Proceedings of the 5th International Conference on Computing and Data Science
DOI: 10.54254/2755-2721/18/20230978

AutoDL-based convolutional neural networks for wildfire detection

Ching-Syuan Chien

Knowledge-First Empowerment Academy, Missouri City, Texas, 77489, United States

leoli@xischina.com.cn

Abstract. Wildfires have emerged as a pressing issue in many regions of the world due to the ongoing impact of global warming on the planet. However, a reliable and high-performance detection system is currently lacking. This study strives to introduce a wildfire detection system that is based on neural networks and image recognition. This study utilized Edge Impulse to train a neural network to identify wildfire occurrences in the given pictures. To optimize the performance and adaptability of the model, an extensive dataset was compiled by collating images from two Kaggle projects, resulting in a final dataset of over 3000 images. The core technological advancements that Edge Impulse is based on are Auto Deep Learning (AutoDL) and Convolutional Neural Networks (CNN). By applying technologies like Neural Architecture Search (NAS), hyperparameters optimization, and transfer learning, AutoDL enables people interested in machine learning to approach the technology without an extensive understanding of math or programming that machine learning was built upon. CNN is a highly effective and efficient form of neural network popular for image classification. It consists of three different layers: the convolutional layer, the pooling layer, and the fully connected layer. The result of this study consists of a fully functional model for wildfire detection that is ready to be deployed, with a final testing accuracy of over 99%.

Keywords: image recognition, wildfires, convolutional neural networks, Edge Impulse.

1. Introduction

The detection of wildfires reliable manner has been a challenging but pivotal issue. Over the past few decades, wildfires have been an increasing concern, with their prevalence and severity surging in tandem with global warming. Vegetation around the globe becomes more vulnerable to ignition as the temperature rises, and the dry environment leads to the faster spreading of wildfire. With urbanization advances through wildlands, a direct correlation is formed between wildfires and human injuries. 2020 witnessed some of the most destructive and massive wildfires in recent history, particularly in California. The 2020 California wildfires burned 1,741,920 hectares of land, destroyed 11,116 structures, and took away 33 lives [1]. Around the same period, Australia also suffered a wildfire that is on a similar scale, unfortunately causing 34 direct deaths and 445 indirect deaths [2, 3]. These massive wildfires have been a wake-up call for all organizations worldwide; there is a need to deescalate global warming as a long-term goal and implement a dependable system that detects wildfires as a short-term goal.

Current popular methods for wildfire detection include ground-based monitoring, aerial surveillance, and satellite imagery [4-6]. Ground-based monitoring relies on weather stations and sensors to detect

DOI: 10.54254/2755-2721/18/20230978

sudden environmental changes, including temperature, humidity, wind speed, etc. Ground-based monitoring is heavily limited by its range of detection, as it is only assigned to specific areas and often cannot detect wildfires appearing in remote or hard-to-access spaces. Ground-based monitoring also often delays detection, as it relies on the wildfire to spread to areas containing sensors, which could have already done heavy damage. The other option for wildfire detection is aerial surveillance. Aerial surveillance uses airplanes and helicopters with cameras to monitor wildfires. The most significant downsides of aerial surveillance are its need for human operation and costs, as there is a need for a human to pilot the airplane or the helicopter, and the price of maintaining constant air patrols is high. Ground-based monitoring and aerial surveillance strongly depend on the environment's weather; if extreme weather conditions occur in the monitored area, the effectiveness of both methods decreases significantly. The most prominent option of the three is satellite imagery, which utilizes satellites with sensors to detect heat signatures or appearances of smoke. However, satellite imagery has its own set of downsides. The delay in the transmission time between the data collected and the data analysed by ground stations can hinder the detection of real-time wildfire occurrences. Using satellite imagery to capture high-resolution images also comes at a high cost, especially for regions with limited resources.

