```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

```
model_path = '/content/drive/MyDrive/vggbest_model.keras'
model = load_model(model_path)
print("Model loaded successfully.")
```

```
Model loaded successfully.
```

```
import os
import numpy as np
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
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

```
train_dir = '/content/drive/MyDrive/split_minip/train'
val_dir = '/content/drive/MyDrive/split_minip/val'
test_dir = '/content/drive/MyDrive/split_minip/test'

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

train_data = train_gen.flow_from_directory(train_dir, target_size=(224, 224), batch_size=32,
val_data = val_gen.flow_from_directory(val_dir, target_size=(224, 224), batch_size=32, class
test_data = test_gen.flow_from_directory(test_dir, target_size=(224, 224), batch_size=32, cl
```

```
Found 5600 images belonging to 8 classes.
Found 1201 images belonging to 8 classes.
Found 1162 images belonging to 8 classes.
```

```
import os
def count_images_in_directory(directory):
    return sum(len(files) for _, _, files in os.walk(directory))
train_dir = '/content/drive/MyDrive/split_minip/train'
val_dir = '/content/drive/MyDrive/split_minip/val'
test_dir = '/content/drive/MyDrive/split_minip/test'
demo_dir = '/content/drive/MyDrive/split_minip/demo'
print(f"Number of training images: {count_images_in_directory(train_dir)}")
print(f"Number of validation images: {count_images_in_directory(val_dir)}")
print(f"Number of test images: {count_images_in_directory(test_dir)}")
print(f"Number of demo images: {count_images_in_directory(demo_dir)}")
```

```
Number of training images: 5600
Number of validation images: 1201
Number of test images: 1162
Number of demo images: 40
```

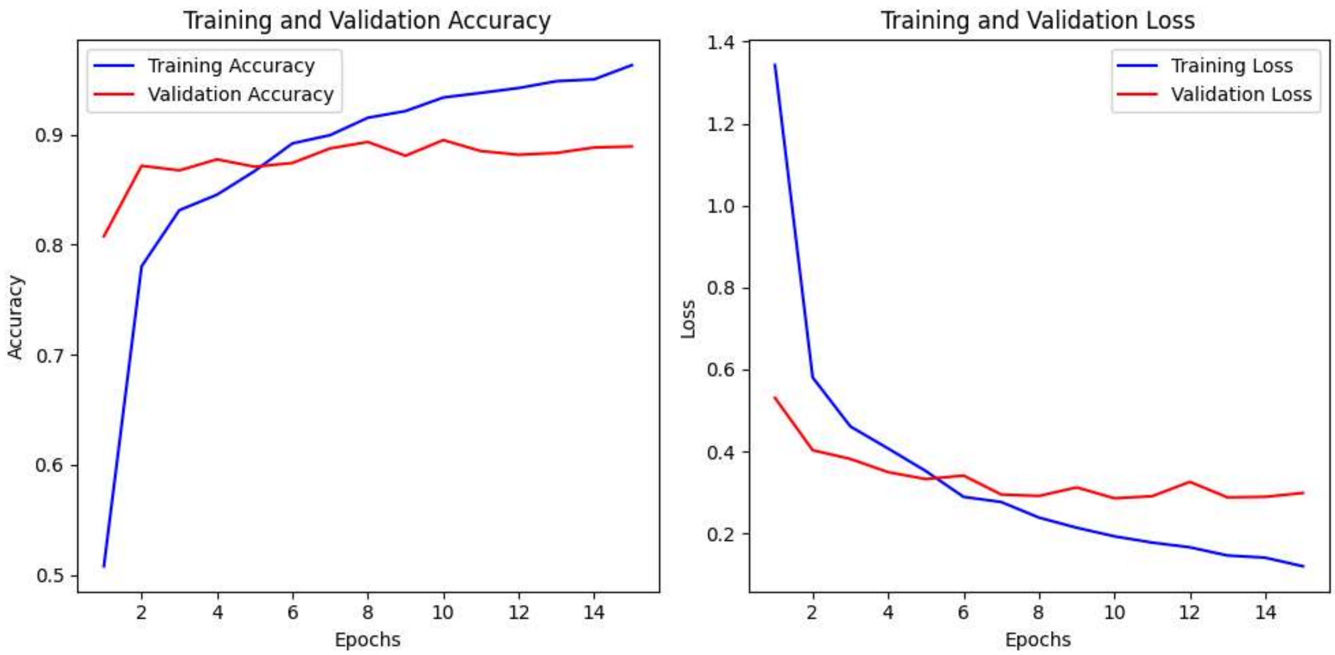
```
val_loss, val_accuracy = model.evaluate(val_data)
print(f'Validation Loss: {val_loss:4f}')
print(f'Validation Accuracy: {val_accuracy * 100}')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adap
  self._warn_if_super_not_called()
38/38 ━━━━━━━━━━━━━━━━━━━━ 819s 22s/step - accuracy: 0.8849 - loss: 0.3042
Validation Loss: 0.285749
Validation Accuracy: 89.50874209403992
```

```
import warnings
warnings.filterwarnings("ignore", category=UserWarning, module="keras.src.trainers.data_adap
test_data = test_gen.flow_from_directory(test_dir, target_size=(224, 224), batch_size=32, cl
test_loss, test_accuracy = model.evaluate(test_data)
print(f"Test Loss: {test_loss:.4f}")
print(f"Test Accuracy: {test_accuracy * 100:.4f}")
```

```
Found 1162 images belonging to 8 classes.
37/37 ━━━━━━━━━━━━━━━━━━━━ 511s 14s/step - accuracy: 0.8945 - loss: 0.2863
Test Loss: 0.2907
Test Accuracy: 88.7263
```

