PACKAGES AND LIBRARIES

In [183]:

```
#GENERAL
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
#PATH PROCESS
import os
import os.path
from pathlib import Path
import glob
#IMAGE PROCESS
from PIL import Image
from keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import cv2
from keras.applications.vgg16 import preprocess_input, decode_predictions
#SCALER & TRANSFORMATION
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split
from keras import regularizers
from sklearn.preprocessing import LabelEncoder
#ACCURACY CONTROL
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report, roc_auc_scor
e, roc_curve
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.metrics import mean_squared_error, r2_score
#OPTIMIZER
from keras.optimizers import RMSprop,Adam,Optimizer,Optimizer
#MODEL LAYERS
from tensorflow.keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, BatchNormalization,MaxPoolin
g2D,BatchNormalization,\
                         Permute, TimeDistributed, Bidirectional,GRU, SimpleRNN, LSTM, GlobalAvera
gePooling2D, SeparableConv2D
from keras import models
from keras import layers
import tensorflow as tf
from keras.applications import VGG16,VGG19,inception_v3
from keras import backend as K
from keras.utils import plot_model
#SKLEARN CLASSIFIER
from xgboost import XGBClassifier, XGBRegressor
from lightgbm import LGBMClassifier, LGBMRegressor
from catboost import CatBoostClassifier, CatBoostRegressor
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
```

```
from sklearn.neural_network import MLPClassifier, MLPRegressor
from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.cross_decomposition import PLSRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import RidgeCV
from sklearn.linear_model import Lasso
from sklearn.linear_model import LassoCV
from sklearn.linear_model import ElasticNet
from sklearn.linear_model import ElasticNetCV
#IGNORING WARNINGS
from warnings import filterwarnings
filterwarnings("ignore",category=DeprecationWarning)
filterwarnings("ignore", category=FutureWarning)
filterwarnings("ignore", category=UserWarning)
```

PATH & LABEL PROCESS

MAIN PATH

In [2]:
```
Fire_Dataset_Path = Path("../input/fire-dataset/fire_dataset")
```

PATH PROCESS

In [3]:
```
PNG_Path = list(Fire_Dataset_Path.glob(r"*/*.png"))
```

LABEL PROCESS

In [4]:
```
PNG_Labels = list(map(lambda x: os.path.split(os.path.split(x)[0])[1],PNG_Path))
```

In [5]:
```
print("FIRE: ", PNG_Labels.count("fire_images"))
print("NO_FIRE: ", PNG_Labels.count("non_fire_images"))
```

```
FIRE:  755
NO_FIRE:  244
```

TRANSFORMATION TO SERIES STRUCTURE

In [6]:
```
PNG_Path_Series = pd.Series(PNG_Path,name="PNG").astype(str)
PNG_Labels_Series = pd.Series(PNG_Labels,name="CATEGORY")
```

In [7]:
```
print(PNG_Path_Series)
```

```
0      ../input/fire-dataset/fire_dataset/non_fire_im...
1      ../input/fire-dataset/fire_dataset/non_fire_im...
2      ../input/fire-dataset/fire_dataset/non_fire_im...
3      ../input/fire-dataset/fire_dataset/non_fire_im...
4      ../input/fire-dataset/fire_dataset/non_fire_im...
                           ...
994    ../input/fire-dataset/fire_dataset/fire_images...
995    ../input/fire-dataset/fire_dataset/fire_images...
996    ../input/fire-dataset/fire_dataset/fire_images...
997    ../input/fire-dataset/fire_dataset/fire_images...
998    ../input/fire-dataset/fire_dataset/fire_images...
Name: PNG, Length: 999, dtype: object
```

In [8]:
```
print(PNG_Labels_Series)
```

```
0      non_fire_images
1      non_fire_images
2      non_fire_images
3      non_fire_images
4      non_fire_images
           ...
994        fire_images
995        fire_images
996        fire_images
997        fire_images
998        fire_images
Name: CATEGORY, Length: 999, dtype: object
```

In [9]:
```
PNG_Labels_Series.replace({"non_fire_images":"NO_FIRE","fire_images":"FIRE"},inplace=True)
```

In [10]:
```
print(PNG_Labels_Series)
```

```
0      NO_FIRE
1      NO_FIRE
2      NO_FIRE
3      NO_FIRE
4      NO_FIRE
        ...
994       FIRE
995       FIRE
996       FIRE
997       FIRE
998       FIRE
Name: CATEGORY, Length: 999, dtype: object
```

TRANSFORMATION TO DATAFRAME STRUCTURE

In [11]:
```
Main_Train_Data = pd.concat([PNG_Path_Series,PNG_Labels_Series],axis=1)
```

In [12]:
```
print(Main_Train_Data.head(-1))
```

```
                                                 PNG CATEGORY
0    ../input/fire-dataset/fire_dataset/non_fire_im...  NO_FIRE
1    ../input/fire-dataset/fire_dataset/non_fire_im...  NO_FIRE
2    ../input/fire-dataset/fire_dataset/non_fire_im...  NO_FIRE
3    ../input/fire-dataset/fire_dataset/non_fire_im...  NO_FIRE
4    ../input/fire-dataset/fire_dataset/non_fire_im...  NO_FIRE
..                                                 ...      ...
993  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
994  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
995  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
996  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
997  ../input/fire-dataset/fire_dataset/fire_images...     FIRE

[998 rows x 2 columns]
```

SHUFFLING

In [13]:
```
Main_Train_Data = Main_Train_Data.sample(frac=1).reset_index(drop=True)
```

In [14]:
```
print(Main_Train_Data.head(-1))
```

```
                                                   PNG CATEGORY
0    ../input/fire-dataset/fire_dataset/non_fire_im...  NO_FIRE
1    ../input/fire-dataset/fire_dataset/non_fire_im...  NO_FIRE
2    ../input/fire-dataset/fire_dataset/fire_images...     FIRE
3    ../input/fire-dataset/fire_dataset/non_fire_im...  NO_FIRE
4    ../input/fire-dataset/fire_dataset/non_fire_im...  NO_FIRE
..                                                 ...      ...
993  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
994  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
995  ../input/fire-dataset/fire_dataset/non_fire_im...  NO_FIRE
996  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
997  ../input/fire-dataset/fire_dataset/fire_images...     FIRE

[998 rows x 2 columns]
```

In [15]:

```
print(Main_Train_Data["PNG"][2])
print(Main_Train_Data["CATEGORY"][2])
print(Main_Train_Data["PNG"][200])
print(Main_Train_Data["CATEGORY"][200])
print(Main_Train_Data["PNG"][45])
print(Main_Train_Data["CATEGORY"][45])
print(Main_Train_Data["PNG"][852])
print(Main_Train_Data["CATEGORY"][852])
```

```
../input/fire-dataset/fire_dataset/fire_images/fire.698.png
FIRE
../input/fire-dataset/fire_dataset/fire_images/fire.548.png
FIRE
../input/fire-dataset/fire_dataset/fire_images/fire.197.png
FIRE
../input/fire-dataset/fire_dataset/fire_images/fire.574.png
FIRE
```

In [17]:

```
remove_PNG = '../input/fire-dataset/fire_dataset/non_fire_images/non_fire.189.png'
Main_Train_Data = Main_Train_Data.loc[~(Main_Train_Data.loc[:,'PNG'] == remove_PNG),:]
```

In [18]:

```
print(Main_Train_Data.loc[Main_Train_Data.loc[:,'PNG'] == remove_PNG,:])
```

```
Empty DataFrame
Columns: [PNG, CATEGORY]
Index: []
```

- We need to remove Non_Fire_189, this PNG is broken

In [19]:
```
print(Main_Train_Data.head(-1))
```

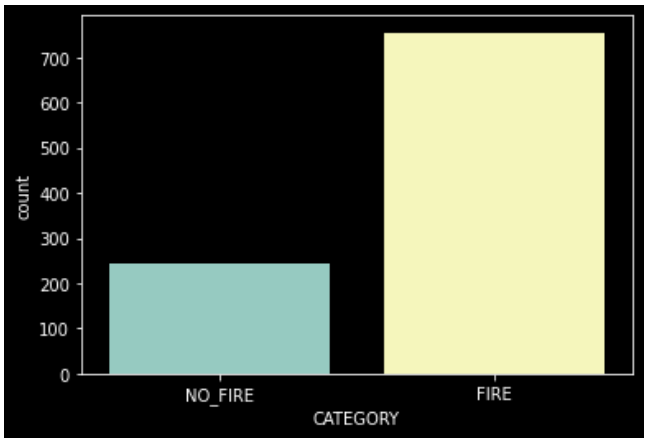
```
                                                   PNG CATEGORY
0    ../input/fire-dataset/fire_dataset/non_fire_im...  NO_FIRE
1    ../input/fire-dataset/fire_dataset/non_fire_im...  NO_FIRE
2    ../input/fire-dataset/fire_dataset/fire_images...     FIRE
3    ../input/fire-dataset/fire_dataset/non_fire_im...  NO_FIRE
4    ../input/fire-dataset/fire_dataset/non_fire_im...  NO_FIRE
..                                                 ...      ...
993  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
994  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
995  ../input/fire-dataset/fire_dataset/non_fire_im...  NO_FIRE
996  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
997  ../input/fire-dataset/fire_dataset/fire_images...     FIRE

[997 rows x 2 columns]
```

VISUALIZATION

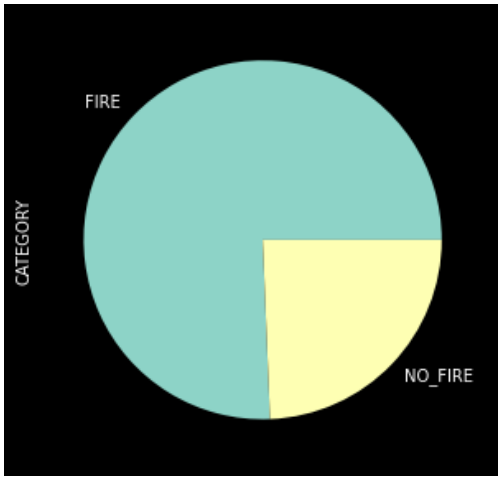
In [20]:
```
plt.style.use("dark_background")
```

