FOREST FIRE DETECTION

PRIYA ENUGANTI , RANA GHONEIM, SAJA AL KARAWI,
SHRIDHAR KSHIRSAGAR, SHREETI UPRETI

Project Link : https://github.com/Shreets/Forest-Fire-Detection
VideoLink:
https://drive.google.com/file/d/1hFa8XuqWcI6hyWO-DdbZt3GRAUvNkoir/view?usp=sharing

INTRODUCTION

Forests being one of the many valuable natural resources, form a vital part of every ecosystem. They are responsible for producing almost 1/3rd of the total oxygen on earth. Considering the heavy dependency of humans on these forests, a forest fire might have apocalyptic consequences if not controlled within a certain time frame.

The computer vision-based fire detection method has increasingly superseded the conventional method and emerged as a significant trend as a result of the quick development of digital camera and image processing technologies. Although, this method comes with challenges like the complex background of the forest fire, which becomes one of the obstacles when deciding on the features that may help identify the characteristic patterns of a forest fire.

Applying convolutional neural networks (CNN) on images to identify patterns being a theoretically proven concept, can help extract deeper features and minimize blindness and unpredictability to a substantial extent in the feature extraction process, which subsequently increase the accuracy of fire image recognition.

The model's capacity to correctly identify and distinguish one class of the image from another is what gives the project its significance. The image data collection contains many distinctive objects, but only the most crucial ones must be chosen in order to train our model as accurately as possible. For instance, smoke in the pictures can be easily

confused with fog. In these situations, it would be practical for the image to also recognize fog as the first fire detection. To prevent these situations, we extract an image feature that searches for instances of fire as well as smoke in the images. A model's performance is based on the features that it acquires during training. Therefore, the project places a lot of emphasis on feature extraction and selection so that the model can quickly capture new, relevant, and varied possibilities.

The major goal of this project is to create a system that can detect fires, especially forest fires, which spread more quickly and have a more negative impact on the ecology. Due to the project's sole reliance on picture data, it must address several false alarms that could indicate fire. False alarms include things like a cloudy or foggy sky that looks to be smoke or trees with yellow leaves that give the appearance of fire. With the use of specific feature engineering techniques, it will be possible to overcome these obstacles to real-fire detection, improving the system's prediction accuracy in the process.

BACKGROUND

1. The Forest Fire Smoke Detection in Video Based on Digital Image Processing study described a technique for detecting fire using video. The concept is based on the examination of static and dynamic characteristic data in digital image processing. The approach consists of the following steps: The first step is to determine the area of change between the current input frame and the background image. The second step is to find regions of interest (ROIs) using a connected component algorithm. The area of ROI is calculated using a convex hull algorithm, which separates the area of change from the image. The third step is to calculate static and dynamic characteristics.

Using this result, we decide whether to move forward with the proposed method. The result shows how proficient this technology is at picking up fire smoke.

2. In a work titled Survey on Different Smoke Detection Techniques Using Image Processing, a method based on the wavelet model and a color model of the smoke was

suggested. The recommended method makes use of the energy variation of the wavelet model and the color model of the smoke. Smoke can be recognized based on a reduction in the energy ratio between background and current in the wavelet domain. The variance of the current pixel color is calculated using the color model. The Bayesian classifier can identify smoke by combining these two features.

3. The system created by Osman Gunay and Habiboglu is built on Covariance Descriptors, Color Models, and SVM Classifiers, which were explained in their paper, Flame Detection technique in video using Covariance descriptors. Video data is used by this program. In this method, the video data is broken up into temporal blocks, and the spatio-temporal covariance matrix is used to derive covariance features. The fire can be located using this feature. The SVM Classifier is used to categorize areas with fire and areas that resemble fire. Only clear data is supported by this technology; fuzzy data is not supported.

MODEL ARCHITECTURE

Training Architecture

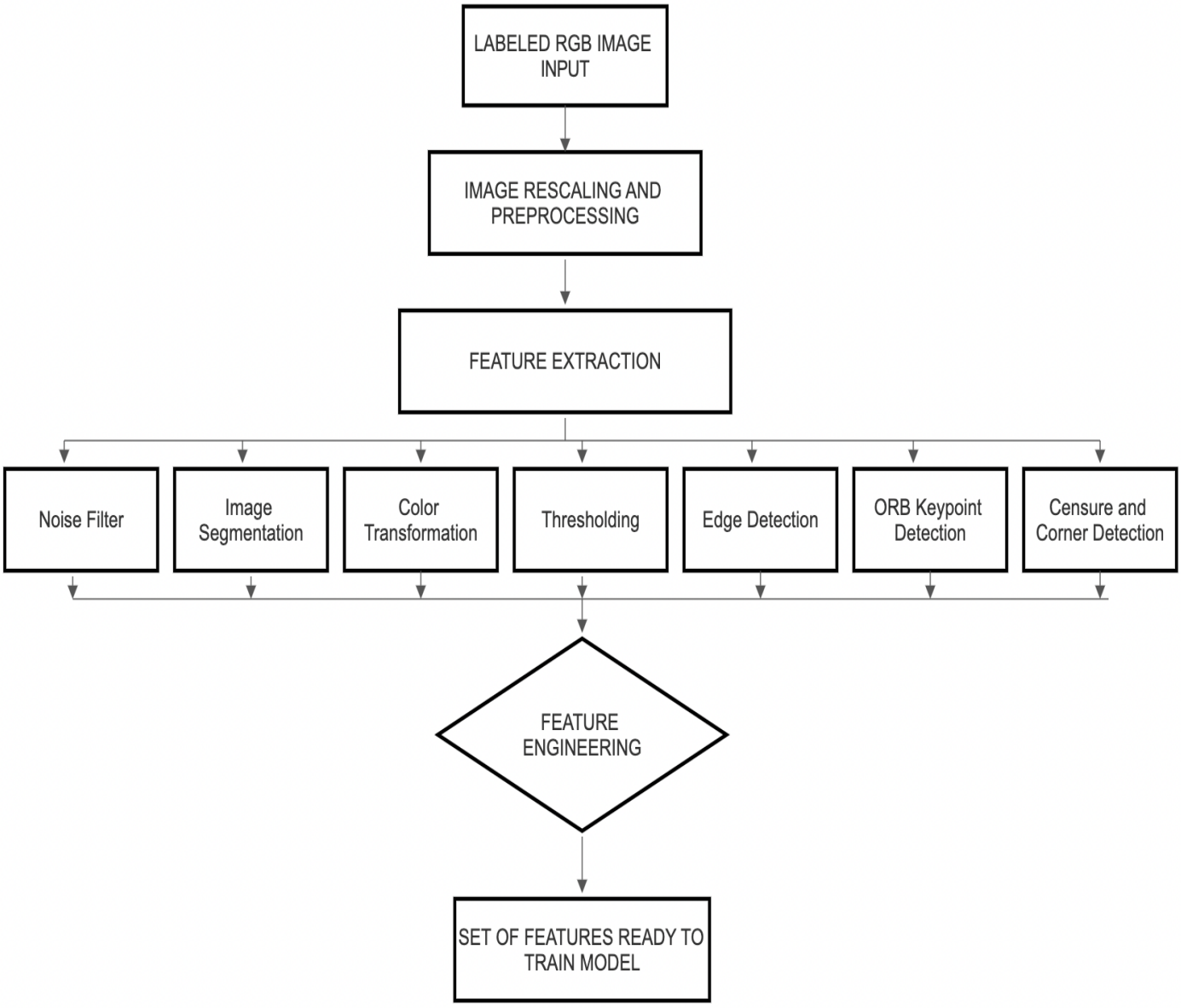


Fig. Architecture to generate set of features to train the model

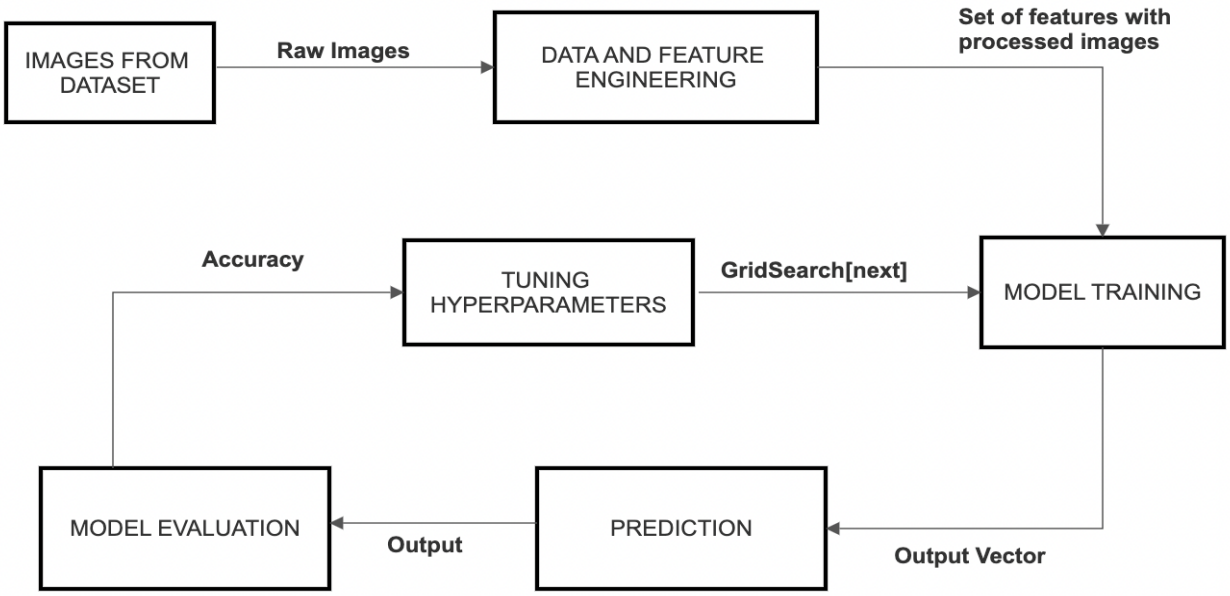


Fig. Overall Architecture to Train Model for Fire Detection

i) Image input : The data set has images in their raw form where they have been labeled as fire, no fire or start fire. The image needs to be further processed before using it to train the model.