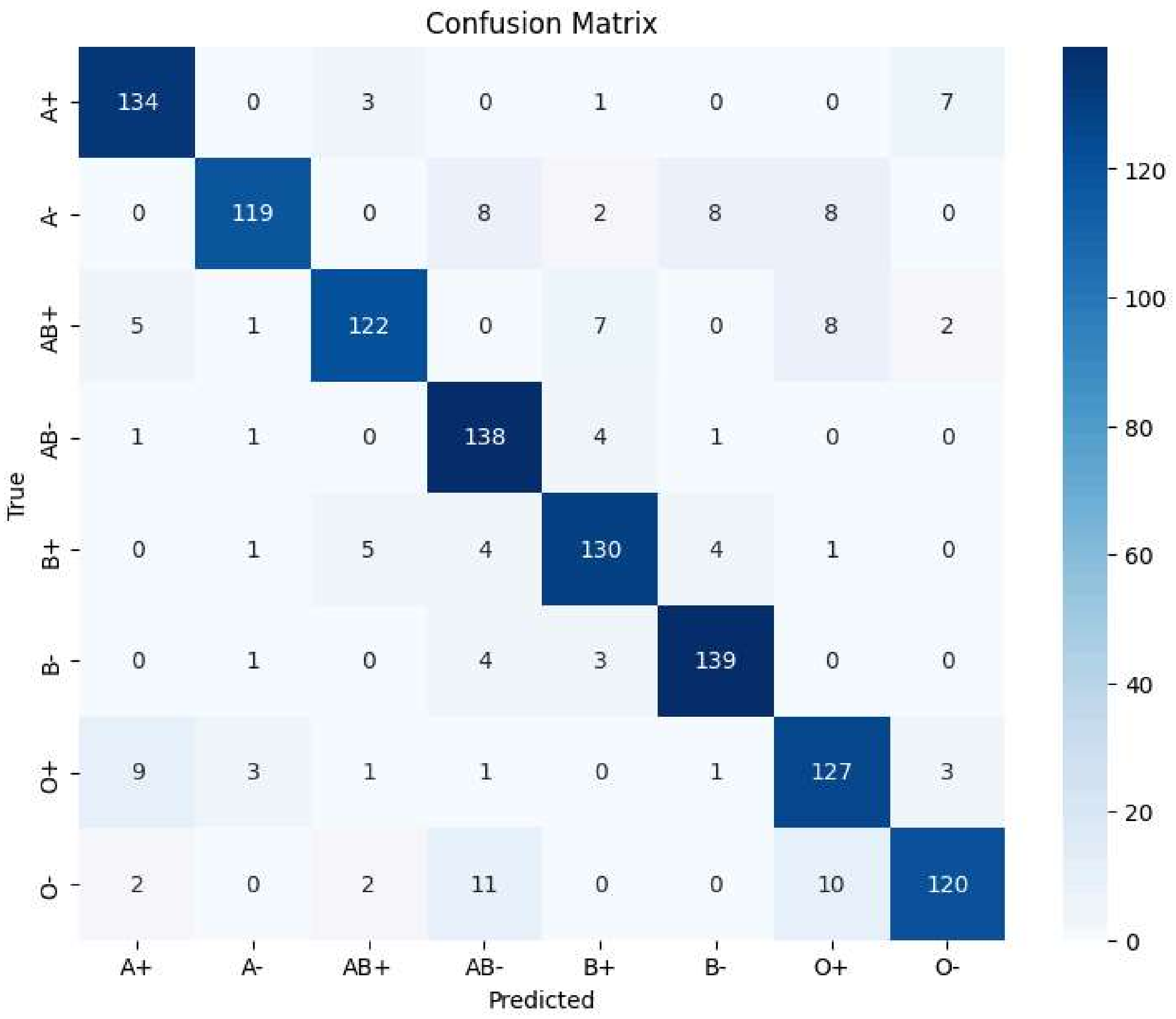
```
import matplotlib.pyplot as plt
epochs = range(1, 16)
train_accuracy = [0.5079, 0.7804, 0.8314, 0.8455, 0.8670, 0.8921, 0.8996, 0.9154, 0.9215, 0.
val_accuracy = [0.8077, 0.8718, 0.8676, 0.8776, 0.8709, 0.8743, 0.8876, 0.8934, 0.8809, 0.89
train_loss = [1.3424, 0.5804, 0.4609, 0.4072, 0.3521, 0.2892, 0.2766, 0.2387, 0.2139, 0.1936
val_loss = [0.5308, 0.4030, 0.3817, 0.3495, 0.3326, 0.3412, 0.2949, 0.2915, 0.3123, 0.2857,
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.plot(epochs, train_accuracy, label='Training Accuracy', color='b')
plt.plot(epochs, val_accuracy, label='Validation Accuracy', color='r')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(epochs, train_loss, label='Training Loss', color='b')
plt.plot(epochs, val_loss, label='Validation Loss', color='r')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```



