Sheep_Face_Classification

April 28, 2024

[2]:
```
import numpy as np
import pandas as pd
import os
from PIL import Image
from keras.applications.vgg16 import VGG16, preprocess_input
from tensorflow.keras.applications import VGG19
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Model, load_model
from keras.layers import Dense, GlobalAveragePooling2D, Dropout,␣
↪BatchNormalization, Flatten, Conv2D, Activation, MaxPooling2D
from keras.callbacks import EarlyStopping, Callback, ModelCheckpoint
from sklearn.model_selection import train_test_split, KFold
from tensorflow.keras.utils import to_categorical
import tensorflow as tf
from tensorflow.keras.optimizers import SGD
from sklearn.utils import shuffle
from sklearn.model_selection import StratifiedKFold
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score
```

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[3]:
```
# Load images and labels
path = '/content/drive/MyDrive/Deep_learning_projects/SheepFaceImages'
sheep_breed_list = os.listdir(path)
print(sheep_breed_list)
```

```
['White Suffolk', 'Poll Dorset', 'Marino', 'Suffolk']
```

[4]:
```
def load_and_preprocess_images(path, target_size=(224, 224)):
    img_data_list = []
    labels = []
```

```
    for idx, sheep_breed in enumerate(sheep_breed_list):
        sheep_breed_files = os.listdir(os.path.join(path, sheep_breed))
        print(sheep_breed_files)
        for img_file in sheep_breed_files:
            sheep_image_path = os.path.join(path, sheep_breed, img_file)
            try:
                img = image.load_img(sheep_image_path, target_size=target_size)
                img_array = image.img_to_array(img)
                img_array = preprocess_input(img_array)
                img_data_list.append(img_array)
                labels.append(idx)
            except Exception as e:
                print(f"Error loading image {img_file}: {e}")

    # Convert lists to numpy arrays and preprocess
    img_data = np.array(img_data_list)
    labels = np.array(labels)

    # Shuffle data
    img_data, labels = shuffle(img_data, labels, random_state=777)

    return img_data, labels, sheep_breed_list

img_data_path = '/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/
 ↪sheep_img_data.npy'
labels_path = '/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/
 ↪sheep_labels.npy'

if os.path.exists(img_data_path) and os.path.exists(labels_path):
    img_data = np.load(img_data_path)
    labels = np.load(labels_path)
else:
    img_data, labels, sheep_breed_list = load_and_preprocess_images(path)
    # Save the processed data
    np.save(img_data_path, img_data)
    np.save(labels_path, labels)

print('Data Shape:', img_data.shape)
print('Labels Shape:', labels.shape)
```

```
Data Shape: (1680, 224, 224, 3)
Labels Shape: (1680,)
```

[5]:
```
# One-hot encode labels
labels_categorical = to_categorical(labels, num_classes=len(sheep_breed_list))

print(labels_categorical.shape)
```

(1680, 4)

[8]:
```
def build_model():
    base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224,␣
↪224, 3))

    # Freeze first 10 convolutional layers
    for layer in base_model.layers[:15]:
        layer.trainable = False

    x = base_model.output

    # Flatten to prepare for the fully connected layers
    x = Flatten()(x)

    # Fully connected + ReLU
    x = Dense(64, activation='relu')(x)
    x = Dense(64, activation='relu')(x)

    # Adding output layer
    predictions = Dense(len(sheep_breed_list), activation='softmax')(x)

    # Create the final model
    model = Model(inputs=base_model.input, outputs=predictions)

    # Custom learning rates
    lr_mult = {}

    # Set learning rate to 0 for the first 10 layers
    for layer in model.layers[:15]:
        lr_mult[layer.name + '/kernel:0'] = 0.0
        lr_mult[layer.name + '/bias:0'] = 0.0

    # Set learning rate to 0.0001 for the next Conv2D layers and the following␣
↪layers
    for layer in model.layers[15:-4]:
        if isinstance(layer, Conv2D):
            lr_mult[layer.name + '/kernel:0'] = 0.0001
            lr_mult[layer.name + '/bias:0'] = 0.0001

    # Set learning rate to 10 for the last two dense layers
    for layer in model.layers[-4:]:
        lr_mult[layer.name + '/kernel:0'] = 10
        lr_mult[layer.name + '/bias:0'] = 10

    # Use SGD optimizer with initial learning rate of 1e-4
    optimizer = SGD(learning_rate=1e-4,momentum=0.9)
```

```
    # Compile the model
    model.compile(optimizer=optimizer, loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'], loss_weights=[1.] + [10.]*2)

    return model

accuracy_values = []
precision_values = []
recall_values = []
f1_values = []

train_acc_list = []
val_acc_list = []
train_loss_list = []
val_loss_list = []

# KFold Cross-validation
n_splits = 5
kf = KFold(n_splits=n_splits, shuffle=True, random_state=777)
fold_var = 1

for train_index, test_index in kf.split(img_data):
    X_train, X_test = img_data[train_index], img_data[test_index]
    y_train, y_test = labels_categorical[train_index],␣
 ↪labels_categorical[test_index]

    model = build_model()

    #early_stop = EarlyStopping(monitor='val_loss', patience=3, verbose=1,␣
 ↪mode='min')
    save_path='/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/
 ↪sheepFace_one.h5'
    checkpoint =␣
 ↪ModelCheckpoint(save_path,monitor='val_loss',save_best_only=True,verbose=1)
    history = model.fit(X_train, y_train, batch_size=10, epochs=10, verbose=1,␣
 ↪validation_split=0.2, callbacks=[checkpoint])

    print("\n")
    print(f'Evaluating the Test metrics')
    model=load_model(save_path)
    # Evaluate the model
    scores = model.evaluate(X_test, y_test, verbose=1)
    print(f'Test loss for fold {fold_var}: {scores[0]}')
    print(f'Test accuracy for fold {fold_var}: {scores[1]}')
```

```python
    accuracy_values.append(scores[1])  # Appending accuracy to the list

    # Get predictions from the model
    y_true = np.argmax(y_test, axis=1)  # Convert one-hot encoded labels back to categorical labels
    y_pred = np.argmax(model.predict(X_test), axis=1)  # Get predictions from the model

    # Calculate and append evaluation metrics for this fold
    precision = precision_score(y_true, y_pred, average='weighted', zero_division=1)
    recall = recall_score(y_true, y_pred, average='weighted')
    f1 = f1_score(y_true, y_pred, average='weighted')

    precision_values.append(precision)
    recall_values.append(recall)
    f1_values.append(f1)

    # Print evaluation metrics for this fold
    print(f'Precision for fold {fold_var}: {precision}')
    print(f'Recall for fold {fold_var}: {recall}')
    print(f'F1 Score for fold {fold_var}: {f1}')
    print("\n")

    train_acc_list.append(history.history['accuracy'])
    val_acc_list.append(history.history['val_accuracy'])
    train_loss_list.append(history.history['loss'])
    val_loss_list.append(history.history['val_loss'])

    # Increment the fold number
    fold_var += 1

# Calculate standard deviation of accuracy across folds
accuracy_std_dev = np.std(accuracy_values)
print(f'Standard deviation of accuracy across folds: {accuracy_std_dev}')

# Calculate average of accuracy,precision, recall, and F1 across folds
avg_accuracy = np.mean(accuracy_values)
avg_precision = np.mean(precision_values)
avg_recall = np.mean(recall_values)
avg_f1 = np.mean(f1_values)

print(f'Average Accuracy across folds: {avg_accuracy}')
print(f'Average Precision across folds: {avg_precision}')
print(f'Average Recall across folds: {avg_recall}')
print(f'Average F1 Score across folds: {avg_f1}')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 3s 0us/step
Epoch 1/10
108/108 [==============================] - ETA: 0s - loss: 1.0960 - accuracy:
0.6521
Epoch 1: val_loss improved from inf to 0.55297, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5

/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103:
UserWarning: You are saving your model as an HDF5 file via `model.save()`. This
file format is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(

108/108 [==============================] - 31s 227ms/step - loss: 1.0960 -
accuracy: 0.6521 - val_loss: 0.5530 - val_accuracy: 0.7695
Epoch 2/10
107/108 [============================>.] - ETA: 0s - loss: 0.1838 - accuracy:
0.9383
Epoch 2: val_loss improved from 0.55297 to 0.21772, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 8s 75ms/step - loss: 0.1830 -
accuracy: 0.9386 - val_loss: 0.2177 - val_accuracy: 0.9368
Epoch 3/10
107/108 [============================>.] - ETA: 0s - loss: 0.0263 - accuracy:
0.9953
Epoch 3: val_loss improved from 0.21772 to 0.18921, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 8s 75ms/step - loss: 0.0265 -
accuracy: 0.9953 - val_loss: 0.1892 - val_accuracy: 0.9331
Epoch 4/10
107/108 [============================>.] - ETA: 0s - loss: 0.0057 - accuracy:
0.9991
Epoch 4: val_loss improved from 0.18921 to 0.16608, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 8s 76ms/step - loss: 0.0057 -
accuracy: 0.9991 - val_loss: 0.1661 - val_accuracy: 0.9517
Epoch 5/10
107/108 [============================>.] - ETA: 0s - loss: 0.0023 - accuracy:
1.0000
Epoch 5: val_loss did not improve from 0.16608
108/108 [==============================] - 8s 73ms/step - loss: 0.0023 -
accuracy: 1.0000 - val_loss: 0.1664 - val_accuracy: 0.9517
Epoch 6/10
107/108 [============================>.] - ETA: 0s - loss: 0.0015 - accuracy:
1.0000
Epoch 6: val_loss improved from 0.16608 to 0.16448, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
```

```
108/108 [==============================] - 9s 86ms/step - loss: 0.0015 -
accuracy: 1.0000 - val_loss: 0.1645 - val_accuracy: 0.9517
Epoch 7/10
107/108 [============================>.] - ETA: 0s - loss: 0.0012 - accuracy:
1.0000
Epoch 7: val_loss improved from 0.16448 to 0.16355, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 8s 76ms/step - loss: 0.0012 -
accuracy: 1.0000 - val_loss: 0.1636 - val_accuracy: 0.9517
Epoch 8/10
107/108 [============================>.] - ETA: 0s - loss: 9.6354e-04 -
accuracy: 1.0000
Epoch 8: val_loss improved from 0.16355 to 0.16288, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 8s 76ms/step - loss: 9.5954e-04 -
accuracy: 1.0000 - val_loss: 0.1629 - val_accuracy: 0.9517
Epoch 9/10
107/108 [============================>.] - ETA: 0s - loss: 8.0315e-04 -
accuracy: 1.0000
Epoch 9: val_loss improved from 0.16288 to 0.16151, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 8s 76ms/step - loss: 8.1136e-04 -
accuracy: 1.0000 - val_loss: 0.1615 - val_accuracy: 0.9517
Epoch 10/10
107/108 [============================>.] - ETA: 0s - loss: 7.0952e-04 -
accuracy: 1.0000
Epoch 10: val_loss did not improve from 0.16151
108/108 [==============================] - 8s 74ms/step - loss: 7.1443e-04 -
accuracy: 1.0000 - val_loss: 0.1626 - val_accuracy: 0.9517


Evaluating the Test metrics
11/11 [==============================] - 12s 493ms/step - loss: 0.2275 -
accuracy: 0.9345
Test loss for fold 1: 0.22748687863349915
Test accuracy for fold 1: 0.9345238208770752
11/11 [==============================] - 1s 122ms/step
Precision for fold 1: 0.9351425705647779
Recall for fold 1: 0.9345238095238095
F1 Score for fold 1: 0.9344968942079089


Epoch 1/10
107/108 [============================>.] - ETA: 0s - loss: 1.1425 - accuracy:
0.6645
Epoch 1: val_loss improved from inf to 0.45110, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103:
UserWarning: You are saving your model as an HDF5 file via `model.save()`. This
file format is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

