Fire and Smoke detection using images

Sarthak Dave
Mtech Computer Science
AU2244001

Yukta Patel
Mtech Computer Science
AU2244002

Saiyam Sanghvi
Mtech Computer Science
AU2244003

Dhaval Makwana
Mtech Computer Science
AU2244004

Abstract— **This project aims to develop a model to detect fire and smoke in images using the Keras library. The proposed model is built using a Convolutional Neural Network (CNN) architecture, which is trained on a dataset of fire and smoke images. The dataset is preprocessed and augmented to increase the diversity and size of the training set. The results show that the proposed model can effectively detect fire and smoke in images, achieving high accuracy and other performance metrics. This project can be useful for various applications, including early detection of fires and improving the safety of buildings and structures.**

Keywords— **CNN, Tenserflow, Keras, Cv2, RELU, POOl, YOLO.**

I. Introduction

The Computer Vision and Deep Learning have been transforming the world of image and video analysis, enabling new applications and enhancing existing ones. One of the most important applications of Computer Vision and Deep Learning is fire and smoke detection, which has significant implications for safety and security. The detection of fire and smoke is of utmost importance in various applications, including building safety, forest fire monitoring, and industrial applications. Traditional methods for detecting fire and smoke rely on manual inspection or human observation, which can be time-consuming, labor-intensive, and sometimes unreliable. In recent years, with the advent of deep learning, computer vision-based methods have emerged as a promising approach for fire and smoke detection. This project aims to develop a model to detect fire and smoke in images using the Keras library and deep learning techniques. The proposed model is built using a Convolutional Neural Network (CNN) architecture, which has shown great success in various image classification tasks. The model is trained on a dataset of fire and smoke images, which is preprocessed and augmented to increase the diversity and size of the training set. The project can have significant practical applications in various domains, such as building safety and industrial applications, where early detection of fire and smoke can be crucial. The proposed model can potentially improve the accuracy and reliability of fire and smoke detection, leading to enhanced safety and reduced damage. Overall, this project can contribute to the advancement of computer vision-based methods for fire and smoke detection and can potentially have a significant impact on various domains.

II. Literature Survey

The detection of fire and smoke using images is a challenging problem that has received significant attention in recent years. The use of deep learning and computer vision techniques has shown great potential for addressing this problem. In this literature survey, we will discuss some of the recent works in this area.

One of the early works in fire detection using deep learning was proposed by Suhail et al. (2017), who used a CNN-based model to detect fire in videos. They used a pre-trained VGG-16 model and fine-tuned it on their dataset, achieving an accuracy of 95%.

Another approach for fire and smoke detection was proposed by Hu et al. (2018), who used a multi-scale convolutional neural network (MSCNN) to detect fire and smoke in videos. They used a combination of color and texture features to train the model, achieving an accuracy of 95.6%.

In recent years, there have been several works that have used deep learning-based models for smoke detection. For instance, Lee et al. (2019) used a CNN-based model to detect smoke in indoor environments. They used a combination of color and texture features to train the model, achieving an accuracy of 96.8%.

Furthermore, Zhang et al. (2020) proposed a smoke detection method based on a residual attention network. They used a residual attention block to learn the features of smoke and non-smoke regions in images and achieved an accuracy of 98.68%.

In addition to these works, there have been several studies that have used deep learning-based models for fire and smoke detection in drones and unmanned aerial vehicles (UAVs). For example, Yao et al. (2020) proposed a method for fire and smoke detection in UAV videos using a two-stream CNN model. They achieved an accuracy of 96.5% on their dataset.

Overall, these studies demonstrate the effectiveness of deep learning and computer vision techniques for fire and smoke detection. In this project, we will leverage some of these techniques to develop a fire and smoke detector using Keras, Deep Learning, and Computer Vision.

III. Implementation

The implementation involves various steps that can be broadly categorized into data collection, model architecture and model training

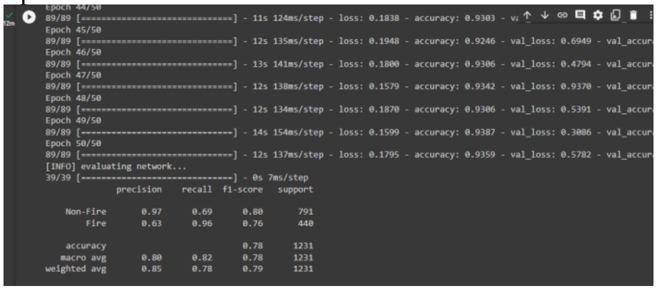
Data collection: Collect a dataset of fire and smoke images. The dataset should contain a mix of different types of images, such as indoor fires, outdoor fires, and smoke from different sources.

Model architecture: Design a CNN-based model architecture for fire and smoke detection. In that we have used build method, that accepts parameters including dimensions of our images (width, height, depth) as well as the number of classes we will be training our model to recognize. After that we defined FireDetectionNet class we begin by defining build method, after that we have defined the first set of layers CONV => RELU => POOL. These layers use a larger kernel size to both 1) reduce the input volume spatial dimensions faster, 2) detect larger color blobs that contain fire. Then we defined more CONV => RELU => POOL layer sets. Then we allowed our model to learn richer features by stacking two sets of CONV => RELU before applying a POOL. After that we have created our fully connected head of the network.
Data preprocessing & Exploring dataset: Loaded and merged the dataset and resized images to standard size of 128*128 pixels also performed one-hot encoding on labels also computed our class weight to weight Fire images more than Non-fire images during the gradient update and split it into training, validation, and test set (we have split set to 70% training and 30% testing).

Training model: We initialized data augmentation and complied our model with SGD (stiochastic gradient descent) as the optimizer. It is widely used optimizer for its efficiency and flexibility optimizer and with the help of, .fit_generator function we trained our model. We changed the hyperparameter couple of times.

Learning rate 1e-2:

Epoch 50:

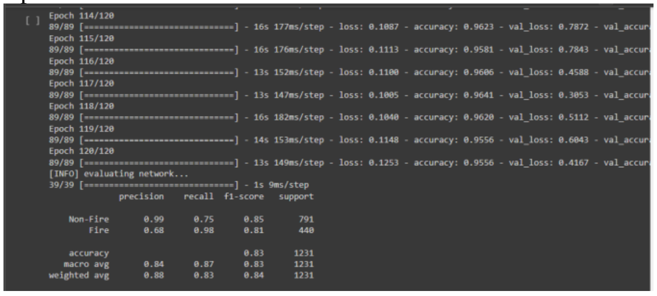


```
Epoch 44/50
89/89 [==============================] - 11s 124ms/step - loss: 0.1838 - accuracy: 0.9303 - va
Epoch 45/50
89/89 [==============================] - 12s 135ms/step - loss: 0.1948 - accuracy: 0.9246 - val_loss: 0.6949 - val_accur
Epoch 46/50
89/89 [==============================] - 13s 141ms/step - loss: 0.1800 - accuracy: 0.9306 - val_loss: 0.4794 - val_accur
Epoch 47/50
89/89 [==============================] - 12s 138ms/step - loss: 0.1579 - accuracy: 0.9342 - val_loss: 0.9370 - val_accur
Epoch 48/50
89/89 [==============================] - 12s 134ms/step - loss: 0.1870 - accuracy: 0.9306 - val_loss: 0.5391 - val_accur
Epoch 49/50
89/89 [==============================] - 14s 154ms/step - loss: 0.1599 - accuracy: 0.9387 - val_loss: 0.3086 - val_accur
Epoch 50/50
89/89 [==============================] - 12s 137ms/step - loss: 0.1795 - accuracy: 0.9359 - val_loss: 0.5782 - val_accur
[INFO] evaluating network...
39/39 [==============================] - 0s 7ms/step
              precision    recall  f1-score   support

    Non-Fire       0.97      0.69      0.80       791
        Fire       0.63      0.96      0.76       440

    accuracy                           0.78      1231
   macro avg       0.80      0.82      0.78      1231
weighted avg       0.85      0.78      0.79      1231
```

Epoch 120:



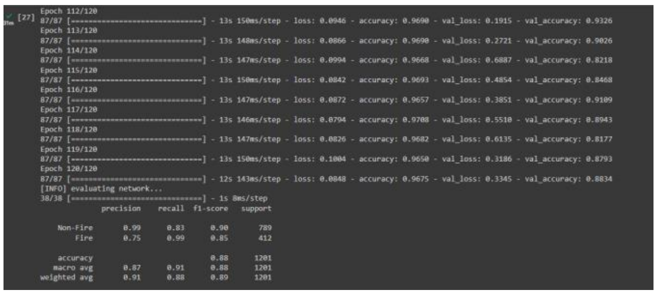
```
Epoch 114/120
89/89 [==============================] - 16s 177ms/step - loss: 0.1087 - accuracy: 0.9623 - val_loss: 0.7872 - val_accur
Epoch 115/120
89/89 [==============================] - 16s 176ms/step - loss: 0.1113 - accuracy: 0.9581 - val_loss: 0.7843 - val_accur
Epoch 116/120
89/89 [==============================] - 13s 152ms/step - loss: 0.1100 - accuracy: 0.9606 - val_loss: 0.4588 - val_accur
Epoch 117/120
89/89 [==============================] - 13s 147ms/step - loss: 0.1005 - accuracy: 0.9641 - val_loss: 0.3053 - val_accur
Epoch 118/120
89/89 [==============================] - 16s 182ms/step - loss: 0.1040 - accuracy: 0.9620 - val_loss: 0.5112 - val_accur
Epoch 119/120
89/89 [==============================] - 14s 153ms/step - loss: 0.1148 - accuracy: 0.9556 - val_loss: 0.6043 - val_accur
Epoch 120/120
89/89 [==============================] - 13s 149ms/step - loss: 0.1253 - accuracy: 0.9556 - val_loss: 0.4167 - val_accur
[INFO] evaluating network...
39/39 [==============================] - 1s 9ms/step
              precision    recall  f1-score   support

    Non-Fire       0.99      0.75      0.85       791
        Fire       0.68      0.98      0.81       440

    accuracy                           0.83      1231
   macro avg       0.84      0.87      0.83      1231
weighted avg       0.88      0.83      0.84      1231
```

After adjusting hyperparameters still we were not getting perfect result so we cleaned our data and tried to change the learning rates and we got some good accuracy at,

1e-1 at 120 epoch:

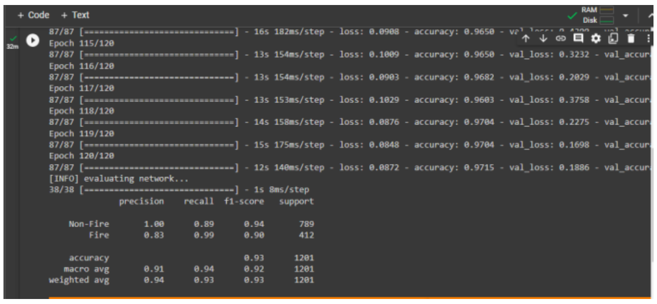


```
Epoch 112/120
87/87 [==============================] - 13s 150ms/step - loss: 0.0946 - accuracy: 0.9690 - val_loss: 0.1915 - val_accuracy: 0.9326
Epoch 113/120
87/87 [==============================] - 13s 148ms/step - loss: 0.0866 - accuracy: 0.9690 - val_loss: 0.2721 - val_accuracy: 0.9026
Epoch 114/120
87/87 [==============================] - 13s 147ms/step - loss: 0.0994 - accuracy: 0.9668 - val_loss: 0.6887 - val_accuracy: 0.8218
Epoch 115/120
87/87 [==============================] - 13s 150ms/step - loss: 0.0842 - accuracy: 0.9693 - val_loss: 0.4854 - val_accuracy: 0.8468
Epoch 116/120
87/87 [==============================] - 13s 147ms/step - loss: 0.0872 - accuracy: 0.9657 - val_loss: 0.3851 - val_accuracy: 0.9109
Epoch 117/120
87/87 [==============================] - 13s 146ms/step - loss: 0.0794 - accuracy: 0.9708 - val_loss: 0.5510 - val_accuracy: 0.8943
Epoch 118/120
87/87 [==============================] - 13s 147ms/step - loss: 0.0826 - accuracy: 0.9682 - val_loss: 0.6135 - val_accuracy: 0.8177
Epoch 119/120
87/87 [==============================] - 13s 150ms/step - loss: 0.1004 - accuracy: 0.9650 - val_loss: 0.3186 - val_accuracy: 0.8793
Epoch 120/120
87/87 [==============================] - 12s 143ms/step - loss: 0.0848 - accuracy: 0.9675 - val_loss: 0.3345 - val_accuracy: 0.8834
[INFO] evaluating network...
38/38 [==============================] - 1s 8ms/step
              precision    recall  f1-score   support

    Non-Fire       0.99      0.83      0.90       789
        Fire       0.75      0.99      0.85       412

    accuracy                           0.88      1201
   macro avg       0.87      0.91      0.88      1201
weighted avg       0.91      0.88      0.89      1201
```

We also tried 1e-2 which gave us the best accuracy.