```
import numpy as np
from sklearn.metrics import classification_report, accuracy_score
y_pred_prob = model.predict(test_data)
y_pred = np.argmax(y_pred_prob, axis=1)
y_true = test_data.classes
print(classification_report(y_true, y_pred))

accuracy = accuracy_score(y_true, y_pred)
print(f'Accuracy: {accuracy}')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adap
  self._warn_if_super_not_called()
37/37 ━━━━━━━━━━━━━━━━━━━━ 872s 24s/step
              precision    recall  f1-score   support

           0       0.89      0.92      0.91       145
           1       0.94      0.82      0.88       145
           2       0.92      0.84      0.88       145
           3       0.83      0.95      0.89       145
           4       0.88      0.90      0.89       145
           5       0.91      0.95      0.93       147
           6       0.82      0.88      0.85       145
           7       0.91      0.83      0.87       145

    accuracy                           0.89      1162
   macro avg       0.89      0.89      0.89      1162
weighted avg       0.89      0.89      0.89      1162

Accuracy: 0.8855421686746988
```

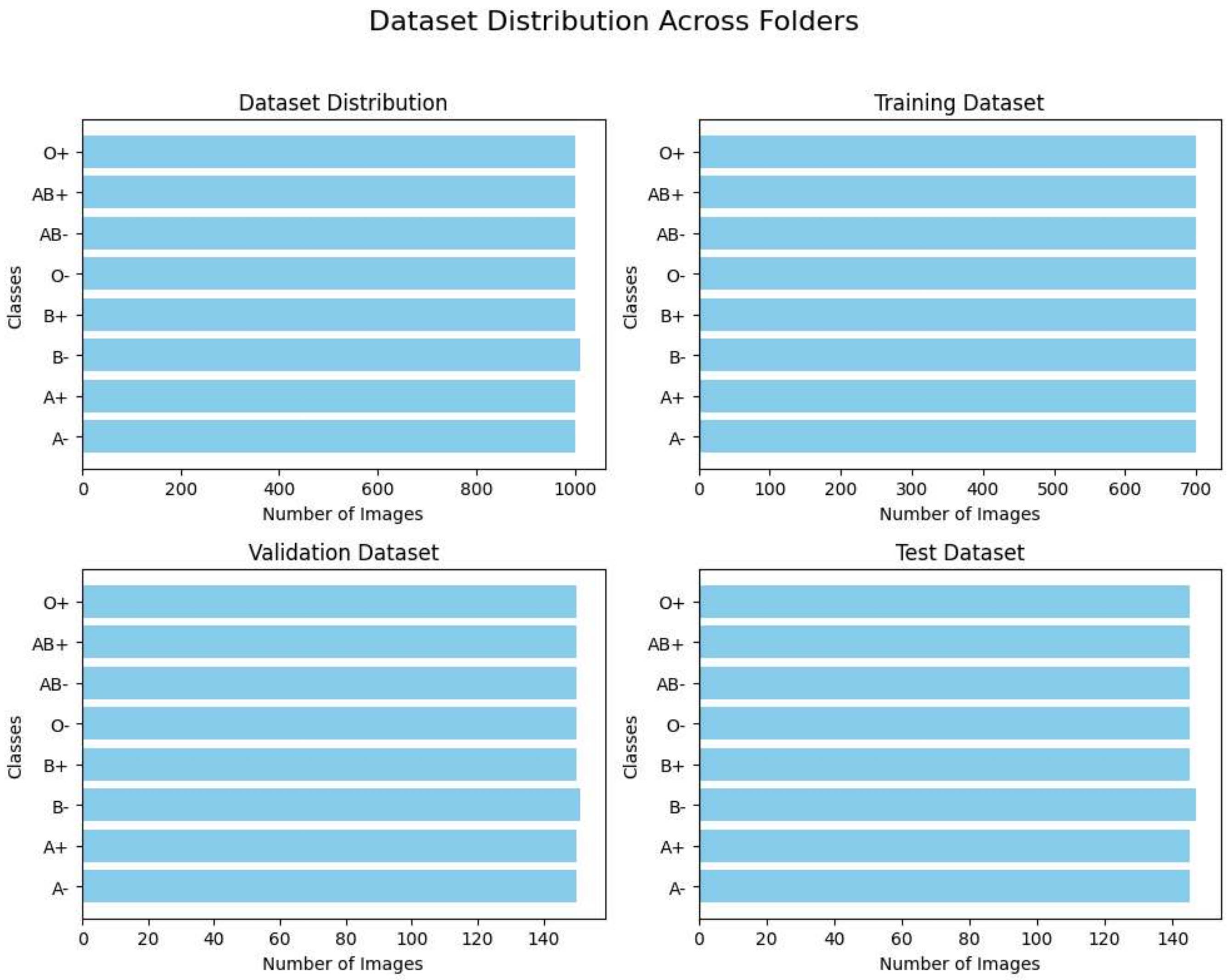
```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

cm = confusion_matrix(y_true, y_pred)

plt.figure(figsize=(9, 7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=test_data.class_indices.keys(
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```