```
108/108 [==============================] - 11s 91ms/step - loss: 1.1376 -
accuracy: 0.6660 - val_loss: 0.4511 - val_accuracy: 0.8104
Epoch 2/10
107/108 [============================>.] - ETA: 0s - loss: 0.1745 - accuracy:
0.9402
Epoch 2: val_loss improved from 0.45110 to 0.31211, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 8s 78ms/step - loss: 0.1738 -
accuracy: 0.9405 - val_loss: 0.3121 - val_accuracy: 0.8959
Epoch 3/10
107/108 [============================>.] - ETA: 0s - loss: 0.0288 - accuracy:
0.9944
Epoch 3: val_loss improved from 0.31211 to 0.18634, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 8s 79ms/step - loss: 0.0293 -
accuracy: 0.9935 - val_loss: 0.1863 - val_accuracy: 0.9219
Epoch 4/10
107/108 [============================>.] - ETA: 0s - loss: 0.0064 - accuracy:
1.0000
Epoch 4: val_loss improved from 0.18634 to 0.16263, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 8s 79ms/step - loss: 0.0064 -
accuracy: 1.0000 - val_loss: 0.1626 - val_accuracy: 0.9368
Epoch 5/10
107/108 [============================>.] - ETA: 0s - loss: 0.0023 - accuracy:
1.0000
Epoch 5: val_loss improved from 0.16263 to 0.15170, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 79ms/step - loss: 0.0024 -
accuracy: 1.0000 - val_loss: 0.1517 - val_accuracy: 0.9442
Epoch 6/10
107/108 [============================>.] - ETA: 0s - loss: 0.0016 - accuracy:
1.0000
Epoch 6: val_loss improved from 0.15170 to 0.14723, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 79ms/step - loss: 0.0016 -
accuracy: 1.0000 - val_loss: 0.1472 - val_accuracy: 0.9517
Epoch 7/10
107/108 [============================>.] - ETA: 0s - loss: 0.0012 - accuracy:
1.0000
Epoch 7: val_loss improved from 0.14723 to 0.14434, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
```

```
108/108 [==============================] - 9s 80ms/step - loss: 0.0012 -
accuracy: 1.0000 - val_loss: 0.1443 - val_accuracy: 0.9517
Epoch 8/10
107/108 [============================>.] - ETA: 0s - loss: 0.0010 - accuracy:
1.0000
Epoch 8: val_loss improved from 0.14434 to 0.14239, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 80ms/step - loss: 0.0010 -
accuracy: 1.0000 - val_loss: 0.1424 - val_accuracy: 0.9480
Epoch 9/10
107/108 [============================>.] - ETA: 0s - loss: 8.4061e-04 -
accuracy: 1.0000
Epoch 9: val_loss improved from 0.14239 to 0.14113, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 80ms/step - loss: 8.4538e-04 -
accuracy: 1.0000 - val_loss: 0.1411 - val_accuracy: 0.9517
Epoch 10/10
107/108 [============================>.] - ETA: 0s - loss: 7.3293e-04 -
accuracy: 1.0000
Epoch 10: val_loss improved from 0.14113 to 0.14062, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 80ms/step - loss: 7.3283e-04 -
accuracy: 1.0000 - val_loss: 0.1406 - val_accuracy: 0.9480


Evaluating the Test metrics
11/11 [==============================] - 2s 121ms/step - loss: 0.1686 -
accuracy: 0.9613
Test loss for fold 2: 0.16860640048980713
Test accuracy for fold 2: 0.961309552192688
11/11 [==============================] - 1s 122ms/step
Precision for fold 2: 0.963667290886392
Recall for fold 2: 0.9613095238095238
F1 Score for fold 2: 0.9613963940006381


Epoch 1/10
107/108 [============================>.] - ETA: 0s - loss: 1.2414 - accuracy:
0.6467
Epoch 1: val_loss improved from inf to 0.40683, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103:
UserWarning: You are saving your model as an HDF5 file via `model.save()`. This
file format is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

```
108/108 [==============================] - 12s 104ms/step - loss: 1.2392 -
```

```
accuracy: 0.6465 - val_loss: 0.4068 - val_accuracy: 0.8178
Epoch 2/10
107/108 [============================>.] - ETA: 0s - loss: 0.2131 - accuracy:
0.9206
Epoch 2: val_loss improved from 0.40683 to 0.24257, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 80ms/step - loss: 0.2122 -
accuracy: 0.9209 - val_loss: 0.2426 - val_accuracy: 0.9108
Epoch 3/10
107/108 [============================>.] - ETA: 0s - loss: 0.0479 - accuracy:
0.9907
Epoch 3: val_loss improved from 0.24257 to 0.16371, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 80ms/step - loss: 0.0477 -
accuracy: 0.9907 - val_loss: 0.1637 - val_accuracy: 0.9480
Epoch 4/10
107/108 [============================>.] - ETA: 0s - loss: 0.0080 - accuracy:
0.9991
Epoch 4: val_loss improved from 0.16371 to 0.15574, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 80ms/step - loss: 0.0080 -
accuracy: 0.9991 - val_loss: 0.1557 - val_accuracy: 0.9405
Epoch 5/10
107/108 [============================>.] - ETA: 0s - loss: 0.0035 - accuracy:
1.0000
Epoch 5: val_loss improved from 0.15574 to 0.14406, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 81ms/step - loss: 0.0036 -
accuracy: 1.0000 - val_loss: 0.1441 - val_accuracy: 0.9405
Epoch 6/10
107/108 [============================>.] - ETA: 0s - loss: 0.0027 - accuracy:
1.0000
Epoch 6: val_loss improved from 0.14406 to 0.13473, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 81ms/step - loss: 0.0027 -
accuracy: 1.0000 - val_loss: 0.1347 - val_accuracy: 0.9480
Epoch 7/10
107/108 [============================>.] - ETA: 0s - loss: 0.0019 - accuracy:
1.0000
Epoch 7: val_loss did not improve from 0.13473
108/108 [==============================] - 8s 78ms/step - loss: 0.0019 -
accuracy: 1.0000 - val_loss: 0.1361 - val_accuracy: 0.9442
Epoch 8/10
107/108 [============================>.] - ETA: 0s - loss: 0.0015 - accuracy:
1.0000
Epoch 8: val_loss did not improve from 0.13473
108/108 [==============================] - 8s 78ms/step - loss: 0.0015 -
accuracy: 1.0000 - val_loss: 0.1407 - val_accuracy: 0.9480
```

```
Epoch 9/10
107/108 [============================>.] - ETA: 0s - loss: 0.0013 - accuracy:
1.0000
Epoch 9: val_loss did not improve from 0.13473
108/108 [==============================] - 8s 78ms/step - loss: 0.0013 -
accuracy: 1.0000 - val_loss: 0.1400 - val_accuracy: 0.9442
Epoch 10/10
107/108 [============================>.] - ETA: 0s - loss: 0.0011 - accuracy:
1.0000
Epoch 10: val_loss did not improve from 0.13473
108/108 [==============================] - 8s 78ms/step - loss: 0.0011 -
accuracy: 1.0000 - val_loss: 0.1381 - val_accuracy: 0.9442


Evaluating the Test metrics
11/11 [==============================] - 2s 120ms/step - loss: 0.2372 -
accuracy: 0.9286
Test loss for fold 3: 0.23715221881866455
Test accuracy for fold 3: 0.9285714030265808
11/11 [==============================] - 1s 123ms/step
Precision for fold 3: 0.9309787305944204
Recall for fold 3: 0.9285714285714286
F1 Score for fold 3: 0.9288600084970795


Epoch 1/10
107/108 [============================>.] - ETA: 0s - loss: 1.0911 - accuracy:
0.6682
Epoch 1: val_loss improved from inf to 0.39910, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103:
UserWarning: You are saving your model as an HDF5 file via `model.save()`. This
file format is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

```
108/108 [==============================] - 10s 86ms/step - loss: 1.0881 -
accuracy: 0.6688 - val_loss: 0.3991 - val_accuracy: 0.8290
Epoch 2/10
107/108 [============================>.] - ETA: 0s - loss: 0.1584 - accuracy:
0.9523
Epoch 2: val_loss improved from 0.39910 to 0.31895, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 80ms/step - loss: 0.1579 -
accuracy: 0.9526 - val_loss: 0.3190 - val_accuracy: 0.8922
Epoch 3/10
107/108 [============================>.] - ETA: 0s - loss: 0.0258 - accuracy:
0.9944
```

```
Epoch 3: val_loss improved from 0.31895 to 0.15681, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 80ms/step - loss: 0.0258 -
accuracy: 0.9944 - val_loss: 0.1568 - val_accuracy: 0.9480
Epoch 4/10
107/108 [============================>.] - ETA: 0s - loss: 0.0048 - accuracy:
1.0000
Epoch 4: val_loss improved from 0.15681 to 0.14781, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 81ms/step - loss: 0.0048 -
accuracy: 1.0000 - val_loss: 0.1478 - val_accuracy: 0.9480
Epoch 5/10
107/108 [============================>.] - ETA: 0s - loss: 0.0025 - accuracy:
1.0000
Epoch 5: val_loss improved from 0.14781 to 0.14488, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 81ms/step - loss: 0.0025 -
accuracy: 1.0000 - val_loss: 0.1449 - val_accuracy: 0.9442
Epoch 6/10
107/108 [============================>.] - ETA: 0s - loss: 0.0018 - accuracy:
1.0000
Epoch 6: val_loss improved from 0.14488 to 0.14384, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 81ms/step - loss: 0.0018 -
accuracy: 1.0000 - val_loss: 0.1438 - val_accuracy: 0.9480
Epoch 7/10
107/108 [============================>.] - ETA: 0s - loss: 0.0014 - accuracy:
1.0000
Epoch 7: val_loss improved from 0.14384 to 0.14070, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 81ms/step - loss: 0.0014 -
accuracy: 1.0000 - val_loss: 0.1407 - val_accuracy: 0.9517
Epoch 8/10
107/108 [============================>.] - ETA: 0s - loss: 0.0012 - accuracy:
1.0000
Epoch 8: val_loss improved from 0.14070 to 0.14065, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 81ms/step - loss: 0.0012 -
accuracy: 1.0000 - val_loss: 0.1407 - val_accuracy: 0.9480
Epoch 9/10
107/108 [============================>.] - ETA: 0s - loss: 9.9359e-04 -
accuracy: 1.0000
Epoch 9: val_loss did not improve from 0.14065
108/108 [==============================] - 8s 78ms/step - loss: 9.9759e-04 -
accuracy: 1.0000 - val_loss: 0.1429 - val_accuracy: 0.9480
Epoch 10/10
107/108 [============================>.] - ETA: 0s - loss: 8.7069e-04 -
accuracy: 1.0000
```

