PROJECT PRESENTATION

TOPIC:FIRE CLASSIFICATION USING CNN

PRESENTED BY
ALEENA ROY
02
SEM 1
MTECH

OUTLINE

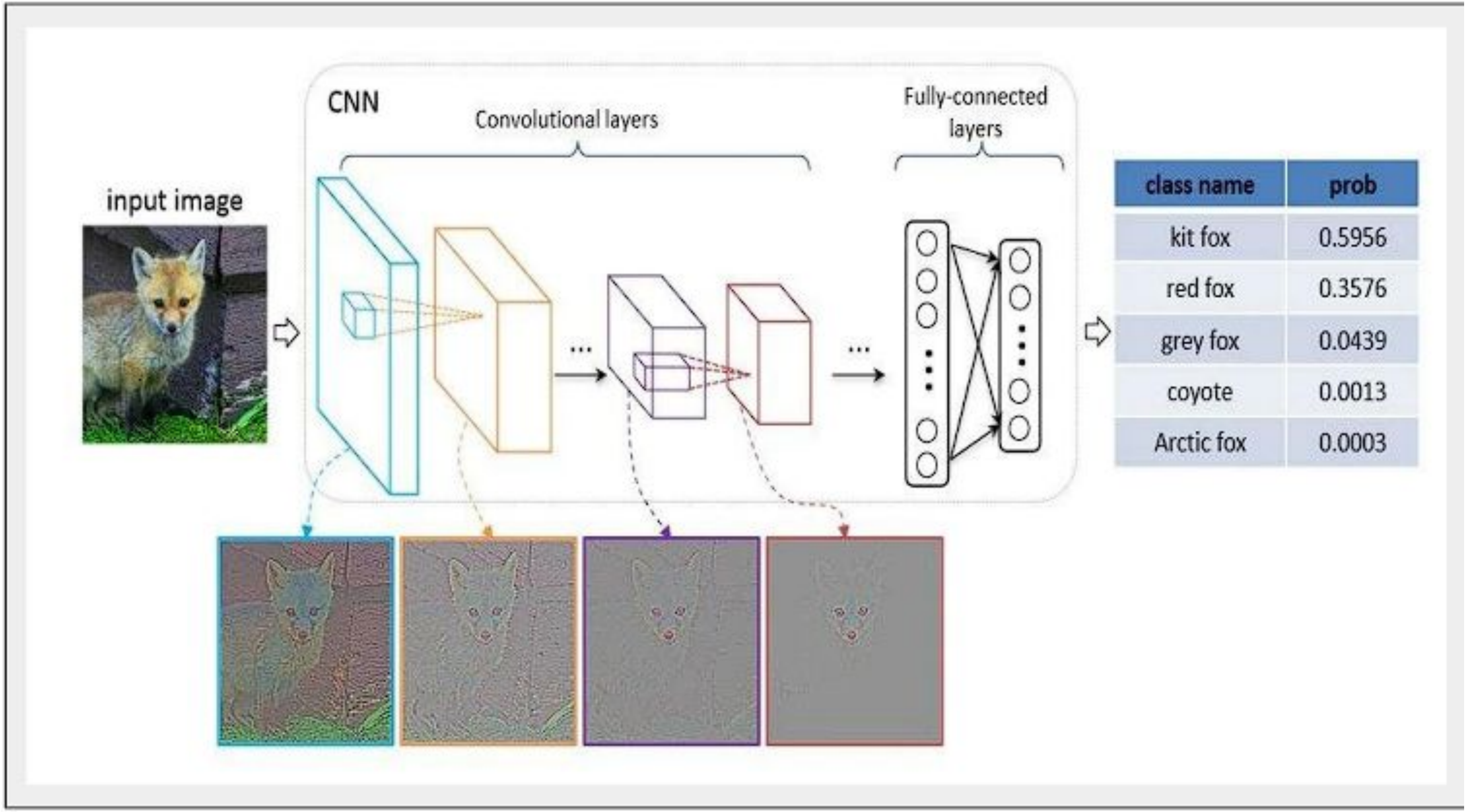
- Introduction
- Motivation
- Models Compared
- Methodology
- Dataset
- Model Architecture
- Performance Evaluation
- Conclusion
- References