GENERAL

In [21]:
```
sns.countplot(Main_Train_Data["CATEGORY"])
plt.show()
```



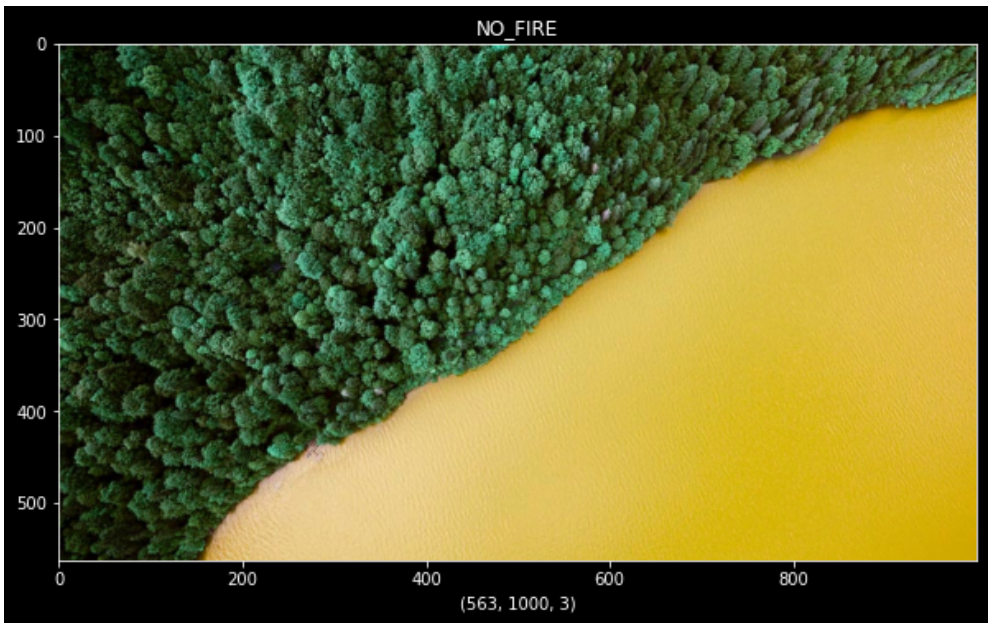
In [22]:
```
Main_Train_Data['CATEGORY'].value_counts().plot.pie(figsize=(5,5))
plt.show()
```



IMAGES

In [23]:
```
figure = plt.figure(figsize=(10,10))
x = cv2.imread(Main_Train_Data["PNG"][0])
plt.imshow(x)
plt.xlabel(x.shape)
plt.title(Main_Train_Data["CATEGORY"][0])
```

Out[23]:
```
Text(0.5, 1.0, 'NO_FIRE')
```



In [24]:
```
figure = plt.figure(figsize=(10,10))
x = cv2.imread(Main_Train_Data["PNG"][993])
plt.imshow(x)
plt.xlabel(x.shape)
plt.title(Main_Train_Data["CATEGORY"][993])
```

Out[24]:

```
Text(0.5, 1.0, 'FIRE')
```

In [25]:

```
figure = plt.figure(figsize=(10,10))
x = cv2.imread(Main_Train_Data["PNG"][20])
plt.imshow(x)
plt.xlabel(x.shape)
plt.title(Main_Train_Data["CATEGORY"][20])
```

Out[25]:

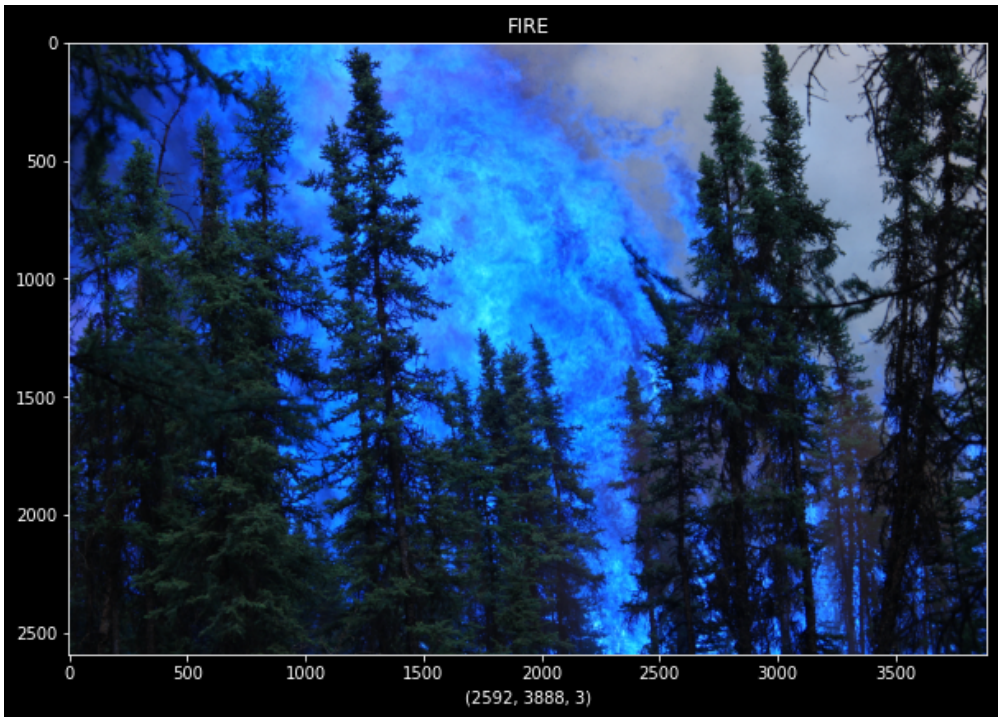
```
Text(0.5, 1.0, 'FIRE')
```

In [26]:

```
figure = plt.figure(figsize=(10,10))
x = cv2.imread(Main_Train_Data["PNG"][48])
plt.imshow(x)
plt.xlabel(x.shape)
plt.title(Main_Train_Data["CATEGORY"][48])
```

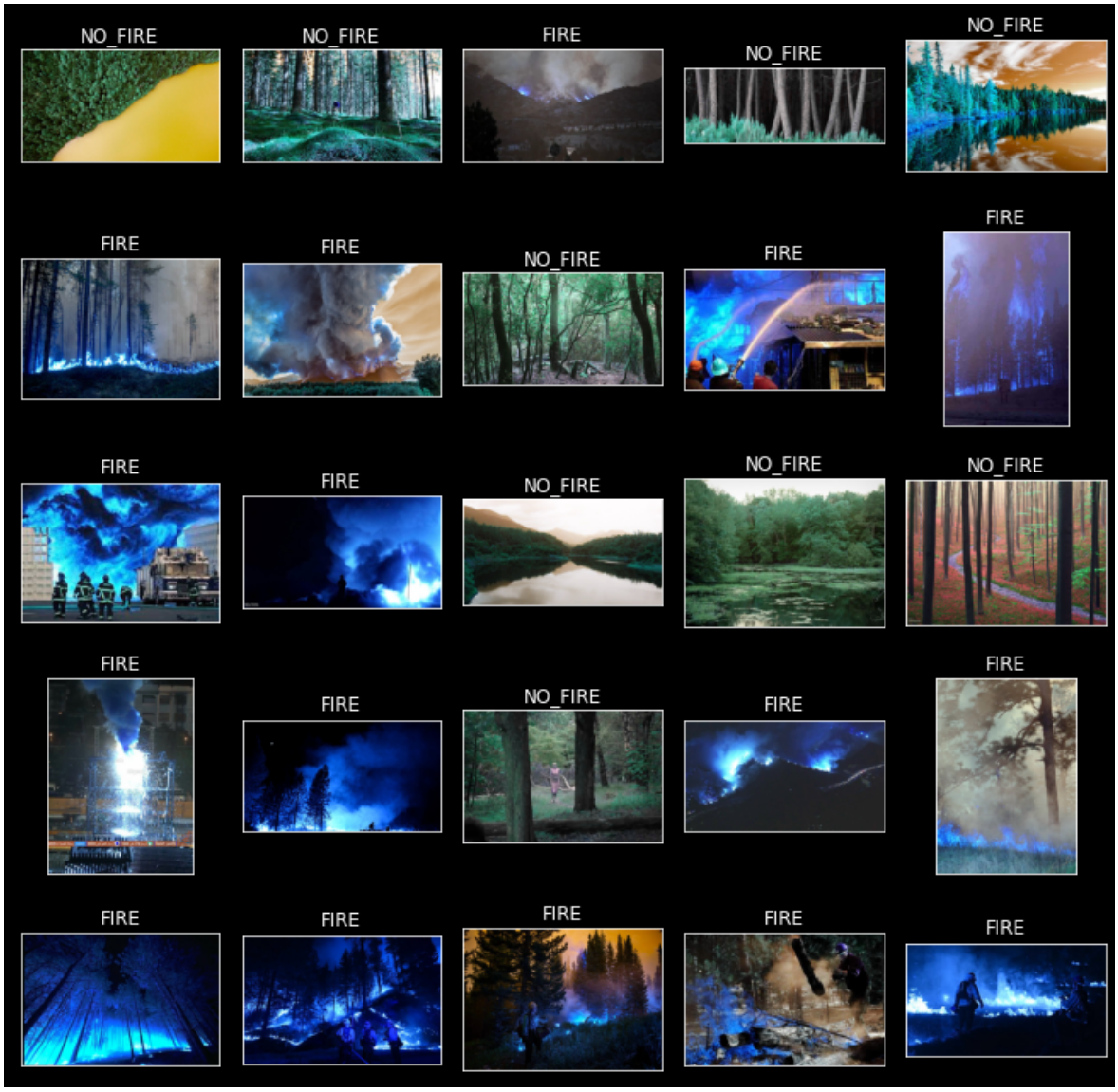
Out[26]:

```
Text(0.5, 1.0, 'FIRE')
```



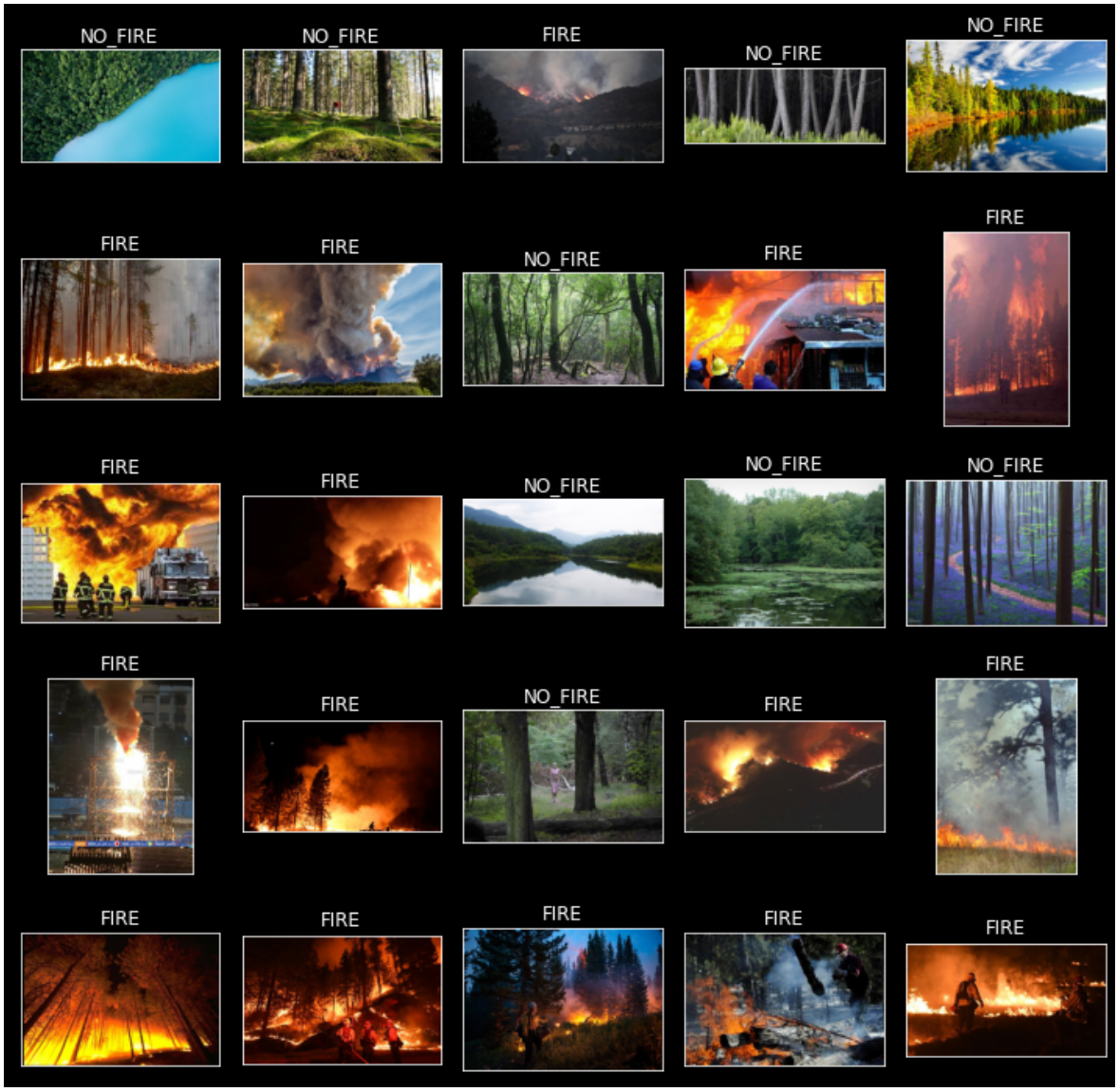
In [27]:

```
fig, axes = plt.subplots(nrows=5,
                         ncols=5,
                         figsize=(10,10),
                         subplot_kw={"xticks":[],"yticks":[]})

for i,ax in enumerate(axes.flat):
    ax.imshow(cv2.imread(Main_Train_Data["PNG"][i]))
    ax.set_title(Main_Train_Data["CATEGORY"][i])
plt.tight_layout()
plt.show()
```