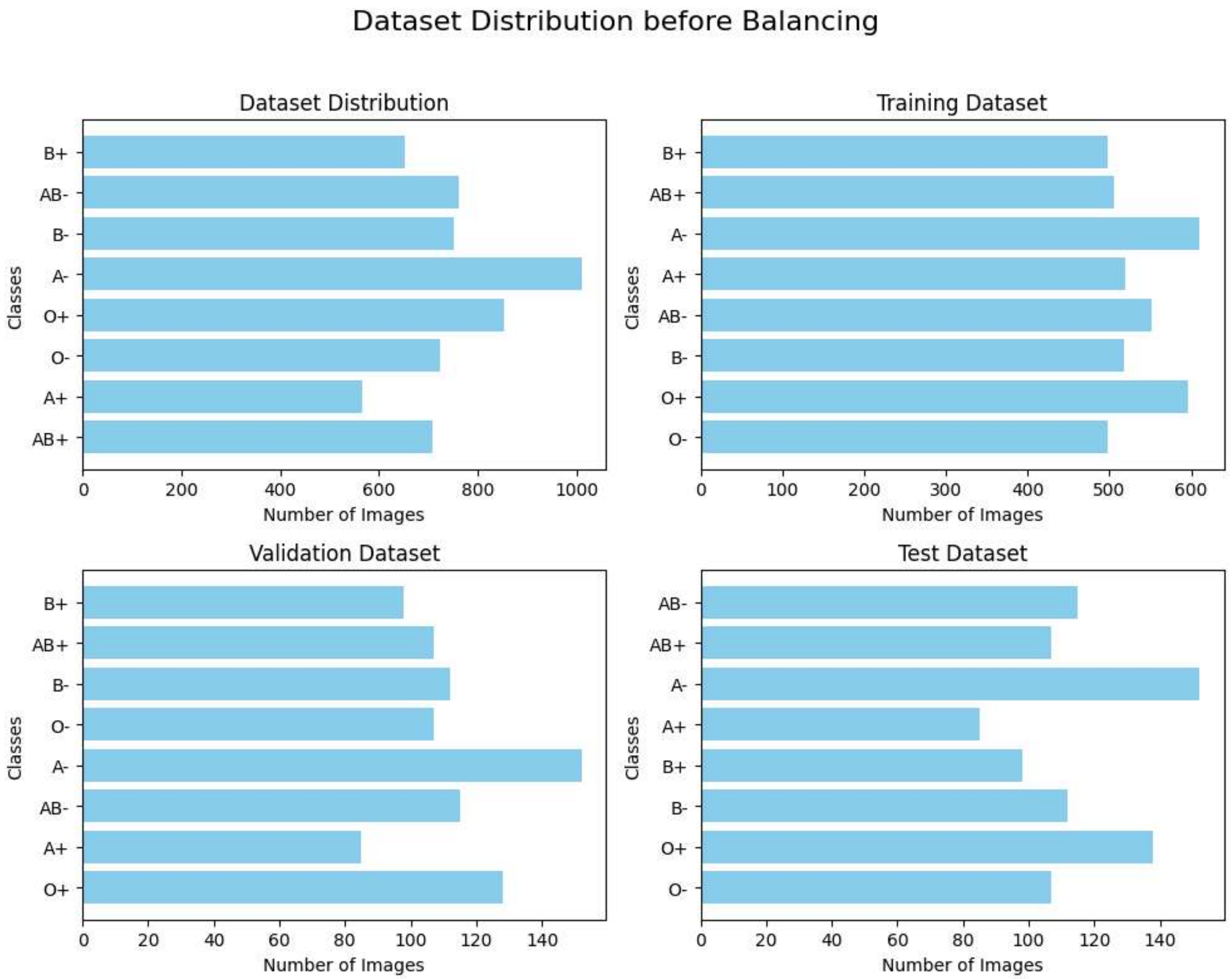
```
print("Class indices:", train_data.class_indices)
print("Classes:", train_data.classes)
print("Number of samples per class:", {k: list(train_data.classes).count(v) for k, v in trai
```

```
Class indices: {'A+': 0, 'A-': 1, 'AB+': 2, 'AB-': 3, 'B+': 4, 'B-': 5, 'O+': 6, 'O-': 7
Classes: [0 0 0 ... 7 7 7]
Number of samples per class: {'A+': 700, 'A-': 700, 'AB+': 700, 'AB-': 700, 'B+': 700, '
```

```
def count_images_in_folder(folder_path, extension=".bmp"):
    count = 0
    for root, _, files in os.walk(folder_path):
        count += len([file for file in files if file.lower().endswith(extension)])
    return count

dataset_folders = [
    '/content/drive/MyDrive/MiniP',
    '/content/drive/MyDrive/split_minip/train',
    '/content/drive/MyDrive/split_minip/val',
    '/content/drive/MyDrive/split_minip/test'
]
titles = [
    'Dataset Distribution',
    'Training Dataset',
    'Validation Dataset',
    'Test Dataset'
]
all_subfolder_names = []
all_image_counts = []

for dataset_folder in dataset_folders:
    subfolders = [os.path.join(dataset_folder, subfolder)
                  for subfolder in os.listdir(dataset_folder)
                  if os.path.isdir(os.path.join(dataset_folder, subfolder))]
    subfolder_names = []
    image_counts = []
    for subfolder in subfolders:
        subfolder_name = os.path.basename(subfolder)
        num_images = count_images_in_folder(subfolder, extension=".bmp")
        subfolder_names.append(subfolder_name)
        image_counts.append(num_images)
    all_subfolder_names.append(subfolder_names)
    all_image_counts.append(image_counts)

fig, axes = plt.subplots(2, 2, figsize=(10, 8))
fig.suptitle('Dataset Distribution Across Folders', fontsize=16)

for i, ax in enumerate(axes.flat):
    if i < len(dataset_folders):
        ax.barh(all_subfolder_names[i], all_image_counts[i], color='skyblue')
        ax.set_xlabel('Number of Images')
        ax.set_ylabel('Classes')
        ax.set_title(titles[i])
    else:
        ax.axis('off')
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```



