```
import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense, Flatten, Dropout, BatchNormalization,
Conv2D, MaxPooling2D
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from tensorflow.keras.preprocessing.image import ImageDataGenerator,
load_img, img_to_array, save_img, array_to_img
```

```
df = pd.read_csv('/content/emergency_classification.csv')
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 2352,\n  \"fields\":
[\n    {\n      \"column\": \"image_names\",\n      \"properties\": {\
n        \"dtype\": \"string\",\n        \"num_unique_values\": 2352,\
n        \"samples\": [\n          \"1960.jpg\",\n
\"668.jpg\",\n          \"2082.jpg\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"emergency_or_not\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


```
df['emergency_or_not'].value_counts()
```

```
emergency_or_not
0    1361
1     991
Name: count, dtype: int64
```

```
import os
len(os.listdir('/content/sample_data/Images'))
```

```
2345
```

For some reason all the 2352 images were not uploading, tried partially uploading them too. Hence, will upload the remaining 7 images in a different folder and will merge them later.

Handling Class Imbalance

```
data_dir = '/content/sample_data/Images'
output_dir = '/content/sample_data/aug_images_2'
os.makedirs(output_dir, exist_ok=True)

class_counts = df['emergency_or_not'].value_counts()  # Counts images
per class
minority_class = class_counts.idxmin()  # Class with the fewest images
majority_class_count = class_counts.max()  # Count of the majority
class
num_augmented_images = majority_class_count - class_counts.min()  #
Number of images to generate
num_augmented_images
```

```
370
```

```
datagen = ImageDataGenerator(
    rotation_range=30,          # Random rotation up to 30 degrees
    width_shift_range=0.2,      # Random horizontal shifts up to 20%
    height_shift_range=0.2,     # Random vertical shifts up to 20%
    shear_range=0.2,            # Shearing transformations
    zoom_range=0.2,             # Random zoom
    horizontal_flip=True,       # Randomly flip images horizontally
    fill_mode='nearest'         # Fill empty pixels after
transformation
)



minority_images = df[df['emergency_or_not'] == minority_class]
['image_names']
current_count = 0

for image_name in minority_images:
  img_path = os.path.join(data_dir, image_name)
  img = load_img(img_path)
  img_arr = img_to_array(img)
  img_arr = np.expand_dims(img_arr, axis = 0)

  #Genearte augmented images
  for _ in range(num_augmented_images - current_count):
    aug_img = next(datagen.flow(img_arr, batch_size = 1))
[0].astype('uint8')
    save_img(os.path.join(output_dir, f'aug_{current_count}.jpg'),
```

```
aug_img)
    current_count +=1

    if current_count >= num_augmented_images:
      break


print(f"Generated {num_augmented_images} augmented images.")
```

```
Generated 370 augmented images.
```

Created a folder with the name 'extra' for the 7 remaining images

```
len(os.listdir('/content/sample_data/extra'))
```

```
7
```

```
x = list(os.listdir(data_dir))
y = list(os.listdir(output_dir))
z = list(os.listdir('/content/sample_data/extra'))
len(x), len(y), len(z)
```

```
(2345, 370, 7)
```

```
res = x+y+z
len(res)
```

```
2722
```

Created a new dataframe for augmented images that will balance my minority class

```
df_1 = pd.DataFrame({'image_names' : [file for file in
os.listdir(output_dir)],
                     'emergency_or_not' : [1 for i in
range(len(os.listdir(output_dir)))]})

df_1
```


```
{"summary":"{\n  \"name\": \"df_1\",\n  \"rows\": 370,\n  \"fields\":
[\n    {\n      \"column\": \"image_names\",\n      \"properties\": {\
n        \"dtype\": \"string\",\n        \"num_unique_values\": 370,\n
\"samples\": [\n          \"aug_317.jpg\",\n          \"aug_9.jpg\",\n
\"aug_70.jpg\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"emergency_or_not\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 1,\n
\"max\": 1,\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df_1"}
```

Combining all the dataframes and handling image paths so that i can use flow_from_dataframe() method