```
Epoch 10: val_loss improved from 0.14065 to 0.14020, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 10s 90ms/step - loss: 8.6784e-04 -
accuracy: 1.0000 - val_loss: 0.1402 - val_accuracy: 0.9517


Evaluating the Test metrics
11/11 [==============================] - 2s 125ms/step - loss: 0.1534 -
accuracy: 0.9554
Test loss for fold 4: 0.15336784720420837
Test accuracy for fold 4: 0.9553571343421936
11/11 [==============================] - 1s 128ms/step
Precision for fold 4: 0.9557374744769702
Recall for fold 4: 0.9553571428571429
F1 Score for fold 4: 0.9554331036007045


Epoch 1/10
107/108 [============================>.] - ETA: 0s - loss: 1.3031 - accuracy:
0.6047
Epoch 1: val_loss improved from inf to 0.39503, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103:
UserWarning: You are saving your model as an HDF5 file via `model.save()`. This
file format is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

```
108/108 [==============================] - 11s 96ms/step - loss: 1.2978 -
accuracy: 0.6065 - val_loss: 0.3950 - val_accuracy: 0.8364
Epoch 2/10
107/108 [============================>.] - ETA: 0s - loss: 0.2018 - accuracy:
0.9271
Epoch 2: val_loss improved from 0.39503 to 0.20749, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 81ms/step - loss: 0.2009 -
accuracy: 0.9274 - val_loss: 0.2075 - val_accuracy: 0.9331
Epoch 3/10
107/108 [============================>.] - ETA: 0s - loss: 0.0347 - accuracy:
0.9925
Epoch 3: val_loss improved from 0.20749 to 0.16876, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 81ms/step - loss: 0.0346 -
accuracy: 0.9926 - val_loss: 0.1688 - val_accuracy: 0.9442
Epoch 4/10
107/108 [============================>.] - ETA: 0s - loss: 0.0053 - accuracy:
1.0000
Epoch 4: val_loss improved from 0.16876 to 0.15597, saving model to
```

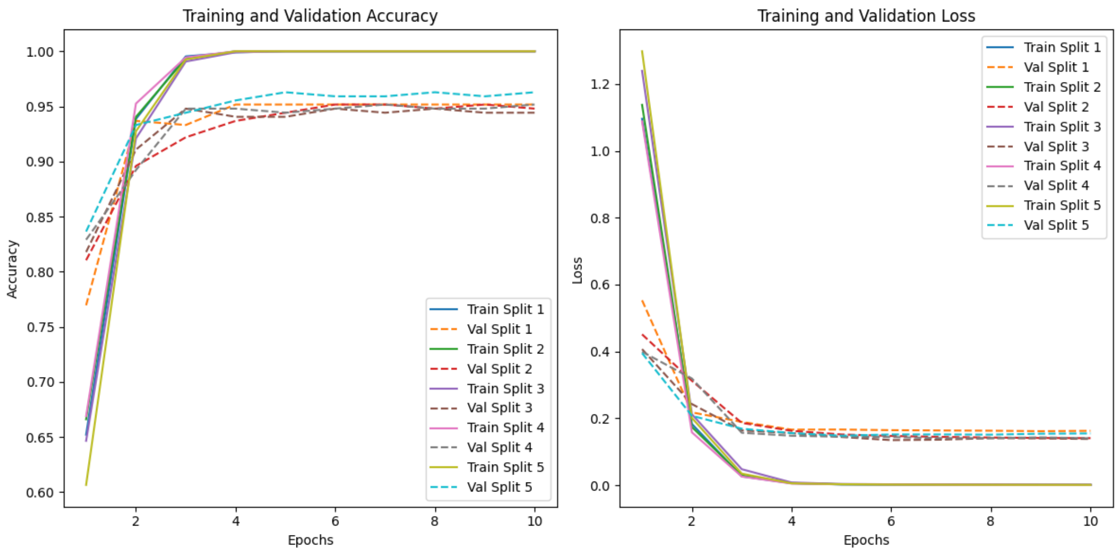
```
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 81ms/step - loss: 0.0055 -
accuracy: 1.0000 - val_loss: 0.1560 - val_accuracy: 0.9554
Epoch 5/10
107/108 [============================>.] - ETA: 0s - loss: 0.0030 - accuracy:
1.0000
Epoch 5: val_loss improved from 0.15597 to 0.14799, saving model to
/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/sheepFace_one.h5
108/108 [==============================] - 9s 82ms/step - loss: 0.0030 -
accuracy: 1.0000 - val_loss: 0.1480 - val_accuracy: 0.9628
Epoch 6/10
107/108 [============================>.] - ETA: 0s - loss: 0.0019 - accuracy:
1.0000
Epoch 6: val_loss did not improve from 0.14799
108/108 [==============================] - 8s 79ms/step - loss: 0.0019 -
accuracy: 1.0000 - val_loss: 0.1517 - val_accuracy: 0.9591
Epoch 7/10
107/108 [============================>.] - ETA: 0s - loss: 0.0014 - accuracy:
1.0000
Epoch 7: val_loss did not improve from 0.14799
108/108 [==============================] - 8s 79ms/step - loss: 0.0014 -
accuracy: 1.0000 - val_loss: 0.1517 - val_accuracy: 0.9591
Epoch 8/10
107/108 [============================>.] - ETA: 0s - loss: 0.0012 - accuracy:
1.0000
Epoch 8: val_loss did not improve from 0.14799
108/108 [==============================] - 8s 79ms/step - loss: 0.0012 -
accuracy: 1.0000 - val_loss: 0.1511 - val_accuracy: 0.9628
Epoch 9/10
107/108 [============================>.] - ETA: 0s - loss: 9.7564e-04 -
accuracy: 1.0000
Epoch 9: val_loss did not improve from 0.14799
108/108 [==============================] - 9s 79ms/step - loss: 9.7703e-04 -
accuracy: 1.0000 - val_loss: 0.1539 - val_accuracy: 0.9591
Epoch 10/10
107/108 [============================>.] - ETA: 0s - loss: 8.4613e-04 -
accuracy: 1.0000
Epoch 10: val_loss did not improve from 0.14799
108/108 [==============================] - 9s 79ms/step - loss: 8.4295e-04 -
accuracy: 1.0000 - val_loss: 0.1547 - val_accuracy: 0.9628


Evaluating the Test metrics
11/11 [==============================] - 2s 124ms/step - loss: 0.2970 -
accuracy: 0.9077
Test loss for fold 5: 0.2970311641693115
Test accuracy for fold 5: 0.9077380895614624
11/11 [==============================] - 1s 124ms/step
```

```
Precision for fold 5: 0.908393509773728
Recall for fold 5: 0.9077380952380952
F1 Score for fold 5: 0.9059085571465838


Standard deviation of accuracy across folds: 0.01928792794246443
Average Accuracy across folds: 0.9375
Average Precision across folds: 0.9387839152592576
Average Recall across folds: 0.9375
Average F1 Score across folds: 0.937218991490583
```

[9]:

```python
import matplotlib.pyplot as plt

# Determine the maximum length among all lists
max_length = max(len(train_acc) for train_acc in train_acc_list)

# Pad the shorter lists with zeros to match the maximum length
for i in range(n_splits):
    train_acc_list[i] += [0] * (max_length - len(train_acc_list[i]))
    val_acc_list[i] += [0] * (max_length - len(val_acc_list[i]))
    train_loss_list[i] += [0] * (max_length - len(train_loss_list[i]))
    val_loss_list[i] += [0] * (max_length - len(val_loss_list[i]))

# Plotting
epochs = range(1, max_length + 1)

plt.figure(figsize=(12, 6))

# Plotting Training and Validation Accuracy
plt.subplot(1, 2, 1)
for i in range(n_splits):
    plt.plot(epochs, train_acc_list[i], label=f'Train Split {i+1}')
    plt.plot(epochs, val_acc_list[i], label=f'Val Split {i+1}', linestyle='--')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# Plotting Training and Validation Loss
plt.subplot(1, 2, 2)
for i in range(n_splits):
    plt.plot(epochs, train_loss_list[i], label=f'Train Split {i+1}')
    plt.plot(epochs, val_loss_list[i], label=f'Val Split {i+1}', linestyle='--')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
```