In [28]:

```
fig, axes = plt.subplots(nrows=5,
                         ncols=5,
                         figsize=(10,10),
                         subplot_kw={"xticks":[],"yticks":[]})

for i,ax in enumerate(axes.flat):
    x = cv2.imread(Main_Train_Data["PNG"][i])
    x = cv2.cvtColor(x,cv2.COLOR_RGB2BGR)
    ax.imshow(x)
    ax.set_title(Main_Train_Data["CATEGORY"][i])
plt.tight_layout()
plt.show()
```



DETERMINATION TRAIN AND TEST DATA

IMAGE GENERATOR

In [29]:
```
Train_Generator = ImageDataGenerator(rescale=1./255,
                                     shear_range=0.3,
                                     zoom_range=0.2,
                                     brightness_range=[0.2,0.9],
                                     rotation_range=30,
                                     horizontal_flip=True,
                                     vertical_flip=True,
                                     fill_mode="nearest",
                                     validation_split=0.1)
```

In [30]:
```
Test_Generator = ImageDataGenerator(rescale=1./255)
```

SPLITTING TRAIN AND TEST

In [31]:
```
Train_Data,Test_Data = train_test_split(Main_Train_Data,train_size=0.9,random_state=42,shuffle=Tr
ue)
```

In [32]:
```
print("TRAIN SHAPE: ",Train_Data.shape)
print("TEST SHAPE: ",Test_Data.shape)
```

```
TRAIN SHAPE:  (898, 2)
TEST SHAPE:  (100, 2)
```

In [33]:
```
print(Train_Data.head(-1))
print("----"*20)
print(Test_Data.head(-1))
```

```
                                                   PNG CATEGORY
600  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
432  ../input/fire-dataset/fire_dataset/non_fire_im...  NO_FIRE
221  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
973  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
526  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
..                                                 ...      ...
71   ../input/fire-dataset/fire_dataset/fire_images...     FIRE
106  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
270  ../input/fire-dataset/fire_dataset/non_fire_im...  NO_FIRE
861  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
435  ../input/fire-dataset/fire_dataset/fire_images...     FIRE

[897 rows x 2 columns]
--------------------------------------------------------------------------------
                                                   PNG CATEGORY
453  ../input/fire-dataset/fire_dataset/non_fire_im...  NO_FIRE
793  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
209  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
309  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
740  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
..                                                 ...      ...
797  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
88   ../input/fire-dataset/fire_dataset/fire_images...     FIRE
63   ../input/fire-dataset/fire_dataset/fire_images...     FIRE
825  ../input/fire-dataset/fire_dataset/fire_images...     FIRE
621  ../input/fire-dataset/fire_dataset/fire_images...     FIRE

[99 rows x 2 columns]
```

In [57]:
```
print(Test_Data["CATEGORY"].value_counts())
```

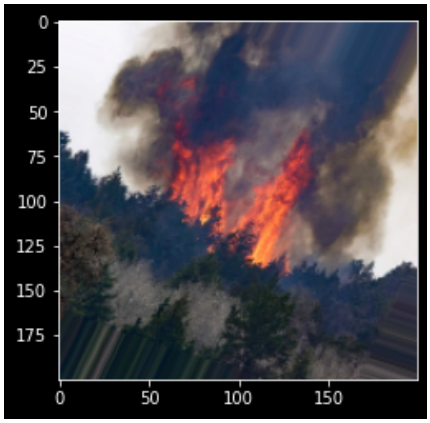
```
FIRE       80
NO_FIRE    20
Name: CATEGORY, dtype: int64
```

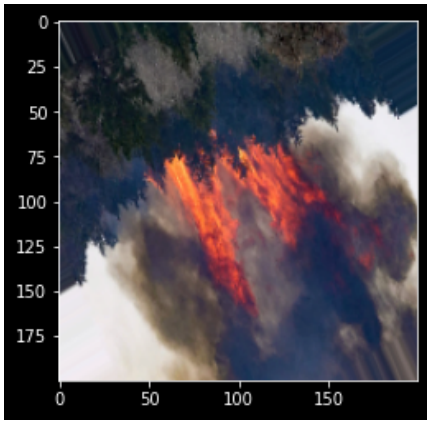
In [127]:
```
encode = LabelEncoder()
```

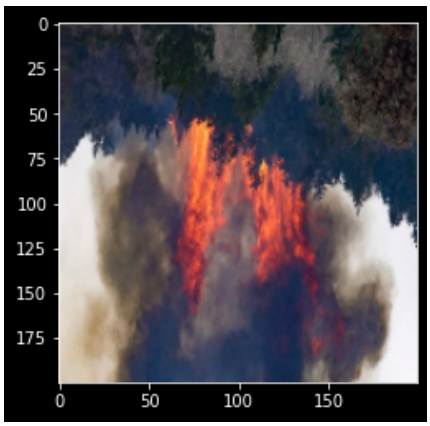
In [128]:
```
For_Prediction_Class = encode.fit_transform(Test_Data["CATEGORY"])
```

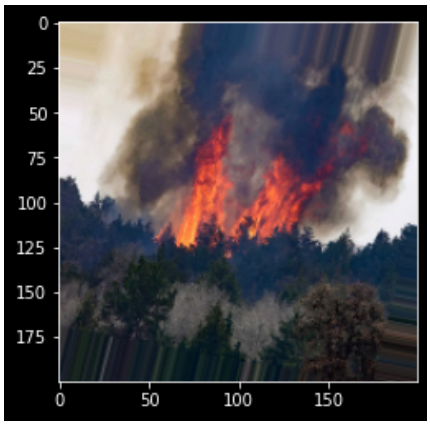
How Generator Applied Image Look Like

In [34]:
```
example_Image = Train_Data["PNG"][99]
Load_Image = image.load_img(example_Image,target_size=(200,200))
Array_Image = image.img_to_array(Load_Image)
Array_Image = Array_Image.reshape((1,) + Array_Image.shape)

i = 0
for batch in Train_Generator.flow(Array_Image,batch_size=1):
    plt.figure(i)
    IMG = plt.imshow(image.array_to_img(batch[0]))
    i += 1
    if i % 4 == 0:
        break
plt.show()
```









APPLYING GENERATOR AND TRANSFORMATION TO TENSOR

In [100]:

```
Train_IMG_Set = Train_Generator.flow_from_dataframe(dataframe=Train_Data,
                                                    x_col="PNG",
                                                    y_col="CATEGORY",
                                                    color_mode="rgb",
                                                    class_mode="categorical",
                                                    batch_size=32,
                                                    subset="training")
```

```
Found 809 validated image filenames belonging to 2 classes.
```

In [101]:
```
Validation_IMG_Set = Train_Generator.flow_from_dataframe(dataframe=Train_Data,
                                                         x_col="PNG",
                                                         y_col="CATEGORY",
                                                         color_mode="rgb",
                                                         class_mode="categorical",
                                                         batch_size=32,
                                                         subset="validation")
```

```
Found 89 validated image filenames belonging to 2 classes.
```

In [102]:
```
Test_IMG_Set = Test_Generator.flow_from_dataframe(dataframe=Test_Data,
                                                  x_col="PNG",
                                                  y_col="CATEGORY",
                                                  color_mode="rgb",
                                                  class_mode="categorical",
                                                  batch_size=32)
```

```
Found 100 validated image filenames belonging to 2 classes.
```

CHECKING

In [38]:
```
for data_batch,label_batch in Train_IMG_Set:
    print("DATA SHAPE: ",data_batch.shape)
    print("LABEL SHAPE: ",label_batch.shape)
    break
```

```
DATA SHAPE:  (32, 256, 256, 3)
LABEL SHAPE:  (32,)
```

In [39]:
```
for data_batch,label_batch in Validation_IMG_Set:
    print("DATA SHAPE: ",data_batch.shape)
    print("LABEL SHAPE: ",label_batch.shape)
    break
```

```
DATA SHAPE:  (32, 256, 256, 3)
LABEL SHAPE:  (32,)
```

In [40]:
```
for data_batch,label_batch in Test_IMG_Set:
    print("DATA SHAPE: ",data_batch.shape)
    print("LABEL SHAPE: ",label_batch.shape)
    break
```

```
DATA SHAPE:  (32, 256, 256, 3)
LABEL SHAPE:  (32,)
```

In [41]:

```python
print("TRAIN: ")
print(Train_IMG_Set.class_indices)
print(Train_IMG_Set.classes[0:5])
print(Train_IMG_Set.image_shape)
print("---"*20)
print("VALIDATION: ")
print(Validation_IMG_Set.class_indices)
print(Validation_IMG_Set.classes[0:5])
print(Validation_IMG_Set.image_shape)
print("---"*20)
print("TEST: ")
print(Test_IMG_Set.class_indices)
print(Test_IMG_Set.classes[0:5])
print(Test_IMG_Set.image_shape)
```

```
TRAIN:
{'FIRE': 0, 'NO_FIRE': 1}
[0, 0, 0, 0, 1]
(256, 256, 3)
------------------------------------------------------------
VALIDATION:
{'FIRE': 0, 'NO_FIRE': 1}
[0, 1, 0, 0, 0]
(256, 256, 3)
------------------------------------------------------------
TEST:
{'FIRE': 0, 'NO_FIRE': 1}
[1, 0, 0, 0, 0]
(256, 256, 3)
```