```
ex_df = df.tail(7)
len(df), len(df_1), len(ex_df)
df_2 = df.iloc[:len(df)-len(ex_df)]
len(df_2)
```

```
2345
```

```
image_data_dir = '/content/sample_data/Images'
aug_data_dir = '/content/sample_data/aug_images_2'
ex_dir = '/content/sample_data/extra'

ex_df = df.tail(7)
df = df.iloc[:len(df)-len(ex_df)]
combined_df = pd.concat([df, df_1, ex_df], ignore_index=True)
combined_df['image_paths'] = combined_df['image_names'].apply(lambda x
: image_data_dir + '/' + x if x in df['image_names'].values else
(aug_data_dir + x if x in df_1['image_names'].values else ex_dir + x))
combined_df
```


```
{"summary":"{\n  \"name\": \"combined_df\",\n  \"rows\": 2722,\n
\"fields\": [\n    {\n      \"column\": \"image_names\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 2722,\n        \"samples\": [\n
\"1061.jpg\",\n          \"408.jpg\",\n          \"aug_81.jpg\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"emergency_or_not\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"image_paths\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 2722,\n
\"samples\": [\n          \"/content/sample_data/Images/1061.jpg\",\n
\"/content/sample_data/Images/408.jpg\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"combined_df"}
```


```
combined_df = combined_df.sample(2722) # Reshuffling my dataframe

combined_df['emergency_or_not'] =
combined_df['emergency_or_not'].astype('str')

len(combined_df)
```

```
2722
```

```
combined_df['emergency_or_not'].value_counts()
```

```
emergency_or_not
0    1361
1    1361
Name: count, dtype: int64
```

```
train_data = combined_df.sample(frac = 0.8, random_state = 42)
test_data = combined_df.drop(train_data.index)

# Create ImageDataGenerator for training data (with augmentations)

train_datagen = ImageDataGenerator(rescale=1./255,  # Rescale pixel
values to be between 0 and 1
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

# Create ImageDataGenerator for test data (only rescaling, no
augmentations)
test_datagen = ImageDataGenerator(rescale=1./255)

# Create the training data generator
train_generator = train_datagen.flow_from_dataframe(train_data,
directory = None,
                                                    x_col =
'image_paths',
                                                    y_col =
'emergency_or_not',
                                                    target_size =
(224,224),
                                                    batch_size = 32,
                                                    class_mode =
'binary')

#Create the test data generator
test_generator = test_datagen.flow_from_dataframe(test_data,
                                                  directory = None,
                                                  x_col =
'image_paths',
                                                  y_col =
'emergency_or_not',
                                                  batch_size = 32,
                                                  target_size =
(224,224),
                                                  class_mode =
'binary')
```

```
Found 1875 validated image filenames belonging to 2 classes.
Found 463 validated image filenames belonging to 2 classes.
```

#VGG 16

```
model = Sequential()

#Block 1

model.add(Conv2D(32, kernel_size = (3,3), activation = 'relu',
input_shape = (224,224,3)))
model.add(BatchNormalization())
model.add(Conv2D(32, kernel_size = (3,3), activation = 'relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(0.25))

#Block 2

model.add(Conv2D(64, kernel_size = (3,3), activation = 'relu'))
model.add(BatchNormalization())
model.add(Conv2D(64, kernel_size = (3,3), activation = 'relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(0.25))

#Block 3

model.add(Conv2D(128, kernel_size = (3,3), activation = 'relu'))
model.add(BatchNormalization())
model.add(Conv2D(128, kernel_size = (3,3), activation = 'relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(0.25))



#Flatten layer

model.add(Flatten())

#Dense layers

model.add(Dense(128, activation = 'relu', kernel_regularizer =
keras.regularizers.l2(0.01)))
model.add(Dropout(0.5))
model.add(Dense(1, activation = 'sigmoid'))
model.summary()
```