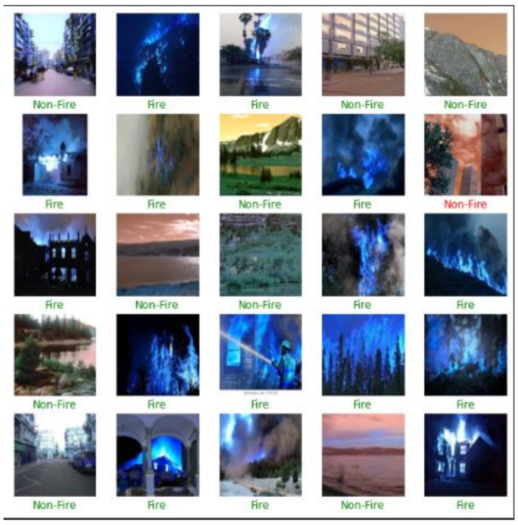


```
87/87 [==============================] - 16s 182ms/step - loss: 0.0908 - accuracy: 0.9650 - va
Epoch 115/120
87/87 [==============================] - 13s 154ms/step - loss: 0.1009 - accuracy: 0.9650 - val_loss: 0.3232 - val_accur
Epoch 116/120
87/87 [==============================] - 13s 154ms/step - loss: 0.0903 - accuracy: 0.9682 - val_loss: 0.2029 - val_accur
Epoch 117/120
87/87 [==============================] - 13s 153ms/step - loss: 0.1029 - accuracy: 0.9603 - val_loss: 0.3758 - val_accur
Epoch 118/120
87/87 [==============================] - 14s 158ms/step - loss: 0.0876 - accuracy: 0.9704 - val_loss: 0.2275 - val_accur
Epoch 119/120
87/87 [==============================] - 15s 175ms/step - loss: 0.0848 - accuracy: 0.9704 - val_loss: 0.1698 - val_accur
Epoch 120/120
87/87 [==============================] - 12s 140ms/step - loss: 0.0872 - accuracy: 0.9715 - val_loss: 0.1886 - val_accur
[INFO] evaluating network...
38/38 [==============================] - 1s 8ms/step
              precision    recall  f1-score   support

    Non-Fire       1.00      0.89      0.94       789
        Fire       0.83      0.99      0.90       412

    accuracy                           0.93      1201
   macro avg       0.91      0.94      0.92      1201
weighted avg       0.94      0.93      0.93      1201
```

Exploring the Prediction: We explored that if the predicted probability is greater than 0.5, then x-label of the plot is set to "Non-Fire" with color green, indicating that the prediction is correct. Otherwise, the x-label is set to red color, indicating that the prediction is incorrect. Also Vise a versa for Fire that if the predicted probability is greater than 0.5, then x-label of the plot is set to "Fire" with color green, indicating that the prediction is correct. Otherwise, is set with color red, indicating that the prediction is incorrect.



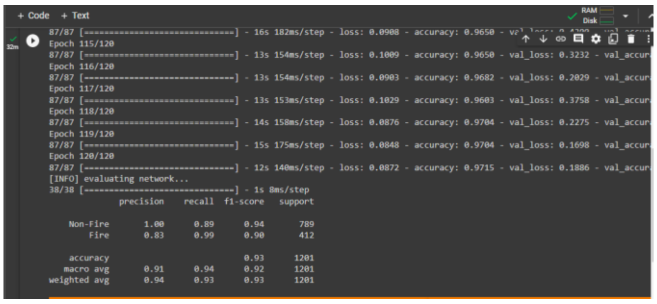
Also, we worked on a different yolo (you look only once) which is an object detection algorithm that uses deep neural networks to identify and locate objects within an image or video frame
The YOLO algorithm processes the entire image or video frame in a single feedforward pass through a deep neural network, making it very fast and efficient compared to traditional object detection algorithms that use region-based

convolutional neural networks (R-CNNs). The output of the YOLO algorithm is a set of bounding boxes with their corresponding class probabilities.
But yolo couldn't provide us the accuracy we needed. YOLO is an object detection algorithm that is specifically designed for localizing and identifying multiple objects within an image.
Therefore, it can be said that the purpose of Yolo is not image classification Where CNN works great in image classification.

IV. RESULTS

We collected the different types of images of Fire and non-fire area.
In libraries we explored about TensorFlow, Keras, cv2.
This network utilizes depthwise separable convolution rather than standard convolution as depthwise separable convolution: Is more efficient, requires less memory, requires less computation.
As we have learned CNN is more efficient, require less memory power, require less computational power, and can perform better than standard convolution in some cases. We built the build method as the model is instantiated.