CNN

In [42]:

```
Model = Sequential()

Model.add(Conv2D(32,(3,3),activation="relu",
                 input_shape=(256,256,3)))
Model.add(BatchNormalization())
Model.add(MaxPooling2D((2,2)))

#
Model.add(Conv2D(64,(3,3),
                 activation="relu",padding="same"))
Model.add(Dropout(0.2))
Model.add(MaxPooling2D((2,2)))

#
Model.add(Conv2D(128,(3,3),
                 activation="relu",padding="same"))
Model.add(Dropout(0.5))
Model.add(MaxPooling2D((2,2)))



#
Model.add(Flatten())
Model.add(Dense(256,activation="relu"))
Model.add(Dropout(0.5))
Model.add(Dense(1,activation="sigmoid"))
```

In [43]:

```
Call_Back = tf.keras.callbacks.EarlyStopping(monitor="loss",patience=5,mode="min")
```

In [44]:

```
Model.compile(optimizer="rmsprop",loss="binary_crossentropy",metrics=["accuracy"])
```

In [45]:

```
CNN_Model = Model.fit(Train_IMG_Set,
                      validation_data=Validation_IMG_Set,
                            callbacks=Call_Back,
                      epochs=50)
```

```
Epoch 1/50
26/26 [==============================] - 61s 2s/step - loss: 15.8056 - accuracy: 0.7163 - val_
loss: 0.5792 - val_accuracy: 0.9101
Epoch 2/50
26/26 [==============================] - 40s 2s/step - loss: 0.3194 - accuracy: 0.9043 - val_l
oss: 0.5541 - val_accuracy: 0.8876
Epoch 3/50
26/26 [==============================] - 39s 2s/step - loss: 0.5434 - accuracy: 0.8847 - val_l
oss: 0.6414 - val_accuracy: 0.6966
Epoch 4/50
26/26 [==============================] - 39s 1s/step - loss: 0.4849 - accuracy: 0.8876 - val_l
oss: 0.6656 - val_accuracy: 0.5169
Epoch 5/50
26/26 [==============================] - 39s 2s/step - loss: 0.3260 - accuracy: 0.9067 - val_l
oss: 0.3692 - val_accuracy: 0.8652
Epoch 6/50
26/26 [==============================] - 39s 2s/step - loss: 0.3717 - accuracy: 0.9066 - val_l
oss: 0.4796 - val_accuracy: 0.9663
Epoch 7/50
26/26 [==============================] - 39s 2s/step - loss: 0.2214 - accuracy: 0.9341 - val_l
oss: 0.6235 - val_accuracy: 0.9101
Epoch 8/50
26/26 [==============================] - 40s 2s/step - loss: 0.2796 - accuracy: 0.9147 - val_l
oss: 0.3116 - val_accuracy: 0.8539
Epoch 9/50
26/26 [==============================] - 39s 2s/step - loss: 0.4163 - accuracy: 0.9149 - val_l
oss: 0.5475 - val_accuracy: 0.8427
Epoch 10/50
26/26 [==============================] - 40s 2s/step - loss: 0.2502 - accuracy: 0.9328 - val_l
oss: 0.4526 - val_accuracy: 0.9663
Epoch 11/50
26/26 [==============================] - 40s 2s/step - loss: 0.1751 - accuracy: 0.9362 - val_l
oss: 0.4806 - val_accuracy: 0.9101
Epoch 12/50
26/26 [==============================] - 39s 1s/step - loss: 0.3184 - accuracy: 0.9003 - val_l
oss: 0.5169 - val_accuracy: 0.8989
```

CHECKING

In [46]:

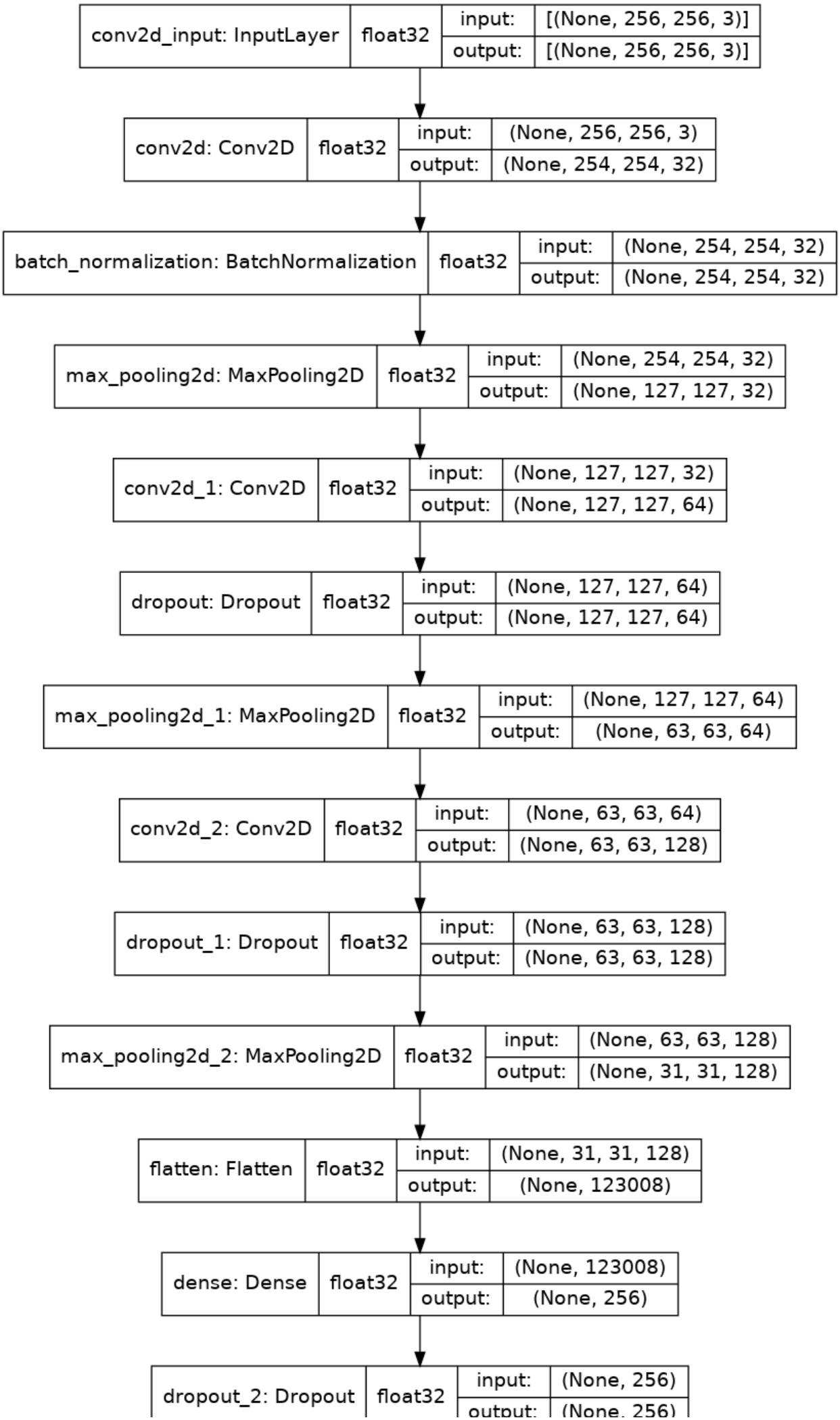
```
print(Model.summary())
```

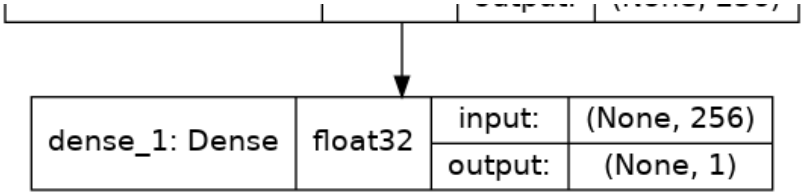
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 254, 254, 32)      896
_________________________________________________________________
batch_normalization (BatchNo (None, 254, 254, 32)      128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 127, 127, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 127, 127, 64)      18496
_________________________________________________________________
dropout (Dropout)            (None, 127, 127, 64)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 63, 63, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 63, 63, 128)       73856
_________________________________________________________________
dropout_1 (Dropout)          (None, 63, 63, 128)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 31, 31, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 123008)            0
_________________________________________________________________
dense (Dense)                (None, 256)               31490304
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 257
=================================================================
Total params: 31,583,937
Trainable params: 31,583,873
Non-trainable params: 64
_________________________________________________________________
None
```

In [49]:

```
plot_model(Model,to_file="Model_One.png",show_layer_names=True,show_dtype=True,show_shapes=True)
```

Out[49]:


conv2d_input: InputLayer | float32 | input: [(None, 256, 256, 3)] | output: [(None, 256, 256, 3)]
conv2d: Conv2D | float32 | input: (None, 256, 256, 3) | output: (None, 254, 254, 32)
batch_normalization: BatchNormalization | float32 | input: (None, 254, 254, 32) | output: (None, 254, 254, 32)
max_pooling2d: MaxPooling2D | float32 | input: (None, 254, 254, 32) | output: (None, 127, 127, 32)
conv2d_1: Conv2D | float32 | input: (None, 127, 127, 32) | output: (None, 127, 127, 64)
dropout: Dropout | float32 | input: (None, 127, 127, 64) | output: (None, 127, 127, 64)
max_pooling2d_1: MaxPooling2D | float32 | input: (None, 127, 127, 64) | output: (None, 63, 63, 64)
conv2d_2: Conv2D | float32 | input: (None, 63, 63, 64) | output: (None, 63, 63, 128)
dropout_1: Dropout | float32 | input: (None, 63, 63, 128) | output: (None, 63, 63, 128)
max_pooling2d_2: MaxPooling2D | float32 | input: (None, 63, 63, 128) | output: (None, 31, 31, 128)
flatten: Flatten | float32 | input: (None, 31, 31, 128) | output: (None, 123008)
dense: Dense | float32 | input: (None, 123008) | output: (None, 256)
dropout_2: Dropout | float32 | input: (None, 256) | output: (None, 256)



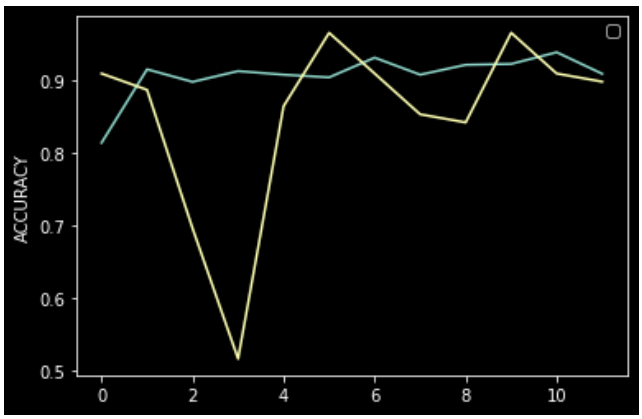
In [50]:

```
Model_Results = Model.evaluate(Test_IMG_Set)
print("LOSS:  " + "%.4f" % Model_Results[0])
print("ACCURACY:  " + "%.2f" % Model_Results[1])
```

```
4/4 [==============================] - 5s 1s/step - loss: 0.3309 - accuracy: 0.9100
LOSS:  0.3309
ACCURACY:  0.91
```

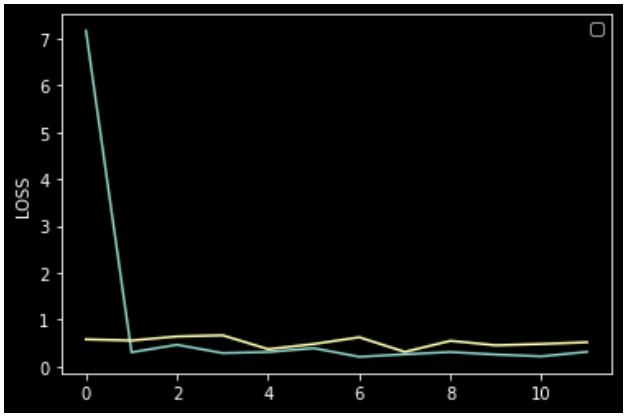
In [51]:

```
plt.plot(CNN_Model.history["accuracy"])
plt.plot(CNN_Model.history["val_accuracy"])
plt.ylabel("ACCURACY")
plt.legend()
plt.show()
```



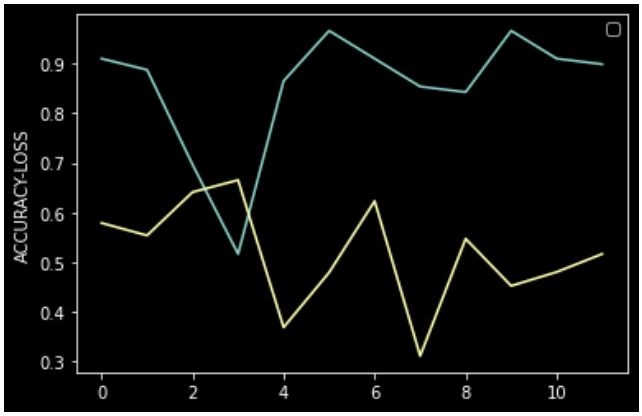
In [149]:

```
plt.plot(CNN_Model.history["loss"])
plt.plot(CNN_Model.history["val_loss"])
plt.ylabel("LOSS")
plt.legend()
plt.show()
```



In [148]:

```
plt.plot(CNN_Model.history["val_accuracy"])
plt.plot(CNN_Model.history["val_loss"])
plt.ylabel("ACCURACY-LOSS")
plt.legend()
plt.show()
```

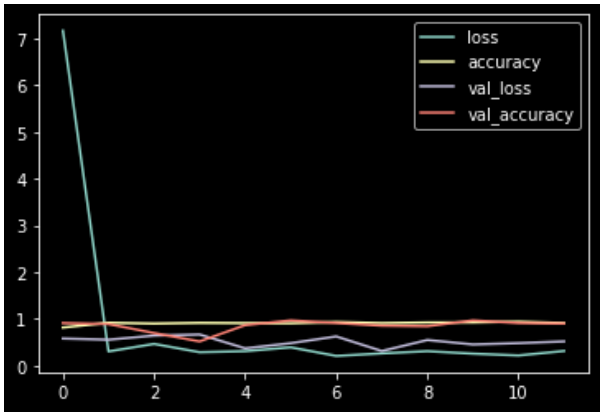


In [55]:

```
Dict_Summary_One = pd.DataFrame(CNN_Model.history)
Dict_Summary_One.plot()
```

Out[55]:

```
<AxesSubplot:>
```



PREDICTION

In [54]:
```python
Prediction_One = Model.predict(Test_IMG_Set)
Prediction_One = Prediction_One.argmax(axis=-1)
```

In [56]:
```python
print(Prediction_One)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

In [121]:
```python
Predict_Class = Model.predict_classes(Test_IMG_Set)
```

In [122]:
```python
print(Predict_Class)
```

```
[[0]
 [0]
 [0]
 [1]
 [0]
 [0]
 [0]
 [1]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [1]
 [1]
 [0]
 [1]
 [0]
 [1]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [1]
 [0]
 [1]
 [0]
 [1]
 [0]
 [0]
 [0]
 [0]
 [0]
 [1]
 [0]
 [0]
 [0]
 [0]
```

```
[0]
[1]
[1]
[0]
[0]
[0]
[0]
[0]
[0]
[0]
[0]
[1]
[0]
[0]
[0]
[0]
[0]
[1]
[1]
[0]
[0]
[1]
[0]
[0]
[0]
[0]
[1]
[0]
[0]
[0]
[0]
[1]
[1]
[0]
[0]
[1]
[1]
[1]
[0]
[0]
[1]
[1]
[1]
[0]
[0]
[0]
[0]
[1]]
```

- {'FIRE': 0, 'NO_FIRE': 1}

In [123]:

```
fig, axes = plt.subplots(nrows=8,
                         ncols=8,
                         figsize=(20, 20),
                        subplot_kw={'xticks': [], 'yticks': []})

for i, ax in enumerate(axes.flat):
    ax.imshow(cv2.imread(Test_Data["PNG"].iloc[i]))
    ax.set_title(f"TEST:{Test_Data.CATEGORY.iloc[i]}\n PREDICTION:{Predict_Class[i]}")
plt.tight_layout()
plt.show()
```

CLASSIFICATION & CONFUSION REPORT

In [132]:

```
print(confusion_matrix(For_Prediction_Class,Predict_Class))
```

```
[[58 22]
 [15  5]]
```

In [133]:
```
print(classification_report(For_Prediction_Class,Predict_Class))
```

```
              precision    recall  f1-score   support

           0       0.79      0.72      0.76        80
           1       0.19      0.25      0.21        20

    accuracy                           0.63       100
   macro avg       0.49      0.49      0.49       100
weighted avg       0.67      0.63      0.65       100
```

PREDICTION FOR DIFFERENT SOURCE

- FIRE

In [193]:
```
image_path = "../input/test-dataset/Fire-Detection/1/12.jpg"
img = image.load_img(image_path,target_size=(256,256))
x = image.img_to_array(img)
x = np.expand_dims(x,axis=0)
```

In [194]:
```
Diff_Pred = Model.predict(x)
Diff_Pred = Diff_Pred.argmax(axis=-1)
print(Diff_Pred)
```

```
[0]
```

- NOT FIRE

In [201]:
```
image_path_Two = "../input/test-dataset/Fire-Detection/0/12.jpg"
img_Two = image.load_img(image_path_Two,target_size=(256,256))
x_Two = image.img_to_array(img_Two)
x_Two = np.expand_dims(x_Two,axis=0)
```

In [202]:
```
Diff_Pred_Two = Model.predict(x_Two)
Diff_Pred_Two = Diff_Pred_Two.argmax(axis=-1)
print(Diff_Pred_Two)
```

```
[0]
```

FULLY-CONNECTED

In [103]:
```
Model_Two = tf.keras.models.Sequential([
  # inputs
  tf.keras.layers.experimental.preprocessing.Rescaling(1./255),
  tf.keras.layers.Flatten(input_shape=(256,)),
  # hiddens layers
  tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  # output layer
  tf.keras.layers.Dense(2,activation="softmax")
])
```

In [104]:
```
Model_Two.compile(optimizer = 'adam' , loss = 'categorical_crossentropy' , metrics = ['accurac
y'])
```

In [105]:
```
ANN_Model = Model_Two.fit(Train_IMG_Set,
                          validation_data=Validation_IMG_Set,
                          callbacks=Call_Back,
                      epochs=100)
```

```
Epoch 1/100
26/26 [==============================] - 37s 1s/step - loss: 0.5780 - accuracy: 0.6921 - val_l
oss: 0.4248 - val_accuracy: 0.7865
Epoch 2/100
26/26 [==============================] - 36s 1s/step - loss: 0.3962 - accuracy: 0.8193 - val_l
oss: 0.3260 - val_accuracy: 0.8989
Epoch 3/100
26/26 [==============================] - 36s 1s/step - loss: 0.3354 - accuracy: 0.8624 - val_l
oss: 0.2449 - val_accuracy: 0.9326
Epoch 4/100
26/26 [==============================] - 35s 1s/step - loss: 0.2652 - accuracy: 0.8776 - val_l
oss: 0.2100 - val_accuracy: 0.9326
Epoch 5/100
26/26 [==============================] - 36s 1s/step - loss: 0.2554 - accuracy: 0.9055 - val_l
oss: 0.1890 - val_accuracy: 0.9438
Epoch 6/100
26/26 [==============================] - 36s 1s/step - loss: 0.2306 - accuracy: 0.9188 - val_l
oss: 0.2673 - val_accuracy: 0.8989
Epoch 7/100
26/26 [==============================] - 36s 1s/step - loss: 0.2158 - accuracy: 0.9300 - val_l
oss: 0.1788 - val_accuracy: 0.9326
Epoch 8/100
26/26 [==============================] - 36s 1s/step - loss: 0.2058 - accuracy: 0.9215 - val_l
oss: 0.2316 - val_accuracy: 0.9438
Epoch 9/100
26/26 [==============================] - 35s 1s/step - loss: 0.1929 - accuracy: 0.9272 - val_l
oss: 0.1441 - val_accuracy: 0.9438
Epoch 10/100
26/26 [==============================] - 36s 1s/step - loss: 0.1600 - accuracy: 0.9416 - val_l
oss: 0.1740 - val_accuracy: 0.9438
Epoch 11/100
26/26 [==============================] - 36s 1s/step - loss: 0.1603 - accuracy: 0.9533 - val_l
oss: 0.1717 - val_accuracy: 0.9326
Epoch 12/100
26/26 [==============================] - 35s 1s/step - loss: 0.1989 - accuracy: 0.9214 - val_l
oss: 0.1720 - val_accuracy: 0.9326
Epoch 13/100
26/26 [==============================] - 36s 1s/step - loss: 0.1611 - accuracy: 0.9318 - val_l
oss: 0.1804 - val_accuracy: 0.9326
Epoch 14/100
26/26 [==============================] - 35s 1s/step - loss: 0.1715 - accuracy: 0.9388 - val_l
oss: 0.1698 - val_accuracy: 0.9551
Epoch 15/100
26/26 [==============================] - 36s 1s/step - loss: 0.1701 - accuracy: 0.9374 - val_l
oss: 0.1152 - val_accuracy: 0.9438
Epoch 16/100
26/26 [==============================] - 36s 1s/step - loss: 0.1829 - accuracy: 0.9324 - val_l
oss: 0.1942 - val_accuracy: 0.9438
Epoch 17/100
26/26 [==============================] - 37s 1s/step - loss: 0.1556 - accuracy: 0.9459 - val_l
oss: 0.1447 - val_accuracy: 0.9326
Epoch 18/100
```

```
26/26 [==============================] - 36s 1s/step - loss: 0.1874 - accuracy: 0.9295 - val_l
oss: 0.3976 - val_accuracy: 0.8876
Epoch 19/100
26/26 [==============================] - 36s 1s/step - loss: 0.2780 - accuracy: 0.9071 - val_l
oss: 0.1569 - val_accuracy: 0.9551
Epoch 20/100
26/26 [==============================] - 36s 1s/step - loss: 0.1734 - accuracy: 0.9326 - val_l
oss: 0.1500 - val_accuracy: 0.9551
Epoch 21/100
26/26 [==============================] - 36s 1s/step - loss: 0.1621 - accuracy: 0.9441 - val_l
oss: 0.1794 - val_accuracy: 0.9326
Epoch 22/100
26/26 [==============================] - 36s 1s/step - loss: 0.1550 - accuracy: 0.9523 - val_l
oss: 0.1352 - val_accuracy: 0.9438
Epoch 23/100
26/26 [==============================] - 36s 1s/step - loss: 0.1560 - accuracy: 0.9478 - val_l
oss: 0.1118 - val_accuracy: 0.9326
Epoch 24/100
26/26 [==============================] - 36s 1s/step - loss: 0.1997 - accuracy: 0.9224 - val_l
oss: 0.1814 - val_accuracy: 0.9438
Epoch 25/100
26/26 [==============================] - 37s 1s/step - loss: 0.1400 - accuracy: 0.9474 - val_l
oss: 0.1558 - val_accuracy: 0.9438
Epoch 26/100
26/26 [==============================] - 36s 1s/step - loss: 0.1901 - accuracy: 0.9286 - val_l
oss: 0.2079 - val_accuracy: 0.9213
Epoch 27/100
26/26 [==============================] - 36s 1s/step - loss: 0.1951 - accuracy: 0.9068 - val_l
oss: 0.1494 - val_accuracy: 0.9326
Epoch 28/100
26/26 [==============================] - 36s 1s/step - loss: 0.1590 - accuracy: 0.9469 - val_l
oss: 0.1887 - val_accuracy: 0.9551
```