INTRODUCTION

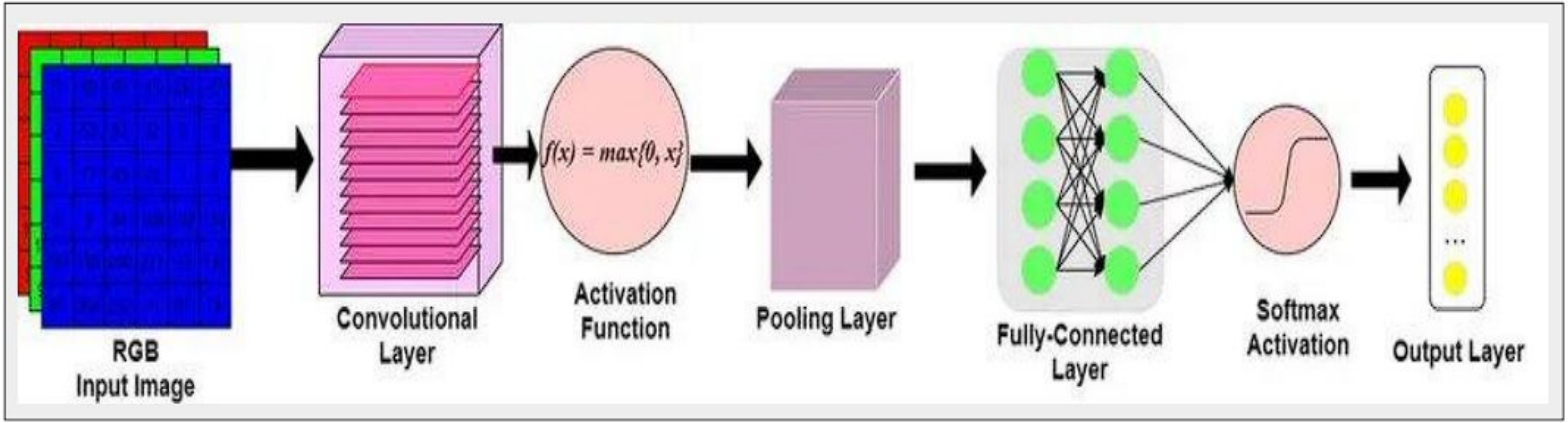
- CNNs are deep learning neural networks widely used for visual data tasks.
- Designed to automatically learn patterns and features in data, inspired by human vision.
- They excel in tasks like image classification, object detection, and video recognition.
- CNNs are applied in surveillance systems to detect fires.
- They continuously analyze visual camera feeds for fire-related patterns.

What is CNN?

- CNN stands for Convolutional Neural Network.
- It is a class of deep learning neural networks primarily used for tasks involving visual data, such as image and video recognition, image classification, object detection, and more.
- CNNs are particularly effective in handling grid-like data, making them well-suited for tasks where the spatial relationships between data points are important.
- CNNs are designed to automatically and adaptively learn patterns and features from input data



Basic Concept Of How A CNN Works



CNN Architecture

MOTIVATION

- Fire is one of the worst disasters for human life. Fire can happen anywhere and the leading cause can be natural or man.
- Over the last century, scientists have invented sensor-based methods to minimize damage and provide early warning of fires. However, these applications are only applied in a limited space and distance.
- This leads to the introduction of a vision-based method using convolutional neural network.

Models Compared

- **SqueezeNet**: A lightweight model optimized for high accuracy with fewer parameters.
- **MobileNetV2**: A mobile-optimized architecture designed for resource-constrained environments.

