## A. Innovation Introduced in Design

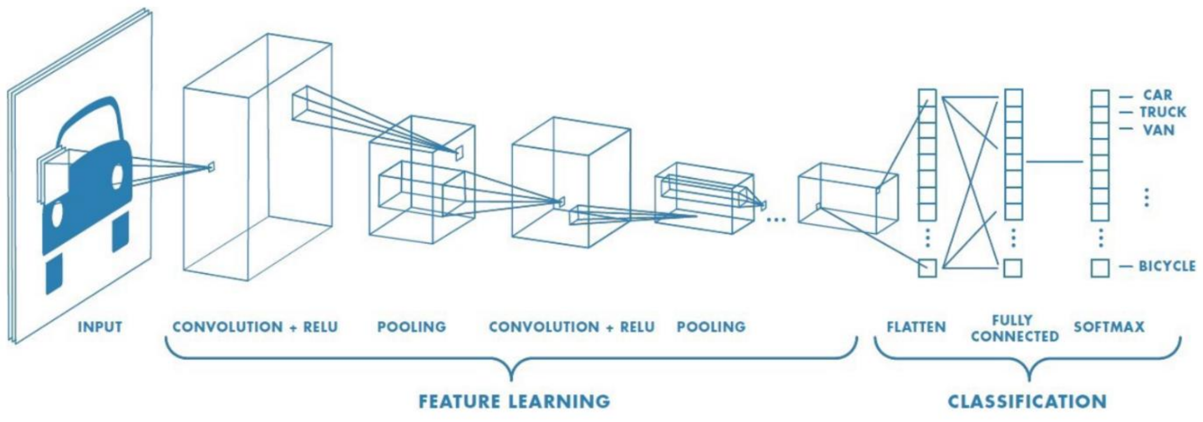
The technologies underlying fire and smoke detection systems play a crucial role in ensuring and delivering optimal performance in modern surveillance environments. For this majority of cities have already installed camera-monitoring systems, this encouraged us to take advantage of the availability of these systems to develop cost-effective vision detection methods. However, this is a complex vision detection task from the perspective of deformations, unusual camera angles and viewpoints, and seasonal changes. The fire sensor detectors detect the fire only when there is huge density of fire flame, this reduces the major damages but what if we have a system that even recognizes that there is fire around, this will cause very little or no damage. For this, we proposed a new method based on a deep learning approach, which uses a convolutional neural network that employs dilated convolutions. The surveillance cameras keep capturing the images, video clips of the region, which then are acted as input to the proposed system and design. It then checks the if the image has been observed with some fire or not and if found rings the alarm and thus the fire management is known in advance and the fire is prevented to avoid any kind of small damage even. In this article, two custom CNN models have been implemented for a cost-effective fire detection CNN architecture for surveillance videos. The first model is a customized basic CNN architecture inspired by AlexNet architecture. We will implement and see its output and limitations and create a customized InceptionV3 model. To balance the efficiency and accuracy, the model is fine-tuned considering the nature of the target problem and fire data. In addition, the method would be designed to be well generalized for unseen data, which offers effective generalization and reduces the number of false alarms.

## B. Ultimate Utility value of the project to the society and/or industry

Pioneering networks of cameras that can search for wildland fire signatures have been in development for some years (High Performance Wireless Research & Education Network—HPWREN cameras and the ALERT Wildfire camera). While these cameras have proven their worth in monitoring fires reported by other means, we have developed a functioning prototype system that can detect smoke from fires usually within 15 min of ignition, while averaging less than one false positive per day per camera. This fire detection system relies on machine learning-based image recognition software and a cloud-based work-flow capable of scanning hundreds of cameras every minute. Ground-based cameras are not going to be able to detect every wildfire, and so we are building a system that combines the best of terrestrial camera based detection with the best approaches to satellite-based detection. Recent advancements in embedded processing have allowed vision-based systems to detect fire using Convolutional Neural Networks during surveillance. The ultimate utility value of the project is to the society is to cause no damage or very little damage due to fire by detection of fire in advance itself. The surveillance cameras keep capturing the images, video clips of the region, which then are acted as input to the proposed system and design. It then checks the if the image has been observed with some fire or not and if found rings the alarm and thus the fire management is known in advance and the fire is prevented to avoid any kind of small damage even. Thus, by the following proposed system the utility of the project is maintained and brings prosperous fire detection method to the field of science and artificial intelligence

## C. New Learning experience gained in the process of the project

Studying various research paper brought up with many new terminologies, with many techniques, with various datasets and their way to extract it, also about the implementation ide and its working. Major part of the project was letting know how does the Convolution neural network is and how does it work. Convolutional neural network (CNN), a class of artificial neural networks that has become dominant in various computer vision tasks, attracted our interest across a variety of domains. The learning of CNN which is used to designed the automatic and adaptive learn spatial hierarchies of features through backpropagation by using multiple building blocks, such as convolution layers, pooling layers, and fully connected layers was an interesting part of learning. Understanding each convolution layer, poling layer, fully connected layers in context with real world scenario was a huge and interesting thing. The steps involved in the convolution layer including its working and various techniques like VGG, Inception V3, alexnet, Resnet brought up a new boost up to our mind and made us learn a deep more about it. In spite of the algorithms or techniques understanding the importing of packages, for performing the convolution techniques was a major challenge. Techniques like image data generator, data augmentation and understanding epochs was a highly gainable knowledge. Letting know the complex artificial diagram, understanding them reading them, making a 3D image understandable to 1D format and its working via the above techniques mentioned. To overcome this problem of limited quantity and limited diversity of data, we generate our own data with the existing data which we have. This methodology of generating our own data is known as data augmentation. Various new generating images techniques could be inclided as zooming, horizontal flipping, rotation, etc. In context to the implementation part, understanding the working of Google Collab and how its importing tasks works, the tesnorflow, keras, shutil, why it is need of training and testing data distribution, the methods involved in making the model fit, making the data augmented was a great learning skill up to the brain. The increasing of efficiency with increase in epochs and number of internal steps in the epochs and the dependence of one to other in code imports was a interesting task for deep learning the context.