ii) Data and feature engineering : The images are processed in order to fit the model so that model can make precise predictions. The images are rescaled and then the most relevant features are extracted that can distinguish existence or absence of fire in images. In doing so multiple feature engineering techniques were applied to highlight the key aspects in the images. The techniques used were noise filtering, image segmentation, edge detection, color transformation, thresholding, keypoints detection and censure detection. Despite attempting to highlight and extract the most prominent features that are the signs of forest fire, not all features can be equally helpful. So, the most relevant

features that in combination yielded the most accurate results were prepared to train the model.

iii) Training Model : The input that goes into the model are images, hence Convolutional Neural Network is being used. To save the resources on training huge data in neural networks we are using the initially training weights and retraining them with a newer set of inputs, i.e transfer learning is being implemented using InceptionV3 model. Certain layers of the networks are frozen and only part of the layers are trained with new data.

iv) Prediction : The model takes in the input data and predicts the probabilities for each class. The probabilities are then transformed to a single prediction for a particular class.

v) Model Evaluation : The training sets were being used to train the model while the validation and test sets were held out to tune and test the model accuracy respectively. Once the model was trained. The data from the test set were passed through the trained model to predict the outcome. The accuracy recorded in this set is the actual overall accuracy of the model.

vi) Hyperparameter Tuning : Once the model is tested with the test set, it can be decided if the model's performance is adequate or needs to be improved. This is done by changing the hyperparameters of the model that tiles the best result. This is done using the validation datasets.

Once the model is trained and the optimum performance is achieved, then the model can be deployed to use in real life applications to check for forest fires. Although the model is being trained with images, in application, the model uses video feed to detect fire in forests. So the architecture for the deployed model differs from that during the training.

Deployment Architecture

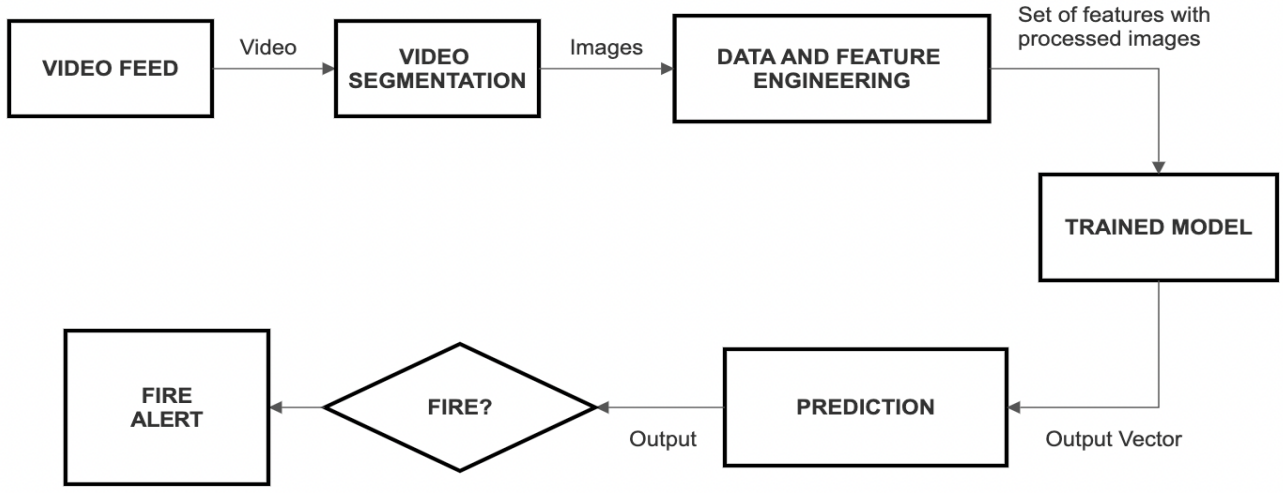


Fig. Architecture of Trained Model for Fire Detection

i) Video input : The data set has images in their raw form where they have been labeled as fire, no fire or start fire. The image needs to be further processed before using it to train the model.

ii) Segmenting Videos : The continuous video feed input is converted to multiple image segments frame by frame in order to test for presence of fire in them.

iii) Data and feature engineering : Data engineering and feature engineering in deployed model words in the same manner as mentioned for the training architecture.

v) Prediction : The features obtained from the processed images are then fed into the trained model. The model attempts to correctly predict the class of the images that have not been provided with any labels. These predictions are based on the patterns that the model had previously learned from the selected set of features during the model training process. This gives the output vector with probabilities of the possible classes which is then transformed to a single prediction for a particular class.

DATASET

Detailed description of Dataset

There are a total of 6000 images in the dataset. It is classified into three classes

No fire (this denotes that there isn't any fire in the images), Start Fire (this shows that the fire is basically getting started and hasn't yet spread throughout the surroundings) and Fire (this denotes the lighting of a disastrous fire). The dataset is balanced because there are about similar amounts of samples within each of the three classes.

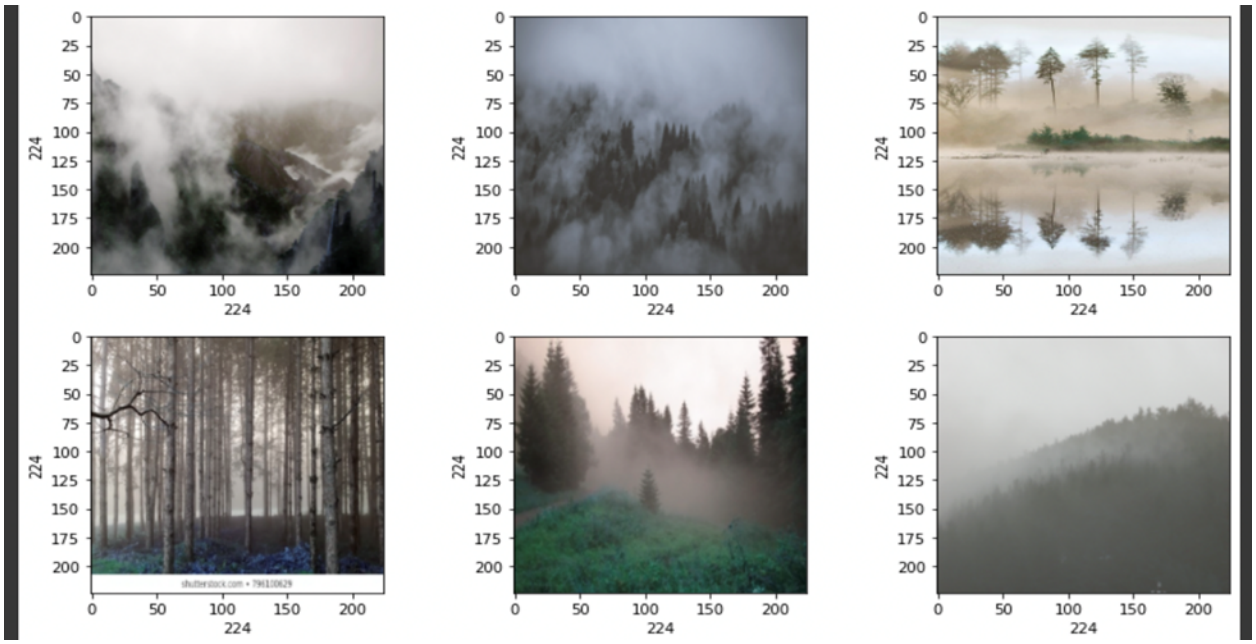


Fig. Sample images

The dataset was then divided into three sets of data to train the model. 80% of the data was used as a training set that was fed into the model with labels for the model to learn the patterns so as to be able to classify the images with or without fire. 20% of these data were separated and only used later to test the performance of the

model, once the model was fully trained. This was necessary to make sure that the model generalizes well the images it had not seen during the training phase.