```
Model: "sequential_2"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃ Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d_13 (Conv2D)                   │ (None, 222, 222, 32)        │ 896 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization_13               │ (None, 222, 222, 32)        │ 128 │
│ (BatchNormalization)                 │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_14 (Conv2D)                   │ (None, 220, 220, 32)        │ 9,248 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization_14               │ (None, 220, 220, 32)        │ 128 │
│ (BatchNormalization)                 │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_7 (MaxPooling2D)       │ (None, 110, 110, 32)        │ 0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout_9 (Dropout)                  │ (None, 110, 110, 32)        │ 0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_15 (Conv2D)                   │ (None, 108, 108, 64)        │ 18,496 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization_15               │ (None, 108, 108, 64)        │ 256 │
│ (BatchNormalization)                 │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_16 (Conv2D)                   │ (None, 106, 106, 64)        │ 36,928 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
```

```
| batch_normalization_16                | (None, 106, 106, 64)        |
256 |
| (BatchNormalization)                  |                             |
|
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| max_pooling2d_8 (MaxPooling2D)        | (None, 53, 53, 64)          |
0 |
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| dropout_10 (Dropout)                  | (None, 53, 53, 64)          |
0 |
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| conv2d_17 (Conv2D)                    | (None, 51, 51, 128)         |
73,856 |
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| batch_normalization_17                | (None, 51, 51, 128)         |
512 |
| (BatchNormalization)                  |                             |
|
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| conv2d_18 (Conv2D)                    | (None, 49, 49, 128)         |
147,584 |
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| batch_normalization_18                | (None, 49, 49, 128)         |
512 |
| (BatchNormalization)                  |                             |
|
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| max_pooling2d_9 (MaxPooling2D)        | (None, 24, 24, 128)         |
0 |
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| dropout_11 (Dropout)                  | (None, 24, 24, 128)         |
0 |
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| flatten_2 (Flatten)                   | (None, 73728)               |
0 |
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| dense_4 (Dense)                       | (None, 128)                 |
9,437,312 |
├──────────────────────────────────────┼─────────────────────────────┼
```

```
──────────────────┤
│ dropout_12 (Dropout)                    │ (None, 128)                    │
0 │
├─────────────────────────────────────────┼────────────────────────────────┼
──────────────────┤
│ dense_5 (Dense)                         │ (None, 1)                      │
129 │
└─────────────────────────────────────────┴────────────────────────────────┴
──────────────────┘

 Total params: 9,726,241 (37.10 MB)

 Trainable params: 9,725,345 (37.10 MB)

 Non-trainable params: 896 (3.50 KB)

model.compile(optimizer = 'adam',
              loss = 'binary_crossentropy',
              metrics = ['accuracy'])

history = model.fit(train_generator, epochs = 5, batch_size = 64,
validation_data = test_generator)

Epoch 1/5
59/59 ━━━━━━━━━━━━━━━━━━━━ 52s 656ms/step - accuracy: 0.5377 - loss:
13.6902 - val_accuracy: 0.5940 - val_loss: 6.7442
Epoch 2/5
59/59 ━━━━━━━━━━━━━━━━━━━━ 25s 385ms/step - accuracy: 0.5808 - loss:
5.8600 - val_accuracy: 0.4060 - val_loss: 18.3813
Epoch 3/5
59/59 ━━━━━━━━━━━━━━━━━━━━ 25s 383ms/step - accuracy: 0.6506 - loss:
4.0446 - val_accuracy: 0.5940 - val_loss: 3.4718
Epoch 4/5
59/59 ━━━━━━━━━━━━━━━━━━━━ 25s 383ms/step - accuracy: 0.6518 - loss:
3.0382 - val_accuracy: 0.4752 - val_loss: 3.6349
Epoch 5/5
59/59 ━━━━━━━━━━━━━━━━━━━━ 25s 382ms/step - accuracy: 0.6793 - loss:
2.2889 - val_accuracy: 0.5940 - val_loss: 2.2644

history = model.fit(train_generator, epochs = 5, batch_size = 64,
validation_data = test_generator)