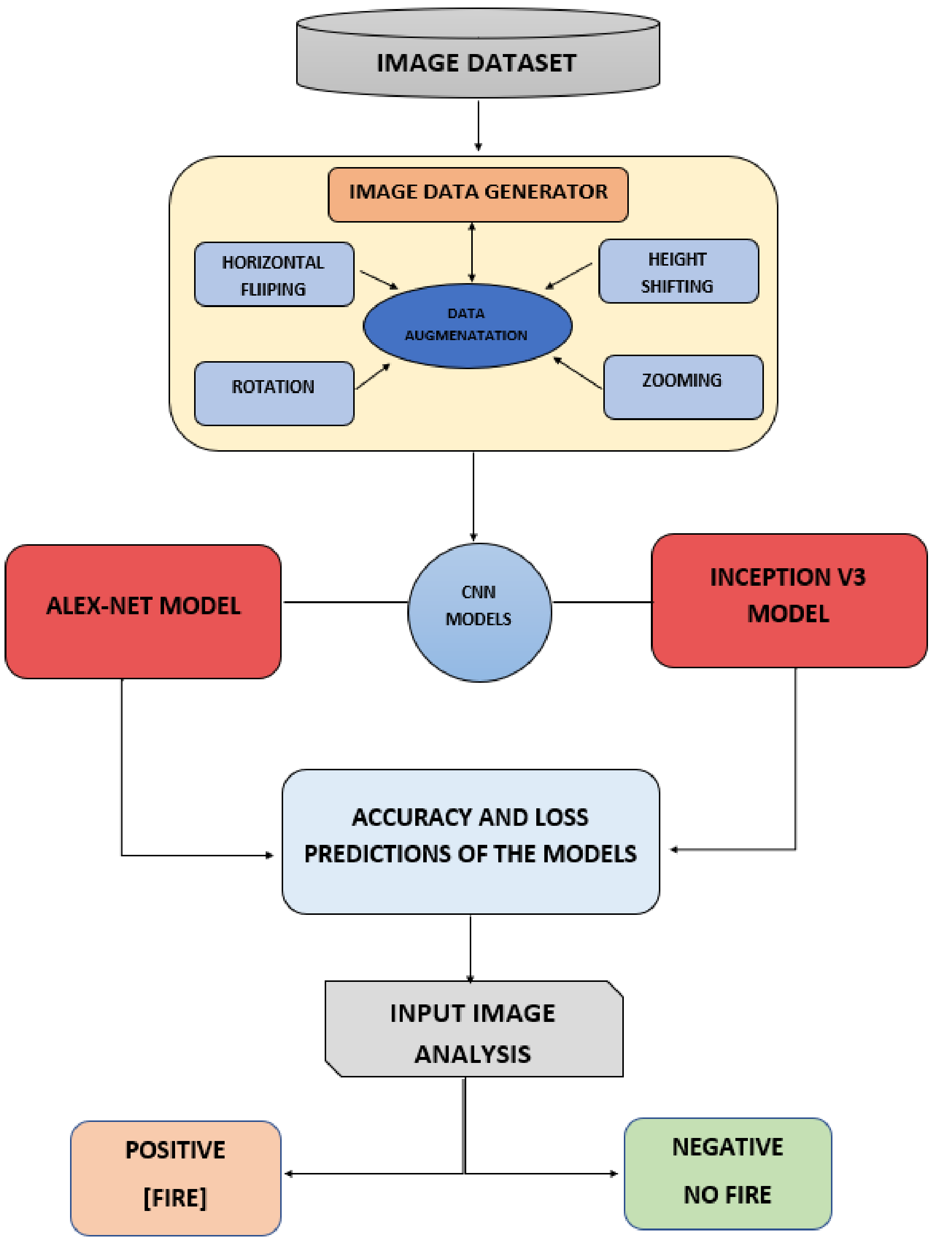


## D. Detailed Model (Architecture) Description

AI architecture is based on the same idea to deal with the complexities in analytics and AI. An AI architect deals about the information and infrastructure we are working in as well as the current technology landscape in order to build architecture that will work and adapt both now and in the future. As such our system architecture is also about the analytic and complexities of the fire detection.

- The architecture starts with extracting the image dataset which includes images of various formats which can be classified as fire and non-fire images.
- The image dataset is then carried out for the image data generator process where each individual image is then made to data value by considering the pixel values of each of it.
- To overcome the problem of limited quantity and limited diversity of data, we generate our own data with the existing data which we have methodology of generating our own data known as data augmentation. The various methods in it includes rotation, horizontal flipping, zooming, height shifting. This brings a huge amount of data to the system.
- The previous two steps are made in a single domain which states that either of these can be done before or after. Performing image data generator before is convenient because flipping matrices is easy for a machine than the image. And also, it reduces the load on the system by decreasing its performance time.

- Once after the pre processing techniques are done, the main part of the system, the working with the CNN models and the proposed techniques performed in the system.
- The CNN models: Alexnet and Inception V3 are then brought upon work. The datasets are divided into two phases of training and testing which is a part of the model architecture and on applying the both the models individually, predictions are made.
- On predicting the accuracy and loss values for both the techniques, the better one is brought up to the next level of the architecture and finally the model is all set for the predicting the results.
- The model is asked for the input data which the user want to test on. Then the prediction of the input and analyzing if it is positive result that is fire or negative result that is no fire. And thus, there the architecture comes to end.