Detail design of Features

The process of extracting important and useful properties for modeling from raw data is commonly known as feature engineering. This usually entails extracting information from images, such as color, texture, and shape. There are many approaches to execute feature engineering, and the methodology that is taken depends on the type of data we are dealing with. As our goal is to identify real fire, we will try various relevant features as follows:

1.Noise Filter:

It's a variety of operations used to reduce the noise from data collected on the construction and infrastructure areas. We tried with three different filters to make our images clear and smooth before moving on to the next step in the image processing. The first filter we used was the Gaussian filter which is a Low pass filter that is used to remove noise and as we all know blurs images. The second filter we applied was the Bilateral Filter, which also uses a technique to smooth images while preserving edges. The third filter was the Median filter, which is well known for maintaining edges during noise removal also it uses in order to reduce random noise.

Fig. Gaussian Noise Filter

2.Image Segmentation:

The technique of dividing a picture into various portions and segments is known as image segmentation. This method is used to identify boundaries in an image as lines and curves, as well as to name each pixel in the image. Thus, segmentation aids in streamlining and altering an image's representation to make it simpler and more understandable to examine. Image segmentation is particularly helpful for splitting and grouping images. Machine learning, computer vision, artificial intelligence, medical imaging, recognition tasks, video surveillance, object detection, and other fields all use image segmentation extensively. It affects a variety of fields, including space research and healthcare.

We will use Otsu's segmentation which is categorized as a threshold-based segment. In Otsu's Segmentation, the input image is first processed before attempting to obtain the image's histogram, which will display the distribution of pixels in the image. Here, our attention is on peak value. The threshold value must then be calculated and compared to the image pixels in the following phase. If the value is more than the threshold but less than black, set the pixel to white. It carries out automatic thresholding as a result.

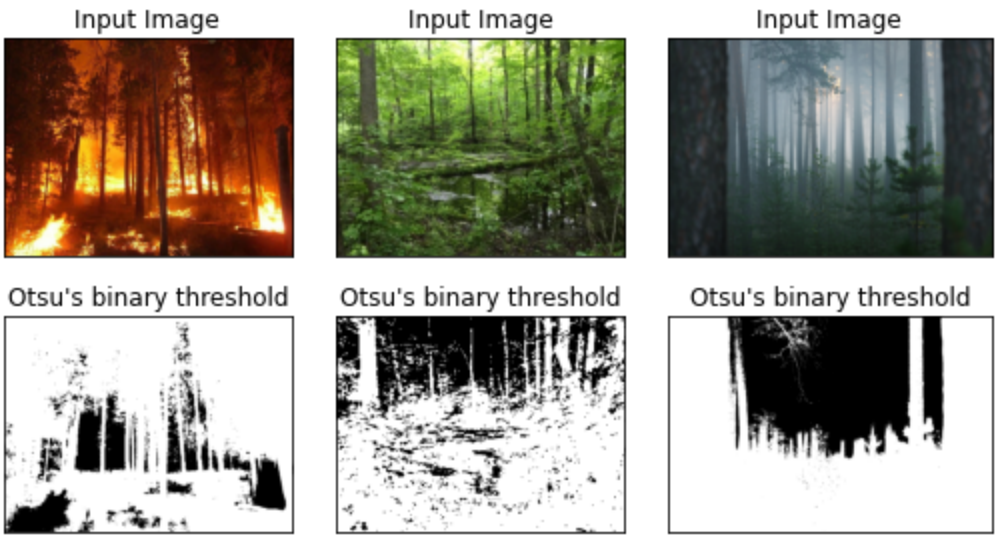


Fig. Image Segmentation

3. Color Transformation : RGB to LAB:

Designed using the frequently used and well known RGB color palette, LAB is another effective way of communicating colors. LAB is a color space with a much richer and vast color spectrum when compared with other color palettes like RGB, BGR or YCbCr. 'L' stands for lightness in the range of 0 to 100. A and B traverse all the colors from red to green and yellow to blue respectively. In the context of this project, color transformation has been applied to add a new flavor to the process of extracting features from the image. Additionally, fire and smoke being the vital higher level features for this project, can be detected only by studying the color changes throughout the image. As the spectrum of LAB captures much more minute details in the color patterns of the image, its use will only enhance the capabilities of the classifier.

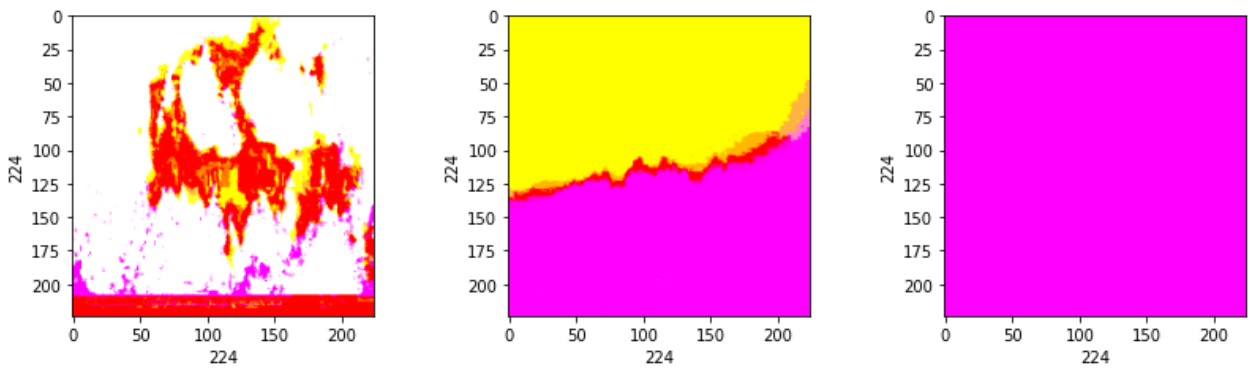


Fig. RGB to LAB color transformation

4. Thresholding :

Thresholding is a technique used for image segmentation and is achieved by varying pixels. This helps to better analyze the image. Particularly for this dataset, thresholding helps in segmenting fire/smoke and forest background. This will help the model learn to differentiate the details better from the image.

Here two types of thresholding are applied to images. The first is binary thresholding which outputs a binary image and it can be applied to both gray scale and colored images. The second is Adaptive thresholding which is subclassified into Adaptive Gaussian, Adaptive Mean thresholding. It is observed that binary thresholding has given better image segmentation for analysis.

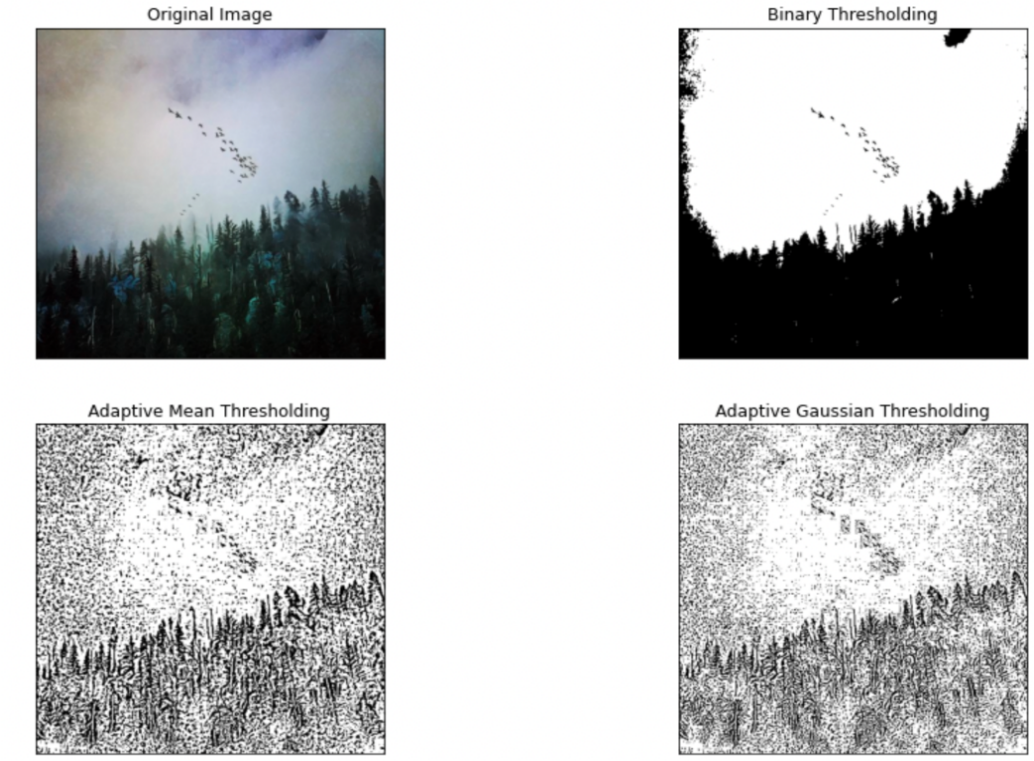


Fig. Thresholding

5. Edge Detection:

Edge detection here is being done to check if the edges can separate the fire entity in the image. Through edge detection the discontinuity in the brightness is checked when one image section changes to another. It is understood that the image section with the fire must have a sudden change in the intensity of brightness and the change in edge should give some indication of the image segment with fire in it. The images were not grayscale before applying edge detection techniques as it is also necessary that the hue is detected in the image as hue is an important factor to detect fire in the image. First the horizontal and the vertical edges were checked to see if significant segments of images are detected through these edges. Further, the vertical edges were also checked to identify similar edges. Then further the vertical and horizontal edges were combined to see if any significant difference is observed in the images with and without fire where the images with fire have distinct characteristic differences compared to the ones without fire.

