Untitled1

November 17, 2023

[1]:

```python
import os
import shutil
import random

# Set the path to your dataset containing different folders
dataset_path = "dataset2"

# Set the ratio for the validation set (e.g., 0.2 for 20% validation)
validation_ratio = 0.2

# Set the path where you want to create the train and validation directories
output_path = "BI681"

# Create the output directory if it doesn't exist
os.makedirs(output_path, exist_ok=True)

# List all folders in the dataset directory
folders = [f for f in os.listdir(dataset_path) if os.path.isdir(os.path.
 ↪join(dataset_path, f))]

# Create train and validation subdirectories
train_dir = os.path.join(output_path, "train")
val_dir = os.path.join(output_path, "val")
os.makedirs(train_dir, exist_ok=True)
os.makedirs(val_dir, exist_ok=True)

# Iterate through each folder and divide the data
for folder in folders:
    folder_path = os.path.join(dataset_path, folder)
    files = [f for f in os.listdir(folder_path) if os.path.isfile(os.path.
 ↪join(folder_path, f))]
    random.shuffle(files)

    # Calculate the split point based on the validation ratio
    split_point = int(len(files) * validation_ratio)

    # Split files into train and validation sets
    train_files = files[split_point:]
    val_files = files[:split_point]

    # Copy files to train directory
    for file in train_files:
        src_path = os.path.join(folder_path, file)
        dest_path = os.path.join(train_dir, folder, file)
        os.makedirs(os.path.dirname(dest_path), exist_ok=True)
        shutil.copy(src_path, dest_path)

    # Copy files to validation directory
    for file in val_files:
        src_path = os.path.join(folder_path, file)
        dest_path = os.path.join(val_dir, folder, file)
        os.makedirs(os.path.dirname(dest_path), exist_ok=True)
        shutil.copy(src_path, dest_path)

print("Dataset division into train and validation sets is complete.")
```

```
Dataset division into train and validation sets is complete.
```

[2]:

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
```

[3]:

```
train_data_dir = 'BI681/train'
test_data_dir = 'BI681/val'
```

[4]:

```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)

# Create generators for training and testing datasets
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary'
)

test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary'
)
```

```
Found 8 images belonging to 2 classes.
Found 2 images belonging to 2 classes.
```

[5]:
```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150,␣
 ↪3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

[6]:
```
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

[7]:
```
model.fit(train_generator, epochs=10, validation_data=test_generator)
```

```
Epoch 1/10
1/1 [==============================] - 5s 5s/step - loss: 0.6944 - accuracy:
0.5000 - val_loss: 0.9447 - val_accuracy: 0.5000
Epoch 2/10
1/1 [==============================] - 1s 794ms/step - loss: 0.6673 - accuracy:
0.5000 - val_loss: 2.5991 - val_accuracy: 0.5000
Epoch 3/10
1/1 [==============================] - 1s 811ms/step - loss: 4.5043 - accuracy:
0.5000 - val_loss: 0.5353 - val_accuracy: 0.5000
Epoch 4/10
1/1 [==============================] - 1s 798ms/step - loss: 0.9886 - accuracy:
0.5000 - val_loss: 0.7945 - val_accuracy: 0.5000
Epoch 5/10
1/1 [==============================] - 1s 782ms/step - loss: 0.6067 - accuracy:
0.5000 - val_loss: 0.7481 - val_accuracy: 0.5000
Epoch 6/10
1/1 [==============================] - 1s 789ms/step - loss: 0.5287 - accuracy:
0.7500 - val_loss: 0.5577 - val_accuracy: 1.0000
Epoch 7/10
1/1 [==============================] - 1s 804ms/step - loss: 0.3584 - accuracy:
1.0000 - val_loss: 0.4869 - val_accuracy: 1.0000
Epoch 8/10
1/1 [==============================] - 1s 729ms/step - loss: 0.4555 - accuracy:
0.8750 - val_loss: 0.5286 - val_accuracy: 0.5000
```

```
Epoch 9/10
1/1 [==============================] - 1s 758ms/step - loss: 0.2058 - accuracy:
1.0000 - val_loss: 0.7360 - val_accuracy: 0.5000
Epoch 10/10
1/1 [==============================] - 1s 789ms/step - loss: 0.2955 - accuracy:
0.7500 - val_loss: 1.4344 - val_accuracy: 0.5000
```

```
[7]: <keras.src.callbacks.History at 0x1c7ebe4fdf0>
```

```
[8]: test_loss, test_acc = model.evaluate(test_generator)
     print(f'Test accuracy: {test_acc}')
```