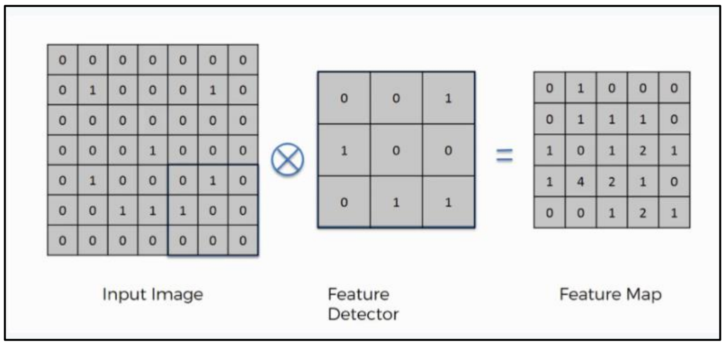


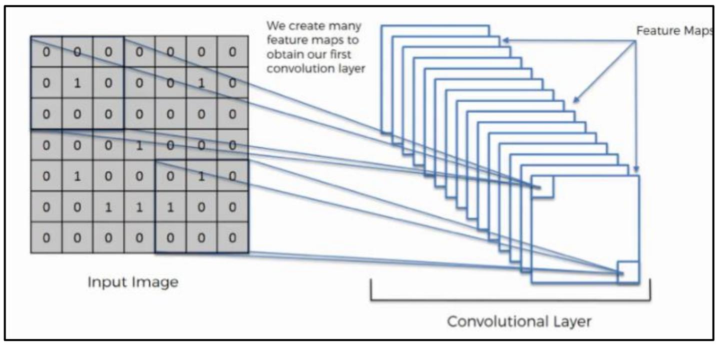
## E. Explanation about complete description

The whole project description is the nutshell of working of the CNN model and the Inception V3 model and Alexnet model. A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. CNNs are powerful image processing, artificial intelligence (AI) that use deep learning to perform both generative and descriptive tasks, often using machine vison that includes image and video recognition, along with recommender systems and natural language processing.

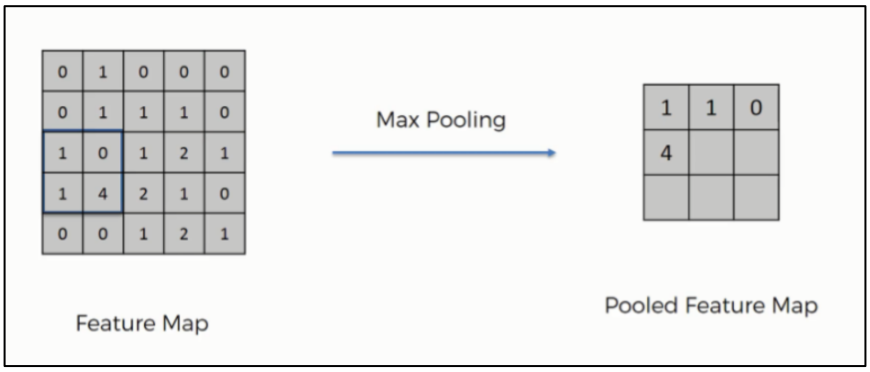
The understanding of working of the CNN model is usually based on the steps and the diagram described as below.

1. **Convolution Operation**: The basic of neural networks filter the feature detectors happen here. Learning the parameters, features maps, how pattern is detected, the layers of detection and findings are mapped out here. The term convolution refers to the mathematical combination of two functions to produce a third function. It merges two sets of information. In the case of a CNN, the convolution is performed on the input data with the use of a filter or kernel to then produce a feature map.

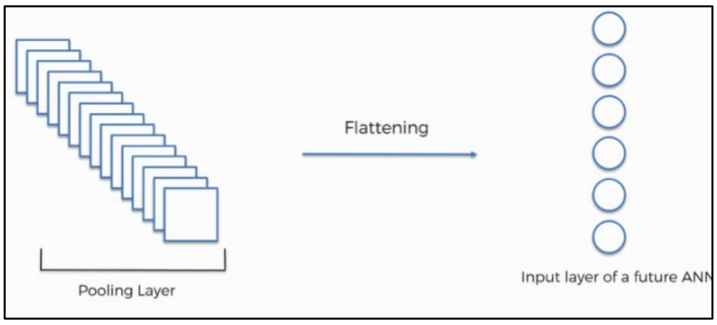


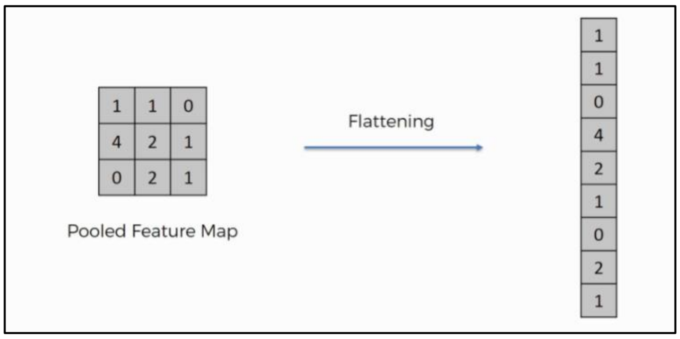

We create many feature maps to obtain our first convolution layer
Feature Maps
Input Image
Convolutional Layer

2. **Pooling:** Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network. The pooling layer summarizes the features present in a region of the feature map generated by a convolution layer.