With a clear need for inventing and implementing a new detection method, using image recognition to detect wildfires automatically could be the new method needed. Training a Convolutional Neural Network (CNN) for image recognition is the foremost approach for identifying objects and patterns of images automatically. CNNs are a branch of deep neural networks that took inspiration from the visual biological features of animals, as they are heavily influenced by how visual cortexes work in animals [7]. Auto Machine Learning (AutoML) is the process of automating configuration, data processing, and optimization for machine learning models [8]. The existence of AutoML allows more accessibility to individuals without proficient technical understanding by simplifying model selection and manual tuning. AutoML algorithms can automate selecting and constructing the best features, pre-processing and cleaning the data, and optimizing the model's hyperparameters for the best possible performance. AutoML can significantly reduce the time and effort needed to develop an effective machine-learning model.

In order to achieve the goal of detecting wildfire with satisfactory performances, this research built a deployable model by utilizing the AutoML platform called edge impulse that was developed based on the Convolutional Neural Network architecture. With the use of edge impulse, it is possible to train a highly efficient model for wildfire detection with a high correction rate and relatively low resource consumption. This study was carried out to show the possibility of designing a successful wildfire detection model by using a trained neural network for image recognition while having limited resources and a small data set.

2. Methods

2.1. Data set preparation

The data set utilized in this study to train the proposed model was acquired through two different Kaggle projects, namely the wildfire image data and the forest fire dataset [9, 10]. Both data sets' training section contains two categories, the label fire and the label nofire. As the label name suggests, the fire category includes images of various burning landscapes. The nofire category contains pictures that consist of typical landscapes. In addition, images of both categories and data sets have a consistent resolution of 250 by 250 pixels.

The wildfire image data set has a training category with 1780 images equally distributed between the fire and nofire categories. The data set's testing procedure includes the same two categories as the training section, with 40 images while maintaining the 50% split. The forest fire data set has a training category containing 1520 images, with the same 50% split between the two categories. The forest fire data set has a larger testing size, containing 380 images with the same 50% split between the two categories.

DOI: 10.54254/2755-2721/18/20230978

The inclusion of two data sets was crucial to obtain a sufficiently large number of images to train the model. A more extensive training dataset allows the model to adapt better to complex scenarios. It was observed that setting the images to RGB yielded better accuracy rates, as the bright colors of fire generated a strong contrast against typical landscape sets, so the images within both data sets were determined to be set to RGB.

2.2. AutoDL

Auto Deep Learning (AutoDL) is a subfield of AutoML that utilizes machine learning to automate deep learning tasks. AutoDL enables the automation of neural network training by employing multiple technological advancements, including neural architecture search (NAS), hyperparameters optimization, and transfer learning [11]. Auto Deep Learning's fundamental philosophy is to allow more accessibility for non-experts to leverage deep learning, as AutoDL simplifies and automates the most challenging and time-consuming configurations.

One core technology that AutoDL built upon was NAS algorithms. NAS is developed to search for high-performing deep learning models for given tasks automatically. NAS algorithms use different optimization techniques, such as reinforcement learning, genetic algorithms, and other optimization techniques, to examine vast arrays of neural network architectures to select the most optimized one suited for the given task [11].

Hyperparameter optimization is another essential technology that AutoDL uses. Hyperparameters are parameters used to define the structure and methodology of a neural network. Hyperparameters contain necessary configurations like learning rate, batch size, and regularization. AutoDL also makes use of transfer learning, a technique that reuses pre-trained models for new tasks. By applying pre-trained models, AutoDL can significantly minimize the time and computational power required for further model training and achieve the best performance on relatively small datasets [11].

2.3. Introduction of CNN model

CNN is a form of neural network popular for image classification and pattern recognition tasks [12-14], as it is highly effective and efficient. CNN's architecture consists of three layers, the convolutional layer, the pooling layer, and the fully connected layer.

The convolutional layer is the core building block of CNN. CNN uses a filter or kernel, a 2-D array of weights that scans the receptive fields of the image to detect features. As it moves across the image, the filter performs a convolution, which calculates a dot product between the filter and the input pixels, resulting in an output array. This process repeats until the entire image has been convolved, producing a feature map. After each convolution operation, CNN applies Rectified Linear Unit (ReLU) to introduce nonlinearity to the model [12].