A Sobel filter has been used to identify the edges. Sobel filter calculates the gradient of the intensity of each pixel. Using sobel filter the intensity changes in the image was identified

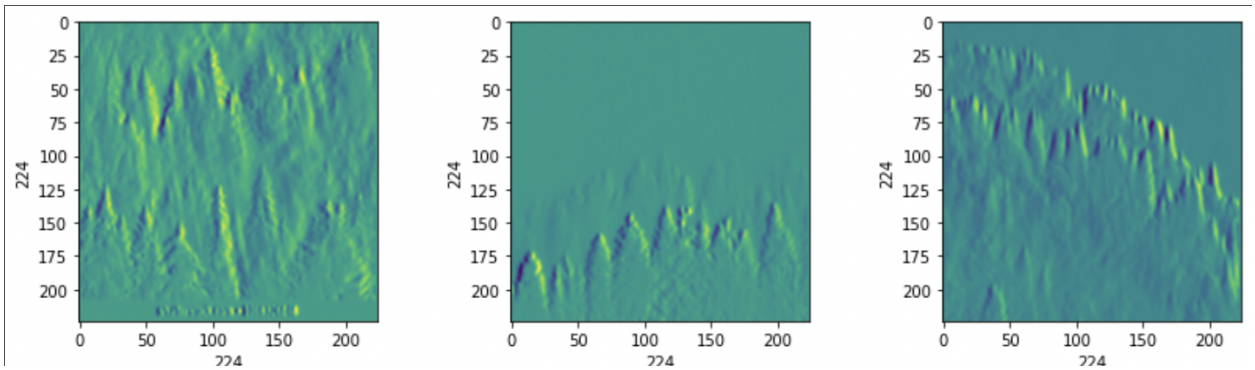


Fig : Horizontal Edge Detection

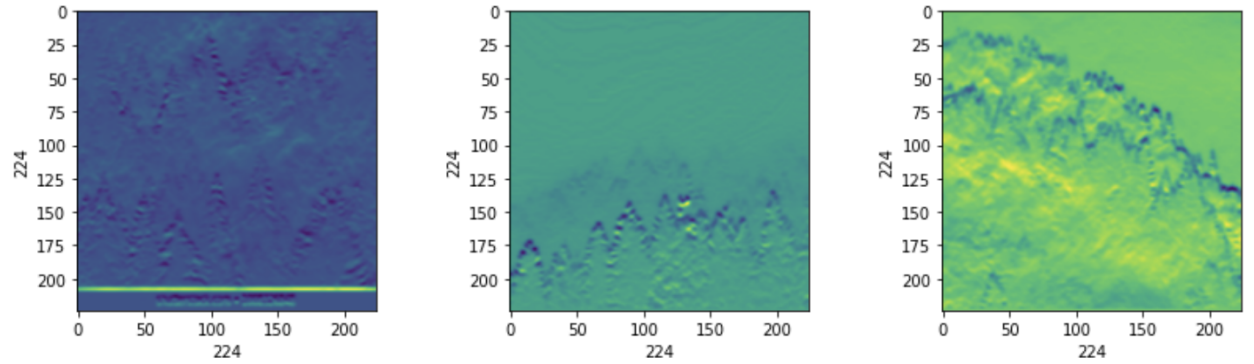


Fig : Vertical Edge Detection

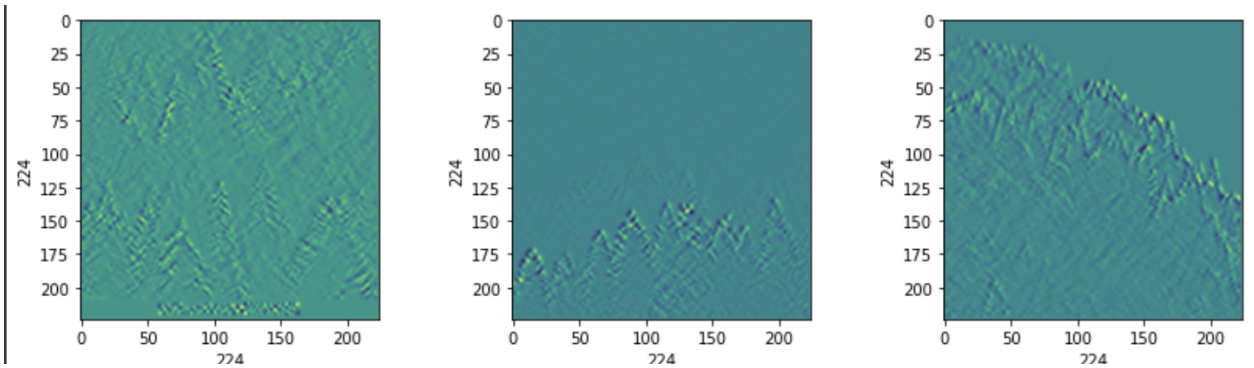


Fig : Horizontal and Vertical Edges Combined

6. ORB Keypoint Detection:

ORB(Oriented FAST and Rotated BRIEF) is an amalgamation of two different Image Processing Techniques. FAST (Features from Accelerated Segment Testing) which is an open source adaptation of a keypoint detector like SIFT or SURF. BRIEF (Binary Robust Independent Elementary Features) is an effective keypoint descriptor algorithm which is usually used in tandem with a keypoint extractor. The purpose of ORB in the project is to detect sudden shifts in the pixel intensities, which can be looked upon as detecting corners in the image. Output generated in the notebook file suggests that the algorithm detects corners of the trees, sharp rocky edges,etc. in the images. A set of descriptors generated by the BRIEF part in ORB is beneficial in matching the key points of a certain image with any other image. This comes in handy for the classifier model to train itself with this information and further predict an outcome based on the learnt keypoint descriptors.

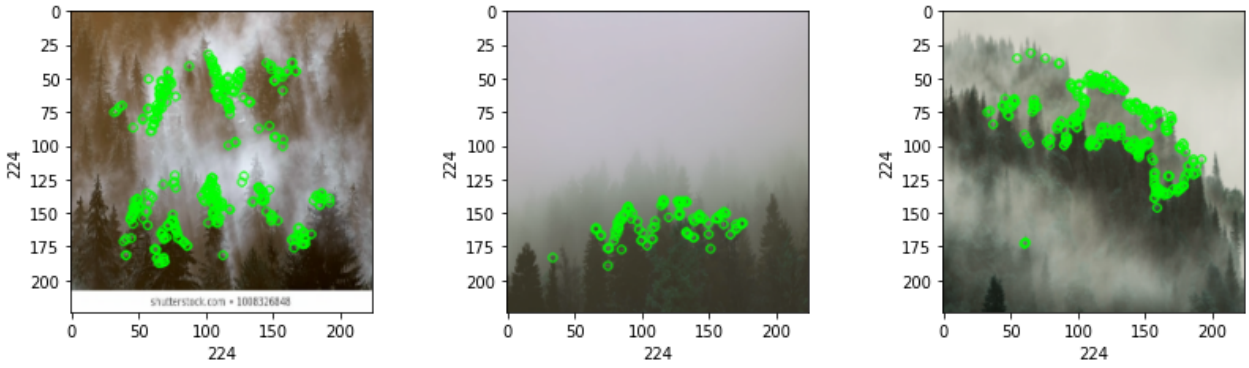


Fig. Oriented FAST and Rotated BRIEF keypoint detection

7. Image Orientation

Orientation is varying the angle of the image to get a desired real time scenario than an inverted or tilted image. This would ensure images of similar class and pattern are learnt as the same category by the model which avoids confusion while the model is learning by updating weights.

Fig. Image Orientation

8. Censure and Corner Detection

Censure feature detector detects the significant features and is highly used in real time images to extract important details. Harris Corner detector as name suggests helps in locating the corners and helps in inferring image features.