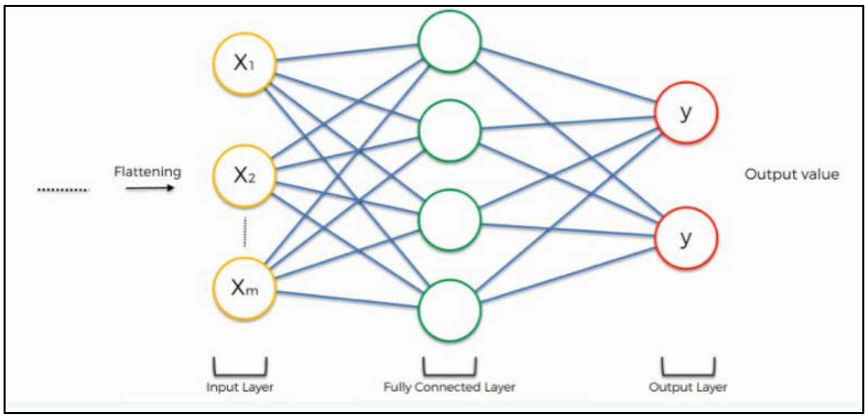


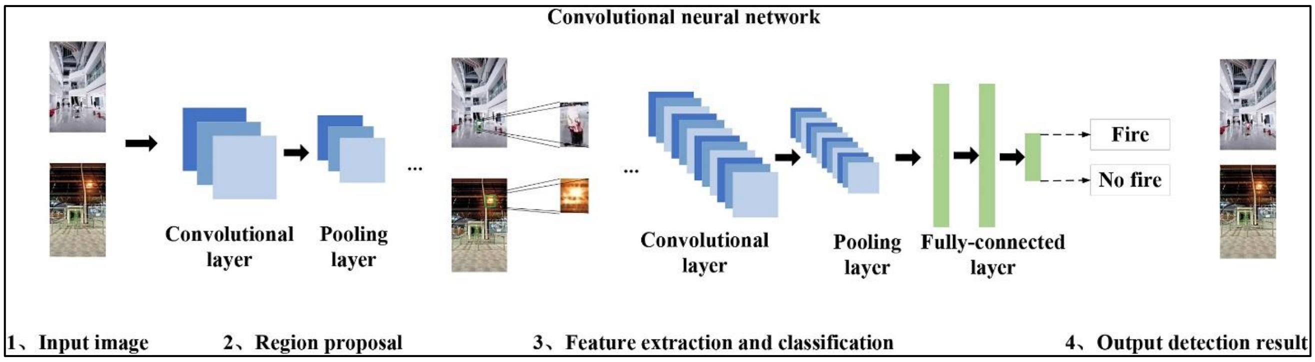
3. **Flattening:** Flattening is converting the data into a 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a fully-connected layer.





4. **Full Connection:** A fully connected neural network consists of a series of fully connected layers that connect every neuron in one layer to every neuron in the other layer. The major advantage of fully connected networks is that they are "structure agnostic" i.e., there are no special assumptions needed to be made about the input.