The pooling layer is the second layer of a CNN, which is responsible for reducing images' dimensions, thereby reducing needed parameters in the input. It does this by deploying a filter across the entire input, like the convolutional layer. However, the input data does not have any weights; instead, the kernel applies a function to the values of the receptive field, populating the output array. There are two primary pooling types: max pooling and average pooling. Compared to average pooling, max pooling is more commonly used [12].

The last layer, the fully connected layer, marks the last crucial part of CNN's performance. CNN finalizes the classification tasks and brings a definitive resolution. CNN assembles the flattened output from the previous layers in a small package. It then applies a set of nodes connected to previous layers to generate the final output. Fully connected layers use a softmax activation function rather than a ReLu function, producing the result of possibilities ranging from 0 to 1 [12].

2.4. Procedure of using Edge Impulse

Regarding the specific procedures, the initial phase in training a neural network with Edge Impulse necessitates the uploading of the gathered dataset, which will serve as the training data for the neural network, tailored to the specified task. The interface of uploading data is shown in Figure 1.

DOI: 10.54254/2755-2721/18/20230978

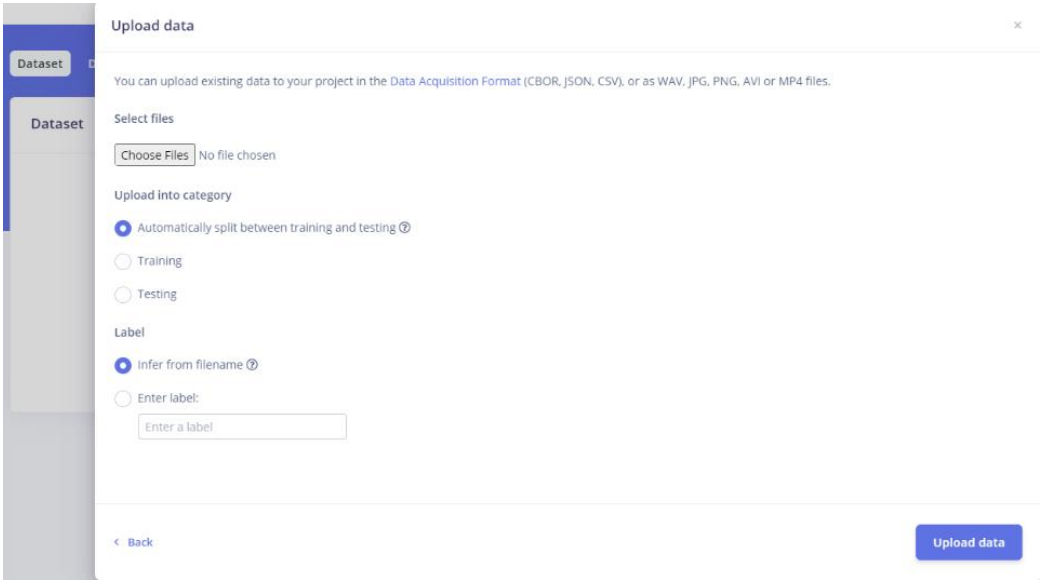

Figure 1. The interface of uploading data.

The second step is to design the neural network architecture. In this study, the images are first resized to 96×96 pixels. After that, it was passed into the image processing block, which pre-processes and normalizes the dataset while reducing the colour depth and using the transfer learning block. The interface of impulse design is shown in Figure 2.

