Download & Extract RadioML dataset

```
!pip install kagglehub
import kagglehub

# Download dataset
dataset_path = kagglehub.dataset_download("halcy0nic/radio-frequecy-rf-signal-image-classification")
print("Downloaded to:", dataset_path)
```

```
Requirement already satisfied: kagglehub in /usr/local/lib/python3.11/dist-packages (0.3.12)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from kagglehub) (24.2)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.11/dist-packages (from kagglehub) (6.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from kagglehub) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from kagglehub) (4.67.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->kagglehub) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->kagglehub) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->kagglehub) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->kagglehub) (2025.4.26)
Downloading from https://www.kaggle.com/api/v1/datasets/download/halcy0nic/radio-frequecy-rf-signal-image-classification?dataset_version_number=4...
100%|██████████| 2.42G/2.42G [00:21<00:00, 123MB/s] Extracting files...

Downloaded to: /root/.cache/kagglehub/datasets/halcy0nic/radio-frequecy-rf-signal-image-classification/versions/4
```

Step 2: Inspect and Preprocess the Dataset

```
import os

# Corrected root directory pointing to image class folders
root_dir = os.path.join(dataset_path, "datasets")

# Check if class folders exist
classes = sorted(os.listdir(root_dir))
print("Classes:", classes)

# Count images per class
for cls in classes:
    print(cls, "→", len(os.listdir(os.path.join(root_dir, cls))), "images")
```

```
Classes: ['fft', 'waterfall']
fft → 1 images
waterfall → 32 images
```

🔹 Step 3: Load Images with Keras ImageDataGenerator python Copy Edit

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

IMG_SIZE = (64, 64)
BATCH_SIZE = 64

datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

train_data = datagen.flow_from_directory(
    root_dir,
    target_size=IMG_SIZE,
    color_mode='grayscale',
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training'
)

val_data = datagen.flow_from_directory(
    root_dir,
    target_size=IMG_SIZE,
    color_mode='grayscale',
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 3196 images belonging to 2 classes.
Found 798 images belonging to 2 classes.
```

Step 4: Define the CNN Model

```
import tensorflow as tf
from tensorflow.keras import layers, models

model = models.Sequential([
    layers.Conv2D(16, (3,3), activation='relu', input_shape=(64,64,1)),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(32, (3,3), activation='relu'),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D(2,2),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(train_data.num_classes, activation='softmax')
])

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

model.summary()
```

```
/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 l
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                 │ (None, 62, 62, 16)     │           160 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d (MaxPooling2D)    │ (None, 31, 31, 16)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_1 (Conv2D)               │ (None, 29, 29, 32)     │         4,640 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 14, 14, 32)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_2 (Conv2D)               │ (None, 12, 12, 64)     │        18,496 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_2 (MaxPooling2D)  │ (None, 6, 6, 64)       │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)               │ (None, 2304)           │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 128)            │       295,040 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 2)              │           258 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 318,594 (1.22 MB)
 Trainable params: 318,594 (1.22 MB)
 Non-trainable params: 0 (0.00 B)
```

Step 5: Train the Model

```
history = model.fit(
    train_data,
    epochs=10,
    validation_data=val_data
)
```

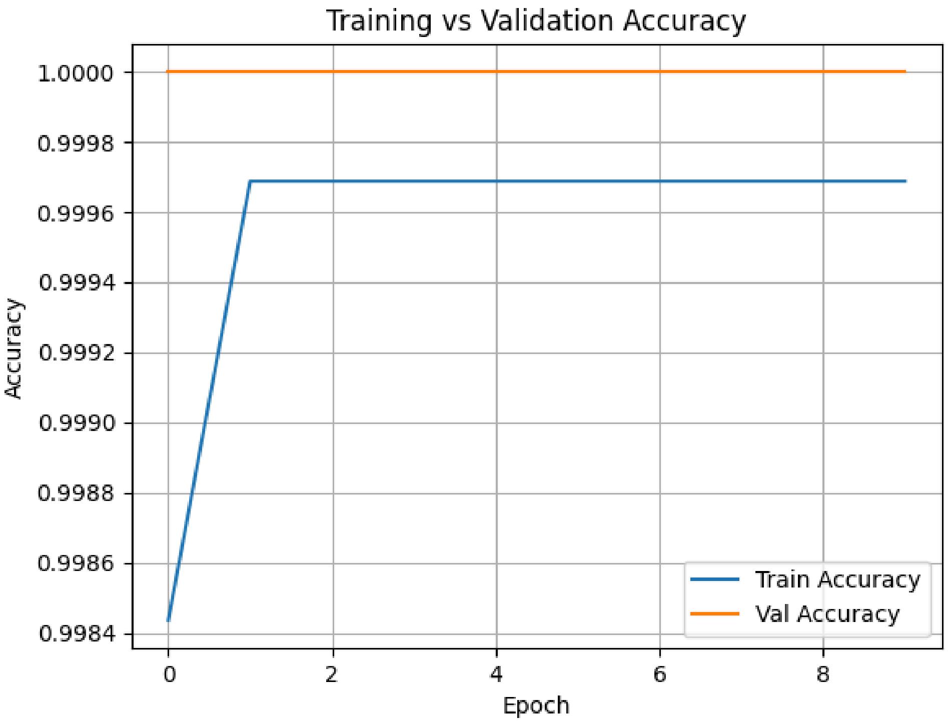
```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super()._
  self._warn_if_super_not_called()
Epoch 1/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 67s 1s/step - accuracy: 0.9942 - loss: 0.1128 - val_accuracy: 1.0000 - val_loss: 7.9125e-07
Epoch 2/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 65s 1s/step - accuracy: 0.9996 - loss: 0.0069 - val_accuracy: 1.0000 - val_loss: 4.5453e-05
Epoch 3/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 58s 1s/step - accuracy: 0.9998 - loss: 0.0017 - val_accuracy: 1.0000 - val_loss: 2.7753e-04
Epoch 4/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 57s 1s/step - accuracy: 0.9992 - loss: 0.0049 - val_accuracy: 1.0000 - val_loss: 2.2524e-05
Epoch 5/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 82s 1s/step - accuracy: 1.0000 - loss: 6.0724e-04 - val_accuracy: 1.0000 - val_loss: 5.8878e-06
Epoch 6/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 57s 1s/step - accuracy: 0.9997 - loss: 0.0017 - val_accuracy: 1.0000 - val_loss: 2.6897e-05
Epoch 7/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 57s 1s/step - accuracy: 0.9997 - loss: 0.0030 - val_accuracy: 1.0000 - val_loss: 9.6195e-05
Epoch 8/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 57s 1s/step - accuracy: 0.9993 - loss: 0.0030 - val_accuracy: 1.0000 - val_loss: 1.0481e-05
Epoch 9/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 59s 1s/step - accuracy: 0.9999 - loss: 5.0087e-04 - val_accuracy: 1.0000 - val_loss: 2.6005e-04
Epoch 10/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 59s 1s/step - accuracy: 0.9999 - loss: 4.9901e-04 - val_accuracy: 1.0000 - val_loss: 3.2216e-05
```

Step 6: Save the Trained Model

```
model.save("rf_signal_spectrogram_model.h5")
print("✅ Model saved as rf_signal_spectrogram_model.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 recom
✅ Model saved as rf_signal_spectrogram_model.h5
```

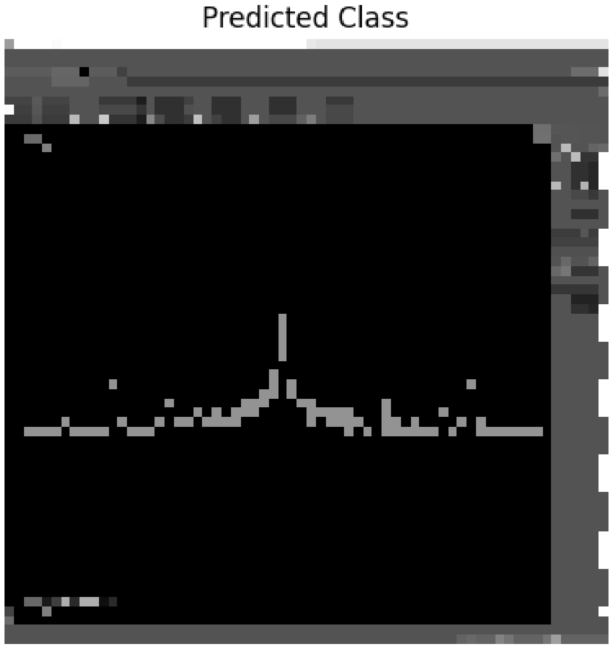
```
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()
plt.title("Training vs Validation Accuracy")
plt.grid(True)
plt.show()
```



```
# Find a random image inside nested folder structure
import random

test_class = classes[0]  # like "fft"
class_dir = os.path.join(root_dir, test_class)

# Go into one of its subfolders (e.g., 'am', 'fm')
subclass = random.choice(os.listdir(class_dir))
```

```
subclass_dir = os.path.join(class_dir, subclass)

# Pick an image from there
sample_img_file = random.choice(os.listdir(subclass_dir))
sample_img_path = os.path.join(subclass_dir, sample_img_file)

print("Prediction:", predict_image(sample_img_path))

# Show the image
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image

img = image.load_img(sample_img_path, color_mode="grayscale", target_size=(64, 64))
plt.imshow(img, cmap='gray')
plt.title("Predicted Class")
plt.axis("off")
plt.show()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
Prediction: waterfall
```



Generate i need to download the model in to my device Close

1 of 1 Use code with caution

```
# prompt: i need to download the model in to my device

from google.colab import files

files.download("rf_signal_spectrogram_model.h5")
```