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




```
[15]: def build_model():
    base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224,␣
 ↪224, 3))

    # Freeze first 10 convolutional layers
    for layer in base_model.layers[:15]:
        layer.trainable = False

    x = base_model.output

    # Adding Conv2D + ReLU Layers
    x = Conv2D(512, (3, 3), padding='same')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2D(512, (3, 3), padding='same', activation='relu')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2D(512, (3, 3), padding='same', activation='relu')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    # Adding Max Pooling
    x = MaxPooling2D((2, 2), strides=(2, 2))(x)
```

```
    # Add GlobalAveragePooling2D layer
    x = GlobalAveragePooling2D()(x)

    # Replacing the last three layers with new fully connected layers
    x = Dense(256)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Dropout(0.5)(x)

    x = Dense(256)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Dropout(0.5)(x)

    predictions = Dense(len(sheep_breed_list), activation='softmax')(x)  # Add␣
↪output layer

    model_normalizedd = Model(inputs=base_model.input, outputs=predictions)

    # Custom learning rates
    lr_mult = {}

    # Set learning rate to 0 for the first 10 layers
    for layer in model_normalizedd.layers[:15]:
        lr_mult[layer.name + '/kernel:0'] = 0.0
        lr_mult[layer.name + '/bias:0'] = 0.0

    # Set learning rate to 0.0001 for the next Conv2D layers and the following␣
↪layers
    for layer in model_normalizedd.layers[15:30]:
        if isinstance(layer, Conv2D):
            lr_mult[layer.name + '/kernel:0'] = 0.0001
            lr_mult[layer.name + '/bias:0'] = 0.0001

    # Set learning rate to 10 for the last two dense layers
    for layer in model_normalizedd.layers[30:]:
        lr_mult[layer.name + '/kernel:0'] = 1
        lr_mult[layer.name + '/bias:0'] = 1

    # Use SGD optimizer with initial learning rate of 1e-4
    optimizer = SGD(learning_rate=1e-4, momentum=0.9)

    # Compile the model
    model_normalizedd.compile(optimizer=optimizer,␣
↪loss='categorical_crossentropy', metrics=['accuracy'], loss_weights=[1.] +␣
↪[10.]*2)
```

```
    return model_normalizedd

accuracy_values = []
precision_values = []
recall_values = []
f1_values = []

train_acc_list = []
val_acc_list = []
train_loss_list = []
val_loss_list = []

# KFold Cross-validation
n_splits = 5
kf = KFold(n_splits=n_splits, shuffle=True, random_state=777)
fold_var = 1

for train_index, test_index in kf.split(img_data):
    X_train, X_test = img_data[train_index], img_data[test_index]
    y_train, y_test = labels_categorical[train_index],␣
 ↪labels_categorical[test_index]

    model_normalizedd = build_model()

        # Define the ImageDataGenerator for data augmentation
    datagen = ImageDataGenerator(
        width_shift_range=0.1,  # translate horizontally by 10% of total width
        height_shift_range=0.1,  # translate vertically by 10% of total height
        fill_mode='nearest'  # strategy for filling in newly created pixels
    )

    # Generate augmented images batches during training
    train_datagen = datagen.flow(X_train, y_train, batch_size=10)

    #early_stop = EarlyStopping(monitor='val_loss', patience=3, verbose=1,␣
 ↪mode='min')
    save_path='/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/
 ↪sheepFace_four.keras'
    checkpoint =␣
 ↪ModelCheckpoint(save_path,monitor='val_loss',save_best_only=True,verbose=0)
    history = model_normalizedd.fit(train_datagen, epochs=20,␣
 ↪validation_data=(X_test,y_test), verbose=1, callbacks=[checkpoint])

    print("\n")
    print(f'Evaluating the Test metrics')
    model_normalizedd=load_model(save_path)
    # Evaluate the model
```

```
    scores = model_normalizedd.evaluate(X_test, y_test, verbose=1)
    print(f'Test loss for fold {fold_var}: {scores[0]}')
    print(f'Test accuracy for fold {fold_var}: {scores[1]}')

    accuracy_values.append(scores[1])  # Appending accuracy to the list

    # Get predictions from the model
    y_true = np.argmax(y_test, axis=1)  # Convert one-hot encoded labels back to categorical labels
    y_pred = np.argmax(model_normalizedd.predict(X_test), axis=1)  # Get predictions from the model

    # Calculate and append evaluation metrics for this fold
    precision = precision_score(y_true, y_pred, average='weighted', zero_division=1)
    recall = recall_score(y_true, y_pred, average='weighted')
    f1 = f1_score(y_true, y_pred, average='weighted')

    precision_values.append(precision)
    recall_values.append(recall)
    f1_values.append(f1)

    # Print evaluation metrics for this fold
    print(f'Precision for fold {fold_var}: {precision}')
    print(f'Recall for fold {fold_var}: {recall}')
    print(f'F1 Score for fold {fold_var}: {f1}')

    train_acc_list.append(history.history['accuracy'])
    val_acc_list.append(history.history['val_accuracy'])
    train_loss_list.append(history.history['loss'])
    val_loss_list.append(history.history['val_loss'])
    print("\n")

    # Increment the fold number
    fold_var += 1

# Calculate standard deviation of accuracy across folds
accuracy_std_dev = np.std(accuracy_values)
print(f'Standard deviation of accuracy across folds: {accuracy_std_dev}')

# Calculate average of accuracy,precision, recall, and F1 across folds
avg_accuracy = np.mean(accuracy_values)
avg_precision = np.mean(precision_values)
avg_recall = np.mean(recall_values)
avg_f1 = np.mean(f1_values)

print(f'Average Accuracy across folds: {avg_accuracy}')
```

```python
print(f'Average Precision across folds: {avg_precision}')
print(f'Average Recall across folds: {avg_recall}')
print(f'Average F1 Score across folds: {avg_f1}')
```

```
Epoch 1/20
135/135 [==============================] - 18s 112ms/step - loss: 1.4963 -
accuracy: 0.3884 - val_loss: 1.0836 - val_accuracy: 0.5327
Epoch 2/20
135/135 [==============================] - 15s 110ms/step - loss: 1.0848 -
accuracy: 0.5603 - val_loss: 0.6992 - val_accuracy: 0.7530
Epoch 3/20
135/135 [==============================] - 14s 102ms/step - loss: 0.8431 -
accuracy: 0.6704 - val_loss: 0.9500 - val_accuracy: 0.6071
Epoch 4/20
135/135 [==============================] - 15s 109ms/step - loss: 0.6366 -
accuracy: 0.7552 - val_loss: 0.4003 - val_accuracy: 0.8512
Epoch 5/20
135/135 [==============================] - 15s 109ms/step - loss: 0.5589 -
accuracy: 0.7917 - val_loss: 0.3772 - val_accuracy: 0.8571
Epoch 6/20
135/135 [==============================] - 14s 106ms/step - loss: 0.4512 -
accuracy: 0.8519 - val_loss: 0.5725 - val_accuracy: 0.8006
Epoch 7/20
135/135 [==============================] - 15s 109ms/step - loss: 0.4128 -
accuracy: 0.8504 - val_loss: 0.2462 - val_accuracy: 0.9196
Epoch 8/20
135/135 [==============================] - 14s 103ms/step - loss: 0.3677 -
accuracy: 0.8810 - val_loss: 0.2631 - val_accuracy: 0.9226
Epoch 9/20
135/135 [==============================] - 15s 110ms/step - loss: 0.3378 -
accuracy: 0.8839 - val_loss: 0.2285 - val_accuracy: 0.9405
Epoch 10/20
135/135 [==============================] - 15s 109ms/step - loss: 0.3156 -
accuracy: 0.8996 - val_loss: 0.2204 - val_accuracy: 0.9315
Epoch 11/20
135/135 [==============================] - 14s 106ms/step - loss: 0.2666 -
accuracy: 0.9196 - val_loss: 0.2230 - val_accuracy: 0.9167
Epoch 12/20
135/135 [==============================] - 15s 109ms/step - loss: 0.2717 -
accuracy: 0.9122 - val_loss: 0.1781 - val_accuracy: 0.9554
Epoch 13/20
135/135 [==============================] - 14s 106ms/step - loss: 0.2107 -
accuracy: 0.9368 - val_loss: 0.1963 - val_accuracy: 0.9375
Epoch 14/20
135/135 [==============================] - 15s 108ms/step - loss: 0.2298 -
accuracy: 0.9278 - val_loss: 0.1557 - val_accuracy: 0.9673
Epoch 15/20
```

```
135/135 [==============================] - 14s 105ms/step - loss: 0.2024 -
accuracy: 0.9345 - val_loss: 0.1844 - val_accuracy: 0.9464
Epoch 16/20
135/135 [==============================] - 15s 110ms/step - loss: 0.1863 -
accuracy: 0.9405 - val_loss: 0.1517 - val_accuracy: 0.9613
Epoch 17/20
135/135 [==============================] - 15s 110ms/step - loss: 0.1719 -
accuracy: 0.9464 - val_loss: 0.1282 - val_accuracy: 0.9762
Epoch 18/20
135/135 [==============================] - 14s 104ms/step - loss: 0.1531 -
accuracy: 0.9539 - val_loss: 0.1640 - val_accuracy: 0.9494
Epoch 19/20
135/135 [==============================] - 14s 105ms/step - loss: 0.1667 -
accuracy: 0.9568 - val_loss: 0.1479 - val_accuracy: 0.9702
Epoch 20/20
135/135 [==============================] - 14s 106ms/step - loss: 0.1302 -
accuracy: 0.9680 - val_loss: 0.1393 - val_accuracy: 0.9673


Evaluating the Test metrics
11/11 [==============================] - 2s 125ms/step - loss: 0.1282 -
accuracy: 0.9762
Test loss for fold 1: 0.12819179892539978
Test accuracy for fold 1: 0.976190447807312
11/11 [==============================] - 1s 126ms/step
Precision for fold 1: 0.9763740521090573
Recall for fold 1: 0.9761904761904762
F1 Score for fold 1: 0.9761648922458949


Epoch 1/20
135/135 [==============================] - 17s 112ms/step - loss: 1.4216 -
accuracy: 0.4018 - val_loss: 1.0859 - val_accuracy: 0.5714
Epoch 2/20
135/135 [==============================] - 15s 110ms/step - loss: 0.9931 -
accuracy: 0.6004 - val_loss: 0.6225 - val_accuracy: 0.7113
Epoch 3/20
135/135 [==============================] - 15s 110ms/step - loss: 0.7850 -
accuracy: 0.6868 - val_loss: 0.4340 - val_accuracy: 0.8244
Epoch 4/20
135/135 [==============================] - 15s 111ms/step - loss: 0.5969 -
accuracy: 0.7812 - val_loss: 0.3238 - val_accuracy: 0.8988
Epoch 5/20
135/135 [==============================] - 15s 110ms/step - loss: 0.5076 -
accuracy: 0.8058 - val_loss: 0.2617 - val_accuracy: 0.9196
Epoch 6/20
135/135 [==============================] - 14s 105ms/step - loss: 0.4539 -
accuracy: 0.8296 - val_loss: 0.2634 - val_accuracy: 0.9107
```

```
Epoch 7/20
135/135 [==============================] - 15s 108ms/step - loss: 0.3958 -
accuracy: 0.8586 - val_loss: 0.1421 - val_accuracy: 0.9673
Epoch 8/20
135/135 [==============================] - 14s 105ms/step - loss: 0.3693 -
accuracy: 0.8847 - val_loss: 0.1731 - val_accuracy: 0.9464
Epoch 9/20
135/135 [==============================] - 14s 104ms/step - loss: 0.3281 -
accuracy: 0.8906 - val_loss: 0.1751 - val_accuracy: 0.9405
Epoch 10/20
135/135 [==============================] - 15s 108ms/step - loss: 0.2552 -
accuracy: 0.9271 - val_loss: 0.1268 - val_accuracy: 0.9554
Epoch 11/20
135/135 [==============================] - 14s 105ms/step - loss: 0.2250 -
accuracy: 0.9286 - val_loss: 0.1692 - val_accuracy: 0.9345
Epoch 12/20
135/135 [==============================] - 15s 109ms/step - loss: 0.2719 -
accuracy: 0.9159 - val_loss: 0.1008 - val_accuracy: 0.9702
Epoch 13/20
135/135 [==============================] - 14s 103ms/step - loss: 0.2054 -
accuracy: 0.9427 - val_loss: 0.1370 - val_accuracy: 0.9494
Epoch 14/20
135/135 [==============================] - 14s 105ms/step - loss: 0.2200 -
accuracy: 0.9345 - val_loss: 0.1213 - val_accuracy: 0.9583
Epoch 15/20
135/135 [==============================] - 14s 105ms/step - loss: 0.1942 -
accuracy: 0.9427 - val_loss: 0.1017 - val_accuracy: 0.9702
Epoch 16/20
135/135 [==============================] - 15s 108ms/step - loss: 0.1608 -
accuracy: 0.9516 - val_loss: 0.0799 - val_accuracy: 0.9821
Epoch 17/20
135/135 [==============================] - 14s 106ms/step - loss: 0.1830 -
accuracy: 0.9472 - val_loss: 0.1207 - val_accuracy: 0.9554
Epoch 18/20
135/135 [==============================] - 14s 102ms/step - loss: 0.1508 -
accuracy: 0.9546 - val_loss: 0.1240 - val_accuracy: 0.9613
Epoch 19/20
135/135 [==============================] - 15s 109ms/step - loss: 0.1591 -
accuracy: 0.9554 - val_loss: 0.0734 - val_accuracy: 0.9762
Epoch 20/20
135/135 [==============================] - 14s 105ms/step - loss: 0.1386 -
accuracy: 0.9665 - val_loss: 0.1089 - val_accuracy: 0.9524


Evaluating the Test metrics
11/11 [==============================] - 2s 128ms/step - loss: 0.0734 -
accuracy: 0.9762
Test loss for fold 2: 0.07339658588171005
```

```
Test accuracy for fold 2: 0.976190447807312
11/11 [==============================] - 2s 130ms/step
Precision for fold 2: 0.9763802209676707
Recall for fold 2: 0.9761904761904762
F1 Score for fold 2: 0.9761993841087628


Epoch 1/20
135/135 [==============================] - 18s 114ms/step - loss: 1.5529 -
accuracy: 0.3787 - val_loss: 1.2174 - val_accuracy: 0.3423
Epoch 2/20
135/135 [==============================] - 15s 111ms/step - loss: 1.0124 -
accuracy: 0.5863 - val_loss: 0.8827 - val_accuracy: 0.5833
Epoch 3/20
135/135 [==============================] - 14s 104ms/step - loss: 0.7682 -
accuracy: 0.6994 - val_loss: 0.9962 - val_accuracy: 0.5863
Epoch 4/20
135/135 [==============================] - 15s 108ms/step - loss: 0.6080 -
accuracy: 0.7634 - val_loss: 0.4824 - val_accuracy: 0.8125
Epoch 5/20
135/135 [==============================] - 14s 104ms/step - loss: 0.5265 -
accuracy: 0.8207 - val_loss: 0.5382 - val_accuracy: 0.7738
Epoch 6/20
135/135 [==============================] - 15s 109ms/step - loss: 0.4607 -
accuracy: 0.8363 - val_loss: 0.2290 - val_accuracy: 0.9226
Epoch 7/20
135/135 [==============================] - 14s 103ms/step - loss: 0.3941 -
accuracy: 0.8668 - val_loss: 0.2806 - val_accuracy: 0.9107
Epoch 8/20
135/135 [==============================] - 14s 104ms/step - loss: 0.3529 -
accuracy: 0.8772 - val_loss: 0.2308 - val_accuracy: 0.9315
Epoch 9/20
135/135 [==============================] - 15s 108ms/step - loss: 0.2866 -
accuracy: 0.8996 - val_loss: 0.1788 - val_accuracy: 0.9435
Epoch 10/20
135/135 [==============================] - 15s 109ms/step - loss: 0.2476 -
accuracy: 0.9204 - val_loss: 0.1663 - val_accuracy: 0.9583
Epoch 11/20
135/135 [==============================] - 14s 105ms/step - loss: 0.2449 -
accuracy: 0.9286 - val_loss: 0.1998 - val_accuracy: 0.9464
Epoch 12/20
135/135 [==============================] - 14s 104ms/step - loss: 0.2849 -
accuracy: 0.9018 - val_loss: 0.2388 - val_accuracy: 0.9256
Epoch 13/20
135/135 [==============================] - 14s 104ms/step - loss: 0.2358 -
accuracy: 0.9204 - val_loss: 0.1670 - val_accuracy: 0.9494
Epoch 14/20
135/135 [==============================] - 14s 102ms/step - loss: 0.1901 -
```

```
accuracy: 0.9457 - val_loss: 0.1879 - val_accuracy: 0.9464
Epoch 15/20
135/135 [==============================] - 15s 109ms/step - loss: 0.1850 -
accuracy: 0.9464 - val_loss: 0.1614 - val_accuracy: 0.9583
Epoch 16/20
135/135 [==============================] - 15s 110ms/step - loss: 0.1727 -
accuracy: 0.9516 - val_loss: 0.1476 - val_accuracy: 0.9613
Epoch 17/20
135/135 [==============================] - 14s 102ms/step - loss: 0.1994 -
accuracy: 0.9382 - val_loss: 0.1584 - val_accuracy: 0.9673
Epoch 18/20
135/135 [==============================] - 14s 105ms/step - loss: 0.1652 -
accuracy: 0.9539 - val_loss: 0.1660 - val_accuracy: 0.9673
Epoch 19/20
135/135 [==============================] - 14s 104ms/step - loss: 0.1677 -
accuracy: 0.9479 - val_loss: 0.2092 - val_accuracy: 0.9494
Epoch 20/20
135/135 [==============================] - 14s 105ms/step - loss: 0.1446 -
accuracy: 0.9561 - val_loss: 0.2040 - val_accuracy: 0.9345


Evaluating the Test metrics
11/11 [==============================] - 2s 127ms/step - loss: 0.1476 -
accuracy: 0.9613
Test loss for fold 3: 0.1475822627544403
Test accuracy for fold 3: 0.961309552192688
11/11 [==============================] - 2s 132ms/step
Precision for fold 3: 0.9615065524279209
Recall for fold 3: 0.9613095238095238
F1 Score for fold 3: 0.9612206995201181


Epoch 1/20
135/135 [==============================] - 18s 114ms/step - loss: 1.4243 -
accuracy: 0.3921 - val_loss: 0.9675 - val_accuracy: 0.6845
Epoch 2/20
135/135 [==============================] - 15s 109ms/step - loss: 0.9540 -
accuracy: 0.6071 - val_loss: 0.6019 - val_accuracy: 0.8065
Epoch 3/20
135/135 [==============================] - 15s 109ms/step - loss: 0.8068 -
accuracy: 0.6786 - val_loss: 0.3997 - val_accuracy: 0.8661
Epoch 4/20
135/135 [==============================] - 15s 108ms/step - loss: 0.6091 -
accuracy: 0.7723 - val_loss: 0.2613 - val_accuracy: 0.9196
Epoch 5/20
135/135 [==============================] - 15s 110ms/step - loss: 0.4858 -
accuracy: 0.8140 - val_loss: 0.2122 - val_accuracy: 0.9494
Epoch 6/20
```