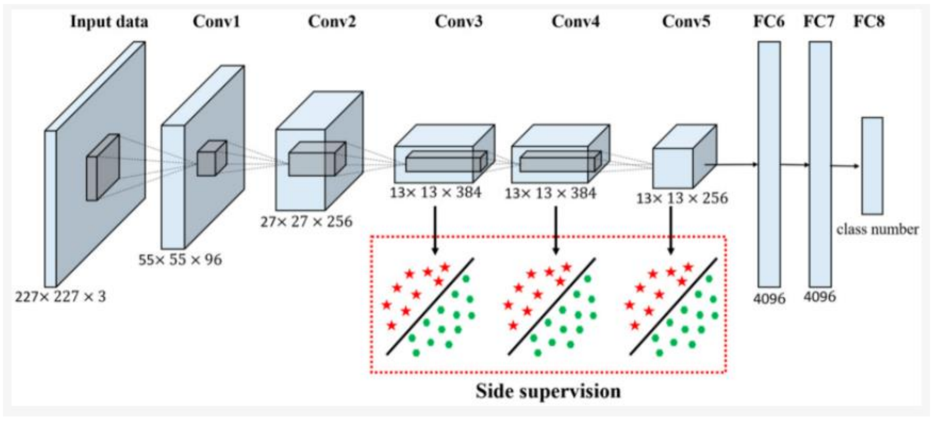




**OVERALL DIAGRAM REPRESENTING THE CNN DIAGRAM**

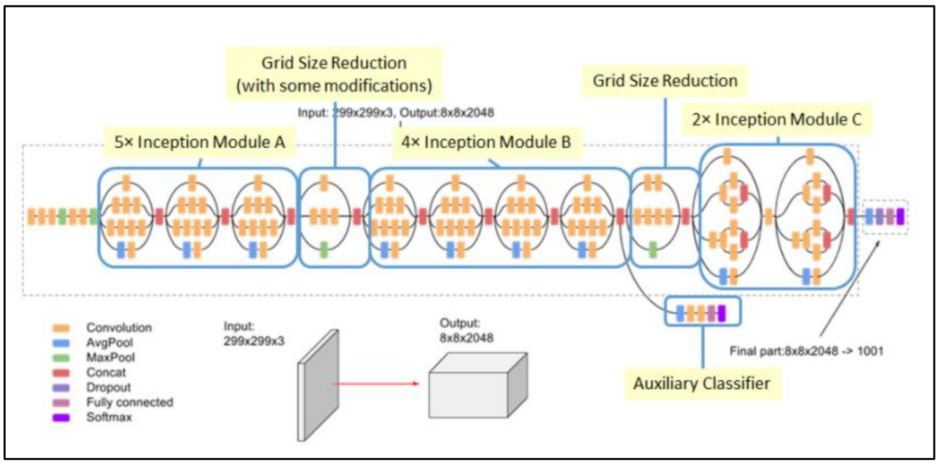
## CUSTOMIZED ALEXNET MODEL

- AlexNet the first convolutional network has eight layers with learnable parameters
- It consists of 5 convolutional layers, 3 max-pooling layers, 2 normalization layers, 2 fully connected layers, and 1 SoftMax layer.
- Each convolutional layer consists of convolutional filters and a nonlinear activation function ReLU.
- The pooling layers are used to perform max pooling. Input size is fixed due to the presence of fully connected layers
- The input size is mentioned at most of the places as 224x224x3 but due to some padding which happens it works out to be 227x227x3.
- AlexNet overall has 60 million parameters



## CUSTOMIZED INCEPTION V3 MODEL

- Inception-v3 is a convolutional neural network that is 48 layers deep.
- It can load a pretrained version of the network trained on more than a million images from a dataset. The pretrained network can classify images into 1000 object categories.
- As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 299-by-299.
- In an Inception v3 model, several techniques for optimizing the network are suggested to loosen the constraints for easier model adaptation.
- The techniques include factorized convolutions, regularization, dimension reduction, and parallelized computations.



## F. IMPLEMENTATION

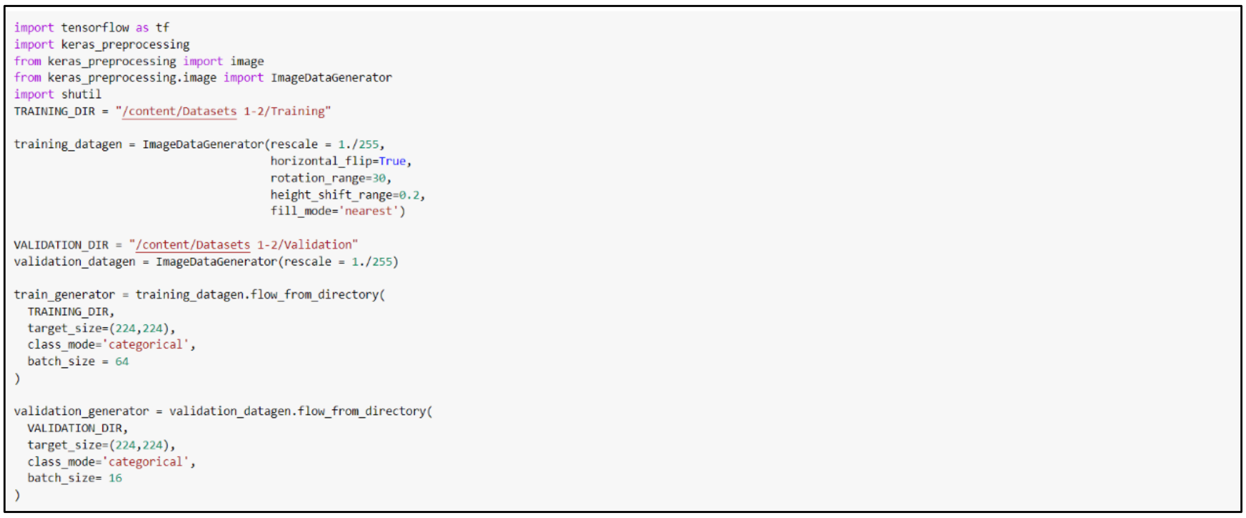
The very first step in implementation requires the importing of dataset to desired directory. Here in our case, we store the dataset into the drive and using the code we import it into the google collab notes.

After the datasets get imported to the desired path, we go for the implementation of the models individually.

```
from google.colab import drive
drive.mount('/content/gdrive')
```

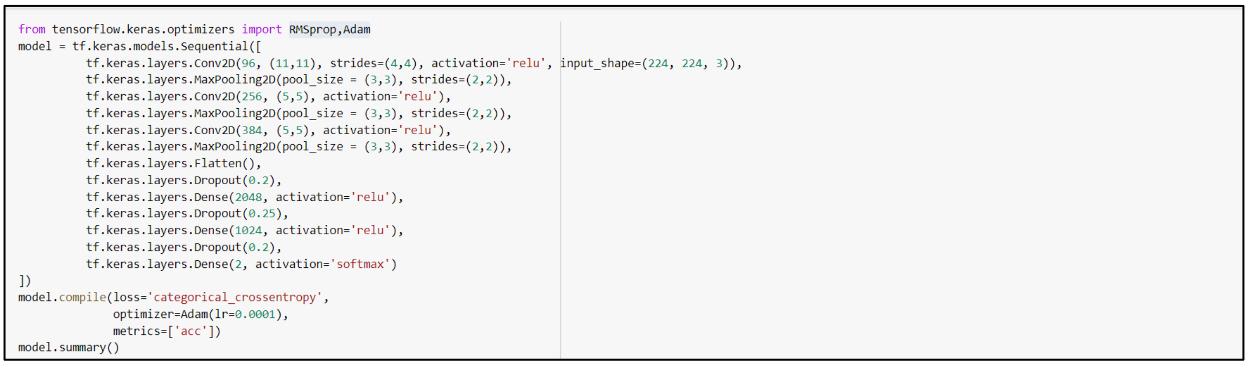
### IMPLEMENTATION OF ALEXNET

The very first panel needs to import the dataset path and apply the image data generator along with the augmentation techniques on both the training and testing data.