Epoch 1/5
59/59 ━━━━━━━━━━━━━━━━━━━━ 25s 365ms/step - accuracy: 0.7042 - loss:
1.7871 - val_accuracy: 0.5961 - val_loss: 2.2076
Epoch 2/5
59/59 ━━━━━━━━━━━━━━━━━━━━ 40s 373ms/step - accuracy: 0.7117 - loss:
1.5605 - val_accuracy: 0.5940 - val_loss: 1.6798
Epoch 3/5
59/59 ━━━━━━━━━━━━━━━━━━━━ 25s 393ms/step - accuracy: 0.7357 - loss:
1.3938 - val_accuracy: 0.6069 - val_loss: 1.5436
```

```
Epoch 4/5
59/59 ━━━━━━━━━━━━━━━━━━━━ 41s 392ms/step - accuracy: 0.7477 - loss:
1.1556 - val_accuracy: 0.6933 - val_loss: 1.1716
Epoch 5/5
59/59 ━━━━━━━━━━━━━━━━━━━━ 25s 380ms/step - accuracy: 0.7445 - loss:
1.0020 - val_accuracy: 0.7646 - val_loss: 1.0005
```

```
history = model.fit(train_generator, epochs = 5, batch_size = 64,
validation_data = test_generator)
```

```
Epoch 1/5
59/59 ━━━━━━━━━━━━━━━━━━━━ 24s 375ms/step - accuracy: 0.7697 - loss:
1.0497 - val_accuracy: 0.7927 - val_loss: 1.0564
Epoch 2/5
59/59 ━━━━━━━━━━━━━━━━━━━━ 24s 366ms/step - accuracy: 0.7689 - loss:
1.0638 - val_accuracy: 0.7689 - val_loss: 1.1063
Epoch 3/5
59/59 ━━━━━━━━━━━━━━━━━━━━ 24s 361ms/step - accuracy: 0.7572 - loss:
1.0632 - val_accuracy: 0.7192 - val_loss: 1.1973
Epoch 4/5
59/59 ━━━━━━━━━━━━━━━━━━━━ 25s 365ms/step - accuracy: 0.7522 - loss:
1.2554 - val_accuracy: 0.7711 - val_loss: 1.3010
Epoch 5/5
59/59 ━━━━━━━━━━━━━━━━━━━━ 40s 373ms/step - accuracy: 0.7693 - loss:
1.1286 - val_accuracy: 0.7495 - val_loss: 1.3508
```

ALexnet

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import os
import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense, Flatten, Dropout, Conv2D,
MaxPooling2D, BatchNormalization
from tensorflow.keras.preprocessing.image import ImageDataGenerator,
load_img, save_img, img_to_array, array_to_img
```

```
df = pd.read_csv('/content/emergency_classification.csv')
df.head()
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 2352,\n  \"fields\":
[\n    {\n      \"column\": \"image_names\",\n      \"properties\": {\
```

```
n        \"dtype\": \"string\",\n        \"num_unique_values\": 2352,\
n        \"samples\": [\n          \"1960.jpg\",\n
\"668.jpg\",\n          \"2082.jpg\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"emergency_or_not\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

```
df['emergency_or_not'].value_counts()
```

```
emergency_or_not
0    1361
1     991
Name: count, dtype: int64
```

```
data_dir = '/content/sample_data/Images'
len(os.listdir(data_dir))
```

```
2352
```

```
aug_dir = '/content/sample_data/aug_images'
os.makedirs(aug_dir, exist_ok = True)

datagen = ImageDataGenerator(rotation_range=0.4,
width_shift_range=0.25,
                             height_shift_range=0.12,
                             shear_range=0.21, zoom_range=0.38,
horizontal_flip=True,
                             vertical_flip=True, rescale=1./255,
                             )

minority_class = df['emergency_or_not'].value_counts().min()
majority_class_count = df['emergency_or_not'].value_counts().max()
num_augmented_images = majority_class_count - minority_class
num_augmented_images
```