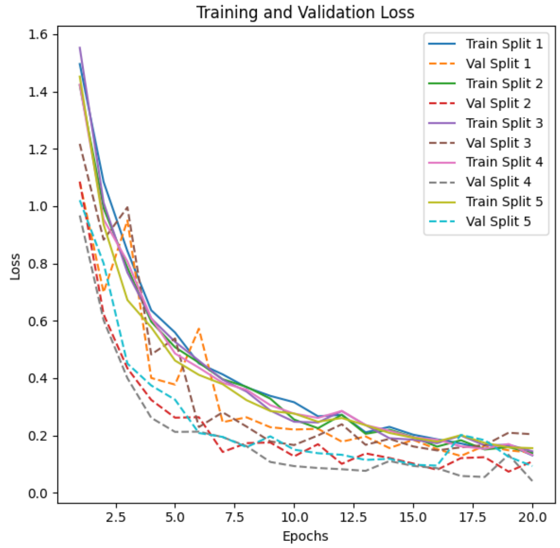
```
135/135 [==============================] - 14s 103ms/step - loss: 0.4362 -
accuracy: 0.8415 - val_loss: 0.2130 - val_accuracy: 0.9196
Epoch 7/20
135/135 [==============================] - 15s 109ms/step - loss: 0.3833 -
accuracy: 0.8616 - val_loss: 0.1938 - val_accuracy: 0.9464
Epoch 8/20
135/135 [==============================] - 15s 108ms/step - loss: 0.3593 -
accuracy: 0.8780 - val_loss: 0.1625 - val_accuracy: 0.9583
Epoch 9/20
135/135 [==============================] - 15s 109ms/step - loss: 0.3054 -
accuracy: 0.8973 - val_loss: 0.1072 - val_accuracy: 0.9732
Epoch 10/20
135/135 [==============================] - 15s 110ms/step - loss: 0.2759 -
accuracy: 0.9174 - val_loss: 0.0928 - val_accuracy: 0.9851
Epoch 11/20
135/135 [==============================] - 15s 109ms/step - loss: 0.2609 -
accuracy: 0.9137 - val_loss: 0.0861 - val_accuracy: 0.9881
Epoch 12/20
135/135 [==============================] - 15s 109ms/step - loss: 0.2854 -
accuracy: 0.9129 - val_loss: 0.0818 - val_accuracy: 0.9851
Epoch 13/20
135/135 [==============================] - 15s 110ms/step - loss: 0.2360 -
accuracy: 0.9353 - val_loss: 0.0762 - val_accuracy: 0.9970
Epoch 14/20
135/135 [==============================] - 14s 103ms/step - loss: 0.2162 -
accuracy: 0.9345 - val_loss: 0.1105 - val_accuracy: 0.9643
Epoch 15/20
135/135 [==============================] - 14s 104ms/step - loss: 0.1932 -
accuracy: 0.9427 - val_loss: 0.0934 - val_accuracy: 0.9762
Epoch 16/20
135/135 [==============================] - 14s 104ms/step - loss: 0.1851 -
accuracy: 0.9420 - val_loss: 0.0854 - val_accuracy: 0.9792
Epoch 17/20
135/135 [==============================] - 15s 108ms/step - loss: 0.1618 -
accuracy: 0.9546 - val_loss: 0.0583 - val_accuracy: 0.9851
Epoch 18/20
135/135 [==============================] - 15s 109ms/step - loss: 0.1531 -
accuracy: 0.9568 - val_loss: 0.0548 - val_accuracy: 0.9881
Epoch 19/20
135/135 [==============================] - 14s 105ms/step - loss: 0.1699 -
accuracy: 0.9524 - val_loss: 0.1322 - val_accuracy: 0.9464
Epoch 20/20
135/135 [==============================] - 15s 108ms/step - loss: 0.1313 -
accuracy: 0.9650 - val_loss: 0.0408 - val_accuracy: 0.9881


Evaluating the Test metrics
11/11 [==============================] - 2s 129ms/step - loss: 0.0408 -
```

```
accuracy: 0.9881
Test loss for fold 4: 0.04082081839442253
Test accuracy for fold 4: 0.988095223903656
11/11 [==============================] - 2s 133ms/step
Precision for fold 4: 0.9881338899196043
Recall for fold 4: 0.9880952380952381
F1 Score for fold 4: 0.9880961589106386


Epoch 1/20
135/135 [==============================] - 18s 114ms/step - loss: 1.4524 -
accuracy: 0.4211 - val_loss: 1.0209 - val_accuracy: 0.5685
Epoch 2/20
135/135 [==============================] - 15s 108ms/step - loss: 0.9361 -
accuracy: 0.6124 - val_loss: 0.8006 - val_accuracy: 0.6518
Epoch 3/20
135/135 [==============================] - 15s 110ms/step - loss: 0.6725 -
accuracy: 0.7433 - val_loss: 0.4505 - val_accuracy: 0.8423
Epoch 4/20
135/135 [==============================] - 15s 109ms/step - loss: 0.5771 -
accuracy: 0.7812 - val_loss: 0.3741 - val_accuracy: 0.8869
Epoch 5/20
135/135 [==============================] - 15s 108ms/step - loss: 0.4611 -
accuracy: 0.8400 - val_loss: 0.3246 - val_accuracy: 0.8958
Epoch 6/20
135/135 [==============================] - 15s 108ms/step - loss: 0.4111 -
accuracy: 0.8519 - val_loss: 0.2083 - val_accuracy: 0.9435
Epoch 7/20
135/135 [==============================] - 15s 109ms/step - loss: 0.3787 -
accuracy: 0.8728 - val_loss: 0.1946 - val_accuracy: 0.9554
Epoch 8/20
135/135 [==============================] - 15s 109ms/step - loss: 0.3230 -
accuracy: 0.8943 - val_loss: 0.1591 - val_accuracy: 0.9732
Epoch 9/20
135/135 [==============================] - 14s 104ms/step - loss: 0.2851 -
accuracy: 0.9055 - val_loss: 0.1967 - val_accuracy: 0.9345
Epoch 10/20
135/135 [==============================] - 15s 109ms/step - loss: 0.2763 -
accuracy: 0.9085 - val_loss: 0.1501 - val_accuracy: 0.9732
Epoch 11/20
135/135 [==============================] - 15s 110ms/step - loss: 0.2476 -
accuracy: 0.9226 - val_loss: 0.1378 - val_accuracy: 0.9702
Epoch 12/20
135/135 [==============================] - 15s 109ms/step - loss: 0.2606 -
accuracy: 0.9174 - val_loss: 0.1322 - val_accuracy: 0.9762
Epoch 13/20
135/135 [==============================] - 16s 117ms/step - loss: 0.2354 -
accuracy: 0.9219 - val_loss: 0.1143 - val_accuracy: 0.9821
```

```
Epoch 14/20
135/135 [==============================] - 16s 116ms/step - loss: 0.2086 -
accuracy: 0.9345 - val_loss: 0.1184 - val_accuracy: 0.9762
Epoch 15/20
135/135 [==============================] - 16s 116ms/step - loss: 0.1916 -
accuracy: 0.9427 - val_loss: 0.0975 - val_accuracy: 0.9821
Epoch 16/20
135/135 [==============================] - 15s 111ms/step - loss: 0.1782 -
accuracy: 0.9501 - val_loss: 0.0946 - val_accuracy: 0.9821
Epoch 17/20
135/135 [==============================] - 14s 104ms/step - loss: 0.2007 -
accuracy: 0.9397 - val_loss: 0.2015 - val_accuracy: 0.9256
Epoch 18/20
135/135 [==============================] - 14s 105ms/step - loss: 0.1738 -
accuracy: 0.9457 - val_loss: 0.1833 - val_accuracy: 0.9405
Epoch 19/20
135/135 [==============================] - 14s 106ms/step - loss: 0.1581 -
accuracy: 0.9576 - val_loss: 0.1255 - val_accuracy: 0.9702
Epoch 20/20
135/135 [==============================] - 15s 108ms/step - loss: 0.1555 -
accuracy: 0.9546 - val_loss: 0.0929 - val_accuracy: 0.9821


Evaluating the Test metrics
11/11 [==============================] - 2s 128ms/step - loss: 0.0929 -
accuracy: 0.9821
Test loss for fold 5: 0.09289499372243881
Test accuracy for fold 5: 0.9821428656578064
11/11 [==============================] - 2s 130ms/step
Precision for fold 5: 0.9822686116700202
Recall for fold 5: 0.9821428571428571
F1 Score for fold 5: 0.9821862673879936


Standard deviation of accuracy across folds: 0.00890869638175907
Average Accuracy across folds: 0.9767857074737549
Average Precision across folds: 0.9769326654188546
Average Recall across folds: 0.9767857142857143
Average F1 Score across folds: 0.9767734804346816
```