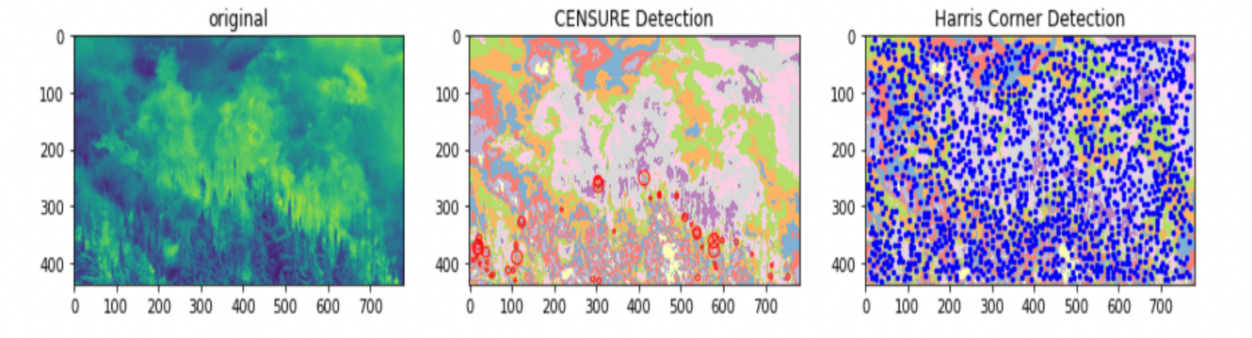
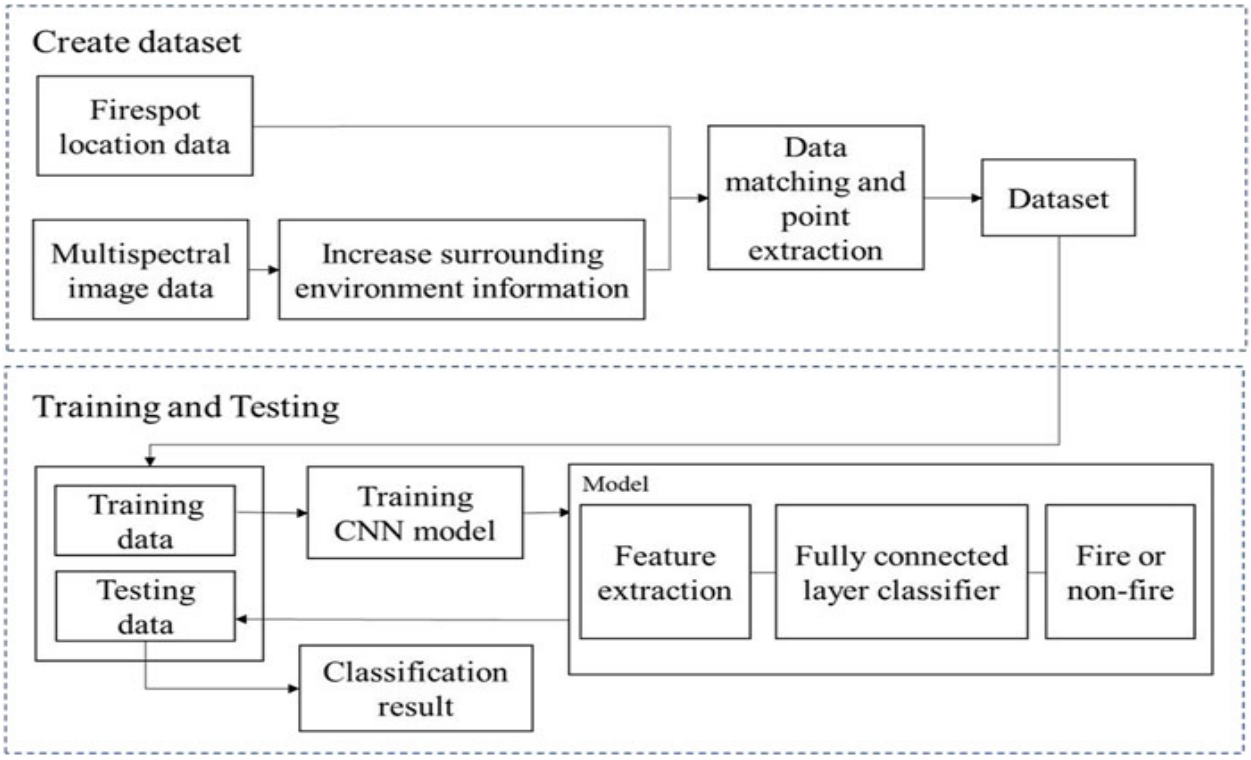


Fig. Images after censure and corner detection

ANALYSIS OF DATA



Images from the fire, start fire, and no fire classes make up the dataset. Images with the no fire class only contain forested or vegetated regions that are in their natural states. Images containing fire can be found in the image class fire. The class start fire contains all of the early warning indicators that show when a forest fire starts, such as smoke and sporadic little spots of fire.

The dataset in this case is balanced since there are about identical numbers of samples for each of the three classes.

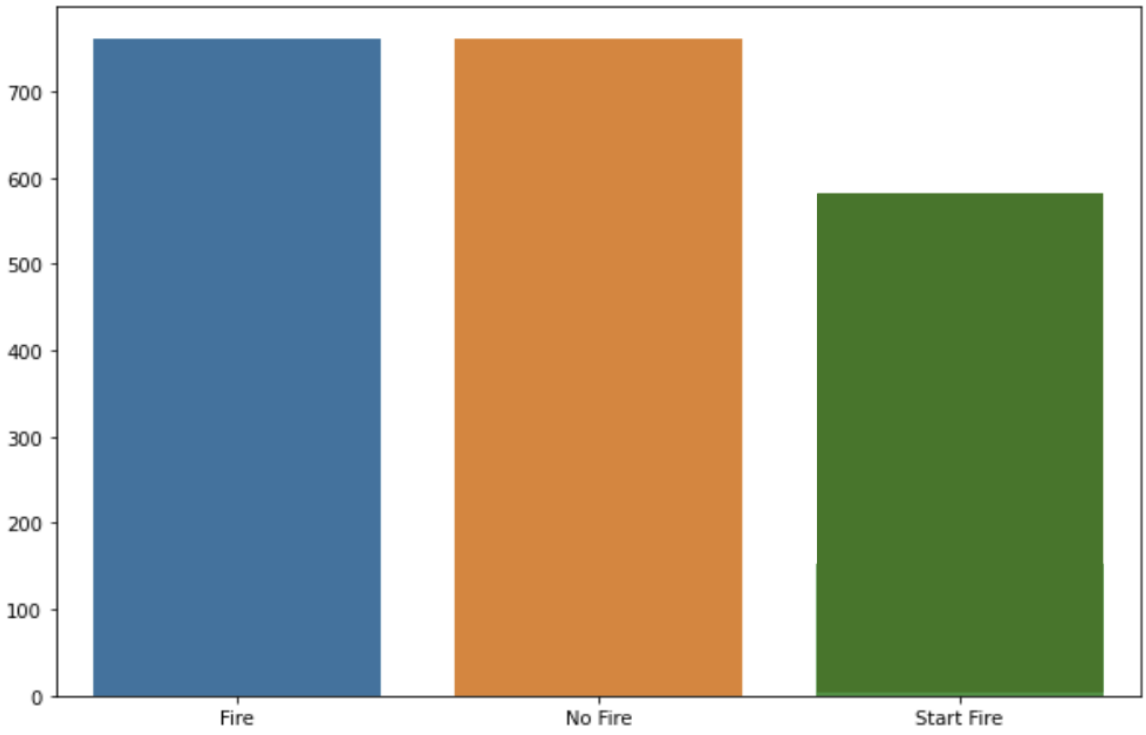


Fig. Data Distribution

In order to examine intensity changes within data samples, the dataset was then further displayed as a histogram. This was done to determine if the histogram can detect any variations between the photos with and without fire.