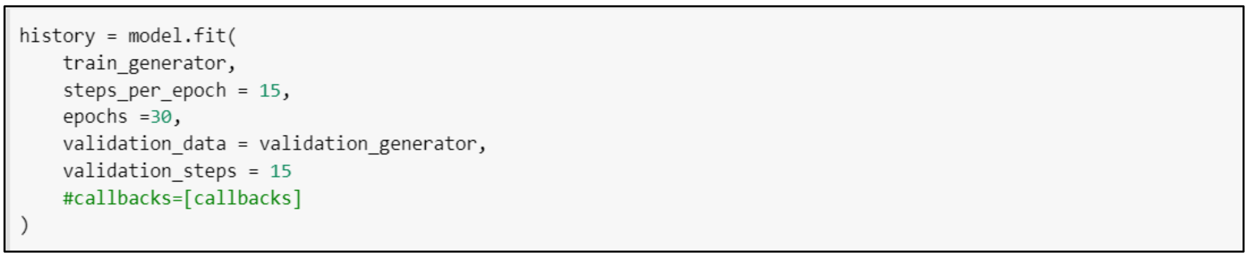
```
import tensorflow as tf
import keras_preprocessing
from keras_preprocessing import image
from keras_preprocessing.image import ImageDataGenerator
import shutil
TRAINING_DIR = "/content/Datasets 1-2/Training"

training_datagen = ImageDataGenerator(rescale = 1./255,
                                      horizontal_flip=True,
                                      rotation_range=30,
                                      height_shift_range=0.2,
                                      fill_mode='nearest')

VALIDATION_DIR = "/content/Datasets 1-2/Validation"
validation_datagen = ImageDataGenerator(rescale = 1./255)

train_generator = training_datagen.flow_from_directory(
  TRAINING_DIR,
  target_size=(224,224),
  class_mode='categorical',
  batch_size = 64
)

validation_generator = validation_datagen.flow_from_directory(
  VALIDATION_DIR,
  target_size=(224,224),
  class_mode='categorical',
  batch_size= 16
)
```

As soon as we get the output of count of the number of images belonging to each class, we go with the convolution layer and poling code as implemented below.



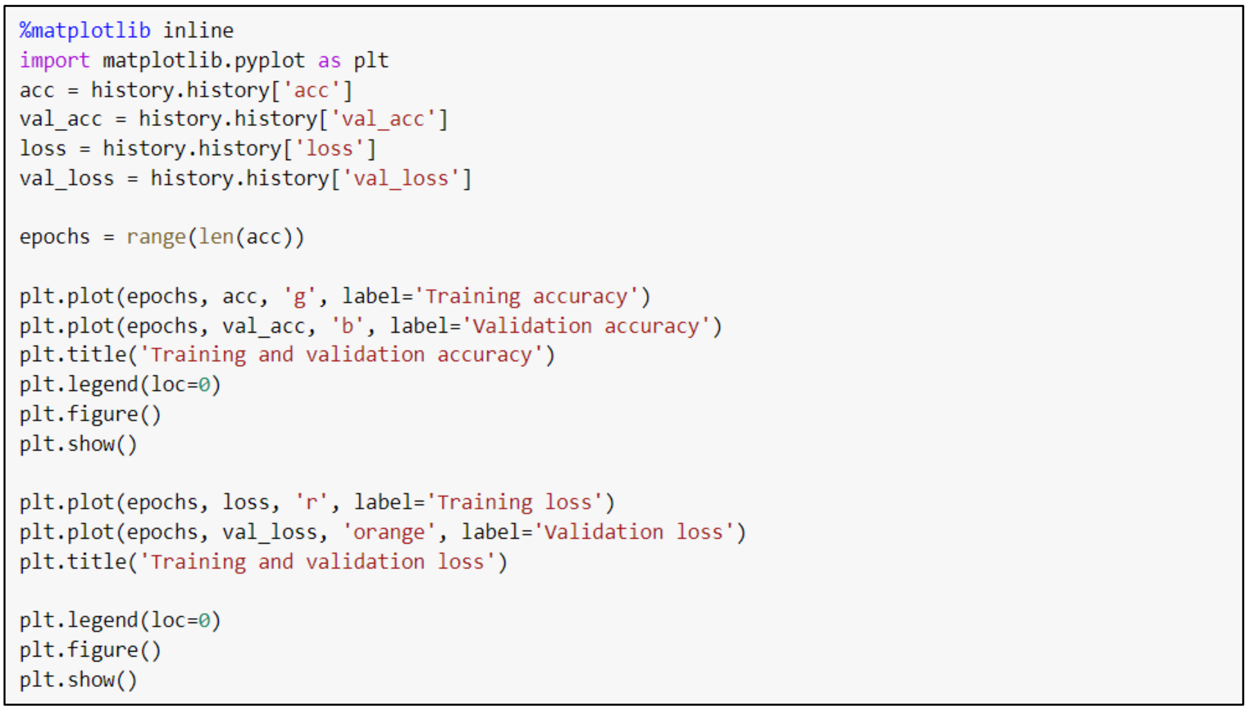
```
from tensorflow.keras.optimizers import RMSprop,Adam
model = tf.keras.models.Sequential([
          tf.keras.layers.Conv2D(96, (11,11), strides=(4,4), activation='relu', input_shape=(224, 224, 3)),
          tf.keras.layers.MaxPooling2D(pool_size = (3,3), strides=(2,2)),
          tf.keras.layers.Conv2D(256, (5,5), activation='relu'),
          tf.keras.layers.MaxPooling2D(pool_size = (3,3), strides=(2,2)),
          tf.keras.layers.Conv2D(384, (5,5), activation='relu'),
          tf.keras.layers.MaxPooling2D(pool_size = (3,3), strides=(2,2)),
          tf.keras.layers.Flatten(),
          tf.keras.layers.Dropout(0.2),
          tf.keras.layers.Dense(2048, activation='relu'),
          tf.keras.layers.Dropout(0.25),
          tf.keras.layers.Dense(1024, activation='relu'),
          tf.keras.layers.Dropout(0.2),
          tf.keras.layers.Dense(2, activation='softmax')
])
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=0.0001),
              metrics=['acc'])
model.summary()
```

