In [53]:

```
# Utils
import os
import numpy as np
import pandas as pd
import cv2
from IPython.display import clear_output as cls
import imutils
import zipfile

# Modeling
import tensorflow as tf
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, GlobalAveragePooling2D as
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential, load_model
from keras.callbacks import EarlyStopping, ModelCheckpoint

# Visualization
import plotly.express as px
import matplotlib.pyplot as plt

# pre-trained model
from keras.applications import ResNet50V2

import warnings
warnings.filterwarnings('ignore')
import numpy as np
import matplotlib.pyplot as plt
import os
import math
import shutil
import glob
import cv2
import imutils
import seaborn as sns
from sklearn.utils import shuffle

from keras.layers import Conv2D, MaxPool2D, Dropout, Flatten, Dense, BatchNormalization,
from keras.models import Model, Sequential
import keras
!pip install visualkeras
import os
import warnings
import itertools
import cv2
import seaborn as sns
import pandas as pd
import numpy  as np
from PIL import Image
from sklearn.utils import class_weight
from sklearn.metrics import confusion_matrix, classification_report
from collections import Counter

import tensorflow as tf
import tensorflow_addons as tfa
import visualkeras
import plotly.express as px
import matplotlib.pyplot as plt
from sklearn.metrics import multilabel_confusion_matrix
```

```
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import plot_model
from tensorflow.keras import layers
from tensorflow.keras import regularizers
from sklearn.model_selection   import train_test_split
from keras.preprocessing.image import ImageDataGenerator

warnings.filterwarnings('ignore')
%matplotlib inline
```

```
Requirement already satisfied: visualkeras in c:\users\pirat\anaconda3\lib
\site-packages (0.0.2)
Requirement already satisfied: numpy>=1.18.1 in c:\users\pirat\anaconda3\l
ib\site-packages (from visualkeras) (1.21.5)
Requirement already satisfied: aggdraw>=1.3.11 in c:\users\pirat\anaconda3
\lib\site-packages (from visualkeras) (1.3.16)
Requirement already satisfied: pillow>=6.2.0 in c:\users\pirat\anaconda3\l
ib\site-packages (from visualkeras) (9.2.0)
```

In [ ]:

```

```

In [5]:

```
local_zip = './archive.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('./')
zip_ref.close()
```

In [6]:

```
# defining main directory
main_directory = 'brain_tumor_dataset/'
# checkin classes names
class_names = sorted(os.listdir(main_directory))
# checking number of classes
n_classes = len(class_names)

# class distribution
class_diss = [len(os.listdir(main_directory + name)) for name in class_names]
print(f"Total Number of classes : {n_classes} \nClasse Names : {class_names}")
```

```
Total Number of classes : 2
Classe Names : ['no', 'yes']
```

In [7]:

```
# Visualizing class distribution
fig = px.pie(names=class_names, values=class_diss, title="Class Distribution", hole=0.4)
fig.update_layout({'title':{'x':0.5}})
fig.show()
```

In [8]:

```
def crop_brain_contour(image, plot=False):
    # Convert the image to grayscale, and blur it slightly
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray, (5, 5), 0)

    thresh = cv2.threshold(gray, 45, 255, cv2.THRESH_BINARY)[1]
    thresh = cv2.erode(thresh, None, iterations=2)
    thresh = cv2.dilate(thresh, None, iterations=2)

    # Find contours in thresholded image, then grab the largest one
    cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(cnts)
    c = max(cnts, key=cv2.contourArea)
    # extreme points
    extLeft = tuple(c[c[:, :, 0].argmin()][0])
    extRight = tuple(c[c[:, :, 0].argmax()][0])
    extTop = tuple(c[c[:, :, 1].argmin()][0])
    extBot = tuple(c[c[:, :, 1].argmax()][0])

    # crop new image out of the original image using the four extreme points (left, righ
    new_image = image[extTop[1]:extBot[1], extLeft[0]:extRight[0]]

    if plot:
        plt.figure()
        plt.subplot(1, 2, 1)
        plt.imshow(image)
        plt.tick_params(axis='both', which='both', top=False, bottom=False, left=False,
        plt.title('Original Image')
        plt.subplot(1, 2, 2)
        plt.imshow(new_image)
        plt.tick_params(axis='both', which='both',top=False, bottom=False, left=False, r
        plt.title('Cropped Image')
        plt.show()

    return new_image
```

In [9]:

```
# resizing and cropping images
subdir_list = os.listdir(main_directory)

target_size = (630, 630)

for subdir in subdir_list:
    subdir_path = os.path.join(main_directory, subdir)
    if os.path.isdir(subdir_path):
        image_list = os.listdir(subdir_path)
        for image_name in image_list:
            # read image
            image = cv2.imread(os.path.join(subdir_path, image_name))
            # resize image
            image = cv2.resize(image, target_size)
            # crop image
            image = crop_brain_contour(image)
            # replacing the old image in directory
            cv2.imwrite(os.path.join(subdir_path, image_name), image)
```

In [10]:

```
# data loading and augmentation
datagen = ImageDataGenerator(rotation_range=15,
                             width_shift_range=0.05,
                             height_shift_range=0.05,
                             rescale=1./255,
                             shear_range=0.05,
                             brightness_range=[0.1, 1.5],
                             horizontal_flip=True,
                             vertical_flip=True,
                             validation_split=0.3)

# loading training set
train = datagen.flow_from_directory(main_directory, class_mode='binary', target_size=(63
subset='training')

# loading validation set
validation = datagen.flow_from_directory(main_directory, class_mode='binary', target_siz
subset='validation')
```

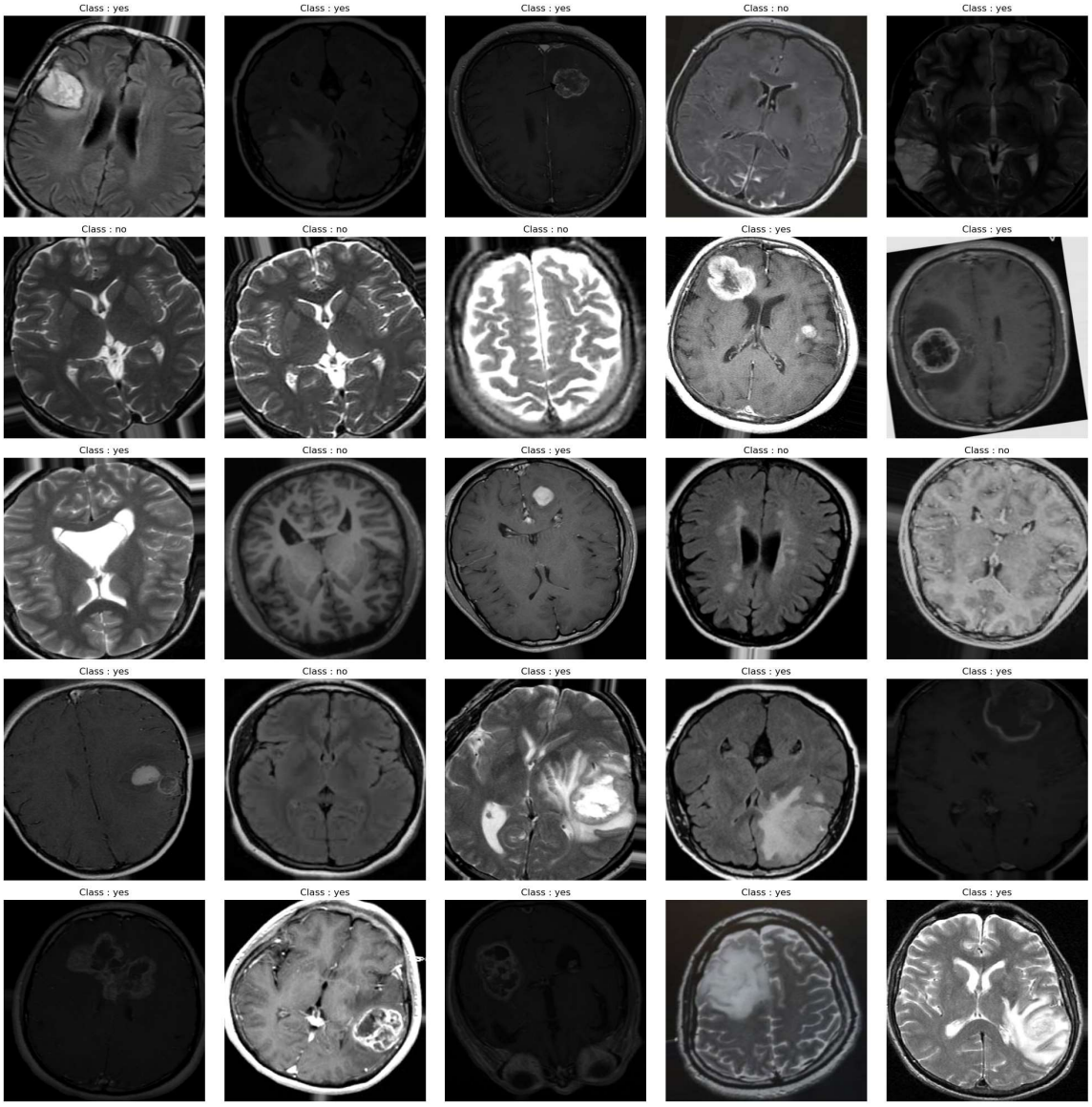
```
Found 178 images belonging to 2 classes.
Found 75 images belonging to 2 classes.
```

In [14]:

```
def show_images(GRID= [5,5], model=None, size = (20,20), data=train):
    n_rows = GRID[0]
    n_cols = GRID[1]
    n_images = n_rows * n_cols

    i = 1
    plt.figure(figsize=size)
    for images, labels in data:
        id = np.random.randint(len(images))
        image, label = images[id], class_names[int(labels[id])]
        plt.subplot(n_rows, n_cols, i)
        plt.imshow(image)

        if model is None:
            title = f"Class : {label}"
        else:
            pred = class_names[int(np.argmax(model.predict(image[np.newaxis, ...])))]
            title = f"Org : {label}, Pred : {pred}"
            cls()

        plt.title(title)
        plt.axis('off')

        i+=1
        if i>=(n_images+1):
            break

    plt.tight_layout()
    plt.show()
```

In [15]:

```
show_images()
```



In [16]:

```
# defining the base pre-trained model
base_model = ResNet50V2(input_shape=(630,630,3), include_top=False)
base_model.trainable = False

# defining our model
name = "ResNet50V2"
model = Sequential([
    base_model,
    GAP(),
    Dense(256, activation='relu', kernel_initializer='he_normal'),
    Dense(n_classes, activation='softmax'),
], name=name)

# compiling model
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accura

# training model
history = model.fit(train, validation_data=validation, epochs=30)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-appl
ications/resnet/resnet50v2_weights_tf_dim_ordering_tf_kernels_notop.h5 (ht
tps://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50
v2_weights_tf_dim_ordering_tf_kernels_notop.h5)
94668760/94668760 [==============================] - 15s 0us/step
Epoch 1/30
12/12 [==============================] - 74s 6s/step - loss: 0.5900 - accu
racy: 0.6966 - val_loss: 0.6019 - val_accuracy: 0.6800
Epoch 2/30
12/12 [==============================] - 74s 6s/step - loss: 0.4859 - accu
racy: 0.7640 - val_loss: 0.4352 - val_accuracy: 0.8000
Epoch 3/30
12/12 [==============================] - 75s 6s/step - loss: 0.3910 - accu
racy: 0.8202 - val_loss: 0.4139 - val_accuracy: 0.8000
Epoch 4/30
12/12 [==============================] - 73s 6s/step - loss: 0.4067 - accu
racy: 0.8539 - val_loss: 0.4858 - val_accuracy: 0.7867
Epoch 5/30
12/12 [==============================] - 72s 6s/step - loss: 0.2765 - accu
racy: 0.8708 - val_loss: 0.3928 - val_accuracy: 0.8667
Epoch 6/30
12/12 [==============================] - 73s 6s/step - loss: 0.2050 - accu
racy: 0.9213 - val_loss: 0.3028 - val_accuracy: 0.8800
Epoch 7/30
12/12 [==============================] - 74s 6s/step - loss: 0.2240 - accu
racy: 0.9157 - val_loss: 0.3712 - val_accuracy: 0.8133
Epoch 8/30
12/12 [==============================] - 74s 6s/step - loss: 0.3310 - accu
racy: 0.8652 - val_loss: 0.3366 - val_accuracy: 0.8933
Epoch 9/30
12/12 [==============================] - 73s 6s/step - loss: 0.1536 - accu
racy: 0.9551 - val_loss: 0.3761 - val_accuracy: 0.8533
Epoch 10/30
12/12 [==============================] - 73s 6s/step - loss: 0.1640 - accu
racy: 0.9326 - val_loss: 0.2974 - val_accuracy: 0.8667
Epoch 11/30
12/12 [==============================] - 72s 7s/step - loss: 0.1569 - accu
racy: 0.9438 - val_loss: 0.3880 - val_accuracy: 0.8133
Epoch 12/30
12/12 [==============================] - 73s 6s/step - loss: 0.1907 - accu
racy: 0.9213 - val_loss: 0.3456 - val_accuracy: 0.8133
Epoch 13/30
12/12 [==============================] - 75s 6s/step - loss: 0.1450 - accu
racy: 0.9438 - val_loss: 0.5141 - val_accuracy: 0.7867
Epoch 14/30
12/12 [==============================] - 72s 6s/step - loss: 0.1348 - accu
racy: 0.9382 - val_loss: 0.3823 - val_accuracy: 0.8400
Epoch 15/30
12/12 [==============================] - 73s 6s/step - loss: 0.1349 - accu
racy: 0.9551 - val_loss: 0.2597 - val_accuracy: 0.8533
Epoch 16/30
12/12 [==============================] - 73s 6s/step - loss: 0.1177 - accu
racy: 0.9663 - val_loss: 0.3127 - val_accuracy: 0.8400
Epoch 17/30
12/12 [==============================] - 74s 6s/step - loss: 0.1280 - accu
racy: 0.9494 - val_loss: 0.3212 - val_accuracy: 0.8400
Epoch 18/30
12/12 [==============================] - 77s 6s/step - loss: 0.0950 - accu
racy: 0.9663 - val_loss: 0.2670 - val_accuracy: 0.9067
Epoch 19/30
12/12 [==============================] - 74s 6s/step - loss: 0.1092 - accu
```

```
racy: 0.9719 - val_loss: 0.2539 - val_accuracy: 0.9200
Epoch 20/30
12/12 [==============================] - 72s 7s/step - loss: 0.1090 - accu
racy: 0.9663 - val_loss: 0.2571 - val_accuracy: 0.8933
Epoch 21/30
12/12 [==============================] - 73s 6s/step - loss: 0.1380 - accu
racy: 0.9382 - val_loss: 0.2897 - val_accuracy: 0.8667
Epoch 22/30
12/12 [==============================] - 73s 6s/step - loss: 0.1784 - accu
racy: 0.9438 - val_loss: 0.3085 - val_accuracy: 0.8667
Epoch 23/30
12/12 [==============================] - 72s 6s/step - loss: 0.1624 - accu
racy: 0.9551 - val_loss: 0.2719 - val_accuracy: 0.8800
Epoch 24/30
12/12 [==============================] - 74s 6s/step - loss: 0.1012 - accu
racy: 0.9663 - val_loss: 0.3910 - val_accuracy: 0.8667
Epoch 25/30
12/12 [==============================] - 74s 6s/step - loss: 0.0940 - accu
racy: 0.9663 - val_loss: 0.5221 - val_accuracy: 0.8267
Epoch 26/30
12/12 [==============================] - 74s 6s/step - loss: 0.1352 - accu
racy: 0.9438 - val_loss: 0.5493 - val_accuracy: 0.8133
Epoch 27/30
12/12 [==============================] - 74s 6s/step - loss: 0.2140 - accu
racy: 0.9157 - val_loss: 0.4584 - val_accuracy: 0.8533
Epoch 28/30
12/12 [==============================] - 74s 6s/step - loss: 0.1501 - accu
racy: 0.9494 - val_loss: 0.6198 - val_accuracy: 0.7600
Epoch 29/30
12/12 [==============================] - 74s 6s/step - loss: 0.1404 - accu
racy: 0.9326 - val_loss: 0.8031 - val_accuracy: 0.7600
Epoch 30/30
12/12 [==============================] - 74s 6s/step - loss: 0.1524 - accu
racy: 0.9382 - val_loss: 0.5827 - val_accuracy: 0.8400
```

In [17]:

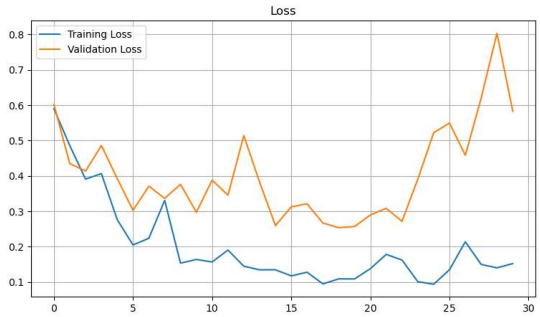
```
data = pd.DataFrame(history.history)
data[-5:]
```

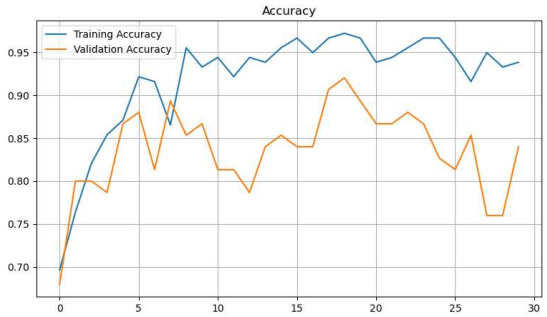
Out[17]:

	loss	accuracy	val_loss	val_accuracy
25	0.135167	0.943820	0.549306	0.813333
26	0.214014	0.915730	0.458418	0.853333
27	0.150073	0.949438	0.619797	0.760000
28	0.140351	0.932584	0.803087	0.760000
29	0.152440	0.938202	0.582651	0.840000