[16]:
```
import matplotlib.pyplot as plt

# Determine the maximum length among all lists
max_length = max(len(train_acc) for train_acc in train_acc_list)

# Pad the shorter lists with zeros to match the maximum length
for i in range(n_splits):
```

```
    train_acc_list[i] += [0] * (max_length - len(train_acc_list[i]))
    val_acc_list[i] += [0] * (max_length - len(val_acc_list[i]))
    train_loss_list[i] += [0] * (max_length - len(train_loss_list[i]))
    val_loss_list[i] += [0] * (max_length - len(val_loss_list[i]))

# Plotting
epochs = range(1, max_length + 1)

plt.figure(figsize=(12, 6))

# Plotting Training and Validation Accuracy
plt.subplot(1, 2, 1)
for i in range(n_splits):
    plt.plot(epochs, train_acc_list[i], label=f'Train Split {i+1}')
    plt.plot(epochs, val_acc_list[i], label=f'Val Split {i+1}', linestyle='--')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# Plotting Training and Validation Loss
plt.subplot(1, 2, 2)
for i in range(n_splits):
    plt.plot(epochs, train_loss_list[i], label=f'Train Split {i+1}')
    plt.plot(epochs, val_loss_list[i], label=f'Val Split {i+1}', linestyle='--')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

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



[10]:
```
def build_model():
    base_model = VGG19(weights='imagenet', include_top=False, input_shape=(224,␣
 ↪224, 3))

    # Freeze first 10 convolutional layers
    for layer in base_model.layers[:17]:
        layer.trainable = False

    x = base_model.output

    # Flatten to prepare for the fully connected layers
    x = Flatten()(x)

    # Fully connected + ReLU
    x = Dense(512, activation='relu')(x)
    x = Dense(1024, activation='relu')(x)

    # Adding output layer
    predictions = Dense(len(sheep_breed_list), activation='softmax')(x)

    # Create the final model
    model_VGG19 = Model(inputs=base_model.input, outputs=predictions)

    # Custom learning rates
    lr_mult = {}

    # Set learning rate to 0 for the first 10 layers
    for layer in model_VGG19 .layers[:17]:
```

```python
        lr_mult[layer.name + '/kernel:0'] = 0.0
        lr_mult[layer.name + '/bias:0'] = 0.0

    # Set learning rate to 0.0001 for the next Conv2D layers and the following␣
↪layers
    for layer in model_VGG19 .layers[17:23]:
        if isinstance(layer, Conv2D):
            lr_mult[layer.name + '/kernel:0'] = 0.001
            lr_mult[layer.name + '/bias:0'] = 0.001

    # Set learning rate to 10 for the last two dense layers
    for layer in model_VGG19 .layers[23:]:
        lr_mult[layer.name + '/kernel:0'] = 0.001
        lr_mult[layer.name + '/bias:0'] = 0.001

    # Use SGD optimizer with initial learning rate of 1e-4
    optimizer = SGD(learning_rate=1e-4,momentum=0.9)

    # Compile the model
    model_VGG19.compile(optimizer=optimizer, loss='categorical_crossentropy',␣
↪metrics=['accuracy'], loss_weights=[1.] + [10.]*2)

    return model_VGG19

accuracy_values = []
precision_values = []
recall_values = []
f1_values = []

train_acc_list = []
val_acc_list = []
train_loss_list = []
val_loss_list = []

# KFold Cross-validation
n_splits = 5
kf = KFold(n_splits=n_splits, shuffle=True, random_state=777)
fold_var = 1

for train_index, test_index in kf.split(img_data):
    X_train, X_test = img_data[train_index], img_data[test_index]
    y_train, y_test = labels_categorical[train_index],␣
↪labels_categorical[test_index]

    model_VGG19 = build_model()
```

```
    #early_stop = EarlyStopping(monitor='val_loss', patience=3, verbose=1,␣
↪mode='min')
    save_path='/content/drive/MyDrive/Deep_learning_projects/SheepFace_Data/
↪SheepFace_three.keras'
    checkpoint =␣
↪ModelCheckpoint(save_path,monitor='val_loss',save_best_only=True,verbose=0)
    history = model_VGG19.fit(X_train, y_train, batch_size=10, epochs=10,␣
↪verbose=1, validation_split=0.2, callbacks=[checkpoint])

    print("\n")
    print(f'Evaluating the Test metrics')
    model_VGG19=load_model(save_path)
    # Evaluate the model
    scores = model_VGG19.evaluate(X_test, y_test, verbose=1)
    print(f'Test loss for fold {fold_var}: {scores[0]}')
    print(f'Test accuracy for fold {fold_var}: {scores[1]}')

    accuracy_values.append(scores[1])  # Appending accuracy to the list

    # Get predictions from the model
    y_true = np.argmax(y_test, axis=1)  # Convert one-hot encoded labels back␣
↪to categorical labels
    y_pred = np.argmax(model_VGG19.predict(X_test), axis=1)  # Get predictions␣
↪from the model

    # Calculate and append evaluation metrics for this fold
    precision = precision_score(y_true, y_pred, average='weighted',␣
↪zero_division=1)
    recall = recall_score(y_true, y_pred, average='weighted')
    f1 = f1_score(y_true, y_pred, average='weighted')

    precision_values.append(precision)
    recall_values.append(recall)
    f1_values.append(f1)

    # Print evaluation metrics for this fold
    print(f'Precision for fold {fold_var}: {precision}')
    print(f'Recall for fold {fold_var}: {recall}')
    print(f'F1 Score for fold {fold_var}: {f1}')
    print("\n")

    train_acc_list.append(history.history['accuracy'])
    val_acc_list.append(history.history['val_accuracy'])
    train_loss_list.append(history.history['loss'])
    val_loss_list.append(history.history['val_loss'])
```

