```
# Mount Google Drive
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).
```

```
# Define paths
import os

dataset_path = "/content/drive/MyDrive/Colab Notebooks/archive.zip (Unzipped Files)"
train_dir = dataset_path + "/train"
valid_dir = dataset_path + "/valid"

print("Classes in train directory:", os.listdir(train_dir))
print("Classes in valid directory:", os.listdir(valid_dir)
)
```

```
Classes in train directory: ['Ramipril 5 MG', 'Atomoxetine 25 MG', 'Oseltamivir 45 MG', 'Calcitriol 0.00025 MG', 'Amoxicillin 500 MG
Classes in valid directory: ['Oseltamivir 45 MG', 'Calcitriol 0.00025 MG', 'Ramipril 5 MG', 'Amoxicillin 500 MG', 'Atomoxetine 25 MG
```

```
# Import necessary libraries
import cv2
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
import numpy as np
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
import pandas as pd


# Define constants
IMG_SIZE = 224
BATCH_SIZE = 32
NUM_CLASSES = 20  # Number of classes


from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define ImageDataGenerator for preprocessing
datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    validation_split=0.2  # Splitting data into training and validation
)

# Define directories (ensure these variables are properly assigned)
train_dir = "/content/drive/MyDrive/Colab Notebooks/archive.zip (Unzipped Files)/train"
valid_dir =  "/content/drive/MyDrive/Colab Notebooks/archive.zip (Unzipped Files)/valid"

IMG_SIZE = 224  # Adjust as needed
BATCH_SIZE = 32  # Adjust as needed

# Creating training and validation data generators
train_generator = datagen.flow_from_directory(
    train_dir,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode="categorical",
    subset="training"  # Training subset
)

valid_generator = datagen.flow_from_directory(
    valid_dir,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode="categorical",
    subset="validation"  # Validation subset
)
```

```
Found 803 images belonging to 20 classes.
Found 44 images belonging to 20 classes.
```

```
from tensorflow.keras import Input

model = Sequential([
    Input(shape=(IMG_SIZE, IMG_SIZE, 3)),  # Define input layer explicitly
    Conv2D(32, (3,3), activation='relu'),
```

```
    MaxPooling2D(2,2),

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

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

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(NUM_CLASSES, activation='softmax')
])


import matplotlib.pyplot as plt
import numpy as np

# A batch of images and labels
images, labels = next(train_generator)

# Plot some images
plt.figure(figsize=(10, 10))
for i in range(9):
    plt.subplot(3, 3, i + 1)
    plt.imshow(images[i])
    plt.axis("off")
plt.show()
```


```
# Compile the model with categorical loss
model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# Display model summary
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
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 20)	2,580

Total params: 11,171,540 (42.62 MB)

```
model.add(Dense(20, activation='softmax'))  # Ensure 20 classes in the final layer


print(train_generator.class_indices)  # Should match your output neurons
```

```
{'Amoxicillin 500 MG': 0, 'Atomoxetine 25 MG': 1, 'Calcitriol 0.00025 MG': 2, 'Oseltamivir 45 MG': 3, 'Ramipril 5 MG': 4, 'apixaban
```

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


train_generator = datagen.flow_from_directory(
    train_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'  # Ensure this matches model output
)
```

```
Found 994 images belonging to 20 classes.
```

```
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten

# Load a pre-trained model without the top layer
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Freeze the base model layers
base_model.trainable = False

# Build the new model
model = Sequential([
    base_model,
    Flatten(),
    Dense(512, activation='relu'),
    Dense(256, activation='relu'),
    Dense(20, activation='softmax')  # Ensure this matches your number of classes
])

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

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_nc
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
num_classes = len(train_generator.class_indices)
print("Number of classes:", num_classes)
```

```
Number of classes: 20
```

```
datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
```

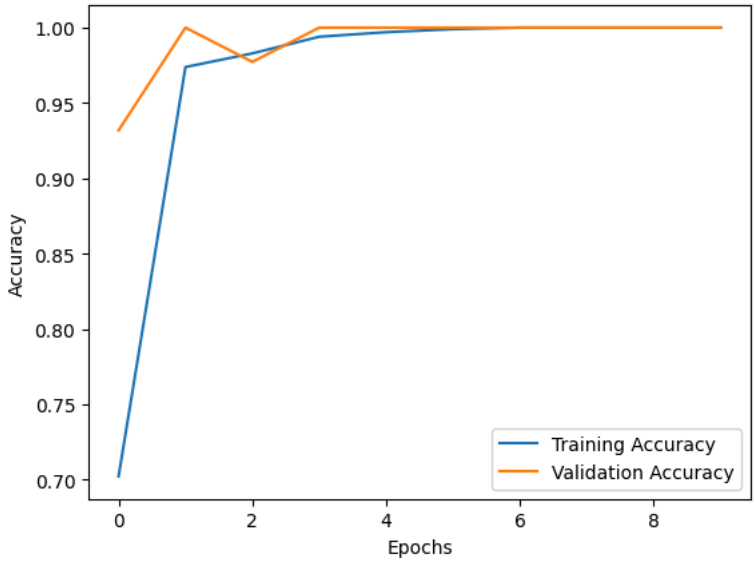
```
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)


from tensorflow.keras.optimizers import Adam
model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])


history = model.fit(
    train_generator,
    epochs=10,
    validation_data=valid_generator
)
```

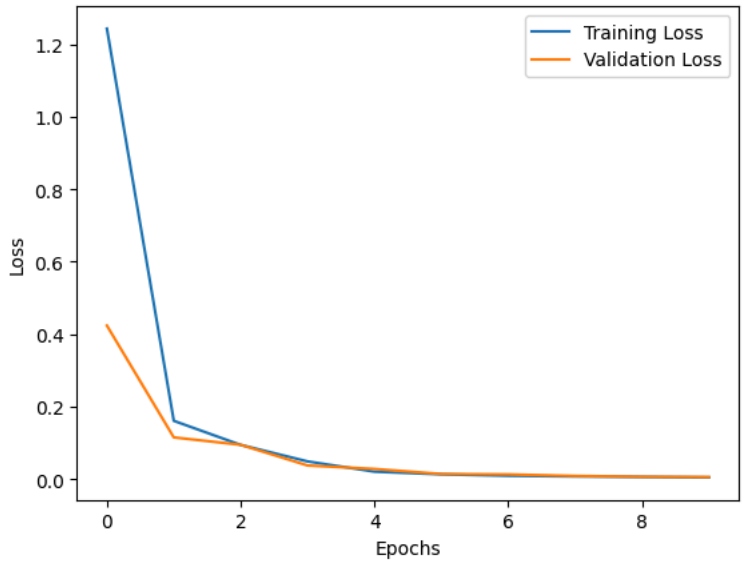
```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` cl
  self._warn_if_super_not_called()
Epoch 1/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 422s 13s/step - accuracy: 0.4811 - loss: 1.9817 - val_accuracy: 0.9318 - val_loss: 0.4237
Epoch 2/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 30s 249ms/step - accuracy: 0.9675 - loss: 0.1907 - val_accuracy: 1.0000 - val_loss: 0.1152
Epoch 3/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 236ms/step - accuracy: 0.9873 - loss: 0.0892 - val_accuracy: 0.9773 - val_loss: 0.0947
Epoch 4/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 243ms/step - accuracy: 0.9910 - loss: 0.0617 - val_accuracy: 1.0000 - val_loss: 0.0376
Epoch 5/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 243ms/step - accuracy: 0.9971 - loss: 0.0220 - val_accuracy: 1.0000 - val_loss: 0.0281
Epoch 6/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 7s 224ms/step - accuracy: 0.9987 - loss: 0.0129 - val_accuracy: 1.0000 - val_loss: 0.0142
Epoch 7/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 248ms/step - accuracy: 1.0000 - loss: 0.0085 - val_accuracy: 1.0000 - val_loss: 0.0131
Epoch 8/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 7s 223ms/step - accuracy: 1.0000 - loss: 0.0070 - val_accuracy: 1.0000 - val_loss: 0.0088
Epoch 9/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 246ms/step - accuracy: 1.0000 - loss: 0.0067 - val_accuracy: 1.0000 - val_loss: 0.0065
Epoch 10/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 240ms/step - accuracy: 1.0000 - loss: 0.0044 - val_accuracy: 1.0000 - val_loss: 0.0058
```

```
#Evaluate on validation set
loss, accuracy = model.evaluate(valid_generator)
print(f"Validation Loss: {loss:.4f}")
print(f"Validation Accuracy: {accuracy * 100:.2f}%")
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 94ms/step - accuracy: 1.0000 - loss: 0.0059
Validation Loss: 0.0058
Validation Accuracy: 100.00%
```

```
import matplotlib.pyplot as plt

# Plot accuracy
plt.plot(history.history['accuracy'], label="Training Accuracy")
plt.plot(history.history['val_accuracy'], label="Validation Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()

# Plot loss
plt.plot(history.history['loss'], label="Training Loss")
plt.plot(history.history['val_loss'], label="Validation Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```

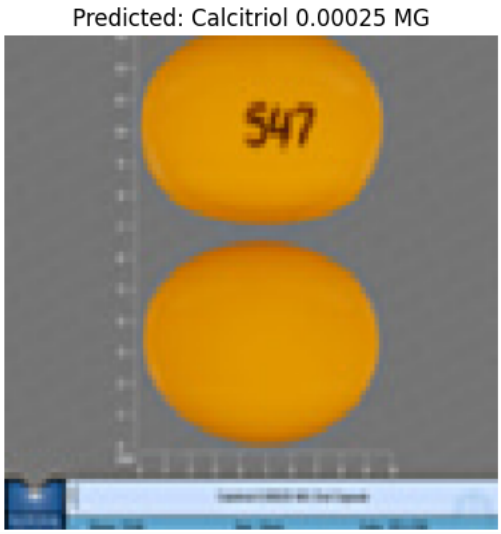





```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image

# Define image size
IMG_SIZE = 224  # Ensure this matches your model input size

# Define the class names based on your dataset
class_names = {
    0: "Amoxicillin 500 MG",
    1: "Atomoxetine 25 MG",
    2: "Calcitriol 0.00025 MG",
    3: "Oseltamivir 45 MG",
    4: "Ramipril 5 MG",
    5: "apixaben 2.5 MG",
    6: "aprepitant 80 MG",
    7: "benzonatate 100 MG",
    8: "carvedilol 3.125 MG",
    9: "celecoxib 200 MG",
    10: "duloxetine 30 MG",
    11: "eltrombopag 25 MG",
    12: "montelukast 10 MG",
    13: "mycophenolate mofetil 250 MG",
    14: "pantoprazole 40 MG",
    15: "pitavastatin 1 MG",
    16: "prasugrel 10 MG",
    17: "saxagliptin 5 MG",
    18: "sitagliptin 50 MG",
    19: "tadalafil 5 MG"
}

def predict_pill(image_path, model):
    # Load and preprocess the image
    img = image.load_img(image_path, target_size=(IMG_SIZE, IMG_SIZE))
    img_array = image.img_to_array(img) / 255.0  # Normalize pixel values
```

```
    img_array = np.expand_dims(img_array, axis=0)  # Expand dimensions for model input

    # Make prediction
    prediction = model.predict(img_array)
    predicted_class = np.argmax(prediction)  # Get the index of highest probability class
    predicted_pill = class_names.get(predicted_class, "Unknown Pill")

    # Print and display results
    print(f"Predicted Pill: {predicted_pill}")
    plt.imshow(img)
    plt.axis("off")
    plt.title(f"Predicted: {predicted_pill}")
    plt.show()


# Example prediction
img_path = "/content/drive/MyDrive/Colab Notebooks/archive.zip (Unzipped Files)/valid/Calcitriol 0.00025 MG/Calcitriol 0.00025 MG (10).jp
predict_pill(img_path,model)

# Batch prediction for all images in valid directory
folder_path = valid_dir + "/images"
predictions = []
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
Predicted Pill: Calcitriol 0.00025 MG
```




```
import os

folder_path = "/content/drive/MyDrive/Colab Notebooks/archive.zip (Unzipped Files)/valid/Amoxicillin 500 MG"

for filename in os.listdir(folder_path):
    img_path = os.path.join(folder_path, filename)
    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img) / 255.0
    img_array = np.expand_dims(img_array, axis=0)

    prediction = model.predict(img_array)
    predicted_class = np.argmax(prediction)
    predicted_pill = class_names.get(predicted_class, "Unknown Pill")

    print(f"{filename} -> Identified as: {predicted_pill}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
Amoxicillin 500 MG (2) - Copy - Copy - Copy.jpg -> Identified as: Amoxicillin 500 MG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
Amoxicillin 500 MG (15) - Copy.jpg -> Identified as: Amoxicillin 500 MG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
Amoxicillin 500 MG (3).jpg -> Identified as: Amoxicillin 500 MG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
Amoxicillin 500 MG (20) - Copy.jpg -> Identified as: Amoxicillin 500 MG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
Amoxicillin 500 MG (1).jpg -> Identified as: Amoxicillin 500 MG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
Amoxicillin 500 MG (4).jpg -> Identified as: Amoxicillin 500 MG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
Amoxicillin 500 MG (13) - Copy - Copy.jpg -> Identified as: Amoxicillin 500 MG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
Amoxicillin 500 MG (13) - Copy.jpg -> Identified as: Amoxicillin 500 MG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
Amoxicillin 500 MG (22) - Copy.jpg -> Identified as: Amoxicillin 500 MG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
```

```
Amoxicillin 500 MG (24) - Copy.jpg -> Identified as: Amoxicillin 500 MG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
Amoxicillin 500 MG (10).jpg -> Identified as: Amoxicillin 500 MG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
Amoxicillin 500 MG (9).jpg -> Identified as: Amoxicillin 500 MG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
Amoxicillin 500 MG (14) - Copy.jpg -> Identified as: Amoxicillin 500 MG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
Amoxicillin 500 MG (11) - Copy.jpg -> Identified as: Amoxicillin 500 MG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
Amoxicillin 500 MG (6).jpg -> Identified as: Amoxicillin 500 MG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
Amoxicillin 500 MG (12).jpg -> Identified as: Amoxicillin 500 MG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step
Amoxicillin 500 MG (5).jpg -> Identified as: Amoxicillin 500 MG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step
Amoxicillin 500 MG (7).jpg -> Identified as: Amoxicillin 500 MG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step
Amoxicillin 500 MG (8).jpg -> Identified as: Amoxicillin 500 MG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
Amoxicillin 500 MG (2).jpg -> Identified as: Amoxicillin 500 MG
```

```
import pandas as pd

predictions = []

for filename in os.listdir(folder_path):
    img_path = os.path.join(folder_path, filename)
    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img) / 255.0
    img_array = np.expand_dims(img_array, axis=0)

    prediction = model.predict(img_array)
    predicted_class = np.argmax(prediction)
    predicted_pill = class_names.get(predicted_class, "Unknown Pill")

    predictions.append({"Filename": filename, "Predicted Pill": predicted_pill})

# Save to CSV
df = pd.DataFrame(predictions)
df.to_csv("pill_predictions.csv", index=False)
print("Predictions saved to pill_predictions.csv")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
Predictions saved to pill_predictions.csv
```

```
import matplotlib.pyplot as plt
# Load image
img = image.load_img(img_path, target_size=(224, 224))
plt.imshow(img)
plt.axis("off")
plt.title(f"Identified as: {predicted_pill}")
plt.show()
```

Identified as: Amoxicillin 500 MG