In [18]:

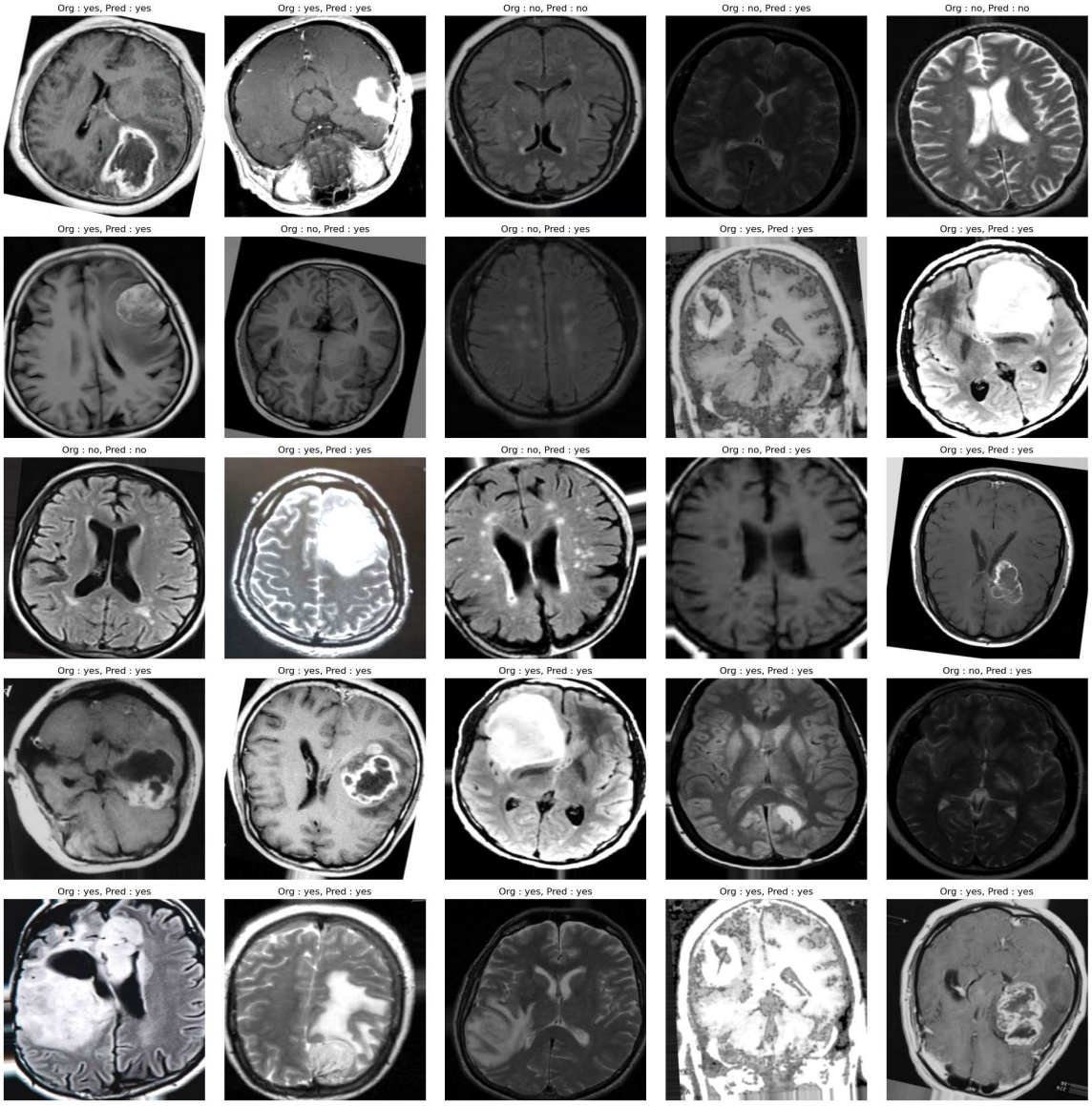
```
# plotting training / validation history
plt.figure(figsize=(20,5))

plt.subplot(1,2,1)
plt.plot(data.loss, label='Training Loss')
plt.plot(data.val_loss, label='Validation Loss')
plt.title("Loss")
plt.grid()
plt.legend()

plt.subplot(1,2,2)
plt.plot(data.accuracy, label='Training Accuracy')
plt.plot(data.val_accuracy, label='Validation Accuracy')
plt.title("Accuracy")
plt.grid()
plt.legend()

plt.show()
```





In [19]:

```
# plotting predictions
show_images(model=model, data=validation)
```



In [54]:

```
# General parameters
epochs = 15
pic_size = 240
np.random.seed(42)
tf.random.set_seed(42)
```

In [56]:

```
folder_path = "C:/Users/pirat/Desktop/Brain tumor/brain_tumor_dataset"
no_images = os.listdir(folder_path + '/no/')
yes_images = os.listdir(folder_path + '/yes/')
dataset=[]
lab=[]

for image_name in no_images:
    image=cv2.imread(folder_path + '/no/' + image_name)
    image=Image.fromarray(image,'RGB')
    image=image.resize((240,240))
    dataset.append(np.array(image))
    lab.append(0)

for image_name in yes_images:
    image=cv2.imread(folder_path + '/yes/' + image_name)
    image=Image.fromarray(image,'RGB')
    image=image.resize((240,240))
    dataset.append(np.array(image))
    lab.append(1)
```

In [57]:

```
dataset = np.array(dataset)
lab = np.array(lab)
print(dataset.shape, lab.shape)
```

```
(253, 240, 240, 3) (253,)
```

In [58]:

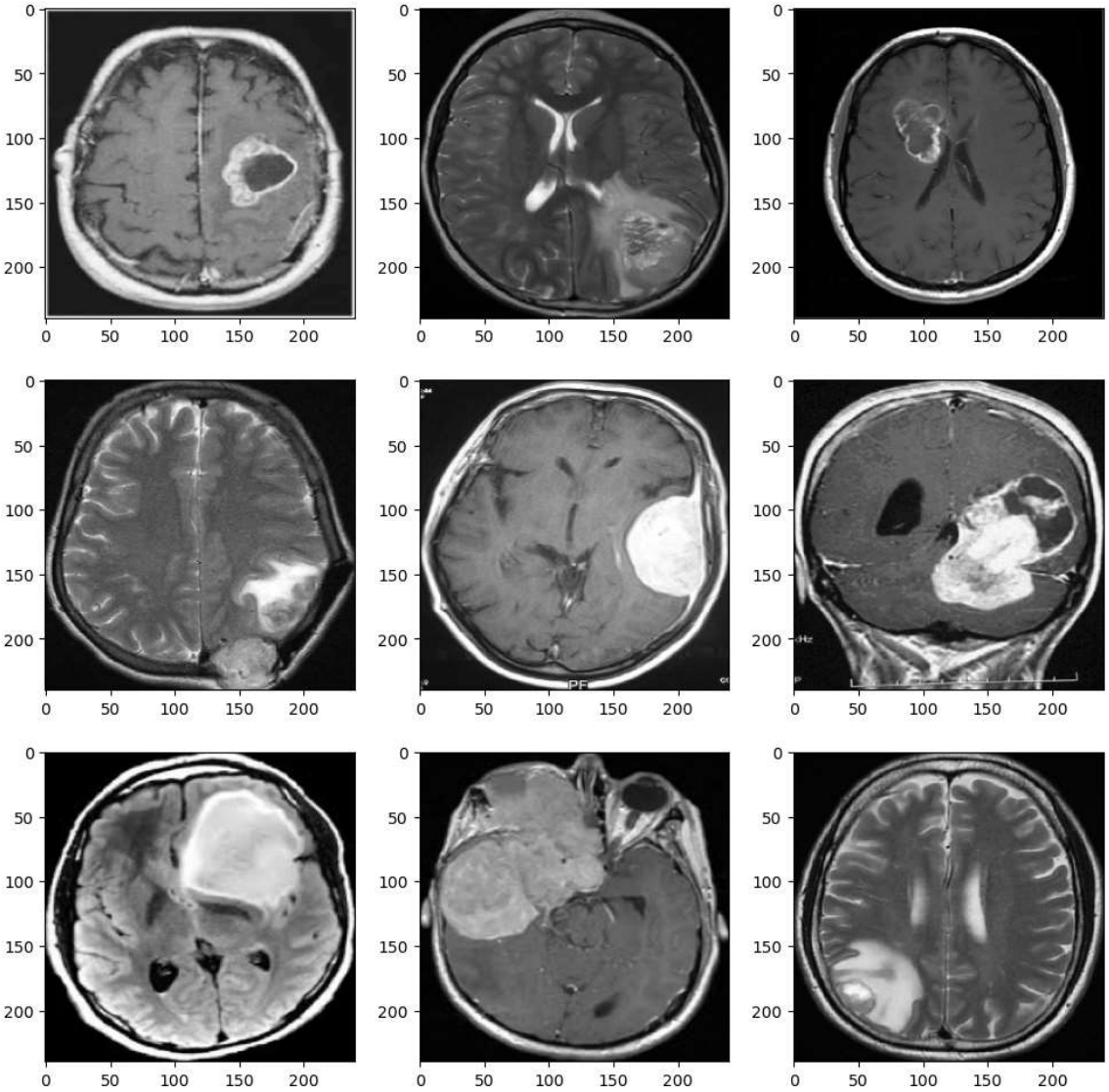
```
x_train, x_test, y_train, y_test = train_test_split(dataset, lab, test_size=0.2, shuffle
```

In [59]:

```
def plot_state(state):
    plt.figure(figsize= (12,12))
    for i in range(1, 10, 1):
        plt.subplot(3,3,i)
        img = load_img(folder_path + "/" + state + "/" + os.listdir(folder_path + "/" +
        plt.imshow(img)
    plt.show()
```

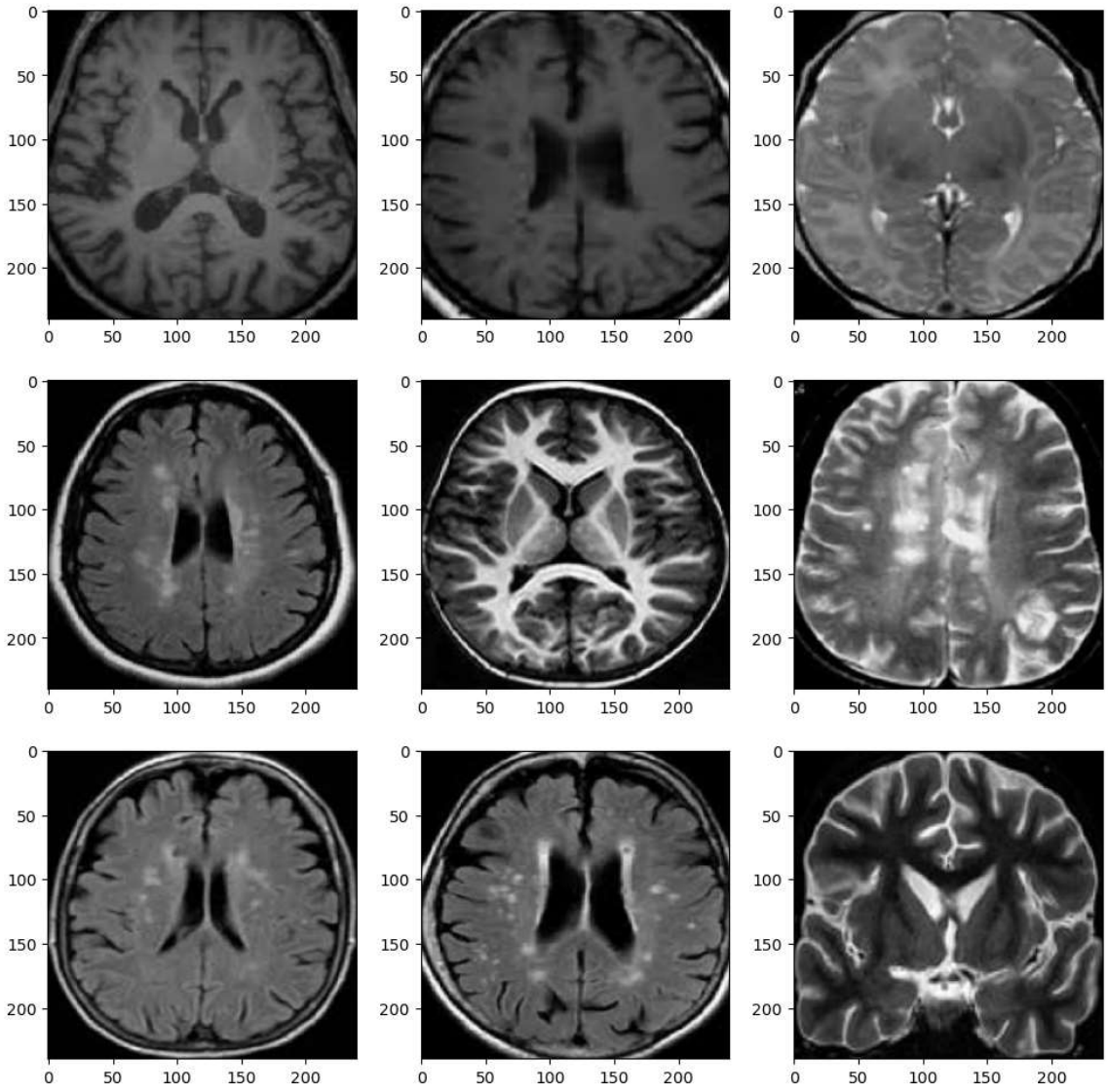
In [60]:

```
plot_state('yes')
```



In [61]:

```
plot_state("no")
```



Modeling using CNN

In [62]:

```
model = tf.keras.Sequential([

    tf.keras.layers.Conv2D(filters=32,kernel_size=(3,3),strides=(2,2), activation="relu"
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(filters=32,kernel_size=(3,3),strides=(2,2), activation="relu"
    tf.keras.layers.MaxPooling2D((2, 2)),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(units=64, activation='relu',
                          kernel_regularizer=regularizers.L1L2(l1=1e-3, l2=1e-3),
                          bias_regularizer=regularizers.L2(1e-2),
                          activity_regularizer=regularizers.L2(1e-3)),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(units=1, activation='sigmoid'),
])
```

In [63]:

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

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 119, 119, 32)      896

 max_pooling2d_3 (MaxPooling  (None, 59, 59, 32)       0
 2D)

 conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248

 max_pooling2d_4 (MaxPooling  (None, 14, 14, 32)       0
 2D)

 flatten_2 (Flatten)         (None, 6272)              0

 dense_4 (Dense)             (None, 64)                401472

 dropout (Dropout)           (None, 64)                0

 dense_5 (Dense)             (None, 1)                 65

=================================================================
Total params: 411,681
Trainable params: 411,681
Non-trainable params: 0
_________________________________________________________________
```

In [67]:

```
class_weights = class_weight.compute_class_weight(class_weight = "balanced", classes= np
class_weights = dict(zip(np.unique(y_train), class_weights))
class_weights
```

Out[67]:

```
{0: 1.294871794871795, 1: 0.8145161290322581}
```

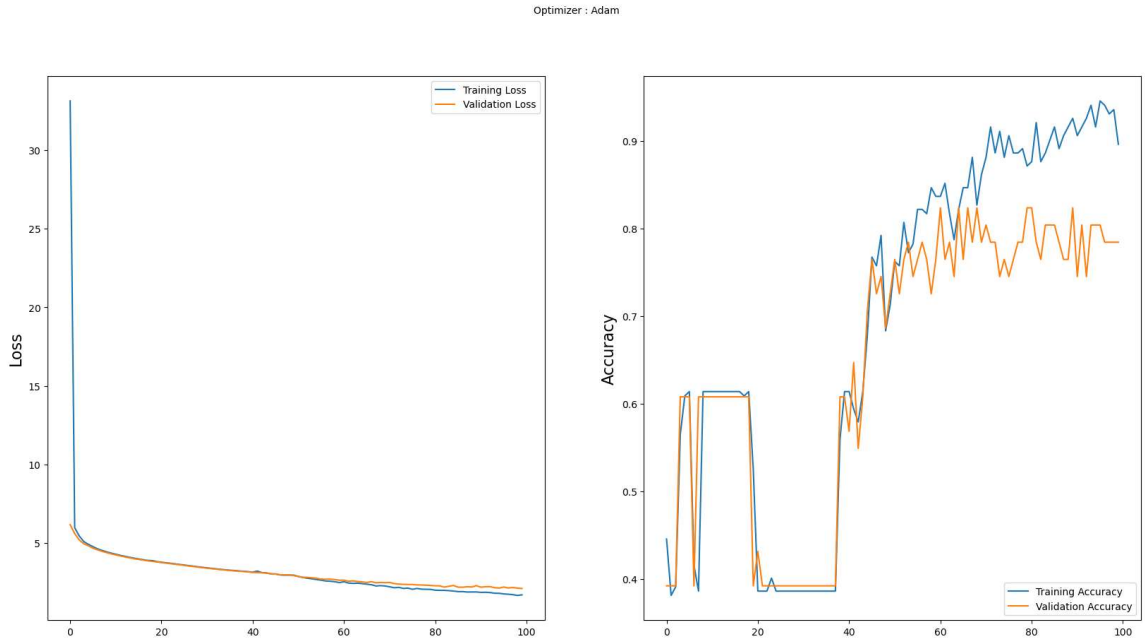
In [68]:

```
history = model.fit(x_train,y_train,epochs = 100, class_weight=class_weights, validation
```

```
ccuracy: 0.9406 - val_loss: 2.1689 - val_accuracy: 0.8039
Epoch 95/100
7/7 [==============================] - 1s 131ms/step - loss: 1.8132 - a
ccuracy: 0.9158 - val_loss: 2.1418 - val_accuracy: 0.8039
Epoch 96/100
7/7 [==============================] - 1s 142ms/step - loss: 1.7735 - a
ccuracy: 0.9455 - val_loss: 2.2111 - val_accuracy: 0.8039
Epoch 97/100
7/7 [==============================] - 1s 144ms/step - loss: 1.7526 - a
ccuracy: 0.9406 - val_loss: 2.1497 - val_accuracy: 0.7843
Epoch 98/100
7/7 [==============================] - 1s 135ms/step - loss: 1.7279 - a
ccuracy: 0.9307 - val_loss: 2.1802 - val_accuracy: 0.7843
Epoch 99/100
7/7 [==============================] - 1s 134ms/step - loss: 1.6753 - a
ccuracy: 0.9356 - val_loss: 2.1411 - val_accuracy: 0.7843
Epoch 100/100
7/7 [==============================] - 1s 136ms/step - loss: 1.7139 - a
ccuracy: 0.8960 - val_loss: 2.1148 - val_accuracy: 0.7843
```

In [69]:

```
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : Adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```



In [70]:

```
results = model.evaluate(x_test, y_test)
print('The current model achieved an accuracy of {}%!'.format(round(results[1]*100,2)))
```

```
2/2 [==============================] - 0s 23ms/step - loss: 2.1148 - accur
acy: 0.7843
The current model achieved an accuracy of 78.43%!
```

In [71]:

```
# compute predictions
predictions = model.predict(x_test)
y_pred = []
for i in predictions:
    if i >= 0.5:
        y_pred.append(1)
    else:
        y_pred.append(0)
```

```
2/2 [==============================] - 0s 21ms/step
```

Modeling using Vision Transformers(ViT)

In [73]:

```
learning_rate = 0.001
weight_decay = 0.0001
batch_size = 256
num_epochs = 100
image_size = 240  # We'll resize input images to this size
patch_size = 20  # Size of the patches to be extract from the input images
num_patches = (image_size // patch_size) ** 2
projection_dim = 64
num_heads = 4
transformer_units = [
    projection_dim * 2,
    projection_dim,
]  # Size of the transformer layers
transformer_layers = 8
mlp_head_units = [2048, 1024]  # Size of the dense layers of the final classifier
```

In [74]:

```
data_augmentation = tf.keras.Sequential(
    [
        layers.Normalization(),
        layers.Resizing(image_size, image_size),
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(factor=0.02),
        layers.RandomZoom(
            height_factor=0.2, width_factor=0.2
        ),
    ],
    name="data_augmentation",
)
# Compute the mean and the variance of the training data for normalization.
data_augmentation.layers[0].adapt(x_train)
```

In [75]:

```
def mlp(x, hidden_units, dropout_rate):
    for units in hidden_units:
        x = layers.Dense(units, activation=tf.nn.gelu)(x)
        x = layers.Dropout(dropout_rate)(x)
    return x
```

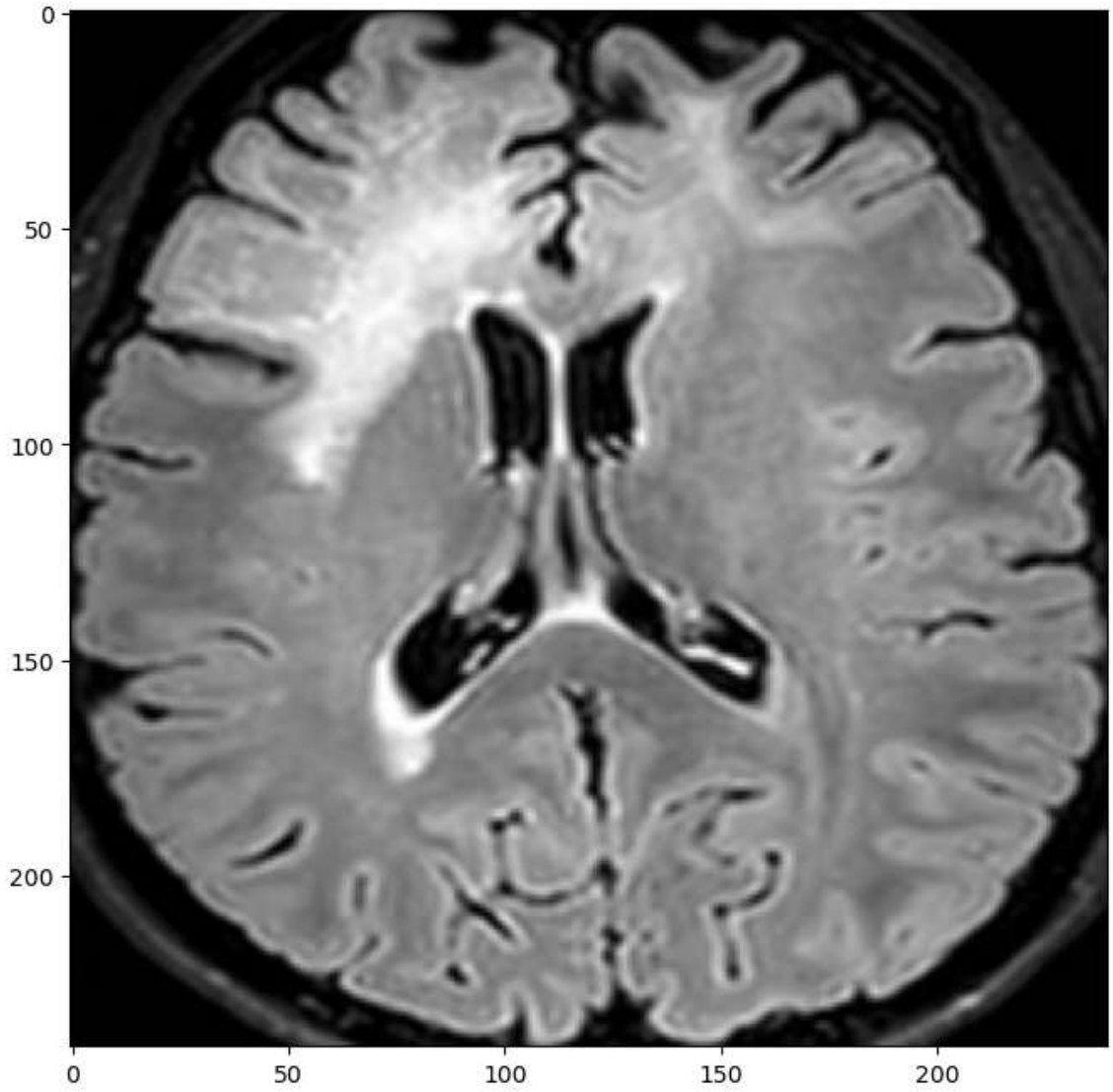
In [76]:

```
class Patches(layers.Layer):
    def __init__(self, patch_size):
        super(Patches, self).__init__()
        self.patch_size = patch_size

    def call(self, images):
        batch_size = tf.shape(images)[0]
        patches = tf.image.extract_patches(
            images=images,
            sizes=[1, self.patch_size, self.patch_size, 1],
            strides=[1, self.patch_size, self.patch_size, 1],
            rates=[1, 1, 1, 1],
            padding="VALID",
        )
        patch_dims = patches.shape[-1]
        patches = tf.reshape(patches, [batch_size, -1, patch_dims])
        return patches
```

In [77]:

```
plt.figure(figsize=(8, 8))
image = x_train[np.random.choice(range(x_train.shape[0]))]
plt.imshow(image.astype("uint8"))

resized_image = tf.image.resize(
    tf.convert_to_tensor([image]), size=(image_size, image_size)
)
patches = Patches(patch_size)(resized_image)
print(f"Image size: {image_size} X {image_size}")
print(f"Patch size: {patch_size} X {patch_size}")
print(f"Patches per image: {patches.shape[1]}")
print(f"Elements per patch: {patches.shape[-1]}")

n = int(np.sqrt(patches.shape[1]))
plt.figure(figsize=(8, 8))
for i, patch in enumerate(patches[0]):
    ax = plt.subplot(n, n, i + 1)
    patch_img = tf.reshape(patch, (patch_size, patch_size, 3))
    plt.imshow(patch_img.numpy().astype("uint8"))
    plt.axis("off")
```

```
Image size: 240 X 240
Patch size: 20 X 20
Patches per image: 144
Elements per patch: 1200
```



In [78]:

```
class PatchEncoder(tf.keras.layers.Layer):
    def __init__(self, num_patches, projection_dim):
        super(PatchEncoder, self).__init__()
        self.num_patches = num_patches
        self.projection = layers.Dense(units=projection_dim)
        self.position_embedding = layers.Embedding(
            input_dim=num_patches, output_dim=projection_dim
        )

    def call(self, patch):
        positions = tf.range(start=0, limit=self.num_patches, delta=1)
        encoded = self.projection(patch) + self.position_embedding(positions)
        return encoded
```

```
def create_vit_classifier():
    inputs = layers.Input(shape=(240, 240, 3))
    # Augment data.
    augmented = data_augmentation(inputs)
    # Create patches.
    patches = Patches(patch_size)(augmented)
    # Encode patches.
    encoded_patches = PatchEncoder(num_patches, projection_dim)(patches)

    # Create multiple layers of the Transformer block.
    for _ in range(transformer_layers):
        # Layer normalization 1.
        x1 = layers.LayerNormalization(epsilon=1e-6)(encoded_patches)
        # Create a multi-head attention layer.
        attention_output = layers.MultiHeadAttention(
            num_heads=num_heads, key_dim=projection_dim, dropout=0.1
        )(x1, x1)
        # Skip connection 1.
        x2 = layers.Add()([attention_output, encoded_patches])
        # Layer normalization 2.
        x3 = layers.LayerNormalization(epsilon=1e-6)(x2)
        # MLP.
        x3 = mlp(x3, hidden_units=transformer_units, dropout_rate=0.1)
        # Skip connection 2.
        encoded_patches = layers.Add()([x3, x2])

    # Create a [batch_size, projection_dim] tensor.
    representation = layers.LayerNormalization(epsilon=1e-6)(encoded_patches)
    representation = layers.Flatten()(representation)
    representation = layers.Dropout(0.5)(representation)
    # Add MLP.
    features = mlp(representation, hidden_units=mlp_head_units, dropout_rate=0.5)
    # Classify outputs.
    logits = layers.Dense(2)(features)
    # Create the Keras model.
    model = tf.keras.Model(inputs=inputs, outputs=logits)
    return model
```

In [80]:

```
def run_experiment(model):
    optimizer = tfa.optimizers.AdamW(
        learning_rate=learning_rate, weight_decay=weight_decay
    )

    model.compile(
        optimizer=optimizer,
        loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
        metrics=[
            tf.keras.metrics.SparseCategoricalAccuracy(name="accuracy"),
            tf.keras.metrics.SparseTopKCategoricalAccuracy(5, name="top-5-accuracy"),
        ],
    )

    checkpoint_filepath = "/tmp/checkpoint"
    checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
        checkpoint_filepath,
        monitor="val_accuracy",
        save_best_only=True,
        save_weights_only=True,
    )

    history = model.fit(
        x=x_train,
        y=y_train,
        batch_size=batch_size,
        epochs=num_epochs,
        validation_data=(x_test, y_test),
        callbacks=[checkpoint_callback],
    )

    model.load_weights(checkpoint_filepath)
    _, accuracy, top_5_accuracy = model.evaluate(x_test, y_test)
    print(f"Test accuracy: {round(accuracy * 100, 2)}%")
    print(f"Test top 5 accuracy: {round(top_5_accuracy * 100, 2)}%")

    return history
```

In [81]:

```
vit_classifier = create_vit_classifier()
vit_history = run_experiment(vit_classifier)
```

```
Epoch 1/100
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause
there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there i
s no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there i
s no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUnifor
mV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransf
ormV3 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause
there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there i
s no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there i
s no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUnifor
mV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause
there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there i
s no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there i
s no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUnifor
mV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransf
ormV3 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause
there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there i
s no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there i
s no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUnifor
mV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransf
ormV3 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause
there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there i
s no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there i
s no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUnifor
mV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause
there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there i
s no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there i
s no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUnifor
mV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransf
ormV3 cause there is no registered converter for this op.
1/1 [==============================] - ETA: 0s - loss: 1.4120 - accuracy:
0.5347 - top-5-accuracy: 1.0000
```

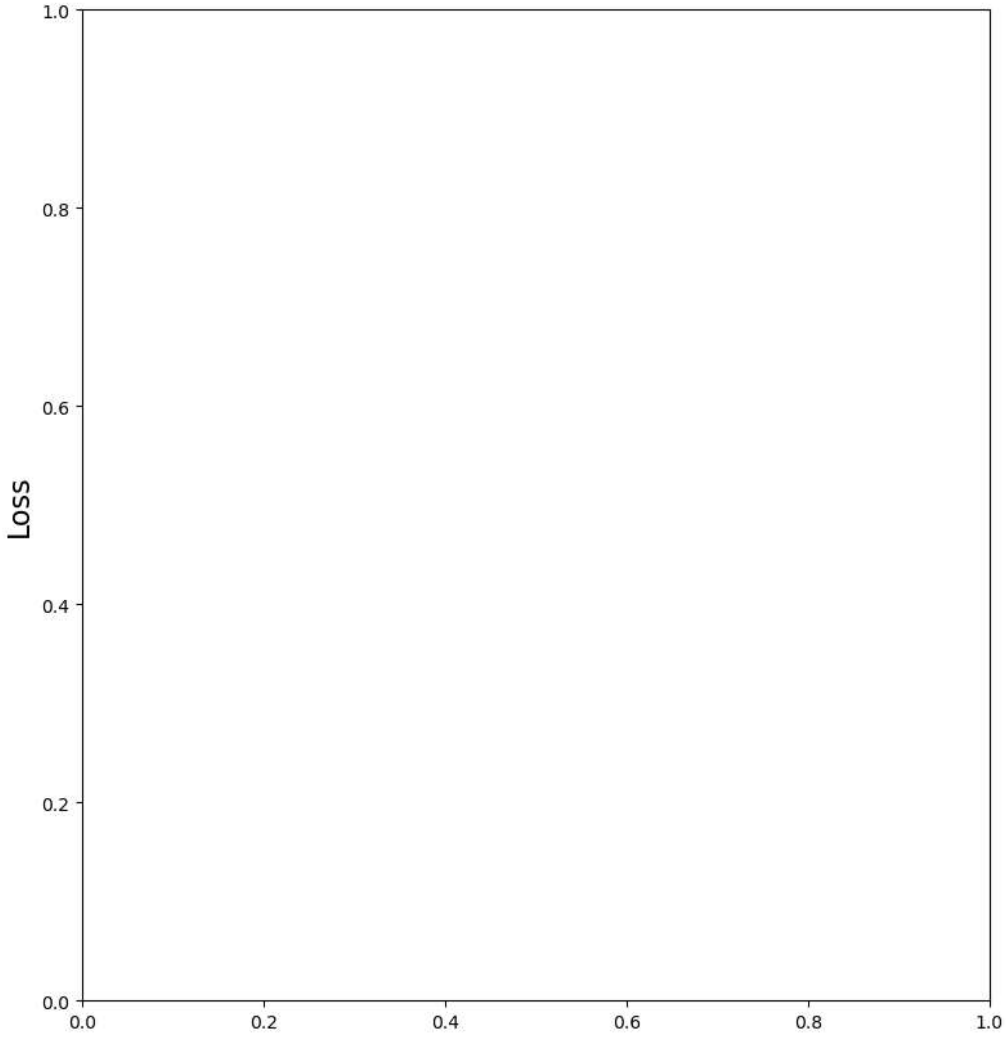
```
---------------------------------------------------------------------------
-
PermissionDeniedError                     Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_1976\2084520009.py in <module>
      1 vit_classifier = create_vit_classifier()
----> 2 vit_history = run_experiment(vit_classifier)

~\AppData\Local\Temp\ipykernel_1976\3601100103.py in run_experiment(model)
     21     )
     22
---> 23     history = model.fit(
     24         x=x_train,
     25         y=y_train,

~\anaconda3\lib\site-packages\keras\utils\traceback_utils.py in error_hand
ler(*args, **kwargs)
     68             # To get the full stack trace, call:
     69             # `tf.debugging.disable_traceback_filtering()`
---> 70             raise e.with_traceback(filtered_tb) from None
     71         finally:
     72             del filtered_tb

~\anaconda3\lib\site-packages\tensorflow\python\eager\execute.py in quick_
execute(op_name, num_outputs, inputs, attrs, ctx, name)
     52   try:
     53     ctx.ensure_initialized()
---> 54     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name,
op_name,
     55                                         inputs, attrs, num_output
s)
     56   except core._NotOkStatusException as e:

PermissionDeniedError: {{function_node __wrapped__MergeV2Checkpoints_devic
e_/job:localhost/replica:0/task:0/device:CPU:0}} Failed to create a direct
ory: /; Permission denied [Op:MergeV2Checkpoints]
```

In [82]:

```
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : Adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(vit_history.history['loss'], label='Training Loss')
plt.plot(vit_history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(vit_history.history['accuracy'], label='Training Accuracy')
plt.plot(vit_history.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_1976\1375200356.py in <module>
      3 plt.suptitle('Optimizer : Adam', fontsize=10)
      4 plt.ylabel('Loss', fontsize=16)
----> 5 plt.plot(vit_history.history['loss'], label='Training Loss')
      6 plt.plot(vit_history.history['val_loss'], label='Validation Loss')
      7 plt.legend(loc='upper right')

NameError: name 'vit_history' is not defined
```

