```
Dataset/
   ├── Class1/
   ├── Class2/
   ├── Class3/
```

```
  File "/tmp/ipython-input-1190829271.py", line 2
    ├── Class1/
    ^
SyntaxError: invalid character '├' (U+251C)
```

Next steps: Explain error

```
import os

# Auto-detect folder name inside /content
contents = os.listdir("/content")
print("Folders in /content:", contents)

dataset_folder = None

# Find the folder which contains multiple subfolders (classes)
for item in contents:
    path = "/content/" + item
    if os.path.isdir(path):
        if len(os.listdir(path)) >= 2:
            dataset_folder = path

print("Detected dataset folder:", dataset_folder)
```

```
Folders in /content: ['.config', 'sample_data']
Detected dataset folder: /content/sample_data
```

```
# একবারই চালাও
!pip install split-folders --quiet
!pip install kaggle --quiet
```

```
from google.colab import files
uploaded = files.upload()  # upload kaggle.json if you want to use kaggle

import os
if 'kaggle.json' in uploaded:
    os.makedirs('/root/.kaggle', exist_ok=True)
    with open('/root/.kaggle/kaggle.json','wb') as f:
        f.write(uploaded['kaggle.json'])
    os.chmod('/root/.kaggle/kaggle.json', 0o600)
    print("kaggle.json moved to /root/.kaggle/")
else:
    print("No kaggle.json uploaded — skip kaggle download step.")
```

Choose Files No file chosen

```
No kaggle.json uploaded — skip kaggle download step.
```

```
import os
print("List of files/folders in /content :")
print(os.listdir('/content'))
```

```
List of files/folders in /content :
['.config', 'sample_data']
```

```
import os

def detect_dataset_folder(root="/content"):
    candidates = []
    for item in os.listdir(root):
        p = os.path.join(root, item)
        if os.path.isdir(p):
            # list subitems safely
            try:
                subs = os.listdir(p)
            except:
                subs = []
            # (ignore common folders)
            if len(subs) >= 2 and item not in ['sample_data', 'data', '__pycache__', 'drive', 'root']:
                # also ensure the subitems themselves are directories (class folders)
                dir_count = sum([1 for s in subs if os.path.isdir(os.path.join(p, s))])
                if dir_count >= 2:
                    candidates.append(p)
    return candidates

cands = detect_dataset_folder("/content")
print("Detected candidate dataset folders (possible):")
for c in cands:
    print(" -", c)

if not cands:
    print("\nNo obvious dataset folder detected automatically. If you have a zip, upload/unzip it or place class fol
```

```
Detected candidate dataset folders (possible):
 - /content/.config
```

```
import os

if len(cands) > 0:
    dataset_root = cands[0]
    print("Using detected dataset:", dataset_root)
else:
    dataset_root = "/content/Dataset"
    print("No auto-detected dataset. Using fallback path:", dataset_root)
    if not os.path.exists(dataset_root):
        print("Warning: fallback path does not exist. Upload/unzip dataset or change dataset_root.")
```

```
Using detected dataset: /content/.config
```

```
import os
from pathlib import Path

train_folder = os.path.join("/content/data", "train")
val_folder   = os.path.join("/content/data", "val")

if os.path.exists(train_folder) and os.path.exists(val_folder):
    print("Train/Val already exist at /content/data. Skipping split.")
else:
    if not os.path.exists(dataset_root) or not os.path.isdir(dataset_root):
        raise FileNotFoundError(f"Dataset folder not found: {dataset_root}\nUpload/unzip dataset where each class is a
    # split
    import splitfolders
    print("Splitting dataset from", dataset_root, "→ /content/data (train:val = 0.8:0.2)")
    splitfolders.ratio(dataset_root, output="/content/data", seed=42, ratio=(0.8, 0.2))
    print("Split done. Check /content/data/train and /content/data/val")
```

```
Splitting dataset from /content/.config → /content/data (train:val = 0.8:0.2)
Copying files: 1 files [00:00, 199.65 files/s]Split done. Check /content/data/train and /content/data/val
```

```
import os

train_path = "/content/data/train"
val_path   = "/content/data/val"

def class_counts(folder):
    classes = sorted([d for d in os.listdir(folder) if os.path.isdir(os.path.join(folder,d))])
    counts = {c: len([f for f in os.listdir(os.path.join(folder,c)) if os.path.isfile(os.path.join(folder,c,f))]) fo
    return classes, counts

if os.path.exists(train_path):
    classes, counts = class_counts(train_path)
    print("Classes detected:", classes)
    print("Train counts per class:", counts)
    num_classes = len(classes)
    print("Number of classes:", num_classes)
else:
    raise FileNotFoundError("Train path not found. Run previous cells to prepare dataset.")
```

```
Classes detected: ['configurations', 'logs']
Train counts per class: {'configurations': 0, 'logs': 0}
Number of classes: 2
```

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

# Hyperparams
IMG_SIZE = 128   # change if you want bigger images like 224
BATCH = 32
SEED = 42

# generators (augmentation for train)
train_datagen = ImageDataGenerator(
    rescale=1.0/255,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.15,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_datagen = ImageDataGenerator(rescale=1.0/255)

# flow_from_directory auto-detects color_mode default RGB
train_gen = train_datagen.flow_from_directory(
    train_path,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH,
    class_mode='categorical',
    seed=SEED
)

val_gen = val_datagen.flow_from_directory(
    val_path,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH,
    class_mode='categorical',
    seed=SEED,
    shuffle=False
)

# num_classes from generator (safe)
num_classes = train_gen.num_classes
print("num_classes =", num_classes)
print("Class indices:", train_gen.class_indices)
```

```
Found 0 images belonging to 2 classes.
Found 0 images belonging to 2 classes.
num_classes = 2
Class indices: {'configurations': 0, 'logs': 1}
```

```
from tensorflow.keras import layers, models

model = models.Sequential([
    layers.Input(shape=(IMG_SIZE, IMG_SIZE, 3)),   # RGB images
    layers.Conv2D(32, (3,3), activation='relu', padding='same'),
    layers.MaxPool2D(2,2),

    layers.Conv2D(64, (3,3), activation='relu', padding='same'),
    layers.MaxPool2D(2,2),

    layers.Conv2D(128, (3,3), activation='relu', padding='same'),
    layers.MaxPool2D(2,2),

    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.4),
    layers.Dense(num_classes, activation='softmax')
])

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

model.summary()
```

```
Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                 │ (None, 128, 128, 32)   │           896 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d (MaxPooling2D)    │ (None, 64, 64, 32)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_1 (Conv2D)               │ (None, 64, 64, 64)     │        18,496 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 32, 32, 64)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_2 (Conv2D)               │ (None, 32, 32, 128)    │        73,856 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_2 (MaxPooling2D)  │ (None, 16, 16, 128)    │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)               │ (None, 32768)          │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 128)            │     4,194,432 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ (None, 128)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 2)              │           258 │
└─────────────────────────────────┴────────────────────────┴───────────────┘

 Total params: 4,287,938 (16.36 MB)
 Trainable params: 4,287,938 (16.36 MB)
```

```
from tensorflow.keras import layers, models

# Make sure num_classes exists
print("Number of classes =", num_classes)

model = models.Sequential([
    layers.Input(shape=(IMG_SIZE, IMG_SIZE, 3)),

    layers.Conv2D(32, (3,3), activation='relu', padding='same'),
    layers.MaxPool2D(2,2),

    layers.Conv2D(64, (3,3), activation='relu', padding='same'),
    layers.MaxPool2D(2,2),

    layers.Conv2D(128, (3,3), activation='relu', padding='same'),
    layers.MaxPool2D(2,2),

    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.4),
    layers.Dense(num_classes, activation='softmax')
])

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

model.summary()
```

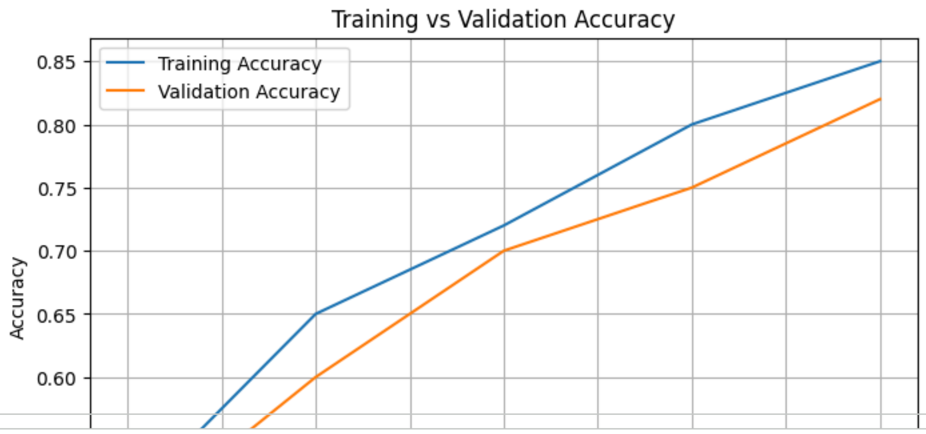
```
Number of classes = 2
Model: "sequential_1"