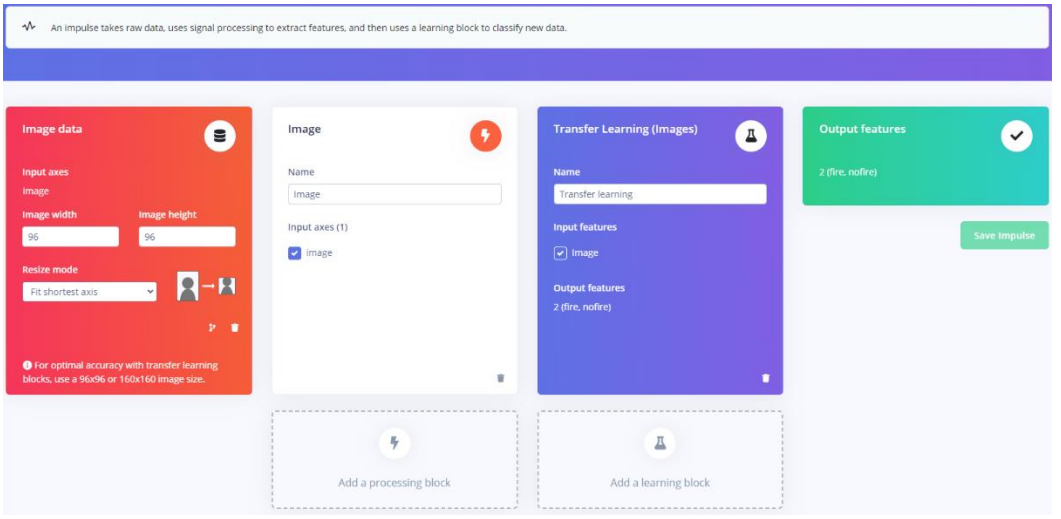

Figure 2. The interface of impulse design.

The third step is configuring the parameters, mainly whether the images will be set as grayscale or coloured. The interface of setting image parameters is shown in Figure 3.

DOI: 10.54254/2755-2721/18/20230978

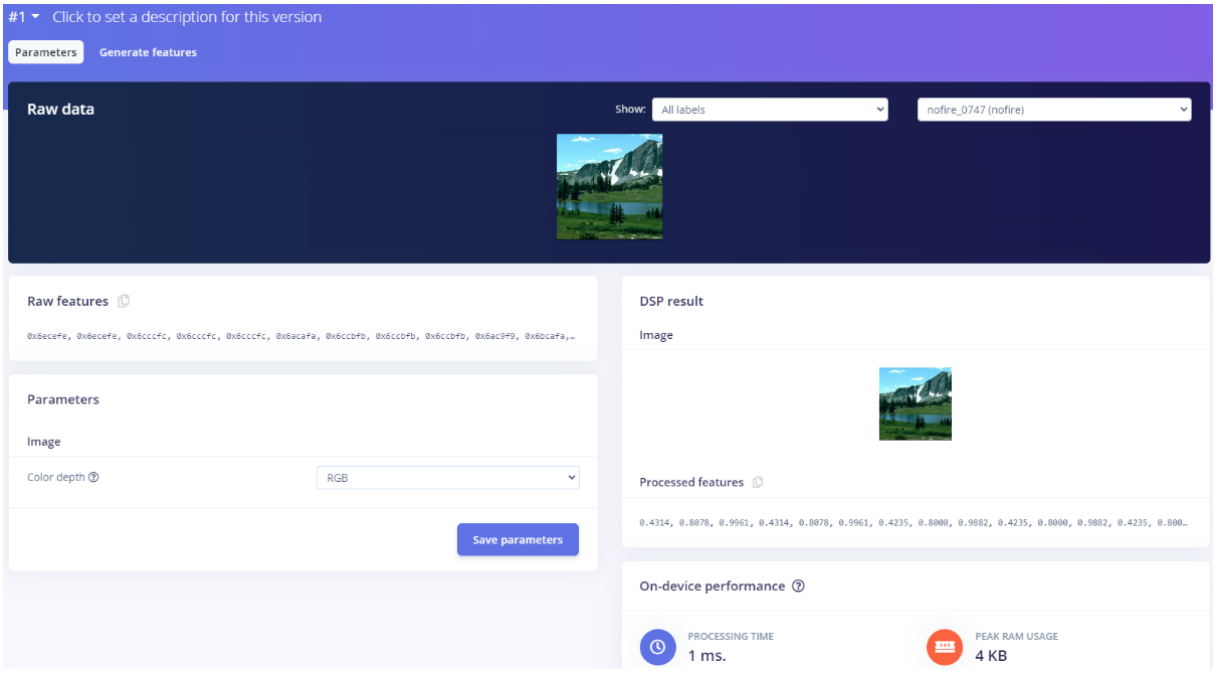

Figure 3. The interface of setting image parameters.

The fourth step is to generate features for all the images within the data set. The interface of generating features is shown in Figure 4.

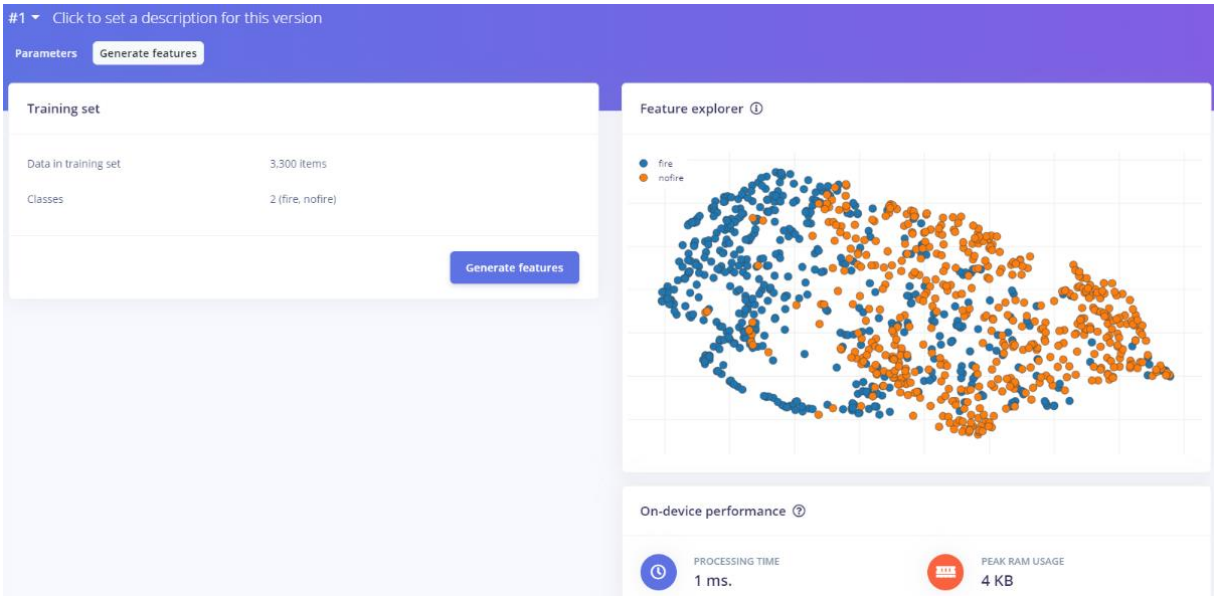

Figure 4. The interface of generating features.

The last step is configuring the exact parameters of the transfer learning method, with parameters like epochs, learning rate, and validation set size for modification. There are also extra options like data augmentation, auto-balance data set, and using the profile int8 model to change how the model functions. There is also an option on the model on which the transfer learning would be based on. The interface of setting training hyperparameters is shown in Figure 5.

DOI: 10.54254/2755-2721/18/20230978

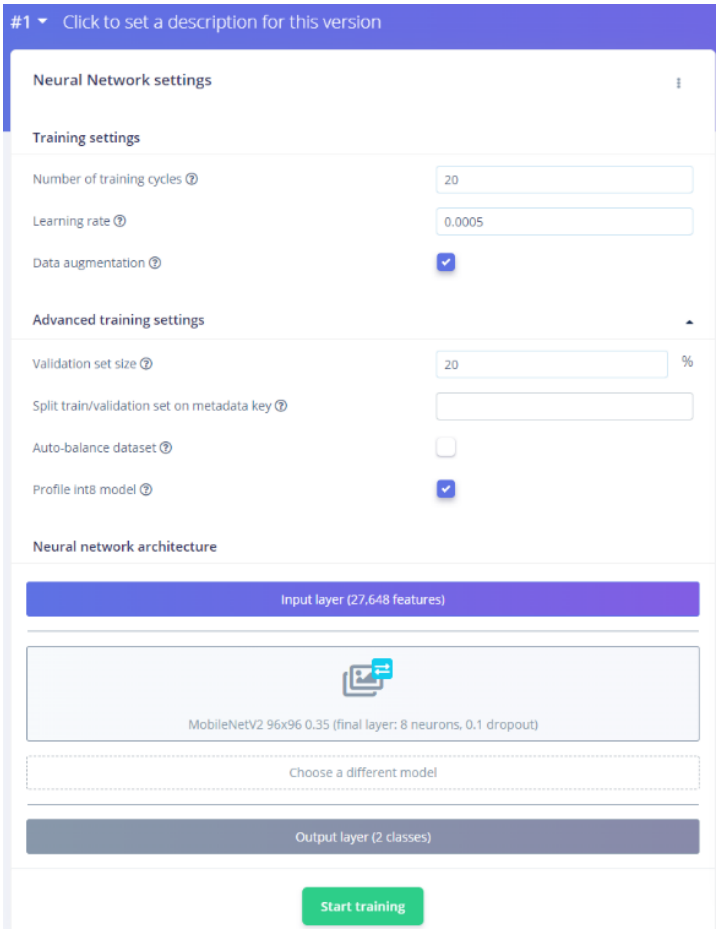

Figure 5. The interface of setting training hyperparameters.

2.5. Implementation details

The hyperparameter settings in this study include epochs, learning rate, data augmentation, validation set size, profile int8 model, and choosing the models the neural network trains on. The epochs were set to 57 during training to achieve satisfactory performance on this task. The learning rate was based on the default setting, which is 0.0005. The validation set size was set to 20%, which is unchanged due to the model performing well with the default settings. The two critical changes done to the parameters were toggling on data augmentation, which allows the model to run more training cycles without overfitting, and the profile int8 model, which profiles the quantized model for an alternative model version for reference. The last decision made during the training of this project's neural network was the transfer learning model it was based on. Due to the previous decision to resize the images to 96×96 pixels, it is best suited for the model pre-trained with 96×96 images. The chosen model for transfer learning utilizes the most resources out of all the possible options, which still remain in reasonable resource consumption under the consideration of real-world applications.

3. Results and discussion

Table 1 presents the training and on-device performance of two different possible models for deployment, the quantized (int8) model and the unoptimized (float32) model. The int8 model was able to achieve 98.0% accuracy, while the float32 model was able to reach 99.4%. The on-device performances of the int8 model are 149 ms of inferencing time, 333.8K of peak RAM usage, and 569.7K of flash memory usage; the on-device performance of the float32 model are 242 ms of inferencing time, 333.8K of peak RAM usage, and 947.8K of flash memory usage.

DOI: 10.54254/2755-2721/18/20230978

Table 1. Training performance and on-device performance.

Model	Training Performance		On-device performance		
	Accuracy	Loss	Inferencing Time	Peak RAM usage	Flash memory usage
Quantized (int8) model	98.0%	0.06	149 ms.	333.8K	569.7K
Unoptimized (float32) model	99.4%	0.02	242 ms.	947.8K	1.6M

Based on Table 1, the table showcases the advantages and drawbacks of the two different models, namely int8 and float32. The int8 model appears to be better suited for most scenarios, given its significantly lower resource consumption, shorter inferencing time, and high accuracy rate of 98%. Although the float32 model achieved high accuracy and less loss, it consumes far more resources and requires longer inferencing time. Almost all on-device performance metrics of the float32 model are doubled or even tripled compared to the int8 model. The more resources required by the float32 model means the hardware of the surveillance devices must be more robust, which would increase the cost of implementing such systems. It is only in specific circumstances, like an area with a high likelihood of wildfire occurrence, which would benefit from the high accuracy, should the float32 model be considered for implementation.

Table 2. Confusion matrix of the quantized (int8) model.

	FIRE	NOFIRE
FIRE	98.2%	1.8%
NOFIRE	2.2%	97.8%
F1 Score	0.98	0.98

Table 2 presents the confusion matrix of the int8 model, including false positive rate and rate of inability to correctly detect wildfires.

Table 3. Confusion matrix of the unoptimized (float32) model.

	FIRE	NOFIRE
FIRE	99.7%	0.3%
NOFIRE	0.9%	99.1%
F1 Score	0.99	0.99

Table 3 presents the confusion matrix of the float32 model, including the percentage of false detecting rate and the percentage of the model's inability to detect wildfires.

Table 2 and Table 3 present the expected result with consideration of Table 1's performances. The float32 model had a higher correction rate in all categories, which is consistent with the result of Table 1. Both the confusion matrices of Table 2 and Table 3 showcase the same tendencies, with the percentage of a false positive rate higher than the rate of the model unable to detect the wildfire. The higher false positive rate could possibly be caused by landscapes with no wildfires consisting of smoke or fog. The landscapes with wildfires all contain a large amount of smoke due to the burning of the vegetation, which could mislead the model to believe that smoke and fog are a crucial part of wildfire detection. However, even if the model tends to categorize the result as fire through smoke and fog, probability remains relatively low, and the landscape must consist of more than average smoke or fog. In locations with significant smoke or fog concentration, consideration of the float32 model might prove to be superior, as it reduces the chance of the false positive rate by a margin of around 1.3%.

DOI: 10.54254/2755-2721/18/20230978

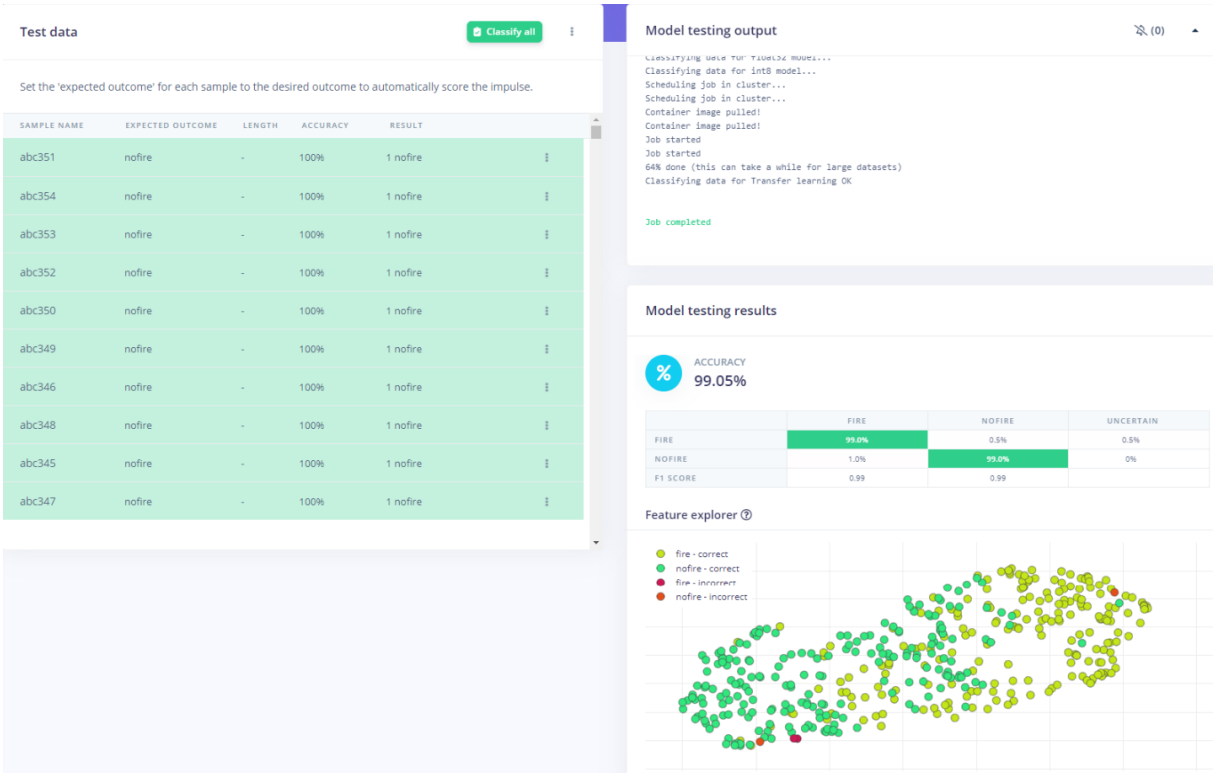

Figure 6. The testing results of the model.