Optimizer : Adam



CNN Model

In [33]:

```
#Build our model
model = Sequential()

model.add(Conv2D(filters = 16, kernel_size = (3,3), activation = 'relu', input_shape = (

model.add(Conv2D(filters = 32, kernel_size = (3,3), activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2)))

model.add(Conv2D(filters = 64, kernel_size = (3,3), activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2)))

model.add(Conv2D(filters = 128, kernel_size = (3,3), activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2)))

model.add(Dropout(rate = 0.25))

model.add(Flatten())
model.add(Dense(units = 64, activation = 'relu'))
model.add(Dropout(rate = 0.25))
model.add(Dense(units = 1, activation = 'sigmoid'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 238, 238, 16)      448

 conv2d_1 (Conv2D)           (None, 236, 236, 32)      4640

 max_pooling2d (MaxPooling2D  (None, 118, 118, 32)     0
 )

 conv2d_2 (Conv2D)           (None, 116, 116, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 58, 58, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 56, 56, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 28, 28, 128)      0
 2D)

 dropout (Dropout)           (None, 28, 28, 128)       0

 flatten (Flatten)           (None, 100352)            0

 dense (Dense)               (None, 64)                6422592

 dropout_1 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 6,520,097
Trainable params: 6,520,097
Non-trainable params: 0
_________________________________________________________________
```

In [34]:

```
#Compile our model
model.compile(optimizer = 'adam', loss = keras.losses.binary_crossentropy, metrics = ['a
```

In [35]:

```
#Early stopping and model checkpoint
from keras.callbacks import ModelCheckpoint, EarlyStopping

es = EarlyStopping(monitor = 'val_accuracy', min_delta = 0.01, patience = 5, verbose = 1
mc = ModelCheckpoint(monitor ='val_accuracy', filepath = './bestmodel.h5', verbose = 1,

cd = [es, mc]
```

In [36]:

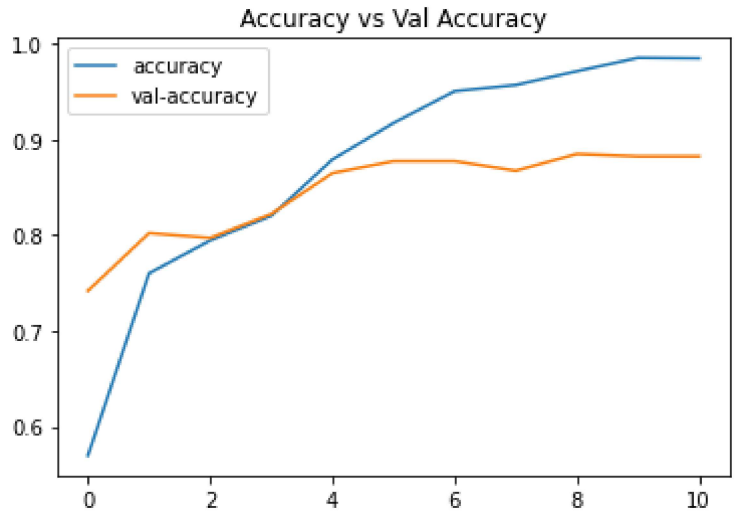
```
#Train our model
hist = model.fit(x = X_train, y = y_train, batch_size = 32, epochs = 30, validation_data
```

```
Epoch 1/30
59/59 [==============================] - ETA: 0s - loss: 0.7015 - accurac
y: 0.5699
Epoch 00001: val_accuracy improved from -inf to 0.74185, saving model to
./bestmodel.h5
59/59 [==============================] - 44s 209ms/step - loss: 0.7015 - a
ccuracy: 0.5699 - val_loss: 0.5538 - val_accuracy: 0.7419
Epoch 2/30
59/59 [==============================] - ETA: 0s - loss: 0.5135 - accurac
y: 0.7600
Epoch 00002: val_accuracy improved from 0.74185 to 0.80200, saving model t
o ./bestmodel.h5
59/59 [==============================] - 10s 177ms/step - loss: 0.5135 - a
ccuracy: 0.7600 - val_loss: 0.4544 - val_accuracy: 0.8020
Epoch 3/30
59/59 [==============================] - ETA: 0s - loss: 0.4548 - accurac
y: 0.7943
Epoch 00003: val_accuracy did not improve from 0.80200
59/59 [==============================] - 10s 174ms/step - loss: 0.4548 - a
ccuracy: 0.7943 - val_loss: 0.4386 - val_accuracy: 0.7970
Epoch 4/30
59/59 [==============================] - ETA: 0s - loss: 0.4072 - accurac
y: 0.8200
Epoch 00004: val_accuracy improved from 0.80200 to 0.82206, saving model t
o ./bestmodel.h5
59/59 [==============================] - 10s 177ms/step - loss: 0.4072 - a
ccuracy: 0.8200 - val_loss: 0.3844 - val_accuracy: 0.8221
Epoch 5/30
59/59 [==============================] - ETA: 0s - loss: 0.2959 - accurac
y: 0.8790
Epoch 00005: val_accuracy improved from 0.82206 to 0.86466, saving model t
o ./bestmodel.h5
59/59 [==============================] - 10s 177ms/step - loss: 0.2959 - a
ccuracy: 0.8790 - val_loss: 0.3483 - val_accuracy: 0.8647
Epoch 6/30
59/59 [==============================] - ETA: 0s - loss: 0.2041 - accurac
y: 0.9170
Epoch 00006: val_accuracy improved from 0.86466 to 0.87719, saving model t
o ./bestmodel.h5
59/59 [==============================] - 11s 178ms/step - loss: 0.2041 - a
ccuracy: 0.9170 - val_loss: 0.3660 - val_accuracy: 0.8772
Epoch 7/30
59/59 [==============================] - ETA: 0s - loss: 0.1320 - accurac
y: 0.9502
Epoch 00007: val_accuracy did not improve from 0.87719
59/59 [==============================] - 10s 176ms/step - loss: 0.1320 - a
ccuracy: 0.9502 - val_loss: 0.3585 - val_accuracy: 0.8772
Epoch 8/30
59/59 [==============================] - ETA: 0s - loss: 0.1202 - accurac
y: 0.9566
Epoch 00008: val_accuracy did not improve from 0.87719
59/59 [==============================] - 10s 176ms/step - loss: 0.1202 - a
ccuracy: 0.9566 - val_loss: 0.3720 - val_accuracy: 0.8672
Epoch 9/30
59/59 [==============================] - ETA: 0s - loss: 0.0852 - accurac
y: 0.9711
Epoch 00009: val_accuracy improved from 0.87719 to 0.88471, saving model t
o ./bestmodel.h5
59/59 [==============================] - 11s 179ms/step - loss: 0.0852 - a
ccuracy: 0.9711 - val_loss: 0.3249 - val_accuracy: 0.8847
Epoch 10/30
```

```
59/59 [==============================] - ETA: 0s - loss: 0.0457 - accurac
y: 0.9850
Epoch 00010: val_accuracy did not improve from 0.88471
59/59 [==============================] - 10s 176ms/step - loss: 0.0457 - a
ccuracy: 0.9850 - val_loss: 0.4838 - val_accuracy: 0.8822
Epoch 11/30
59/59 [==============================] - ETA: 0s - loss: 0.0456 - accurac
y: 0.9845
Epoch 00011: val_accuracy did not improve from 0.88471
59/59 [==============================] - 10s 176ms/step - loss: 0.0456 - a
ccuracy: 0.9845 - val_loss: 0.4322 - val_accuracy: 0.8822
Epoch 00011: early stopping
```

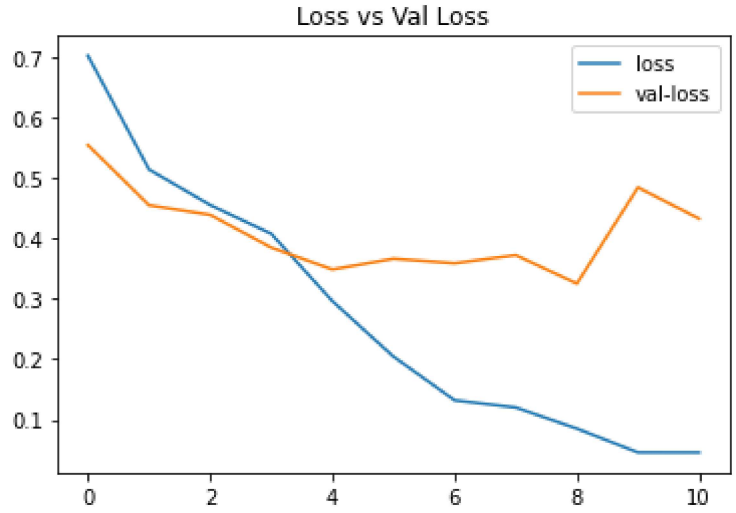
In [37]:

```
#Plot the graphical interpretation
h = hist.history
plt.plot(h['accuracy'], label = 'accuracy')
plt.plot(h['val_accuracy'], label = 'val-accuracy')
plt.title('Accuracy vs Val Accuracy')
plt.legend()
plt.show()
```



In [38]:

```
#Plot the graphical interpretation
h = hist.history
plt.plot(h['loss'], label = 'loss')
plt.plot(h['val_loss'], label = 'val-loss')
plt.title('Loss vs Val Loss')
plt.legend()
plt.show()
```



In [39]:

```
#Test our model on the test set
from keras.models import load_model
model = load_model('/content/bestmodel.h5')
acc = model.evaluate(X_test, y_test)[1]
print(f'The accuracy of our model is {acc}')
```

```
13/13 [==============================] - 1s 54ms/step - loss: 0.3359 - acc
uracy: 0.8947
The accuracy of our model is 0.8947368264198303
```

In [75]:

```
#Try our model on a random image that it has never seen before
from keras.preprocessing.image import load_img, img_to_array
#Choose a MRI image with tumor
path = "/content/aug_data/yes/aug_Y53_0_8799.jpg"
img_yes = load_img(path, target_size=(240,240))
img_array_yes = img_to_array(img_yes)/255
img_array_yes = np.expand_dims(img_array_yes, axis=0)
prediction1 = model.predict(img_array_yes)
class1 = np.round(prediction1).astype(int)
if class1 == 0:
  print("The MRI image doesn't have a Tumor")
else:
  print("The MRI image has a tumor")
```

```
The MRI image has a tumor
```

In [76]:

```
#Try our model on a random image that it has never seen before
from keras.preprocessing.image import load_img, img_to_array
#Choose a MRI image without tumor
path = "/content/aug_data/no/aug_11 no_0_8546.jpg"
img_no = load_img(path, target_size=(240,240))
img_array_no = img_to_array(img_no)/255
img_array_no = np.expand_dims(img_array_no, axis=0)
prediction2 = model.predict(img_array_no)[0][0]
class2 = np.round(prediction2).astype(int)
if class2 == 0:
  print("The MRI image doesn't have a Tumor")
else:
  print("The MRI image has a tumor")
```

```
The MRI image doesn't have a Tumor
```

Transfer Learning

In [85]:

```
#Import MobileNet Model
from keras.applications.mobilenet import MobileNet
base_model = MobileNet(input_shape=(240,240,3), include_top=False)
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is
not in [128, 160, 192, 224]. Weights for input shape (224, 224) will be lo
aded as the default.
```

In [86]:

```
#Make the layers of the model untrainable
for layer in base_model.layers:
  layer.trainable = False
```

In [87]:

```
#Add a flatten and dense layers to the base model
X = Flatten()(base_model.output)
X = Dense(units=1, activation='sigmoid')(X)
transfer_model = Model(base_model.input, X)
```

In [ ]:

```
transfer_model.summary()
```

In [89]:

```
#Compile the new model
transfer_model.compile(optimizer='rmsprop', loss=keras.losses.binary_crossentropy, metri
```

In [90]:

```
#Early stopping and model checkpoint
est = EarlyStopping(monitor = 'val_accuracy', min_delta = 0.01, patience = 5, verbose =
mct = ModelCheckpoint(monitor ='val_accuracy', filepath = './besttransfermodel.h5', verb

cdt = [est, mct]
```

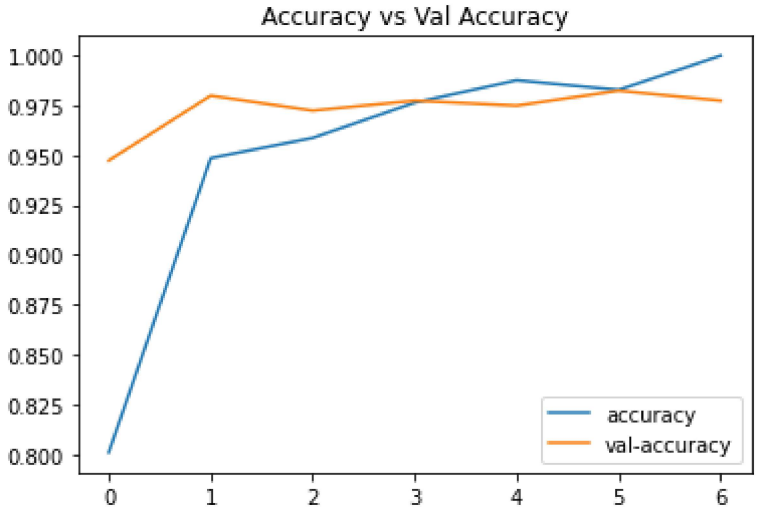
In [91]:

```
#Fit our new model
transfer_hist = transfer_model.fit(x = X_train, y = y_train, batch_size = 32, epochs = 3
```

```
Epoch 1/30
59/59 [==============================] - ETA: 0s - loss: 2.9944 - accurac
y: 0.8007
Epoch 00001: val_accuracy improved from -inf to 0.94737, saving model to
./besttransfermodel.h5
59/59 [==============================] - 12s 158ms/step - loss: 2.9944 - a
ccuracy: 0.8007 - val_loss: 0.4652 - val_accuracy: 0.9474
Epoch 2/30
58/59 [============================>.] - ETA: 0s - loss: 0.3483 - accurac
y: 0.9483
Epoch 00002: val_accuracy improved from 0.94737 to 0.97995, saving model t
o ./besttransfermodel.h5
59/59 [==============================] - 6s 107ms/step - loss: 0.3462 - ac
curacy: 0.9486 - val_loss: 0.0931 - val_accuracy: 0.9799
Epoch 3/30
58/59 [============================>.] - ETA: 0s - loss: 0.4720 - accurac
y: 0.9585
Epoch 00003: val_accuracy did not improve from 0.97995
59/59 [==============================] - 6s 103ms/step - loss: 0.4692 - ac
curacy: 0.9588 - val_loss: 0.1107 - val_accuracy: 0.9724
Epoch 4/30
58/59 [============================>.] - ETA: 0s - loss: 0.1917 - accurac
y: 0.9763
Epoch 00004: val_accuracy did not improve from 0.97995
59/59 [==============================] - 6s 103ms/step - loss: 0.1905 - ac
curacy: 0.9764 - val_loss: 0.1512 - val_accuracy: 0.9774
Epoch 5/30
58/59 [============================>.] - ETA: 0s - loss: 0.1570 - accurac
y: 0.9876
Epoch 00005: val_accuracy did not improve from 0.97995
59/59 [==============================] - 6s 103ms/step - loss: 0.1561 - ac
curacy: 0.9877 - val_loss: 0.2381 - val_accuracy: 0.9749
Epoch 6/30
58/59 [============================>.] - ETA: 0s - loss: 0.2401 - accurac
y: 0.9828
Epoch 00006: val_accuracy improved from 0.97995 to 0.98246, saving model t
o ./besttransfermodel.h5
59/59 [==============================] - 6s 107ms/step - loss: 0.2387 - ac
curacy: 0.9829 - val_loss: 0.1965 - val_accuracy: 0.9825
Epoch 7/30
58/59 [============================>.] - ETA: 0s - loss: 5.5239e-05 - accu
racy: 1.0000
Epoch 00007: val_accuracy did not improve from 0.98246
59/59 [==============================] - 6s 102ms/step - loss: 5.4914e-05
- accuracy: 1.0000 - val_loss: 0.1317 - val_accuracy: 0.9774
Epoch 00007: early stopping
```

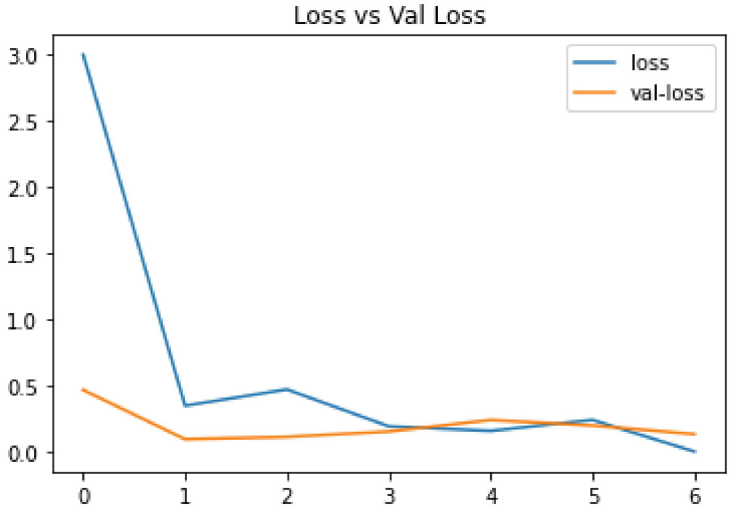
In [92]:

```
#Plot the graphical interpretation
h = transfer_hist.history
plt.plot(h['accuracy'], label = 'accuracy')
plt.plot(h['val_accuracy'], label = 'val-accuracy')
plt.title('Accuracy vs Val Accuracy')
plt.legend()
plt.show()
```