```
370
```

```
minority_images = df[df['emergency_or_not'] == 1]['image_names']

current_count = 0

for image_name in minority_images:
  if current_count >= num_augmented_images:
    break

  img_path = os.path.join(data_dir, image_name)
  img = load_img(img_path)
```

```
  img_array = img_to_array(img)
  img_array = np.expand_dims(img_array, axis = 0)

  for _ in range(num_augmented_images - current_count):
    aug_image = next(datagen.flow(img_array, batch_size = 1))
[0].astype('uint8')
    save_img(os.path.join(aug_dir, f'aug_{current_count}.jpg'),
aug_image)
    current_count += 1

    if current_count >= num_augmented_images:
      break

print(f'Generated {num_augmented_images} augmented images.')
```

```
Generated 370 augmented images.
```

data_dir = '/content/sample_data/Images' aug_dir = '/content/sample_data/aug_images' os.makedirs(aug_dir, exist_ok = True)

minority_images = df[df['emergency_or_not' == 1]]['image_names']

minority_class_count = df['emergency_or_not'].value_counts().min() majority_class_count = df['emergency_or_not'].value_counts().max() num_augmented_images = minority_Class_count - majority_class_count

current_count = 0 for image_file in minority_images: if current_count >= num_augmented_images: break

image_path = os.path.join(data_dir, image_file) img = img.load(image_path) img_array = img_to_array(img) img_array = np.expand_dims(img_array, axis = 0)

for _ in range(num_augmented_images - current_count): aug_image = next(datagen.flow(img_array, batch_size = 1))[0].astype('uint8') img.save(os.path.join(output_dir, f'aug_{current_count}.jpg'),aug_image) current_count += 1

```
if current_count >= num_augmented_images:
  break
```

print(f'Generated {num_augmented_images} images')

```
df_1 = pd.DataFrame({'image_names' : [file for file in
os.listdir(aug_dir)],
                     'emergency_or_not' : [1 for i in
range(len(os.listdir(aug_dir)))]})
df_1
```


```
{"summary":"{\n  \"name\": \"df_1\",\n  \"rows\": 370,\n  \"fields\":
[\n    {\n      \"column\": \"image_names\",\n      \"properties\": {\
n        \"dtype\": \"string\",\n        \"num_unique_values\": 370,\n
```

\"samples\": [\n   \"aug_317.jpg\",\n   \"aug_9.jpg\",\n
\"aug_70.jpg\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"emergency_or_not\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0,\n   \"min\": 1,\n
\"max\": 1,\n   \"num_unique_values\": 1,\n   \"samples\":
[\n   1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df_1"}

```
combined_df = pd.concat([df, df_1], ignore_index = True)
combined_df
```

{"summary":"{\n \"name\": \"combined_df\",\n \"rows\": 2722,\n
\"fields\": [\n   {\n   \"column\": \"image_names\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 2722,\n   \"samples\": [\n
\"1061.jpg\",\n   \"408.jpg\",\n   \"aug_94.jpg\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"emergency_or_not\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0,\n   \"min\": 0,\n   \"max\": 1,\n
\"num_unique_values\": 2,\n   \"samples\": [\n   0,\n
1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"combined_df"}

```
combined_df['emergency_or_not'].value_counts()
```

```
emergency_or_not
1    1361
0    1361
Name: count, dtype: int64
```

```
combined_df['image_paths'] = combined_df['image_names'].apply(lambda x
: data_dir + '/' + x if x in df['image_names'].values else aug_dir +
'/' + x)
combined_df
```

{"summary":"{\n \"name\": \"combined_df\",\n \"rows\": 2722,\n
\"fields\": [\n   {\n   \"column\": \"image_names\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 2722,\n   \"samples\": [\n
\"1061.jpg\",\n   \"408.jpg\",\n   \"aug_94.jpg\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"emergency_or_not\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0,\n   \"min\": 0,\n   \"max\": 1,\n
\"num_unique_values\": 2,\n   \"samples\": [\n   0,\n
1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":