CHECKING

In [106]:

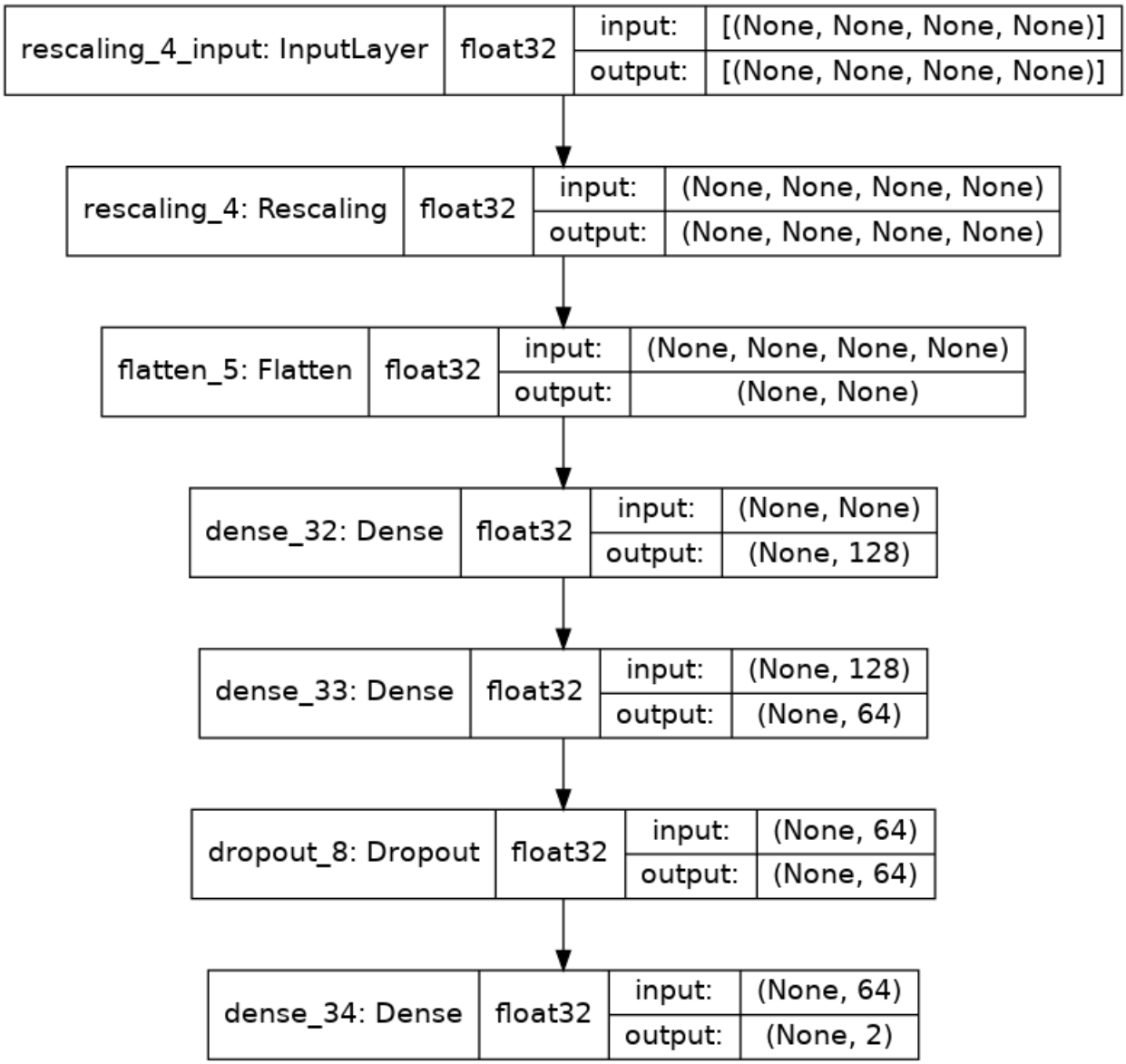
```
print(Model_Two.summary())
```

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rescaling_4 (Rescaling)      (None, None, None, None)  0
_________________________________________________________________
flatten_5 (Flatten)          (None, None)              0
_________________________________________________________________
dense_32 (Dense)             (None, 128)               25165952
_________________________________________________________________
dense_33 (Dense)             (None, 64)                8256
_________________________________________________________________
dropout_8 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_34 (Dense)             (None, 2)                 130
=================================================================
Total params: 25,174,338
Trainable params: 25,174,338
Non-trainable params: 0
_________________________________________________________________
None
```

In [107]:

```
plot_model(Model_Two,to_file="Model_Two.png",show_layer_names=True,show_dtype=True,show_shapes=True)
```

Out[107]:



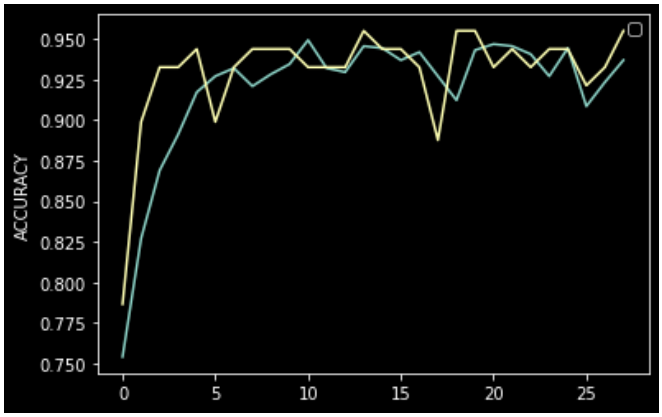
In [108]:

```
Model_Results_Two = Model_Two.evaluate(Test_IMG_Set)
print("LOSS:  " + "%.4f" % Model_Results_Two[0])
print("ACCURACY:  " + "%.2f" % Model_Results_Two[1])
```

```
4/4 [==============================] - 2s 570ms/step - loss: 0.2889 - accuracy: 0.9100
LOSS:  0.2889
ACCURACY:  0.91
```

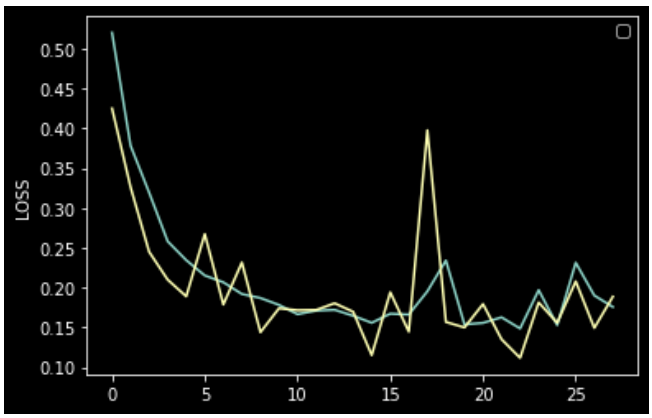
In [109]:

```
plt.plot(ANN_Model.history["accuracy"])
plt.plot(ANN_Model.history["val_accuracy"])
plt.ylabel("ACCURACY")
plt.legend()
plt.show()
```



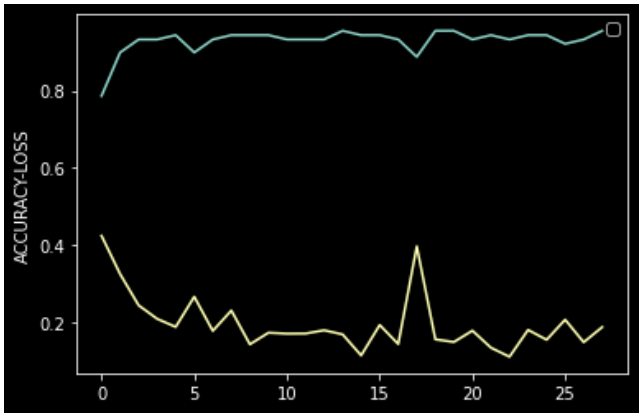
In [147]:

```
plt.plot(ANN_Model.history["loss"])
plt.plot(ANN_Model.history["val_loss"])
plt.ylabel("LOSS")
plt.legend()
plt.show()
```



In [146]:

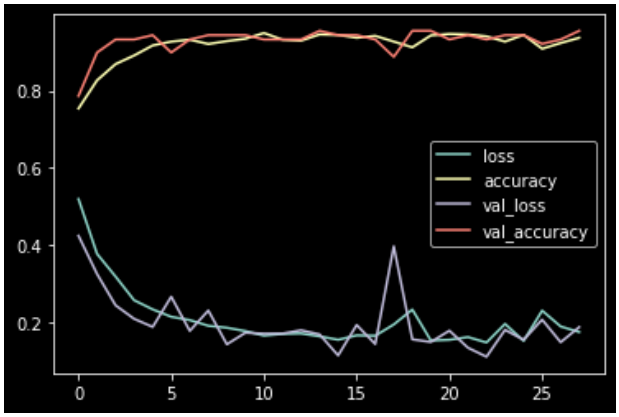
```
plt.plot(ANN_Model.history["val_accuracy"])
plt.plot(ANN_Model.history["val_loss"])
plt.ylabel("ACCURACY-LOSS")
plt.legend()
plt.show()
```



In [112]:
```
Dict_Summary_Two = pd.DataFrame(ANN_Model.history)
Dict_Summary_Two.plot()
```

Out[112]:

<AxesSubplot:>



PREDICTION

In [113]:
```
Prediction_Two = Model_Two.predict(Test_IMG_Set)
Prediction_Two = Prediction_Two.argmax(axis=-1)
```

In [114]:
```
print(Prediction_Two)
```