Metrics	**SqueezeNet**	**MobileNetV2**
Model Size	1.25 MB	14 MB
Accuracy on Dataset	97 %	94 %
Inference Speed	Faster	Moderate
Suitability for Edge Devices	Excellent	Good

Advantages of SqueezeNet

- **Compactness**: Extremely lightweight with fewer parameters, ideal for deployment in low-resource settings.
- **Accuracy**: Maintains high classification accuracy despite its small size.
- **Faster Inference**: Outperforms larger models in real-time scenarios like fire detection.

Why SqueezeNet is Best for Fire Detection

- High accuracy makes it reliable for critical tasks like identifying fire.
- Lightweight nature ensures quick inference, vital for real-time safety applications.
- Requires less computational power, making it ideal for edge devices like drones or IoT systems.

METHODOLOGY

1) **Model Selection:** For the image classification task, I chose ***SqueezeNet***, a lightweight Convolutional Neural Network (CNN) that achieves high accuracy while having a small model size. SqueezeNet is ideal for deployment on devices with limited computational resources, making it suitable for real-time fire detection applications. It provides an excellent trade-off between model size and performance, and its architecture uses fire modules to reduce the number of parameters without compromising performance.

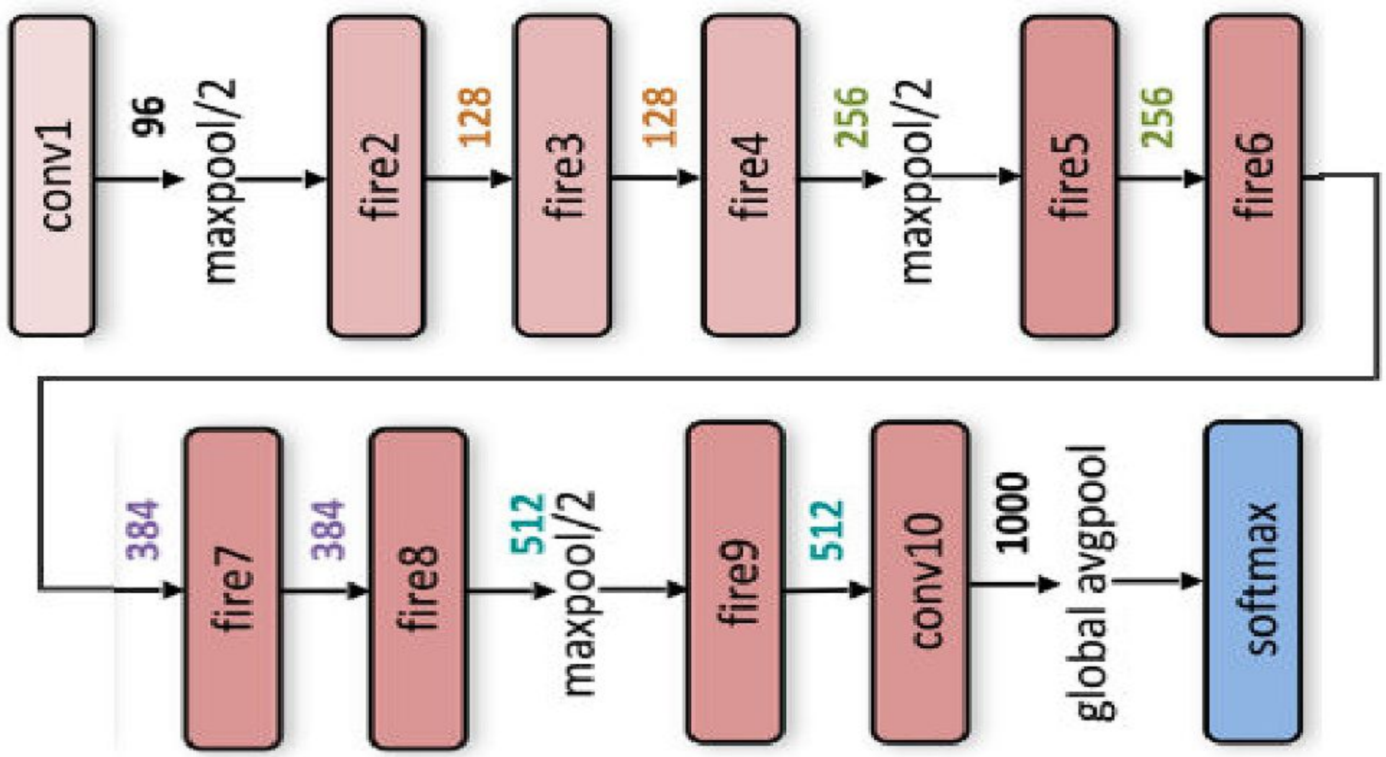
2) **Training Process:**

- **Dataset Split**: I used an 80-20 split, where 80% of the data was used for training the model, and 20% was reserved for validation.For better generalization, data augmentation techniques were also applied.
- **Model Training**:The model was trained using a *binary classification* approach, with two classes: "fire" (0) and "non-fire" (1).The model used a *binary cross-entropy loss function*, which is suitable for binary classification tasks.
- **Validation**: After training, the model was evaluated on a separate *test dataset* to measure its generalization ability. The metrics included *accuracy, precision, recall,F1-score and Confusion Matrix* which are commonly used for classification tasks to assess model performance..

3) Thresholding for Prediction:

- After training, I used a *thresholding technique* to improve prediction accuracy. For each image, the model outputs probabilities for each class (fire or non-fire). Instead of relying on the raw output probabilities, I set a *threshold value* of 0.6. This means:
  - If the probability for the "non-fire" class (class 1) is greater than 0.6, the image is classified as "non-fire" (class 1).
  - If the probability for the "fire" class (class 0) is greater than 0.6, the image is classified as "fire" (class 0).
- This threshold helps in fine-tuning the sensitivity of the model to make the final predictions more reliable.

MODEL ARCHITECTURE



To implement Squeeze Net with TensorFlow we define the fire module first. Each fire module consists of a squeeze convolution layer (1x) filterss, followed by an expand layer that has a mix of 1x1 and 3x3 convolution filters. The squeeze and expand layers introduce a mechanism for channel-wise feature recalibration, contributing to Squeeze Net's efficiency

The Squeeze Net model is built by stacking these fire modules with max-pooling layers for downsampling at specific intervals, and a final convolution and average pooling layer for classification .The model in then compiled with the Adam optimizer categorical cross entropy.

Fire Modules:

- The core building block of SqueezeNet is the **Fire Module**.
- It consists of two key layers:
  - **Squeeze Layer**: A 1x1 convolution layer that reduces the number of input channels, effectively acting as a bottleneck.
  - **Expand Layer**: A combination of 1x1 and 3x3 convolution layers, which expand the squeezed output to a higher-dimensional space.
- The outputs of the 1x1 and 3x3 convolutions in the expand layer are concatenated along the channel dimension.

Convolutional Layers:

- The network begins with an initial convolutional layer (Conv1) with a large kernel (7x7) and a stride of 2 for down-sampling.
- After Conv1, the feature maps are reduced using a max-pooling layer.

Global Average Pooling:

- Instead of fully connected layers, SqueezeNet uses global average pooling (GAP) in the final layer. This significantly reduces the parameter count by averaging each feature map across its spatial dimensions.
- The output is directly connected to the softmax activation for classification.

Reduction in Parameters:

- SqueezeNet achieves a drastic reduction in parameters (50x fewer than AlexNet) by:
  - Replacing fully connected layers with GAP.
  - Using 1x1 convolutions extensively in the squeeze layers.
  - Delaying down-sampling to maintain a larger activation map for most of the network.

Down-sampling:

- Down-sampling is strategically applied via max-pooling layers after Conv1 and certain Fire modules to preserve important spatial information for as long as possible.

Summary of Architecture

1. Conv1: 96 filters, 7x7 kernel, stride 2, followed by MaxPooling (3x3, stride 2).
2. Fire Modules: A series of Fire modules with varying squeeze (1x1) and expand (1x1 and 3x3) filters.
3. MaxPooling: Applied after specific Fire modules to reduce spatial dimensions.
4. Conv10: A 1x1 convolutional layer with the number of filters equal to the number of classes.
5. Global Average Pooling: Reduces the output to class probabilities.
6. Softmax: Final activation for classification.

Performance Evaluation

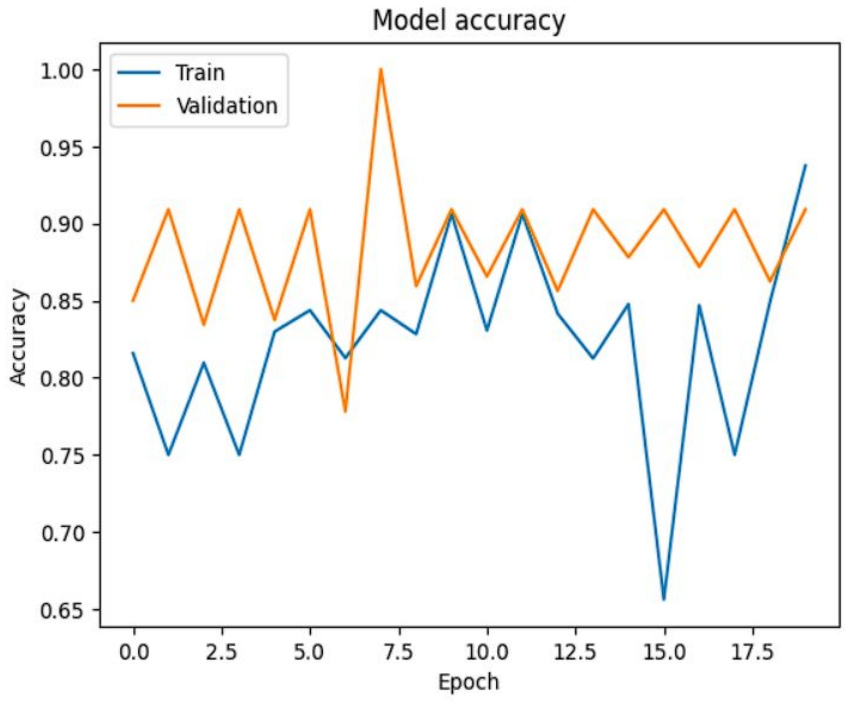
Model: "sequential_4"

Layer (type)	Output Shape	Param #
conv2d_135 (Conv2D)	(None, 224, 224, 32)	896
max_pooling2d_15 (MaxPooling2D)	(None, 112, 112, 32)	0
conv2d_136 (Conv2D)	(None, 112, 112, 64)	18,496
max_pooling2d_16 (MaxPooling2D)	(None, 56, 56, 64)	0
conv2d_137 (Conv2D)	(None, 56, 56, 128)	73,856
max_pooling2d_17 (MaxPooling2D)	(None, 28, 28, 128)	0
flatten_1 (Flatten)	(None, 100352)	0
dense_5 (Dense)	(None, 128)	12,845,184
dense_6 (Dense)	(None, 2)	258