```
def count_images_in_folder(folder_path, extension=".bmp"):
    count = 0
    for root, _, files in os.walk(folder_path):
        count += len([file for file in files if file.lower().endswith(extension)])
    return count

dataset_folders = [
    '/content/drive/MyDrive/dataset_blood_group(unbalanced)',
    '/content/drive/MyDrive/split_data/train',
    '/content/drive/MyDrive/split_data/val',
    '/content/drive/MyDrive/split_data/test',
]
titles = [
    'Dataset Distribution',
    'Training Dataset',
    'Validation Dataset',
    'Test Dataset'
]
all_subfolder_names = []
all_image_counts = []

for dataset_folder in dataset_folders:
    subfolders = [os.path.join(dataset_folder, subfolder)
                  for subfolder in os.listdir(dataset_folder)
                  if os.path.isdir(os.path.join(dataset_folder, subfolder))]
    subfolder_names = []
    image_counts = []
    for subfolder in subfolders:
        subfolder_name = os.path.basename(subfolder)
        num_images = count_images_in_folder(subfolder, extension=".bmp")
        subfolder_names.append(subfolder_name)
        image_counts.append(num_images)
    all_subfolder_names.append(subfolder_names)
    all_image_counts.append(image_counts)

fig, axes = plt.subplots(2, 2, figsize=(10, 8))
fig.suptitle('Dataset Distribution before Balancing', fontsize=16)

for i, ax in enumerate(axes.flat):
    if i < len(dataset_folders):
        ax.barh(all_subfolder_names[i], all_image_counts[i], color='skyblue')
        ax.set_xlabel('Number of Images')
        ax.set_ylabel('Classes')
        ax.set_title(titles[i])
    else:
        ax.axis('off')
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```



Dataset Distribution before Balancing

Dataset Distribution

Training Dataset

Validation Dataset

Test Dataset

```
Start coding or generate with AI.
```