Task for Today

Fish Image Species Classification

Given *images of fish*, let's try to predict the **species** of fish present in a given image.

We will use a TensorFlow/Keras pretrained CNN to make our predictions.

Getting Started

```
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).
```

```
import numpy as np
import pandas as pd
from pathlib import Path
import os.path

from sklearn.model_selection import train_test_split

import tensorflow as tf
import cv2
import matplotlib.pyplot as plt
import seaborn as sns

from tqdm import tqdm
```

```
image_dir = Path('/content/drive/MyDrive/fish classification dataset/Fish_Dataset')
```

Creating File DataFrame

```
# Get filepaths and labels
filepaths = list(image_dir.glob(r'**/*.png'))
labels = list(map(lambda x: os.path.split(os.path.split(x)[0])[1], filepaths))

filepaths = pd.Series(filepaths, name='Filepath').astype(str)
labels = pd.Series(labels, name='Label')

# Concatenate filepaths and labels
image_df = pd.concat([filepaths, labels], axis=1)

# Drop GT images
image_df['Label'] = image_df['Label'].apply(lambda x: np.NaN if x[-2:] == 'GT' else x)
image_df = image_df.dropna(axis=0)

# Sample 200 images from each class
samples = []

for category in image_df['Label'].unique():
    category_slice = image_df.query("Label == @category")
    samples.append(category_slice.sample(20, random_state=1))

image_df = pd.concat(samples, axis=0).sample(frac=1.0, random_state=1).reset_index(drop=True)
```

```
image_df
```

	Filepath	Label
0	/content/drive/MyDrive/fish classification dat...	Sea Bass
1	/content/drive/MyDrive/fish classification dat...	Sea Bass
2	/content/drive/MyDrive/fish classification dat...	Red Mullet
3	/content/drive/MyDrive/fish classification dat...	Black Sea Sprat
4	/content/drive/MyDrive/fish classification dat...	Shrimp

```
train_df, test_df = train_test_split(image_df, train_size=0.7, shuffle=True, random_state=1)
```

Loading the Images

[illegible]	/content/drive/MyDrive/fish classification dat...	Hourse Mackerel

```
train_generator = tf.keras.preprocessing.image.ImageDataGenerator(
    preprocessing_function=tf.keras.applications.mobilenet_v2.preprocess_input,
    validation_split=0.2
)

test_generator = tf.keras.preprocessing.image.ImageDataGenerator(
    preprocessing_function=tf.keras.applications.mobilenet_v2.preprocess_input
)


train_images = train_generator.flow_from_dataframe(
    dataframe=train_df,
    x_col='Filepath',
    y_col='Label',
    target_size=(224, 224),
    color_mode='rgb',
    class_mode='categorical',
    batch_size=32,
    shuffle=True,
    seed=42,
    subset='training'
)

val_images = train_generator.flow_from_dataframe(
    dataframe=train_df,
    x_col='Filepath',
    y_col='Label',
    target_size=(224, 224),
    color_mode='rgb',
    class_mode='categorical',
    batch_size=32,
    shuffle=True,
    seed=42,
    subset='validation'
)

test_images = test_generator.flow_from_dataframe(
    dataframe=test_df,
    x_col='Filepath',
    y_col='Label',
    target_size=(224, 224),
    color_mode='rgb',
    class_mode='categorical',
    batch_size=32,
    shuffle=False
)
```

```
Found 100 validated image filenames belonging to 9 classes.
Found 25 validated image filenames belonging to 9 classes.
Found 55 validated image filenames belonging to 9 classes.
```

Load Pretrained Model

```
pretrained_model = tf.keras.applications.MobileNetV2(
    input_shape=(224, 224, 3),
    include_top=False,
    weights='imagenet',
    pooling='avg'
)

pretrained_model.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering
9406464/9406464 [==============================] - 0s 0us/step
```

Training

```
inputs = pretrained_model.input

x = tf.keras.layers.Dense(128, activation='relu')(pretrained_model.output)
x = tf.keras.layers.Dense(128, activation='relu')(x)

outputs = tf.keras.layers.Dense(9, activation='softmax')(x)


model = tf.keras.Model(inputs=inputs, outputs=outputs)


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


history = model.fit(
    train_images,
    validation_data=val_images,
    epochs=5,
    callbacks=[
        tf.keras.callbacks.EarlyStopping(
            monitor='val_loss',
            patience=3,
            restore_best_weights=True
        )
    ]
)
```

```
Epoch 1/5
4/4 [==============================] - 11s 2s/step - loss: 1.9501 - accuracy: 0.3800 - val_loss: 1.4451 - val_accuracy: 0.6400
Epoch 2/5
4/4 [==============================] - 6s 2s/step - loss: 0.8921 - accuracy: 0.8600 - val_loss: 0.9302 - val_accuracy: 0.7600
Epoch 3/5
4/4 [==============================] - 6s 1s/step - loss: 0.3994 - accuracy: 0.9800 - val_loss: 0.2831 - val_accuracy: 1.0000
Epoch 4/5
4/4 [==============================] - 7s 2s/step - loss: 0.1337 - accuracy: 1.0000 - val_loss: 0.2332 - val_accuracy: 0.9600
Epoch 5/5
4/4 [==============================] - 5s 1s/step - loss: 0.0628 - accuracy: 1.0000 - val_loss: 0.0860 - val_accuracy: 0.9600
```

Results

```
results = model.evaluate(test_images, verbose=0)

print("    Test Loss: {:.5f}".format(results[0]))
print("Test Accuracy: {:.2f}%".format(results[1] * 100))
```

```
    Test Loss: 0.06870
Test Accuracy: 98.18%
```

Data Every Day

This notebook is featured on Data Every Day, a YouTube series where I train models on a new dataset each day.

Check it out!
https://youtu.be/E_3-9sGq7jk