Q. **Write a program to implement CNN using image dataset.**

```
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split

dataset_dir = 'Tea leaves/tea sickness dataset'       # Replace with the correct path
image_size = (128, 128)
batch_size = 32
num_classes = 8

train_datagen = ImageDataGenerator(
    rescale=1.0/255.0,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2     # 20% validation data
)

# Load training and validation data
train_generator = train_datagen.flow_from_directory(
    dataset_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

validation_generator = train_datagen.flow_from_directory(
    dataset_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 711 images belonging to 8 classes.
Found 174 images belonging to 8 classes.
```

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    Flatten(),

    Dense(128, activation='relu'),
    Dropout(0.5),

    Dense(num_classes, activation='softmax')
])
```

```
d:\Python12\Lib\site-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='categorical_crossentropy', metrics=['accuracy'])

# Early stopping callback to prevent overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=5)

history = model.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=20,
    callbacks=[early_stopping]
)
```

```
d:\Python12\Lib\site-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 1/20
23/23 ━━━━━━━━━━━━━━━━━━━━ 65s 3s/step - accuracy: 0.1531 - loss: 2.2671 - val_accuracy: 0.1609 - val_loss: 1.9789
Epoch 2/20
23/23 ━━━━━━━━━━━━━━━━━━━━ 64s 3s/step - accuracy: 0.2409 - loss: 1.8760 - val_accuracy: 0.3448 - val_loss: 1.4833
Epoch 3/20
23/23 ━━━━━━━━━━━━━━━━━━━━ 67s 3s/step - accuracy: 0.3042 - loss: 1.5140 - val_accuracy: 0.3506 - val_loss: 1.3622
Epoch 4/20
23/23 ━━━━━━━━━━━━━━━━━━━━ 68s 3s/step - accuracy: 0.3939 - loss: 1.3891 - val_accuracy: 0.4023 - val_loss: 1.2157
Epoch 5/20
23/23 ━━━━━━━━━━━━━━━━━━━━ 51s 2s/step - accuracy: 0.3800 - loss: 1.2869 - val_accuracy: 0.4425 - val_loss: 1.1804
Epoch 6/20
```

```
23/23 ━━━━━━━━━━━━━━━━━━━━ 47s 2s/step - accuracy: 0.4147 - loss: 1.2598 -
val_accuracy: 0.3448 - val_loss: 1.2800
Epoch 7/20
23/23 ━━━━━━━━━━━━━━━━━━━━ 48s 2s/step - accuracy: 0.4126 - loss: 1.2366 -
val_accuracy: 0.4080 - val_loss: 1.1436
Epoch 8/20
23/23 ━━━━━━━━━━━━━━━━━━━━ 45s 2s/step - accuracy: 0.3834 - loss: 1.2161 -
val_accuracy: 0.4195 - val_loss: 1.1864
Epoch 9/20
23/23 ━━━━━━━━━━━━━━━━━━━━ 44s 2s/step - accuracy: 0.4594 - loss: 1.1879 -
val_accuracy: 0.4310 - val_loss: 1.1338
Epoch 10/20
23/23 ━━━━━━━━━━━━━━━━━━━━ 46s 2s/step - accuracy: 0.4622 - loss: 1.1429 -
val_accuracy: 0.4540 - val_loss: 1.0894
Epoch 11/20
23/23 ━━━━━━━━━━━━━━━━━━━━ 45s 2s/step - accuracy: 0.4155 - loss: 1.1673 -
val_accuracy: 0.4310 - val_loss: 1.1316
Epoch 12/20
23/23 ━━━━━━━━━━━━━━━━━━━━ 52s 2s/step - accuracy: 0.4182 - loss: 1.1722 -
val_accuracy: 0.4828 - val_loss: 1.0611
Epoch 13/20
23/23 ━━━━━━━━━━━━━━━━━━━━ 45s 2s/step - accuracy: 0.4703 - loss: 1.0676 -
val_accuracy: 0.4828 - val_loss: 1.0725
Epoch 14/20
23/23 ━━━━━━━━━━━━━━━━━━━━ 41s 2s/step - accuracy: 0.4721 - loss: 1.1274 -
val_accuracy: 0.5517 - val_loss: 1.0067
Epoch 15/20
23/23 ━━━━━━━━━━━━━━━━━━━━ 42s 2s/step - accuracy: 0.5270 - loss: 0.9798 -
val_accuracy: 0.4368 - val_loss: 1.1473
Epoch 16/20
23/23 ━━━━━━━━━━━━━━━━━━━━ 44s 2s/step - accuracy: 0.4603 - loss: 1.2426 -
val_accuracy: 0.4310 - val_loss: 1.1645
Epoch 17/20
23/23 ━━━━━━━━━━━━━━━━━━━━ 50s 2s/step - accuracy: 0.4637 - loss: 1.1036 -
val_accuracy: 0.4713 - val_loss: 1.1643
Epoch 18/20
23/23 ━━━━━━━━━━━━━━━━━━━━ 75s 3s/step - accuracy: 0.5239 - loss: 1.0683 -
val_accuracy: 0.4540 - val_loss: 1.0965
Epoch 19/20
23/23 ━━━━━━━━━━━━━━━━━━━━ 71s 3s/step - accuracy: 0.5791 - loss: 0.9652 -
val_accuracy: 0.5057 - val_loss: 1.0394
```

```
loss, accuracy = model.evaluate(validation_generator)
print(f'Validation accuracy: {accuracy*100:.2f}%')
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 11s 2s/step - accuracy: 0.4969 - loss: 0.9733
Validation accuracy: 47.70%
```