```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator

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

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

```
IMG_SIZE = 224
BATCH_SIZE = 32

train_datagen=ImageDataGenerator(rescale=1./255,validation_split=0.2)

train_generator=train_datagen.flow_from_directory(
     '/content/drive/MyDrive/Alzimer_segment dataset',
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training'
)
```

```
Found 1025 images belonging to 4 classes.
```

```
val_generator=train_datagen.flow_from_directory(
  '/content/drive/MyDrive/Alzimer_segment dataset',
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 254 images belonging to 4 classes.
```

```
class_names=list(train_generator.class_indices.keys())
print(class_names)
print(train_generator.class_indices)
```

```
['MildDemented', 'Moderate Demented', 'Non Demented', 'very mild
Demented']
{'MildDemented': 0, 'Moderate Demented': 1, 'Non Demented': 2, 'very
mild Demented': 3}
```

```
model = tf.keras.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(IMG_SIZE,IMG_SIZE,3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
```

```
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(4, activation='softmax') # Change units to number of
classes
])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

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

```
model.summary()
```

```
Model: "sequential_3"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━
┓
┃ Layer (type)                    ┃ Output Shape           ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━
┩
│ conv2d_9 (Conv2D)               │ (None, 222, 222, 32)   │
896 │
├─────────────────────────────────┼────────────────────────┼──────────────
─┤
│ max_pooling2d_9 (MaxPooling2D)  │ (None, 111, 111, 32)   │
0 │
├─────────────────────────────────┼────────────────────────┼──────────────
─┤
│ conv2d_10 (Conv2D)              │ (None, 109, 109, 64)   │
18,496 │
├─────────────────────────────────┼────────────────────────┼──────────────
─┤
│ max_pooling2d_10 (MaxPooling2D) │ (None, 54, 54, 64)     │
0 │
├─────────────────────────────────┼────────────────────────┼──────────────
─┤
│ conv2d_11 (Conv2D)              │ (None, 52, 52, 128)    │
73,856 │
├─────────────────────────────────┼────────────────────────┼──────────────
─┤
│ max_pooling2d_11 (MaxPooling2D) │ (None, 26, 26, 128)    │
0 │
```

```
├──────────────────────────────────┼─────────────────────────┼──────────
───────┤
│ flatten_3 (Flatten)              │ (None, 86528)           │
0 │
├──────────────────────────────────┼─────────────────────────┼──────────
───────┤
│ dense_6 (Dense)                  │ (None, 128)             │
11,075,712 │
├──────────────────────────────────┼─────────────────────────┼──────────
───────┤
│ dense_7 (Dense)                  │ (None, 4)               │
516 │
└──────────────────────────────────┴─────────────────────────┴──────────
───────┘

 Total params: 11,169,476 (42.61 MB)

 Trainable params: 11,169,476 (42.61 MB)

 Non-trainable params: 0 (0.00 B)
```

v

```
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy']) # Corrected loss function name

model.fit(train_generator, epochs=3, validation_data=val_generator,
batch_size=BATCH_SIZE)

Epoch 1/3
33/33 ━━━━━━━━━━━━━━━━━━━━ 0s 4s/step - accuracy: 0.4192 - loss:
1.1766

/usr/local/lib/python3.11/dist-packages/keras/src/trainers/
data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset`
class should call `super().__init__(**kwargs)` in its constructor.
`**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()

 33/33 ━━━━━━━━━━━━━━━━━━━━ 188s 6s/step - accuracy: 0.4206 - loss:
1.1741 - val_accuracy: 0.5039 - val_loss: 0.9974
Epoch 2/3
33/33 ━━━━━━━━━━━━━━━━━━━━ 129s 4s/step - accuracy: 0.5266 - loss:
0.9795 - val_accuracy: 0.3701 - val_loss: 1.0434
Epoch 3/3
33/33 ━━━━━━━━━━━━━━━━━━━━ 131s 4s/step - accuracy: 0.5182 - loss:
0.9615 - val_accuracy: 0.5472 - val_loss: 0.9113

<keras.src.callbacks.history.History at 0x7c4f3fedb290>
```

```
model.save('/content/drive/MyDrive/Alzimer_segment dataset.h5')
```

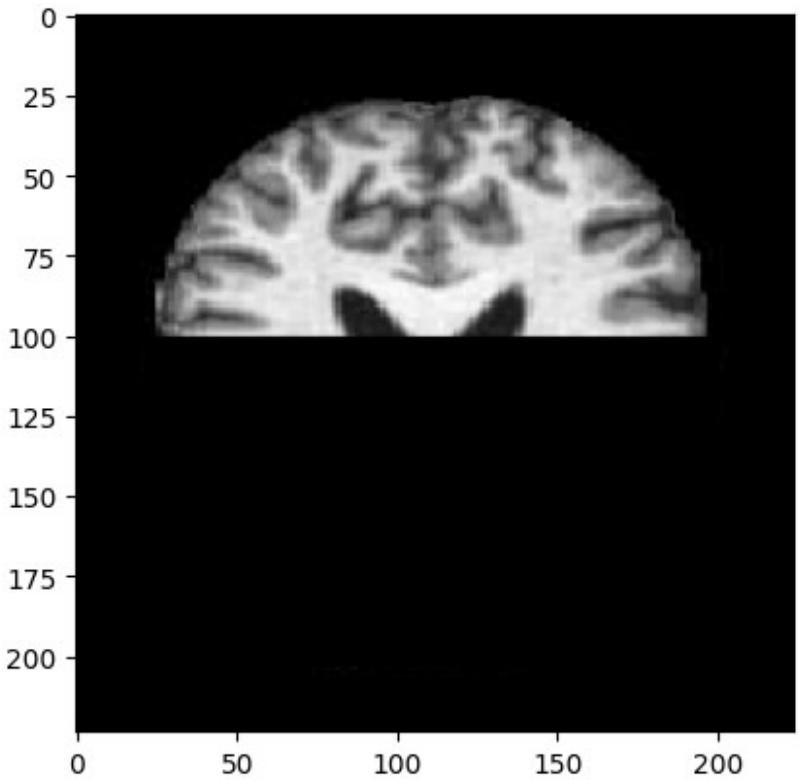
```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
import numpy as np
model = load_model('/content/drive/MyDrive/Alzimer_segment dataset.h5')
print("Model Loaded Sucessfully")
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
```

```
Model Loaded Sucessfully
```

```
test_image_path="/content/drive/MyDrive/Alzimer_segment dataset/Non Demented/26 (100).jpg"
img=image.load_img(test_image_path,target_size=(224,224,3))
plt.imshow(img)
plt.axis()
plt.show()
```




```
img_array=img = image.img_to_array(img)
img_array=np.expand_dims(img_array,axis=0)
img_array=img_array/255

 prediction = model.predict(img_array)
 ind=np.argmax(prediction)
 prediction=prediction[0][ind]
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 68ms/step
```

```
prediction=model.predict(img_array)
print(prediction)
ind=np.argmax(prediction)
print(class_names[ind])
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 69ms/step
[[0.19756073 0.01930289 0.49096417 0.29217228]]
Non Demented
```