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

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/Alzheimer/train',
target_size=(IMG_SIZE,IMG_SIZE),
batch_size=BATCH_SIZE,
class_mode='categorical',
subset='training'
)
```

```
Found 3342 images belonging to 4 classes.
```

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

```
Found 834 images belonging to 4 classes.
```

```
class_indices=train_generator.class_indices
class_names=list(class_indices.keys())
print("class indices",class_indices)
print("class name:",class_names)
```

```
class indices {'MildDemented': 0, 'ModerateDemented': 1,
'NonDemented': 2, 'VeryMildDemented': 3}
class name: ['MildDemented', 'ModerateDemented', 'NonDemented',
'VeryMildDemented']
```

```
model=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')
])
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/
convolutional/base_conv.py:113: 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.summary()
```

```
Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━
━━━━━┓
┃ Layer (type)                     ┃ Output Shape            ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━
━━━━━┩
│ conv2d (Conv2D)                  │ (None, 222, 222, 32)    │
896 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ max_pooling2d (MaxPooling2D)     │ (None, 111, 111, 32)    │
0 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ conv2d_1 (Conv2D)                │ (None, 109, 109, 64)    │
18,496 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ max_pooling2d_1 (MaxPooling2D)   │ (None, 54, 54, 64)      │
0 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ conv2d_2 (Conv2D)                │ (None, 52, 52, 128)     │
73,856 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ max_pooling2d_2 (MaxPooling2D)   │ (None, 26, 26, 128)     │
0 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ flatten (Flatten)                │ (None, 86528)           │
0 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ dense (Dense)                    │ (None, 128)             │
```

```
11,075,712 │
├─────────────────────────────────┼────────────────────────┼───────
─────┤
│ dense_1 (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)
```

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

```
model.fit(train_generator,epochs=5,validation_data=val_generator,batch
_size=BATCH_SIZE)
```

```
/usr/local/lib/python3.12/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()

Epoch 1/5
105/105 ━━━━━━━━━━━━━━━━━━━━ 1363s 13s/step - accuracy: 0.3020 - loss:
1.6687 - val_accuracy: 0.4556 - val_loss: 1.2130
Epoch 2/5
105/105 ━━━━━━━━━━━━━━━━━━━━ 373s 4s/step - accuracy: 0.5633 - loss:
0.9845 - val_accuracy: 0.6247 - val_loss: 0.8576
Epoch 3/5
105/105 ━━━━━━━━━━━━━━━━━━━━ 383s 4s/step - accuracy: 0.6678 - loss:
0.7434 - val_accuracy: 0.6463 - val_loss: 0.7974
Epoch 4/5
105/105 ━━━━━━━━━━━━━━━━━━━━ 378s 4s/step - accuracy: 0.7568 - loss:
0.5777 - val_accuracy: 0.6223 - val_loss: 0.8407
Epoch 5/5
105/105 ━━━━━━━━━━━━━━━━━━━━ 376s 4s/step - accuracy: 0.8112 - loss:
0.4510 - val_accuracy: 0.6715 - val_loss: 0.8575

<keras.src.callbacks.history.History at 0x7d9fd0447260>
```

```
model.save('/content/drive/MyDrive/Alzheimer/alzheimer.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
```

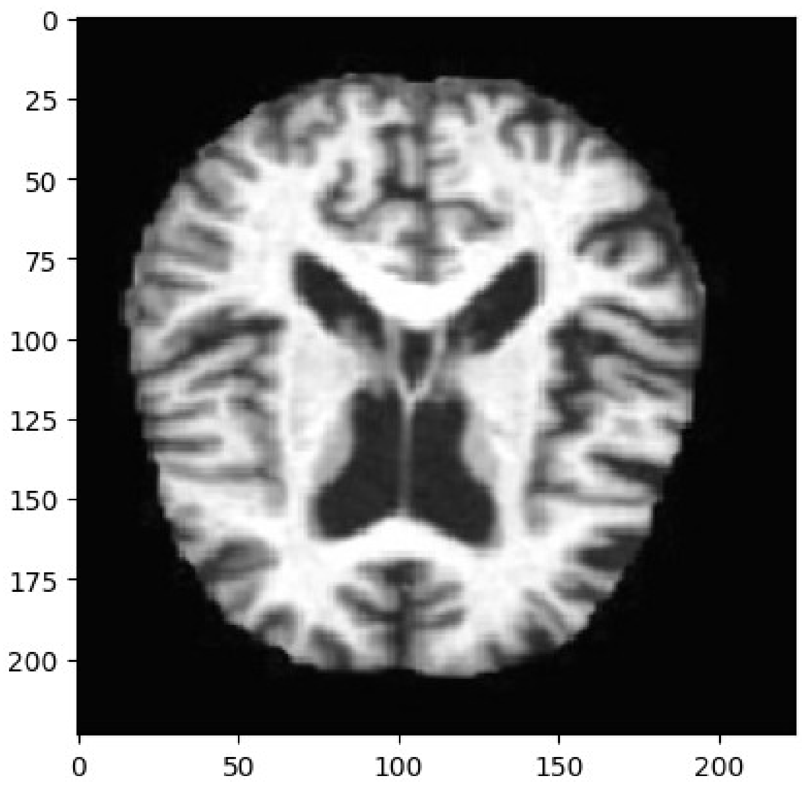
```
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.
```

```
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/Alzheimer/alzheimer.h5')
print("Model Loaded")
```

```
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
```

```
test_image_path="/content/drive/MyDrive/Alzheimer/train/
VeryMildDemented/d0111dbc-5718-4e5f-87ab-9c34391d8d09.jpg"
img=image.load_img(test_image_path,target_size=(224,224))
plt.imshow(img)
plt.axis()
plt.show()
```



```python
img_array=image.img_to_array(img)
img_array=np.expand_dims(img_array,axis=0)
img_array=img_array/255.0

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

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 164ms/step
NonDemented
```