Once after the applying of convolution layer and the poling to the dataset frame, a model summary is analyzed and stored for further calculation of epochs including the result giving the total params, trainable params and non-trainable quantity.



```
history = model.fit(
    train_generator,
    steps_per_epoch = 15,
    epochs =30,
    validation_data = validation_generator,
    validation_steps = 15
    #callbacks=[callbacks]
)
```

Applying the model fit to the training and testing data generators, for 30 epochs with 15 steps per each epoch, with spending a time duration of 100 second of time on an average for each epoch we are given out with the result as training accuracy, testing accuracy, training loss, testing loss values for each epoch. Graphs are then made for easy understanding of the result



```
%matplotlib inline
import matplotlib.pyplot as plt
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'g', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend(loc=0)
plt.figure()
plt.show()

plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'orange', label='Validation loss')
plt.title('Training and validation loss')

plt.legend(loc=0)
plt.figure()
plt.show()
```

After understanding the whole model of the Alexnet its graphs and result we move ahead for the implementation of the Inception V3 model and get the result out for further comparison and discussion.

## G. RESULTS AND DISCUSSIONS

In this section of the project involved the discussion on the classification models and outcomes from different perspectives. As stated above first, we read about the different deep learning algorithms such as ResNet, VGG, Inception V3, Alexnet, Deep Neural Network dataset on full features. We gathered a lot of information, analyzed the pros and cons of the models, studied on the theoretical accuracy and loss values of all the models and choose pair of models which were analyzed as best models among all on the basis of theoretical study. This gives the first part of the theoretical study result and discussions.

In the second phase, we used feature selection algorithm which we got from the analyzed theoretical study and choose Alexnet and Inception V3 model for important features selection. Among these two we performed the experimental value detection of testing accuracy, training accuracy, training loss accuracy, testing loss accuracy and compared the both models using the graphs like line and bar, table format.

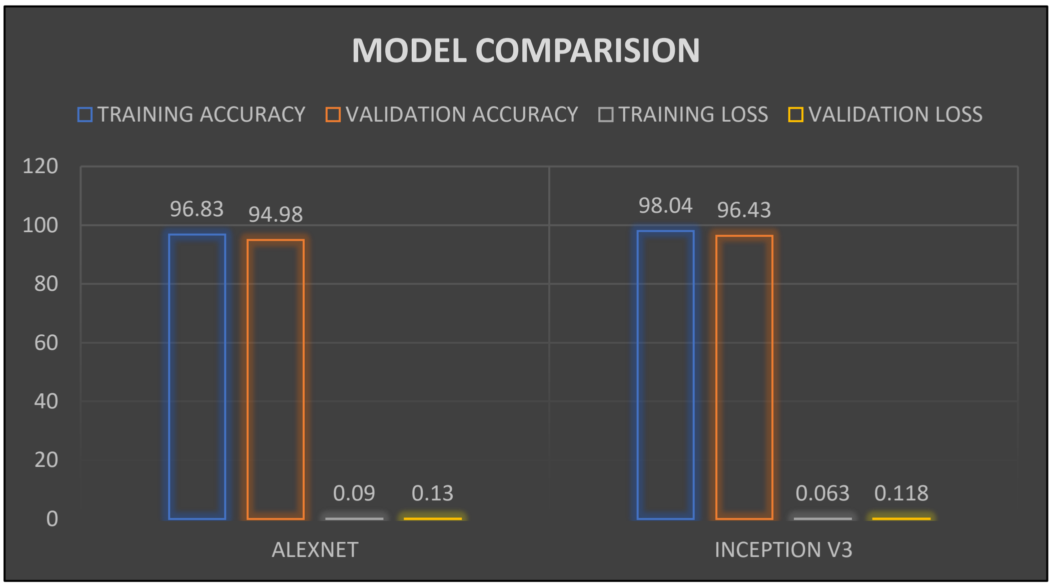
| **Inception V3 Model** | |
|---|---|
| **Type** | **Value** |
| **Training accuracy** | 98.04 |
| **Validation accuracy** | 96.43 |
| **Training loss** | 0.063 |
| **Validation loss** | 0.118 |

| **AlexNet Model** | |
|---|---|
| **Type** | **Value** |
| **Training accuracy** | 96.83 |
| **Validation accuracy** | 94.98 |
| **Training loss** | 0.09 |
| **Validation loss** | 0.13 |

Training accuracy means state about the identical images that are used for both training and testing section. Whereas the validation accuracy represents the trained model identification of independent images that were not used in training. More the accuracy value the better is the model.

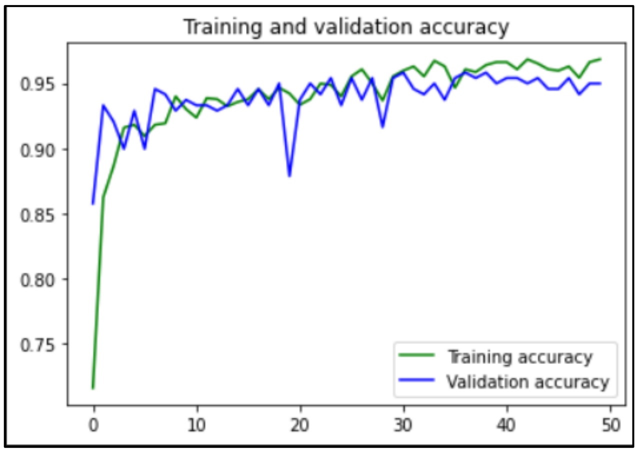
Understanding of training loss is same as training accuracy. Training loss indicates how well the model is fitting the training data, while the validation loss indicates how well the model fits the new data. The less the loss is the more the better it is fit in the model. Since we have been going via various architecture methods like the Inception V3 model and AlexNet model, we majorly compare the training accuracy, validation accuracy, training losses and validation losses. We train the model using the training data and check its performance on both the training and validation sets.

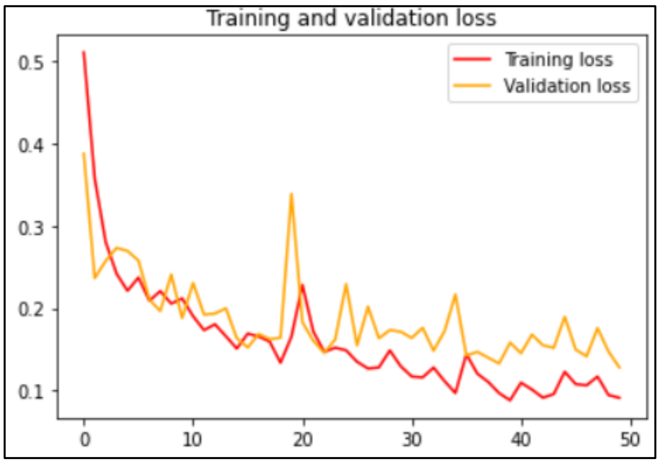
From the above table format and the bar graph model shown below compares both models and accordingly the prediction of which to choose is analyzed. As in our case as the accuracy value are high for Inception v3 model and loss values are less for the same model we chose Inception V3 over and made the system of fire detection to fit with the Inception V3 model.



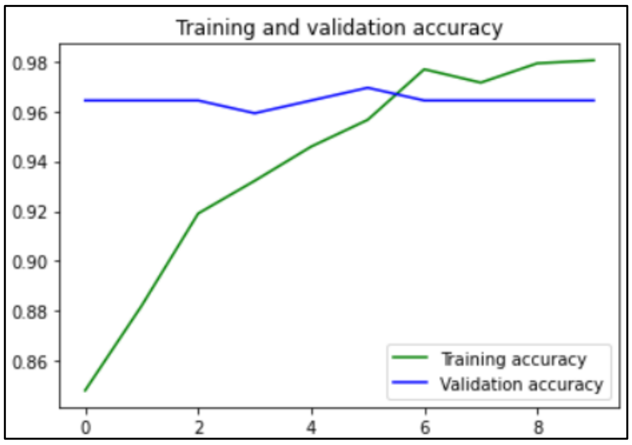
The line graphs show the fluctuating of the accuracy, loss values for each model. Understanding of line graphs here can be made by understanding that more the fluctuations are the more the inaccurate the model is.

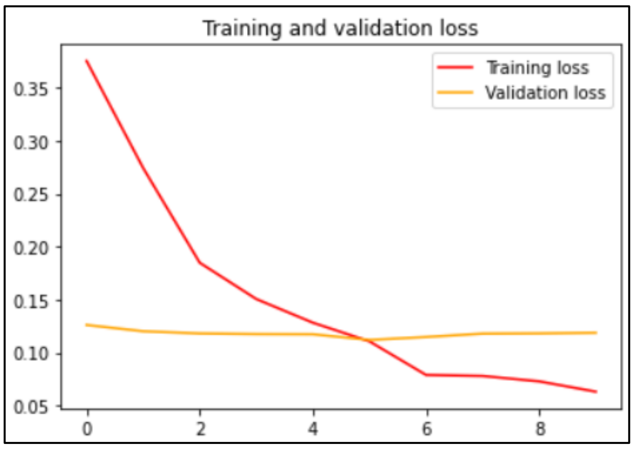
Comparing the line graphs, we observed more fluctuations for the Alexnet model and very straight one for Inception V3 one. Thus here with the graphical study also we could analyze the better model.





**AlexNet Model**





**Inception V3 Model**

In third classifiers, after the performances were checked on selected features and one among these was chosen, the further prediction of input images was analyzed using the selected model. On checking with various input images in the implementation phase we analyzed that the model gives out perfect results. The discussion for the system doesn't need to end, there is always a scope for the improvement. For this future work like syncing the AI domain with the technical domain would be a major task. Looking after new models and applying it for our model could be a better idea for a future and better work and study.

## H. CONCLUSION AND FUTURE WORK

In this embedded project work, we understood that by using smart cameras we can identify various suspicious incidents such as collisions, medical emergencies, and fires. Of such, fire is the most dangerous abnormal occurrence, because failure to control it at an early stage can lead to huge disasters, leading to human, ecological and economic losses. Inspired by the great potential of CNNs, we can detect fire from images or videos at an early stage. It dealt with two custom models for fire detection. Considering the fair fire detection accuracy of the CNN model, it can be of assistance to disaster management teams in managing fire disasters on time, thus preventing huge losses We analyzed that training and validation accuracy of Inception V3 model was better than the AlexNet on and so the losses were also less.