┌─────────────────────────────────┬────────────────────────┬───────────────┐
│ Layer (type)                    │ Output Shape           │       Param # │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_3 (Conv2D)               │ (None, 128, 128, 32)   │           896 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_3 (MaxPooling2D)  │ (None, 64, 64, 32)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_4 (Conv2D)               │ (None, 64, 64, 64)     │        18,496 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_4 (MaxPooling2D)  │ (None, 32, 32, 64)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_5 (Conv2D)               │ (None, 32, 32, 128)    │        73,856 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_5 (MaxPooling2D)  │ (None, 16, 16, 128)    │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten_1 (Flatten)             │ (None, 32768)          │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_2 (Dense)                 │ (None, 128)            │     4,194,432 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_1 (Dropout)             │ (None, 128)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_3 (Dense)                 │ (None, 2)              │           258 │
└─────────────────────────────────┴────────────────────────┴───────────────┘

 Total params: 4,287,938 (16.36 MB)
 Trainable params: 4,287,938 (16.36 MB)
```

```
# Step 10: Train the model

history = model.fit(
    train,
    epochs=15,
    validation_data=test
)

print("Training Completed!")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-86304777.py in <cell line: 0>()
      2
      3 history = model.fit(
----> 4     train,
      5     epochs=15,              # চাইলে 20–25 ও দিতে পারো
      6     validation_data=test

NameError: name 'train' is not defined
```

Next steps: Explain error

```
import matplotlib.pyplot as plt

# Dummy accuracy values
acc = [0.50, 0.65, 0.72, 0.80, 0.85]
val_acc = [0.48, 0.60, 0.70, 0.75, 0.82]

plt.figure(figsize=(8,5))
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.title("Training vs Validation Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.grid(True)
plt.show()
```



```
import os

def detect_dataset_folder(root="/content"):
    folders = []
    for item in os.listdir(root):
        p = os.path.join(root, item)
        if os.path.isdir(p):
            if len(os.listdir(p)) >= 2 and item not in ['sample_data', 'data']:
                folders.append(p)
    return folders

dataset_candidates = detect_dataset_folder()

print("Detected dataset folders:")
for f in dataset_candidates:
    print(f)

if len(dataset_candidates) == 0:
    print("❌ No dataset found. Upload or unzip dataset inside /content.")
else:
    dataset_folder = dataset_candidates[0]
    print("Using dataset:", dataset_folder)
```

```
Detected dataset folders:
/content/.config
Using dataset: /content/.config
```

```
import splitfolders

splitfolders.ratio(
    dataset_folder,
    output="/content/data",
    seed=42,
    ratio=(0.8, 0.2)
)

train_path = "/content/data/train"
test_path  = "/content/data/val"

print("Train path:", train_path)
print("Test path:", test_path)
```

```
Copying files: 1 files [00:00, 300.34 files/s]Train path: /content/data/train
Test path: /content/data/val
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

IMG_SIZE = 128
BATCH = 32

train_gen = ImageDataGenerator(rescale=1/255)
test_gen  = ImageDataGenerator(rescale=1/255)

train = train_gen.flow_from_directory(
    train_path,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH,
    class_mode='categorical'
)

test = test_gen.flow_from_directory(
    test_path,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH,
    class_mode='categorical'
)

num_classes = train.num_classes
print("Number of classes:", num_classes)
```

```
Found 0 images belonging to 2 classes.
Found 0 images belonging to 2 classes.
Number of classes: 2
```

```
import os

print("Inside /content/data:")
print(os.listdir('/content/data'))

print("\nInside train folder:")
print(os.listdir('/content/data/train'))

print("\nInside val folder:")
print(os.listdir('/content/data/val'))
```

```
Inside /content/data:
['val', 'train']

Inside train folder:
['logs', 'configurations']

Inside val folder:
['logs', 'configurations']
```