```
\"image_paths\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 2722,\n
\"samples\": [\n          \"/content/sample_data/Images/1061.jpg\",\n
\"/content/sample_data/Images/408.jpg\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"combined_df"}

combined_df = combined_df.sample(2722)

combined_df['emergency_or_not'] =
combined_df['emergency_or_not'].astype('str')

train_data = combined_df.sample(frac = 0.8, random_state = 42)
test_data = combined_df.drop(train_data.index)

train_datagen = ImageDataGenerator(rotation_range=0.4,
width_shift_range=0.25,
                                   height_shift_range=0.12,
                                   shear_range=0.21, zoom_range=0.38,
horizontal_flip=True,
                                   vertical_flip=True, rescale=1./255)

test_datagen = ImageDataGenerator(rescale = 1./255)

train_generator = train_datagen.flow_from_dataframe(train_data,
                                                    directory = None,
                                                    x_col =
'image_paths',
                                                    y_col =
'emergency_or_not',
                                                    target_size =
(224,224),
                                                    batch_size = 32,
                                                    class_mode =
'binary')

test_generator = test_datagen.flow_from_dataframe(test_data,
                                                  directory = None,
                                                  x_col =
'image_paths',
                                                  y_col =
'emergency_or_not',
                                                  target_size =
(224,224),
                                                  batch_size = 32,
                                                  class_mode =
'binary')

Found 2178 validated image filenames belonging to 2 classes.
Found 544 validated image filenames belonging to 2 classes.
```

```
# ALexnet model building

model = Sequential()
model.add(Conv2D(32, kernel_size = (5,5), strides = 2, activation =
'relu', kernel_regularizer=keras.regularizers.l2(0.01), input_shape =
(224,224,3)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size = (2,2), strides = 2))
model.add(Dropout(0.25))

model.add(Conv2D(64, kernel_size = (3,3), padding = 'same', activation
= 'relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size = (2,2), strides = 2))
model.add(Dropout(0.25))

model.add(Conv2D(128, kernel_size = (3,3), padding = 'same',
activation = 'relu'))
model.add(BatchNormalization())
model.add(Conv2D(128, kernel_size = (3,3), padding = 'same',
activation = 'relu'))
model.add(BatchNormalization())
model.add(Conv2D(128, kernel_size = (3,3), padding = 'same',
activation = 'relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size = (2,2), strides = 2))
model.add(Dropout(0.30))

model.add(Flatten())

model.add(Dense(128, activation = 'relu', kernel_regularizer =
keras.regularizers.l2(0.01)))
print(model.output_shape)
model.add(Dropout(0.5))
model.add(Dense(1, activation = 'sigmoid'))
model.summary()
```

```
(None, 128)
```

```
Model: "sequential_2"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃ Param #         ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d_10 (Conv2D)                   │ (None, 110, 110, 32)        │           2,432 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
```

```
| batch_normalization_10                | (None, 110, 110, 32)        |
128 |
| (BatchNormalization)                  |                             |
|
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| max_pooling2d_6 (MaxPooling2D)        | (None, 55, 55, 32)          |
0 |
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| dropout_8 (Dropout)                   | (None, 55, 55, 32)          |
0 |
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| conv2d_11 (Conv2D)                    | (None, 55, 55, 64)          |
18,496 |
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| batch_normalization_11                | (None, 55, 55, 64)          |
256 |
| (BatchNormalization)                  |                             |
|
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| max_pooling2d_7 (MaxPooling2D)        | (None, 27, 27, 64)          |
0 |
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| dropout_9 (Dropout)                   | (None, 27, 27, 64)          |
0 |
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| conv2d_12 (Conv2D)                    | (None, 27, 27, 128)         |
73,856 |
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| batch_normalization_12                | (None, 27, 27, 128)         |
512 |
| (BatchNormalization)                  |                             |
|
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| conv2d_13 (Conv2D)                    | (None, 27, 27, 128)         |
147,584 |
├──────────────────────────────────────┼─────────────────────────────┼
──────────────────┤
| batch_normalization_13                | (None, 27, 27, 128)         |
512 |
| (BatchNormalization)                  |                             |
```

