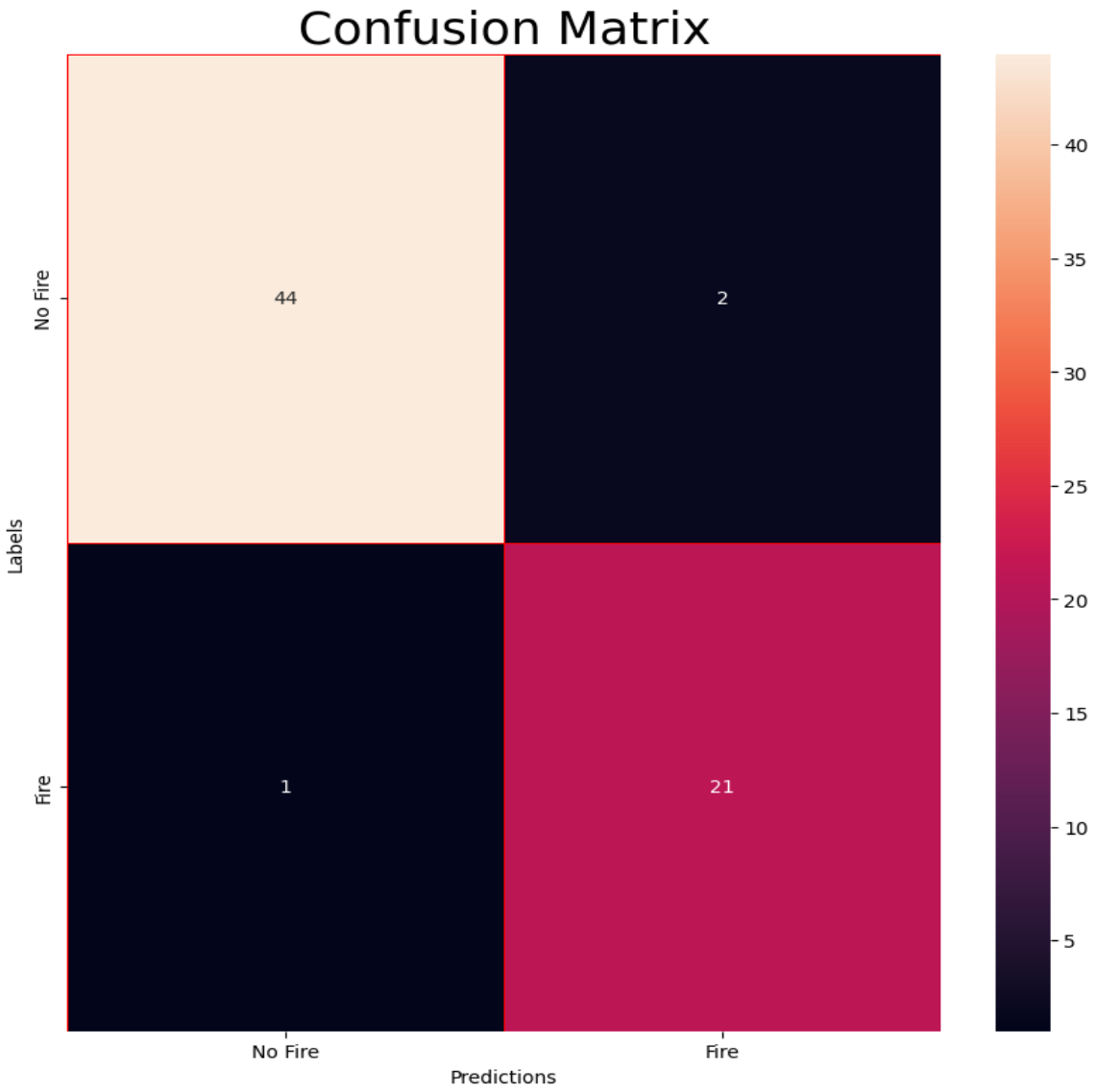

Figure 6. Confusion matrix of predictions on images from the "testing" subfolder. Grids, where labels of both axes match, indicate accurate predictions.

In total, the model took 9 minutes and 31 seconds to train, and trained for a total of 37 epochs, averaging 15 seconds and 448 milliseconds per epoch (Table 3).

Table 3. Total and average times for training the model.