Figure 6 presents the final result of the model's performance for detecting wildfires using the testing dataset. The overall accuracy achieved about 99%, with the chance of false positive rate around 1%, the chance of failing to detect around 0.5%, and an uncertainty rate of 0.5%.

The final testing outcome of the model demonstrated the neural network this study trained to be highly successful, maintaining the detection accuracy above 99%. In the few instances the trained model of this study struggled, the neural network had difficulties differentiating between the sun with fire and the smoke or fog with the smoke produced in the wildfire pictures. The challenges in determining between the sun and fire only seem to occur during landscapes with sunsets, as the scene creates an effect of red or orange objects above the landscape, leading to misidentification. One possible solution to this issue could be implementing another neural network model trained for categorizing sunset and wildfire pictures to filter out the images before sending the data set to the current model. This method can almost completely resolve the issue of misidentifying sunsets with wildfires, given that the trained model for categorization can perform well enough.

4. Conclusion

This study strives to demonstrate that it is possible to construct a successful wildfire detection model based on neural networks and the upsides of implementing such a system compared to traditional methods. With relatively small data sets and resources, this study used the Edge Impulse platform to train a neural network for wildfire detection, which is deployable. This study combined two data sets from two Kaggle projects to have a more extensive data set for training the neural network, which would improve the model's adaptability. By utilizing Edge Impulse, the neural network was trained multiple times with different configurations, with the final result showcasing the most optimized version. The result showcased in this study proves that neural networks, as a system for wildfire detection, should be seriously considered for worldwide implementations. A promising avenue for future research could be

DOI: 10.54254/2755-2721/18/20230978

to specialize the model for specific terrains by training it on corresponding data sets to improve accuracy further.

References

[1] California Department of Forestry and Fire Protection 2020 2020 Incident Archive https://www.fire.ca.gov/incidents/2020/
[2] USGS 2020 Australia's historic fires, the Black Summer fires, raged from September 2019 into February of 2020, and weren't fully extinguished until this past March. https://www.usgs.gov/news/geoscience-australias-oliver-discusses-use-landsat-during-countrys-historic-fires?qt-news_science_products=1#qt-news_science_products
[3] ABC News 2020 Bushfire royal commission hears that Black Summer smoke killed nearly 450 people https://www.abc.net.au/news/2020-05-26/bushfire-royal-commission-hearings-smoke-killed-445-people/12286094
[4] Kharchenko V S et al 2012 Monitoring network-based infrastructure for forest fire detection WIT Transactions on Ecology and the Environment 158 91-99
[5] Skyrora 2022 Using satellites to monitor wildfires https://www.skyrora.com/using-satellites-to-monitor-wildfires/
[6] Alkhatib Ahmad AA 2014 A review on forest fire detection techniques International Journal of Distributed Sensor Networks 10.3 597368
[7] Lindsay G W 2021 Convolutional neural networks as a model of the visual system: Past, present, and future Journal of cognitive neuroscience 33.10 2017-2031
[8] Hutter F Kotthoff L Vanschoren J 2019 Automated machine learning: methods, systems, challenges Springer Nature
[9] Kaggle 2021 Wildfire Detection Image Data https://www.kaggle.com/datasets/brsdincer/wildfire-detection-image-data?resource=download
[10] Kaggle 2022 Forest Fire Dataset https://www.kaggle.com/datasets/alik05/forest-fire-dataset
[11] Dong X Y et al. 2021 Automated deep learning: Neural architecture search is not the end arXiv preprint arXiv:2112.09245
[12] O'Shea K Nash R 2015 An introduction to convolutional neural networks arXiv preprint arXiv:1511.08458
[13] Yu Q Wang J Jin Z et al 2022 Pose-guided matching based on deep learning for assessing quality of action on rehabilitation training Biomedical Signal Processing and Control 72: 103323
[14] Yang Z Huang Y Jiang Y et al 2018 Clinical assistant diagnosis for electronic medical record based on convolutional neural network Scientific reports 8(1): 6329