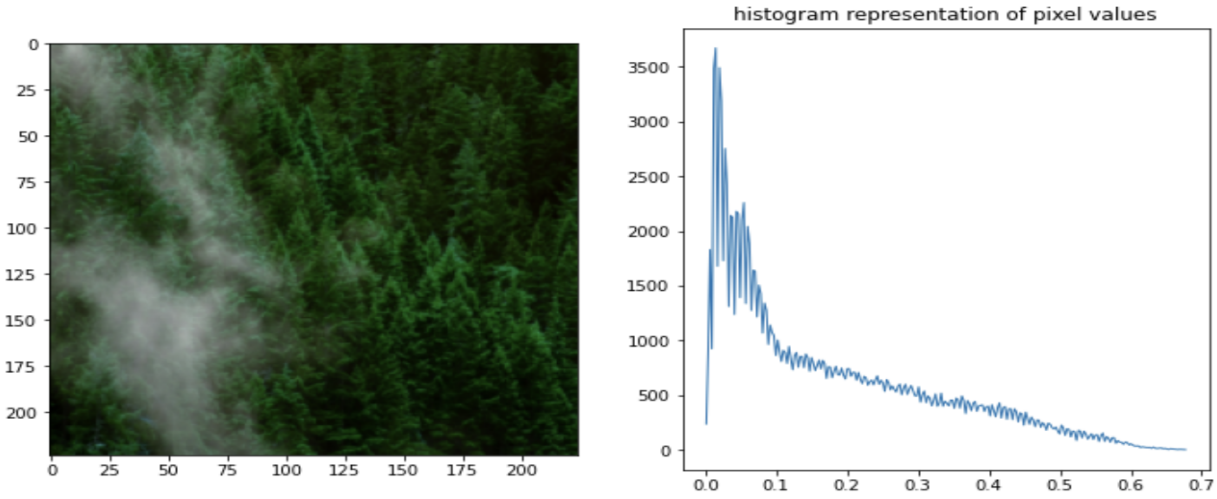


Fig: Histogram for start fire

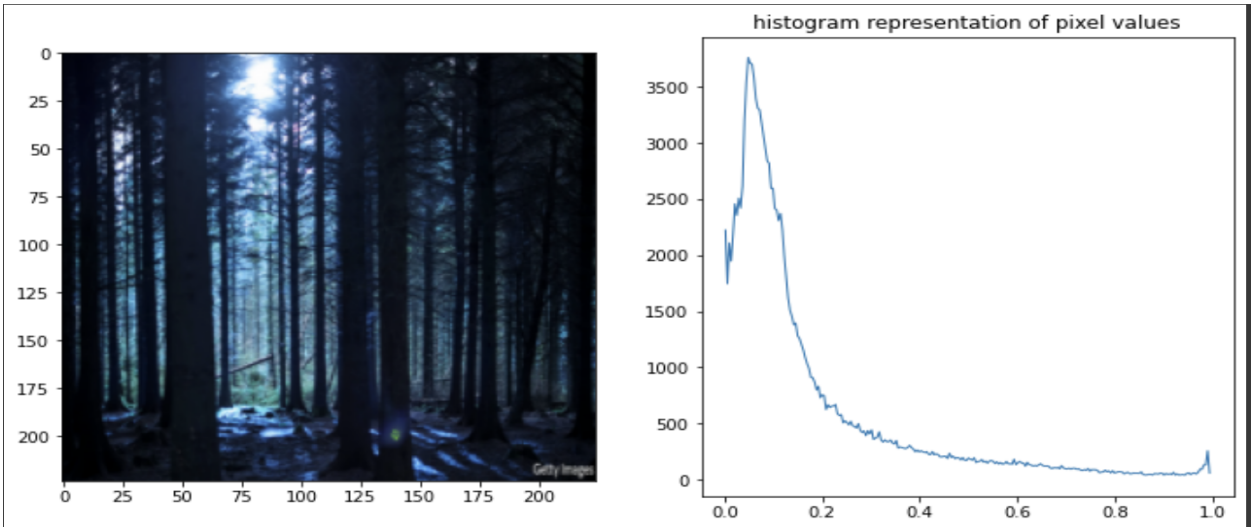


Fig: Histogram for no fire

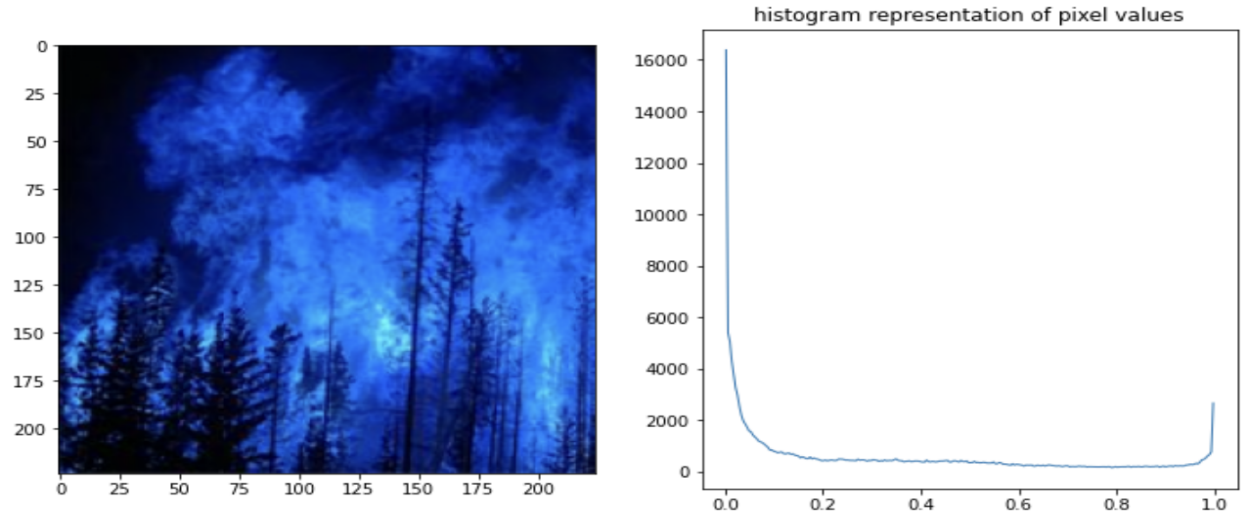


Fig: Histogram for fire

The histogram of the image samples was examined to determine whether the images in each of the three classes had distinguishing pixel properties that would aid the model in differentiating between them. However, it is evident that the histogram cannot

provide sufficient detail on its own and may further worsen confusion if there is any uncertainty. So, in order to find pertinent features, we employ feature engineering.

IMPLEMENTATION

ALGORITHM

STEP 1 : Prepare Input dataset.

STEP 2 : Process image. Gray scale. Re-scale.

STEP 3 : Initiate feature extraction on the input images.

STEP 3 : Filter noise from the existing image

STEP 4 : Segment image into parts that distinct one element of image from another.

STEP 5 : Apply color transformation techniques on the raw images to highlight images with fire.

STEP 6 : Apply thresholding for further image segmentation.

STEP 7 : Detect edges in the images to identify the edge characteristic of fire in the image.

STEP 8 : Detect keypoints to find shifts in pixel intensities.

STEP 9 : Locate the corners to help in inferring image features.

STEP 10 : Create a combination of features that seem fit for the models to find correct patterns.

STEP 11 : Train the pre trained CNN model with the data and test its performance.

STEP 12 : Repeat STEP 10 and STEP 11 until satisfactory performance is achieved.

The model that is being used to train the model is inceptionV3 which is a form of convolutional neural network used to train image data in detecting objects. It is a model popularly used for transfer learning where the model is pre-trained and has certain weights learned. The same model can later be used where some layers are frozen with the initially learned weight, and the rest of the layers train with the new data where the new information from the images are learned. This is done to preserve the information which is previously learnt by the network from a relatively large dataset. At the same time, the

unfreezed layers are allowed to train and adapt themselves to features in the data that is being used.

PSEUDOCODE

STEP 1 :- Combining the feature maps

After implementing various permutations and combinations of the extracted features, observing the effect of one feature map on another and a detailed study of the individual features, we stack the feature maps together.

Code:-

```
feature1 = feature1_function(image)
feature2 = feature2_function(image)
feature3 = feature3_function(image)
features = [feature1, feature2, feature]

def feature_stack(features):
        final_feature_map = features[0] + features[1]..... + features[n]
        return final_feature_map
```

STEP 2 :- Training the CNN architecture

Once the features are compiled together, we treat this stacked feature map as an image consisting of information from n-channels (n being the number of features map stacked together).

Code :-

```
image = feature_stack(features)
```

We then import a pre-trained CNN architecture, i.e. Inception V3 and freeze all its layers except the input and a few output layers.

Code :-

```
import InceptionV3
model = InceptionV3
model.freeze(1)
model.freeze(-1)
model.freeze(-2)
.
.
model.freeze(-n)
```

Now that the architecture of our Neural Network is ready, we pass our stacked feature maps into Inception's architecture with an open input and few of the output layers and train our model for 50 epochs.

Code:-

```
model.fit(image, epochs = 50)
```

STEP 3 :- Hyperparameter tuning

As most of the network hyperparameters like number of neurons, number of dense layers in the network, etc. are fixed, the only hyperparameter we fine-tune here is the number of epochs. We tune the number of epochs in order to avoid overfitting our model.

In order to overcome the issue of overfitting, we use a pre-defined method in keras called early_stopping. The method observes loss on validation set and stops training if the loss increases for consecutive epochs.

Code :-

```
Early_stopping = callbacks(monitor = validation loss, patience = 10 epochs)
model.fit(image, epochs = 50, early_stopping)
```

Explanation of implementation

Implementation has been done in multiple steps. Starting with research where each of us looked into papers that worked on forest fires. We inferred that transfer learning on the Inception V3 model would eventually give better results. Next we proceed with data analysis, here we have gone through images across different classes in the dataset and inferred that certain feature extraction techniques would improve the model at the end result.

We then worked on feature extraction techniques like edge detection, image segmentation, Noise filtering, Color Transformation ORB, Censure and corner detection e.t.c. After integrating and applying them to the dataset, we proceeded to build a basic transfer learning model on InceptionV3 with 50 epochs to test our architecture and test the model's performance in this first phase of testing.

In the second phase, we then experimented with feature engineering in order to achieve the optimum performance from the model. Several combinations of the extracted features were used to train the model separately. Based on the maximum performance recorded by the feature combination the final set of features were selected which for us was a combination of all the features, that was ; image segmentation, edge detection, key points detection, thresholding, censure detection, noise filtering and color transformation.

Further the model was tuned with varying the hyperparameters such as batch size, epocha, number of units. The recorded accuracy and results for other performance metrics like ROC curve, confusion Matrix will be discussed further in the next section. These results were achieved after testing the integrated application with all possible scenarios of video footage captured during forest fires.

RESULTS

After trying out multiple combinations of features, the following two combinations have proven to be most effective.