	Total Time	Average Time per Epoch
Training	9min31sec	15.448sec

The model was able to complete the prediction in 233 milliseconds, averaging just 3.43 milliseconds per image (Table 4).

Table 4. Total and average times for the model's predictions of data from the "testing" subfolder.

	Total Time	Average Time per Image
Prediction	233ms	3.43ms

Discussion

The results above suggest that artificial intelligence is remarkably effective in identifying wildfires. In just 35 epochs, the model reached a peak accuracy of 99% on the training dataset and 97% on the validation dataset (Table 1). Additionally, the model reached a minimum loss of 0.0272 on the training dataset and 0.1184 on the validation dataset (Table 2). This data suggests the model was able to learn well and produced fairly accurate predictions. However, the loss of the model on the validation dataset reaches its minimum and begins increasing again at epoch 22, suggesting that the model began to overfit (Figure 5). At the same time, the validation accuracy of the model had reached its peak and remained relatively constant afterwards, suggesting diminishing returns from the model (Figure 4). These results suggest that the current model could be fine-tuned to reduce overfitting and increase accuracy. This is to be expected as the model created was quite simple, and due to time limits the model was not optimized to its full potential. When assessed with images from the "testing" subfolder, the model predicted correctly on 65 out of the 68 images, resulting in an accuracy of 95.59% (Figure 6). This is to be expected as the model was a basic convolution neural network trained on a small dataset. The simplicity of the model also led to a fast training time of just 9 minutes and 31 seconds (Table 3), and a similarly fast prediction time of 3.43 milliseconds per image (Table 4). However, this speed of the model does come as a tradeoff for lower accuracy of the predictions.

Conclusion

The model created in this research achieved relatively good accuracy in quite a short training time, suggesting that CNN is a very promising approach to the early detection of fires. The small size and simplicity of the model means it could be easily distributed and incorporated within a larger, more sophisticated system. As similar research suggests, deeper, more sophisticated CNNs could be created to provide better accuracy and more comprehensive detection (Zhang et al., 2021). It is suggested that future researchers attempt to incorporate other data into the model, such as the GPS coordinates of the location or even weather data, to provide more comprehensive detection of wildfires. It is also suggested that the model be trained on larger, more diverse datasets, as due to time and resource constraints, the dataset used was fairly limited.

Acknowledgments

My research and paper owe a lot to the skills and guidance I received from Dr. Guillermo Goldsztein, a Georgia Tech professor, who taught me how to create and conduct my project. I am also grateful to Mr. Alfred Renaud, who offered valuable advice and instruction throughout the research process. Their support was indispensable for the completion of this work, and I want to express my sincere gratitude to them for their help.

References

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In 2017 international conference on engineering and technology (ICET) (pp. 1-6). Ieee.

Barmpoutis, P., Papaioannou, P., Dimitropoulos, K., & Grammalidis, N. (2020). A Review on Early Forest Fire Detection Systems Using Optical Remote Sensing. Sensors (Basel, Switzerland), 20(22), 6442. https://doi.org/10.3390/s20226442

Dincer, B. (2021, May 18). *Wildfire Detection Image Data*, Version 1. Retrieved August 20, 2023 from https://www.kaggle.com/datasets/brsdincer/wildfire-detection-image-data.

Govil, K., Welch, M. L., Ball, J. T., & Pennypacker, C. R. (2020). Preliminary Results from a Wildfire Detection System Using Deep Learning on Remote Camera Images. Remote Sensing, 12(1), 166. MDPI AG. http://dx.doi.org/10.3390/rs12010166

Hoover, K., & Hanson, L. A. (2023, June 1). *Wildfire Statistics - CRS Reports*. Congressional Research Service. https://crsreports.congress.gov/product/pdf/IF/IF10244

Jindal, P., Gupta, H., Pachauri, N., Sharma, V., & Verma, O. P. (2021). Real-time wildfire detection via image-based deep learning algorithm. In Soft Computing: Theories and Applications: Proceedings of SoCTA 2020, Volume 2 (pp. 539-550). Springer Singapore.

Zhang, J., Zhu, H., Wang, P., & Ling, X. (2021). ATT squeeze U-Net: A lightweight network for forest fire detection and recognition. IEEE Access, 9, 10858-10870.