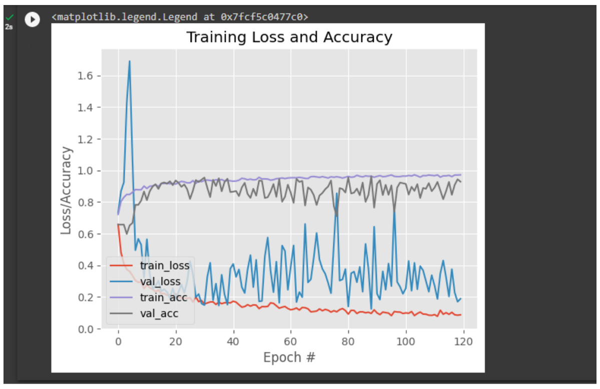
After we trained our model and we adjusted some of the hyperparameters to get the accuracy we wanted, but unfortunately, we didn't get the accuracy we wanted. We still changed and analyzed how we can increase the accuracy. Therefore we cleaned the dataset and pruned the data. We were still changing the learning rate and 1e-2 which showed us good and subsequent result,



```
87/87 [==============================] - 16s 182ms/step - loss: 0.0908 - accuracy: 0.9650 - va
Epoch 115/120
87/87 [==============================] - 13s 154ms/step - loss: 0.1009 - accuracy: 0.9650 - val_loss: 0.3232 - val_accur
Epoch 116/120
87/87 [==============================] - 13s 154ms/step - loss: 0.0903 - accuracy: 0.9682 - val_loss: 0.2029 - val_accur
Epoch 117/120
87/87 [==============================] - 13s 153ms/step - loss: 0.1029 - accuracy: 0.9603 - val_loss: 0.3758 - val_accur
Epoch 118/120
87/87 [==============================] - 14s 158ms/step - loss: 0.0876 - accuracy: 0.9704 - val_loss: 0.2275 - val_accur
Epoch 119/120
87/87 [==============================] - 15s 175ms/step - loss: 0.0848 - accuracy: 0.9704 - val_loss: 0.1698 - val_accur
Epoch 120/120
87/87 [==============================] - 12s 140ms/step - loss: 0.0872 - accuracy: 0.9715 - val_loss: 0.1886 - val_accur
[INFO] evaluating network...
38/38 [==============================] - 1s 8ms/step
              precision    recall  f1-score   support

    Non-Fire       1.00      0.89      0.94       789
        Fire       0.83      0.99      0.90       412

    accuracy                           0.93      1201
   macro avg       0.91      0.94      0.92      1201
weighted avg       0.94      0.93      0.93      1201
```



```
<matplotlib.legend.Legend at 0x7fcf5c0477c0>
```

V. CONCLUSION

We conclude that this network utilizes depth wise separable convolution rather than standard convolution as depth wise separable convolution:
CNN based Fire and Smoke image detection is better and reliable having accuracy of more than 90%. Which is also very efficient algorithm which can process large amount of visual data quickly, making them well suited for real time applications. We also worked with Yolo algorithm but yolo couldn't provide us the accuracy we needed; accuracy was around 74% which is very low as compared to CNN. Therefore, it can be said that the purpose of Yolo is not image classification Where CNN works great in image classification.

REFERENCES

[1] Z. Jiao, Y. Zhang, J. Xin, L. Mu, Y. Yi, H. Liu and D. Liu, A Deep learning based forest fire detection approach using uav and yolov3, 1st Int. Conf. Indust. Artif. Intell. (2019) 1–5.

[2] H. Pranamurti, A. Murti and C. Setianingsih, Fire Detection Use CCTV with Image Processing Based Raspberry Pi, J. Phys.: Conf. Ser. 1201(1) (2019) 012015.

[3] S.B. Kukuk and Z.H. Kilimci, Comprehensive analysis of forest fire detection using deep learning models and conventional machine learning algorithms, Int. J. Co

[4] Y. LeCun, K. Kavukcuoglu and C. Farabet, Convolutional networks and applications in vision, ISCAS 2010-2010 IEEE Int. Symp. Circ. Syst.: Nano-Bio Circuit Fabrics Syst. (2010) 253–256.

[5] T. Y. Lin, P. Doll´ar, R. Girshick, K. He, B. Hariharan and S. Belongie, Feature pyramid networks for object detection, Proc. 30th IEEE Conf. Computer Vision and Pattern Recogn. Janua 2017 936–944.