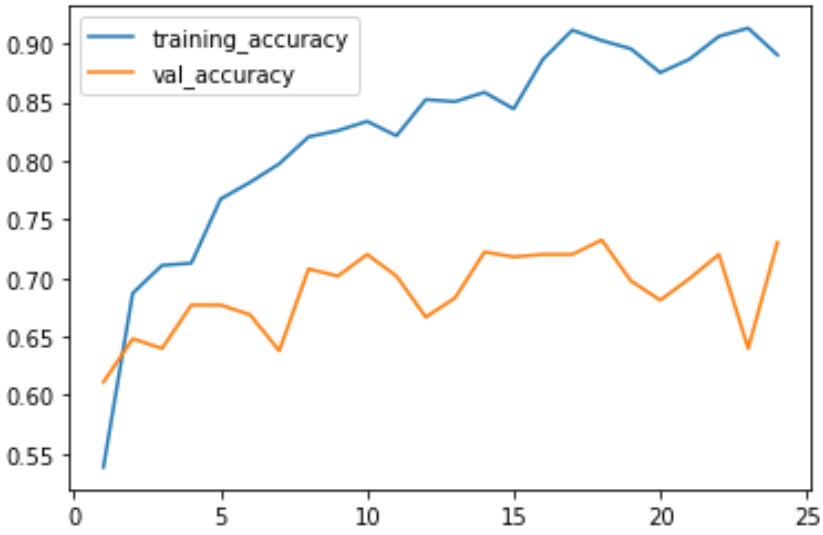
After a good amount of research on the suitable metrics for the context of this project, the metrics we will be using to evaluate our models are Accuracy Score, Confusion Matrix, Precision-Recall-F1 Score.

1. Gaussian Noise Filter + Segmentation + LAB color space :-

Herein, we stack the feature maps extracted from the Gaussian Noise filter, Segmentation filter and color components from LAB space feature extraction techniques and train our CNN architecture using this map.

Following are the results obtained :-

- Loss and Accuracy curves



Fiq. Accuracy curve for combination 1

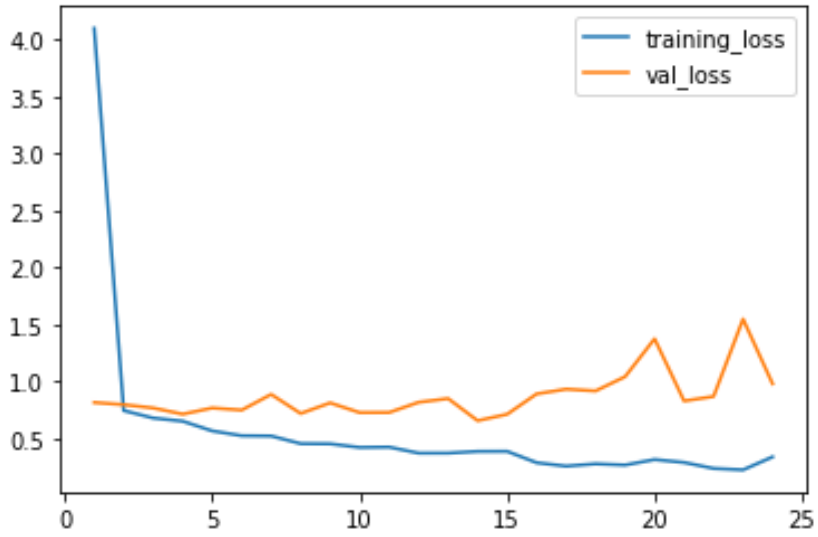


Fig. Loss curve for combination1

As usual, we can observe that the validation accuracy goes increasing as the network trains in identifying fire from the images.

- Accuracy Score :-

  The optimum accuracy score obtained for this combination after fine tuning the number of epochs, on an unseen test data set is 77.23%

- Confusion Matrix :-

  Confusion Matrix for the three target labels in the context of this project is as follows,

	Fire	No Fire	Start Fire
Fire	286	30	10
No Fire	79	224	19
Start Fire	11	9	26

As we can see from the above matrix, from a dataset of 694, most of the test instances are classified correctly. 286 of the fire images were classified as fire, 224 of the non-fire forest images were classified under the no-fire category, while the

remaining 26 of the images which displayed initial stages of fire, were classified as start fire.

- Precision and Recall:-

  Precision and Recall values estimated for the model trained using this combination of features are 70% and 71% respectively, which is good enough compared to other combinations.

2. Edge Detection Sobel Filters + Gaussian Noise Filter + ORB Keypoins :-

The next set of features we stack up are edges in the images using the sobel filters for horizontal and vertical edge detection in the images, gaussian noise filters to get rid of any noisy pixels in the images and smoothen them. The last feature we pick is the ORB keypoints detected in the images as these key points may carry vital information about the nature of fire or forest vegetation in the images.

Following results are obtained :-

- Loss and Accuracy curves

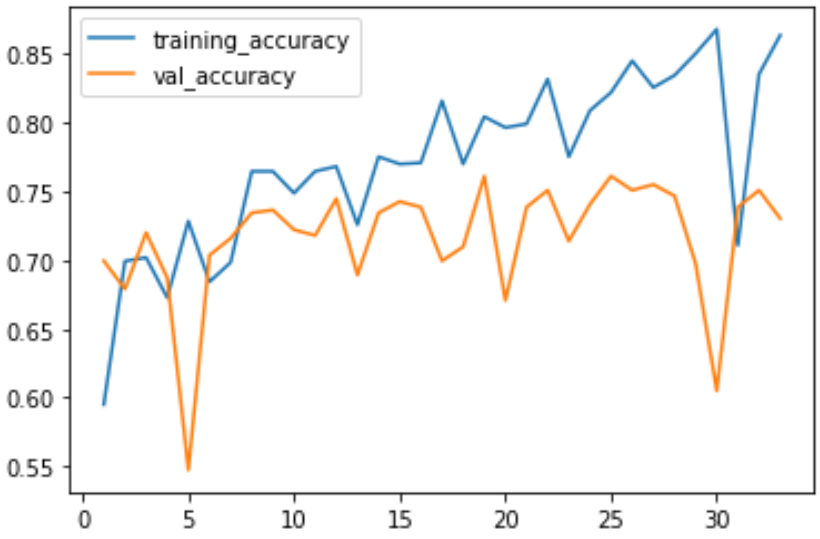


Fig. Accuracy curve for combination 2

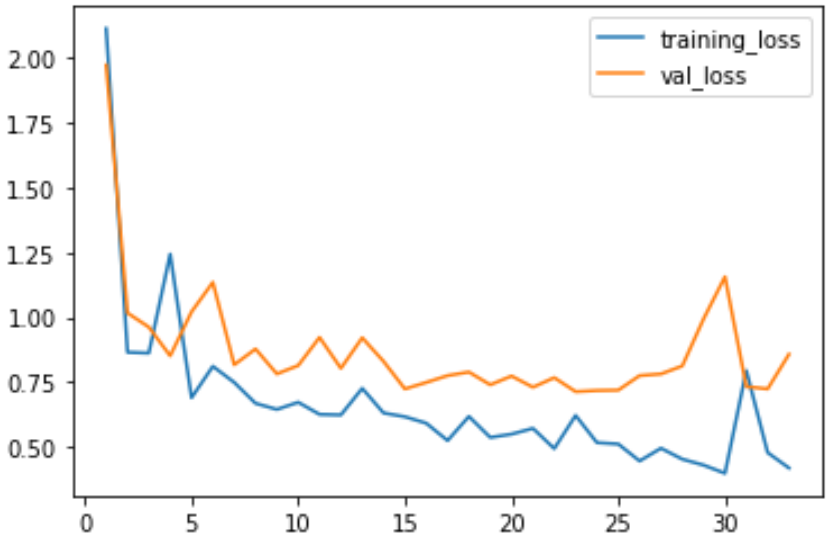


Fig. Loss curve for combination2

We can observe the sharp transitions in accuracy and loss for both the training and validation curves. Accuracy goes on significantly increasing while the loss gradually decreases.

- Accuracy Score :-

  We obtained an accuracy score of 77% using this model to make predictions on the same unseen data set.

- Confusion Matrix :-

  Following confusion matrix is obtained,

	Fire	No Fire	Start Fire
Fire	283	43	0
No Fire	70	252	0
Start Fire	24	22	0

From the above matrix we can say that the model has improved in its performance over the Fire and No Fire classes but has completely failed in identifying the start fire category. One of the reasons for this may be that the features involved in this combination mostly deal with the physical components of the image like shapes

and structures. Out of 694 test instances, 283 of the fire images were classified as fire images and 252 of the non-fire images were classified under the no fire category. The third category, which is the start fire category, has failed miserably.

- Precision and Recall:-

  Precision and Recall values obtained for this model are 51% and 55% which is poor compared to the earlier combination, but holds a vital insight which will be discussed further in this report.

3. All Features combined :-

Observing the results from both set of features, some of the conclusions we can draw are

1. First set of features performs well on all three classes. The precision and recall values for the model convey the same story.
2. Second set of features gives exceptionally good results on the Fire and No Fire classes but fails to predict any of the Start Fire instances.
3. Combination of all the features would be a good option to go ahead with as it will combine the positives from both the models.