```python
    # Increment the fold number
    fold_var += 1

# Calculate standard deviation of accuracy across folds
accuracy_std_dev = np.std(accuracy_values)
print(f'Standard deviation of accuracy across folds: {accuracy_std_dev}')

# Calculate average of accuracy,precision, recall, and F1 across folds
avg_accuracy = np.mean(accuracy_values)
avg_precision = np.mean(precision_values)
avg_recall = np.mean(recall_values)
avg_f1 = np.mean(f1_values)

print(f'Average Accuracy across folds: {avg_accuracy}')
print(f'Average Precision across folds: {avg_precision}')
print(f'Average Recall across folds: {avg_recall}')
print(f'Average F1 Score across folds: {avg_f1}')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80134624/80134624 [==============================] - 3s 0us/step
Epoch 1/10
108/108 [==============================] - 33s 295ms/step - loss: 0.9192 -
accuracy: 0.7293 - val_loss: 0.2874 - val_accuracy: 0.8736
Epoch 2/10
108/108 [==============================] - 11s 103ms/step - loss: 0.0734 -
accuracy: 0.9786 - val_loss: 0.2228 - val_accuracy: 0.9071
Epoch 3/10
108/108 [==============================] - 11s 103ms/step - loss: 0.0050 -
accuracy: 1.0000 - val_loss: 0.1990 - val_accuracy: 0.9294
Epoch 4/10
108/108 [==============================] - 11s 104ms/step - loss: 0.0017 -
accuracy: 1.0000 - val_loss: 0.1823 - val_accuracy: 0.9368
Epoch 5/10
108/108 [==============================] - 11s 105ms/step - loss: 0.0011 -
accuracy: 1.0000 - val_loss: 0.1803 - val_accuracy: 0.9368
Epoch 6/10
108/108 [==============================] - 11s 106ms/step - loss: 8.3498e-04 -
accuracy: 1.0000 - val_loss: 0.1789 - val_accuracy: 0.9368
Epoch 7/10
108/108 [==============================] - 11s 106ms/step - loss: 6.7482e-04 -
accuracy: 1.0000 - val_loss: 0.1785 - val_accuracy: 0.9368
Epoch 8/10
108/108 [==============================] - 11s 106ms/step - loss: 5.6869e-04 -
accuracy: 1.0000 - val_loss: 0.1783 - val_accuracy: 0.9405
Epoch 9/10
108/108 [==============================] - 11s 106ms/step - loss: 4.9131e-04 -
```

```
accuracy: 1.0000 - val_loss: 0.1779 - val_accuracy: 0.9405
Epoch 10/10
108/108 [==============================] - 11s 106ms/step - loss: 4.3272e-04 -
accuracy: 1.0000 - val_loss: 0.1777 - val_accuracy: 0.9405


Evaluating the Test metrics
11/11 [==============================] - 2s 141ms/step - loss: 0.2327 -
accuracy: 0.9345
Test loss for fold 1: 0.23270520567893982
Test accuracy for fold 1: 0.9345238208770752
11/11 [==============================] - 2s 145ms/step
Precision for fold 1: 0.9347579126015797
Recall for fold 1: 0.9345238095238095
F1 Score for fold 1: 0.9338567012238453


Epoch 1/10
108/108 [==============================] - 13s 112ms/step - loss: 0.9492 -
accuracy: 0.7153 - val_loss: 0.2284 - val_accuracy: 0.9145
Epoch 2/10
108/108 [==============================] - 11s 106ms/step - loss: 0.0796 -
accuracy: 0.9786 - val_loss: 0.1523 - val_accuracy: 0.9480
Epoch 3/10
108/108 [==============================] - 11s 107ms/step - loss: 0.0082 -
accuracy: 1.0000 - val_loss: 0.1110 - val_accuracy: 0.9740
Epoch 4/10
108/108 [==============================] - 11s 107ms/step - loss: 0.0025 -
accuracy: 1.0000 - val_loss: 0.1068 - val_accuracy: 0.9665
Epoch 5/10
108/108 [==============================] - 11s 107ms/step - loss: 0.0016 -
accuracy: 1.0000 - val_loss: 0.1053 - val_accuracy: 0.9665
Epoch 6/10
108/108 [==============================] - 12s 107ms/step - loss: 0.0012 -
accuracy: 1.0000 - val_loss: 0.1051 - val_accuracy: 0.9628
Epoch 7/10
108/108 [==============================] - 12s 107ms/step - loss: 9.9825e-04 -
accuracy: 1.0000 - val_loss: 0.1048 - val_accuracy: 0.9628
Epoch 8/10
108/108 [==============================] - 12s 107ms/step - loss: 8.4112e-04 -
accuracy: 1.0000 - val_loss: 0.1040 - val_accuracy: 0.9628
Epoch 9/10
108/108 [==============================] - 10s 97ms/step - loss: 7.2585e-04 -
accuracy: 1.0000 - val_loss: 0.1042 - val_accuracy: 0.9628
Epoch 10/10
108/108 [==============================] - 11s 98ms/step - loss: 6.4048e-04 -
accuracy: 1.0000 - val_loss: 0.1042 - val_accuracy: 0.9628
```

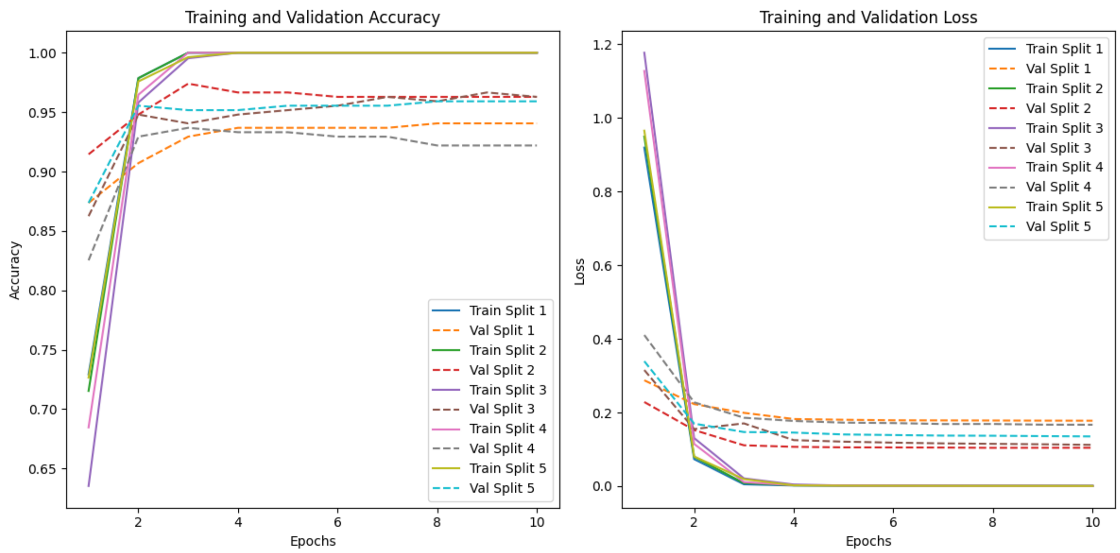
```
Evaluating the Test metrics
11/11 [==============================] - 2s 147ms/step - loss: 0.1571 -
accuracy: 0.9494
Test loss for fold 2: 0.15712346136569977
Test accuracy for fold 2: 0.949404776096344
11/11 [==============================] - 2s 153ms/step
Precision for fold 2: 0.9496049867884554
Recall for fold 2: 0.9494047619047619
F1 Score for fold 2: 0.948893698388513


Epoch 1/10
108/108 [==============================] - 13s 113ms/step - loss: 1.1769 -
accuracy: 0.6353 - val_loss: 0.3152 - val_accuracy: 0.8625
Epoch 2/10
108/108 [==============================] - 12s 107ms/step - loss: 0.1306 -
accuracy: 0.9581 - val_loss: 0.1551 - val_accuracy: 0.9480
Epoch 3/10
108/108 [==============================] - 10s 97ms/step - loss: 0.0212 -
accuracy: 0.9953 - val_loss: 0.1704 - val_accuracy: 0.9405
Epoch 4/10
108/108 [==============================] - 11s 106ms/step - loss: 0.0045 -
accuracy: 1.0000 - val_loss: 0.1249 - val_accuracy: 0.9480
Epoch 5/10
108/108 [==============================] - 12s 107ms/step - loss: 0.0017 -
accuracy: 1.0000 - val_loss: 0.1208 - val_accuracy: 0.9517
Epoch 6/10
108/108 [==============================] - 12s 107ms/step - loss: 0.0013 -
accuracy: 1.0000 - val_loss: 0.1182 - val_accuracy: 0.9554
Epoch 7/10
108/108 [==============================] - 12s 108ms/step - loss: 9.8905e-04 -
accuracy: 1.0000 - val_loss: 0.1161 - val_accuracy: 0.9628
Epoch 8/10
108/108 [==============================] - 12s 108ms/step - loss: 8.2004e-04 -
accuracy: 1.0000 - val_loss: 0.1150 - val_accuracy: 0.9591
Epoch 9/10
108/108 [==============================] - 12s 109ms/step - loss: 7.0010e-04 -
accuracy: 1.0000 - val_loss: 0.1137 - val_accuracy: 0.9665
Epoch 10/10
108/108 [==============================] - 12s 109ms/step - loss: 6.1486e-04 -
accuracy: 1.0000 - val_loss: 0.1121 - val_accuracy: 0.9628


Evaluating the Test metrics
11/11 [==============================] - 2s 146ms/step - loss: 0.2644 -
accuracy: 0.9315
Test loss for fold 3: 0.26438698172569275
```

```
Test accuracy for fold 3: 0.9315476417541504
11/11 [==============================] - 2s 152ms/step
Precision for fold 3: 0.9352569286588563
Recall for fold 3: 0.9315476190476191
F1 Score for fold 3: 0.9315647185936404