```
1/1 [==============================] - 0s 170ms/step - loss: 1.4344 - accuracy:
0.5000
Test accuracy: 0.5
```

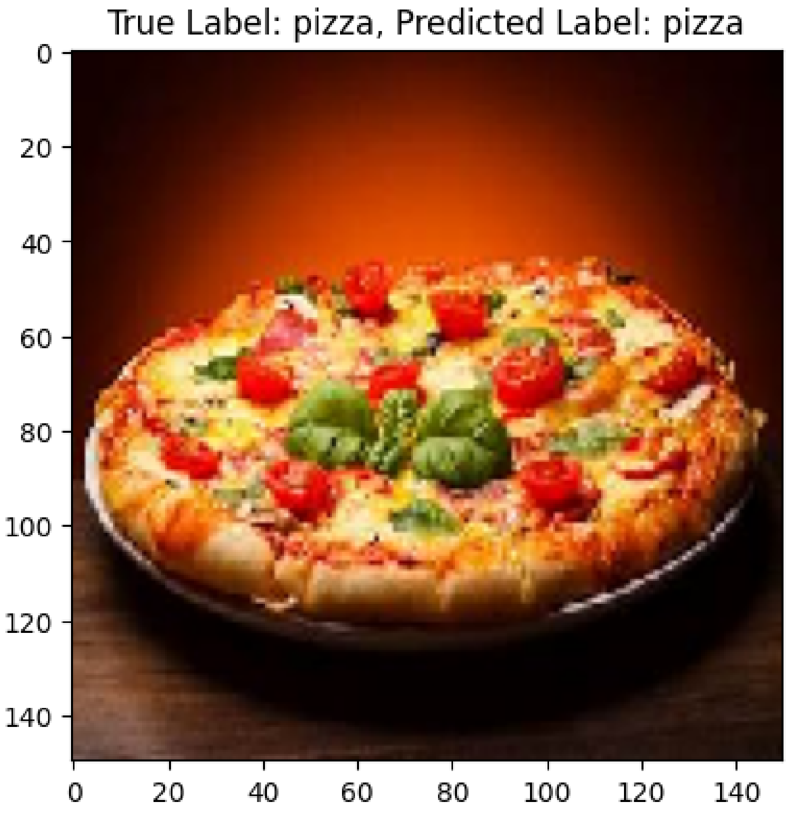
```
[10]: sample_images, sample_labels = next(test_generator)
      predictions = model.predict(sample_images)
```

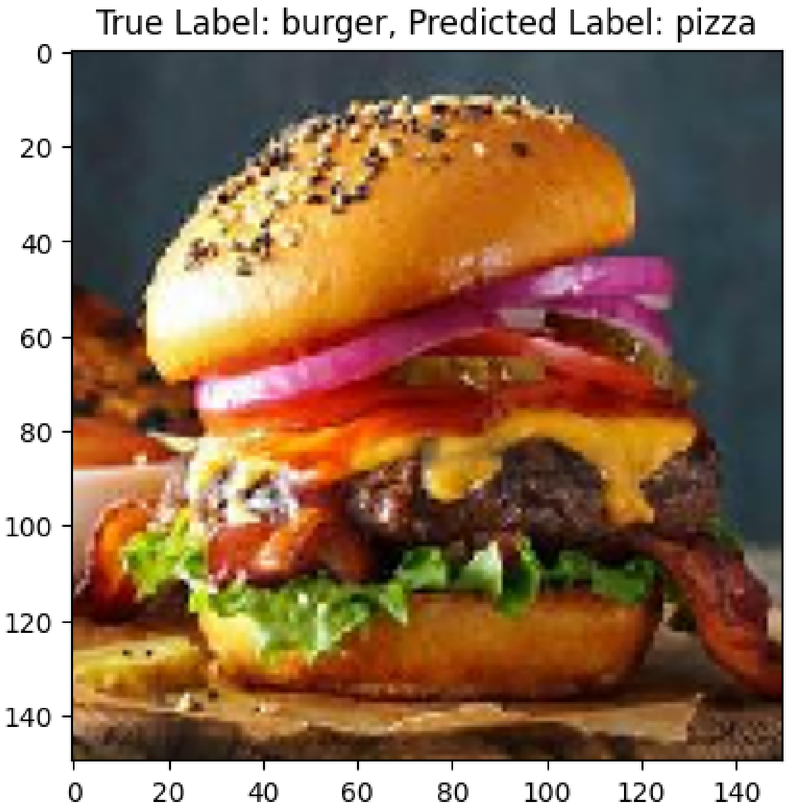
```
1/1 [==============================] - 0s 335ms/step
```

```
[11]: predicted_labels = [1 if p > 0.5 else 0 for p in predictions]
```

```
[14]: for i in range(2):
          true_label = "burger" if sample_labels[i] == 0 else "pizza"
          predicted_label = "burger" if predicted_labels[i] == 0 else "pizza"

          # Display the image
          plt.imshow(sample_images[i])
          plt.title(f"True Label: {true_label}, Predicted Label: {predicted_label}")
          plt.show()
```


True Label: pizza, Predicted Label: pizza



True Label: burger, Predicted Label: pizza

[ ]: