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

IMG_SIZE = (256, 256)
BATCH_SIZE = 32

train_datagen = ImageDataGenerator(rescale=1./255,
validation_split=0.2)
train_generator=train_datagen.flow_from_directory(
    '/content/drive/MyDrive/Alzheimer_Dataset/data',
    target_size = IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training'
)
```

```
Found 160 images belonging to 4 classes.
```

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

```
Found 40 images belonging to 4 classes.
```

```
Class_indices=train_generator.class_indices
Class_names=list(Class_indices.keys())
print("Class indices:",Class_indices)
print("Class names:",Class_names)
```

```
Class indices: {'Mild Demented': 0, 'Moderate Demented': 1, 'Non
Demented': 2, 'VeryMild Demented': 3}
Class names: ['Mild Demented', 'Moderate Demented', 'Non Demented',
'VeryMild Demented']
```

```
model = keras.Sequential([
    layers.Conv2D(32, (3,3), activation='relu',
input_shape=(IMG_SIZE[0],IMG_SIZE[1],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/python3.11/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.summary()
```

```
Model: "sequential"
```

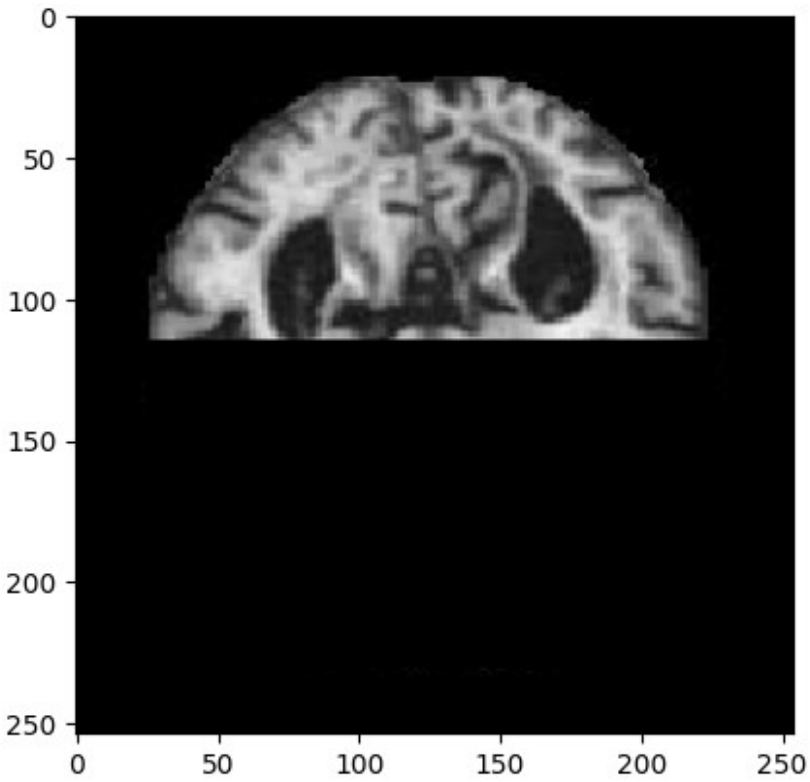
```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                      │ (None, 254, 254, 32)        │
896 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d (MaxPooling2D)         │ (None, 127, 127, 32)        │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_1 (Conv2D)                    │ (None, 125, 125, 64)        │
18,496 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_1 (MaxPooling2D)       │ (None, 62, 62, 64)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_2 (Conv2D)                    │ (None, 60, 60, 128)         │
73,856 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_2 (MaxPooling2D)       │ (None, 30, 30, 128)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ flatten (Flatten)                    │ (None, 115200)              │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense (Dense)                        │ (None, 128)                 │
```

```
14,745,728 │
├─────────────────────────────────────┼─────────────────────────────────┼
──────────────┤
│ dense_1 (Dense)                     │ (None, 4)                       │
516 │
└─────────────────────────────────────┴─────────────────────────────────┴
──────────────┘

 Total params: 14,839,492 (56.61 MB)

 Trainable params: 14,839,492 (56.61 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/python3.11/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()

Epoch 1/5
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 6s/step - accuracy: 0.2421 - loss: 1.8702

/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()

 5/5 ━━━━━━━━━━━━━━━━━━━━ 51s 10s/step - accuracy: 0.2455 - loss:
1.8596 - val_accuracy: 0.2750 - val_loss: 1.3824
Epoch 2/5
5/5 ━━━━━━━━━━━━━━━━━━━━ 31s 7s/step - accuracy: 0.3503 - loss: 1.3736
- val_accuracy: 0.2750 - val_loss: 1.3762
Epoch 3/5
5/5 ━━━━━━━━━━━━━━━━━━━━ 30s 6s/step - accuracy: 0.3661 - loss: 1.3551
- val_accuracy: 0.6000 - val_loss: 1.3449
Epoch 4/5
5/5 ━━━━━━━━━━━━━━━━━━━━ 29s 6s/step - accuracy: 0.4589 - loss: 1.2616
- val_accuracy: 0.5250 - val_loss: 1.2843
```

```
Epoch 5/5
5/5 ━━━━━━━━━━━━━━━━━━━━ 28s 6s/step - accuracy: 0.5265 - loss: 1.1121
- val_accuracy: 0.4250 - val_loss: 1.1602

<keras.src.callbacks.history.History at 0x7f6a9719d410>

model.save('/MyDrive/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('/MyDrive/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_Dataset/data/Mild
Demented/mildDem10.jpg"
img = image.load_img(test_image_path, target_size=(254, 254))
plt.imshow(img)
plt.axis()
plt.show()
```




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

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

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 174ms/step
Mild Demented
```