In [93]:

```
#Plot the graphical interpretation
h = transfer_hist.history
plt.plot(h['loss'], label = 'loss')
plt.plot(h['val_loss'], label = 'val-loss')
plt.title('Loss vs Val Loss')
plt.legend()
plt.show()
```



In [94]:

```
#Test our model on the test set
from keras.models import load_model
model = load_model('/content/besttransfermodel.h5')
acc = model.evaluate(X_test, y_test)[1]
print(f'The accuracy of our model is {acc}')
```

```
13/13 [==============================] - 2s 90ms/step - loss: 0.0711 - acc
uracy: 0.9875
The accuracy of our model is 0.9874686598777771
```

Import the VGG16 base model

In [ ]:

```
image_size = [224,224] # choose image size
# import the base model
vgg = VGG16(input_shape= image_size+[3],weights='imagenet',include_top=False)
```

Add layers on top of the base model

In [ ]:

```
x = vgg.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024,activation='relu')(x)
x = Dense(1024,activation='relu')(x)
x = Dense(512, activation='relu')(x)
preds = Dense(4,activation='softmax')(x)
model = Model(inputs = vgg.input,outputs=preds)
```

In [ ]:

```
# freeze base layers for training
for layer in vgg.layers:
    layer.trainable = False
```

Compile the model

In [ ]:

```
# initial optimizer
optimizer_initial = 'Adam'

# compile the model
model.compile(optimizer=optimizer_initial,
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Fit the model

In [ ]:

```
# initial parameters
epochs_initial = 5
step_size_train_initial=train_generator_initial.n//train_generator_initial.batch_size #

tic = time.time()
# fit the model
r = model.fit_generator(generator=train_generator_initial,
                        validation_data=val_generator_initial,
                        steps_per_epoch=step_size_train_initial,
                        epochs=epochs_initial)

toc = time.time()
print("Minutes taken = " + str((toc-tic)/60.0))
```

```
C:\Users\Student\anaconda_3\lib\site-packages\tensorflow\python\keras\engi
ne\training.py:1940: UserWarning: `Model.fit_generator` is deprecated and
will be removed in a future version. Please use `Model.fit`, which support
s generators.
  warnings.warn('`Model.fit_generator` is deprecated and '

Epoch 1/5
358/358 [==============================] - 932s 3s/step - loss: 0.7073 - a
ccuracy: 0.7491 - val_loss: 0.4140 - val_accuracy: 0.8334
Epoch 2/5
358/358 [==============================] - 941s 3s/step - loss: 0.3818 - a
ccuracy: 0.8557 - val_loss: 0.2514 - val_accuracy: 0.9056
Epoch 3/5
358/358 [==============================] - 949s 3s/step - loss: 0.2920 - a
ccuracy: 0.8945 - val_loss: 0.2006 - val_accuracy: 0.9362
Epoch 4/5
358/358 [==============================] - 957s 3s/step - loss: 0.2659 - a
ccuracy: 0.9001 - val_loss: 0.1971 - val_accuracy: 0.9373
Epoch 5/5
358/358 [==============================] - 968s 3s/step - loss: 0.2071 - a
ccuracy: 0.9186 - val_loss: 0.2009 - val_accuracy: 0.9289
```

Evaluate the model

In [ ]:

```
# evaluation on validation data
scores = model.evaluate(val_generator_initial)
print("%s%s: %.2f%%" % ("evaluate ",model.metrics_names[1], scores[1]*100))
```

```
359/359 [==============================] - 471s 1s/step - loss: 0.2009 - a
ccuracy: 0.9289
evaluate accuracy: 92.89%
```

In [ ]:

```
TheBaseModel = VGG16(input_shape= image_size+[3],weights='imagenet',include_top=False) #

param_label = 'optimizer'
param_list = ['Adam', 'SGD', 'RMSprop', 'Adagrad', 'Adadelta'] # ['Adam', 'SGD', 'RMSpro

accuracy_table = {param_label: [], 'accuracy': []}
tic = time.time()
for param in tqdm.tqdm_notebook(param_list):
    # Train, predict and evaluate model
    accuracy, _ = train_evaluate_the_model(train_generator_initial,val_generator_initial

    # Collect results
    accuracy_table[param_label].append(param)
    accuracy_table['accuracy'].append(accuracy)

accuracy_table = pd.DataFrame(accuracy_table) # convert the table to a dataframe
toc = time.time()
print("Minutes taken = " + str((toc-tic)/60.0))
accuracy_table # display the resullts
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=5.0), HTML(val
ue='')))
```

```
Epoch 1/5
358/358 [==============================] - 938s 3s/step - loss: 0.7257 - a
ccuracy: 0.7530 - val_loss: 0.3804 - val_accuracy: 0.8460
Epoch 2/5
358/358 [==============================] - 933s 3s/step - loss: 0.3693 - a
ccuracy: 0.8592 - val_loss: 0.3186 - val_accuracy: 0.8774
Epoch 3/5
358/358 [==============================] - 932s 3s/step - loss: 0.3001 - a
ccuracy: 0.8823 - val_loss: 0.1630 - val_accuracy: 0.9373
Epoch 4/5
358/358 [==============================] - 932s 3s/step - loss: 0.2670 - a
ccuracy: 0.9018 - val_loss: 0.1885 - val_accuracy: 0.9376
Epoch 5/5
358/358 [==============================] - 932s 3s/step - loss: 0.1999 - a
ccuracy: 0.9217 - val_loss: 0.1324 - val_accuracy: 0.9505
359/359 [==============================] - 467s 1s/step - loss: 0.1324 - a
ccuracy: 0.9505
Epoch 1/5
358/358 [==============================] - 938s 3s/step - loss: 0.6824 - a
ccuracy: 0.7568 - val_loss: 0.4280 - val_accuracy: 0.8073
Epoch 2/5
358/358 [==============================] - 937s 3s/step - loss: 0.3185 - a
ccuracy: 0.8756 - val_loss: 0.2820 - val_accuracy: 0.8815
Epoch 3/5
358/358 [==============================] - 936s 3s/step - loss: 0.2280 - a
ccuracy: 0.9175 - val_loss: 0.2479 - val_accuracy: 0.8958
Epoch 4/5
358/358 [==============================] - 935s 3s/step - loss: 0.1511 - a
ccuracy: 0.9441 - val_loss: 0.3753 - val_accuracy: 0.8557
Epoch 5/5
358/358 [==============================] - 937s 3s/step - loss: 0.1181 - a
ccuracy: 0.9532 - val_loss: 0.0713 - val_accuracy: 0.9812
359/359 [==============================] - 468s 1s/step - loss: 0.0713 - a
ccuracy: 0.9812
Epoch 1/5
358/358 [==============================] - 942s 3s/step - loss: 1.0042 - a
ccuracy: 0.7034 - val_loss: 0.3995 - val_accuracy: 0.8251
Epoch 2/5
358/358 [==============================] - 942s 3s/step - loss: 0.5534 - a
ccuracy: 0.8211 - val_loss: 0.4177 - val_accuracy: 0.8488
Epoch 3/5
358/358 [==============================] - 943s 3s/step - loss: 0.4907 - a
ccuracy: 0.8581 - val_loss: 0.2933 - val_accuracy: 0.8930
Epoch 4/5
358/358 [==============================] - 942s 3s/step - loss: 0.4174 - a
ccuracy: 0.8637 - val_loss: 0.3442 - val_accuracy: 0.8798
Epoch 5/5
358/358 [==============================] - 943s 3s/step - loss: 0.4177 - a
ccuracy: 0.8819 - val_loss: 0.1657 - val_accuracy: 0.9394
359/359 [==============================] - 470s 1s/step - loss: 0.1657 - a
ccuracy: 0.9394
Epoch 1/5
358/358 [==============================] - 940s 3s/step - loss: 0.6181 - a
ccuracy: 0.7655 - val_loss: 0.2996 - val_accuracy: 0.9063
Epoch 2/5
358/358 [==============================] - 939s 3s/step - loss: 0.3131 - a
ccuracy: 0.8864 - val_loss: 0.2158 - val_accuracy: 0.9282
Epoch 3/5
358/358 [==============================] - 942s 3s/step - loss: 0.2288 - a
ccuracy: 0.9224 - val_loss: 0.1662 - val_accuracy: 0.9551
Epoch 4/5
```

```
358/358 [==============================] - 944s 3s/step - loss: 0.1759 - a
ccuracy: 0.9462 - val_loss: 0.1386 - val_accuracy: 0.9617
Epoch 5/5
358/358 [==============================] - 937s 3s/step - loss: 0.1452 - a
ccuracy: 0.9546 - val_loss: 0.1093 - val_accuracy: 0.9774
```

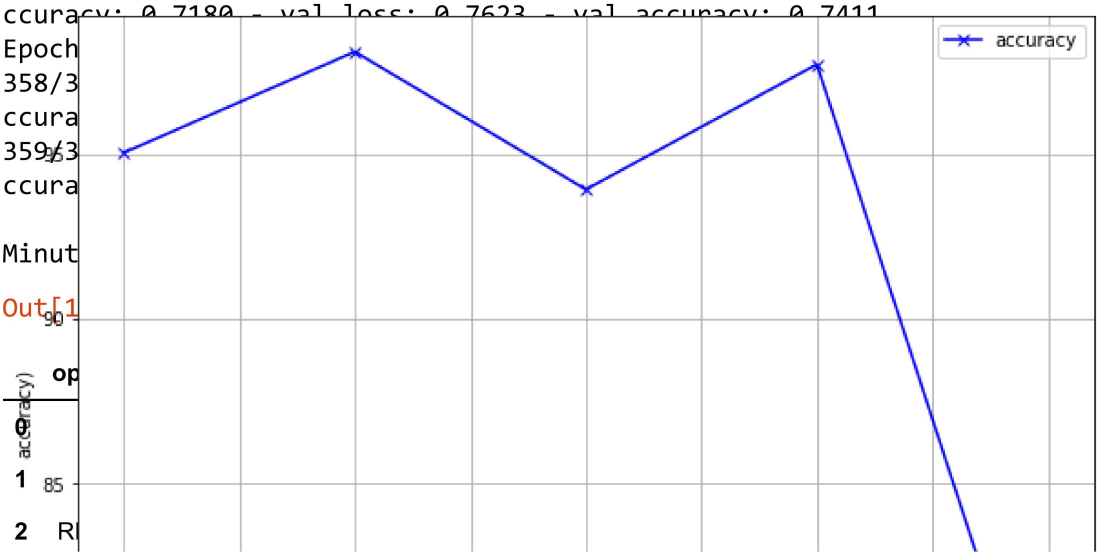
In [ ]:

```
from pylab import rcParams
#plot results
rcParams['figure.figsize'] = 10, 8 #width 10, height 8
accuracy_table.plot(x='optimizer', y='accuracy',style='bx-', grid=True)
#fix_xticklabels(param_list) #gave me errors when used
axes.set_xlabel("Optimizer")
axes.set_ylabel("accuracy)")
```

```
358/358 [==============================] - 470s 1s/step - loss: 0.1093 - a
ccuracy: 0.9774
Epoch 1/5
358/358 [==============================] - 941s 3s/step - loss: 1.3341 - a
ccuracy: 0.4050 - val_loss: 1.1604 - val_accuracy: 0.5073
Epoch 2/5
358/358 [==============================] - 939s 3s/step - loss: 1.0695 - a
ccuracy: 0.5709 - val_loss: 0.9767 - val_accuracy: 0.6268
Epoch 3/5
358/358 [==============================] - 941s 3s/step - loss: 0.9166 - a
ccuracy: 0.6565 - val_loss: 0.8528 - val_accuracy: 0.6990
Epoch 4/5
358/358 [==============================] - 941s 3s/step - loss: 0.8117 - a
ccuracy: 0.7180 - val_loss: 0.7623 - val_accuracy: 0.7411
Epoch
358/3                                                          - a
ccura
359/3                                                          - a
ccura

Minut

Out[1
```

Out[13]:

```
Text(0, 0.5, 'accuracy)')
```



3	Adagrad	97.735190
4	Adadelta	76.202089

In [ ]:

```
# Get optimum value for param
temp = accuracy_table[accuracy_table['accuracy'] == accuracy_table['accuracy'].max()]
optimizer_opt = temp[param_label].values[0]
print("max Accuracy = %0.3f" % accuracy_table['accuracy'].max())
print("optimum " + param_label + " = " + str(optimizer_opt))
```

```
max Accuracy = 98.118
optimum optimizer = SGD
```

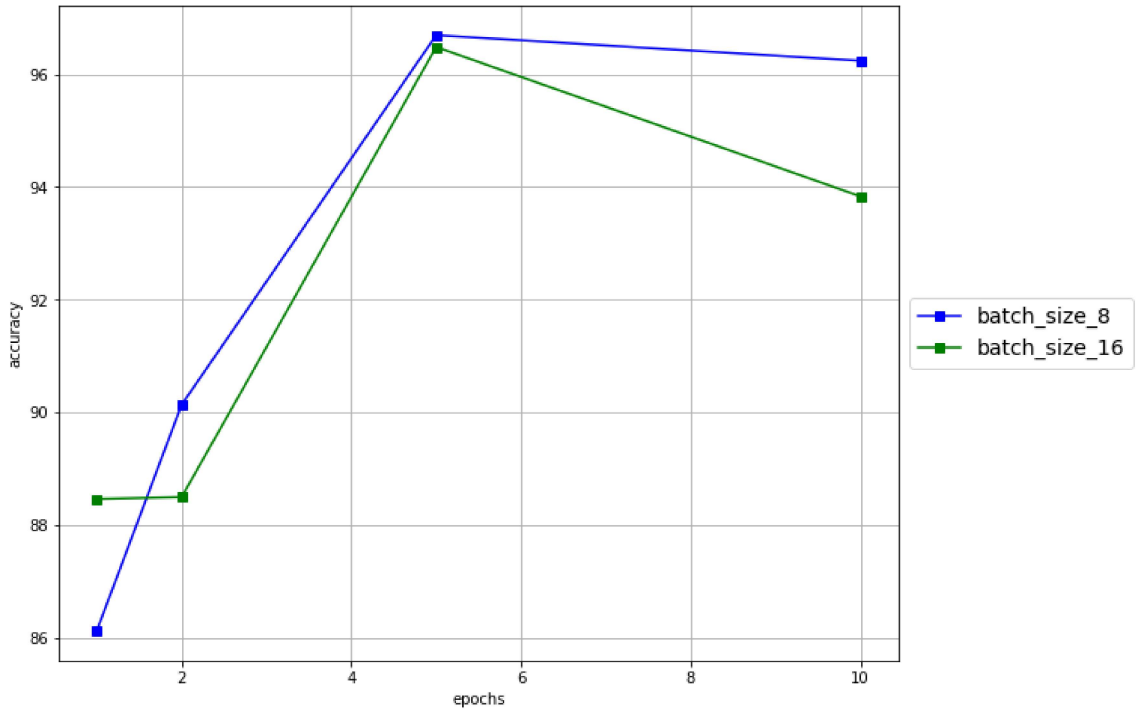
Tuning batch size and epochs of the model

In [ ]:

```
# Plot performance versus params
rcParams['figure.figsize'] = 10, 8 # width 10, height 8
temp = accuracy_table[accuracy_table[param2_label]==param2_list[0]]
ax = temp.plot(x=param_label, y='accuracy', style='bs-', grid=True)
legend_list = [param2_label + '_' + str(param2_list[0])]

color_list = ['r', 'g', 'k', '0.75']
for i in range(1,len(param2_list)):
    temp = accuracy_table[accuracy_table[param2_label]==param2_list[i]]
    ax = temp.plot(x=param_label, y='accuracy', color=color_list[i%len(color_list)], mar
    legend_list.append(param2_label + '_' + str(param2_list[i]))

ax.set_xlabel(param_label)
ax.set_ylabel("accuracy")
plt.rcParams.update({'font.size': 14})
plt.legend(legend_list, loc='center left', bbox_to_anchor=(1.0, 0.5)) # positions legend
```

Out[18]:

```
<matplotlib.legend.Legend at 0x213afd64af0>
```



In [ ]:

```
# Get optimum value for param and param2
temp = accuracy_table[accuracy_table['accuracy'] == accuracy_table['accuracy'].max()]
epochs_opt = temp[param_label].values[0]
batch_size_opt = temp[param2_label].values[0]
print("max Accuracy = %0.3f" % accuracy_table['accuracy'].max())
print("optimum " + param_label + " = " + str(epochs_opt))
print("optimum " + param2_label + " = " + str(batch_size_opt))
```