Hence, the final model we train combines all the extracted features that are mentioned above in this report.

Following are the results obtained for this final model,

- Loss and Accuracy curves :-

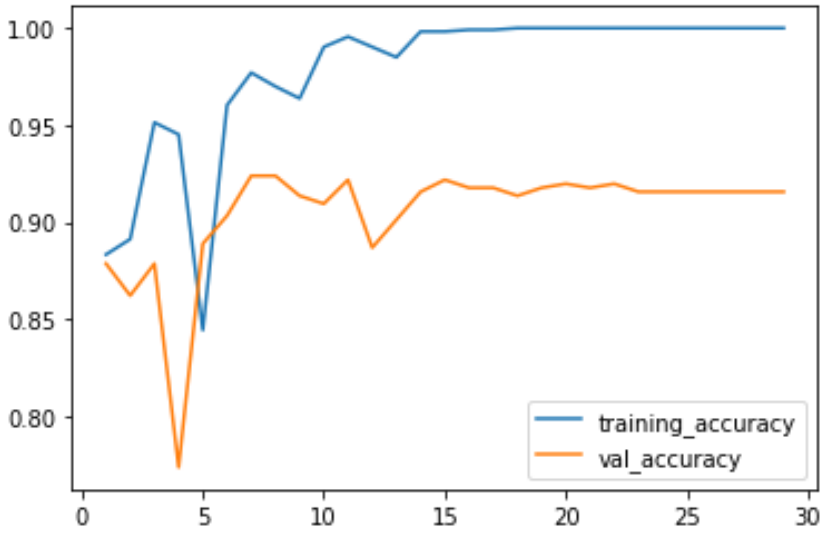


Fig. Accuracy curve

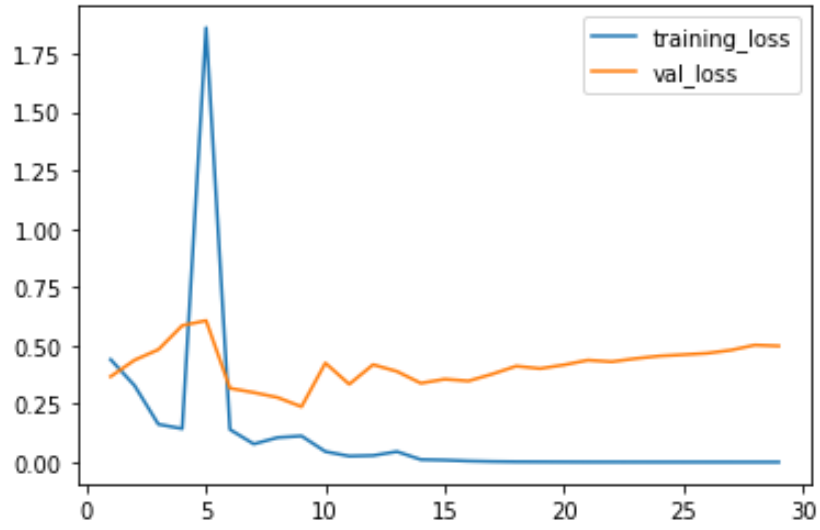


Fig. Loss curve

We can observe from the above curves that there is a significant increase in the accuracy values compared to the previous models.

- Accuracy Score :-

  Based on the predictions of this model on an unseen test dataset, estimated accuracy of the model comes out to be 92%, which supports the hypothesis that

the combination of all the features proves to be more effective when detecting all the fire patterns in images.

- Confusion Matrix :-

  Following is the confusion matrix obtained for this model's predictions,

	Fire	No Fire	Start Fire
Fire	309	16	1
No Fire	20	299	3
Start Fire	16	5	25

  As we can see, the diagonal entries of the above matrix look very promising. Out of 694 test data samples, 309 which consisted of fire were predicted as Fire, 299 of no-fire images were predicted as no-fire and 25 of the images which displayed the initial stages of fire were classified under the Start Fire category.

- Precision and Recall :-

  Precision and Recall values for the model's predictions also show a significant change from the earlier combinations. We achieve a precision value of 90% while a recall value of 81% for this model.

PROJECT MANAGEMENT

IMPLEMENTATION STATUS

Various features are extracted from the images using different techniques. These features are then engineered to work in tandem in order to increase the classification accuracy of the classifier model. A simple CNN model is trained using the extracted features and a small slice of the entire dataset.

Work Completed:

The work done in this increment heavily leans towards finding the right set of features to train the model so that the model can correctly learn the patterns in images with fire in them. The features extracted in the initial increments were formed into a set of varied combinations of features. This was done because not all the features that were extracted can contribute to the learning of the model. In some cases they may just be introduced as noise. Hence, to filter out the unhelpful features multiple feature combinations were experimented with. As a result we achieved a model that was able to identify forests with or without fire with accuracy of 92%, with a combination of all the features that were extracted during the process.

Task	Description	Person
Model training phase I	Combining all the extracted features and training the model. Testing their accuracy with the test set	Shridhar
Model training phase II	Combining the features Gaussian Noise Filter + Segmentation + LAB color space to train the model and test the results and performance from the model to decide if the set of features fit well than the others	Shreeti

Model training Phase III	Combining the features Edge Detection Sobel Filters + Gaussian Noise Filter + ORB Keypoints to train the model and test the results and performance from the model to decide if the set of features fit well than the others	Shreeti
Tuning	Model tuning by changing the hyperparameters such as epochs, batch size and number of units in each layer to achieve better performance	Shridhar
Testing model	Testing the model performance on unseen data after each tuning done on the model	Rana
Accuracy/Performance Measures	Accuracy measures such as confusion matrix, ROC curves, Precision, Recall are to be calculated and visualized	Saja

Limitation and Future Scope	The task is to describe the tradeoffs or outliers in this project and what can to be done to further improve it	Rana and Saja
Script for Video Segmentation	The model is being trained with images but the model in application uses real time video feed. This task involves writing a script to take video input and segment them into images frame by frame.	Priya
Integration and testing on real time video	Here All the modules will be integrated and testing would be done using various real time video footages of forest fire in different scenarios and would discuss on effectiveness of the application	Priya
Documentation and Video compilation	Compilation of project implementation and results in written document and videos	All team members

CONTRIBUTIONS

PRIYA ENUGANTI: Segmenting the videos containing forest fire, no fire and start fire classes and arranging the dataset accordingly. Testing the robustness of application on various different scenarios in forests. Providing valuable feedback in order to extract the features with keeping these scenarios in mind.

SAJA ALKARAWI: Research on the limitations and caveats of training the model with extracted features and on the ways to deal with them. Also, study of the possible future scope relevant to this project.

RANA GHONEIM: Testing the models after every combination of features and shortlisting the final three to go ahead with. Additionally, making the test dataset rich in terms of the class that model performs the worst for.

SHRIDHAR KSHIRSAGAR : Training the model after combining all the features and making predictions on the hold out test data set. Fine-tuning the hyperparameters like number of epochs after observing the loss and accuracy curves in order to stop training and avoid overfitting. Also, deciding on the number of layers from InceptionV3 to be unfreezed.

SHREETI UPRETI : Training model with different combinations of features and comparing the performances and results from each model to pin down a set of features that best fit the model. Combining the features Edge Detection Sobel Filters + Gaussian Noise Filter + ORB Keypoints and Gaussian Noise Filter + Segmentation + LAB color space to train the model and test the results and performance from the model to decide if the set of features fit well than the others. Deciding on the suitable evaluation methods to use for the models and implementing the same.

ISSUES / CONCERNS

In the first increment, it has been stated that the characteristics of images where fire is just starting is just smoke coming out of a forested area. In most cases it is accompanied with scattered fire or the smoke is clearly distinguishable. However, some of these images can be easily mistaken for an image with a fog in it. This issue has been solved to some extent where the heavy fog and lighter smokes are being differentiated by the model. However, heavy fog and even heavier smoke coming from the fire are still being confused with one another. This requires further research which is not limited to the scope of feature engineering.

REFERENCES / BIBLIOGRAPHY

1. https://aryamansharda.medium.com/how-image-edge-detection-works-b759baac01e2
2. https://journals.sagepub.com/doi/full/10.1177/1748302619887689
3. https://www.frontiersin.org/articles/10.3389/fenvs.2022.794028/full
4. http://fs.unm.edu/ImprovedColorEdgeDetection.pdf
5. https://www.geeksforgeeks.org/image-segmentation-using-pythons-scikit-image-module/#:~:text=The%20process%20of%20splitting%20images,image%20from%20its%20integration%20level.
6. https://www.researchgate.net/figure/Flow-chart-of-forest-fire-image-recognition-algorithm-based-on-CNN_fig1_337419415
7. https://www.analyticsvidhya.com/blog/2021/09/image-segmentation-algorithms-with-implementation-in-python/
8. Wu, S., Zhang, L.: Using popular object detection methods for real time forest fire detection. In: IEEE 11th ISCID, vol. 1, pp. 280–284 (2018)