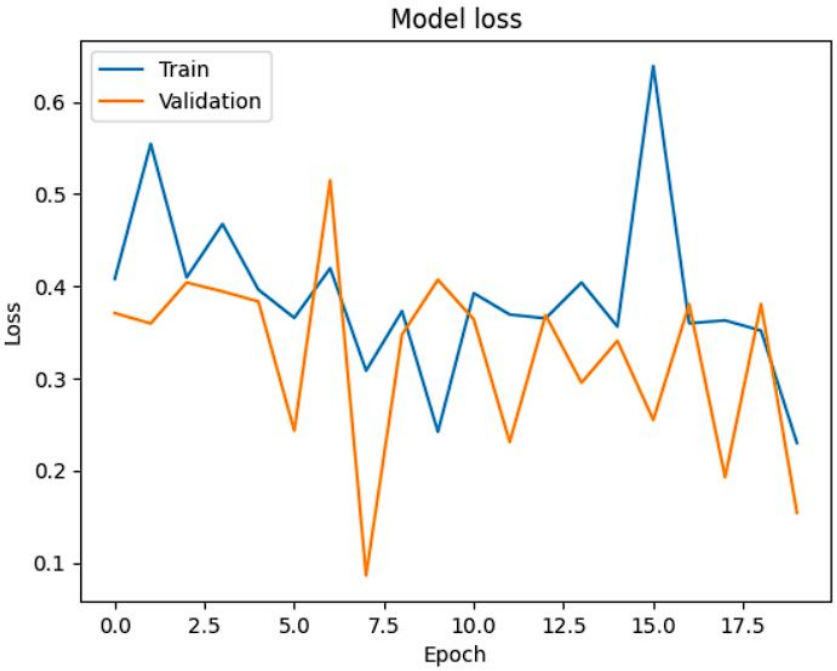
Total params: 12,938,690 (49.36 MB)

Trainable params: 12,938,690 (49.36 MB)

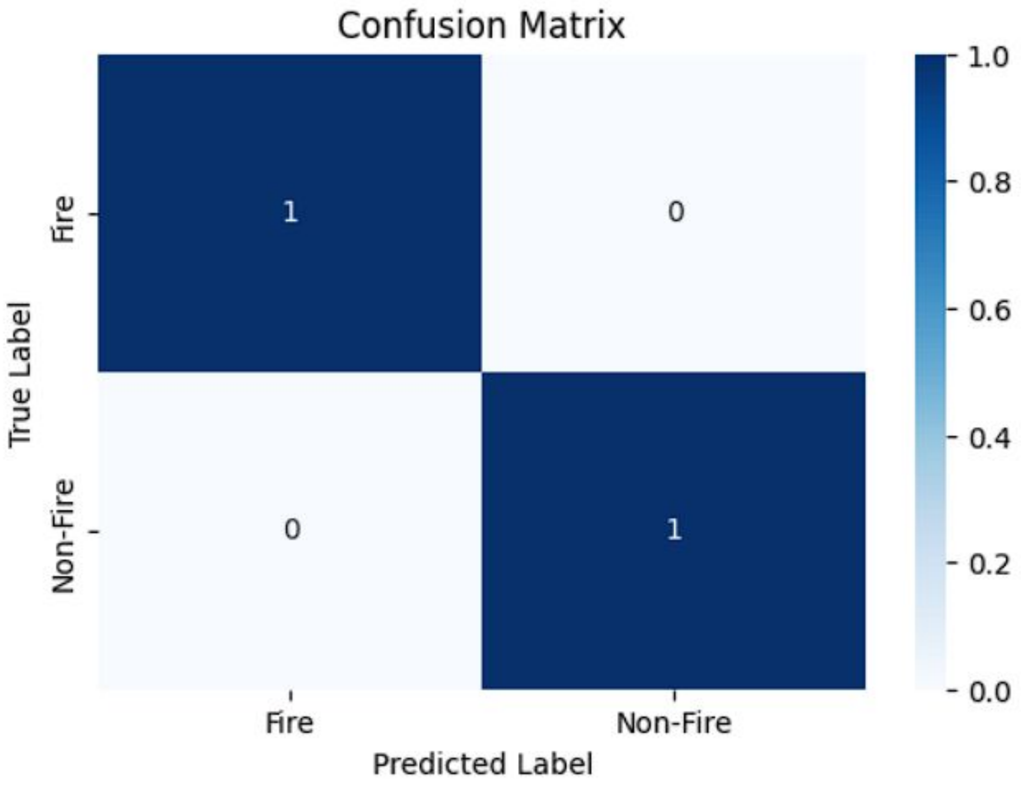
Non-trainable params: 0 (0.00 B)



Model accuracy
Train
Validation
Accuracy
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch



Model loss
Train
Validation
Loss
0.6
0.5
0.4
0.3
0.2
0.1
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch


Confusion Matrix
True Label
Fire
Non-Fire
1
0
0
1
Predicted Label
Fire
Non-Fire
1.0
0.8
0.6
0.4
0.2
0.0

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2
```

```python
accuracy = np.trace(cm) / np.sum(cm)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 1.00
```

- **Precision**: Indicates how many of the predicted positive instances were actually positive. Here, both classes have perfect precision (1.00), meaning there were no false positives.
- **Recall**: Measures how many of the actual positive instances were correctly predicted. Both classes have perfect recall (1.00), indicating no false negatives.
- **F1-Score**: The harmonic mean of precision and recall. An F1-score of 1.00 for both classes means the model is performing excellently.
- **Accuracy**: The proportion of correct predictions. In your case, it's 100% (1.00), meaning all predictions were correct.

CONCLUSION

Model Efficiency: The SqueezeNet architecture demonstrated its suitability for fire image classification by balancing accuracy and computational efficiency, making it ideal for deployment on resource-constrained systems.

Dataset Utilization: Proper dataset preprocessing, including splitting and augmentation, was critical in enhancing the model's generalization capability across diverse fire and non-fire scenarios.

Real-World Applications: The trained model has the potential for real-time fire detection in surveillance systems, contributing to timely interventions and reducing the risk of fire-related damage.

Future Scope: Enhancing the dataset diversity, exploring advanced optimization techniques, and deploying the model on edge devices can further improve its performance and applicability in real-world environments.

REFERENCES

[1] F. Khan, Z. Xu, J. Sun, F. M. Khan, A. Ahmed, and Y. Zhao, "Recent advances in sensors for fire detection," Sensors, vol. 22, no. 9, 2022.

[2] A. Filonenko, D. Hernandez, and K.-H. Jo, "Fast smoke detection for video surveillance using cuda," IEEE Transactions on Industrial Informatics, vol. PP, pp. 1–1, 09 2017.

[3] A. Jadon, M. Omama, A. Varshney, M. S. Ansari, and R. Sharma, "Firenet: A specialized lightweight fire & smoke detection model for real-time iot applications," CoRR, vol. abs/1905.11922, 2019.

[4] T.-H. Chen, P.-H. Wu, and Y.-C. Chiou, "An early fire-detection method based on image processing," in 2004 International Conference on Image Processing, 2004. ICIP '04., vol. 3, pp. 1707–1710 Vol. 3, 2004.

[5] T. Celik and H. Demirel, "Fire detection in video sequences using a generic color model," Fire Safety Journal, vol. 44, pp. 147–158, 02 2009. [6] G. Marbach, M. Loepfe, and T. Brupbacher, "An image processing technique for fire detection in video images," Fire Safety Journal, vol. 41, pp. 285–289, jun 2006.

[7] P. Foggia, A. Saggese, and M. Vento, “Real-time fire detection for videosurveillance applications using a combination of experts based on color, shape, and motion,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 25, 01 2015.

[8] Y. Habiboglu, O. Günay, and A. Cetin, “Covariance matrix-based fire ˘ and flame detection method in video,” Machine Vision and Applications, vol. 23, pp. 1–11, 11 2011.

[9] A. Rafiee, R. Dianat, M. Jamshidi, R. Tavakoli, and S. Abbaspour, “Fire and smoke detection using wavelet analysis and disorder characteristics,” in 2011 3rd International Conference on Computer Research and Development, vol. 3, pp. 262–265, 2011.

[10] S. Rinsurongkawong, M. Ekpanyapong, and M. N. Dailey, “Fire detection for early fire alarm based on optical flow video processing,” in 2012 9th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, pp. 1–4, 2012.

[11] M. Mueller, P. Karasev, I. Kolesov, and A. Tannenbaum, “Optical flow estimation for flame detection in videos,” IEEE Transactions on Image Processing, vol. 22, no. 7, pp. 2786–2797, 2013.

THANKYOU