```
max Accuracy = 96.690
optimum epochs = 5
optimum batch_size = 8
```

Tunning dropout of the model

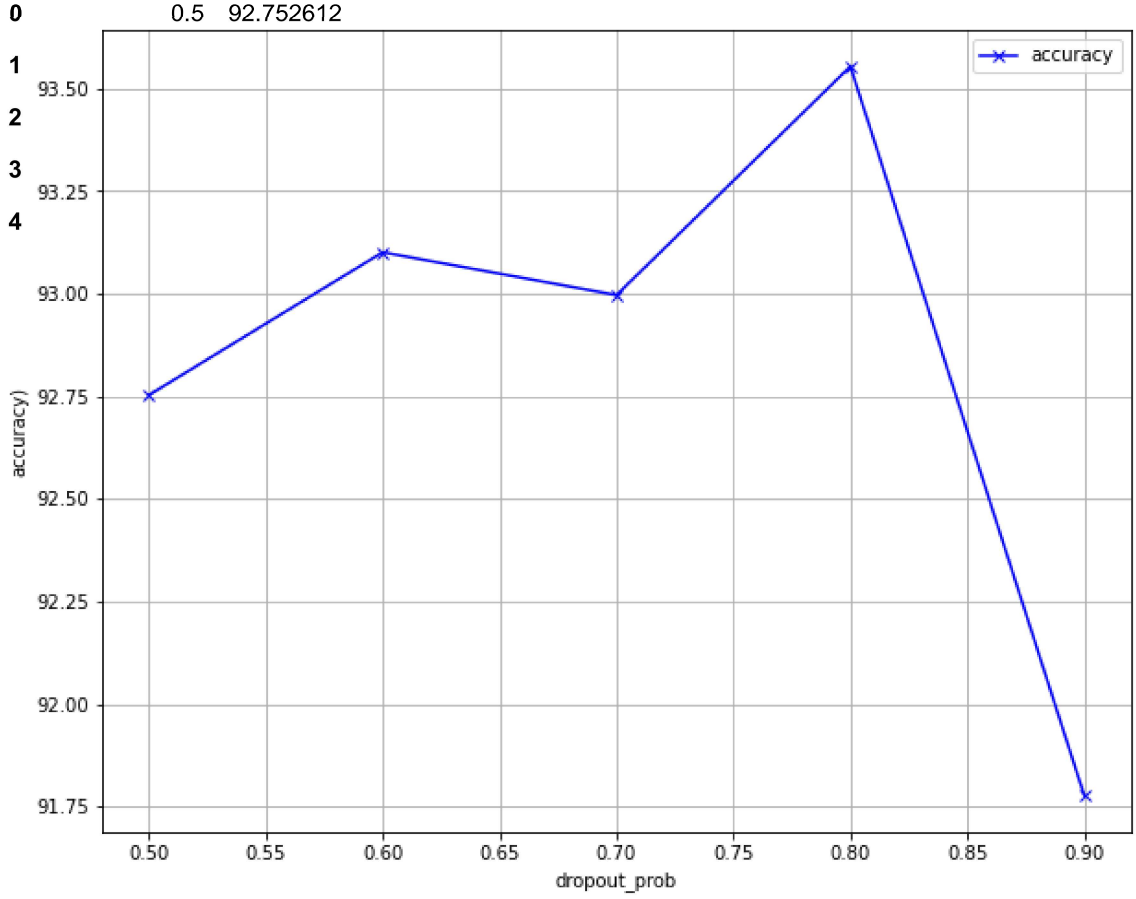
In [ ]:

```
param_label = 'dropout_prob'
param_list = [0.5, 0.6, 0.7, 0.8, 0.9]

TheBaseModel = VGG16(input_shape= image_size+[3],weights='imagenet',include_top=False) #

accuracy_table = {param_label: [], 'accuracy': []}
tic = time.time()
for param in tqdm.tqdm_notebook(param_list):
    # Train, predict and evaluate model
    accuracy, _ = train_evaluate_the_model(train_generator_initial,val_generator_initial

    # Collect results
    accuracy_table[param_label].append(param)
    accuracy_table['accuracy'].append(accuracy)

accuracy_table = pd.DataFrame(accuracy_table) # convert the table to a dataframe
toc = time.time()
print("Minutes taken = " + str((toc-tic)/60.0))
accuracy_table # display the resullts
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=5.0), HTML(val
ue='')))

C:\Users\Student\anaconda_3\lib\site-packages\tensorflow\python\keras\engi
ne\training.py:1940: UserWarning: `Model.fit_generator` is deprecated and
will be removed in a future version. Please use `Model.fit`, which support
s generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
```

```
Epoch 1/4
358/358 [==============================] - 1136s 3s/step - loss: 0.8254 -
accuracy: 0.7324 - val_loss: 0.4870 - val_accuracy: 0.8007
Epoch 2/4
358/358 [==============================] - 1132s 3s/step - loss: 0.4211 -
accuracy: 0.8442 - val_loss: 0.3302 - val_accuracy: 0.8801
Epoch 3/4
358/358 [==============================] - 1076s 3s/step - loss: 0.3479 -
accuracy: 0.8714 - val_loss: 0.2235 - val_accuracy: 0.9202
Epoch 4/4
358/358 [==============================] - 1080s 3s/step - loss: 0.2609 -
accuracy: 0.9060 - val_loss: 0.1825 - val_accuracy: 0.9275
359/359 [==============================] - 538s 1s/step - loss: 0.1825 - a
ccuracy: 0.9275
Epoch 1/4
358/358 [==============================] - 1084s 3s/step - loss: 0.9356 -
accuracy: 0.7261 - val_loss: 0.3973 - val_accuracy: 0.8526
Epoch 2/4
358/358 [==============================] - 1132s 3s/step - loss: 0.4669 -
accuracy: 0.8274 - val_loss: 0.3752 - val_accuracy: 0.8666
Epoch 3/4
358/358 [==============================] - 1086s 3s/step - loss: 0.3197 -
accuracy: 0.8829 - val_loss: 0.2591 - val_accuracy: 0.9045
Epoch 4/4
358/358 [==============================] - 1030s 3s/step - loss: 0.2868 -
accuracy: 0.8945 - val_loss: 0.1707 - val_accuracy: 0.9310
359/359 [==============================] - 515s 1s/step - loss: 0.1707 - a
ccuracy: 0.9310
Epoch 1/4
358/358 [==============================] - 1052s 3s/step - loss: 1.0288 -
accuracy: 0.6855 - val_loss: 0.3851 - val_accuracy: 0.8707
Epoch 2/4
358/358 [==============================] - 1030s 3s/step - loss: 0.4366 -
accuracy: 0.8435 - val_loss: 0.4316 - val_accuracy: 0.8460
Epoch 3/4
358/358 [==============================] - 1029s 3s/step - loss: 0.3176 -
accuracy: 0.8774 - val_loss: 0.2407 - val_accuracy: 0.9129
Epoch 4/4
358/358 [==============================] - 1031s 3s/step - loss: 0.3216 -
accuracy: 0.8924 - val_loss: 0.2067 - val_accuracy: 0.9300
359/359 [==============================] - 513s 1s/step - loss: 0.2067 - a
ccuracy: 0.9300
Epoch 1/4
358/358 [==============================] - 1027s 3s/step - loss: 1.2053 -
accuracy: 0.6548 - val_loss: 0.4078 - val_accuracy: 0.8341
Epoch 2/4
358/358 [==============================] - 1027s 3s/step - loss: 0.4809 -
accuracy: 0.8323 - val_loss: 0.2800 - val_accuracy: 0.8990
Epoch 3/4
358/358 [==============================] - 1024s 3s/step - loss: 0.3574 -
accuracy: 0.8739 - val_loss: 0.2445 - val_accuracy: 0.9045
Epoch 4/4
358/358 [==============================] - 1025s 3s/step - loss: 0.2982 -
accuracy: 0.8973 - val_loss: 0.1775 - val_accuracy: 0.9355
359/359 [==============================] - 512s 1s/step - loss: 0.1775 - a
ccuracy: 0.9355
Epoch 1/4
358/358 [==============================] - 1044s 3s/step - loss: 1.5150 -
accuracy: 0.5975 - val_loss: 0.5066 - val_accuracy: 0.8115
Epoch 2/4
358/358 [==============================] - 1050s 3s/step - loss: 0.6077 -
```

```
accuracy: 0.7890 - val_loss: 0.4126 - val_accuracy: 0.8495
In [ ]:
Epoch 3/4
358/358 [==============================] - 1048s 3s/step - loss: 0.4840 -
from pylab import rcParams
accuracy: 0.8410 - val_loss: 0.2681 - val_accuracy: 0.9063
Epoch 4/4
# plot results
358/358 [==============================] - 1041s 3s/step - loss: 0.4105 -
rcParams['figure.figsize'] = 10,8 # width 10, height 8
accuracy: 0.8651 - val_loss: 0.2440 - val_accuracy: 0.9178
359/359 [==============================] - 521s 1s/step - loss: 0.2440 - a
ax = accuracy_table.plot(x='dropout_prob', y='accuracy',style='bx-', grid=True)
ccuracy: 0.9178
#ax.set_xticklabels(param_list) # gave me an error when used
ax.set_xlabel("dropout_prob")
Minutes taken = 396.4226357460022
ax.set_ylabel("accuracy)")
Out[9]:
Out[10]:
    dropout_prob   accuracy
0            0.5  92.752612
Text(0, 0.5, 'accuracy)')
```




```
In [ ]:
```

```
# Get optimum value for param
temp = accuracy_table[accuracy_table['accuracy'] == accuracy_table['accuracy'].max()]
dropout_opt = temp[param_label].values[0]
print("max Accuracy = %0.3f" % accuracy_table['accuracy'].max())
print("optimum " + param_label + " = " + str(dropout_opt))
```

```
max Accuracy = 93.554
optimum dropout_prob = 0.8
```

Collecting all optimum tuned parameters

In [ ]:

```
d = {'param': ['optimizer', 'epochs', 'batch_size','dropout_prob'],
     'original': [optimizer_initial, epochs_initial, batchsize_initial,dropout_value_ini
     'after_tuning': [optimizer_opt, epochs_opt, batch_size_opt,dropout_opt]}
tuned_params = pd.DataFrame(d)
tuned_params
```

Out[13]:

	param	original	after_tuning
0	optimizer	Adam	SGD
1	epochs	4	5
2	batch_size	8	8
3	dropout_prob	0.5	0.8

Creating the final model with optimum parameters

In [ ]:

```
# get train and validation data generator
train_generator_opt,val_generator_opt,test_generator_opt = get_data_generator(train_data

# load the base model
The_base_Model = VGG16(input_shape= image_size+[3],weights='imagenet',include_top=False)

# train and evaluate the model
accuracy, New_model = train_evaluate_the_model(train_generator_opt, val_generator_opt, o
```

```
Found 2870 images belonging to 4 classes.
Found 2870 images belonging to 4 classes.
Found 394 images belonging to 4 classes.

C:\Users\Student\anaconda_3\lib\site-packages\tensorflow\python\keras\engi
ne\training.py:1940: UserWarning: `Model.fit_generator` is deprecated and
will be removed in a future version. Please use `Model.fit`, which support
s generators.
  warnings.warn('`Model.fit_generator` is deprecated and '

Epoch 1/5
358/358 [==============================] - 1023s 3s/step - loss: 1.1318 -
accuracy: 0.6338 - val_loss: 0.5083 - val_accuracy: 0.7958
Epoch 2/5
358/358 [==============================] - 1050s 3s/step - loss: 0.5398 -
accuracy: 0.7883 - val_loss: 0.3276 - val_accuracy: 0.8808
Epoch 3/5
358/358 [==============================] - 1040s 3s/step - loss: 0.4479 -
accuracy: 0.8239 - val_loss: 0.3719 - val_accuracy: 0.8505
Epoch 4/5
358/358 [==============================] - 1040s 3s/step - loss: 0.3779 -
accuracy: 0.8641 - val_loss: 0.2510 - val_accuracy: 0.9122
Epoch 5/5
358/358 [==============================] - 1035s 3s/step - loss: 0.3171 -
accuracy: 0.8788 - val_loss: 0.2036 - val_accuracy: 0.9230
359/359 [==============================] - 520s 1s/step - loss: 0.2036 - a
ccuracy: 0.9230
```

In [ ]:

```
accuracy
```

Out[15]:

```
92.29965209960938
```

Import the EfficientnetB7 base model

In [ ]:

```
from tensorflow.keras.applications import EfficientNetB7
```

In [ ]:

```
image_size = [224,224] # choose image size
# import the base model
efnB7 = tf.keras.applications.efficientnet.EfficientNetB7(input_shape= image_size+[3],we

# storing the base model in the kernel for later use to avoid loading many times
efnB7_basemodel = efnB7
```

Add layers on top of the base model

In [ ]:

```
# initial parameter value
dropout_value_initial = 0.1

# add layers
x = efnB7.output
x = tf.keras.layers.Flatten()(x)
x = Dense(1024, activation="relu")(x)
x = Dense(1024, activation="relu")(x)
x = Dense(512, activation="relu")(x)
x = tf.keras.layers.Dropout(dropout_value_initial)(x)
predictions = Dense(4, activation="sigmoid")(x)
model = Model(inputs = efnB7.input, outputs = predictions)
```

Out[12]:

```
'\n# add layers\nx = efnB7.output\nx = tf.keras.layers.Flatten()(x)\nx = D
ense(1024, activation="relu")(x)\nx = Dense(1024, activation="relu")(x)\nx
= Dense(512, activation="relu")(x)\nx = tf.keras.layers.Dropout(dropout_va
lue_initial)(x)\npredictions = Dense(4, activation="sigmoid")(x)\nmodel =
Model(inputs = efnB7.input, outputs = predictions)'
```

In [ ]:

```
# freeze base layers for training
for layer in efnB7.layers:
    layer.trainable = False
```

Compile the model

In [ ]:

```
# initial optimizer
optimizer_initial = 'Adam'

# compile the model
model.compile(optimizer=optimizer_initial,
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Out[14]:

```
"\n# compile the model\nmodel.compile(optimizer=optimizer_initial,\n
loss='categorical_crossentropy',\n              metrics=['accuracy'])"
```

Fit the model

In [ ]:

```
# initial parameters
epochs_initial = 4
step_size_train_initial=train_generator_initial.n//train_generator_initial.batch_size #

tic = time.time()
# fit the model
r = model.fit_generator(generator=train_generator_initial,
                        validation_data=val_generator_initial,
                        steps_per_epoch=step_size_train_initial,
                        epochs=epochs_initial)

toc = time.time()
print("Minutes taken = " + str((toc-tic)/60.0))
```

```
C:\Users\Student\anaconda_3\lib\site-packages\tensorflow\python\keras\engi
ne\training.py:1940: UserWarning: `Model.fit_generator` is deprecated and
will be removed in a future version. Please use `Model.fit`, which support
s generators.
  warnings.warn('`Model.fit_generator` is deprecated and '

Epoch 1/4
358/358 [==============================] - 1367s 4s/step - loss: 2.3466 -
accuracy: 0.6600 - val_loss: 0.4941 - val_accuracy: 0.8362
Epoch 2/4
358/358 [==============================] - 1448s 4s/step - loss: 0.4798 -
accuracy: 0.8298 - val_loss: 0.2534 - val_accuracy: 0.9045
Epoch 3/4
358/358 [==============================] - 1494s 4s/step - loss: 0.3648 -
accuracy: 0.8676 - val_loss: 0.3082 - val_accuracy: 0.8895
Epoch 4/4
358/358 [==============================] - 1431s 4s/step - loss: 0.2838 -
accuracy: 0.8941 - val_loss: 0.1202 - val_accuracy: 0.9544
Minutes taken = 95.67590473890304
```

Evaluate the model

In [ ]:

```
# evaluation on validation data
scores = model.evaluate(val_generator_initial)
print("%s%s: %.2f%%" % ("evaluate ",model.metrics_names[1], scores[1]*100))
```

```
359/359 [==============================] - 621s 2s/step - loss: 0.1202 - a
ccuracy: 0.9544
evaluate accuracy: 95.44%
```

start fine tunning

Tunning optimizers of the model

In [ ]:

```
param_label = 'optimizer'
param_list = ['Adam', 'SGD', 'RMSprop', 'Adagrad', 'Adadelta'] # ['Adam', 'SGD', 'RMSpro

accuracy_table = {param_label: [], 'accuracy': []}
tic = time.time()
for param in tqdm.tqdm_notebook(param_list):
    # Train, and evaluate model
    accuracy, _ = train_evaluate_the_model(train_generator_initial,val_generator_initial

    # Collect results
    accuracy_table[param_label].append(param)
    accuracy_table['accuracy'].append(accuracy)

accuracy_table = pd.DataFrame(accuracy_table) # convert the table to a dataframe
toc = time.time()
print("Minutes taken = " + str((toc-tic)/60.0))
accuracy_table # display the resullts
```

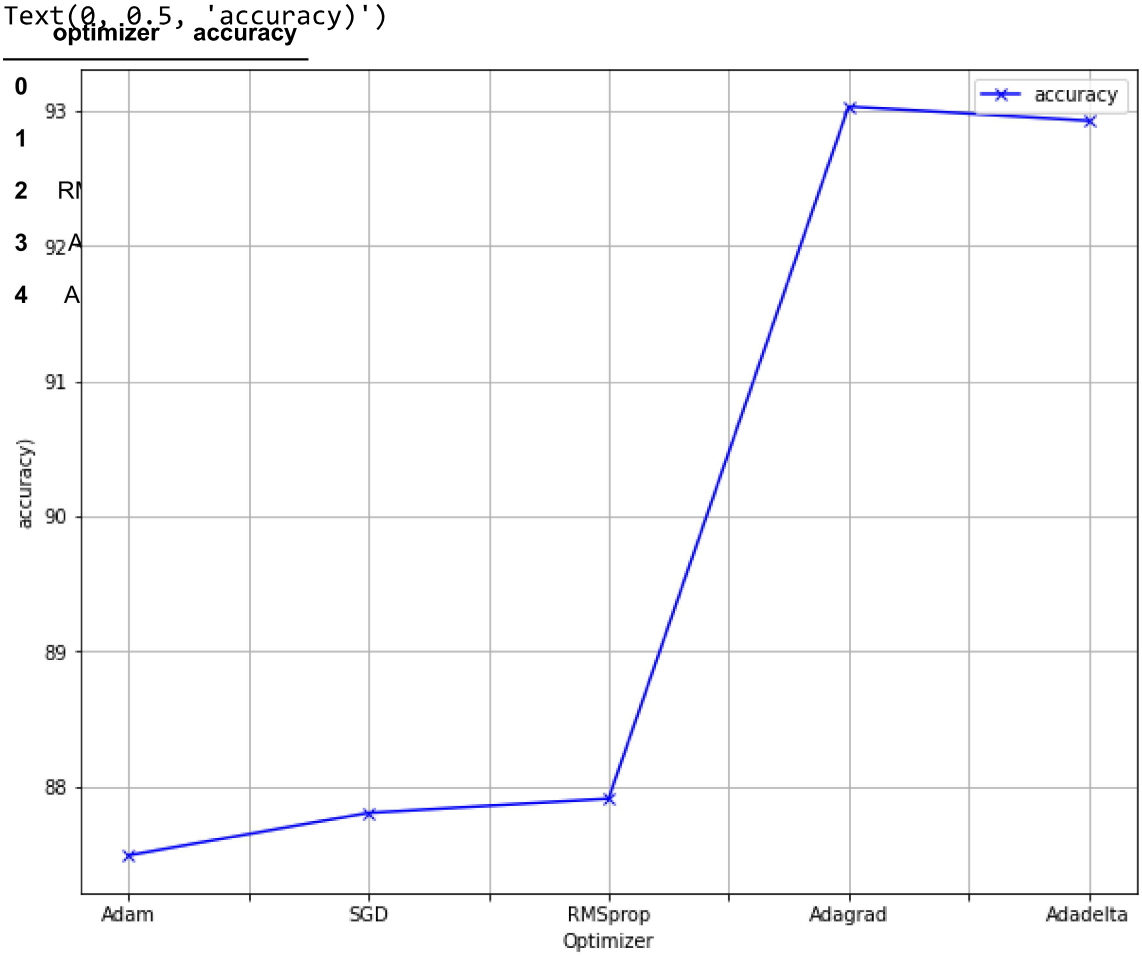
```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=5.0), HTML(val
ue='')))
```

```
Epoch 1/4
358/358 [==============================] - 1339s 4s/step - loss: 2.5084 -
accuracy: 0.6908 - val_loss: 0.4476 - val_accuracy: 0.8220
Epoch 2/4
358/358 [==============================] - 1317s 4s/step - loss: 0.4882 -
accuracy: 0.8281 - val_loss: 0.2847 - val_accuracy: 0.8815
Epoch 3/4
358/358 [==============================] - 1329s 4s/step - loss: 0.3549 -
accuracy: 0.8742 - val_loss: 0.2113 - val_accuracy: 0.9174
Epoch 4/4
358/358 [==============================] - 1334s 4s/step - loss: 0.2783 -
accuracy: 0.8973 - val_loss: 0.5367 - val_accuracy: 0.8749
359/359 [==============================] - 613s 2s/step - loss: 0.5367 - a
ccuracy: 0.8749
Epoch 1/4
358/358 [==============================] - 1294s 4s/step - loss: 0.8377 -
accuracy: 0.6663 - val_loss: 0.5529 - val_accuracy: 0.7780
Epoch 2/4
358/358 [==============================] - 1304s 4s/step - loss: 0.4519 -
accuracy: 0.8180 - val_loss: 0.3014 - val_accuracy: 0.8812
Epoch 3/4
358/358 [==============================] - 1298s 4s/step - loss: 0.3445 -
accuracy: 0.8648 - val_loss: 0.2320 - val_accuracy: 0.9080
Epoch 4/4
358/358 [==============================] - 1324s 4s/step - loss: 0.2578 -
accuracy: 0.9036 - val_loss: 0.2949 - val_accuracy: 0.8780
359/359 [==============================] - 639s 2s/step - loss: 0.2949 - a
ccuracy: 0.8780
Epoch 1/4
358/358 [==============================] - 1636s 5s/step - loss: 5.7865 -
accuracy: 0.5975 - val_loss: 1.1571 - val_accuracy: 0.7087
Epoch 2/4
358/358 [==============================] - 1645s 5s/step - loss: 0.7259 -
accuracy: 0.7813 - val_loss: 0.4371 - val_accuracy: 0.8561
Epoch 3/4
358/358 [==============================] - 1619s 5s/step - loss: 0.6020 -
accuracy: 0.8340 - val_loss: 0.6526 - val_accuracy: 0.7819
Epoch 4/4
358/358 [==============================] - 1604s 4s/step - loss: 0.5373 -
accuracy: 0.8414 - val_loss: 0.2929 - val_accuracy: 0.8791
359/359 [==============================] - 646s 2s/step - loss: 0.2929 - a
ccuracy: 0.8791
Epoch 1/4
358/358 [==============================] - 1401s 4s/step - loss: 0.6773 -
accuracy: 0.7233 - val_loss: 0.3400 - val_accuracy: 0.8746
Epoch 2/4
358/358 [==============================] - 1375s 4s/step - loss: 0.3716 -
accuracy: 0.8595 - val_loss: 0.2172 - val_accuracy: 0.9216
Epoch 3/4
358/358 [==============================] - 1354s 4s/step - loss: 0.2576 -
accuracy: 0.8959 - val_loss: 0.1697 - val_accuracy: 0.9324
Epoch 4/4
358/358 [==============================] - 1355s 4s/step - loss: 0.1901 -
accuracy: 0.9315 - val_loss: 0.1756 - val_accuracy: 0.9303
359/359 [==============================] - 630s 2s/step - loss: 0.1756 - a
ccuracy: 0.9303
Epoch 1/4
358/358 [==============================] - 1420s 4s/step - loss: 0.8166 -
accuracy: 0.6869 - val_loss: 0.5055 - val_accuracy: 0.8237
Epoch 2/4
358/358 [==============================] - 1369s 4s/step - loss: 0.4994 -
```

```
accuracy: 0.8124 - val_loss: 0.3684 - val_accuracy: 0.8739
Epoch 3/4
358/358 [==============================] - 1363s 4s/step - loss: 0.3879 -
accuracy: 0.8732 - val_loss: 0.2866 - val_accuracy: 0.9084
Epoch 4/4
358/358 [==============================] - 1392s 4s/step - loss: 0.3271 -
accuracy: 0.8885 - val_loss: 0.2368 - val_accuracy: 0.9293
359/359 [==============================] - 630s 2s/step - loss: 0.2368 - a
ccuracy: 0.9293
Minutes taken = 520.6418586293856
```

Out[22]:

```
Text(0, 0.5, 'accuracy)')
```



In [ ]:

```
# Get optimum value for param
temp = accuracy_table[accuracy_table['accuracy'] == accuracy_table['accuracy'].max()]
optimizer_opt = temp[param_label].values[0]
print("max Accuracy = %0.3f" % accuracy_table['accuracy'].max())
print("optimum " + param_label + " = " + str(optimizer_opt))
```

```
max Accuracy = 93.031
optimum optimizer = Adagrad
```

Tunning dropout of the model

In [ ]:

```
param_label = 'dropout_prob'
param_list = [0.5, 0.6, 0.7, 0.8, 0.9]


accuracy_table = {param_label: [], 'accuracy': []}
tic = time.time()
for param in tqdm.tqdm_notebook(param_list):
    # Train, predict and evaluate model
    accuracy, _ = train_evaluate_the_model(train_generator_initial,val_generator_initial

    # Collect results
    accuracy_table[param_label].append(param)
    accuracy_table['accuracy'].append(accuracy)

accuracy_table = pd.DataFrame(accuracy_table) # convert the table to a dataframe
toc = time.time()
print("Minutes taken = " + str((toc-tic)/60.0))
accuracy_table # display the resullts
```

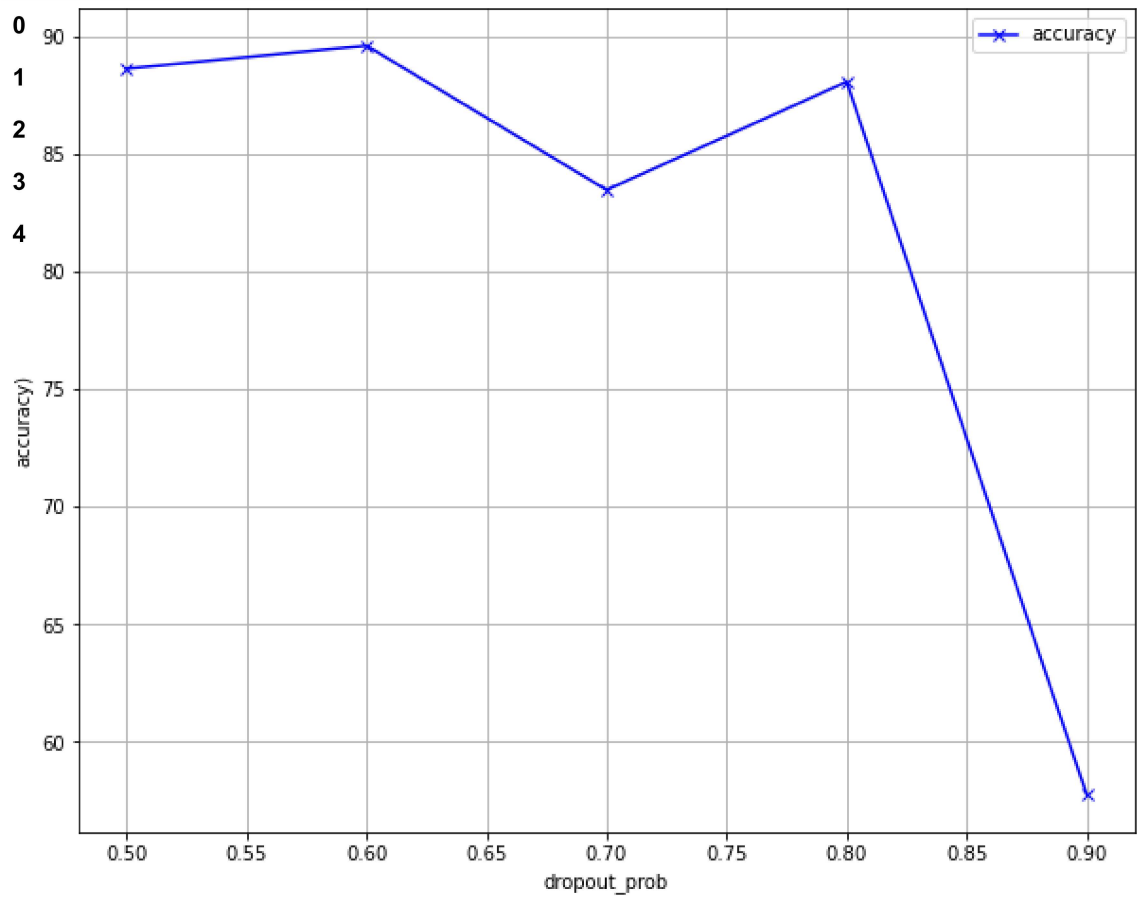
```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=5.0), HTML(val
ue='')))

C:\Users\Student\anaconda_3\lib\site-packages\tensorflow\python\keras\engi
ne\training.py:1940: UserWarning: `Model.fit_generator` is deprecated and
will be removed in a future version. Please use `Model.fit`, which support
s generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
```

```
Epoch 1/4
358/358 [==============================] - 1345s 4s/step - loss: 3.2620 -
accuracy: 0.6565 - val_loss: 0.5753 - val_accuracy: 0.8024
Epoch 2/4
358/358 [==============================] - 1303s 4s/step - loss: 0.5616 -
accuracy: 0.7949 - val_loss: 0.3743 - val_accuracy: 0.8585
Epoch 3/4
358/358 [==============================] - 1302s 4s/step - loss: 0.3792 -
accuracy: 0.8627 - val_loss: 0.2664 - val_accuracy: 0.8962
Epoch 4/4
358/358 [==============================] - 1307s 4s/step - loss: 0.3954 -
accuracy: 0.8620 - val_loss: 0.3597 - val_accuracy: 0.8864
359/359 [==============================] - 600s 2s/step - loss: 0.3597 - a
ccuracy: 0.8864
Epoch 1/4
358/358 [==============================] - 1240s 3s/step - loss: 3.4531 -
accuracy: 0.6359 - val_loss: 0.5186 - val_accuracy: 0.8230
Epoch 2/4
358/358 [==============================] - 1216s 3s/step - loss: 0.5781 -
accuracy: 0.7928 - val_loss: 0.3659 - val_accuracy: 0.8617
Epoch 3/4
358/358 [==============================] - 1217s 3s/step - loss: 0.4623 -
accuracy: 0.8417 - val_loss: 0.3268 - val_accuracy: 0.8972
Epoch 4/4
358/358 [==============================] - 1220s 3s/step - loss: 0.3925 -
accuracy: 0.8655 - val_loss: 0.2433 - val_accuracy: 0.8962
359/359 [==============================] - 557s 2s/step - loss: 0.2433 - a
ccuracy: 0.8962
Epoch 1/4
358/358 [==============================] - 1353s 4s/step - loss: 4.4965 -
accuracy: 0.5699 - val_loss: 0.7670 - val_accuracy: 0.6746
Epoch 2/4
358/358 [==============================] - 1332s 4s/step - loss: 0.7271 -
accuracy: 0.7320 - val_loss: 0.4393 - val_accuracy: 0.8268
Epoch 3/4
358/358 [==============================] - 1340s 4s/step - loss: 0.6005 -
accuracy: 0.7837 - val_loss: 0.4427 - val_accuracy: 0.8286
Epoch 4/4
358/358 [==============================] - 1350s 4s/step - loss: 0.4855 -
accuracy: 0.8319 - val_loss: 0.3842 - val_accuracy: 0.8348
359/359 [==============================] - 596s 2s/step - loss: 0.3842 - a
ccuracy: 0.8348
Epoch 1/4
358/358 [==============================] - 1430s 4s/step - loss: 5.7433 -
accuracy: 0.5611 - val_loss: 0.5523 - val_accuracy: 0.8094
Epoch 2/4
358/358 [==============================] - 1376s 4s/step - loss: 0.7385 -
accuracy: 0.7638 - val_loss: 0.3943 - val_accuracy: 0.8516
Epoch 3/4
358/358 [==============================] - 1483s 4s/step - loss: 0.5969 -
accuracy: 0.8064 - val_loss: 0.5374 - val_accuracy: 0.7969
Epoch 4/4
358/358 [==============================] - 1553s 4s/step - loss: 0.4714 -
accuracy: 0.8449 - val_loss: 0.3470 - val_accuracy: 0.8808
359/359 [==============================] - 721s 2s/step - loss: 0.3470 - a
ccuracy: 0.8808
Epoch 1/4
358/358 [==============================] - 1590s 4s/step - loss: 9.7914 -
accuracy: 0.4071 - val_loss: 1.2320 - val_accuracy: 0.5003
Epoch 2/4
358/358 [==============================] - 1509s 4s/step - loss: 1.2089 -
```

```
accuracy: 0.4675 - val_loss: 0.9424 - val_accuracy: 0.5557
In [ ]:
Epoch 3/4
358/358 [==============================] - 1545s 4s/step - loss: 1.1041 -
from pylab import rcParams
accuracy: 0.4836 - val_loss: 0.9350 - val_accuracy: 0.5554
Epoch 4/4
# plot results
358/358 [==============================] - 1487s 4s/step - loss: 1.0982 -
rcParams['figure.figsize'] = 10,8 # width 10, height 8
accuracy: 0.5049 - val_loss: 0.9019 - val_accuracy: 0.5777
359/359 [==============================] - 672s 2s/step - loss: 0.9019 - a
ax = accuracy_table.plot(x='dropout_prob', y='accuracy',style='bx-', grid=True)
ccuracy: 0.5777
ax.set_xlabel("dropout_prob")
ax.set_ylabel("accuracy)")
Minutes taken = 510.8875221014023
```

Out[8]:

```
Text(0, 0.5, 'accuracy)')
```



In [ ]:

```
# Get optimum value for param
temp = accuracy_table[accuracy_table['accuracy'] == accuracy_table['accuracy'].max()]
dropout_opt = temp[param_label].values[0]
print("max Accuracy = %0.3f" % accuracy_table['accuracy'].max())
print("optimum " + param_label + " = " + str(dropout_opt))
```

```
max Accuracy = 89.617
optimum dropout_prob = 0.6
```

Tuning batch size and epochs of the model

In [ ]:

```
param_label = 'epochs'
param_list = [1,2,5,10] # [5, 10, 15, 20]

param2_label = 'batch_size'
param2_list = [8, 16] # [8, 16, 32] am using this batch sizes for now until we fix the a

accuracy_table = {param_label: [], param2_label: [], 'accuracy': []}
tic = time.time()
for param in tqdm.tqdm_notebook(param_list):
    for param2 in tqdm_notebook(param2_list):

        # generate train and validation data
        train_generator,val_generator, _ = get_data_generator(train_dataset_path,validat
        # train, predict and evaluate model
        accuracy, _ = train_evaluate_the_model(train_generator, val_generator, optimizer

        # collect results
        accuracy_table[param_label].append(param)
        accuracy_table[param2_label].append(param2)
        accuracy_table['accuracy'].append(accuracy)

accuracy_table = pd.DataFrame(accuracy_table) # convert the table to a dataframe
toc = time.time()
print("Minutes taken = " + str((toc-tic)/60.0))
accuracy_table # display the resullts
```