```
│
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ conv2d_14 (Conv2D)                   │ (None, 27, 27, 128)         │
147,584 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ batch_normalization_14               │ (None, 27, 27, 128)         │
512 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ max_pooling2d_8 (MaxPooling2D)       │ (None, 13, 13, 128)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ dropout_10 (Dropout)                 │ (None, 13, 13, 128)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ flatten_2 (Flatten)                  │ (None, 21632)               │
0 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ dense_4 (Dense)                      │ (None, 128)                 │
2,769,024 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ dropout_11 (Dropout)                 │ (None, 128)                 │
0 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ dense_5 (Dense)                      │ (None, 1)                   │
129 │
└──────────────────────────────────────┴─────────────────────────────┴
─────────────────┘

 Total params: 3,161,025 (12.06 MB)

 Trainable params: 3,160,065 (12.05 MB)

 Non-trainable params: 960 (3.75 KB)

model.compile(optimizer = 'adam',
              loss = 'binary_crossentropy',
              metrics = ['accuracy'])

history = model.fit(train_generator, epochs = 5, batch_size = 64,
validation_data = test_generator)
```

```
Epoch 1/5
69/69 ━━━━━━━━━━━━━━━━━━━━ 33s 441ms/step - accuracy: 0.6092 - loss:
6.0481 - val_accuracy: 0.5276 - val_loss: 3.0425
Epoch 2/5
69/69 ━━━━━━━━━━━━━━━━━━━━ 27s 350ms/step - accuracy: 0.6405 - loss:
2.7357 - val_accuracy: 0.5276 - val_loss: 3.2858
Epoch 3/5
69/69 ━━━━━━━━━━━━━━━━━━━━ 40s 345ms/step - accuracy: 0.6683 - loss:
2.1260 - val_accuracy: 0.5276 - val_loss: 2.4026
Epoch 4/5
69/69 ━━━━━━━━━━━━━━━━━━━━ 41s 329ms/step - accuracy: 0.6982 - loss:
1.7314 - val_accuracy: 0.5276 - val_loss: 2.0263
Epoch 5/5
69/69 ━━━━━━━━━━━━━━━━━━━━ 25s 330ms/step - accuracy: 0.7148 - loss:
1.3863 - val_accuracy: 0.6618 - val_loss: 1.4901
```

```
history = model.fit(train_generator, epochs = 5, validation_data =
test_generator)
```

```
Epoch 1/5
69/69 ━━━━━━━━━━━━━━━━━━━━ 26s 333ms/step - accuracy: 0.7397 - loss:
1.1060 - val_accuracy: 0.5662 - val_loss: 1.2707
Epoch 2/5
69/69 ━━━━━━━━━━━━━━━━━━━━ 40s 334ms/step - accuracy: 0.7187 - loss:
1.2497 - val_accuracy: 0.5625 - val_loss: 2.0264
Epoch 3/5
69/69 ━━━━━━━━━━━━━━━━━━━━ 29s 384ms/step - accuracy: 0.7002 - loss:
1.3348 - val_accuracy: 0.5331 - val_loss: 2.2575
Epoch 4/5
69/69 ━━━━━━━━━━━━━━━━━━━━ 27s 357ms/step - accuracy: 0.7414 - loss:
1.0142 - val_accuracy: 0.6746 - val_loss: 1.4489
Epoch 5/5
69/69 ━━━━━━━━━━━━━━━━━━━━ 27s 356ms/step - accuracy: 0.7544 - loss:
0.8914 - val_accuracy: 0.6618 - val_loss: 1.6831
```