```
[0 0 0 0 1 1 1 0 0 0 0 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 0 1
 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 0 0 0
 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1]
```

In [154]:
```
Prediction_Class_Two = Model_Two.predict_classes(Test_IMG_Set)
```

In [155]:
```
print(Prediction_Class_Two)
```

```
[1 0 1 0 1 0 0 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0
 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0
 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 1]
```

- {'FIRE': 0, 'NO_FIRE': 1}

In [161]:

```
fig, axes = plt.subplots(nrows=8,
                         ncols=8,
                         figsize=(20, 20),
                        subplot_kw={'xticks': [], 'yticks': []})

for i, ax in enumerate(axes.flat):
    ax.imshow(cv2.imread(Test_Data["PNG"].iloc[i]))
    ax.set_title(f"TEST:{Test_Data.CATEGORY.iloc[i]}\n PREDICTION:{Prediction_Two[i]}")
plt.tight_layout()
plt.show()
```

CLASSIFICATION & CONFUSION REPORT

In [135]:

```
print(confusion_matrix(For_Prediction_Class,Prediction_Class_Two))
```

```
[[59 21]
 [14  6]]
```

In [134]:
```
print(classification_report(For_Prediction_Class,Prediction_Class_Two))
```

```
              precision    recall  f1-score   support

           0       0.81      0.74      0.77        80
           1       0.22      0.30      0.26        20

    accuracy                           0.65       100
   macro avg       0.52      0.52      0.51       100
weighted avg       0.69      0.65      0.67       100
```

PREDICTION FOR ANOTHER SOURCE

- FIRE

In [203]:
```
image_path = "../input/test-dataset/Fire-Detection/1/12.jpg"
img = image.load_img(image_path,target_size=(256,256))
x = image.img_to_array(img)
x = np.expand_dims(x,axis=0)
```

In [204]:
```
Diff_Pred = Model_Two.predict(x)
Diff_Pred = Diff_Pred.argmax(axis=-1)
print(Diff_Pred)
```

```
[0]
```

- NOT FIRE

In [206]:
```
image_path_Two = "../input/test-dataset/Fire-Detection/0/12.jpg"
img_Two = image.load_img(image_path_Two,target_size=(256,256))
x_Two = image.img_to_array(img_Two)
x_Two = np.expand_dims(x_Two,axis=0)
```

In [207]:
```
Diff_Pred_Two = Model_Two.predict(x_Two)
Diff_Pred_Two = Diff_Pred_Two.argmax(axis=-1)
print(Diff_Pred_Two)
```

[1]

CNN-RCNN

In [137]:
```
Model_Three = Sequential()

Model_Three.add(Conv2D(12,(3,3),activation="relu",
                input_shape=(256,256,3)))
Model_Three.add(BatchNormalization())
Model_Three.add(MaxPooling2D((2,2)))

#
Model_Three.add(Conv2D(24,(3,3),
                activation="relu"))
Model_Three.add(Dropout(0.2))
Model_Three.add(MaxPooling2D((2,2)))



#
Model_Three.add(TimeDistributed(Flatten()))
Model_Three.add(Bidirectional(LSTM(32,
                                   return_sequences=True,
                                   dropout=0.5,
                                   recurrent_dropout=0.5)))
Model_Three.add(Bidirectional(GRU(32,
                                   return_sequences=True,
                                   dropout=0.5,
                                   recurrent_dropout=0.5)))

#
Model_Three.add(Flatten())

Model_Three.add(Dense(256,activation="relu"))
Model_Three.add(Dropout(0.5))
Model_Three.add(Dense(2,activation="softmax"))
```

In [138]:
```
Model_Three.compile(optimizer = 'adam' , loss = 'categorical_crossentropy' , metrics = ['accurac
y'])
```

In [139]:
```
RCNN_Model = Model_Three.fit(Train_IMG_Set,
                         validation_data=Validation_IMG_Set,
                         callbacks=Call_Back,
                     epochs=100)
```

```
Epoch 1/100
26/26 [==============================] - 67s 2s/step - loss: 0.4907 - accuracy: 0.7774 - val_l
oss: 0.5092 - val_accuracy: 0.7640
Epoch 2/100
26/26 [==============================] - 54s 2s/step - loss: 0.1772 - accuracy: 0.9274 - val_l
oss: 0.5371 - val_accuracy: 0.7640
Epoch 3/100
26/26 [==============================] - 54s 2s/step - loss: 0.1856 - accuracy: 0.9462 - val_l
oss: 0.3748 - val_accuracy: 0.7978
Epoch 4/100
26/26 [==============================] - 54s 2s/step - loss: 0.1564 - accuracy: 0.9276 - val_l
oss: 0.3060 - val_accuracy: 0.8202
Epoch 5/100
26/26 [==============================] - 55s 2s/step - loss: 0.1211 - accuracy: 0.9556 - val_l
oss: 0.3546 - val_accuracy: 0.8090
Epoch 6/100
26/26 [==============================] - 54s 2s/step - loss: 0.1329 - accuracy: 0.9420 - val_l
oss: 0.3039 - val_accuracy: 0.8539
Epoch 7/100
26/26 [==============================] - 54s 2s/step - loss: 0.1354 - accuracy: 0.9469 - val_l
oss: 0.2580 - val_accuracy: 0.8539
Epoch 8/100
26/26 [==============================] - 55s 2s/step - loss: 0.1240 - accuracy: 0.9598 - val_l
oss: 0.2441 - val_accuracy: 0.8652
Epoch 9/100
26/26 [==============================] - 55s 2s/step - loss: 0.0923 - accuracy: 0.9708 - val_l
oss: 0.1709 - val_accuracy: 0.8989
Epoch 10/100
26/26 [==============================] - 54s 2s/step - loss: 0.0903 - accuracy: 0.9658 - val_l
oss: 0.2460 - val_accuracy: 0.8764
Epoch 11/100
26/26 [==============================] - 55s 2s/step - loss: 0.0790 - accuracy: 0.9662 - val_l
oss: 0.1485 - val_accuracy: 0.9551
Epoch 12/100
26/26 [==============================] - 55s 2s/step - loss: 0.1056 - accuracy: 0.9632 - val_l
oss: 0.1567 - val_accuracy: 0.9663
Epoch 13/100
26/26 [==============================] - 54s 2s/step - loss: 0.0928 - accuracy: 0.9656 - val_l
oss: 0.1905 - val_accuracy: 0.8989
Epoch 14/100
26/26 [==============================] - 54s 2s/step - loss: 0.0928 - accuracy: 0.9661 - val_l
oss: 0.1663 - val_accuracy: 0.9438
Epoch 15/100
26/26 [==============================] - 54s 2s/step - loss: 0.0842 - accuracy: 0.9728 - val_l
oss: 0.1544 - val_accuracy: 0.9551
Epoch 16/100
26/26 [==============================] - 54s 2s/step - loss: 0.0718 - accuracy: 0.9731 - val_l
oss: 0.1532 - val_accuracy: 0.9663
Epoch 17/100
26/26 [==============================] - 54s 2s/step - loss: 0.1056 - accuracy: 0.9489 - val_l
oss: 0.1087 - val_accuracy: 0.9551
Epoch 18/100
```

```
26/26 [==============================] - 55s 2s/step - loss: 0.1047 - accuracy: 0.9541 - val_l
oss: 0.3033 - val_accuracy: 0.8876
Epoch 19/100
26/26 [==============================] - 54s 2s/step - loss: 0.1407 - accuracy: 0.9459 - val_l
oss: 0.1545 - val_accuracy: 0.9663
Epoch 20/100
26/26 [==============================] - 54s 2s/step - loss: 0.0843 - accuracy: 0.9704 - val_l
oss: 0.1410 - val_accuracy: 0.9551
Epoch 21/100
26/26 [==============================] - 54s 2s/step - loss: 0.0803 - accuracy: 0.9729 - val_l
oss: 0.0969 - val_accuracy: 0.9775
```

In [142]:

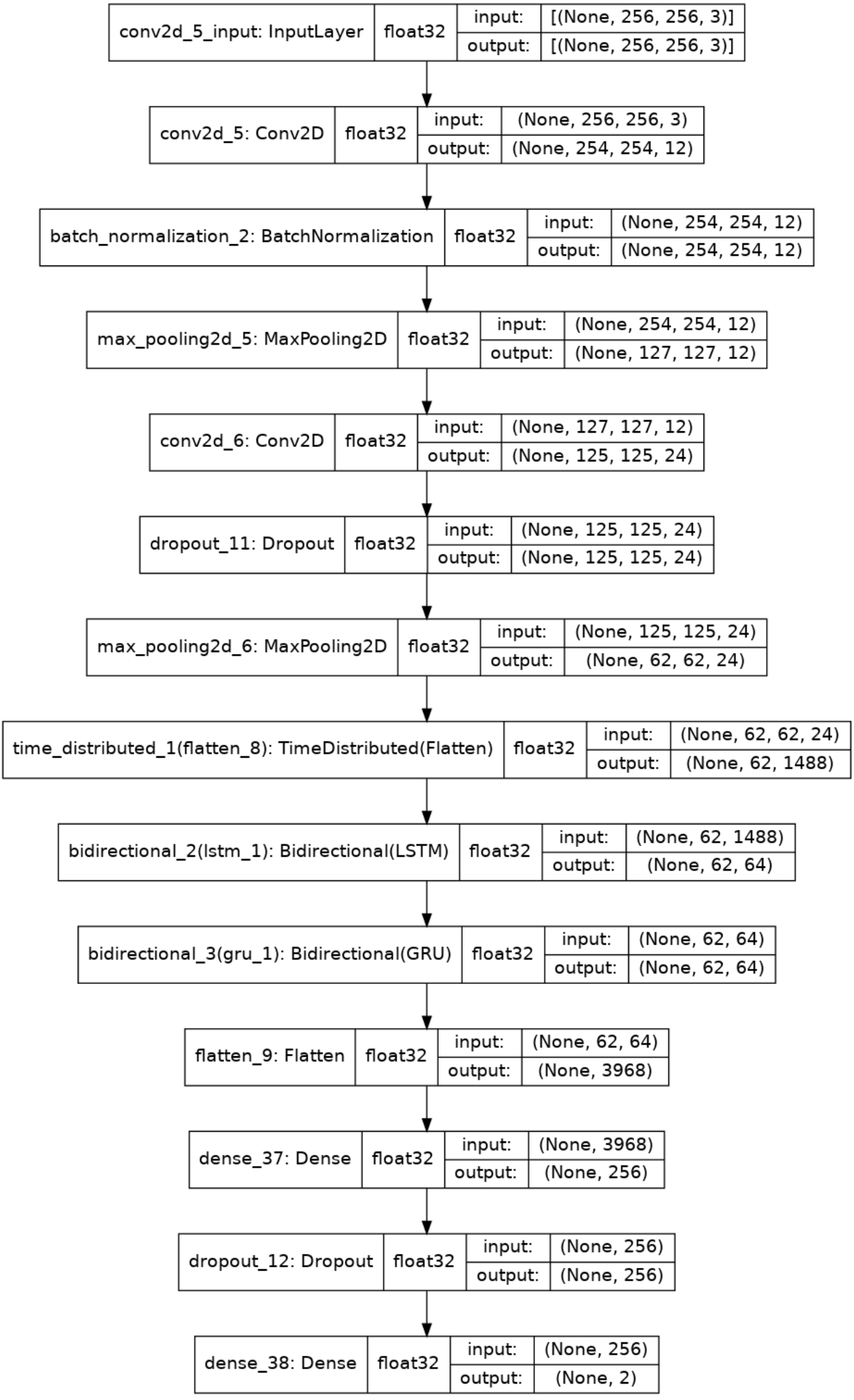
```
print(Model_Three.summary())
```