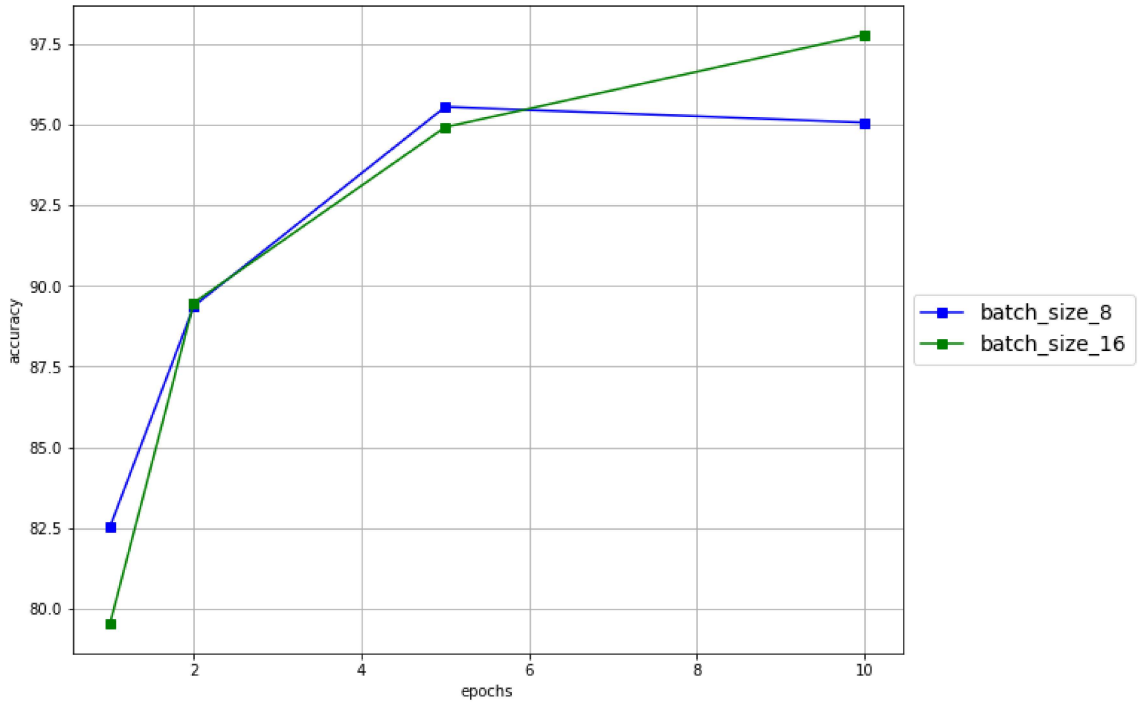
```
- accuracy: 0.8966 - val_loss: 0.1549 - val_accuracy: 0.9488
Epoch 6/10
179/179 [==============================] - 1260s 7s/step - loss: 0.2260
- accuracy: 0.9236 - val_loss: 0.1906 - val_accuracy: 0.9272
Epoch 7/10
179/179 [==============================] - 1269s 7s/step - loss: 0.1960
- accuracy: 0.9320 - val_loss: 0.1484 - val_accuracy: 0.9425
Epoch 8/10
179/179 [==============================] - 1292s 7s/step - loss: 0.1685
- accuracy: 0.9425 - val_loss: 0.1181 - val_accuracy: 0.9578
Epoch 9/10
179/179 [==============================] - 1296s 7s/step - loss: 0.1806
- accuracy: 0.9366 - val_loss: 0.2464 - val_accuracy: 0.9233
Epoch 10/10
179/179 [==============================] - 1305s 7s/step - loss: 0.1912
- accuracy: 0.9376 - val_loss: 0.0814 - val_accuracy: 0.9777
180/180 [==============================] - 623s 3s/step - loss: 0.0814
- accuracy: 0.9777
```

In [ ]:

```
from pylab import rcParams

# Plot performance versus params
rcParams['figure.figsize'] = 10, 8 # width 10, height 8
temp = accuracy_table[accuracy_table[param2_label]==param2_list[0]]
ax = temp.plot(x=param_label, y='accuracy', style='bs-', grid=True)
legend_list = [param2_label + '_' + str(param2_list[0])]

color_list = ['r', 'g', 'k', '0.75']
for i in range(1,len(param2_list)):
    temp = accuracy_table[accuracy_table[param2_label]==param2_list[i]]
    ax = temp.plot(x=param_label, y='accuracy', color=color_list[i%len(color_list)], mar
    legend_list.append(param2_label + '_' + str(param2_list[i]))

ax.set_xlabel(param_label)
ax.set_ylabel("accuracy")
plt.rcParams.update({'font.size': 14})
plt.legend(legend_list, loc='center left', bbox_to_anchor=(1.0, 0.5)) # positions legend
# ax.set_xlim([10, 50])
# ax.set_ylim([0, 5])
```

Out[15]:

```
<matplotlib.legend.Legend at 0x2185e7831f0>
```



In [ ]:

```
# Get optimum value for param and param2
temp = accuracy_table[accuracy_table['accuracy'] == accuracy_table['accuracy'].max()]
epochs_opt = temp[param_label].values[0]
batch_size_opt = temp[param2_label].values[0]
print("max Accuracy = %0.3f" % accuracy_table['accuracy'].max())
print("optimum " + param_label + " = " + str(epochs_opt))
print("optimum " + param2_label + " = " + str(batch_size_opt))
```

```
max Accuracy = 97.770
optimum epochs = 10
optimum batch_size = 16
```

In [ ]:

```
######
dropout_opt = 0.6
batch_size_opt = 16
epochs_opt = 10
optimizer_opt = 'Adagrad'
#####
```

Collecting all optimum tunned parameters

In [ ]:

```
d = {'param': ['optimizer', 'epochs', 'batch_size','dropout_prob'],
     'original': [optimizer_initial, epochs_initial, batchsize_initial,dropout_value_ini
     'after_tuning': [optimizer_opt, epochs_opt, batch_size_opt,dropout_opt]}
tuned_params = pd.DataFrame(d)
tuned_params
```

Out[12]:

	param	original	after_tuning
0	optimizer	Adam	Adagrad
1	epochs	4	10
2	batch_size	8	16
3	dropout_prob	0.5	0.6

Creating the final model with optimum parameters

In [ ]:

```
# get train and validation data generator
train_generator_opt,val_generator_opt,test_generator_opt = get_data_generator(train_data

# train and evaluate the model
accuracy, New_model = train_evaluate_the_model(train_generator_opt, val_generator_opt, o
```

```
Found 2870 images belonging to 4 classes.
Found 2870 images belonging to 4 classes.
Found 394 images belonging to 4 classes.

C:\Users\Student\anaconda_3\lib\site-packages\tensorflow\python\keras\engi
ne\training.py:1940: UserWarning: `Model.fit_generator` is deprecated and
will be removed in a future version. Please use `Model.fit`, which support
s generators.
  warnings.warn('`Model.fit_generator` is deprecated and '

Epoch 1/10
179/179 [==============================] - 1324s 7s/step - loss: 0.8610 -
accuracy: 0.6405 - val_loss: 0.4861 - val_accuracy: 0.8010
Epoch 2/10
179/179 [==============================] - 1328s 7s/step - loss: 0.5183 -
accuracy: 0.8024 - val_loss: 0.3342 - val_accuracy: 0.8746
Epoch 3/10
179/179 [==============================] - 1337s 7s/step - loss: 0.3949 -
accuracy: 0.8542 - val_loss: 0.2442 - val_accuracy: 0.9038
Epoch 4/10
179/179 [==============================] - 1335s 7s/step - loss: 0.3253 -
accuracy: 0.8781 - val_loss: 0.1651 - val_accuracy: 0.9481
Epoch 5/10
179/179 [==============================] - 1339s 8s/step - loss: 0.2588 -
accuracy: 0.9036 - val_loss: 0.1493 - val_accuracy: 0.9495
Epoch 6/10
179/179 [==============================] - 1347s 8s/step - loss: 0.2343 -
accuracy: 0.9135 - val_loss: 0.1214 - val_accuracy: 0.9690
Epoch 7/10
179/179 [==============================] - 1345s 8s/step - loss: 0.1878 -
accuracy: 0.9310 - val_loss: 0.1240 - val_accuracy: 0.9547
Epoch 8/10
179/179 [==============================] - 1344s 8s/step - loss: 0.1602 -
accuracy: 0.9418 - val_loss: 0.0851 - val_accuracy: 0.9697
Epoch 9/10
179/179 [==============================] - 1351s 8s/step - loss: 0.1322 -
accuracy: 0.9509 - val_loss: 0.0596 - val_accuracy: 0.9843
Epoch 10/10
179/179 [==============================] - 1356s 8s/step - loss: 0.1188 -
accuracy: 0.9604 - val_loss: 0.0622 - val_accuracy: 0.9819
180/180 [==============================] - 661s 4s/step - loss: 0.0622 - a
ccuracy: 0.9819
```

In [ ]:

```
accuracy
```

Out[15]:

```
98.18815588951111
```

In [1]:

```
pip install -U notebook-as-pdf
```

```
Collecting notebook-as-pdf
  Downloading notebook_as_pdf-0.5.0-py3-none-any.whl (6.5 kB)
Requirement already satisfied: nbconvert in c:\users\pirat\anaconda3\lib\s
ite-packages (from notebook-as-pdf) (6.4.4)
Collecting PyPDF2
  Downloading pypdf2-3.0.1-py3-none-any.whl (232 kB)
     ---------------------------------- 232.6/232.6 kB 212.4 kB/s eta 0:
00:00
Collecting pyppeteer
  Downloading pyppeteer-1.0.2-py3-none-any.whl (83 kB)
     ---------------------------------------- 83.4/83.4 kB 40.7 kB/s eta 0:
00:00
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\pirat\anacond
a3\lib\site-packages (from nbconvert->notebook-as-pdf) (0.8.4)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\pirat\anac
onda3\lib\site-packages (from nbconvert->notebook-as-pdf) (1.5.0)
Requirement already satisfied: nbformat>=4.4 in c:\users\pirat\anaconda3\l
ib\site-packages (from nbconvert->notebook-as-pdf) (5.5.0)
Requirement already satisfied: jinja2>=2.4 in c:\users\pirat\anaconda3\lib
\site-packages (from nbconvert->notebook-as-pdf) (2.11.3)
Requirement already satisfied: traitlets>=5.0 in c:\users\pirat\anaconda3
\lib\site-packages (from nbconvert->notebook-as-pdf) (5.1.1)
Requirement already satisfied: bleach in c:\users\pirat\anaconda3\lib\site
-packages (from nbconvert->notebook-as-pdf) (4.1.0)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\pirat\anacon
da3\lib\site-packages (from nbconvert->notebook-as-pdf) (0.4)
Requirement already satisfied: beautifulsoup4 in c:\users\pirat\anaconda3
\lib\site-packages (from nbconvert->notebook-as-pdf) (4.11.1)
Requirement already satisfied: pygments>=2.4.1 in c:\users\pirat\anaconda3
\lib\site-packages (from nbconvert->notebook-as-pdf) (2.11.2)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in c:\users\pirat\an
aconda3\lib\site-packages (from nbconvert->notebook-as-pdf) (0.5.13)
Requirement already satisfied: jupyter-core in c:\users\pirat\anaconda3\li
b\site-packages (from nbconvert->notebook-as-pdf) (4.11.1)
Requirement already satisfied: testpath in c:\users\pirat\anaconda3\lib\si
te-packages (from nbconvert->notebook-as-pdf) (0.6.0)
Requirement already satisfied: jupyterlab-pygments in c:\users\pirat\anaco
nda3\lib\site-packages (from nbconvert->notebook-as-pdf) (0.1.2)
Requirement already satisfied: defusedxml in c:\users\pirat\anaconda3\lib
\site-packages (from nbconvert->notebook-as-pdf) (0.7.1)
Requirement already satisfied: typing_extensions>=3.10.0.0 in c:\users\pir
at\anaconda3\lib\site-packages (from PyPDF2->notebook-as-pdf) (4.3.0)
Requirement already satisfied: appdirs<2.0.0,>=1.4.3 in c:\users\pirat\ana
conda3\lib\site-packages (from pyppeteer->notebook-as-pdf) (1.4.4)
Requirement already satisfied: certifi>=2021 in c:\users\pirat\anaconda3\l
ib\site-packages (from pyppeteer->notebook-as-pdf) (2022.9.14)
Collecting websockets<11.0,>=10.0
  Downloading websockets-10.4-cp39-cp39-win_amd64.whl (101 kB)
     ---------------------------------- 101.4/101.4 kB 13.8 kB/s eta 0:
00:00
Requirement already satisfied: tqdm<5.0.0,>=4.42.1 in c:\users\pirat\anaco
nda3\lib\site-packages (from pyppeteer->notebook-as-pdf) (4.64.1)
Requirement already satisfied: urllib3<2.0.0,>=1.25.8 in c:\users\pirat\an
aconda3\lib\site-packages (from pyppeteer->notebook-as-pdf) (1.26.11)
Requirement already satisfied: importlib-metadata>=1.4 in c:\users\pirat\a
naconda3\lib\site-packages (from pyppeteer->notebook-as-pdf) (4.11.3)
Collecting pyee<9.0.0,>=8.1.0
  Downloading pyee-8.2.2-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: zipp>=0.5 in c:\users\pirat\anaconda3\lib\s
ite-packages (from importlib-metadata>=1.4->pyppeteer->notebook-as-pdf)
(3.8.0)
```

```
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\pirat\anaconda
3\lib\site-packages (from jinja2>=2.4->nbconvert->notebook-as-pdf) (2.0.1)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\users\pirat\ana
conda3\lib\site-packages (from nbclient<0.6.0,>=0.5.0->nbconvert->notebook
-as-pdf) (7.3.4)
Requirement already satisfied: nest-asyncio in c:\users\pirat\anaconda3\li
b\site-packages (from nbclient<0.6.0,>=0.5.0->nbconvert->notebook-as-pdf)
(1.5.5)
Requirement already satisfied: fastjsonschema in c:\users\pirat\anaconda3
\lib\site-packages (from nbformat>=4.4->nbconvert->notebook-as-pdf) (2.16.
2)
Requirement already satisfied: jsonschema>=2.6 in c:\users\pirat\anaconda3
\lib\site-packages (from nbformat>=4.4->nbconvert->notebook-as-pdf) (4.16.
0)
Requirement already satisfied: colorama in c:\users\pirat\anaconda3\lib\si
te-packages (from tqdm<5.0.0,>=4.42.1->pyppeteer->notebook-as-pdf) (0.4.5)
Requirement already satisfied: soupsieve>1.2 in c:\users\pirat\anaconda3\l
ib\site-packages (from beautifulsoup4->nbconvert->notebook-as-pdf) (2.3.1)
Requirement already satisfied: six>=1.9.0 in c:\users\pirat\anaconda3\lib
\site-packages (from bleach->nbconvert->notebook-as-pdf) (1.16.0)
Requirement already satisfied: webencodings in c:\users\pirat\anaconda3\li
b\site-packages (from bleach->nbconvert->notebook-as-pdf) (0.5.1)
Requirement already satisfied: packaging in c:\users\pirat\anaconda3\lib\s
ite-packages (from bleach->nbconvert->notebook-as-pdf) (21.3)
Requirement already satisfied: pywin32>=1.0 in c:\users\pirat\anaconda3\li
b\site-packages (from jupyter-core->nbconvert->notebook-as-pdf) (302)
Requirement already satisfied: attrs>=17.4.0 in c:\users\pirat\anaconda3\l
ib\site-packages (from jsonschema>=2.6->nbformat>=4.4->nbconvert->notebook
-as-pdf) (21.4.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.1
4.0 in c:\users\pirat\anaconda3\lib\site-packages (from jsonschema>=2.6->n
bformat>=4.4->nbconvert->notebook-as-pdf) (0.18.0)
Requirement already satisfied: tornado>=6.0 in c:\users\pirat\anaconda3\li
b\site-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbcon
vert->notebook-as-pdf) (6.1)
Requirement already satisfied: pyzmq>=23.0 in c:\users\pirat\anaconda3\lib
\site-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconv
ert->notebook-as-pdf) (23.2.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\pirat\an
aconda3\lib\site-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.
5.0->nbconvert->notebook-as-pdf) (2.8.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\pirat
\anaconda3\lib\site-packages (from packaging->bleach->nbconvert->notebook-
as-pdf) (3.0.9)
Installing collected packages: pyee, websockets, PyPDF2, pyppeteer, notebo
ok-as-pdf
Successfully installed PyPDF2-3.0.1 notebook-as-pdf-0.5.0 pyee-8.2.2 pyppe
teer-1.0.2 websockets-10.4
Note: you may need to restart the kernel to use updated packages.
```

In [3]:

```
pip install pyppeteer
```

```
Requirement already satisfied: pyppeteer in c:\users\pirat\anaconda3\lib\s
ite-packages (1.0.2)
Requirement already satisfied: importlib-metadata>=1.4 in c:\users\pirat\a
naconda3\lib\site-packages (from pyppeteer) (4.11.3)
Requirement already satisfied: pyee<9.0.0,>=8.1.0 in c:\users\pirat\anacon
da3\lib\site-packages (from pyppeteer) (8.2.2)
Requirement already satisfied: urllib3<2.0.0,>=1.25.8 in c:\users\pirat\an
aconda3\lib\site-packages (from pyppeteer) (1.26.11)
Requirement already satisfied: certifi>=2021 in c:\users\pirat\anaconda3\l
ib\site-packages (from pyppeteer) (2022.9.14)
Requirement already satisfied: tqdm<5.0.0,>=4.42.1 in c:\users\pirat\anaco
nda3\lib\site-packages (from pyppeteer) (4.64.1)
Requirement already satisfied: websockets<11.0,>=10.0 in c:\users\pirat\an
aconda3\lib\site-packages (from pyppeteer) (10.4)
Requirement already satisfied: appdirs<2.0.0,>=1.4.3 in c:\users\pirat\ana
conda3\lib\site-packages (from pyppeteer) (1.4.4)
Requirement already satisfied: zipp>=0.5 in c:\users\pirat\anaconda3\lib\s
ite-packages (from importlib-metadata>=1.4->pyppeteer) (3.8.0)
Requirement already satisfied: colorama in c:\users\pirat\anaconda3\lib\si
te-packages (from tqdm<5.0.0,>=4.42.1->pyppeteer) (0.4.5)
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:

```

```