Epoch 1/10
108/108 [==============================] - 13s 112ms/step - loss: 1.1272 -
accuracy: 0.6847 - val_loss: 0.4102 - val_accuracy: 0.8253
Epoch 2/10
108/108 [==============================] - 12s 107ms/step - loss: 0.1148 -
accuracy: 0.9647 - val_loss: 0.2277 - val_accuracy: 0.9294
Epoch 3/10
108/108 [==============================] - 11s 107ms/step - loss: 0.0110 -
accuracy: 1.0000 - val_loss: 0.1858 - val_accuracy: 0.9368
Epoch 4/10
108/108 [==============================] - 12s 107ms/step - loss: 0.0025 -
accuracy: 1.0000 - val_loss: 0.1770 - val_accuracy: 0.9331
Epoch 5/10
108/108 [==============================] - 12s 107ms/step - loss: 0.0016 -
accuracy: 1.0000 - val_loss: 0.1725 - val_accuracy: 0.9331
Epoch 6/10
108/108 [==============================] - 12s 108ms/step - loss: 0.0012 -
accuracy: 1.0000 - val_loss: 0.1713 - val_accuracy: 0.9294
Epoch 7/10
108/108 [==============================] - 12s 109ms/step - loss: 9.7238e-04 -
accuracy: 1.0000 - val_loss: 0.1687 - val_accuracy: 0.9294
Epoch 8/10
108/108 [==============================] - 11s 100ms/step - loss: 8.1678e-04 -
accuracy: 1.0000 - val_loss: 0.1688 - val_accuracy: 0.9219
Epoch 9/10
108/108 [==============================] - 12s 109ms/step - loss: 7.0584e-04 -
accuracy: 1.0000 - val_loss: 0.1670 - val_accuracy: 0.9219
Epoch 10/10
108/108 [==============================] - 12s 109ms/step - loss: 6.2270e-04 -
accuracy: 1.0000 - val_loss: 0.1668 - val_accuracy: 0.9219


Evaluating the Test metrics
11/11 [==============================] - 2s 145ms/step - loss: 0.1404 -
accuracy: 0.9405
Test loss for fold 4: 0.14040616154670715
Test accuracy for fold 4: 0.9404761791229248
11/11 [==============================] - 2s 148ms/step
Precision for fold 4: 0.9408698287927612
Recall for fold 4: 0.9404761904761905
F1 Score for fold 4: 0.940162089054884
```

```
Epoch 1/10
108/108 [==============================] - 13s 112ms/step - loss: 0.9651 -
accuracy: 0.7265 - val_loss: 0.3392 - val_accuracy: 0.8736
Epoch 2/10
108/108 [==============================] - 12s 107ms/step - loss: 0.0807 -
accuracy: 0.9758 - val_loss: 0.1698 - val_accuracy: 0.9554
Epoch 3/10
108/108 [==============================] - 12s 107ms/step - loss: 0.0191 -
accuracy: 0.9963 - val_loss: 0.1469 - val_accuracy: 0.9517
Epoch 4/10
108/108 [==============================] - 12s 107ms/step - loss: 0.0021 -
accuracy: 1.0000 - val_loss: 0.1454 - val_accuracy: 0.9517
Epoch 5/10
108/108 [==============================] - 12s 108ms/step - loss: 0.0012 -
accuracy: 1.0000 - val_loss: 0.1404 - val_accuracy: 0.9554
Epoch 6/10
108/108 [==============================] - 12s 108ms/step - loss: 9.3455e-04 -
accuracy: 1.0000 - val_loss: 0.1388 - val_accuracy: 0.9554
Epoch 7/10
108/108 [==============================] - 12s 109ms/step - loss: 7.5677e-04 -
accuracy: 1.0000 - val_loss: 0.1373 - val_accuracy: 0.9554
Epoch 8/10
108/108 [==============================] - 12s 109ms/step - loss: 6.4048e-04 -
accuracy: 1.0000 - val_loss: 0.1369 - val_accuracy: 0.9591
Epoch 9/10
108/108 [==============================] - 12s 109ms/step - loss: 5.5299e-04 -
accuracy: 1.0000 - val_loss: 0.1359 - val_accuracy: 0.9591
Epoch 10/10
108/108 [==============================] - 12s 109ms/step - loss: 4.8711e-04 -
accuracy: 1.0000 - val_loss: 0.1351 - val_accuracy: 0.9591


Evaluating the Test metrics
11/11 [==============================] - 2s 147ms/step - loss: 0.2572 -
accuracy: 0.9375
Test loss for fold 5: 0.2572050988674164
Test accuracy for fold 5: 0.9375
11/11 [==============================] - 2s 152ms/step
Precision for fold 5: 0.93798679769608
Recall for fold 5: 0.9375
F1 Score for fold 5: 0.9370556523004459


Standard deviation of accuracy across folds: 0.006128348738136682
Average Accuracy across folds: 0.9386904835700989
Average Precision across folds: 0.9396952909075467
```

```
Average Recall across folds: 0.9386904761904763
Average F1 Score across folds: 0.9383065719122656
```

[11]:
```
import matplotlib.pyplot as plt

# Determine the maximum length among all lists
max_length = max(len(train_acc) for train_acc in train_acc_list)

# Pad the shorter lists with zeros to match the maximum length
for i in range(n_splits):
    train_acc_list[i] += [0] * (max_length - len(train_acc_list[i]))
    val_acc_list[i] += [0] * (max_length - len(val_acc_list[i]))
    train_loss_list[i] += [0] * (max_length - len(train_loss_list[i]))
    val_loss_list[i] += [0] * (max_length - len(val_loss_list[i]))

# Plotting
epochs = range(1, max_length + 1)

plt.figure(figsize=(12, 6))

# Plotting Training and Validation Accuracy
plt.subplot(1, 2, 1)
for i in range(n_splits):
    plt.plot(epochs, train_acc_list[i], label=f'Train Split {i+1}')
    plt.plot(epochs, val_acc_list[i], label=f'Val Split {i+1}', linestyle='--')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# Plotting Training and Validation Loss
plt.subplot(1, 2, 2)
for i in range(n_splits):
    plt.plot(epochs, train_loss_list[i], label=f'Train Split {i+1}')
    plt.plot(epochs, val_loss_list[i], label=f'Val Split {i+1}', linestyle='--')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

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



[6]:

```
from keras.applications.vgg16 import VGG16, preprocess_input
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score
from sklearn.model_selection import StratifiedKFold
import numpy as np

encoder = LabelEncoder()
labels_encoded = encoder.fit_transform(labels)

# Load the pre-trained VGG16 model without the top (classification) layers
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224,␣
↪224, 3))

# Freeze the layers of the pre-trained model
for layer in base_model.layers:
    layer.trainable = False

# Define SVM pipeline
svm_pipeline = Pipeline([('scaler', StandardScaler()), ('svm',␣
↪SVC(kernel='linear', probability=True))])

# Initialize lists to store evaluation metrics
accuracy_values = []
precision_values = []
```

```
recall_values = []
f1_values = []

# Perform k-fold cross-validation
n_splits = 5
skf = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=777)
fold_var = 1

for train_index, test_index in skf.split(img_data, labels_encoded):  # Ensure␣
↪labels_encoded is defined correctly as the non-categorical labels
    X_train, X_test = img_data[train_index], img_data[test_index]
    y_train, y_test = labels_encoded[train_index], labels_encoded[test_index]

    # Extract features using the VGG16 model and flatten them
    X_train_features = base_model.predict(preprocess_input(X_train))
    X_train_features = X_train_features.reshape(X_train_features.shape[0], -1)

    X_test_features = base_model.predict(preprocess_input(X_test))
    X_test_features = X_test_features.reshape(X_test_features.shape[0], -1)

    # Fit and transform the training data with StandardScaler and train the SVM
    svm_pipeline.fit(X_train_features, y_train)

    # Predict using the trained SVM pipeline
    predictions = svm_pipeline.predict(X_test_features)
    print(f'Fold {fold_var}:')
    print(f'Accuracy: {accuracy_score(y_test, predictions)}')
    print(f'Precision: {precision_score(y_test, predictions,␣
↪average="weighted")}')
    print(f'Recall: {recall_score(y_test, predictions, average="weighted")}')
    print(f'F1 Score: {f1_score(y_test, predictions, average="weighted")}')

    # Calculate evaluation metrics
    accuracy_values.append(accuracy_score(y_test, predictions))
    precision_values.append(precision_score(y_test, predictions,␣
↪average='weighted'))
    recall_values.append(recall_score(y_test, predictions, average='weighted'))
    f1_values.append(f1_score(y_test, predictions, average='weighted'))

    # Increment the fold number
    fold_var += 1

# Calculate average metrics across all folds
avg_accuracy = np.mean(accuracy_values)
avg_precision = np.mean(precision_values)
avg_recall = np.mean(recall_values)
avg_f1 = np.mean(f1_values)
```

```python
print(f'Average test Accuracy across folds: {avg_accuracy}')
print(f'Average Precision across folds: {avg_precision}')
print(f'Average Recall across folds: {avg_recall}')
print(f'Average F1 Score across folds: {avg_f1}')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 3s 0us/step
42/42 [==============================] - 12s 108ms/step
11/11 [==============================] - 5s 474ms/step
Fold 1:
Accuracy: 0.9553571428571429
Precision: 0.9558990724255652
Recall: 0.9553571428571429
F1 Score: 0.9548037603627753
42/42 [==============================] - 5s 118ms/step
11/11 [==============================] - 1s 116ms/step
Fold 2:
Accuracy: 0.9702380952380952
Precision: 0.9702761627906978
Recall: 0.9702380952380952
F1 Score: 0.9700502152080345
42/42 [==============================] - 5s 121ms/step
11/11 [==============================] - 1s 120ms/step
Fold 3:
Accuracy: 0.9732142857142857
Precision: 0.9734078992890944
Recall: 0.9732142857142857
F1 Score: 0.9731926934849087
42/42 [==============================] - 5s 125ms/step
11/11 [==============================] - 1s 125ms/step
Fold 4:
Accuracy: 0.9642857142857143
Precision: 0.965298202614379
Recall: 0.9642857142857143
F1 Score: 0.9642785203194726
42/42 [==============================] - 5s 130ms/step
11/11 [==============================] - 1s 128ms/step
Fold 5:
Accuracy: 0.9791666666666666
Precision: 0.9793767507002801
Recall: 0.9791666666666666
F1 Score: 0.9792188619227544
Average test Accuracy across folds: 0.9684523809523811
Average Precision across folds: 0.9688516175640034
Average Recall across folds: 0.9684523809523811
```

```
Average F1 Score across folds: 0.968308810259589
```

[12]:
```
!jupyter nbconvert --to pdf '/content/drive/MyDrive/Colab_Notebooks/
↪VGG19_Sheep_Classification.ipynb'
```

```
[NbConvertApp] Converting notebook
/content/drive/MyDrive/Colab_Notebooks/VGG19_Sheep_Classification.ipynb to pdf
[NbConvertApp] Support files will be in VGG19_Sheep_Classification_files/
[NbConvertApp] Making directory ./VGG19_Sheep_Classification_files
[NbConvertApp] Making directory ./VGG19_Sheep_Classification_files
[NbConvertApp] Making directory ./VGG19_Sheep_Classification_files
[NbConvertApp] Writing 153313 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 476732 bytes to
/content/drive/MyDrive/Colab_Notebooks/VGG19_Sheep_Classification.pdf
```