```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 254, 254, 12)      336
_________________________________________________________________
batch_normalization_2 (Batch (None, 254, 254, 12)      48
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 127, 127, 12)      0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 125, 125, 24)      2616
_________________________________________________________________
dropout_11 (Dropout)         (None, 125, 125, 24)      0
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 62, 62, 24)        0
_________________________________________________________________
time_distributed_1 (TimeDist (None, 62, 1488)          0
_________________________________________________________________
bidirectional_2 (Bidirection (None, 62, 64)            389376
_________________________________________________________________
bidirectional_3 (Bidirection (None, 62, 64)            18816
_________________________________________________________________
flatten_9 (Flatten)          (None, 3968)              0
_________________________________________________________________
dense_37 (Dense)             (None, 256)               1016064
_________________________________________________________________
dropout_12 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_38 (Dense)             (None, 2)                 514
=================================================================
Total params: 1,427,770
Trainable params: 1,427,746
Non-trainable params: 24
_________________________________________________________________
None
```

In [141]:

```
plot_model(Model_Three,to_file="Model_Three.png",show_layer_names=True,show_dtype=True,show_shapes=True)
```

Out[141]:


conv2d_5_input: InputLayer | float32 | input: [(None, 256, 256, 3)] | output: [(None, 256, 256, 3)]
conv2d_5: Conv2D | float32 | input: (None, 256, 256, 3) | output: (None, 254, 254, 12)
batch_normalization_2: BatchNormalization | float32 | input: (None, 254, 254, 12) | output: (None, 254, 254, 12)
max_pooling2d_5: MaxPooling2D | float32 | input: (None, 254, 254, 12) | output: (None, 127, 127, 12)
conv2d_6: Conv2D | float32 | input: (None, 127, 127, 12) | output: (None, 125, 125, 24)
dropout_11: Dropout | float32 | input: (None, 125, 125, 24) | output: (None, 125, 125, 24)
max_pooling2d_6: MaxPooling2D | float32 | input: (None, 125, 125, 24) | output: (None, 62, 62, 24)
time_distributed_1(flatten_8): TimeDistributed(Flatten) | float32 | input: (None, 62, 62, 24) | output: (None, 62, 1488)
bidirectional_2(lstm_1): Bidirectional(LSTM) | float32 | input: (None, 62, 1488) | output: (None, 62, 64)
bidirectional_3(gru_1): Bidirectional(GRU) | float32 | input: (None, 62, 64) | output: (None, 62, 64)
flatten_9: Flatten | float32 | input: (None, 62, 64) | output: (None, 3968)
dense_37: Dense | float32 | input: (None, 3968) | output: (None, 256)
dropout_12: Dropout | float32 | input: (None, 256) | output: (None, 256)
dense_38: Dense | float32 | input: (None, 256) | output: (None, 2)

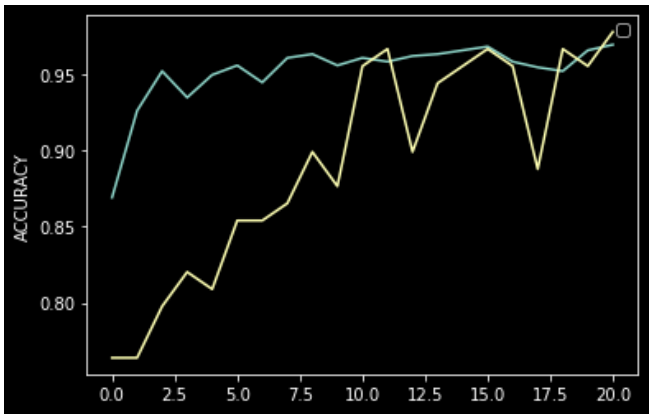
In [140]:

```
Model_Results_Three = Model_Three.evaluate(Test_IMG_Set)
print("LOSS:  " + "%.4f" % Model_Results_Three[0])
print("ACCURACY:  " + "%.2f" % Model_Results_Three[1])
```

```
4/4 [==============================] - 3s 510ms/step - loss: 0.0797 - accuracy: 0.9800
LOSS:  0.0797
ACCURACY:  0.98
```

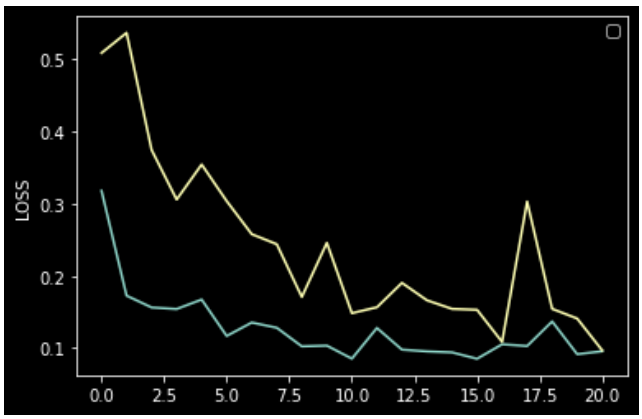
In [143]:

```
plt.plot(RCNN_Model.history["accuracy"])
plt.plot(RCNN_Model.history["val_accuracy"])
plt.ylabel("ACCURACY")
plt.legend()
plt.show()
```

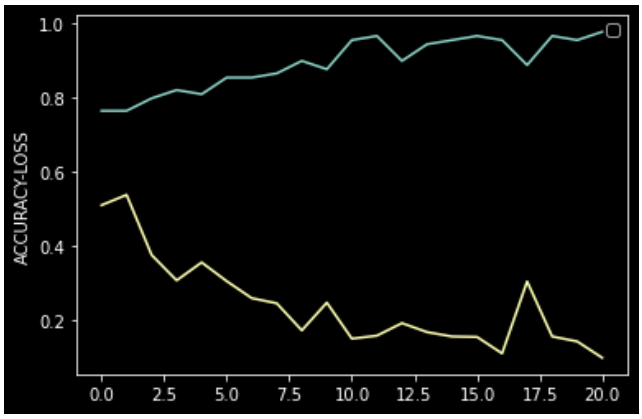


In [145]:

```
plt.plot(RCNN_Model.history["loss"])
plt.plot(RCNN_Model.history["val_loss"])
plt.ylabel("LOSS")
plt.legend()
plt.show()
```

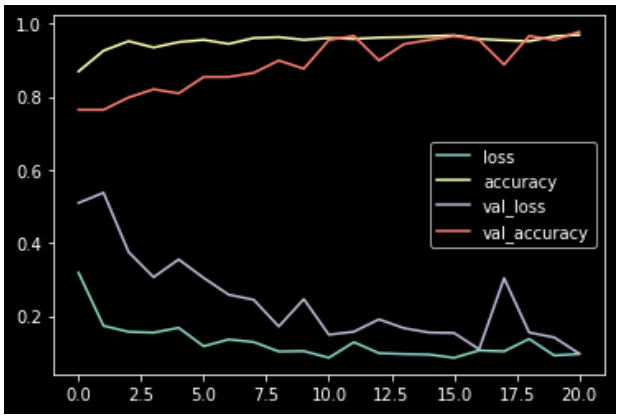


In [150]:
```
plt.plot(RCNN_Model.history["val_accuracy"])
plt.plot(RCNN_Model.history["val_loss"])
plt.ylabel("ACCURACY-LOSS")
plt.legend()
plt.show()
```



In [151]:
```
Dict_Summary_Three = pd.DataFrame(RCNN_Model.history)
Dict_Summary_Three.plot()
```

Out[151]:
```
<AxesSubplot:>
```



In [152]:
```
Prediction_Three = Model_Three.predict(Test_IMG_Set)
Prediction_Three = Prediction_Three.argmax(axis=-1)
```

In [153]:
```
print(Prediction_Three)
```

```
[0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1
 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0]
```

- {'FIRE': 0, 'NO_FIRE': 1}

In [157]:
```
Prediction_Class_Three = Model_Three.predict_classes(Test_IMG_Set)
```

In [158]:
```
print(Prediction_Class_Three)
```

```
[0 0 1 0 1 0 0 1 0 0 0 1 0 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0
 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1]
```

In [160]:
```
fig, axes = plt.subplots(nrows=8,
                         ncols=8,
                         figsize=(20, 20),
                        subplot_kw={'xticks': [], 'yticks': []})

for i, ax in enumerate(axes.flat):
    ax.imshow(cv2.imread(Test_Data["PNG"].iloc[i]))
    ax.set_title(f"TEST:{Test_Data.CATEGORY.iloc[i]}\n PREDICTION:{Prediction_Three[i]}")
plt.tight_layout()
plt.show()
```

CLASSIFICATION & CONFUSION REPORT

In [162]:

```
print(classification_report(For_Prediction_Class,Prediction_Three))
```

```
              precision    recall  f1-score   support

           0       0.81      0.85      0.83        80
           1       0.25      0.20      0.22        20

    accuracy                           0.72       100
   macro avg       0.53      0.53      0.53       100
weighted avg       0.70      0.72      0.71       100
```

In [163]:
```
print(confusion_matrix(For_Prediction_Class,Prediction_Three))
```

```
[[68 12]
 [16  4]]
```

PREDICTION FOR DIFFERENT SOURCE

- FIRE

In [ ]:
```
image_path = "../input/test-dataset/Fire-Detection/1/12.jpg"
img = image.load_img(image_path,target_size=(256,256))
x = image.img_to_array(img)
x = np.expand_dims(x,axis=0)
```

In [ ]:
```
Diff_Pred = Model_Three.predict(x)
Diff_Pred = Diff_Pred.argmax(axis=-1)
print(Diff_Pred)
```

[0]

- NOT FIRE

In [ ]:
```
image_path_Two = "../input/test-dataset/Fire-Detection/0/12.jpg"
img_Two = image.load_img(image_path_Two,target_size=(256,256))
x_Two = image.img_to_array(img_Two)
x_Two = np.expand_dims(x_Two,axis=0)
```

In [ ]:
```
Diff_Pred_Three = Model_Three.predict(x)
Diff_Pred_Three = Diff_Pred_Three.argmax(axis=-1)
print(Diff_Pred_Three)
```