In [1]:
```
# Importing all the required imports

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

# Removing Warnings
import warnings
warnings.filterwarnings('ignore') #Ignoring the warning messages

from PIL import Image
import os
import shutil
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns
import math
from sklearn.model_selection import train_test_split

from sklearn.preprocessing import MinMaxScaler
from matplotlib.offsetbox import AnnotationBbox, OffsetImage
from sklearn.preprocessing import LabelEncoder
from skimage.transform import resize

from sklearn.manifold import TSNE
from sklearn.manifold import LocallyLinearEmbedding
from sklearn.manifold import MDS
from scipy.spatial import procrustes
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.mixture import GaussianMixture

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

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

In [2]:
```
# Deleting the 'Extracted Images' directory to avoid duplicate data
# shutil.rmtree('/content/extracted_images')
```

In [3]:
```python
import zipfile

# Extracting the required Zip File
google_drive_zip_path = '/content/drive/MyDrive/AML_HW3/Data_2.zip'

# Define the target directory to extract the contents
target_directory = '/content/extracted_images/'

# Creating the target directory if it doesn't exist
!mkdir -p {target_directory}

# Extracting the contents of the zip file to the target_directory
with zipfile.ZipFile(google_drive_zip_path, 'r') as zip_ref:
   zip_ref.extractall(target_directory)
```

In [4]:
```python
# Deleting all the un-required folders from the extracted folder

extracted_directory = '/content/extracted_images'

# Deleting all the folders except the following from the above directory

required_folders = ['rps-cv-images']

# Getting a list all items in the directory
all_folders = os.listdir(extracted_directory)

# Delete items that are not in the list of required folders
for folder in all_folders:
    item_path = os.path.join(extracted_directory, folder)
    if folder not in required_folders and os.path.isdir(item_path):
        shutil.rmtree(item_path)

# Verify the result
print("Remaining folders:")
remaining_folders = os.listdir(extracted_directory)
print(remaining_folders)
```

```
Remaining folders:
['README_rpc-cv-images.txt', 'rps-cv-images']
```

In [5]:
```python
# Counting the number of Images in each Folder

folder_path = '/content/extracted_images/rps-cv-images'
```

```
RP = ['rock', 'paper', 'scissors']
# Count the number of image files in the folder
for folder in RP:
  fp = os.path.join(folder_path, folder)
  image_count = len([f for f in os.listdir(fp) if f.endswith(('.jpg', '.png'))])
  print(f"Number of images in the {folder}: {image_count}")
```

```
Number of images in the rock: 726
Number of images in the paper: 712
Number of images in the scissors: 750
```

In [6]:

```
# Renaming the image names in each folder
# Adding the lables ('P' - Paper, 'R' - Rock, 'S' - Scissors before every file name)

# Renaming Rock Images
folder_path = '/content/extracted_images/rps-cv-images/rock'
a = 0

# Iterate over each file in the folder
for filename in os.listdir(folder_path):
    if filename.endswith('.png'):  # Adjust the file extension as needed
        current_filepath = os.path.join(folder_path, filename)

        # Define your renaming logic here
        new_filename = "R_" + str(a) + ".png" #filename

        # Create the new file path
        new_filepath = os.path.join(folder_path, new_filename)

        # Rename the file
        os.rename(current_filepath, new_filepath)
        a = a + 1

print("Rock Images renaming completed.")

# Renaming Paper Images
folder_path = '/content/extracted_images/rps-cv-images/paper'
b = 0
# Iterate over each file in the folder
for filename in os.listdir(folder_path):
    if filename.endswith('.png'):  # Adjust the file extension as needed
        current_filepath = os.path.join(folder_path, filename)

        # Define your renaming logic here
```

```
        new_filename = "P_" + str(b) + ".png"

        # Create the new file path
        new_filepath = os.path.join(folder_path, new_filename)

        # Rename the file
        os.rename(current_filepath, new_filepath)
        b = b + 1

print("Paper Images renaming completed.")

# Renaming Scissors Images
folder_path = '/content/extracted_images/rps-cv-images/scissors'
c = 0
# Iterate over each file in the folder
for filename in os.listdir(folder_path):
    if filename.endswith('.png'):  # Adjust the file extension as needed
        current_filepath = os.path.join(folder_path, filename)

        # Define your renaming logic here
        new_filename = "S_" + str(c) + ".png"

        # Create the new file path
        new_filepath = os.path.join(folder_path, new_filename)

        # Rename the file
        os.rename(current_filepath, new_filepath)
        c = c + 1

print("Scissors Images renaming completed.")
```

```
Rock Images renaming completed.
Paper Images renaming completed.
Scissors Images renaming completed.
```

In [7]:

```
# Counting the number of Renamed Images in each Folder
folder_path = '/content/extracted_images/rps-cv-images'
RP = ['rock', 'paper', 'scissors']
# Count the number of image files in the folder
for folder in RP:
  fp = os.path.join(folder_path, folder)
  image_count = len([f for f in os.listdir(fp) if f.endswith(('.jpg', '.png'))])
  print(f"Number of renamed-images in the {folder}: {image_count}")
```

```
Number of renamed-images in the rock: 726
Number of renamed-images in the paper: 712
Number of renamed-images in the scissors: 750
```

In [8]:
```
# Splitting the Data into Train & Test Data
# Creating train & test data folders, if they do not exists
!mkdir -p {'/content/extracted_images/train'}
!mkdir -p {'/content/extracted_images/test'}

def split(main_directory, train_directory, test_directory, test_size=0.2, random_state=42):
    # Assigning output directories
    train_directory = train_directory
    test_directory = test_directory

    # Gettin a list of all image files in the main directory
    image_files = [f for f in os.listdir(main_directory) if f.endswith('.png')]

    # Split the image files into train and test sets
    train_files, test_files = train_test_split(image_files, test_size=test_size, random_state=random_state)

    # Coppying the files to the respective folders
    for train_file in train_files:
        shutil.copy(os.path.join(main_directory, train_file), os.path.join(train_directory, train_file))

    for test_file in test_files:
        shutil.copy(os.path.join(main_directory, test_file), os.path.join(test_directory, test_file))
```

In [9]:
```
# Splitting images from each folder (rock, paper, scissors):
# Splitting Rock
main_directory = '/content/extracted_images/rps-cv-images/rock'
train_directory = '/content/extracted_images/train'
test_directory = '/content/extracted_images/test'
split(main_directory, train_directory, test_directory, test_size=0.2, random_state=42)
print("Rock Images have been split.")

# Splitting Paper
main_directory = '/content/extracted_images/rps-cv-images/paper'
train_directory = '/content/extracted_images/train'
test_directory = '/content/extracted_images/test'
split(main_directory, train_directory, test_directory, test_size=0.2, random_state=42)
print("Paper Images have been split.")

# Splitting Scissors
main_directory = '/content/extracted_images/rps-cv-images/scissors'
```

```
train_directory = '/content/extracted_images/train'
test_directory = '/content/extracted_images/test'
split(main_directory, train_directory, test_directory, test_size=0.2, random_state=42)
print("Scissors Images have been split.")
```

```
Rock Images have been split.
Paper Images have been split.
Scissors Images have been split.
```

In [10]:
```
# Extracting the category information from the filenames - for training & test data
cat_test, cat_train = [filename[0] for filename in os.listdir('/content/extracted_images/train')], [filename[0] for filename in
categories = list(set(cat_test + cat_train))
print(f" The total number of categories in test & train data is {len(categories)}, and the categories present are - \n\t{catego

# Counting the number of images in Train & Test folder & the number of images from each lable
folder_path = '/content/extracted_images'
RP = ['train', 'test']
# Count the number of image files in the folder
for folder in RP:
  fp = os.path.join(folder_path, folder)
  image_count = len([f for f in os.listdir(fp) if f.endswith(('.jpg', '.png'))])
  print(f"\nTotal Number of images in the {folder} folder: {image_count}")
  # Counting the occurrences of every lable in the list
  for c in np.unique([filename[0] for filename in os.listdir(os.path.join('/content/extracted_images/', folder))]):
    #categories.count("I")
    # Print the result
    print(f"\tThe lable {c} occurs {[filename[0] for filename in os.listdir(os.path.join('/content/extracted_images', folder))]
```

```
 The total number of categories in test & train data is 3, and the categories present are -
       ['S', 'R', 'P']

Total Number of images in the train folder: 1749
       The lable P occurs 569 times in the train folder.
       The lable R occurs 580 times in the train folder.
       The lable S occurs 600 times in the train folder.

Total Number of images in the test folder: 439
       The lable P occurs 143 times in the test folder.
       The lable R occurs 146 times in the test folder.
       The lable S occurs 150 times in the test folder.
```

From the above it can be seen that the test data is a representative of all the lables in the training dataset.

In [11]:
```python
# Specify the folder path
folder_path = './extracted_images/train'
#folder_path = '/Users/malhardhopate/Desktop/IUB/Fall 2023/Applied Machine Learning/HW3/Q1/Data/360 Rocks'

# Initialize a list to store the image data
images = []

# Loop through the files in the folder
for filename in os.listdir(folder_path):
    if filename.endswith('.jpg') or filename.endswith('.png'):
        # Open and load the image
        img = Image.open(os.path.join(folder_path, filename))
        new_size = (img.width // 2, img.height // 2)
        #resized_img = img.resize(new_size, Image.ANTIALIAS)
        resized_img = img.resize(new_size)
        # Convert the image to a numpy array
        img_data = np.array(resized_img)
        # Flatten the image into a 1D vector and append to the list
        images.append(img_data)

# 'images' now contains a list of flattened image data

print(f"The number of total images = {len(images)}")
print(f"The number of total categories = {len(categories)}")

# Counting the occurrences of every lable in the list
for c in np.unique([filename[0] for filename in os.listdir(folder_path)]):
    #categories.count("I")
    # Print the result
    print(f"\tThe lable {c} occurs {[filename[0] for filename in os.listdir(folder_path)].count(c)} times in the train folder."
```

```
The number of total images = 1749
The number of total categories = 3
        The lable P occurs 569 times in the train folder.
        The lable R occurs 580 times in the train folder.
        The lable S occurs 600 times in the train folder.
```

In [12]:
```python
# Flatten the 4D array into a 2D array (num_images, num_features)
images = np.stack(images) # Stacking the 'images' list along a new axis to create a new array
print(f"Original 'images' shape is: {images.shape}")
print("*"*100)

# Determining the Height, Width & Channels for rehaping the 'images' list
num_samples, height, width, channels = images.shape
```

```
images = images.reshape(num_samples, height * width * channels)
print(f"New shape for 'images' shape is: {images.shape}")
print("*"*100)
```

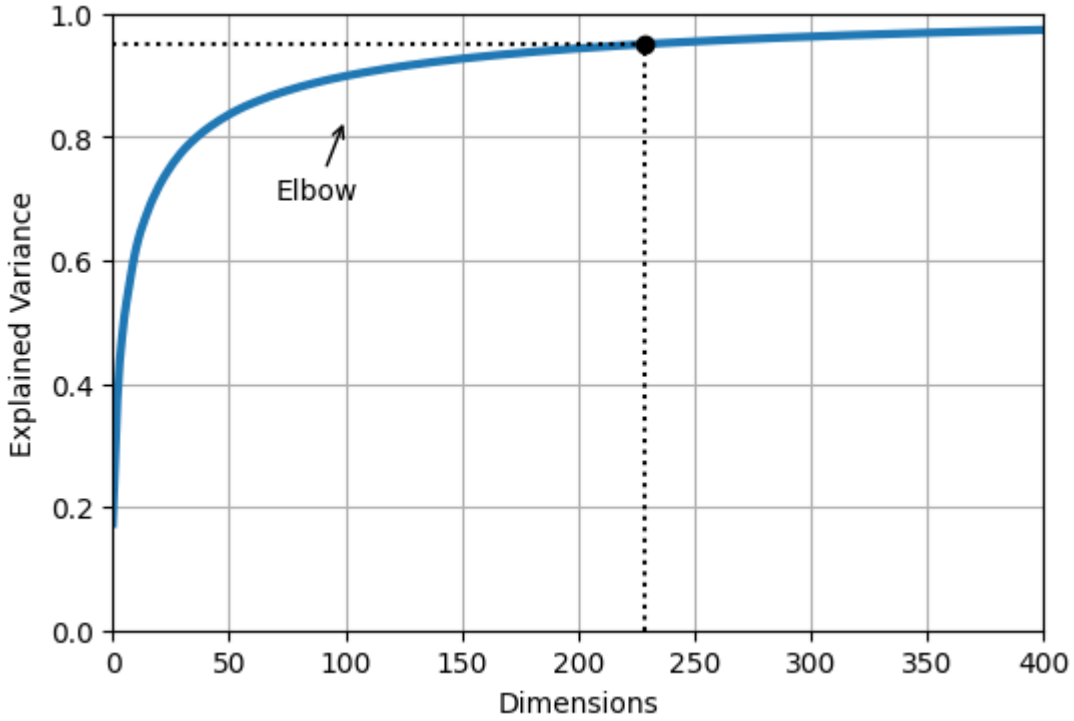
```
Original 'images' shape is: (1749, 100, 150, 3)
****************************************************************************************************
New shape for 'images' shape is: (1749, 45000)
****************************************************************************************************
```

Question 1: Apply PCA to the images. How many components do you need to preserve 95% of the variance?

In [13]:
```
# Implementing PCA
pca = PCA()
pca.fit(images)

# Finding the number of Components required to preserve 95% of the variance
cumulative_explained_variance = np.cumsum(pca.explained_variance_ratio_)
n_components_95_variance = np.argmax(cumulative_explained_variance >= 0.95) + 1

print(f"Number of components needed to preserve 95% of variance: {n_components_95_variance}")
```

```
Number of components needed to preserve 95% of variance: 229
```

In [14]:
```
# extra code – this cell generates saves the Figure to represent the number of components needs to preserve 95% variance
plt.figure(figsize=(6, 4))
plt.plot(cumulative_explained_variance, linewidth=3)
plt.axis([0, 400, 0, 1])

# Adding Lables to Axis
plt.xlabel("Dimensions")
plt.ylabel("Explained Variance")

# Plotting Dotted line for the point
plt.plot([n_components_95_variance, n_components_95_variance], [0, 0.95], "k:")
plt.plot([0, n_components_95_variance], [0.95, 0.95], "k:")
plt.plot(n_components_95_variance, 0.95, "ko")
plt.annotate("Elbow", xy=(100, 0.83), xytext=(70, 0.7),
             arrowprops=dict(arrowstyle="->"))
plt.grid(True)
plt.show()
```



As seen from above, along with the figure, we can observe that the number of required components to preserve 95% variance is - 229.

Question 2

Plot 10 images of your choice in the original form (without PCA) and then plot their reconstruction (projection in the original space) after you kept 95% of variance using PCA.

In [15]:
```
# Checking the shape of the 'images' list
images.shape
```

Out[15]:
```
(1749, 45000)
```

In [16]:
```
# Transforming the images while saving 95% of the variance
pca = PCA(0.95)
```

```
X_reduced = pca.fit_transform(images)
print(f"The shape of the reduced images is: {X_reduced.shape}")
```

```
The shape of the reduced images is: (1749, 229)
```

In [17]:
```
# Recovering and Reshaping the recovered images back to (num_images, height, width, channels)
X_recovered = pca.inverse_transform(X_reduced)
X_recovered = X_recovered.reshape(num_samples, height,width,channels)
print(f"The shape of the recovered images is: {X_recovered.shape}")
```

```
The shape of the recovered images is: (1749, 100, 150, 3)
```

In [18]:
```
X_recovered[1].shape
```

Out[18]:
```
 (100, 150, 3)
```

In [19]:
```
# Reshaping the 'images' to the original shape
images = images.reshape(num_samples, height,width,channels)
print(f"The shape of the reshaped original images is: {images.shape}")
```

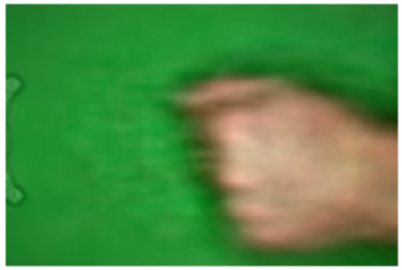
```
The shape of the reshaped original images is: (1749, 100, 150, 3)
```

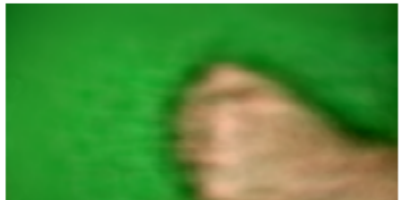
In [20]:
```
# Number of images to plot
num_images_to_plot = 10

# Create a 10x2 grid for displaying images
fig, axes = plt.subplots(num_images_to_plot, 2, figsize=(10, 20))
axes[0, 0].set_title("Original")
axes[0, 1].set_title("Reconstructed")
for i in range(num_images_to_plot):
    # Original image
    original_image = images[i].reshape(100, 150, 3)
    axes[i, 0].imshow(original_image.astype(int))
    axes[i, 0].axis("off")

    # Reconstructed image
    reconstructed_image = X_recovered[i].reshape(100, 150, 3)
    axes[i, 1].imshow(reconstructed_image.astype(int))
    axes[i, 1].axis("off")

plt.show()
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for inte
gers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for inte
gers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for inte
gers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for inte
gers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for inte
gers).
```

Original

Reconstructed





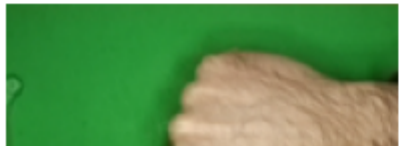
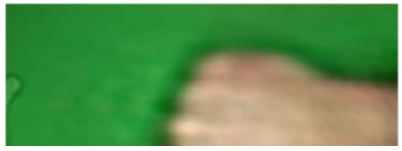










The abover figure plots the original figures again the recovered images after saving 95% of the Variance.

Question 3

Each of the images belongs to one of three rock categories. The category is indicated by the first letter in the filename (R, P & S). We will now try to see if the visualization can help us identify different clusters.

In [21]:
```
# Checking the shape of the 'images' variable
images.shape
```

Out[21]:
```
(1749, 100, 150, 3)
```

(A): Use PCA to reduce dimensionality to only 2 dimensions. How much of the variance is explained with the first two principal components?

In [22]:
```
# Implementing PCA while keeping only 2 components
pca = PCA(n_components=2)
%time X2D = pca.fit_transform(images.reshape(num_samples, height * width * channels))

print(f"\nVariance explained by each on the 2 principle components that were kept is: \n\t {pca.explained_variance_ratio_}")
print(f"\n\tThe variance explained by the 1st dimension is: {round(pca.explained_variance_ratio_[0],2)*100}%")
print(f"\tThe variance explained by the 2nd dimension is: {round(pca.explained_variance_ratio_[1],2)*100}%")

print(f"\nThe total variance explained by the 2 principle components is: {round(sum(pca.explained_variance_ratio_),2)*100}%")
print(f"The variance LOST by keeping only 2 principle components is: {round(1-sum(pca.explained_variance_ratio_),2)*100}%")
```

```
CPU times: user 6.39 s, sys: 2.02 s, total: 8.41 s
Wall time: 4.58 s

Variance explained by each on the 2 principle components that were kept is:
         [0.17287046 0.1028048 ]

        The variance explained by the 1st dimension is: 17.0%
        The variance explained by the 2nd dimension is: 10.0%

The total variance explained by the 2 principle components is: 28.000000000000004%
The variance LOST by keeping only 2 principle components is: 72.0%
```

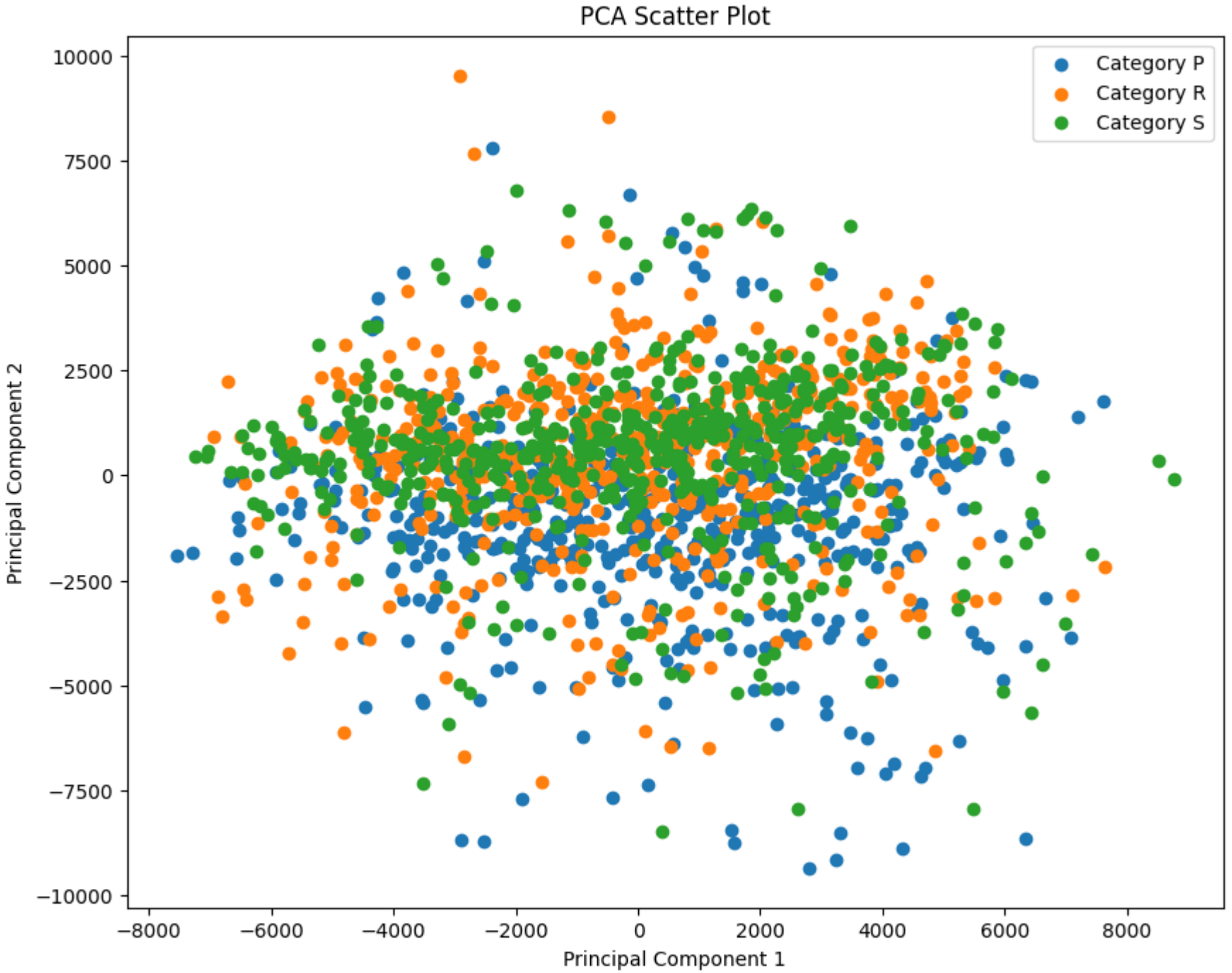
(B): Plot a 2D scatter plot of the images spanned by the first two principal components. Each image will be represented with a dot. Make the color of the dot correspond to the image category (so you will have three different colors). Then add some rock images to the visualization to better understand what features in the images are accounting for the majority of variance in the data (your visualization should look similar to the one after line 71 in this file https://github.com/ageron/handson-ml3/blob/main/08_dimensionality_reduction.ipynbLinks to an external site. but with images of rocks instead of MNIST digits). Repeat the process and create the same type of plots for t-SNE, LLE and MDS.

PCA Plot

In [23]:
```
from matplotlib.cm import get_cmap
# Create a scatter plot with the 2 principle components
categories = [filename[0] for filename in os.listdir('/content/extracted_images/train')]
plt.figure(figsize=(10, 8))
for category in sorted(set(categories)):
    # Select images and their corresponding labels for the current category
    category_images = X2D[np.array(categories) == category]
    plt.scatter(category_images[:, 0], category_images[:, 1], label=f'Category {category}')
    # Getting the default 'cmap' assigned by mat plt
    default_cmap = get_cmap()

# Print the name of the default colormap
print("Default Colormap:", default_cmap.name)
```

```
# Set labels and legend
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA Scatter Plot')
plt.legend(loc='best')

# Show the plot
plt.show()
```

```
Default Colormap: viridis
```



Plot with Image as Lables

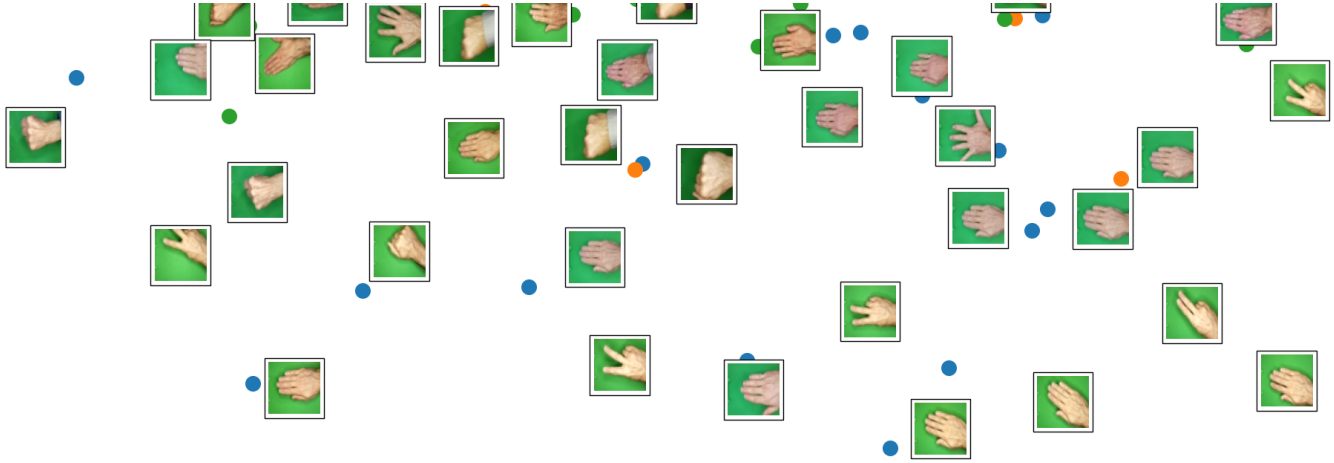
In [24]:

```
# Creating a function to plot rock images on the graph
def plot_rocks(X, rock_categories, min_distance=0.04, rock_images=None, figsize=(13, 10),image_size=(50,50)):
    # Let's scale the input features so that they range from 0 to 1
    X_normalized = MinMaxScaler().fit_transform(X)
    # Now we create the list of coordinates of the rocks plotted so far.
    neighbors = np.array([[10., 10.]])
    # The rest should be self-explanatory
    plt.figure(figsize=figsize)
    cmap = plt.cm.jet

    label_encoder = LabelEncoder()
    encoded_rock_categories = label_encoder.fit_transform(rock_categories)

    unique_rock_categories = np.unique(encoded_rock_categories)
    for category in unique_rock_categories:
        category_indices = np.where(encoded_rock_categories == category)
        category_images = X_normalized[category_indices]
        plt.scatter(category_images[:, 0], category_images[:, 1],
                    cmap="binary",
                    s=200, label="Category "+label_encoder.inverse_transform([category])[0])

    #Adding Legend
    plt.legend(fontsize=25, loc='best')

    plt.axis("off")
    ax = plt.gca()  # get current axes

    for index, image_coord in enumerate(X_normalized):
        closest_distance = np.linalg.norm(neighbors - image_coord, axis=1).min()
        if closest_distance > min_distance:
            neighbors = np.r_[neighbors, [image_coord]]
            if rock_images is None:
                category_label = label_encoder.inverse_transform([encoded_rock_categories[index]])[0]
                plt.text(image_coord[0], image_coord[1], str(category_label),
                         cmap="binary",
                         fontdict={"weight": "bold", "size": 16})
            else:
                rock_image = rock_images[index].reshape(100, 150, 3)
                rock_image = resize(rock_image, image_size, anti_aliasing=True)
                imagebox = AnnotationBbox(OffsetImage(rock_image, cmap="binary"),
                                          image_coord)
                ax.add_artist(imagebox)
```

In [25]:
```
# Plotting the required plot
plot_rocks(X2D,categories,rock_images=images.reshape(num_samples, height * width * channels),figsize=(35, 35))
plt.title('PCA Plot with Images', fontsize=40)
```

Out[25]:
```
Text(0.5, 1.0, 'PCA Plot with Images')
```

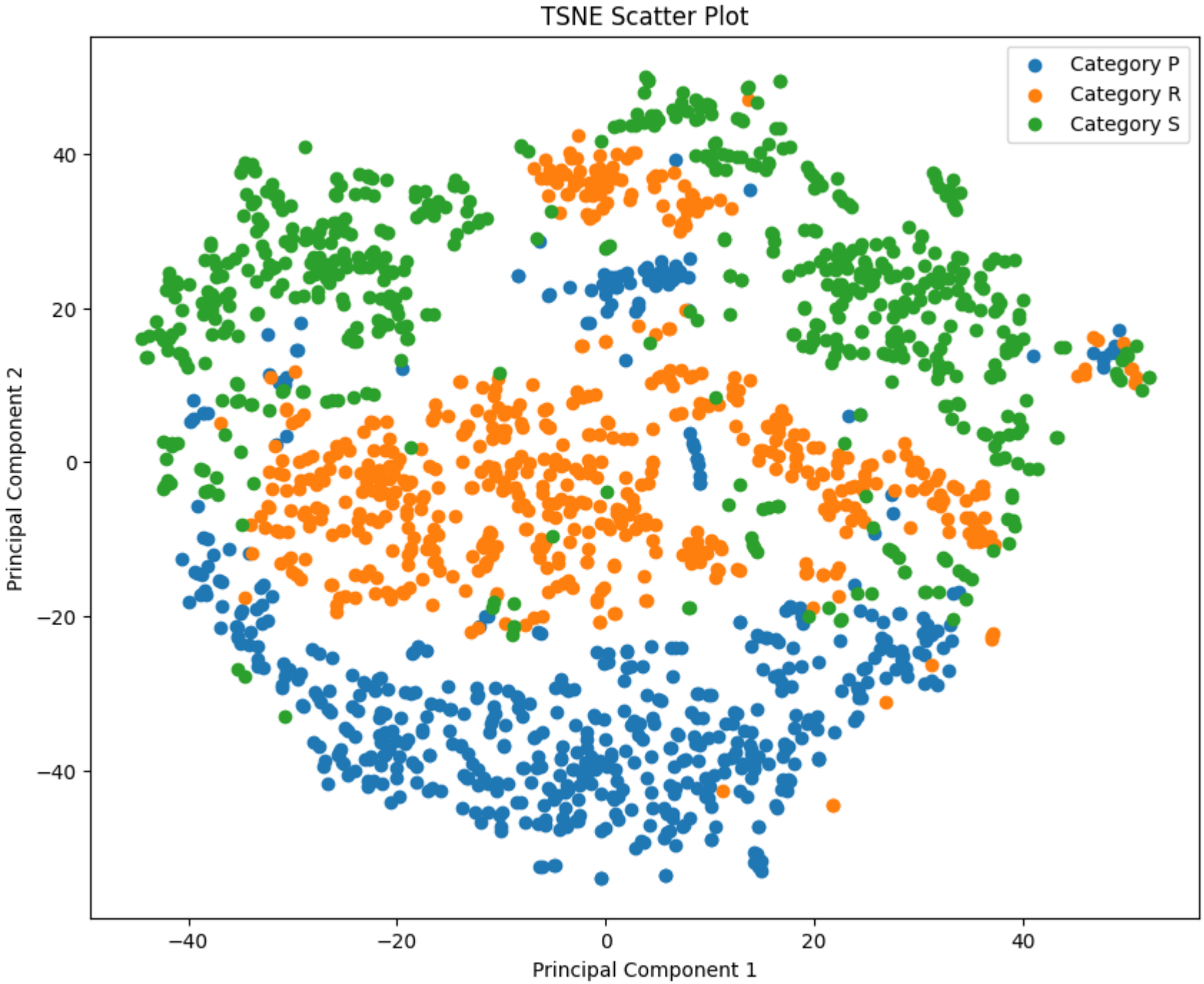

PCA Plot with Images
Category P
Category R
Category S



T-SNE Plot

In [26]:
```
# Perform t-SNE to reduce the data to 2 dimensions
tsne = TSNE(n_components=2, random_state=0)
%time X_tsne = tsne.fit_transform(images.reshape(num_samples, height * width * channels))

# Create a scatter plot
plt.figure(figsize=(10, 8))
for category in sorted(set(categories)):
    # Select images and their corresponding labels for the current category
    category_images = X_tsne[np.array(categories) == category]
    #plt.scatter(category_images[:, 0], category_images[:, 1], c=color, label=f'Category {category}')
    plt.scatter(category_images[:, 0], category_images[:, 1], cmap="binary", label=f'Category {category}')

# Set labels and legend
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('TSNE Scatter Plot')
plt.legend(loc='best')

# Show the plot
plt.show()
```

```
CPU times: user 38.8 s, sys: 1.79 s, total: 40.6 s
Wall time: 28.3 s
```

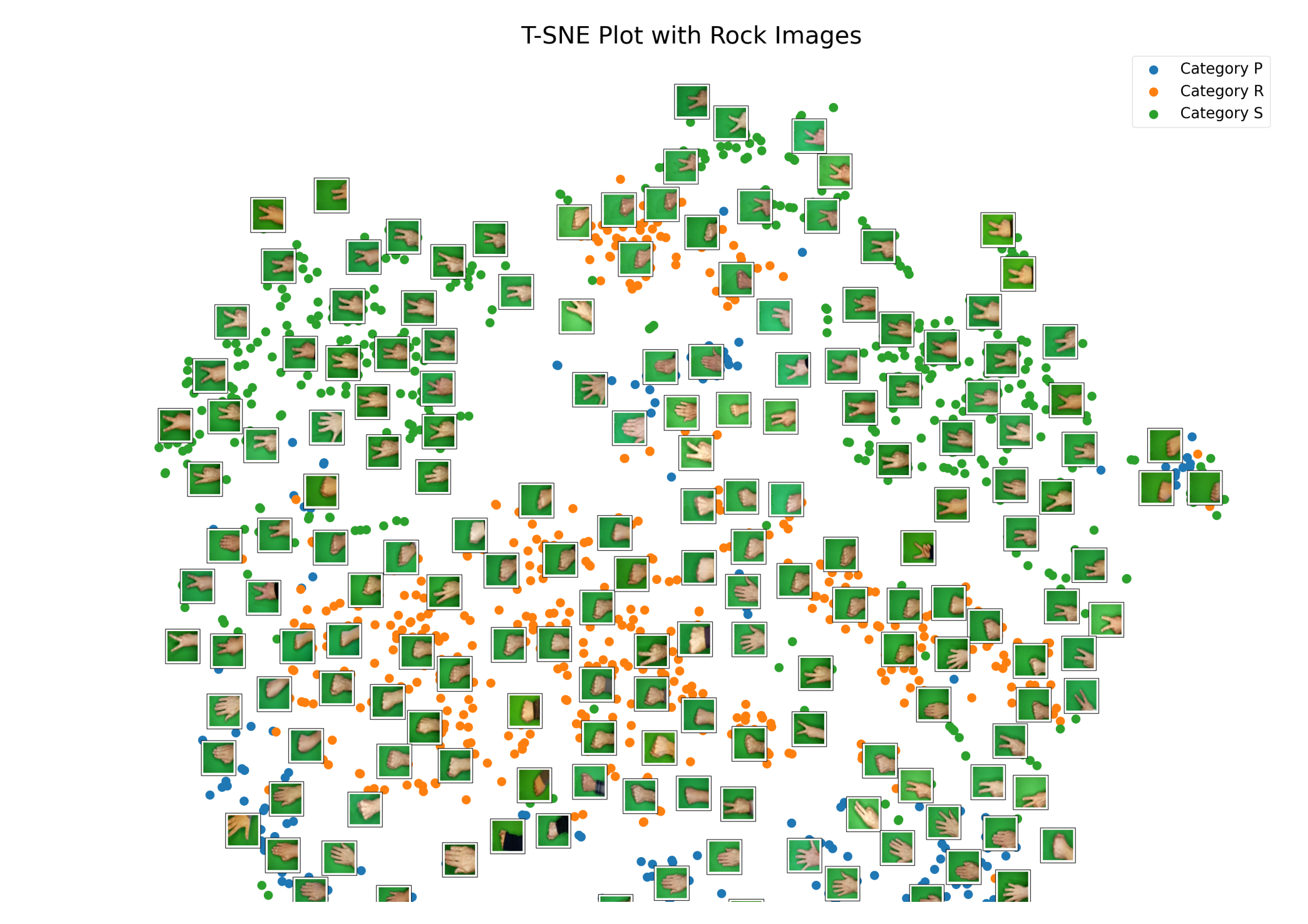


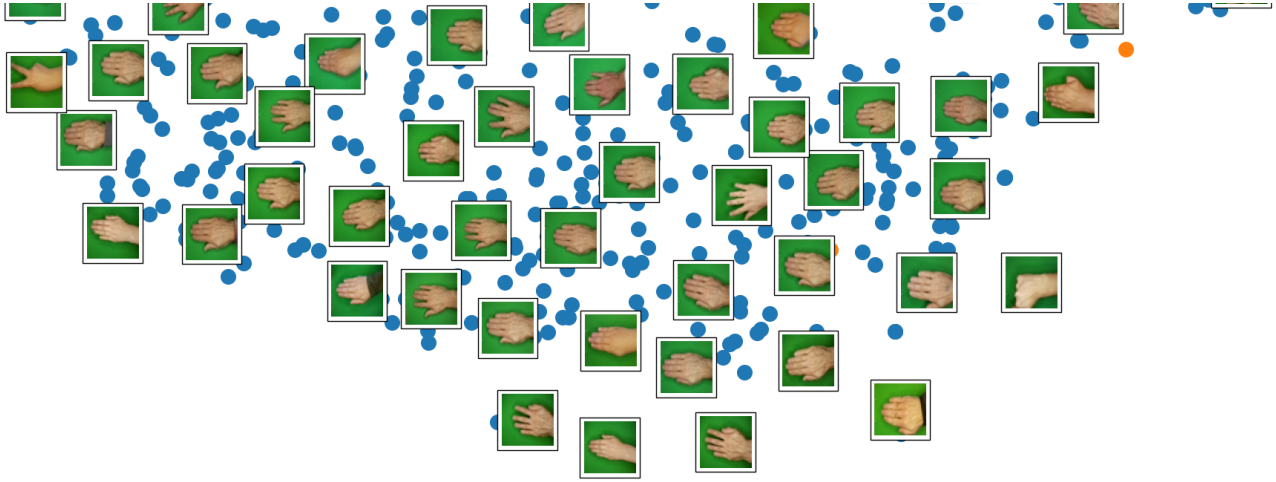
In [27]:
```
# Plotting the required plot
plot_rocks(X_tsne,categories,rock_images=images.reshape(num_samples, height * width * channels),figsize=(35, 35))
```

```
plt.title('T-SNE Plot with Rock Images', fontsize=40)
```

Out[27]:

```
Text(0.5, 1.0, 'T-SNE Plot with Rock Images')
```

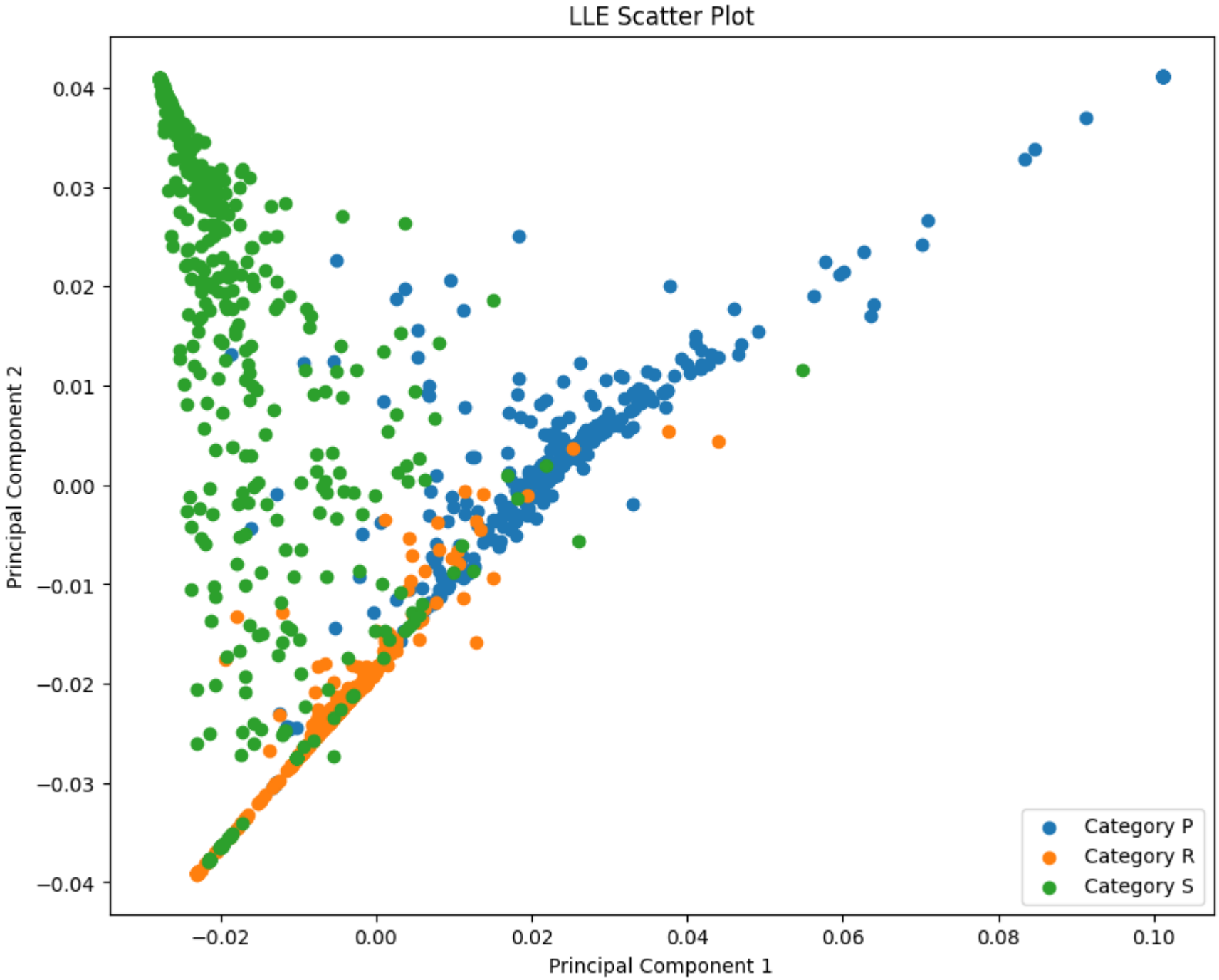

T-SNE Plot with Rock Images
Category P
Category R
Category S



LLE Plot

In [28]:
```
# Perform LLE to reduce the data to 2 dimensions
lle = LocallyLinearEmbedding(n_components=2, n_neighbors=5, random_state=0)
%time X_lle = lle.fit_transform(images.reshape(num_samples, height * width * channels))

# Create a scatter plot
plt.figure(figsize=(10, 8))
for category in sorted(set(categories)):
    # Select images and their corresponding labels for the current category
    category_images = X_lle[np.array(categories) == category]
    plt.scatter(category_images[:, 0], category_images[:, 1], cmap="binary", label=f'Category {category}')

# Set labels and legend
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(loc='best')
plt.title('LLE Scatter Plot')
# Show the plot
plt.show()
```

```
CPU times: user 12.2 s, sys: 363 ms, total: 12.5 s
Wall time: 10.1 s
```

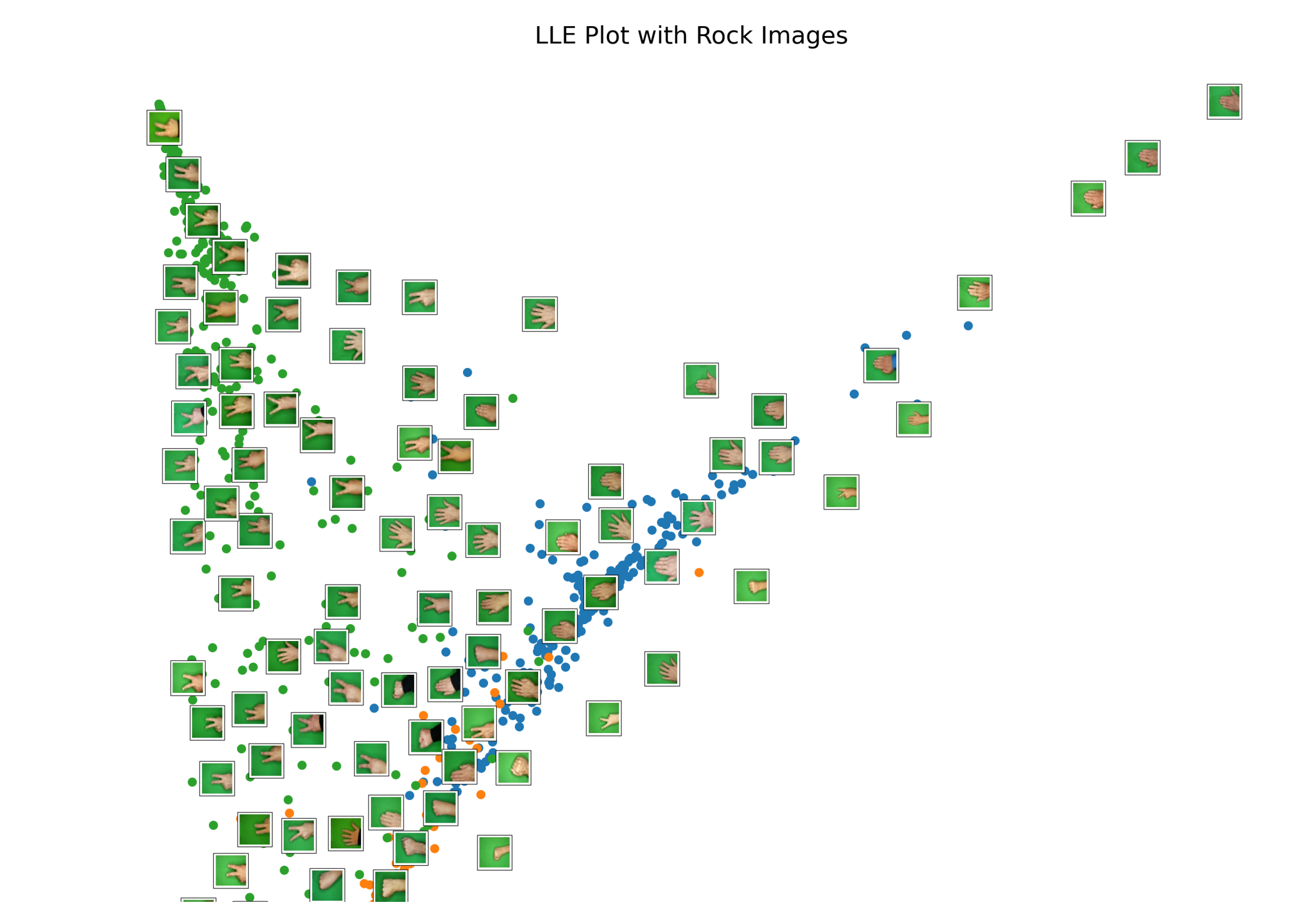


In [29]:
```
# Plotting the required plot
plot_rocks(X_lle,categories,rock_images=images.reshape(num_samples, height * width * channels),figsize=(35, 35))
```

```
plt.title('LLE Plot with Rock Images', fontsize=40)
```

Out[29]:

```
Text(0.5, 1.0, 'LLE Plot with Rock Images')
```


LLE Plot with Rock Images

MDS Plot

In [30]:
```
# Perform MDS to reduce the data to 2 dimensions
mds = MDS(n_components=2, random_state=0)
%time X_mds = mds.fit_transform(images.reshape(num_samples, height * width * channels))

# Create a scatter plot
plt.figure(figsize=(10, 8))
for category in sorted(set(categories)):
    # Select images and their corresponding labels for the current category
    category_images = X_mds[np.array(categories) == category]
    plt.scatter(category_images[:, 0], category_images[:, 1], cmap="binary", label=f'Category {category}')

# Set labels and legend
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(loc='best')
plt.title('MDS Scatter Plot')
# Show the plot
plt.show()
```

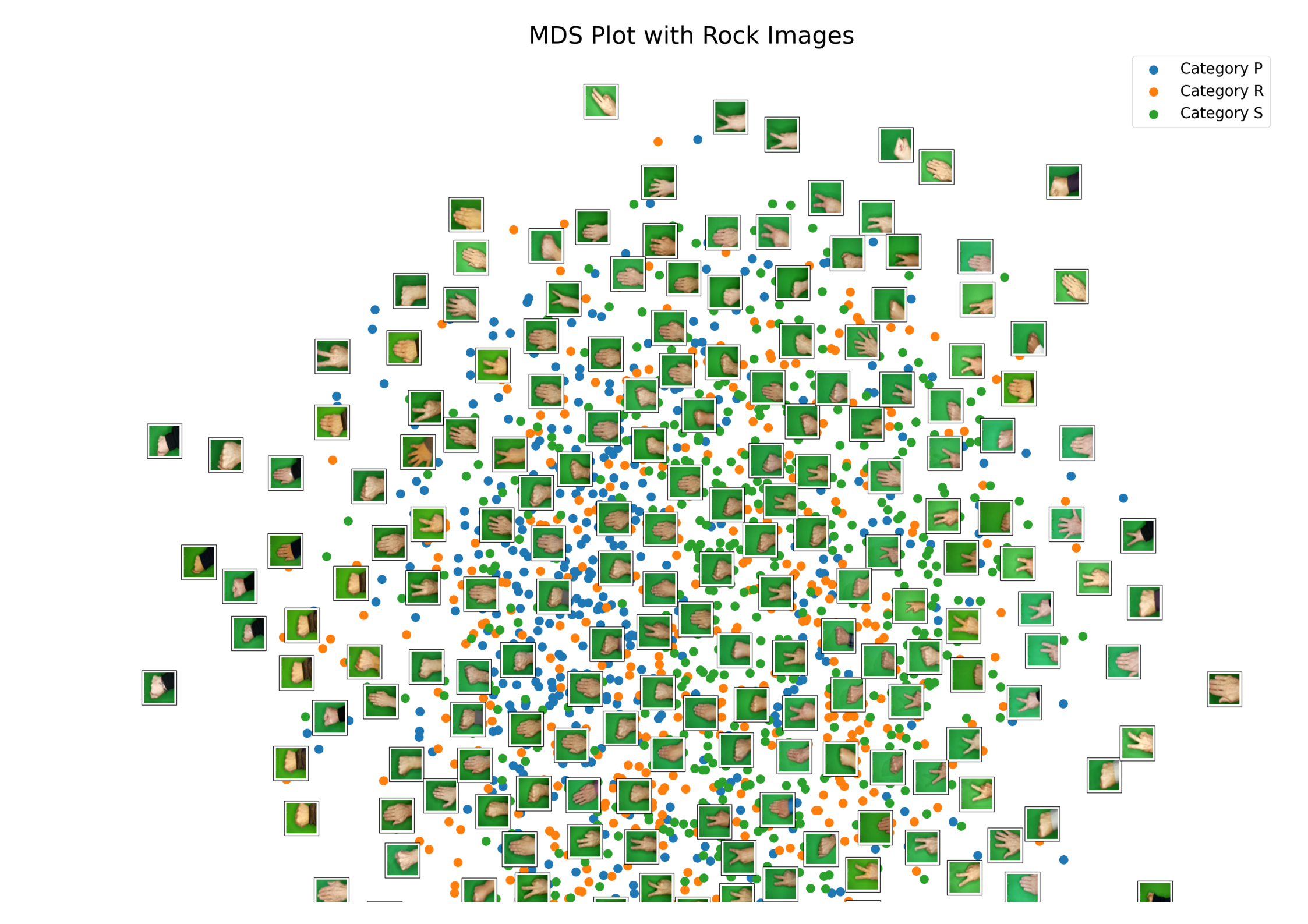
```
CPU times: user 1min 29s, sys: 52.8 s, total: 2min 22s
Wall time: 1min 29s
```

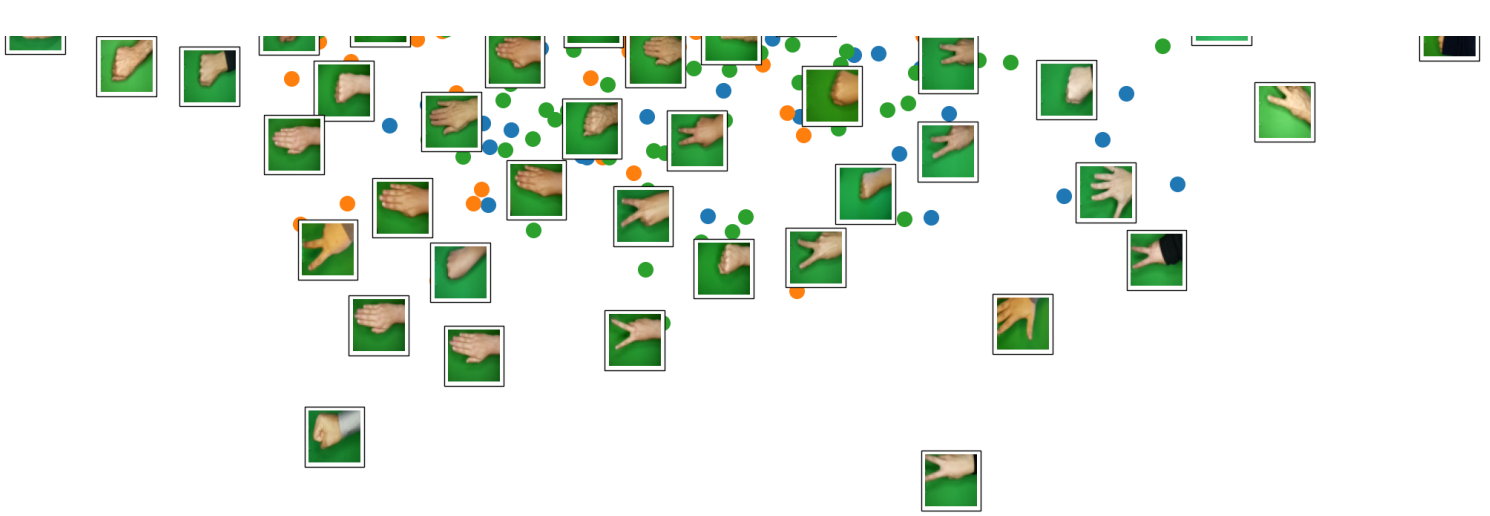
In [31]:
```
# Plotting the required plot
plot_rocks(X_mds,categories,rock_images=images.reshape(num_samples, height * width * channels),figsize=(35, 35))
```

```
plt.title('MDS Plot with Rock Images', fontsize=40)
```

Out[31]:

```
Text(0.5, 1.0, 'MDS Plot with Rock Images')
```


MDS Plot with Rock Images
Category P
Category R
Category S



(C): Which of the visualizations do you prefer?

Answer -

- T-SNE produces the best visualization in the above, as we can clearly observe some clustering. Whereas, no distinct clusters are observed in PCA, LLE, or MDS
- The PCA, and MDS plot similar scatter plots. While the LLE does not do a good job as there are many datapoints from different categories being overlapped.
- Therefore, T-SNE by far produces the graph which provides best clustering outcome, as it is able to display some distinct clustering and some overlapping datapoints.

Question 4

Now let's see if these dimensionality reduction techniques can give us similar features to those that humans use to judge the images. File mds_360.txt contains 8 features for each of the images (rankings are in the same order as the images in '360 Rocks' folder. Run PCA, t-SNE, LLE and MDS to reduce the dimensionality of the images to 8. Then, compare those

image embeddings with the ones from humans that are in the mds_360.txt file. Use Procrustes analysis to do the comparison (here is one example of how to do that mtx1, mtx2, disparity = procrustes(matrix_with_human_data, matrix_with_pca_embeddings_data). Here matrix_with_human_data and matrix_with_pca_embeddings_data should be 360 by 8. disparity will tell you the difference in the data. Report disparity for each of the four dimensionality reduction methods. Compute the correlation coefficient between each dimension of mtx1 and mtx2 for each of the four methods - display results in a table.

Not required for Personal Dataset

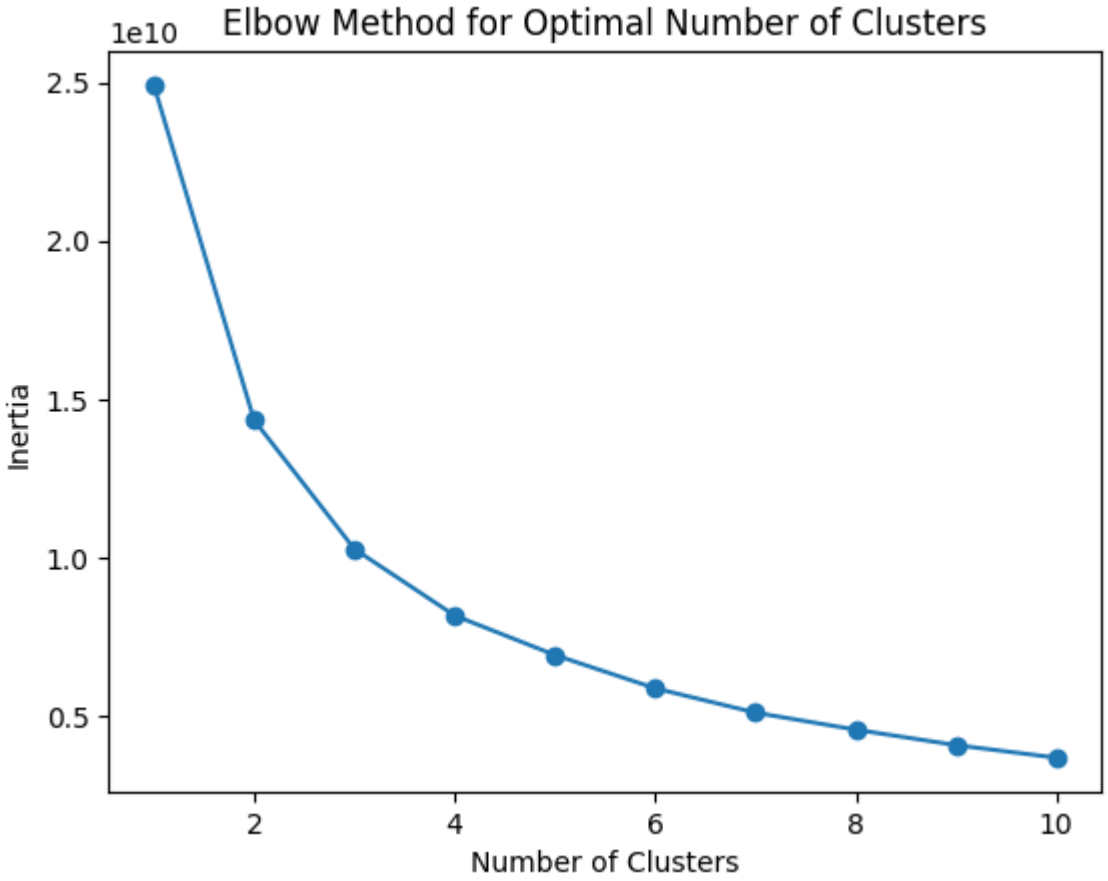
Question 5

Cluster the 360 images using K-Means.

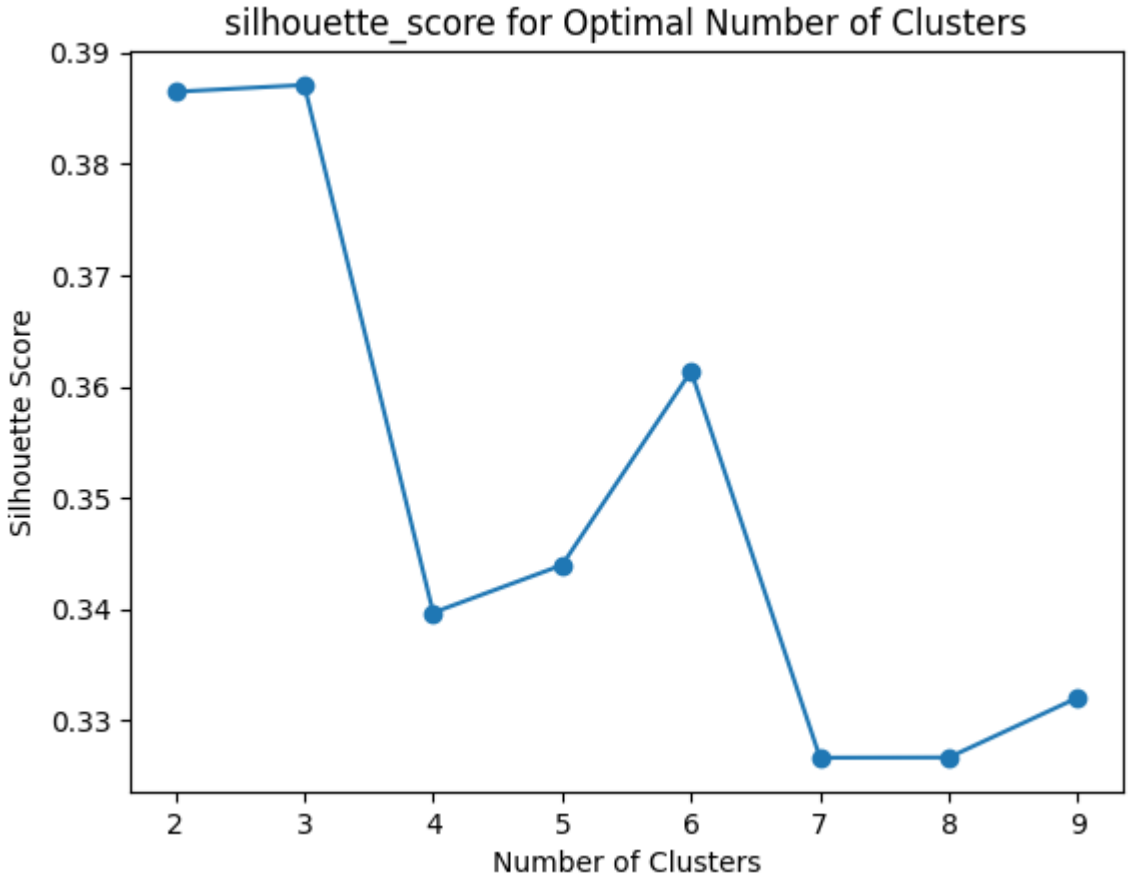
(A)To speed up the algorithm, use PCA to reduce the dimensionality of the dataset to two. Determine the number of clusters using one of the techniques we discussed in class.

In [32]:
```
# Conducting PCA to reduce the dimensions to 2
pca = PCA(n_components=2)
X_reduced = pca.fit_transform(images.reshape(num_samples, height * width * channels))

# Determine the optimal number of clusters using the K-Means++ method
inertia = []
for n_clusters in range(1, 11):  # You can adjust the range as needed
    kmeans = KMeans(n_clusters=n_clusters, init='k-means++', random_state=0)
    kmeans.fit(X_reduced)
    inertia.append(kmeans.inertia_)

# Plot the elbow method to determine the optimal number of clusters
plt.plot(range(1, 11), inertia, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
```

```
plt.title('Elbow Method for Optimal Number of Clusters')
plt.show()
```



In [33]:
```
# Determining the number of optimum clusters using silhouette_score
sil_coeff = []
for n_clusters in range(2, 10):  # You can adjust the range as needed
    kmeans = KMeans(n_clusters=n_clusters, init='k-means++', random_state=0)
    kmeans.fit(X_reduced)
    sil_coeff.append(silhouette_score(X_reduced, kmeans.labels_))

# Plot the silhouette_score to determine the optimal number of clusters
plt.plot(range(2, 10), sil_coeff, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')
plt.title('silhouette_score for Optimal Number of Clusters')
plt.show()
```



Based on the above graphs, the optimum number of clusters obtained from 'K-means++' is 3

In [34]:
```
# Based on the above graphs, we choose the optimal number of clusters
optimal_n_clusters = 3  # Adjust this based on the plot

# Apply K-Means clustering with the optimal number of clusters
kmeans = KMeans(n_clusters=optimal_n_clusters, init='k-means++', random_state=0)
kmeans.fit(X_reduced)
labels = kmeans.labels_
centroids = kmeans.cluster_centers_
print (f"The centroids are -\n {centroids}")
```

```
The centroids are -
 [[-2804.79053915    93.90578552]
 [ 2119.6021563   1361.86022562]
 [ 1951.51133188 -3184.10614678]]
```

(B) Visualize the clusters in a similar way to the visualization after line 28 here: https://github.com/ageron/handson-ml3/blob/main/09_unsupervised_learning.ipynbLinks to an external site., but color each dot based on the clusters it belongs to using the labels taken from the filename as in question 3 (I, M and S).

In [35]:

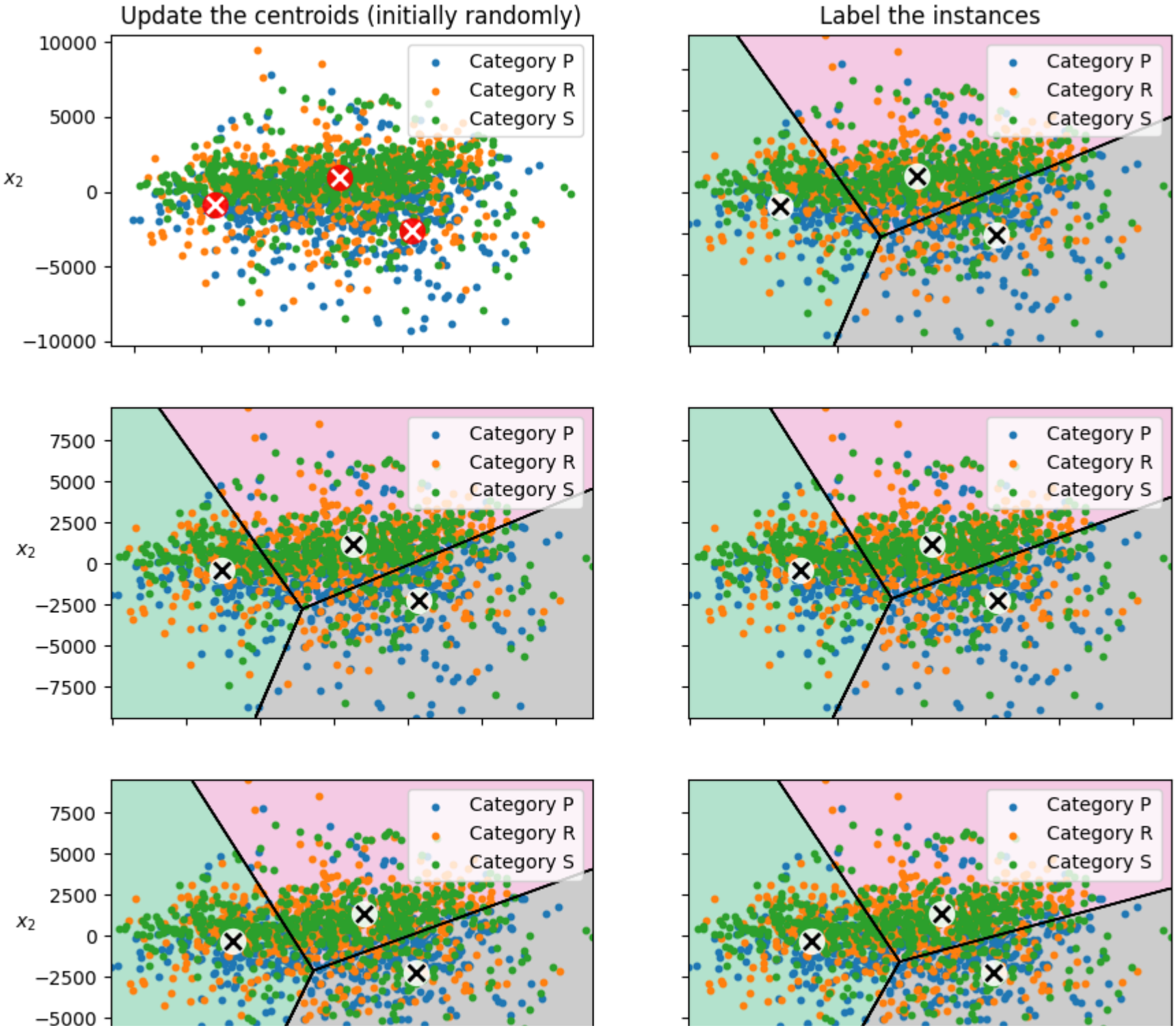
```python
# Defining functions to plot decision boundaries, centroids, and data
def plot_data(X, cat, c="viridis"):
    for c in sorted(set(cat)):
        cat_img = X[np.array(cat) == c]
        plt.scatter(cat_img[:, 0], cat_img[:, 1], cmap=c, s=10, label="Category " + c)
    plt.legend(loc='best', fontsize=10)

def plot_centroids(centroids, weights=None, circle_color='w', cross_color='k'):
    if weights is not None:
        centroids = centroids[weights > weights.max() / 10]
    plt.scatter(centroids[:, 0], centroids[:, 1],
                marker='o', s=35, linewidths=8,
                color=circle_color, zorder=10, alpha=0.9)
    plt.scatter(centroids[:, 0], centroids[:, 1],
                marker='x', s=2, linewidths=12,
                color=cross_color, zorder=11, alpha=1)

def plot_decision_boundaries(clusterer, X, cat, resolution=1000, show_centroids=True,
                             show_xlabels=True, show_ylabels=True):
    mins = X.min(axis=0) - 0.1
    maxs = X.max(axis=0) + 0.1
    xx, yy = np.meshgrid(np.linspace(mins[0], maxs[0], resolution),
                         np.linspace(mins[1], maxs[1], resolution))
    Z = clusterer.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.contourf(Z, extent=(mins[0], maxs[0], mins[1], maxs[1]),
                cmap="Pastel2")
    plt.contour(Z, extent=(mins[0], maxs[0], mins[1], maxs[1]),
                linewidths=1, colors='k')
    plot_data(X, cat, c="viridis")
```

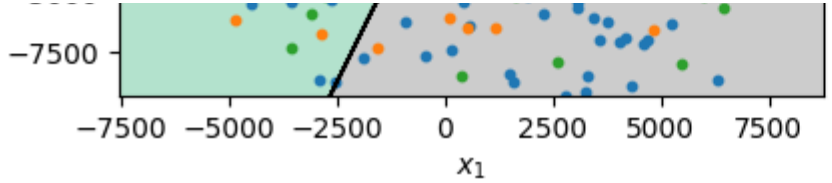
```
    if show_centroids:
        plot_centroids(clusterer.cluster_centers_)

    if show_xlabels:
        plt.xlabel("$x_1$")
    else:
        plt.tick_params(labelbottom=False)
    if show_ylabels:
        plt.ylabel("$x_2$", rotation=0)
    else:
        plt.tick_params(labelleft=False)
```

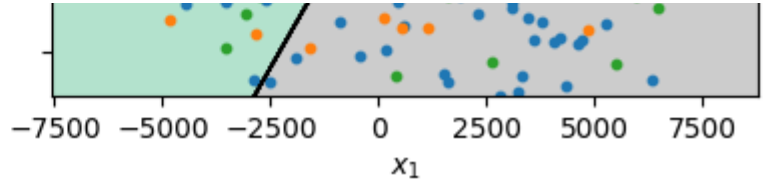
In [36]:

```
# Running 'k-means++' Clustering through many iteration
kmeans_iter1 = KMeans(n_clusters=optimal_n_clusters, init="k-means++", n_init=1, max_iter=1,
                      random_state=5)
kmeans_iter2 = KMeans(n_clusters=optimal_n_clusters, init="k-means++", n_init=1, max_iter=2,
                      random_state=5)
kmeans_iter3 = KMeans(n_clusters=optimal_n_clusters, init="k-means++", n_init=1, max_iter=3,
                      random_state=5)
kmeans_iter1.fit(X_reduced)
kmeans_iter2.fit(X_reduced)
kmeans_iter3.fit(X_reduced)

plt.figure(figsize=(10, 10))

plt.subplot(321)
plot_data(X_reduced, categories)
plot_centroids(kmeans_iter1.cluster_centers_, circle_color='r', cross_color='w')
plt.ylabel("$x_2$", rotation=0)
plt.tick_params(labelbottom=False)
plt.title("Update the centroids (initially randomly)")

plt.subplot(322)
plot_decision_boundaries(kmeans_iter1, X_reduced, categories, show_xlabels=False,
                         show_ylabels=False)
plt.title("Label the instances")

plt.subplot(323)
plot_decision_boundaries(kmeans_iter1, X_reduced, categories, show_centroids=False,
                         show_xlabels=False)
plot_centroids(kmeans_iter2.cluster_centers_)

plt.subplot(324)
plot_decision_boundaries(kmeans_iter2, X_reduced, categories, show_xlabels=False,
```

```
                          show_ylabels=False)

plt.subplot(325)
plot_decision_boundaries(kmeans_iter2, X_reduced, categories, show_centroids=False)
plot_centroids(kmeans_iter3.cluster_centers_)

plt.subplot(326)
plot_decision_boundaries(kmeans_iter3, X_reduced, categories, show_ylabels=False)

plt.show()
```

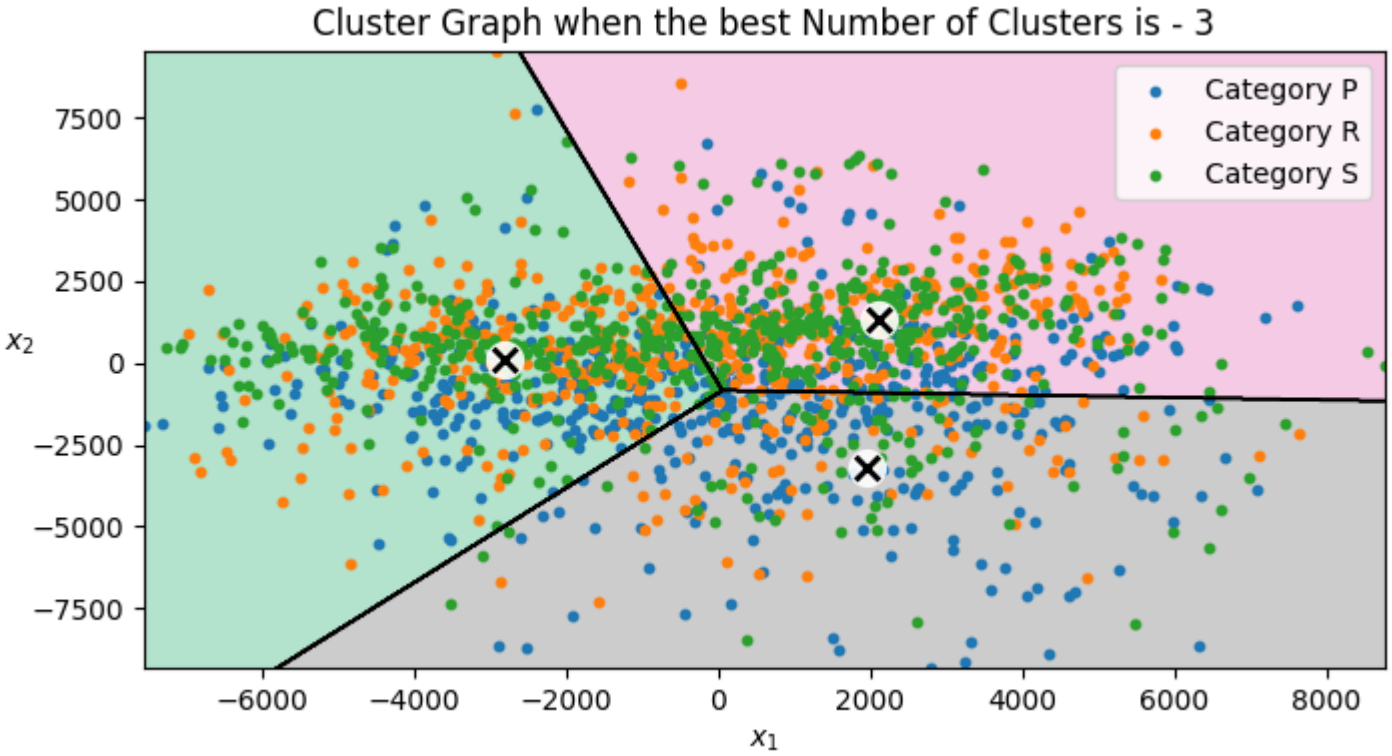

Update the centroids (initially randomly)
Label the instances
Category P
Category R
Category S
x_2
10000
5000
0
−5000
−10000
7500
2500
−2500
−7500





The above graphs check whether the data point change clusters with multiple iterations, for the optimum number of clusters. It can be seen that though the centroid location changes minutely, the data clustering does not change significantly.

In [37]:
```
# Plotting the best Clustering Graph
plt.figure(figsize=(8, 4))
plot_decision_boundaries(kmeans, X_reduced, categories)
plt.title(f"Cluster Graph when the best Number of Clusters is - {optimal_n_clusters}")
plt.show()
```



Question 6

Cluster the 360 images using EM.

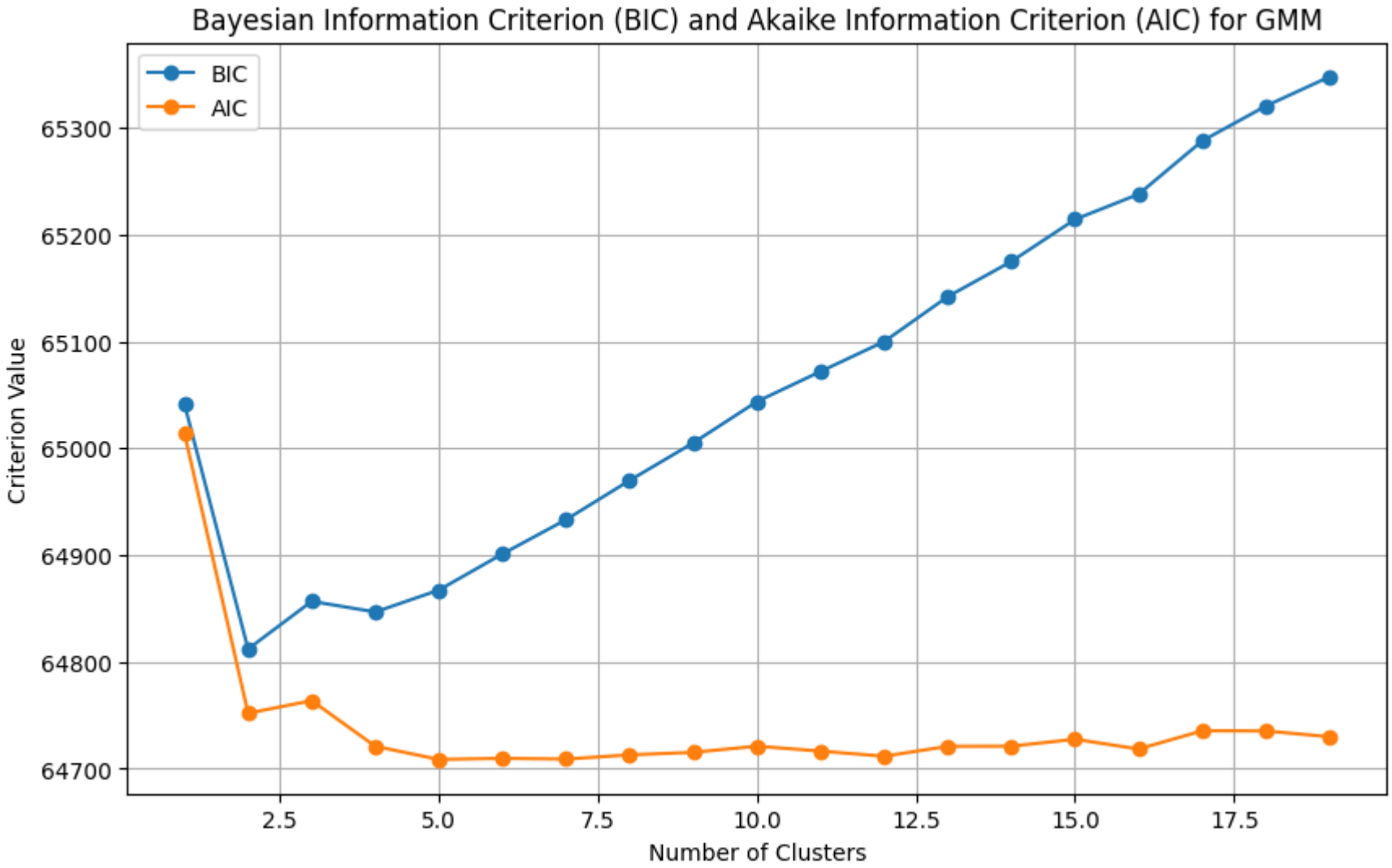
(A) Same as in the previous question, to speed up the algorithm, use PCA to reduce the dimensionality of the dataset to two. Determine the number of clusters using one of the techniques we discussed in class.

In [38]:
```
# Conducting PCA to reduce the dimensions to 2
pca = PCA(n_components=2)
X_reduced = pca.fit_transform(images.reshape(num_samples, height * width * channels))
print(f"Original Image shape = {images.shape}")
print(f"Reduced Size = {X_reduced.shape}")
```

```
Original Image shape = (1749, 100, 150, 3)
Reduced Size = (1749, 2)
```

In [39]:
```
#Determining the number of optimum Components for the Gaussian Mixture Model
n_components = range(1, 20)
bic_values = []
aic_values = []

# Covariance Type = "full"
for n in n_components:
    GMM = GaussianMixture(n_components=n, n_init=10, covariance_type="full", random_state=0)
    GMM.fit(X_reduced)
    bic_values.append(GMM.bic(X_reduced))
    aic_values.append(GMM.aic(X_reduced))

plt.figure(figsize=(10, 6))
plt.plot(n_components, bic_values, marker='o', label='BIC')
plt.plot(n_components, aic_values, marker='o', label='AIC')
plt.xlabel('Number of Clusters')
plt.ylabel('Criterion Value')
plt.title('Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC) for GMM')
plt.legend()
plt.grid()
plt.show()
```

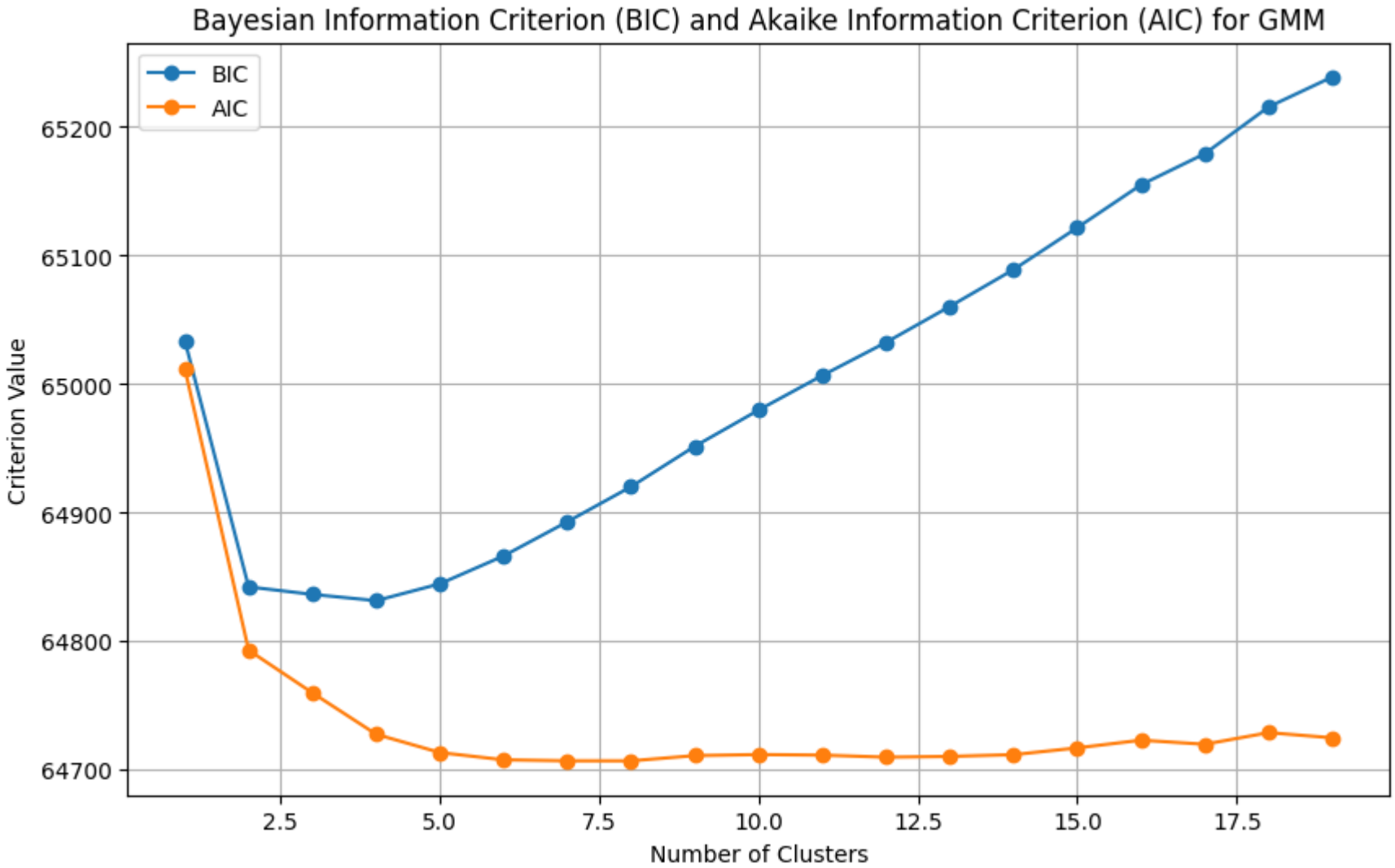
```
# Finding the number of Optimum Components is
optimal_bic_n_components = n_components[np.argmin(bic_values)]
optimal_aic_n_components = n_components[np.argmin(aic_values)]
print(f"The number of Best Components for Gaussian Mixture (with covariance_type = 'full'), According to BIC values is - {optim
print(f"The number of Best Components for Gaussian Mixture (with covariance_type = 'full'), According to AIC values is - {optim
```




```
The number of Best Components for Gaussian Mixture (with covariance_type = 'full'), According to BIC values is - 2
The number of Best Components for Gaussian Mixture (with covariance_type = 'full'), According to AIC values is - 2
```

In [40]:
```
# Covariance Type = "diag"

# Determining the number of optimum Components for the Gaussian Mixture Model
```

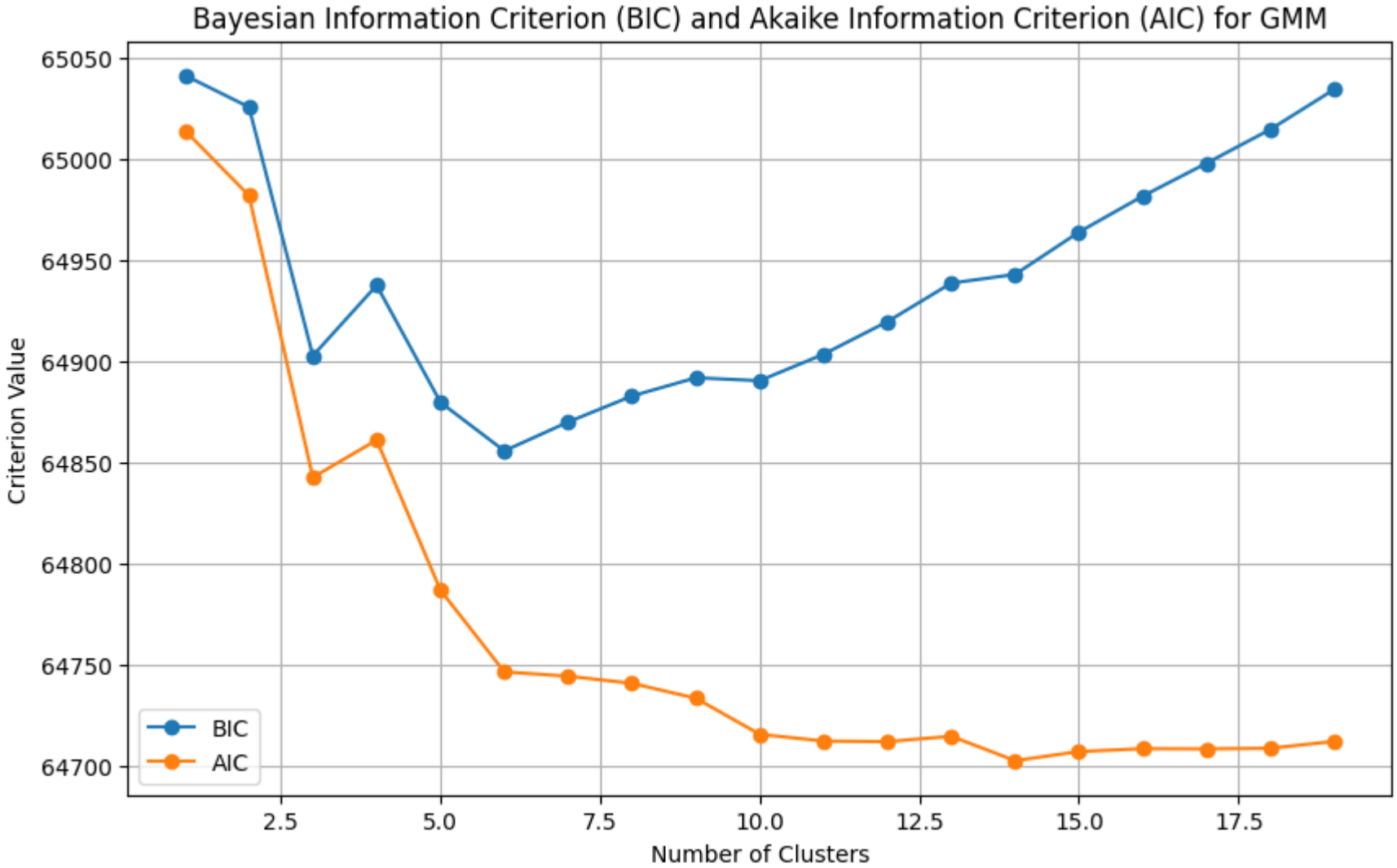
```
n_components = range(1, 20)
bic_values = []
aic_values = []

for n in n_components:
    GMM = GaussianMixture(n_components=n, n_init=10, covariance_type="diag", random_state=0)
    GMM.fit(X_reduced)
    bic_values.append(GMM.bic(X_reduced))
    aic_values.append(GMM.aic(X_reduced))

plt.figure(figsize=(10, 6))
plt.plot(n_components, bic_values, marker='o', label='BIC')
plt.plot(n_components, aic_values, marker='o', label='AIC')
plt.xlabel('Number of Clusters')
plt.ylabel('Criterion Value')
plt.title('Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC) for GMM')
plt.legend()
plt.grid()
plt.show()

# Finding the number of Optimum Components is
optimal_bic_n_components = n_components[np.argmin(bic_values)]
optimal_aic_n_components = n_components[np.argmin(aic_values)]
print(f"The number of Best Components for Gaussian Mixture (with covariance_type = 'diag'), According to BIC values is - {optim
print(f"The number of Best Components for Gaussian Mixture (with covariance_type = 'diag'), According to AIC values is - {optim
```




```
The number of Best Components for Gaussian Mixture (with covariance_type = 'diag'), According to BIC values is - 4
The number of Best Components for Gaussian Mixture (with covariance_type = 'diag'), According to AIC values is - 4
```

In [41]:
```
# Covariance Type = "tied"

# Determining the number of optimum Components for the Gaussian Mixture Model
n_components = range(1, 20)
bic_values = []
aic_values = []

for n in n_components:
    GMM = GaussianMixture(n_components=n, n_init=10, covariance_type="tied", random_state=0)
```

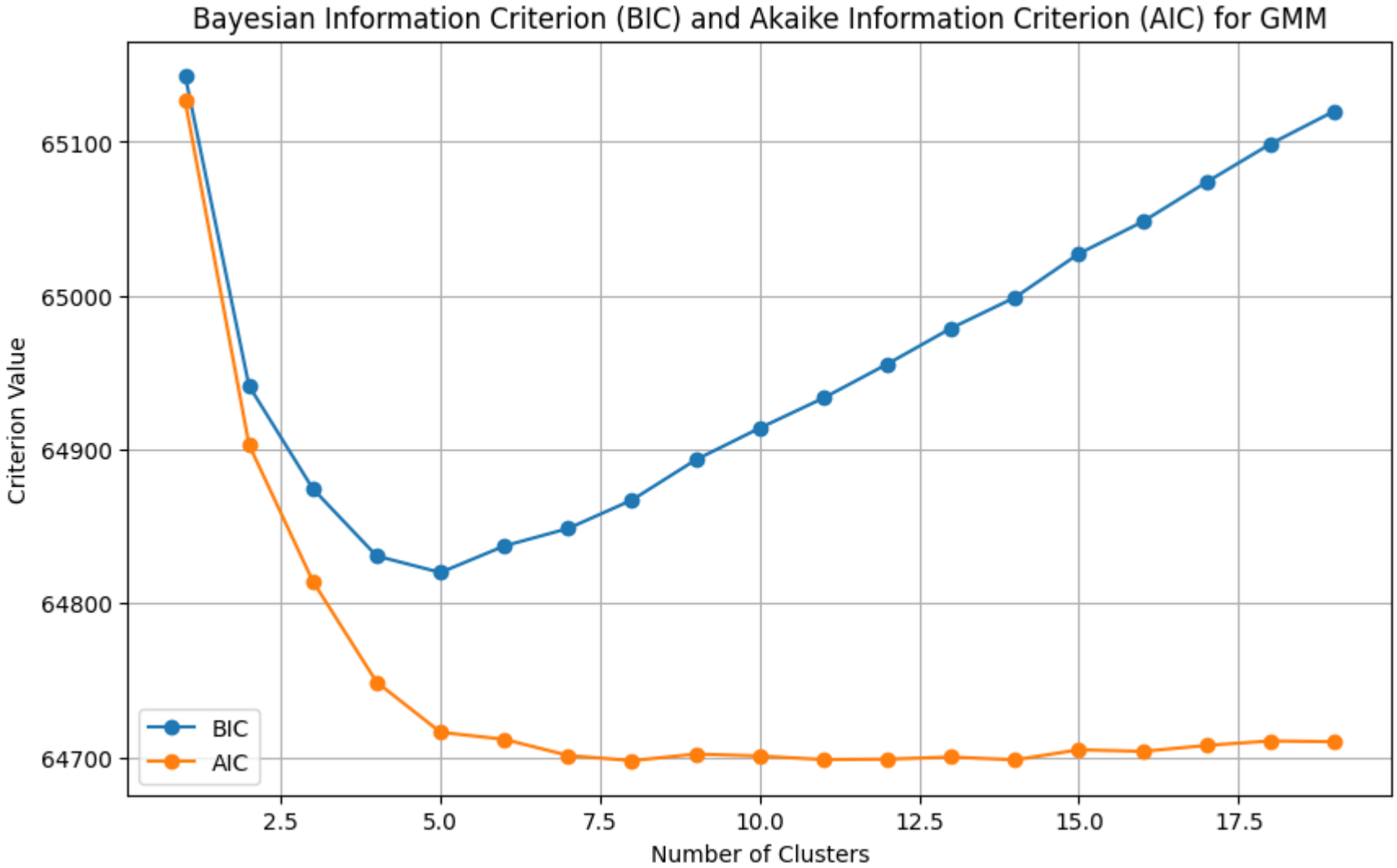
```
    GMM.fit(X_reduced)
    bic_values.append(GMM.bic(X_reduced))
    aic_values.append(GMM.aic(X_reduced))

plt.figure(figsize=(10, 6))
plt.plot(n_components, bic_values, marker='o', label='BIC')
plt.plot(n_components, aic_values, marker='o', label='AIC')
plt.xlabel('Number of Clusters')
plt.ylabel('Criterion Value')
plt.title('Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC) for GMM')
plt.legend()
plt.grid()
plt.show()

# Finding the number of Optimum Components is
optimal_bic_n_components = n_components[np.argmin(bic_values)]
optimal_aic_n_components = n_components[np.argmin(aic_values)]
print(f"The number of Best Components for Gaussian Mixture (with covariance_type = 'tied'), According to BIC values is - {optim
print(f"The number of Best Components for Gaussian Mixture (with covariance_type = 'tied'), According to AIC values is - {optim
```




```
The number of Best Components for Gaussian Mixture (with covariance_type = 'tied'), According to BIC values is - 6
The number of Best Components for Gaussian Mixture (with covariance_type = 'tied'), According to AIC values is - 6
```

In [42]:
```
# Covariance Type = "spherical"

# Determining the number of optimum Components for the Gaussian Mixture Model
n_components = range(1, 20)
bic_values = []
aic_values = []

for n in n_components:
    GMM = GaussianMixture(n_components=n, n_init=10, covariance_type="spherical", random_state=0)
```

```
    GMM.fit(X_reduced)
    bic_values.append(GMM.bic(X_reduced))
    aic_values.append(GMM.aic(X_reduced))

plt.figure(figsize=(10, 6))
plt.plot(n_components, bic_values, marker='o', label='BIC')
plt.plot(n_components, aic_values, marker='o', label='AIC')
plt.xlabel('Number of Clusters')
plt.ylabel('Criterion Value')
plt.title('Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC) for GMM')
plt.legend()
plt.grid()
plt.show()

# Finding the number of Optimum Components is
optimal_bic_n_components = n_components[np.argmin(bic_values)]
optimal_aic_n_components = n_components[np.argmin(aic_values)]
print(f"The number of Best Components for Gaussian Mixture (with covariance_type = 'spherical'), According to BIC values is - {
print(f"The number of Best Components for Gaussian Mixture (with covariance_type = 'spherical'), According to AIC values is - {
```




```
The number of Best Components for Gaussian Mixture (with covariance_type = 'spherical'), According to BIC values is - 5
The number of Best Components for Gaussian Mixture (with covariance_type = 'spherical'), According to AIC values is - 5
```

Based on the above BIC & AIC graphs, we can see that the optimum number of clusters obtained for different Gaussian Mixture models are -

1. with covariance_type="full" = 2
2. with covariance_type="diag" = 4
3. with covariance_type="tied" = 6

4. with covariance_type="spherical" = 5

In [43]:
```
# Based on the above graph, we choose the optimal number of clusters
gmm_optimal_n_clusters = 2  # Adjust this based on the plot

# Apply K-Means clustering with the optimal number of clusters
GMM = GaussianMixture(n_components=gmm_optimal_n_clusters, n_init=10, random_state=0)
GMM.fit(X_reduced)

print(f"Checking whether the Gaussian Mixture model Converges - {GMM.converged_}")
print(f"\nNumber of iternations that took the Gaussian Mixture took is - {GMM.n_iter_}")
print(f"\nThe weights established by the Gaussian Mixture model are - {GMM.weights_}")
print(f"\nThe means established by the Gaussian Mixture model are -\n {GMM.means_}")
print(f"\nThe covariances established by the Gaussian Mixture model are - \n {GMM.covariances_}")
```

```
Checking whether the Gaussian Mixture model Converges - True

Number of iternations that took the Gaussian Mixture took is - 21

The weights established by the Gaussian Mixture model are - [0.49966344 0.50033656]

The means established by the Gaussian Mixture model are -
 [[-1137.31712461   416.15895396]
 [ 1135.78706992  -415.59908729]]

The covariances established by the Gaussian Mixture model are -
 [[[7687085.43767643  773924.75414326]
  [ 773924.75414326 1412915.98081295]]

 [[7594077.96863172  171816.45032515]
  [ 171816.45032515 8860108.25194947]]]
```

(B) Visualize the clusters in a similar way to the visualization after line 28 here: https://github.com/ageron/handson-ml3/blob/main/09_unsupervised_learning.ipynbLinks to an external site., but color each dot based on the clusters it belongs to using the labels taken from the filename as in question 3 (P, R and S).

In [44]:
```
# Defining dunctions to plot Gaussian Mixtures
from matplotlib.colors import LogNorm
```

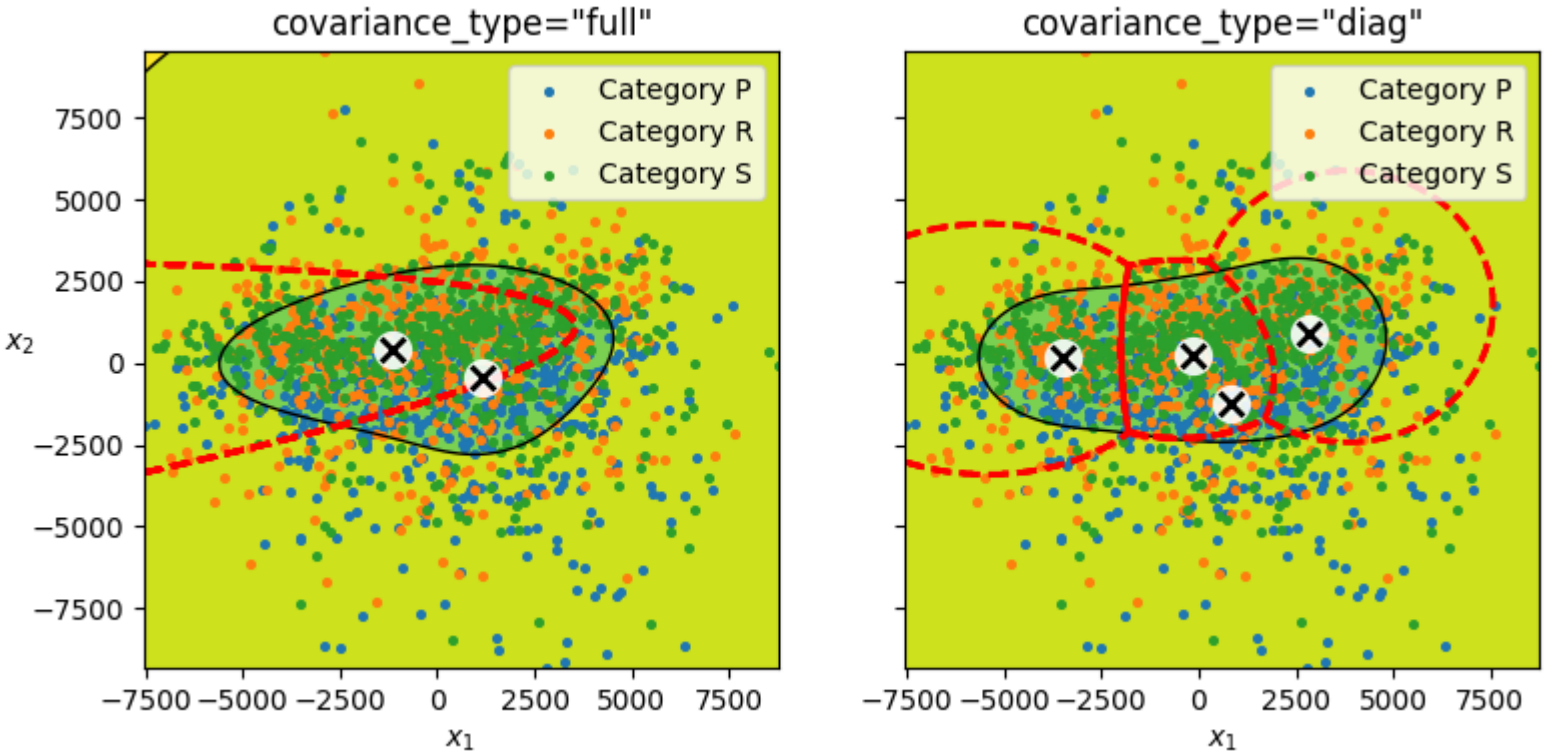
```
def plot_gaussian_mixture(clusterer, X, cat, resolution=1000, show_ylabels=True):
    mins = X.min(axis=0) - 0.1
    maxs = X.max(axis=0) + 0.1
    xx, yy = np.meshgrid(np.linspace(mins[0], maxs[0], resolution),
                         np.linspace(mins[1], maxs[1], resolution))
    Z = -clusterer.score_samples(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.contourf(xx, yy, Z,
                 norm=LogNorm(vmin=1.0, vmax=30.0),
                 levels=np.logspace(0, 2, 12))
    plt.contour(xx, yy, Z,
                norm=LogNorm(vmin=1.0, vmax=30.0),
                levels=np.logspace(0, 2, 12),
                linewidths=1, colors='k')

    Z = clusterer.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    plt.contour(xx, yy, Z,
                linewidths=2, colors='r', linestyles='dashed')

    for c in sorted(set(cat)):
        cat_img = X[np.array(cat)==c]
        plt.scatter(cat_img[:, 0], cat_img[:, 1], cmap = c, s=7, label = "Category "+c)
    plot_centroids(clusterer.means_, clusterer.weights_)
    plt.legend(loc='best', fontsize=10)

    plt.xlabel("$x_1$")
    if show_ylabels:
        plt.ylabel("$x_2$", rotation=0)
    else:
        plt.tick_params(labelleft=False)

plt.figure(figsize=(8, 4))

# # Plotting the graph
# plot_gaussian_mixture(GMM, X_reduced, categories)
# plt.show()
```

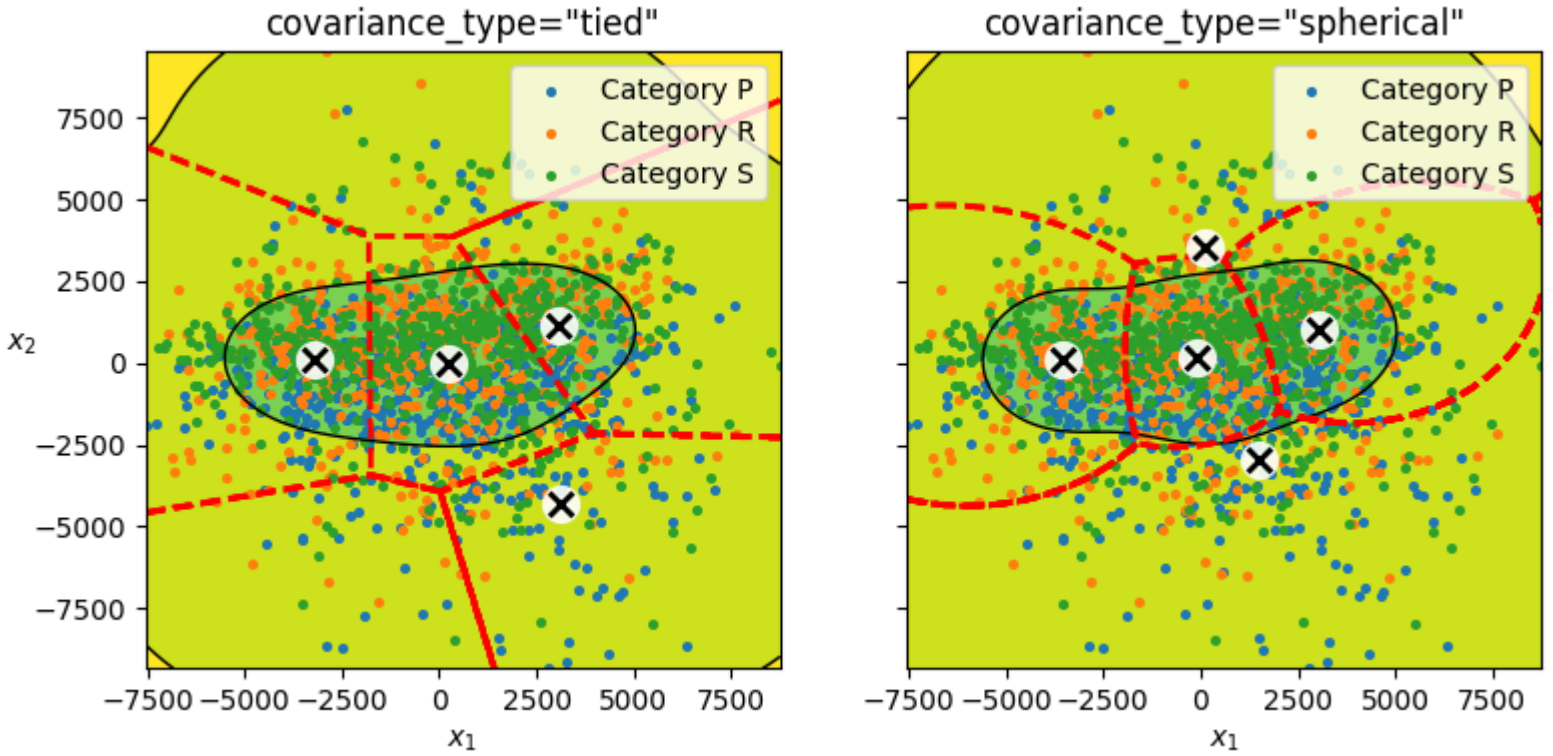
Out[44]: <Figure size 800x400 with 0 Axes>

<Figure size 800x400 with 0 Axes>

In [45]:
```
# Plotting Different GM covariance plots
gm_full = GaussianMixture(n_components=2, n_init=10,
```

```
                           covariance_type="full", random_state=42)
gm_tied = GaussianMixture(n_components=6, n_init=10,
                           covariance_type="tied", random_state=42)
gm_spherical = GaussianMixture(n_components=5, n_init=10,
                               covariance_type="spherical", random_state=42)
gm_diag = GaussianMixture(n_components=4, n_init=10,
                           covariance_type="diag", random_state=42)
gm_full.fit(X_reduced)
gm_tied.fit(X_reduced)
gm_spherical.fit(X_reduced)
gm_diag.fit(X_reduced)

def compare_gaussian_mixtures(gm1, gm2, X_reduced, categories):
    plt.figure(figsize=(9, 4))

    plt.subplot(121)
    plot_gaussian_mixture(gm1, X_reduced, categories)
    plt.title(f'covariance_type="{gm1.covariance_type}"')

    plt.subplot(122)
    plot_gaussian_mixture(gm2, X_reduced, categories, show_ylabels=False)
    plt.title(f'covariance_type="{gm2.covariance_type}"')

compare_gaussian_mixtures(gm_full, gm_diag, X_reduced, categories)
compare_gaussian_mixtures(gm_tied, gm_spherical, X_reduced, categories)

plt.show()
```


covariance_type="full"
covariance_type="diag"
Category P
Category R
Category S
7500
5000
2500
0
−2500
−5000
−7500
−7500 −5000 −2500 0 2500 5000 7500
x_1
x_2



1. From the above graph we can observe that the clustering outcome do not really change based on the 'covariance_type'.

- Though the number of clusters change, they do not produce a cluster that is distinctly better.
- Since the clusters are not better, I choose the covariance_type='tied' with optimum clusters=4 to generate samples in the following question.
- 'Covariance_type' = "tied" is selected as 4 optimum clusters looks better, in-terms of clustering data-points

2. Neither the EM or kmeans graph cluster well, cause 72% variance is lost by reducing the data to only 2 principle components in PCA.

(C) Use the model to generate 20 new rocks (using the sample() method), and visualize them in the original image space (since you used PCA, you will need to use its inverse_transform() method).

In [46]:
```
# Running the Gaussian Mixture with the Optimum Clusters
GMM = GaussianMixture(n_components=4, n_init=10, covariance_type="tied", random_state=0)
GMM.fit(X_reduced)

# Using the Sample method to generate 20 images
n_samples = 20
sample_20, sample_y = GMM.sample(n_samples)
sample_20
```

Out[46]:
```
array([[ 6874.32323957, -2261.73279077],
       [ 6955.4178847 ,   698.98283859],
       [ 1373.92569326,  2888.40675018],
       [ 3141.37738906,  6195.85083457],
       [ 3722.05607027,  2183.87432308],
       [ -805.72438344,   446.58772366],
       [-2566.75913285,  -626.30546805],
       [-1993.71884328, -1610.95739976],
       [-2171.01812054,   363.96199269],
       [-3153.49746379, -3295.10632586],
       [ -590.22749019,   404.90953291],
       [-2975.38189663,  -698.17585548],
       [ 1609.29033493,  3899.42144307],
       [  -86.24373293,  1406.20719399],
       [ 3124.43196885,  -740.2234579 ],
       [-2125.47759246,  3132.02100939],
       [ -837.95609544,   313.83167456],
       [ 1868.68297212, -1238.08653148],
       [ -974.52736879,  -891.98034178],
       [-1750.68437309, -4034.0589282 ]])
```

In [47]:
```
# Inversing the Sample Images extracted to the original size
X_recovered = pca.inverse_transform(sample_20)
X_recovered = X_recovered.reshape(n_samples, 100, 150, 3)
print(f"Recovered Size = {X_recovered.shape}")
```

```
Recovered Size = (20, 100, 150, 3)
```

In [48]:
```python
# Visualizing the Generated Images

num_images_to_plot = len(X_recovered)

# Determining the number of rows and columns dynamically
rows = int(math.ceil(num_images_to_plot / 2))
cols = 2

#Adjusting individual image sizes in the plot
image_size = 4

# Create a 10x2 grid for displaying images
fig, axes = plt.subplots(rows, cols, figsize=(10, rows*4))

# Plotting the reconstructed images generated by GMM
a = 0
for i in range(rows):
    for j in range(cols):
        if a < num_images_to_plot:
            reconstructed_image = X_recovered[a].reshape(100, 150, 3)
            axes[i,j].imshow(reconstructed_image.astype(int), extent=[0, image_size, 0, image_size])
            axes[i,j].set_title(f"Sampled fig {a+1} -")
            axes[i,j].axis("off")
            a = a + 1
        else:
            axes[i, j].axis("off")

plt.suptitle("Plotting 20 Generated Images")
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for inte
gers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for inte
gers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for inte
gers).
```

Plotting 20 Generated Images

Sampled fig 1 -

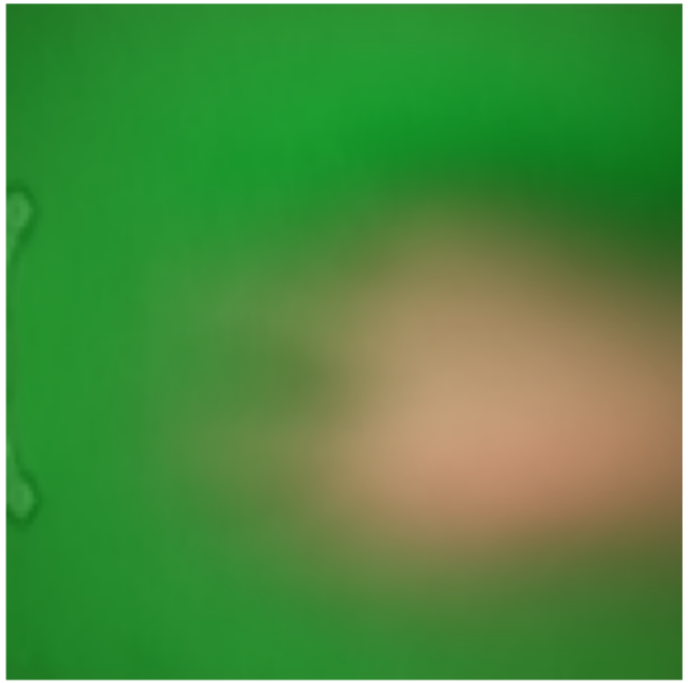
Sampled fig 2 -

Sampled fig 3 -

Sampled fig 4 -

Sampled fig 5 -

Sampled fig 6 -

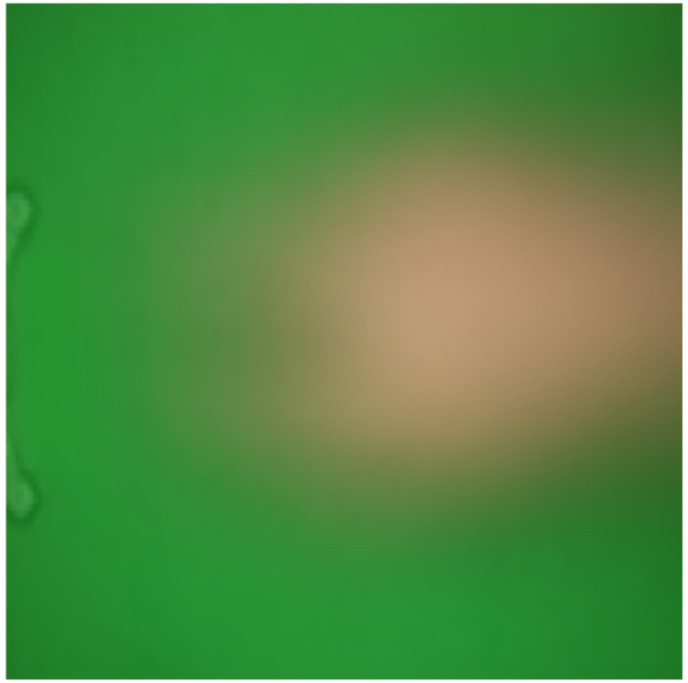


Sampled fig 7 -

Sampled fig 8 -

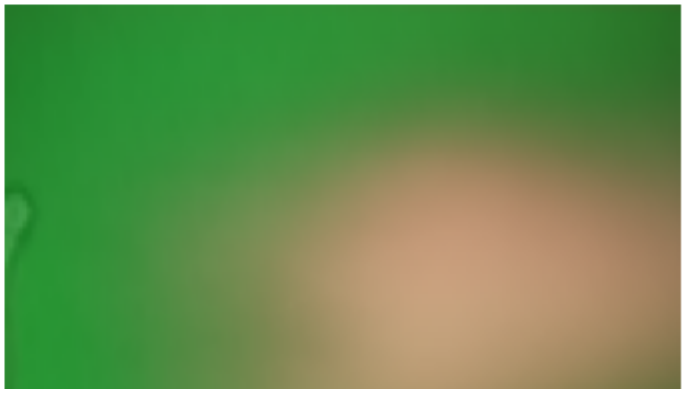
Sampled fig 9 -

Sampled fig 10 -



Sampled fig 11 -

Sampled fig 12 -



Sampled fig 13 -

Sampled fig 14 -

Sampled fig 15 -

Sampled fig 16 -

Sampled fig 17 -

Sampled fig 18 -

Sampled fig 19 -

Sampled fig 20 -

Question 7

In [51]:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Input, Dense, Conv2D, MaxPooling2D, Flatten,Dropout
from tensorflow.keras.models import Model
from tensorflow.keras import layers, models, optimizers
import time
from tensorflow.keras.layers import BatchNormalization

train_data_dir = '/content/extracted_images/train'
valid_data_dir = '/content/extracted_images/test'

batch_size = 32

# Custom data generator
class CustomDataGenerator(keras.utils.Sequence):
    def __init__(self, data_dir, batch_size, input_shape, shuffle=True):
        self.data_dir = data_dir
        self.batch_size = batch_size
        self.input_shape = input_shape
        self.shuffle = shuffle

        # Get a list of all image files in the data directory
        self.image_files = [f for f in os.listdir(data_dir) if f.endswith('.png')]

        self.indices = np.arange(len(self.image_files))
        if self.shuffle:
            np.random.shuffle(self.indices)
```

```
    def __len__(self):
        return int(np.ceil(len(self.indices) / self.batch_size))

    def __getitem__(self, index):
        batch_indices = self.indices[index * self.batch_size:(index + 1) * self.batch_size]

        X = np.zeros((len(batch_indices), *self.input_shape))
        y = []

        for i, idx in enumerate(batch_indices):
            image_path = os.path.join(self.data_dir, self.image_files[idx])
            image = tf.keras.preprocessing.image.load_img(image_path, target_size=self.input_shape)
            image = tf.keras.preprocessing.image.img_to_array(image)
            image = image / 255.0  # Normalize pixel values to [0, 1]
            X[i] = image

            # Extract the class label from the image filename (e.g., "I_image1.jpg" -> "I")
            class_label = self.image_files[idx].split('_')[0]
            y.append(class_label)

        # Convert class labels to unique integers
        unique_labels = list(set(y))
        print(unique_labels)
        label_mapping = {label: idx for idx, label in enumerate(unique_labels)}
        y = [label_mapping[label] for label in y]

        return X, keras.utils.to_categorical(y, num_classes=3)

    def on_epoch_end(self):
        if self.shuffle:
            np.random.shuffle(self.indices)

# Define input shape
input_shape = (200, 300, 3)  # Adjust to match your image dimensions

# Create custom data generators
train_generator = CustomDataGenerator(train_data_dir, batch_size, input_shape)
validation_generator = CustomDataGenerator(valid_data_dir, batch_size, input_shape, shuffle=False)
```

In [58]:

```
# Buildin the Sequential model

epochs = 20 #50
learning_rate = 0.0001 #0.0001
```

```python
# model = models.Sequential()
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape =[200, 300, 3]),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dense(100,activation="relu",kernel_regularizer=tf.keras.regularizers.l2(0.01)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dense(50,activation="relu",kernel_regularizer=tf.keras.regularizers.l2(0.01)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dense(8,activation="relu",name="intermediate_layer"),
    tf.keras.layers.Dense(3,activation="softmax")
])

# Compiling the model
model.compile(
    optimizer=optimizers.Adam(learning_rate),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

start_time = time.time()

# Training the model
history = model.fit(
    train_generator,
    epochs=epochs,
    steps_per_epoch=len(train_generator),
    validation_data=validation_generator,  # Include validation data
    validation_steps=len(validation_generator)  # Number of validation steps
)

# Record the end time
end_time = time.time()

# Evaluate the model on validation data
val_loss, val_accuracy = model.evaluate(validation_generator)
print(f'Validation Loss: {val_loss}, Validation Accuracy: {val_accuracy}')
```

```
['S', 'R', 'P']
Epoch 1/20
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
 1/55 [..............................] - ETA: 2:35 - loss: 4.1016 - accuracy: 0.4375['S', 'R', 'P']
 4/55 [=>.............................] - ETA: 2s - loss: 4.1256 - accuracy: 0.4531  ['S', 'R', 'P']
 5/55 [=>.............................] - ETA: 3s - loss: 4.0510 - accuracy: 0.4688['S', 'R', 'P']
 6/55 [==>............................] - ETA: 3s - loss: 4.0512 - accuracy: 0.4531['S', 'R', 'P']
 7/55 [==>............................] - ETA: 3s - loss: 3.9649 - accuracy: 0.4732['S', 'R', 'P']
 8/55 [===>...........................] - ETA: 3s - loss: 3.9310 - accuracy: 0.4883['S', 'R', 'P']
 9/55 [===>...........................] - ETA: 3s - loss: 3.9310 - accuracy: 0.4722['S', 'R', 'P']
10/55 [====>..........................] - ETA: 4s - loss: 3.9171 - accuracy: 0.4688['S', 'R', 'P']
11/55 [=====>.........................] - ETA: 4s - loss: 3.9244 - accuracy: 0.4631['S', 'R', 'P']
12/55 [=====>.........................] - ETA: 4s - loss: 3.8891 - accuracy: 0.4740['S', 'R', 'P']
13/55 [======>........................] - ETA: 3s - loss: 3.8661 - accuracy: 0.4784['S', 'R', 'P']
14/55 [======>........................] - ETA: 3s - loss: 3.8327 - accuracy: 0.4955['S', 'R', 'P']
15/55 [=======>.......................] - ETA: 3s - loss: 3.8099 - accuracy: 0.4938['S', 'R', 'P']
16/55 [=======>.......................] - ETA: 4s - loss: 3.7875 - accuracy: 0.5059['S', 'R', 'P']
17/55 [========>......................] - ETA: 4s - loss: 3.7696 - accuracy: 0.5129['S', 'R', 'P']
18/55 [========>......................] - ETA: 4s - loss: 3.7501 - accuracy: 0.5191['S', 'R', 'P']
19/55 [=========>.....................] - ETA: 5s - loss: 3.7402 - accuracy: 0.5230['S', 'R', 'P']
20/55 [=========>.....................] - ETA: 5s - loss: 3.7200 - accuracy: 0.5297['S', 'R', 'P']
21/55 [==========>....................] - ETA: 5s - loss: 3.6992 - accuracy: 0.5372['S', 'R', 'P']
22/55 [==========>....................] - ETA: 5s - loss: 3.6808 - accuracy: 0.5469['S', 'R', 'P']
23/55 [==========>....................] - ETA: 6s - loss: 3.6672 - accuracy: 0.5543['S', 'R', 'P']
24/55 [===========>...................] - ETA: 6s - loss: 3.6638 - accuracy: 0.5573['S', 'R', 'P']
25/55 [===========>...................] - ETA: 6s - loss: 3.6501 - accuracy: 0.5612['S', 'R', 'P']
26/55 [============>..................] - ETA: 6s - loss: 3.6367 - accuracy: 0.5661['S', 'R', 'P']
27/55 [============>..................] - ETA: 6s - loss: 3.6401 - accuracy: 0.5602['S', 'R', 'P']
28/55 [==============>................] - ETA: 6s - loss: 3.6277 - accuracy: 0.5647['S', 'R', 'P']
29/55 [==============>................] - ETA: 6s - loss: 3.6255 - accuracy: 0.5636['S', 'R', 'P']
30/55 [===============>...............] - ETA: 6s - loss: 3.6173 - accuracy: 0.5667['S', 'R', 'P']
31/55 [===============>...............] - ETA: 6s - loss: 3.6052 - accuracy: 0.5706['S', 'R', 'P']
32/55 [================>..............] - ETA: 6s - loss: 3.5988 - accuracy: 0.5781['S', 'R', 'P']
33/55 [================>..............] - ETA: 6s - loss: 3.5934 - accuracy: 0.5809['S', 'R', 'P']
34/55 [=================>.............] - ETA: 5s - loss: 3.5817 - accuracy: 0.5859['S', 'R', 'P']
35/55 [=================>.............] - ETA: 5s - loss: 3.5715 - accuracy: 0.5924['S', 'R', 'P']
36/55 [==================>............] - ETA: 5s - loss: 3.5597 - accuracy: 0.6021['S', 'R', 'P']
37/55 [==================>............] - ETA: 4s - loss: 3.5513 - accuracy: 0.6061['S', 'R', 'P']
38/55 [===================>...........] - ETA: 4s - loss: 3.5401 - accuracy: 0.6108['S', 'R', 'P']
39/55 [===================>...........] - ETA: 4s - loss: 3.5295 - accuracy: 0.6168['S', 'R', 'P']
40/55 [====================>..........] - ETA: 3s - loss: 3.5247 - accuracy: 0.6154['S', 'R', 'P']
41/55 [====================>..........] - ETA: 3s - loss: 3.5166 - accuracy: 0.6203['S', 'R', 'P']
```

```
42/55 [=====================>........] - ETA: 3s - loss: 3.5063 - accuracy: 0.6264['S', 'R', 'P']
43/55 [=====================>........] - ETA: 3s - loss: 3.4992 - accuracy: 0.6293['S', 'R', 'P']
44/55 [======================>.......] - ETA: 2s - loss: 3.4947 - accuracy: 0.6314['S', 'R', 'P']
45/55 [======================>.......] - ETA: 2s - loss: 3.4857 - accuracy: 0.6368['S', 'R', 'P']
46/55 [======================>.......] - ETA: 2s - loss: 3.4802 - accuracy: 0.6400['S', 'R', 'P']
47/55 [======================>.....] - ETA: 1s - loss: 3.4764 - accuracy: 0.6403['S', 'R', 'P']
48/55 [=======================>.....] - ETA: 1s - loss: 3.4683 - accuracy: 0.6459['S', 'R', 'P']
49/55 [=======================>.....] - ETA: 1s - loss: 3.4626 - accuracy: 0.6487['S', 'R', 'P']
50/55 [=======================>....] - ETA: 1s - loss: 3.4598 - accuracy: 0.6488['S', 'R', 'P']
51/55 [=======================>....] - ETA: 0s - loss: 3.4517 - accuracy: 0.6533['S', 'R', 'P']
52/55 [========================>...] - ETA: 0s - loss: 3.4476 - accuracy: 0.6552['S', 'R', 'P']
53/55 [========================>...] - ETA: 0s - loss: 3.4414 - accuracy: 0.6570['S', 'R', 'P']
54/55 [========================>..] - ETA: 0s - loss: 3.4360 - accuracy: 0.6599['S', 'R', 'P']
55/55 [==============================] - ETA: 0s - loss: 3.4310 - accuracy: 0.6609['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
55/55 [==============================] - 17s 259ms/step - loss: 3.4310 - accuracy: 0.6609 - val_loss: 3.5335 - val_accuracy: 0.4
852
Epoch 2/20
['S', 'R', 'P']
 1/55 [..............................] - ETA: 6s - loss: 3.0505 - accuracy: 0.9062['S', 'R', 'P']
 2/55 [>.............................] - ETA: 6s - loss: 3.1452 - accuracy: 0.8125['S', 'R', 'P']
 3/55 [>.............................] - ETA: 6s - loss: 3.1207 - accuracy: 0.8229['S', 'R', 'P']
 4/55 [=>............................] - ETA: 5s - loss: 3.0684 - accuracy: 0.8594['S', 'R', 'P']
 5/55 [=>............................] - ETA: 5s - loss: 3.0699 - accuracy: 0.8625['S', 'R', 'P']
 6/55 [==>...........................] - ETA: 5s - loss: 3.0464 - accuracy: 0.8802['S', 'R', 'P']
 7/55 [==>...........................] - ETA: 5s - loss: 3.0387 - accuracy: 0.8750['S', 'R', 'P']
 8/55 [===>..........................] - ETA: 5s - loss: 3.0305 - accuracy: 0.8711['S', 'R', 'P']
 9/55 [===>..........................] - ETA: 4s - loss: 3.0396 - accuracy: 0.8576['S', 'R', 'P']
10/55 [====>.........................] - ETA: 5s - loss: 3.0306 - accuracy: 0.8562['S', 'R', 'P']
11/55 [=====>........................] - ETA: 4s - loss: 3.0254 - accuracy: 0.8608['S', 'R', 'P']
12/55 [=====>........................] - ETA: 4s - loss: 3.0266 - accuracy: 0.8594['S', 'R', 'P']
```

```
13/55 [======>.....................] - ETA: 4s - loss: 3.0195 - accuracy: 0.8630['S', 'R', 'P']
14/55 [======>.....................] - ETA: 4s - loss: 3.0183 - accuracy: 0.8571['S', 'R', 'P']
15/55 [=======>.....................] - ETA: 4s - loss: 3.0183 - accuracy: 0.8542['S', 'R', 'P']
16/55 [=======>.....................] - ETA: 4s - loss: 3.0150 - accuracy: 0.8516['S', 'R', 'P']
17/55 [========>.....................] - ETA: 4s - loss: 3.0193 - accuracy: 0.8499['S', 'R', 'P']
18/55 [========>.....................] - ETA: 3s - loss: 3.0131 - accuracy: 0.8531['S', 'R', 'P']
19/55 [=========>.....................] - ETA: 3s - loss: 3.0165 - accuracy: 0.8476['S', 'R', 'P']
20/55 [=========>.....................] - ETA: 3s - loss: 3.0091 - accuracy: 0.8490['S', 'R', 'P']
21/55 [==========>.....................] - ETA: 3s - loss: 2.9991 - accuracy: 0.8548['S', 'R', 'P']
22/55 [==========>.....................] - ETA: 3s - loss: 2.9902 - accuracy: 0.8615['S', 'R', 'P']
23/55 [==========>.....................] - ETA: 3s - loss: 2.9860 - accuracy: 0.8593['S', 'R', 'P']
24/55 [===========>.....................] - ETA: 3s - loss: 2.9843 - accuracy: 0.8600['S', 'R', 'P']
25/55 [===========>.....................] - ETA: 3s - loss: 2.9767 - accuracy: 0.8644['S', 'R', 'P']
26/55 [============>.....................] - ETA: 3s - loss: 2.9753 - accuracy: 0.8624['S', 'R', 'P']
27/55 [============>.....................] - ETA: 3s - loss: 2.9743 - accuracy: 0.8581['S', 'R', 'P']
28/55 [=============>.....................] - ETA: 2s - loss: 2.9715 - accuracy: 0.8576['S', 'R', 'P']
29/55 [=============>.....................] - ETA: 2s - loss: 2.9673 - accuracy: 0.8593['S', 'R', 'P']
30/55 [==============>.....................] - ETA: 2s - loss: 2.9646 - accuracy: 0.8599['S', 'R', 'P']
31/55 [==============>.....................] - ETA: 2s - loss: 2.9625 - accuracy: 0.8593['S', 'R', 'P']
32/55 [===============>.....................] - ETA: 2s - loss: 2.9616 - accuracy: 0.8598['S', 'R', 'P']
33/55 [================>.....................] - ETA: 2s - loss: 2.9575 - accuracy: 0.8603['S', 'R', 'P']
34/55 [================>.....................] - ETA: 2s - loss: 2.9530 - accuracy: 0.8607['S', 'R', 'P']
35/55 [=================>.....................] - ETA: 2s - loss: 2.9514 - accuracy: 0.8584['S', 'R', 'P']
36/55 [=================>.....................] - ETA: 2s - loss: 2.9457 - accuracy: 0.8606['S', 'R', 'P']
37/55 [==================>.....................] - ETA: 1s - loss: 2.9434 - accuracy: 0.8619['S', 'R', 'P']
38/55 [==================>.....................] - ETA: 1s - loss: 2.9358 - accuracy: 0.8647['S', 'R', 'P']
39/55 [===================>.....................] - ETA: 1s - loss: 2.9344 - accuracy: 0.8650['S', 'R', 'P']
40/55 [===================>.....................] - ETA: 1s - loss: 2.9321 - accuracy: 0.8660['S', 'R', 'P']
41/55 [====================>.....................] - ETA: 1s - loss: 2.9311 - accuracy: 0.8640['S', 'R', 'P']
42/55 [====================>.....................] - ETA: 1s - loss: 2.9297 - accuracy: 0.8642['S', 'R', 'P']
43/55 [=====================>.....................] - ETA: 1s - loss: 2.9273 - accuracy: 0.8645['S', 'R', 'P']
44/55 [=====================>.....................] - ETA: 1s - loss: 2.9221 - accuracy: 0.8669['S', 'R', 'P']
45/55 [=====================>.....................] - ETA: 1s - loss: 2.9178 - accuracy: 0.8691['S', 'R', 'P']
46/55 [======================>.....................] - ETA: 0s - loss: 2.9152 - accuracy: 0.8700['S', 'R', 'P']
47/55 [======================>.....................] - ETA: 0s - loss: 2.9117 - accuracy: 0.8707['S', 'R', 'P']
48/55 [======================>.....................] - ETA: 0s - loss: 2.9066 - accuracy: 0.8721['S', 'R', 'P']
49/55 [======================>.....................] - ETA: 0s - loss: 2.9025 - accuracy: 0.8735['S', 'R', 'P']
50/55 [=======================>.....................] - ETA: 0s - loss: 2.8976 - accuracy: 0.8754['S', 'R', 'P']
51/55 [=======================>.....................] - ETA: 0s - loss: 2.8953 - accuracy: 0.8748['S', 'R', 'P']
52/55 [========================>.....................] - ETA: 0s - loss: 2.8920 - accuracy: 0.8760['S', 'R', 'P']
53/55 [========================>.....................] - ETA: 0s - loss: 2.8881 - accuracy: 0.8772['S', 'R', 'P']
54/55 [========================>.....................] - ETA: 0s - loss: 2.8849 - accuracy: 0.8771['S', 'R', 'P']
55/55 [==============================] - ETA: 0s - loss: 2.8813 - accuracy: 0.8776['S', 'R', 'P']
['S', 'R', 'P']
```

```
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
55/55 [==============================] - 9s 161ms/step - loss: 2.8813 - accuracy: 0.8776 - val_loss: 3.2515 - val_accuracy: 0.57
86
Epoch 3/20
['S', 'R', 'P']
 1/55 [..............................] - ETA: 7s - loss: 2.5954 - accuracy: 0.9375['S', 'R', 'P']
 2/55 [>.............................] - ETA: 6s - loss: 2.6678 - accuracy: 0.9062['S', 'R', 'P']
 3/55 [>.............................] - ETA: 6s - loss: 2.6492 - accuracy: 0.9271['S', 'R', 'P']
 4/55 [=>............................] - ETA: 6s - loss: 2.6888 - accuracy: 0.8906['S', 'R', 'P']
 5/55 [=>............................] - ETA: 6s - loss: 2.6887 - accuracy: 0.9000['S', 'R', 'P']
 6/55 [==>...........................] - ETA: 5s - loss: 2.6553 - accuracy: 0.9167['S', 'R', 'P']
 7/55 [==>...........................] - ETA: 5s - loss: 2.6658 - accuracy: 0.9062['S', 'R', 'P']
 8/55 [===>..........................] - ETA: 5s - loss: 2.6641 - accuracy: 0.9062['S', 'R', 'P']
 9/55 [===>..........................] - ETA: 5s - loss: 2.6708 - accuracy: 0.8958['S', 'R', 'P']
10/55 [====>.........................] - ETA: 5s - loss: 2.6715 - accuracy: 0.8938['S', 'R', 'P']
11/55 [=====>........................] - ETA: 5s - loss: 2.6699 - accuracy: 0.8892['S', 'R', 'P']
12/55 [=====>........................] - ETA: 5s - loss: 2.6631 - accuracy: 0.8932['S', 'R', 'P']
13/55 [======>.......................] - ETA: 4s - loss: 2.6546 - accuracy: 0.8990['S', 'R', 'P']
14/55 [======>.......................] - ETA: 4s - loss: 2.6468 - accuracy: 0.9062['S', 'R', 'P']
15/55 [=======>......................] - ETA: 4s - loss: 2.6419 - accuracy: 0.9125['S', 'R', 'P']
16/55 [=======>......................] - ETA: 4s - loss: 2.6343 - accuracy: 0.9160['S', 'R', 'P']
17/55 [========>.....................] - ETA: 4s - loss: 2.6432 - accuracy: 0.9081['S', 'R', 'P']
18/55 [========>.....................] - ETA: 4s - loss: 2.6421 - accuracy: 0.9062['S', 'R', 'P']
19/55 [=========>....................] - ETA: 4s - loss: 2.6425 - accuracy: 0.9079['S', 'R', 'P']
20/55 [=========>....................] - ETA: 4s - loss: 2.6458 - accuracy: 0.9031['S', 'R', 'P']
21/55 [==========>...................] - ETA: 3s - loss: 2.6433 - accuracy: 0.9048['S', 'R', 'P']
22/55 [==========>...................] - ETA: 3s - loss: 2.6417 - accuracy: 0.9062['S', 'R', 'P']
23/55 [==========>...................] - ETA: 3s - loss: 2.6382 - accuracy: 0.9076['S', 'R', 'P']
24/55 [===========>..................] - ETA: 3s - loss: 2.6333 - accuracy: 0.9089['S', 'R', 'P']
25/55 [===========>..................] - ETA: 3s - loss: 2.6280 - accuracy: 0.9125['S', 'R', 'P']
26/55 [============>.................] - ETA: 3s - loss: 2.6294 - accuracy: 0.9099['S', 'R', 'P']
27/55 [============>.................] - ETA: 3s - loss: 2.6247 - accuracy: 0.9109['S', 'R', 'P']
28/55 [=============>................] - ETA: 3s - loss: 2.6178 - accuracy: 0.9141['S', 'R', 'P']
```

```
29/55 [==============>..............] - ETA: 2s - loss: 2.6155 - accuracy: 0.9138['S', 'R', 'P']
30/55 [==============>..............] - ETA: 2s - loss: 2.6137 - accuracy: 0.9146['S', 'R', 'P']
31/55 [==============>..............] - ETA: 2s - loss: 2.6115 - accuracy: 0.9153['S', 'R', 'P']
32/55 [===============>.............] - ETA: 2s - loss: 2.6109 - accuracy: 0.9141['S', 'R', 'P']
33/55 [================>............] - ETA: 2s - loss: 2.6079 - accuracy: 0.9138['S', 'R', 'P']
34/55 [================>............] - ETA: 2s - loss: 2.6064 - accuracy: 0.9154['S', 'R', 'P']
35/55 [=================>...........] - ETA: 2s - loss: 2.6008 - accuracy: 0.9170['S', 'R', 'P']
36/55 [=================>...........] - ETA: 2s - loss: 2.5972 - accuracy: 0.9175['S', 'R', 'P']
37/55 [==================>..........] - ETA: 2s - loss: 2.5925 - accuracy: 0.9172['S', 'R', 'P']
38/55 [==================>..........] - ETA: 1s - loss: 2.5893 - accuracy: 0.9186['S', 'R', 'P']
39/55 [===================>.........] - ETA: 1s - loss: 2.5854 - accuracy: 0.9191['S', 'R', 'P']
40/55 [===================>.........] - ETA: 1s - loss: 2.5837 - accuracy: 0.9187['S', 'R', 'P']
41/55 [====================>........] - ETA: 1s - loss: 2.5818 - accuracy: 0.9184['S', 'R', 'P']
42/55 [====================>........] - ETA: 1s - loss: 2.5837 - accuracy: 0.9144['S', 'R', 'P']
43/55 [=====================>.......] - ETA: 1s - loss: 2.5821 - accuracy: 0.9142['S', 'R', 'P']
44/55 [=====================>.......] - ETA: 1s - loss: 2.5801 - accuracy: 0.9148['S', 'R', 'P']
45/55 [======================>......] - ETA: 1s - loss: 2.5805 - accuracy: 0.9132['S', 'R', 'P']
46/55 [======================>......] - ETA: 1s - loss: 2.5759 - accuracy: 0.9144['S', 'R', 'P']
47/55 [======================>......] - ETA: 0s - loss: 2.5715 - accuracy: 0.9162['S', 'R', 'P']
48/55 [=======================>.....] - ETA: 0s - loss: 2.5674 - accuracy: 0.9173['S', 'R', 'P']
49/55 [=======================>.....] - ETA: 0s - loss: 2.5680 - accuracy: 0.9171['S', 'R', 'P']
50/55 [========================>....] - ETA: 0s - loss: 2.5635 - accuracy: 0.9181['S', 'R', 'P']
51/55 [========================>....] - ETA: 0s - loss: 2.5619 - accuracy: 0.9179['S', 'R', 'P']
52/55 [=========================>...] - ETA: 0s - loss: 2.5587 - accuracy: 0.9183['S', 'R', 'P']
53/55 [=========================>...] - ETA: 0s - loss: 2.5543 - accuracy: 0.9199['S', 'R', 'P']
54/55 [==========================>..] - ETA: 0s - loss: 2.5525 - accuracy: 0.9202['S', 'R', 'P']
55/55 [==========================>..] - ETA: 0s - loss: 2.5492 - accuracy: 0.9200['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
55/55 [==========================>..] - 8s 142ms/step - loss: 2.5492 - accuracy: 0.9200 - val_loss: 2.9346 - val_accuracy: 0.60
36
Epoch 4/20
['S', 'R', 'P']
```

```
 1/55 [..............................] - ETA: 9s - loss: 2.3168 - accuracy: 1.0000['S', 'R', 'P']
 2/55 [>.............................] - ETA: 7s - loss: 2.3960 - accuracy: 0.9219['S', 'R', 'P']
 3/55 [>.............................] - ETA: 7s - loss: 2.3661 - accuracy: 0.9375['S', 'R', 'P']
 4/55 [=>............................] - ETA: 6s - loss: 2.3958 - accuracy: 0.9219['S', 'R', 'P']
 5/55 [=>............................] - ETA: 6s - loss: 2.3761 - accuracy: 0.9312['S', 'R', 'P']
 6/55 [==>...........................] - ETA: 6s - loss: 2.3574 - accuracy: 0.9427['S', 'R', 'P']
 7/55 [==>...........................] - ETA: 6s - loss: 2.3437 - accuracy: 0.9509['S', 'R', 'P']
 8/55 [===>..........................] - ETA: 6s - loss: 2.3466 - accuracy: 0.9492['S', 'R', 'P']
 9/55 [===>..........................] - ETA: 6s - loss: 2.3557 - accuracy: 0.9444['S', 'R', 'P']
10/55 [====>.........................] - ETA: 6s - loss: 2.3522 - accuracy: 0.9469['S', 'R', 'P']
11/55 [=====>........................] - ETA: 6s - loss: 2.3566 - accuracy: 0.9432['S', 'R', 'P']
12/55 [=====>........................] - ETA: 6s - loss: 2.3450 - accuracy: 0.9479['S', 'R', 'P']
13/55 [======>.......................] - ETA: 6s - loss: 2.3407 - accuracy: 0.9495['S', 'R', 'P']
14/55 [======>.......................] - ETA: 6s - loss: 2.3434 - accuracy: 0.9464['S', 'R', 'P']
15/55 [=======>......................] - ETA: 5s - loss: 2.3414 - accuracy: 0.9438['S', 'R', 'P']
16/55 [=======>......................] - ETA: 5s - loss: 2.3424 - accuracy: 0.9434['S', 'R', 'P']
17/55 [========>.....................] - ETA: 5s - loss: 2.3400 - accuracy: 0.9449['S', 'R', 'P']
18/55 [========>.....................] - ETA: 5s - loss: 2.3412 - accuracy: 0.9444['S', 'R', 'P']
19/55 [=========>....................] - ETA: 5s - loss: 2.3425 - accuracy: 0.9424['S', 'R', 'P']
20/55 [=========>....................] - ETA: 5s - loss: 2.3455 - accuracy: 0.9375['S', 'R', 'P']
21/55 [==========>...................] - ETA: 4s - loss: 2.3430 - accuracy: 0.9390['S', 'R', 'P']
22/55 [==========>...................] - ETA: 4s - loss: 2.3388 - accuracy: 0.9389['S', 'R', 'P']
23/55 [==========>...................] - ETA: 4s - loss: 2.3360 - accuracy: 0.9389['S', 'R', 'P']
24/55 [===========>..................] - ETA: 4s - loss: 2.3355 - accuracy: 0.9375['S', 'R', 'P']
25/55 [===========>..................] - ETA: 4s - loss: 2.3334 - accuracy: 0.9350['S', 'R', 'P']
26/55 [============>.................] - ETA: 4s - loss: 2.3339 - accuracy: 0.9351['S', 'R', 'P']
27/55 [============>.................] - ETA: 4s - loss: 2.3318 - accuracy: 0.9363['S', 'R', 'P']
28/55 [=============>................] - ETA: 3s - loss: 2.3272 - accuracy: 0.9375['S', 'R', 'P']
29/55 [=============>................] - ETA: 3s - loss: 2.3236 - accuracy: 0.9397['S', 'R', 'P']
30/55 [==============>...............] - ETA: 3s - loss: 2.3194 - accuracy: 0.9406['S', 'R', 'P']
31/55 [==============>...............] - ETA: 3s - loss: 2.3165 - accuracy: 0.9405['S', 'R', 'P']
32/55 [===============>..............] - ETA: 3s - loss: 2.3153 - accuracy: 0.9395['S', 'R', 'P']
33/55 [================>.............] - ETA: 3s - loss: 2.3192 - accuracy: 0.9366['S', 'R', 'P']
34/55 [================>.............] - ETA: 3s - loss: 2.3162 - accuracy: 0.9357['S', 'R', 'P']
35/55 [=================>............] - ETA: 2s - loss: 2.3145 - accuracy: 0.9348['S', 'R', 'P']
36/55 [=================>............] - ETA: 2s - loss: 2.3146 - accuracy: 0.9332['S', 'R', 'P']
37/55 [==================>...........] - ETA: 2s - loss: 2.3109 - accuracy: 0.9341['S', 'R', 'P']
38/55 [==================>...........] - ETA: 2s - loss: 2.3074 - accuracy: 0.9334['S', 'R', 'P']
39/55 [===================>..........] - ETA: 2s - loss: 2.3053 - accuracy: 0.9329['S', 'R', 'P']
40/55 [===================>..........] - ETA: 2s - loss: 2.3046 - accuracy: 0.9322['S', 'R', 'P']
41/55 [====================>.........] - ETA: 2s - loss: 2.3037 - accuracy: 0.9316['S', 'R', 'P']
42/55 [====================>.........] - ETA: 1s - loss: 2.3024 - accuracy: 0.9317['S', 'R', 'P']
43/55 [=====================>........] - ETA: 1s - loss: 2.3008 - accuracy: 0.9304['S', 'R', 'P']
44/55 [=====================>........] - ETA: 1s - loss: 2.2992 - accuracy: 0.9306['S', 'R', 'P']
```

```
45/55 [=======================>......] - ETA: 1s - loss: 2.2971 - accuracy: 0.9300['S', 'R', 'P']
46/55 [=======================>......] - ETA: 1s - loss: 2.2928 - accuracy: 0.9316['S', 'R', 'P']
47/55 [=======================>.....] - ETA: 1s - loss: 2.2900 - accuracy: 0.9324['S', 'R', 'P']
48/55 [========================>.....] - ETA: 0s - loss: 2.2874 - accuracy: 0.9325['S', 'R', 'P']
49/55 [========================>.....] - ETA: 0s - loss: 2.2861 - accuracy: 0.9326['S', 'R', 'P']
50/55 [========================>....] - ETA: 0s - loss: 2.2824 - accuracy: 0.9333['S', 'R', 'P']
51/55 [========================>....] - ETA: 0s - loss: 2.2800 - accuracy: 0.9328['S', 'R', 'P']
52/55 [=========================>...] - ETA: 0s - loss: 2.2772 - accuracy: 0.9341['S', 'R', 'P']
53/55 [=========================>...] - ETA: 0s - loss: 2.2761 - accuracy: 0.9335['S', 'R', 'P']
54/55 [=========================>..] - ETA: 0s - loss: 2.2730 - accuracy: 0.9342['S', 'R', 'P']
55/55 [==========================] - ETA: 0s - loss: 2.2710 - accuracy: 0.9337['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
55/55 [==========================] - 9s 162ms/step - loss: 2.2710 - accuracy: 0.9337 - val_loss: 2.5356 - val_accuracy: 0.63
78
Epoch 5/20
['S', 'R', 'P']
 1/55 [..............................] - ETA: 6s - loss: 2.1050 - accuracy: 0.9688['S', 'R', 'P']
 2/55 [>.............................] - ETA: 6s - loss: 2.1080 - accuracy: 0.9688['S', 'R', 'P']
 3/55 [>.............................] - ETA: 5s - loss: 2.1179 - accuracy: 0.9583['S', 'R', 'P']
 4/55 [=>............................] - ETA: 5s - loss: 2.1222 - accuracy: 0.9375['S', 'R', 'P']
 5/55 [=>............................] - ETA: 5s - loss: 2.1017 - accuracy: 0.9500['S', 'R', 'P']
 6/55 [==>...........................] - ETA: 5s - loss: 2.1102 - accuracy: 0.9531['S', 'R', 'P']
 7/55 [==>...........................] - ETA: 5s - loss: 2.0985 - accuracy: 0.9509['S', 'R', 'P']
 8/55 [===>..........................] - ETA: 5s - loss: 2.0868 - accuracy: 0.9570['S', 'R', 'P']
 9/55 [===>..........................] - ETA: 5s - loss: 2.0964 - accuracy: 0.9549['S', 'R', 'P']
10/55 [====>.........................] - ETA: 5s - loss: 2.1016 - accuracy: 0.9500['S', 'R', 'P']
11/55 [=====>........................] - ETA: 4s - loss: 2.0956 - accuracy: 0.9517['S', 'R', 'P']
12/55 [=====>........................] - ETA: 4s - loss: 2.0921 - accuracy: 0.9531['S', 'R', 'P']
13/55 [======>.......................] - ETA: 4s - loss: 2.0892 - accuracy: 0.9519['S', 'R', 'P']
14/55 [======>.......................] - ETA: 4s - loss: 2.0840 - accuracy: 0.9531['S', 'R', 'P']
15/55 [=======>......................] - ETA: 4s - loss: 2.0783 - accuracy: 0.9563['S', 'R', 'P']
16/55 [=======>......................] - ETA: 4s - loss: 2.0749 - accuracy: 0.9590['S', 'R', 'P']
```

```
17/55 [========>.....................] - ETA: 4s - loss: 2.0697 - accuracy: 0.9614['S', 'R', 'P']
18/55 [========>.....................] - ETA: 4s - loss: 2.0685 - accuracy: 0.9601['S', 'R', 'P']
19/55 [=========>....................] - ETA: 4s - loss: 2.0652 - accuracy: 0.9622['S', 'R', 'P']
20/55 [=========>....................] - ETA: 3s - loss: 2.0604 - accuracy: 0.9625['S', 'R', 'P']
21/55 [==========>...................] - ETA: 3s - loss: 2.0559 - accuracy: 0.9628['S', 'R', 'P']
22/55 [==========>...................] - ETA: 3s - loss: 2.0565 - accuracy: 0.9616['S', 'R', 'P']
23/55 [==========>...................] - ETA: 3s - loss: 2.0538 - accuracy: 0.9633['S', 'R', 'P']
24/55 [===========>..................] - ETA: 3s - loss: 2.0486 - accuracy: 0.9648['S', 'R', 'P']
25/55 [===========>..................] - ETA: 3s - loss: 2.0461 - accuracy: 0.9650['S', 'R', 'P']
26/55 [============>.................] - ETA: 3s - loss: 2.0438 - accuracy: 0.9651['S', 'R', 'P']
27/55 [============>.................] - ETA: 3s - loss: 2.0409 - accuracy: 0.9653['S', 'R', 'P']
28/55 [=============>................] - ETA: 2s - loss: 2.0422 - accuracy: 0.9643['S', 'R', 'P']
29/55 [=============>................] - ETA: 2s - loss: 2.0458 - accuracy: 0.9623['S', 'R', 'P']
30/55 [==============>...............] - ETA: 2s - loss: 2.0431 - accuracy: 0.9615['S', 'R', 'P']
31/55 [==============>...............] - ETA: 2s - loss: 2.0412 - accuracy: 0.9607['S', 'R', 'P']
32/55 [===============>..............] - ETA: 2s - loss: 2.0398 - accuracy: 0.9600['S', 'R', 'P']
33/55 [================>.............] - ETA: 2s - loss: 2.0344 - accuracy: 0.9612['S', 'R', 'P']
34/55 [================>.............] - ETA: 2s - loss: 2.0360 - accuracy: 0.9596['S', 'R', 'P']
35/55 [=================>............] - ETA: 2s - loss: 2.0422 - accuracy: 0.9554['S', 'R', 'P']
36/55 [=================>............] - ETA: 2s - loss: 2.0421 - accuracy: 0.9549['S', 'R', 'P']
37/55 [==================>...........] - ETA: 1s - loss: 2.0406 - accuracy: 0.9552['S', 'R', 'P']
38/55 [==================>...........] - ETA: 1s - loss: 2.0388 - accuracy: 0.9548['S', 'R', 'P']
39/55 [===================>..........] - ETA: 1s - loss: 2.0379 - accuracy: 0.9535['S', 'R', 'P']
40/55 [===================>..........] - ETA: 1s - loss: 2.0373 - accuracy: 0.9531['S', 'R', 'P']
41/55 [====================>.........] - ETA: 1s - loss: 2.0340 - accuracy: 0.9543['S', 'R', 'P']
42/55 [====================>.........] - ETA: 1s - loss: 2.0304 - accuracy: 0.9546['S', 'R', 'P']
43/55 [=====================>........] - ETA: 1s - loss: 2.0264 - accuracy: 0.9557['S', 'R', 'P']
44/55 [=====================>........] - ETA: 1s - loss: 2.0236 - accuracy: 0.9553['S', 'R', 'P']
45/55 [======================>.......] - ETA: 1s - loss: 2.0231 - accuracy: 0.9549['S', 'R', 'P']
46/55 [======================>.......] - ETA: 0s - loss: 2.0205 - accuracy: 0.9552['S', 'R', 'P']
47/55 [=======================>......] - ETA: 0s - loss: 2.0182 - accuracy: 0.9555['S', 'R', 'P']
48/55 [=======================>......] - ETA: 0s - loss: 2.0177 - accuracy: 0.9551['S', 'R', 'P']
49/55 [========================>.....] - ETA: 0s - loss: 2.0155 - accuracy: 0.9554['S', 'R', 'P']
50/55 [========================>.....] - ETA: 0s - loss: 2.0138 - accuracy: 0.9550['S', 'R', 'P']
51/55 [=========================>....] - ETA: 0s - loss: 2.0138 - accuracy: 0.9550['S', 'R', 'P']
52/55 [=========================>....] - ETA: 0s - loss: 2.0144 - accuracy: 0.9534['S', 'R', 'P']
53/55 [==========================>...] - ETA: 0s - loss: 2.0109 - accuracy: 0.9543['S', 'R', 'P']
54/55 [==========================>...] - ETA: 0s - loss: 2.0089 - accuracy: 0.9546['S', 'R', 'P']
55/55 [============================>.] - ETA: 0s - loss: 2.0070 - accuracy: 0.9543['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
```

```
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
55/55 [==============================] - 8s 144ms/step - loss: 2.0070 - accuracy: 0.9543 - val_loss: 2.2035 - val_accuracy: 0.77
68
Epoch 6/20
['S', 'R', 'P']
 1/55 [..............................] - ETA: 9s - loss: 1.8650 - accuracy: 0.9688['S', 'R', 'P']
 2/55 [>.............................] - ETA: 7s - loss: 1.9153 - accuracy: 0.9375['S', 'R', 'P']
 3/55 [>.............................] - ETA: 7s - loss: 1.9160 - accuracy: 0.9479['S', 'R', 'P']
 4/55 [=>............................] - ETA: 6s - loss: 1.9049 - accuracy: 0.9453['S', 'R', 'P']
 5/55 [=>............................] - ETA: 6s - loss: 1.8908 - accuracy: 0.9530['S', 'R', 'P']
 6/55 [==>...........................] - ETA: 6s - loss: 1.8765 - accuracy: 0.9558['S', 'R', 'P']
 7/55 [==>...........................] - ETA: 6s - loss: 1.8892 - accuracy: 0.9484['S', 'R', 'P']
 8/55 [===>..........................] - ETA: 6s - loss: 1.8830 - accuracy: 0.9510['S', 'R', 'P']
 9/55 [===>..........................] - ETA: 6s - loss: 1.8673 - accuracy: 0.9567['S', 'R', 'P']
10/55 [====>.........................] - ETA: 6s - loss: 1.8642 - accuracy: 0.9579['S', 'R', 'P']
11/55 [=====>........................] - ETA: 5s - loss: 1.8607 - accuracy: 0.9560['S', 'R', 'P']
12/55 [=====>........................] - ETA: 5s - loss: 1.8553 - accuracy: 0.9598['S', 'R', 'P']
13/55 [======>.......................] - ETA: 5s - loss: 1.8528 - accuracy: 0.9605['S', 'R', 'P']
14/55 [======>.......................] - ETA: 5s - loss: 1.8456 - accuracy: 0.9611['S', 'R', 'P']
15/55 [=======>......................] - ETA: 5s - loss: 1.8458 - accuracy: 0.9595['S', 'R', 'P']
16/55 [=======>......................] - ETA: 4s - loss: 1.8399 - accuracy: 0.9621['S', 'R', 'P']
17/55 [========>.....................] - ETA: 4s - loss: 1.8363 - accuracy: 0.9644['S', 'R', 'P']
18/55 [========>.....................] - ETA: 4s - loss: 1.8359 - accuracy: 0.9628['S', 'R', 'P']
19/55 [=========>....................] - ETA: 4s - loss: 1.8408 - accuracy: 0.9598['S', 'R', 'P']
20/55 [=========>....................] - ETA: 4s - loss: 1.8359 - accuracy: 0.9618['S', 'R', 'P']
21/55 [==========>...................] - ETA: 4s - loss: 1.8365 - accuracy: 0.9592['S', 'R', 'P']
22/55 [==========>...................] - ETA: 4s - loss: 1.8380 - accuracy: 0.9582['S', 'R', 'P']
23/55 [==========>...................] - ETA: 3s - loss: 1.8391 - accuracy: 0.9572['S', 'R', 'P']
24/55 [===========>..................] - ETA: 3s - loss: 1.8378 - accuracy: 0.9577['S', 'R', 'P']
25/55 [===========>..................] - ETA: 3s - loss: 1.8344 - accuracy: 0.9594['S', 'R', 'P']
26/55 [============>.................] - ETA: 3s - loss: 1.8367 - accuracy: 0.9562['S', 'R', 'P']
27/55 [============>.................] - ETA: 3s - loss: 1.8416 - accuracy: 0.9519['S', 'R', 'P']
28/55 [=============>................] - ETA: 3s - loss: 1.8463 - accuracy: 0.9480['S', 'R', 'P']
29/55 [=============>................] - ETA: 3s - loss: 1.8445 - accuracy: 0.9487['S', 'R', 'P']
30/55 [==============>...............] - ETA: 2s - loss: 1.8393 - accuracy: 0.9505['S', 'R', 'P']
31/55 [==============>...............] - ETA: 2s - loss: 1.8425 - accuracy: 0.9470['S', 'R', 'P']
32/55 [===============>..............] - ETA: 2s - loss: 1.8406 - accuracy: 0.9467['S', 'R', 'P']
```

```
33/55 [=================>............] - ETA: 2s - loss: 1.8402 - accuracy: 0.9474['S', 'R', 'P']
34/55 [=================>............] - ETA: 2s - loss: 1.8378 - accuracy: 0.9480['S', 'R', 'P']
35/55 [==================>...........] - ETA: 2s - loss: 1.8358 - accuracy: 0.9486['S', 'R', 'P']
36/55 [==================>...........] - ETA: 2s - loss: 1.8370 - accuracy: 0.9457['S', 'R', 'P']
37/55 [==================>...........] - ETA: 2s - loss: 1.8356 - accuracy: 0.9463['S', 'R', 'P']
38/55 [===================>..........] - ETA: 1s - loss: 1.8332 - accuracy: 0.9461['S', 'R', 'P']
39/55 [===================>..........] - ETA: 1s - loss: 1.8335 - accuracy: 0.9442['S', 'R', 'P']
40/55 [====================>.........] - ETA: 1s - loss: 1.8308 - accuracy: 0.9456['S', 'R', 'P']
41/55 [====================>.........] - ETA: 1s - loss: 1.8281 - accuracy: 0.9462['S', 'R', 'P']
42/55 [====================>.........] - ETA: 1s - loss: 1.8268 - accuracy: 0.9460['S', 'R', 'P']
43/55 [=====================>........] - ETA: 1s - loss: 1.8266 - accuracy: 0.9458['S', 'R', 'P']
44/55 [=====================>........] - ETA: 1s - loss: 1.8254 - accuracy: 0.9463['S', 'R', 'P']
45/55 [======================>.......] - ETA: 1s - loss: 1.8240 - accuracy: 0.9461['S', 'R', 'P']
46/55 [======================>.......] - ETA: 1s - loss: 1.8223 - accuracy: 0.9466['S', 'R', 'P']
47/55 [======================>.......] - ETA: 0s - loss: 1.8190 - accuracy: 0.9478['S', 'R', 'P']
48/55 [=======================>......] - ETA: 0s - loss: 1.8163 - accuracy: 0.9489['S', 'R', 'P']
49/55 [=======================>......] - ETA: 0s - loss: 1.8151 - accuracy: 0.9480['S', 'R', 'P']
50/55 [========================>.....] - ETA: 0s - loss: 1.8153 - accuracy: 0.9478['S', 'R', 'P']
51/55 [========================>.....] - ETA: 0s - loss: 1.8129 - accuracy: 0.9482['S', 'R', 'P']
52/55 [========================>.....] - ETA: 0s - loss: 1.8112 - accuracy: 0.9486['S', 'R', 'P']
53/55 [=========================>....] - ETA: 0s - loss: 1.8096 - accuracy: 0.9484['S', 'R', 'P']
54/55 [=========================>....] - ETA: 0s - loss: 1.8068 - accuracy: 0.9493['S', 'R', 'P']
55/55 [==========================>...] - ETA: 0s - loss: 1.8065 - accuracy: 0.9480['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
55/55 [==========================>...] - 8s 141ms/step - loss: 1.8065 - accuracy: 0.9480 - val_loss: 1.8976 - val_accuracy: 0.88
15
Epoch 7/20
['S', 'R', 'P']
 1/55 [..............................] - ETA: 6s - loss: 1.8881 - accuracy: 0.9062['S', 'R', 'P']
 2/55 [>.............................] - ETA: 5s - loss: 1.7567 - accuracy: 0.9531['S', 'R', 'P']
 3/55 [>.............................] - ETA: 6s - loss: 1.7325 - accuracy: 0.9479['S', 'R', 'P']
 4/55 [=>............................] - ETA: 5s - loss: 1.7139 - accuracy: 0.9531['S', 'R', 'P']
```

```
 5/55 [=>............................] - ETA: 5s - loss: 1.6990 - accuracy: 0.9563['S', 'R', 'P']
 6/55 [==>...........................] - ETA: 5s - loss: 1.6837 - accuracy: 0.9635['S', 'R', 'P']
 7/55 [==>...........................] - ETA: 5s - loss: 1.6663 - accuracy: 0.9688['S', 'R', 'P']
 8/55 [===>..........................] - ETA: 5s - loss: 1.6719 - accuracy: 0.9648['S', 'R', 'P']
 9/55 [===>..........................] - ETA: 5s - loss: 1.6655 - accuracy: 0.9688['S', 'R', 'P']
10/55 [====>.........................] - ETA: 5s - loss: 1.6731 - accuracy: 0.9688['S', 'R', 'P']
11/55 [=====>........................] - ETA: 5s - loss: 1.6753 - accuracy: 0.9688['S', 'R', 'P']
12/55 [=====>........................] - ETA: 4s - loss: 1.6712 - accuracy: 0.9661['S', 'R', 'P']
13/55 [======>.......................] - ETA: 4s - loss: 1.6747 - accuracy: 0.9663['S', 'R', 'P']
14/55 [======>.......................] - ETA: 4s - loss: 1.6726 - accuracy: 0.9665['S', 'R', 'P']
15/55 [=======>......................] - ETA: 4s - loss: 1.6669 - accuracy: 0.9688['S', 'R', 'P']
16/55 [=======>......................] - ETA: 4s - loss: 1.6590 - accuracy: 0.9707['S', 'R', 'P']
17/55 [========>.....................] - ETA: 4s - loss: 1.6530 - accuracy: 0.9724['S', 'R', 'P']
18/55 [========>.....................] - ETA: 4s - loss: 1.6512 - accuracy: 0.9722['S', 'R', 'P']
19/55 [=========>....................] - ETA: 4s - loss: 1.6468 - accuracy: 0.9737['S', 'R', 'P']
20/55 [=========>....................] - ETA: 4s - loss: 1.6452 - accuracy: 0.9734['S', 'R', 'P']
21/55 [==========>...................] - ETA: 3s - loss: 1.6411 - accuracy: 0.9747['S', 'R', 'P']
22/55 [==========>...................] - ETA: 3s - loss: 1.6412 - accuracy: 0.9744['S', 'R', 'P']
23/55 [==========>...................] - ETA: 3s - loss: 1.6365 - accuracy: 0.9755['S', 'R', 'P']
24/55 [===========>..................] - ETA: 3s - loss: 1.6409 - accuracy: 0.9727['S', 'R', 'P']
25/55 [===========>..................] - ETA: 3s - loss: 1.6377 - accuracy: 0.9737['S', 'R', 'P']
26/55 [============>.................] - ETA: 3s - loss: 1.6366 - accuracy: 0.9736['S', 'R', 'P']
27/55 [============>.................] - ETA: 3s - loss: 1.6418 - accuracy: 0.9722['S', 'R', 'P']
28/55 [=============>................] - ETA: 3s - loss: 1.6425 - accuracy: 0.9710['S', 'R', 'P']
29/55 [=============>................] - ETA: 2s - loss: 1.6395 - accuracy: 0.9709['S', 'R', 'P']
30/55 [==============>...............] - ETA: 2s - loss: 1.6378 - accuracy: 0.9715['S', 'R', 'P']
31/55 [==============>...............] - ETA: 2s - loss: 1.6377 - accuracy: 0.9704['S', 'R', 'P']
32/55 [===============>..............] - ETA: 2s - loss: 1.6357 - accuracy: 0.9704['S', 'R', 'P']
33/55 [===============>..............] - ETA: 2s - loss: 1.6342 - accuracy: 0.9703['S', 'R', 'P']
34/55 [================>.............] - ETA: 2s - loss: 1.6335 - accuracy: 0.9703['S', 'R', 'P']
35/55 [================>.............] - ETA: 2s - loss: 1.6303 - accuracy: 0.9711['S', 'R', 'P']
36/55 [=================>............] - ETA: 2s - loss: 1.6289 - accuracy: 0.9702['S', 'R', 'P']
37/55 [=================>............] - ETA: 2s - loss: 1.6255 - accuracy: 0.9710['S', 'R', 'P']
38/55 [==================>...........] - ETA: 2s - loss: 1.6252 - accuracy: 0.9710['S', 'R', 'P']
39/55 [==================>...........] - ETA: 2s - loss: 1.6341 - accuracy: 0.9669['S', 'R', 'P']
40/55 [===================>..........] - ETA: 1s - loss: 1.6322 - accuracy: 0.9669['S', 'R', 'P']
41/55 [===================>..........] - ETA: 1s - loss: 1.6322 - accuracy: 0.9662['S', 'R', 'P']
42/55 [====================>.........] - ETA: 1s - loss: 1.6308 - accuracy: 0.9670['S', 'R', 'P']
43/55 [====================>.........] - ETA: 1s - loss: 1.6306 - accuracy: 0.9670['S', 'R', 'P']
44/55 [=====================>........] - ETA: 1s - loss: 1.6275 - accuracy: 0.9678['S', 'R', 'P']
45/55 [=====================>........] - ETA: 1s - loss: 1.6245 - accuracy: 0.9685['S', 'R', 'P']
46/55 [======================>.......] - ETA: 1s - loss: 1.6251 - accuracy: 0.9678['S', 'R', 'P']
47/55 [======================>.......] - ETA: 1s - loss: 1.6229 - accuracy: 0.9678['S', 'R', 'P']
48/55 [=======================>......] - ETA: 0s - loss: 1.6211 - accuracy: 0.9685['S', 'R', 'P']
```

```
49/55 [=========================>....] - ETA: 0s - loss: 1.6215 - accuracy: 0.9679['S', 'R', 'P']
50/55 [=========================>...] - ETA: 0s - loss: 1.6192 - accuracy: 0.9679['S', 'R', 'P']
51/55 [=========================>...] - ETA: 0s - loss: 1.6172 - accuracy: 0.9685['S', 'R', 'P']
52/55 [=========================>..] - ETA: 0s - loss: 1.6155 - accuracy: 0.9685['S', 'R', 'P']
53/55 [=========================>..] - ETA: 0s - loss: 1.6159 - accuracy: 0.9668['S', 'R', 'P']
54/55 [=========================>.] - ETA: 0s - loss: 1.6154 - accuracy: 0.9662['S', 'R', 'P']
55/55 [==============================] - ETA: 0s - loss: 1.6145 - accuracy: 0.9657['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
55/55 [==============================] - 10s 184ms/step - loss: 1.6145 - accuracy: 0.9657 - val_loss: 1.6813 - val_accuracy: 0.8
793
Epoch 8/20
['S', 'R', 'P']
 1/55 [..............................] - ETA: 6s - loss: 1.5225 - accuracy: 0.9375['S', 'R', 'P']
 2/55 [>.............................] - ETA: 6s - loss: 1.5291 - accuracy: 0.9531['S', 'R', 'P']
 3/55 [>.............................] - ETA: 6s - loss: 1.4978 - accuracy: 0.9688['S', 'R', 'P']
 4/55 [=>............................] - ETA: 5s - loss: 1.5193 - accuracy: 0.9531['S', 'R', 'P']
 5/55 [=>............................] - ETA: 5s - loss: 1.5511 - accuracy: 0.9438['S', 'R', 'P']
 6/55 [==>...........................] - ETA: 5s - loss: 1.5264 - accuracy: 0.9531['S', 'R', 'P']
 7/55 [==>...........................] - ETA: 5s - loss: 1.5315 - accuracy: 0.9509['S', 'R', 'P']
 8/55 [===>..........................] - ETA: 5s - loss: 1.5215 - accuracy: 0.9570['S', 'R', 'P']
 9/55 [===>..........................] - ETA: 5s - loss: 1.5238 - accuracy: 0.9514['S', 'R', 'P']
10/55 [====>.........................] - ETA: 5s - loss: 1.5265 - accuracy: 0.9469['S', 'R', 'P']
11/55 [=====>........................] - ETA: 4s - loss: 1.5199 - accuracy: 0.9501['S', 'R', 'P']
12/55 [=====>........................] - ETA: 4s - loss: 1.5170 - accuracy: 0.9517['S', 'R', 'P']
13/55 [======>.......................] - ETA: 4s - loss: 1.5200 - accuracy: 0.9481['S', 'R', 'P']
14/55 [======>.......................] - ETA: 4s - loss: 1.5254 - accuracy: 0.9451['S', 'R', 'P']
15/55 [=======>......................] - ETA: 4s - loss: 1.5264 - accuracy: 0.9446['S', 'R', 'P']
16/55 [=======>......................] - ETA: 4s - loss: 1.5196 - accuracy: 0.9481['S', 'R', 'P']
17/55 [========>.....................] - ETA: 4s - loss: 1.5160 - accuracy: 0.9493['S', 'R', 'P']
18/55 [========>.....................] - ETA: 4s - loss: 1.5131 - accuracy: 0.9504['S', 'R', 'P']
19/55 [=========>....................] - ETA: 3s - loss: 1.5078 - accuracy: 0.9531['S', 'R', 'P']
20/55 [=========>....................] - ETA: 3s - loss: 1.5061 - accuracy: 0.9523['S', 'R', 'P']
```

```
21/55 [==========>...................] - ETA: 3s - loss: 1.5066 - accuracy: 0.9516['S', 'R', 'P']
22/55 [==========>...................] - ETA: 3s - loss: 1.5025 - accuracy: 0.9538['S', 'R', 'P']
23/55 [==========>...................] - ETA: 3s - loss: 1.5026 - accuracy: 0.9531['S', 'R', 'P']
24/55 [===========>..................] - ETA: 3s - loss: 1.5017 - accuracy: 0.9524['S', 'R', 'P']
25/55 [===========>..................] - ETA: 3s - loss: 1.5043 - accuracy: 0.9518['S', 'R', 'P']
26/55 [============>.................] - ETA: 3s - loss: 1.5007 - accuracy: 0.9525['S', 'R', 'P']
27/55 [============>.................] - ETA: 3s - loss: 1.4968 - accuracy: 0.9543['S', 'R', 'P']
28/55 [=============>................] - ETA: 2s - loss: 1.4962 - accuracy: 0.9548['S', 'R', 'P']
29/55 [=============>................] - ETA: 2s - loss: 1.4940 - accuracy: 0.9553['S', 'R', 'P']
30/55 [==============>...............] - ETA: 2s - loss: 1.4905 - accuracy: 0.9568['S', 'R', 'P']
31/55 [==============>...............] - ETA: 2s - loss: 1.4905 - accuracy: 0.9562['S', 'R', 'P']
32/55 [===============>..............] - ETA: 2s - loss: 1.4870 - accuracy: 0.9576['S', 'R', 'P']
33/55 [===============>..............] - ETA: 2s - loss: 1.4872 - accuracy: 0.9560['S', 'R', 'P']
34/55 [================>.............] - ETA: 2s - loss: 1.4855 - accuracy: 0.9564['S', 'R', 'P']
35/55 [================>.............] - ETA: 2s - loss: 1.4844 - accuracy: 0.9567['S', 'R', 'P']
36/55 [=================>............] - ETA: 2s - loss: 1.4815 - accuracy: 0.9579['S', 'R', 'P']
37/55 [=================>............] - ETA: 1s - loss: 1.4792 - accuracy: 0.9591['S', 'R', 'P']
38/55 [=================>............] - ETA: 1s - loss: 1.4777 - accuracy: 0.9593['S', 'R', 'P']
39/55 [==================>...........] - ETA: 1s - loss: 1.4812 - accuracy: 0.9580['S', 'R', 'P']
40/55 [==================>...........] - ETA: 1s - loss: 1.4782 - accuracy: 0.9590['S', 'R', 'P']
41/55 [===================>..........] - ETA: 1s - loss: 1.4789 - accuracy: 0.9577['S', 'R', 'P']
42/55 [===================>..........] - ETA: 1s - loss: 1.4762 - accuracy: 0.9580['S', 'R', 'P']
43/55 [====================>.........] - ETA: 1s - loss: 1.4739 - accuracy: 0.9590['S', 'R', 'P']
44/55 [====================>.........] - ETA: 1s - loss: 1.4708 - accuracy: 0.9599['S', 'R', 'P']
45/55 [====================>.........] - ETA: 1s - loss: 1.4709 - accuracy: 0.9587['S', 'R', 'P']
46/55 [=====================>........] - ETA: 0s - loss: 1.4681 - accuracy: 0.9596['S', 'R', 'P']
47/55 [=====================>........] - ETA: 0s - loss: 1.4665 - accuracy: 0.9598['S', 'R', 'P']
48/55 [======================>.......] - ETA: 0s - loss: 1.4644 - accuracy: 0.9607['S', 'R', 'P']
49/55 [======================>.......] - ETA: 0s - loss: 1.4663 - accuracy: 0.9595['S', 'R', 'P']
50/55 [======================>.......] - ETA: 0s - loss: 1.4660 - accuracy: 0.9597['S', 'R', 'P']
51/55 [=======================>......] - ETA: 0s - loss: 1.4646 - accuracy: 0.9599['S', 'R', 'P']
52/55 [=======================>......] - ETA: 0s - loss: 1.4620 - accuracy: 0.9607['S', 'R', 'P']
53/55 [========================>.....] - ETA: 0s - loss: 1.4597 - accuracy: 0.9608['S', 'R', 'P']
54/55 [========================>.....] - ETA: 0s - loss: 1.4582 - accuracy: 0.9610['S', 'R', 'P']
55/55 [==============================] - ETA: 0s - loss: 1.4569 - accuracy: 0.9617['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
```

```
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
55/55 [==============================] - 8s 140ms/step - loss: 1.4569 - accuracy: 0.9617 - val_loss: 1.5022 - val_accuracy: 0.92
71
Epoch 9/20
['S', 'R', 'P']
 1/55 [..............................] - ETA: 6s - loss: 1.3541 - accuracy: 0.9688['S', 'R', 'P']
 2/55 [>.............................] - ETA: 6s - loss: 1.3280 - accuracy: 0.9844['S', 'R', 'P']
 3/55 [>.............................] - ETA: 6s - loss: 1.3491 - accuracy: 0.9688['S', 'R', 'P']
 4/55 [=>............................] - ETA: 5s - loss: 1.3609 - accuracy: 0.9688['S', 'R', 'P']
 5/55 [=>............................] - ETA: 5s - loss: 1.3468 - accuracy: 0.9750['S', 'R', 'P']
 6/55 [==>...........................] - ETA: 5s - loss: 1.3423 - accuracy: 0.9740['S', 'R', 'P']
 7/55 [==>...........................] - ETA: 5s - loss: 1.3778 - accuracy: 0.9688['S', 'R', 'P']
 8/55 [===>..........................] - ETA: 5s - loss: 1.3732 - accuracy: 0.9688['S', 'R', 'P']
 9/55 [===>..........................] - ETA: 5s - loss: 1.3654 - accuracy: 0.9688['S', 'R', 'P']
10/55 [====>.........................] - ETA: 5s - loss: 1.3710 - accuracy: 0.9656['S', 'R', 'P']
11/55 [=====>........................] - ETA: 5s - loss: 1.3613 - accuracy: 0.9688['S', 'R', 'P']
12/55 [=====>........................] - ETA: 4s - loss: 1.3650 - accuracy: 0.9661['S', 'R', 'P']
13/55 [======>.......................] - ETA: 4s - loss: 1.3871 - accuracy: 0.9591['S', 'R', 'P']
14/55 [======>.......................] - ETA: 4s - loss: 1.3794 - accuracy: 0.9621['S', 'R', 'P']
15/55 [=======>......................] - ETA: 4s - loss: 1.3751 - accuracy: 0.9646['S', 'R', 'P']
16/55 [=======>......................] - ETA: 4s - loss: 1.3737 - accuracy: 0.9648['S', 'R', 'P']
17/55 [========>.....................] - ETA: 4s - loss: 1.3716 - accuracy: 0.9651['S', 'R', 'P']
18/55 [========>.....................] - ETA: 4s - loss: 1.3686 - accuracy: 0.9653['S', 'R', 'P']
19/55 [=========>....................] - ETA: 4s - loss: 1.3670 - accuracy: 0.9638['S', 'R', 'P']
20/55 [=========>....................] - ETA: 4s - loss: 1.3647 - accuracy: 0.9641['S', 'R', 'P']
21/55 [==========>...................] - ETA: 4s - loss: 1.3646 - accuracy: 0.9643['S', 'R', 'P']
22/55 [==========>...................] - ETA: 4s - loss: 1.3618 - accuracy: 0.9659['S', 'R', 'P']
23/55 [==========>...................] - ETA: 3s - loss: 1.3608 - accuracy: 0.9660['S', 'R', 'P']
24/55 [===========>..................] - ETA: 3s - loss: 1.3588 - accuracy: 0.9661['S', 'R', 'P']
25/55 [===========>..................] - ETA: 3s - loss: 1.3616 - accuracy: 0.9638['S', 'R', 'P']
26/55 [============>.................] - ETA: 3s - loss: 1.3616 - accuracy: 0.9639['S', 'R', 'P']
27/55 [============>.................] - ETA: 3s - loss: 1.3598 - accuracy: 0.9630['S', 'R', 'P']
28/55 [=============>................] - ETA: 3s - loss: 1.3559 - accuracy: 0.9643['S', 'R', 'P']
29/55 [=============>................] - ETA: 3s - loss: 1.3521 - accuracy: 0.9655['S', 'R', 'P']
30/55 [==============>...............] - ETA: 3s - loss: 1.3503 - accuracy: 0.9646['S', 'R', 'P']
31/55 [==============>...............] - ETA: 3s - loss: 1.3480 - accuracy: 0.9657['S', 'R', 'P']
32/55 [===============>..............] - ETA: 3s - loss: 1.3477 - accuracy: 0.9658['S', 'R', 'P']
33/55 [===============>..............] - ETA: 2s - loss: 1.3449 - accuracy: 0.9669['S', 'R', 'P']
34/55 [================>.............] - ETA: 2s - loss: 1.3428 - accuracy: 0.9678['S', 'R', 'P']
35/55 [================>.............] - ETA: 2s - loss: 1.3415 - accuracy: 0.9675['S', 'R', 'P']
36/55 [=================>............] - ETA: 2s - loss: 1.3405 - accuracy: 0.9676['S', 'R', 'P']
```

```
37/55 [===================>.........] - ETA: 2s - loss: 1.3393 - accuracy: 0.9676['S', 'R', 'P']
38/55 [===================>.........] - ETA: 2s - loss: 1.3403 - accuracy: 0.9668['S', 'R', 'P']
39/55 [====================>........] - ETA: 2s - loss: 1.3387 - accuracy: 0.9677['S', 'R', 'P']
40/55 [====================>........] - ETA: 2s - loss: 1.3366 - accuracy: 0.9685['S', 'R', 'P']
41/55 [=====================>.......] - ETA: 1s - loss: 1.3348 - accuracy: 0.9685['S', 'R', 'P']
42/55 [=====================>.......] - ETA: 1s - loss: 1.3328 - accuracy: 0.9677['S', 'R', 'P']
43/55 [======================>......] - ETA: 1s - loss: 1.3310 - accuracy: 0.9685['S', 'R', 'P']
44/55 [======================>......] - ETA: 1s - loss: 1.3339 - accuracy: 0.9656['S', 'R', 'P']
45/55 [======================>......] - ETA: 1s - loss: 1.3314 - accuracy: 0.9664['S', 'R', 'P']
46/55 [=======================>.....] - ETA: 1s - loss: 1.3346 - accuracy: 0.9637['S', 'R', 'P']
47/55 [=======================>.....] - ETA: 1s - loss: 1.3334 - accuracy: 0.9632['S', 'R', 'P']
48/55 [========================>....] - ETA: 0s - loss: 1.3330 - accuracy: 0.9633['S', 'R', 'P']
49/55 [========================>....] - ETA: 0s - loss: 1.3311 - accuracy: 0.9634['S', 'R', 'P']
50/55 [=========================>...] - ETA: 0s - loss: 1.3307 - accuracy: 0.9635['S', 'R', 'P']
51/55 [=========================>...] - ETA: 0s - loss: 1.3325 - accuracy: 0.9630['S', 'R', 'P']
52/55 [==========================>..] - ETA: 0s - loss: 1.3303 - accuracy: 0.9631['S', 'R', 'P']
53/55 [==========================>..] - ETA: 0s - loss: 1.3330 - accuracy: 0.9614['S', 'R', 'P']
54/55 [==========================>.] - ETA: 0s - loss: 1.3329 - accuracy: 0.9610['S', 'R', 'P']
55/55 [==============================] - ETA: 0s - loss: 1.3341 - accuracy: 0.9594['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
55/55 [==============================] - 10s 177ms/step - loss: 1.3341 - accuracy: 0.9594 - val_loss: 1.3877 - val_accuracy: 0.9
339
Epoch 10/20
['S', 'R', 'P']
 1/55 [..............................] - ETA: 6s - loss: 1.3510 - accuracy: 0.9062['S', 'R', 'P']
 2/55 [>.............................] - ETA: 6s - loss: 1.3340 - accuracy: 0.9219['S', 'R', 'P']
 3/55 [>.............................] - ETA: 5s - loss: 1.2838 - accuracy: 0.9479['S', 'R', 'P']
 4/55 [=>............................] - ETA: 5s - loss: 1.2884 - accuracy: 0.9531['S', 'R', 'P']
 5/55 [=>............................] - ETA: 5s - loss: 1.2957 - accuracy: 0.9500['S', 'R', 'P']
 6/55 [==>...........................] - ETA: 5s - loss: 1.2906 - accuracy: 0.9531['S', 'R', 'P']
 7/55 [==>...........................] - ETA: 5s - loss: 1.2809 - accuracy: 0.9554['S', 'R', 'P']
 8/55 [===>..........................] - ETA: 5s - loss: 1.2717 - accuracy: 0.9609['S', 'R', 'P']
```

```
 9/55 [===>..........................] - ETA: 5s - loss: 1.2824 - accuracy: 0.9514['S', 'R', 'P']
10/55 [====>.........................] - ETA: 5s - loss: 1.2789 - accuracy: 0.9500['S', 'R', 'P']
11/55 [=====>........................] - ETA: 5s - loss: 1.2771 - accuracy: 0.9517['S', 'R', 'P']
12/55 [=====>........................] - ETA: 5s - loss: 1.2750 - accuracy: 0.9505['S', 'R', 'P']
13/55 [======>.......................] - ETA: 4s - loss: 1.2711 - accuracy: 0.9543['S', 'R', 'P']
14/55 [======>.......................] - ETA: 4s - loss: 1.2684 - accuracy: 0.9576['S', 'R', 'P']
15/55 [=======>......................] - ETA: 4s - loss: 1.2627 - accuracy: 0.9604['S', 'R', 'P']
16/55 [=======>......................] - ETA: 4s - loss: 1.2592 - accuracy: 0.9629['S', 'R', 'P']
17/55 [========>.....................] - ETA: 4s - loss: 1.2597 - accuracy: 0.9632['S', 'R', 'P']
18/55 [========>.....................] - ETA: 4s - loss: 1.2636 - accuracy: 0.9618['S', 'R', 'P']
19/55 [=========>....................] - ETA: 4s - loss: 1.2610 - accuracy: 0.9622['S', 'R', 'P']
20/55 [=========>....................] - ETA: 4s - loss: 1.2602 - accuracy: 0.9609['S', 'R', 'P']
21/55 [==========>...................] - ETA: 3s - loss: 1.2587 - accuracy: 0.9613['S', 'R', 'P']
22/55 [==========>...................] - ETA: 3s - loss: 1.2638 - accuracy: 0.9588['S', 'R', 'P']
23/55 [==========>...................] - ETA: 3s - loss: 1.2653 - accuracy: 0.9572['S', 'R', 'P']
24/55 [===========>..................] - ETA: 3s - loss: 1.2671 - accuracy: 0.9551['S', 'R', 'P']
25/55 [===========>..................] - ETA: 3s - loss: 1.2653 - accuracy: 0.9556['S', 'R', 'P']
26/55 [===========>..................] - ETA: 3s - loss: 1.2655 - accuracy: 0.9562['S', 'R', 'P']
27/55 [============>.................] - ETA: 3s - loss: 1.2649 - accuracy: 0.9543['S', 'R', 'P']
28/55 [============>.................] - ETA: 3s - loss: 1.2622 - accuracy: 0.9548['S', 'R', 'P']
29/55 [============>.................] - ETA: 2s - loss: 1.2611 - accuracy: 0.9553['S', 'R', 'P']
30/55 [=============>................] - ETA: 2s - loss: 1.2590 - accuracy: 0.9557['S', 'R', 'P']
31/55 [=============>................] - ETA: 2s - loss: 1.2593 - accuracy: 0.9551['S', 'R', 'P']
32/55 [==============>...............] - ETA: 2s - loss: 1.2557 - accuracy: 0.9566['S', 'R', 'P']
33/55 [==============>...............] - ETA: 2s - loss: 1.2527 - accuracy: 0.9579['S', 'R', 'P']
34/55 [==============>...............] - ETA: 2s - loss: 1.2504 - accuracy: 0.9582['S', 'R', 'P']
35/55 [===============>..............] - ETA: 2s - loss: 1.2491 - accuracy: 0.9576['S', 'R', 'P']
36/55 [===============>..............] - ETA: 2s - loss: 1.2460 - accuracy: 0.9588['S', 'R', 'P']
37/55 [================>.............] - ETA: 2s - loss: 1.2447 - accuracy: 0.9574['S', 'R', 'P']
38/55 [================>.............] - ETA: 1s - loss: 1.2446 - accuracy: 0.9568['S', 'R', 'P']
39/55 [=================>............] - ETA: 1s - loss: 1.2419 - accuracy: 0.9580['S', 'R', 'P']
40/55 [=================>............] - ETA: 1s - loss: 1.2404 - accuracy: 0.9590['S', 'R', 'P']
41/55 [==================>...........] - ETA: 1s - loss: 1.2391 - accuracy: 0.9585['S', 'R', 'P']
42/55 [==================>...........] - ETA: 1s - loss: 1.2381 - accuracy: 0.9580['S', 'R', 'P']
43/55 [==================>...........] - ETA: 1s - loss: 1.2366 - accuracy: 0.9590['S', 'R', 'P']
44/55 [===================>..........] - ETA: 1s - loss: 1.2364 - accuracy: 0.9592['S', 'R', 'P']
45/55 [===================>..........] - ETA: 1s - loss: 1.2372 - accuracy: 0.9587['S', 'R', 'P']
46/55 [====================>.........] - ETA: 1s - loss: 1.2367 - accuracy: 0.9589['S', 'R', 'P']
47/55 [====================>.........] - ETA: 0s - loss: 1.2360 - accuracy: 0.9591['S', 'R', 'P']
48/55 [=====================>........] - ETA: 0s - loss: 1.2362 - accuracy: 0.9587['S', 'R', 'P']
49/55 [=====================>........] - ETA: 0s - loss: 1.2386 - accuracy: 0.9583['S', 'R', 'P']
50/55 [======================>.......] - ETA: 0s - loss: 1.2359 - accuracy: 0.9591['S', 'R', 'P']
51/55 [======================>.......] - ETA: 0s - loss: 1.2353 - accuracy: 0.9593['S', 'R', 'P']
52/55 [=======================>......] - ETA: 0s - loss: 1.2338 - accuracy: 0.9595['S', 'R', 'P']
```

```
53/55 [============================>..] - ETA: 0s - loss: 1.2321 - accuracy: 0.9602['S', 'R', 'P']
54/55 [============================>.] - ETA: 0s - loss: 1.2306 - accuracy: 0.9604['S', 'R', 'P']
55/55 [==============================] - ETA: 0s - loss: 1.2310 - accuracy: 0.9605['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
55/55 [==============================] - 8s 141ms/step - loss: 1.2310 - accuracy: 0.9605 - val_loss: 1.2613 - val_accuracy: 0.92
71
Epoch 11/20
['S', 'R', 'P']
 1/55 [..............................] - ETA: 9s - loss: 1.1011 - accuracy: 1.0000['S', 'R', 'P']
 2/55 [>.............................] - ETA: 9s - loss: 1.1287 - accuracy: 0.9844['S', 'R', 'P']
 3/55 [>.............................] - ETA: 8s - loss: 1.1255 - accuracy: 0.9896['S', 'R', 'P']
 4/55 [=>............................] - ETA: 8s - loss: 1.1358 - accuracy: 0.9844['S', 'R', 'P']
 5/55 [=>............................] - ETA: 8s - loss: 1.1444 - accuracy: 0.9812['S', 'R', 'P']
 6/55 [==>...........................] - ETA: 8s - loss: 1.1580 - accuracy: 0.9688['S', 'R', 'P']
 7/55 [==>...........................] - ETA: 7s - loss: 1.1629 - accuracy: 0.9671['S', 'R', 'P']
 8/55 [===>..........................] - ETA: 7s - loss: 1.1545 - accuracy: 0.9714['S', 'R', 'P']
 9/55 [===>..........................] - ETA: 7s - loss: 1.1554 - accuracy: 0.9711['S', 'R', 'P']
10/55 [====>.........................] - ETA: 7s - loss: 1.1616 - accuracy: 0.9676['S', 'R', 'P']
11/55 [=====>........................] - ETA: 6s - loss: 1.1580 - accuracy: 0.9707['S', 'R', 'P']
12/55 [=====>........................] - ETA: 6s - loss: 1.1632 - accuracy: 0.9678['S', 'R', 'P']
13/55 [======>.......................] - ETA: 6s - loss: 1.1560 - accuracy: 0.9704['S', 'R', 'P']
14/55 [======>.......................] - ETA: 6s - loss: 1.1478 - accuracy: 0.9725['S', 'R', 'P']
15/55 [=======>......................] - ETA: 6s - loss: 1.1548 - accuracy: 0.9680['S', 'R', 'P']
16/55 [=======>......................] - ETA: 6s - loss: 1.1523 - accuracy: 0.9681['S', 'R', 'P']
17/55 [========>.....................] - ETA: 6s - loss: 1.1481 - accuracy: 0.9681['S', 'R', 'P']
18/55 [========>.....................] - ETA: 5s - loss: 1.1457 - accuracy: 0.9681['S', 'R', 'P']
19/55 [=========>....................] - ETA: 5s - loss: 1.1455 - accuracy: 0.9665['S', 'R', 'P']
20/55 [=========>....................] - ETA: 5s - loss: 1.1439 - accuracy: 0.9666['S', 'R', 'P']
21/55 [==========>...................] - ETA: 5s - loss: 1.1463 - accuracy: 0.9652['S', 'R', 'P']
22/55 [==========>...................] - ETA: 5s - loss: 1.1464 - accuracy: 0.9654['S', 'R', 'P']
23/55 [==========>...................] - ETA: 5s - loss: 1.1463 - accuracy: 0.9655['S', 'R', 'P']
24/55 [===========>..................] - ETA: 5s - loss: 1.1508 - accuracy: 0.9657['S', 'R', 'P']
```

```
25/55 [============>.................] - ETA: 4s - loss: 1.1521 - accuracy: 0.9645['S', 'R', 'P']
26/55 [============>.................] - ETA: 4s - loss: 1.1594 - accuracy: 0.9622['S', 'R', 'P']
27/55 [============>.................] - ETA: 4s - loss: 1.1588 - accuracy: 0.9625['S', 'R', 'P']
28/55 [=============>................] - ETA: 4s - loss: 1.1548 - accuracy: 0.9638['S', 'R', 'P']
29/55 [=============>................] - ETA: 4s - loss: 1.1539 - accuracy: 0.9640['S', 'R', 'P']
30/55 [==============>...............] - ETA: 3s - loss: 1.1521 - accuracy: 0.9642['S', 'R', 'P']
31/55 [==============>...............] - ETA: 3s - loss: 1.1501 - accuracy: 0.9653['S', 'R', 'P']
32/55 [===============>..............] - ETA: 3s - loss: 1.1467 - accuracy: 0.9664['S', 'R', 'P']
33/55 [================>.............] - ETA: 3s - loss: 1.1455 - accuracy: 0.9675['S', 'R', 'P']
34/55 [================>.............] - ETA: 3s - loss: 1.1458 - accuracy: 0.9666['S', 'R', 'P']
35/55 [=================>............] - ETA: 3s - loss: 1.1477 - accuracy: 0.9657['S', 'R', 'P']
36/55 [=================>............] - ETA: 2s - loss: 1.1543 - accuracy: 0.9623['S', 'R', 'P']
37/55 [==================>...........] - ETA: 2s - loss: 1.1528 - accuracy: 0.9625['S', 'R', 'P']
38/55 [==================>...........] - ETA: 2s - loss: 1.1505 - accuracy: 0.9635['S', 'R', 'P']
39/55 [===================>..........] - ETA: 2s - loss: 1.1496 - accuracy: 0.9636['S', 'R', 'P']
40/55 [===================>..........] - ETA: 2s - loss: 1.1488 - accuracy: 0.9630['S', 'R', 'P']
41/55 [====================>.........] - ETA: 2s - loss: 1.1484 - accuracy: 0.9631['S', 'R', 'P']
42/55 [====================>.........] - ETA: 1s - loss: 1.1462 - accuracy: 0.9640['S', 'R', 'P']
43/55 [=====================>........] - ETA: 1s - loss: 1.1436 - accuracy: 0.9648['S', 'R', 'P']
44/55 [=====================>........] - ETA: 1s - loss: 1.1450 - accuracy: 0.9642['S', 'R', 'P']
45/55 [======================>.......] - ETA: 1s - loss: 1.1430 - accuracy: 0.9650['S', 'R', 'P']
46/55 [======================>.......] - ETA: 1s - loss: 1.1411 - accuracy: 0.9658['S', 'R', 'P']
47/55 [=======================>......] - ETA: 1s - loss: 1.1395 - accuracy: 0.9665['S', 'R', 'P']
48/55 [=======================>......] - ETA: 0s - loss: 1.1417 - accuracy: 0.9646['S', 'R', 'P']
49/55 [========================>.....] - ETA: 0s - loss: 1.1419 - accuracy: 0.9640['S', 'R', 'P']
50/55 [========================>.....] - ETA: 0s - loss: 1.1404 - accuracy: 0.9648['S', 'R', 'P']
51/55 [=========================>....] - ETA: 0s - loss: 1.1415 - accuracy: 0.9642['S', 'R', 'P']
52/55 [=========================>....] - ETA: 0s - loss: 1.1401 - accuracy: 0.9643['S', 'R', 'P']
53/55 [==========================>...] - ETA: 0s - loss: 1.1386 - accuracy: 0.9650['S', 'R', 'P']
54/55 [==========================>...] - ETA: 0s - loss: 1.1424 - accuracy: 0.9645['S', 'R', 'P']
55/55 [==============================] - ETA: 0s - loss: 1.1426 - accuracy: 0.9640['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
```

```
55/55 [==============================] - 9s 167ms/step - loss: 1.1426 - accuracy: 0.9640 - val_loss: 1.1807 - val_accuracy: 0.92
71
Epoch 12/20
['S', 'R', 'P']
 1/55 [..............................] - ETA: 7s - loss: 1.0597 - accuracy: 0.9688['S', 'R', 'P']
 2/55 [>.............................] - ETA: 4s - loss: 1.1065 - accuracy: 0.9531['S', 'R', 'P']
 3/55 [>.............................] - ETA: 4s - loss: 1.0928 - accuracy: 0.9583['S', 'R', 'P']
 4/55 [=>............................] - ETA: 5s - loss: 1.1065 - accuracy: 0.9531['S', 'R', 'P']
 5/55 [=>............................] - ETA: 5s - loss: 1.0972 - accuracy: 0.9625['S', 'R', 'P']
 6/55 [==>...........................] - ETA: 5s - loss: 1.0907 - accuracy: 0.9688['S', 'R', 'P']
 7/55 [==>...........................] - ETA: 5s - loss: 1.0858 - accuracy: 0.9732['S', 'R', 'P']
 8/55 [===>..........................] - ETA: 4s - loss: 1.0878 - accuracy: 0.9688['S', 'R', 'P']
 9/55 [===>..........................] - ETA: 4s - loss: 1.0813 - accuracy: 0.9722['S', 'R', 'P']
10/55 [====>.........................] - ETA: 4s - loss: 1.0852 - accuracy: 0.9719['S', 'R', 'P']
11/55 [=====>........................] - ETA: 4s - loss: 1.0783 - accuracy: 0.9744['S', 'R', 'P']
12/55 [=====>........................] - ETA: 4s - loss: 1.0899 - accuracy: 0.9661['S', 'R', 'P']
13/55 [======>.......................] - ETA: 4s - loss: 1.0915 - accuracy: 0.9663['S', 'R', 'P']
14/55 [======>.......................] - ETA: 4s - loss: 1.0917 - accuracy: 0.9665['S', 'R', 'P']
15/55 [=======>......................] - ETA: 4s - loss: 1.0894 - accuracy: 0.9667['S', 'R', 'P']
16/55 [=======>......................] - ETA: 4s - loss: 1.0883 - accuracy: 0.9688['S', 'R', 'P']
17/55 [========>.....................] - ETA: 4s - loss: 1.0813 - accuracy: 0.9706['S', 'R', 'P']
18/55 [========>.....................] - ETA: 3s - loss: 1.0828 - accuracy: 0.9688['S', 'R', 'P']
19/55 [=========>....................] - ETA: 3s - loss: 1.0803 - accuracy: 0.9688['S', 'R', 'P']
20/55 [=========>....................] - ETA: 3s - loss: 1.0774 - accuracy: 0.9688['S', 'R', 'P']
21/55 [==========>...................] - ETA: 3s - loss: 1.0813 - accuracy: 0.9673['S', 'R', 'P']
22/55 [==========>...................] - ETA: 3s - loss: 1.0919 - accuracy: 0.9602['S', 'R', 'P']
23/55 [==========>...................] - ETA: 3s - loss: 1.0924 - accuracy: 0.9592['S', 'R', 'P']
24/55 [===========>..................] - ETA: 3s - loss: 1.0899 - accuracy: 0.9596['S', 'R', 'P']
25/55 [===========>..................] - ETA: 3s - loss: 1.0899 - accuracy: 0.9588['S', 'R', 'P']
26/55 [============>.................] - ETA: 3s - loss: 1.0931 - accuracy: 0.9579['S', 'R', 'P']
27/55 [============>.................] - ETA: 3s - loss: 1.0958 - accuracy: 0.9572['S', 'R', 'P']
28/55 [============>.................] - ETA: 2s - loss: 1.0940 - accuracy: 0.9576['S', 'R', 'P']
29/55 [============>.................] - ETA: 2s - loss: 1.0947 - accuracy: 0.9558['S', 'R', 'P']
30/55 [=============>................] - ETA: 2s - loss: 1.0926 - accuracy: 0.9563['S', 'R', 'P']
31/55 [=============>................] - ETA: 2s - loss: 1.0891 - accuracy: 0.9577['S', 'R', 'P']
32/55 [==============>...............] - ETA: 2s - loss: 1.0863 - accuracy: 0.9590['S', 'R', 'P']
33/55 [==============>...............] - ETA: 2s - loss: 1.0840 - accuracy: 0.9602['S', 'R', 'P']
34/55 [==============>...............] - ETA: 2s - loss: 1.0842 - accuracy: 0.9605['S', 'R', 'P']
35/55 [===============>..............] - ETA: 2s - loss: 1.0845 - accuracy: 0.9589['S', 'R', 'P']
36/55 [===============>..............] - ETA: 2s - loss: 1.0853 - accuracy: 0.9566['S', 'R', 'P']
37/55 [================>.............] - ETA: 1s - loss: 1.0863 - accuracy: 0.9552['S', 'R', 'P']
38/55 [================>.............] - ETA: 1s - loss: 1.0869 - accuracy: 0.9548['S', 'R', 'P']
39/55 [=================>............] - ETA: 1s - loss: 1.0856 - accuracy: 0.9559['S', 'R', 'P']
40/55 [=================>............] - ETA: 1s - loss: 1.0876 - accuracy: 0.9551['S', 'R', 'P']
```

```
41/55 [=====================>........] - ETA: 1s - loss: 1.0849 - accuracy: 0.9562['S', 'R', 'P']
42/55 [=====================>........] - ETA: 1s - loss: 1.0869 - accuracy: 0.9557['S', 'R', 'P']
43/55 [======================>.......] - ETA: 1s - loss: 1.0847 - accuracy: 0.9568['S', 'R', 'P']
44/55 [======================>.......] - ETA: 1s - loss: 1.0822 - accuracy: 0.9578['S', 'R', 'P']
45/55 [======================>.......] - ETA: 1s - loss: 1.0810 - accuracy: 0.9580['S', 'R', 'P']
46/55 [=======================>......] - ETA: 0s - loss: 1.0791 - accuracy: 0.9582['S', 'R', 'P']
47/55 [=======================>......] - ETA: 0s - loss: 1.0820 - accuracy: 0.9578['S', 'R', 'P']
48/55 [========================>.....] - ETA: 0s - loss: 1.0806 - accuracy: 0.9580['S', 'R', 'P']
49/55 [========================>.....] - ETA: 0s - loss: 1.0799 - accuracy: 0.9583['S', 'R', 'P']
50/55 [=========================>....] - ETA: 0s - loss: 1.0809 - accuracy: 0.9578['S', 'R', 'P']
51/55 [=========================>....] - ETA: 0s - loss: 1.0786 - accuracy: 0.9587['S', 'R', 'P']
52/55 [==========================>...] - ETA: 0s - loss: 1.0789 - accuracy: 0.9589['S', 'R', 'P']
53/55 [==========================>...] - ETA: 0s - loss: 1.0781 - accuracy: 0.9585['S', 'R', 'P']
54/55 [==========================>...] - ETA: 0s - loss: 1.0771 - accuracy: 0.9586['S', 'R', 'P']
55/55 [==============================] - ETA: 0s - loss: 1.0770 - accuracy: 0.9583['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
55/55 [==============================] - 9s 163ms/step - loss: 1.0770 - accuracy: 0.9583 - val_loss: 1.1286 - val_accuracy: 0.93
17
Epoch 13/20
['S', 'R', 'P']
 1/55 [..............................] - ETA: 6s - loss: 0.9873 - accuracy: 1.0000['S', 'R', 'P']
 2/55 [>.............................] - ETA: 6s - loss: 0.9890 - accuracy: 0.9844['S', 'R', 'P']
 3/55 [>.............................] - ETA: 6s - loss: 1.0514 - accuracy: 0.9583['S', 'R', 'P']
 4/55 [=>............................] - ETA: 5s - loss: 1.0720 - accuracy: 0.9609['S', 'R', 'P']
 5/55 [=>............................] - ETA: 5s - loss: 1.0535 - accuracy: 0.9688['S', 'R', 'P']
 6/55 [==>...........................] - ETA: 5s - loss: 1.0408 - accuracy: 0.9740['S', 'R', 'P']
 7/55 [==>...........................] - ETA: 5s - loss: 1.0298 - accuracy: 0.9777['S', 'R', 'P']
 8/55 [===>..........................] - ETA: 5s - loss: 1.0231 - accuracy: 0.9766['S', 'R', 'P']
 9/55 [===>..........................] - ETA: 5s - loss: 1.0348 - accuracy: 0.9722['S', 'R', 'P']
10/55 [====>.........................] - ETA: 5s - loss: 1.0360 - accuracy: 0.9719['S', 'R', 'P']
11/55 [=====>........................] - ETA: 5s - loss: 1.0348 - accuracy: 0.9688['S', 'R', 'P']
12/55 [=====>........................] - ETA: 4s - loss: 1.0338 - accuracy: 0.9714['S', 'R', 'P']
```

```
13/55 [======>.....................] - ETA: 4s - loss: 1.0338 - accuracy: 0.9712['S', 'R', 'P']
14/55 [======>.....................] - ETA: 4s - loss: 1.0256 - accuracy: 0.9732['S', 'R', 'P']
15/55 [=======>.....................] - ETA: 4s - loss: 1.0234 - accuracy: 0.9729['S', 'R', 'P']
16/55 [=======>.....................] - ETA: 4s - loss: 1.0186 - accuracy: 0.9727['S', 'R', 'P']
17/55 [========>.....................] - ETA: 4s - loss: 1.0150 - accuracy: 0.9743['S', 'R', 'P']
18/55 [========>.....................] - ETA: 4s - loss: 1.0222 - accuracy: 0.9705['S', 'R', 'P']
19/55 [=========>.....................] - ETA: 4s - loss: 1.0237 - accuracy: 0.9688['S', 'R', 'P']
20/55 [=========>.....................] - ETA: 4s - loss: 1.0273 - accuracy: 0.9641['S', 'R', 'P']
21/55 [==========>.....................] - ETA: 3s - loss: 1.0292 - accuracy: 0.9643['S', 'R', 'P']
22/55 [==========>.....................] - ETA: 3s - loss: 1.0245 - accuracy: 0.9659['S', 'R', 'P']
23/55 [==========>.....................] - ETA: 3s - loss: 1.0206 - accuracy: 0.9660['S', 'R', 'P']
24/55 [===========>.....................] - ETA: 3s - loss: 1.0166 - accuracy: 0.9674['S', 'R', 'P']
25/55 [===========>.....................] - ETA: 3s - loss: 1.0259 - accuracy: 0.9650['S', 'R', 'P']
26/55 [============>.....................] - ETA: 3s - loss: 1.0271 - accuracy: 0.9639['S', 'R', 'P']
27/55 [============>.....................] - ETA: 3s - loss: 1.0250 - accuracy: 0.9641['S', 'R', 'P']
28/55 [============>.....................] - ETA: 3s - loss: 1.0213 - accuracy: 0.9654['S', 'R', 'P']
29/55 [============>.....................] - ETA: 2s - loss: 1.0198 - accuracy: 0.9666['S', 'R', 'P']
30/55 [=============>.....................] - ETA: 2s - loss: 1.0198 - accuracy: 0.9667['S', 'R', 'P']
31/55 [=============>.....................] - ETA: 2s - loss: 1.0183 - accuracy: 0.9667['S', 'R', 'P']
32/55 [=============>.....................] - ETA: 2s - loss: 1.0155 - accuracy: 0.9678['S', 'R', 'P']
33/55 [==============>.....................] - ETA: 2s - loss: 1.0146 - accuracy: 0.9678['S', 'R', 'P']
34/55 [==============>.....................] - ETA: 2s - loss: 1.0184 - accuracy: 0.9669['S', 'R', 'P']
35/55 [==============>.....................] - ETA: 2s - loss: 1.0171 - accuracy: 0.9670['S', 'R', 'P']
36/55 [==============>.....................] - ETA: 2s - loss: 1.0159 - accuracy: 0.9670['S', 'R', 'P']
37/55 [===============>.....................] - ETA: 2s - loss: 1.0140 - accuracy: 0.9671['S', 'R', 'P']
38/55 [===============>.....................] - ETA: 1s - loss: 1.0162 - accuracy: 0.9646['S', 'R', 'P']
39/55 [===============>.....................] - ETA: 1s - loss: 1.0179 - accuracy: 0.9631['S', 'R', 'P']
40/55 [================>.....................] - ETA: 1s - loss: 1.0161 - accuracy: 0.9641['S', 'R', 'P']
41/55 [================>.....................] - ETA: 1s - loss: 1.0136 - accuracy: 0.9649['S', 'R', 'P']
42/55 [================>.....................] - ETA: 1s - loss: 1.0119 - accuracy: 0.9650['S', 'R', 'P']
43/55 [================>.....................] - ETA: 1s - loss: 1.0125 - accuracy: 0.9644['S', 'R', 'P']
44/55 [=================>.....................] - ETA: 1s - loss: 1.0155 - accuracy: 0.9631['S', 'R', 'P']
45/55 [=================>.....................] - ETA: 1s - loss: 1.0162 - accuracy: 0.9625['S', 'R', 'P']
46/55 [=================>.....................] - ETA: 1s - loss: 1.0168 - accuracy: 0.9617['S', 'R', 'P']
47/55 [=================>.....................] - ETA: 0s - loss: 1.0173 - accuracy: 0.9612['S', 'R', 'P']
48/55 [==================>.....................] - ETA: 0s - loss: 1.0163 - accuracy: 0.9620['S', 'R', 'P']
49/55 [==================>.....................] - ETA: 0s - loss: 1.0166 - accuracy: 0.9621['S', 'R', 'P']
50/55 [==================>.....................] - ETA: 0s - loss: 1.0209 - accuracy: 0.9622['S', 'R', 'P']
51/55 [==================>.....................] - ETA: 0s - loss: 1.0215 - accuracy: 0.9618['S', 'R', 'P']
52/55 [==================>.....................] - ETA: 0s - loss: 1.0199 - accuracy: 0.9625['S', 'R', 'P']
53/55 [==================>.....................] - ETA: 0s - loss: 1.0199 - accuracy: 0.9620['S', 'R', 'P']
54/55 [==================>.....................] - ETA: 0s - loss: 1.0216 - accuracy: 0.9616['S', 'R', 'P']
55/55 [==============================] - ETA: 0s - loss: 1.0244 - accuracy: 0.9605['S', 'R', 'P']
['S', 'R', 'P']
```

```
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
55/55 [==============================] - 8s 143ms/step - loss: 1.0244 - accuracy: 0.9605 - val_loss: 1.2603 - val_accuracy: 0.87
47
Epoch 14/20
['S', 'R', 'P']
 1/55 [..............................] - ETA: 8s - loss: 0.9484 - accuracy: 1.0000['S', 'R', 'P']
 2/55 [>.............................] - ETA: 4s - loss: 1.0034 - accuracy: 0.9375['S', 'R', 'P']
 3/55 [>.............................] - ETA: 5s - loss: 1.0106 - accuracy: 0.9479['S', 'R', 'P']
 4/55 [=>............................] - ETA: 5s - loss: 1.0168 - accuracy: 0.9453['S', 'R', 'P']
 5/55 [=>............................] - ETA: 5s - loss: 1.0099 - accuracy: 0.9438['S', 'R', 'P']
 6/55 [==>...........................] - ETA: 5s - loss: 1.0163 - accuracy: 0.9375['S', 'R', 'P']
 7/55 [==>...........................] - ETA: 5s - loss: 1.0126 - accuracy: 0.9420['S', 'R', 'P']
 8/55 [===>..........................] - ETA: 5s - loss: 1.0373 - accuracy: 0.9375['S', 'R', 'P']
 9/55 [===>..........................] - ETA: 5s - loss: 1.0277 - accuracy: 0.9444['S', 'R', 'P']
10/55 [====>.........................] - ETA: 5s - loss: 1.0198 - accuracy: 0.9500['S', 'R', 'P']
11/55 [=====>........................] - ETA: 4s - loss: 1.0146 - accuracy: 0.9517['S', 'R', 'P']
12/55 [=====>........................] - ETA: 4s - loss: 1.0142 - accuracy: 0.9505['S', 'R', 'P']
13/55 [======>.......................] - ETA: 4s - loss: 1.0112 - accuracy: 0.9519['S', 'R', 'P']
14/55 [======>.......................] - ETA: 4s - loss: 1.0033 - accuracy: 0.9554['S', 'R', 'P']
15/55 [=======>......................] - ETA: 4s - loss: 0.9987 - accuracy: 0.9583['S', 'R', 'P']
16/55 [=======>......................] - ETA: 4s - loss: 1.0125 - accuracy: 0.9551['S', 'R', 'P']
17/55 [========>.....................] - ETA: 4s - loss: 1.0151 - accuracy: 0.9540['S', 'R', 'P']
18/55 [========>.....................] - ETA: 4s - loss: 1.0146 - accuracy: 0.9514['S', 'R', 'P']
19/55 [=========>....................] - ETA: 4s - loss: 1.0101 - accuracy: 0.9539['S', 'R', 'P']
20/55 [=========>....................] - ETA: 4s - loss: 1.0043 - accuracy: 0.9563['S', 'R', 'P']
21/55 [==========>...................] - ETA: 3s - loss: 1.0009 - accuracy: 0.9568['S', 'R', 'P']
22/55 [==========>...................] - ETA: 3s - loss: 0.9965 - accuracy: 0.9574['S', 'R', 'P']
23/55 [==========>...................] - ETA: 3s - loss: 0.9941 - accuracy: 0.9592['S', 'R', 'P']
24/55 [===========>..................] - ETA: 3s - loss: 0.9947 - accuracy: 0.9583['S', 'R', 'P']
25/55 [===========>..................] - ETA: 3s - loss: 0.9962 - accuracy: 0.9563['S', 'R', 'P']
26/55 [============>.................] - ETA: 3s - loss: 0.9922 - accuracy: 0.9579['S', 'R', 'P']
27/55 [============>.................] - ETA: 3s - loss: 0.9891 - accuracy: 0.9595['S', 'R', 'P']
28/55 [============>.................] - ETA: 3s - loss: 0.9875 - accuracy: 0.9598['S', 'R', 'P']
```

```
29/55 [==============>..............] - ETA: 3s - loss: 0.9856 - accuracy: 0.9601['S', 'R', 'P']
30/55 [==============>..............] - ETA: 3s - loss: 0.9855 - accuracy: 0.9604['S', 'R', 'P']
31/55 [==============>..............] - ETA: 3s - loss: 0.9857 - accuracy: 0.9597['S', 'R', 'P']
32/55 [===============>.............] - ETA: 2s - loss: 0.9869 - accuracy: 0.9570['S', 'R', 'P']
33/55 [================>............] - ETA: 2s - loss: 0.9844 - accuracy: 0.9583['S', 'R', 'P']
34/55 [================>............] - ETA: 2s - loss: 0.9835 - accuracy: 0.9586['S', 'R', 'P']
35/55 [=================>...........] - ETA: 2s - loss: 0.9850 - accuracy: 0.9589['S', 'R', 'P']
36/55 [=================>...........] - ETA: 2s - loss: 0.9870 - accuracy: 0.9583['S', 'R', 'P']
37/55 [==================>..........] - ETA: 2s - loss: 0.9862 - accuracy: 0.9586['S', 'R', 'P']
38/55 [==================>..........] - ETA: 2s - loss: 0.9852 - accuracy: 0.9589['S', 'R', 'P']
39/55 [===================>.........] - ETA: 2s - loss: 0.9841 - accuracy: 0.9591['S', 'R', 'P']
40/55 [===================>.........] - ETA: 1s - loss: 0.9812 - accuracy: 0.9602['S', 'R', 'P']
41/55 [====================>........] - ETA: 1s - loss: 0.9815 - accuracy: 0.9593['S', 'R', 'P']
42/55 [====================>........] - ETA: 1s - loss: 0.9793 - accuracy: 0.9595['S', 'R', 'P']
43/55 [=====================>.......] - ETA: 1s - loss: 0.9771 - accuracy: 0.9604['S', 'R', 'P']
44/55 [=====================>.......] - ETA: 1s - loss: 0.9748 - accuracy: 0.9613['S', 'R', 'P']
45/55 [======================>......] - ETA: 1s - loss: 0.9730 - accuracy: 0.9615['S', 'R', 'P']
46/55 [======================>......] - ETA: 1s - loss: 0.9723 - accuracy: 0.9617['S', 'R', 'P']
47/55 [======================>......] - ETA: 1s - loss: 0.9724 - accuracy: 0.9618['S', 'R', 'P']
48/55 [=======================>.....] - ETA: 0s - loss: 0.9736 - accuracy: 0.9613['S', 'R', 'P']
49/55 [=======================>.....] - ETA: 0s - loss: 0.9741 - accuracy: 0.9615['S', 'R', 'P']
50/55 [========================>....] - ETA: 0s - loss: 0.9735 - accuracy: 0.9616['S', 'R', 'P']
51/55 [========================>....] - ETA: 0s - loss: 0.9757 - accuracy: 0.9605['S', 'R', 'P']
52/55 [=========================>...] - ETA: 0s - loss: 0.9751 - accuracy: 0.9607['S', 'R', 'P']
53/55 [=========================>...] - ETA: 0s - loss: 0.9738 - accuracy: 0.9608['S', 'R', 'P']
54/55 [==========================>..] - ETA: 0s - loss: 0.9737 - accuracy: 0.9610['S', 'R', 'P']
55/55 [==========================>..] - ETA: 0s - loss: 0.9731 - accuracy: 0.9611['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
55/55 [==============================] - 10s 174ms/step - loss: 0.9731 - accuracy: 0.9611 - val_loss: 0.9945 - val_accuracy: 0.9
385
Epoch 15/20
['S', 'R', 'P']
```

```
 1/55 [..............................] - ETA: 7s - loss: 0.9171 - accuracy: 0.9688['S', 'R', 'P']
 2/55 [>.............................] - ETA: 6s - loss: 0.9847 - accuracy: 0.9219['S', 'R', 'P']
 3/55 [>.............................] - ETA: 6s - loss: 0.9445 - accuracy: 0.9479['S', 'R', 'P']
 4/55 [=>............................] - ETA: 5s - loss: 0.9315 - accuracy: 0.9573['S', 'R', 'P']
 5/55 [=>............................] - ETA: 5s - loss: 0.9203 - accuracy: 0.9597['S', 'R', 'P']
 6/55 [==>...........................] - ETA: 5s - loss: 0.9138 - accuracy: 0.9669['S', 'R', 'P']
 7/55 [==>...........................] - ETA: 5s - loss: 0.9136 - accuracy: 0.9671['S', 'R', 'P']
 8/55 [===>..........................] - ETA: 5s - loss: 0.9067 - accuracy: 0.9714['S', 'R', 'P']
 9/55 [===>..........................] - ETA: 5s - loss: 0.9082 - accuracy: 0.9711['S', 'R', 'P']
10/55 [====>.........................] - ETA: 5s - loss: 0.9095 - accuracy: 0.9676['S', 'R', 'P']
11/55 [=====>........................] - ETA: 5s - loss: 0.9181 - accuracy: 0.9677['S', 'R', 'P']
12/55 [=====>........................] - ETA: 4s - loss: 0.9148 - accuracy: 0.9705['S', 'R', 'P']
13/55 [======>.......................] - ETA: 4s - loss: 0.9148 - accuracy: 0.9704['S', 'R', 'P']
14/55 [======>.......................] - ETA: 4s - loss: 0.9096 - accuracy: 0.9725['S', 'R', 'P']
15/55 [=======>......................] - ETA: 4s - loss: 0.9056 - accuracy: 0.9744['S', 'R', 'P']
16/55 [=======>......................] - ETA: 4s - loss: 0.8999 - accuracy: 0.9760['S', 'R', 'P']
17/55 [========>.....................] - ETA: 4s - loss: 0.9057 - accuracy: 0.9719['S', 'R', 'P']
18/55 [========>.....................] - ETA: 4s - loss: 0.9010 - accuracy: 0.9735['S', 'R', 'P']
19/55 [=========>....................] - ETA: 4s - loss: 0.8988 - accuracy: 0.9732['S', 'R', 'P']
20/55 [=========>....................] - ETA: 4s - loss: 0.8976 - accuracy: 0.9730['S', 'R', 'P']
21/55 [==========>...................] - ETA: 4s - loss: 0.8968 - accuracy: 0.9728['S', 'R', 'P']
22/55 [==========>...................] - ETA: 3s - loss: 0.8963 - accuracy: 0.9711['S', 'R', 'P']
23/55 [==========>...................] - ETA: 3s - loss: 0.9017 - accuracy: 0.9683['S', 'R', 'P']
24/55 [===========>..................] - ETA: 3s - loss: 0.8998 - accuracy: 0.9696['S', 'R', 'P']
25/55 [===========>..................] - ETA: 3s - loss: 0.8962 - accuracy: 0.9708['S', 'R', 'P']
26/55 [============>.................] - ETA: 3s - loss: 0.9000 - accuracy: 0.9708['S', 'R', 'P']
27/55 [============>.................] - ETA: 3s - loss: 0.8986 - accuracy: 0.9695['S', 'R', 'P']
28/55 [=============>................] - ETA: 3s - loss: 0.8981 - accuracy: 0.9695['S', 'R', 'P']
29/55 [=============>................] - ETA: 3s - loss: 0.8979 - accuracy: 0.9695['S', 'R', 'P']
30/55 [==============>...............] - ETA: 2s - loss: 0.8976 - accuracy: 0.9684['S', 'R', 'P']
31/55 [==============>...............] - ETA: 2s - loss: 0.8966 - accuracy: 0.9694['S', 'R', 'P']
32/55 [===============>..............] - ETA: 2s - loss: 0.8990 - accuracy: 0.9674['S', 'R', 'P']
33/55 [================>.............] - ETA: 2s - loss: 0.8977 - accuracy: 0.9684['S', 'R', 'P']
34/55 [================>.............] - ETA: 2s - loss: 0.8992 - accuracy: 0.9675['S', 'R', 'P']
35/55 [=================>............] - ETA: 2s - loss: 0.8987 - accuracy: 0.9675['S', 'R', 'P']
36/55 [=================>............] - ETA: 2s - loss: 0.8992 - accuracy: 0.9676['S', 'R', 'P']
37/55 [==================>...........] - ETA: 2s - loss: 0.8964 - accuracy: 0.9685['S', 'R', 'P']
38/55 [==================>...........] - ETA: 2s - loss: 0.8951 - accuracy: 0.9693['S', 'R', 'P']
39/55 [===================>..........] - ETA: 1s - loss: 0.8936 - accuracy: 0.9701['S', 'R', 'P']
40/55 [===================>..........] - ETA: 1s - loss: 0.8927 - accuracy: 0.9708['S', 'R', 'P']
41/55 [====================>.........] - ETA: 1s - loss: 0.8919 - accuracy: 0.9708['S', 'R', 'P']
42/55 [====================>.........] - ETA: 1s - loss: 0.8913 - accuracy: 0.9707['S', 'R', 'P']
43/55 [=====================>........] - ETA: 1s - loss: 0.8898 - accuracy: 0.9707['S', 'R', 'P']
44/55 [=====================>........] - ETA: 1s - loss: 0.8874 - accuracy: 0.9714['S', 'R', 'P']
```

```
45/55 [=======================>......] - ETA: 1s - loss: 0.8865 - accuracy: 0.9713['S', 'R', 'P']
46/55 [=======================>......] - ETA: 1s - loss: 0.8862 - accuracy: 0.9719['S', 'R', 'P']
47/55 [=======================>.....] - ETA: 0s - loss: 0.8867 - accuracy: 0.9712['S', 'R', 'P']
48/55 [========================>.....] - ETA: 0s - loss: 0.8852 - accuracy: 0.9718['S', 'R', 'P']
49/55 [========================>.....] - ETA: 0s - loss: 0.8842 - accuracy: 0.9724['S', 'R', 'P']
50/55 [========================>....] - ETA: 0s - loss: 0.8864 - accuracy: 0.9717['S', 'R', 'P']
51/55 [========================>....] - ETA: 0s - loss: 0.8848 - accuracy: 0.9722['S', 'R', 'P']
52/55 [=========================>...] - ETA: 0s - loss: 0.8836 - accuracy: 0.9728['S', 'R', 'P']
53/55 [=========================>...] - ETA: 0s - loss: 0.8818 - accuracy: 0.9733['S', 'R', 'P']
54/55 [=========================>..] - ETA: 0s - loss: 0.8801 - accuracy: 0.9738['S', 'R', 'P']
55/55 [==========================] - ETA: 0s - loss: 0.8785 - accuracy: 0.9743['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
55/55 [==========================] - 8s 147ms/step - loss: 0.8785 - accuracy: 0.9743 - val_loss: 0.9213 - val_accuracy: 0.95
67
Epoch 16/20
['S', 'R', 'P']
 1/55 [..............................] - ETA: 7s - loss: 0.8162 - accuracy: 1.0000['S', 'R', 'P']
 2/55 [>.............................] - ETA: 4s - loss: 0.7991 - accuracy: 1.0000['S', 'R', 'P']
 3/55 [>.............................] - ETA: 5s - loss: 0.8058 - accuracy: 0.9882['S', 'R', 'P']
 4/55 [=>............................] - ETA: 5s - loss: 0.8186 - accuracy: 0.9829['S', 'R', 'P']
 5/55 [=>............................] - ETA: 5s - loss: 0.8272 - accuracy: 0.9799['S', 'R', 'P']
 6/55 [==>...........................] - ETA: 5s - loss: 0.8348 - accuracy: 0.9724['S', 'R', 'P']
 7/55 [==>...........................] - ETA: 5s - loss: 0.8383 - accuracy: 0.9671['S', 'R', 'P']
 8/55 [===>..........................] - ETA: 5s - loss: 0.8294 - accuracy: 0.9714['S', 'R', 'P']
 9/55 [===>..........................] - ETA: 5s - loss: 0.8230 - accuracy: 0.9747['S', 'R', 'P']
10/55 [====>.........................] - ETA: 5s - loss: 0.8201 - accuracy: 0.9773['S', 'R', 'P']
11/55 [=====>........................] - ETA: 5s - loss: 0.8232 - accuracy: 0.9736['S', 'R', 'P']
12/55 [=====>........................] - ETA: 5s - loss: 0.8215 - accuracy: 0.9759['S', 'R', 'P']
13/55 [======>.......................] - ETA: 4s - loss: 0.8251 - accuracy: 0.9728['S', 'R', 'P']
14/55 [======>.......................] - ETA: 5s - loss: 0.8256 - accuracy: 0.9725['S', 'R', 'P']
15/55 [=======>......................] - ETA: 4s - loss: 0.8282 - accuracy: 0.9701['S', 'R', 'P']
16/55 [=======>......................] - ETA: 4s - loss: 0.8291 - accuracy: 0.9701['S', 'R', 'P']
```

```
17/55 [========>.....................] - ETA: 4s - loss: 0.8331 - accuracy: 0.9700['S', 'R', 'P']
18/55 [========>.....................] - ETA: 4s - loss: 0.8339 - accuracy: 0.9699['S', 'R', 'P']
19/55 [=========>....................] - ETA: 4s - loss: 0.8360 - accuracy: 0.9698['S', 'R', 'P']
20/55 [=========>....................] - ETA: 4s - loss: 0.8381 - accuracy: 0.9698['S', 'R', 'P']
21/55 [==========>...................] - ETA: 4s - loss: 0.8374 - accuracy: 0.9697['S', 'R', 'P']
22/55 [==========>...................] - ETA: 4s - loss: 0.8355 - accuracy: 0.9697['S', 'R', 'P']
23/55 [==========>...................] - ETA: 4s - loss: 0.8330 - accuracy: 0.9710['S', 'R', 'P']
24/55 [===========>..................] - ETA: 4s - loss: 0.8318 - accuracy: 0.9723['S', 'R', 'P']
25/55 [===========>..................] - ETA: 4s - loss: 0.8295 - accuracy: 0.9734['S', 'R', 'P']
26/55 [============>.................] - ETA: 4s - loss: 0.8308 - accuracy: 0.9720['S', 'R', 'P']
27/55 [============>.................] - ETA: 3s - loss: 0.8330 - accuracy: 0.9719['S', 'R', 'P']
28/55 [=============>................] - ETA: 3s - loss: 0.8306 - accuracy: 0.9729['S', 'R', 'P']
29/55 [=============>................] - ETA: 3s - loss: 0.8301 - accuracy: 0.9727['S', 'R', 'P']
30/55 [==============>...............] - ETA: 3s - loss: 0.8311 - accuracy: 0.9715['S', 'R', 'P']
31/55 [==============>...............] - ETA: 3s - loss: 0.8321 - accuracy: 0.9715['S', 'R', 'P']
32/55 [===============>..............] - ETA: 3s - loss: 0.8324 - accuracy: 0.9704['S', 'R', 'P']
33/55 [================>.............] - ETA: 3s - loss: 0.8305 - accuracy: 0.9713['S', 'R', 'P']
34/55 [================>.............] - ETA: 2s - loss: 0.8292 - accuracy: 0.9721['S', 'R', 'P']
35/55 [=================>............] - ETA: 2s - loss: 0.8318 - accuracy: 0.9711['S', 'R', 'P']
36/55 [=================>............] - ETA: 2s - loss: 0.8317 - accuracy: 0.9711['S', 'R', 'P']
37/55 [==================>...........] - ETA: 2s - loss: 0.8326 - accuracy: 0.9710['S', 'R', 'P']
38/55 [==================>...........] - ETA: 2s - loss: 0.8320 - accuracy: 0.9710['S', 'R', 'P']
39/55 [===================>..........] - ETA: 2s - loss: 0.8310 - accuracy: 0.9709['S', 'R', 'P']
40/55 [===================>..........] - ETA: 2s - loss: 0.8301 - accuracy: 0.9708['S', 'R', 'P']
41/55 [====================>.........] - ETA: 2s - loss: 0.8282 - accuracy: 0.9716['S', 'R', 'P']
42/55 [====================>.........] - ETA: 1s - loss: 0.8269 - accuracy: 0.9722['S', 'R', 'P']
43/55 [=====================>........] - ETA: 1s - loss: 0.8269 - accuracy: 0.9722['S', 'R', 'P']
44/55 [=====================>........] - ETA: 1s - loss: 0.8259 - accuracy: 0.9721['S', 'R', 'P']
45/55 [======================>.......] - ETA: 1s - loss: 0.8266 - accuracy: 0.9720['S', 'R', 'P']
46/55 [======================>.......] - ETA: 1s - loss: 0.8251 - accuracy: 0.9726['S', 'R', 'P']
47/55 [=======================>......] - ETA: 1s - loss: 0.8232 - accuracy: 0.9732['S', 'R', 'P']
48/55 [=======================>......] - ETA: 1s - loss: 0.8225 - accuracy: 0.9738['S', 'R', 'P']
49/55 [========================>.....] - ETA: 0s - loss: 0.8233 - accuracy: 0.9730['S', 'R', 'P']
50/55 [========================>.....] - ETA: 0s - loss: 0.8226 - accuracy: 0.9736['S', 'R', 'P']
51/55 [=========================>....] - ETA: 0s - loss: 0.8212 - accuracy: 0.9741['S', 'R', 'P']
52/55 [=========================>....] - ETA: 0s - loss: 0.8205 - accuracy: 0.9740['S', 'R', 'P']
53/55 [==========================>...] - ETA: 0s - loss: 0.8231 - accuracy: 0.9733['S', 'R', 'P']
54/55 [==========================>...] - ETA: 0s - loss: 0.8217 - accuracy: 0.9738['S', 'R', 'P']
55/55 [============================>.] - ETA: 0s - loss: 0.8221 - accuracy: 0.9731['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
```

```
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
55/55 [==============================] - 10s 177ms/step - loss: 0.8221 - accuracy: 0.9731 - val_loss: 0.8879 - val_accuracy: 0.9
613
Epoch 17/20
['S', 'R', 'P']
 1/55 [..............................] - ETA: 7s - loss: 0.8277 - accuracy: 0.9688['S', 'R', 'P']
 2/55 [>.............................] - ETA: 5s - loss: 0.8067 - accuracy: 0.9844['S', 'R', 'P']
 3/55 [>.............................] - ETA: 6s - loss: 0.7851 - accuracy: 0.9896['S', 'R', 'P']
 4/55 [=>............................] - ETA: 5s - loss: 0.8120 - accuracy: 0.9766['S', 'R', 'P']
 5/55 [=>............................] - ETA: 5s - loss: 0.7967 - accuracy: 0.9812['S', 'R', 'P']
 6/55 [==>...........................] - ETA: 5s - loss: 0.7954 - accuracy: 0.9792['S', 'R', 'P']
 7/55 [==>...........................] - ETA: 5s - loss: 0.7935 - accuracy: 0.9821['S', 'R', 'P']
 8/55 [===>..........................] - ETA: 5s - loss: 0.7876 - accuracy: 0.9844['S', 'R', 'P']
 9/55 [===>..........................] - ETA: 5s - loss: 0.8082 - accuracy: 0.9792['S', 'R', 'P']
10/55 [====>.........................] - ETA: 5s - loss: 0.8016 - accuracy: 0.9812['S', 'R', 'P']
11/55 [=====>........................] - ETA: 5s - loss: 0.8083 - accuracy: 0.9744['S', 'R', 'P']
12/55 [=====>........................] - ETA: 5s - loss: 0.8040 - accuracy: 0.9766['S', 'R', 'P']
13/55 [======>.......................] - ETA: 4s - loss: 0.8019 - accuracy: 0.9778['S', 'R', 'P']
14/55 [======>.......................] - ETA: 4s - loss: 0.8005 - accuracy: 0.9771['S', 'R', 'P']
15/55 [=======>......................] - ETA: 4s - loss: 0.7973 - accuracy: 0.9787['S', 'R', 'P']
16/55 [=======>......................] - ETA: 4s - loss: 0.8033 - accuracy: 0.9760['S', 'R', 'P']
17/55 [========>.....................] - ETA: 4s - loss: 0.8134 - accuracy: 0.9700['S', 'R', 'P']
18/55 [========>.....................] - ETA: 4s - loss: 0.8098 - accuracy: 0.9699['S', 'R', 'P']
19/55 [=========>....................] - ETA: 4s - loss: 0.8082 - accuracy: 0.9715['S', 'R', 'P']
20/55 [=========>....................] - ETA: 3s - loss: 0.8057 - accuracy: 0.9730['S', 'R', 'P']
21/55 [==========>...................] - ETA: 3s - loss: 0.8056 - accuracy: 0.9743['S', 'R', 'P']
22/55 [==========>...................] - ETA: 3s - loss: 0.8022 - accuracy: 0.9755['S', 'R', 'P']
23/55 [==========>...................] - ETA: 3s - loss: 0.8032 - accuracy: 0.9738['S', 'R', 'P']
24/55 [===========>..................] - ETA: 3s - loss: 0.8023 - accuracy: 0.9736['S', 'R', 'P']
25/55 [===========>..................] - ETA: 3s - loss: 0.7998 - accuracy: 0.9747['S', 'R', 'P']
26/55 [============>.................] - ETA: 3s - loss: 0.8025 - accuracy: 0.9732['S', 'R', 'P']
27/55 [============>.................] - ETA: 3s - loss: 0.8035 - accuracy: 0.9730['S', 'R', 'P']
28/55 [=============>................] - ETA: 3s - loss: 0.8007 - accuracy: 0.9740['S', 'R', 'P']
29/55 [=============>................] - ETA: 2s - loss: 0.8073 - accuracy: 0.9695['S', 'R', 'P']
30/55 [==============>...............] - ETA: 2s - loss: 0.8096 - accuracy: 0.9694['S', 'R', 'P']
31/55 [==============>...............] - ETA: 2s - loss: 0.8082 - accuracy: 0.9704['S', 'R', 'P']
32/55 [===============>..............] - ETA: 2s - loss: 0.8111 - accuracy: 0.9684['S', 'R', 'P']
```

```
33/55 [=================>............] - ETA: 2s - loss: 0.8094 - accuracy: 0.9684['S', 'R', 'P']
34/55 [=================>............] - ETA: 2s - loss: 0.8131 - accuracy: 0.9666['S', 'R', 'P']
35/55 [==================>...........] - ETA: 2s - loss: 0.8120 - accuracy: 0.9675['S', 'R', 'P']
36/55 [==================>...........] - ETA: 2s - loss: 0.8110 - accuracy: 0.9676['S', 'R', 'P']
37/55 [==================>...........] - ETA: 2s - loss: 0.8110 - accuracy: 0.9668['S', 'R', 'P']
38/55 [==================>...........] - ETA: 1s - loss: 0.8096 - accuracy: 0.9676['S', 'R', 'P']
39/55 [===================>..........] - ETA: 1s - loss: 0.8135 - accuracy: 0.9677['S', 'R', 'P']
40/55 [===================>..........] - ETA: 1s - loss: 0.8132 - accuracy: 0.9677['S', 'R', 'P']
41/55 [====================>.........] - ETA: 1s - loss: 0.8126 - accuracy: 0.9677['S', 'R', 'P']
42/55 [====================>.........] - ETA: 1s - loss: 0.8140 - accuracy: 0.9670['S', 'R', 'P']
43/55 [====================>.........] - ETA: 1s - loss: 0.8138 - accuracy: 0.9678['S', 'R', 'P']
44/55 [=====================>........] - ETA: 1s - loss: 0.8143 - accuracy: 0.9671['S', 'R', 'P']
45/55 [=====================>........] - ETA: 1s - loss: 0.8140 - accuracy: 0.9671['S', 'R', 'P']
46/55 [=====================>........] - ETA: 1s - loss: 0.8148 - accuracy: 0.9671['S', 'R', 'P']
47/55 [======================>.......] - ETA: 0s - loss: 0.8134 - accuracy: 0.9678['S', 'R', 'P']
48/55 [======================>.......] - ETA: 0s - loss: 0.8124 - accuracy: 0.9679['S', 'R', 'P']
49/55 [======================>.......] - ETA: 0s - loss: 0.8123 - accuracy: 0.9679['S', 'R', 'P']
50/55 [=======================>......] - ETA: 0s - loss: 0.8115 - accuracy: 0.9685['S', 'R', 'P']
51/55 [=======================>......] - ETA: 0s - loss: 0.8144 - accuracy: 0.9679['S', 'R', 'P']
52/55 [========================>.....] - ETA: 0s - loss: 0.8142 - accuracy: 0.9679['S', 'R', 'P']
53/55 [========================>.....] - ETA: 0s - loss: 0.8145 - accuracy: 0.9674['S', 'R', 'P']
54/55 [========================>.....] - ETA: 0s - loss: 0.8145 - accuracy: 0.9668['S', 'R', 'P']
55/55 [==============================] - ETA: 0s - loss: 0.8137 - accuracy: 0.9674['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
55/55 [==============================] - 8s 141ms/step - loss: 0.8137 - accuracy: 0.9674 - val_loss: 0.8924 - val_accuracy: 0.92
26
Epoch 18/20
['S', 'R', 'P']
 1/55 [..............................] - ETA: 7s - loss: 0.7913 - accuracy: 0.9688['S', 'R', 'P']
 2/55 [>.............................] - ETA: 6s - loss: 0.7577 - accuracy: 0.9844['S', 'R', 'P']
 3/55 [>.............................] - ETA: 5s - loss: 0.7446 - accuracy: 0.9896['S', 'R', 'P']
 4/55 [=>............................] - ETA: 5s - loss: 0.7510 - accuracy: 0.9922['S', 'R', 'P']
```

```
 5/55 [=>............................] - ETA: 5s - loss: 0.7597 - accuracy: 0.9875['S', 'R', 'P']
 6/55 [==>...........................] - ETA: 5s - loss: 0.7611 - accuracy: 0.9834['S', 'R', 'P']
 7/55 [==>...........................] - ETA: 5s - loss: 0.7650 - accuracy: 0.9812['S', 'R', 'P']
 8/55 [===>..........................] - ETA: 5s - loss: 0.7691 - accuracy: 0.9796['S', 'R', 'P']
 9/55 [===>..........................] - ETA: 5s - loss: 0.7660 - accuracy: 0.9819['S', 'R', 'P']
10/55 [====>.........................] - ETA: 5s - loss: 0.7701 - accuracy: 0.9806['S', 'R', 'P']
11/55 [=====>........................] - ETA: 4s - loss: 0.7714 - accuracy: 0.9795['S', 'R', 'P']
12/55 [=====>........................] - ETA: 4s - loss: 0.7724 - accuracy: 0.9786['S', 'R', 'P']
13/55 [======>.......................] - ETA: 4s - loss: 0.7754 - accuracy: 0.9778['S', 'R', 'P']
14/55 [======>.......................] - ETA: 4s - loss: 0.7760 - accuracy: 0.9771['S', 'R', 'P']
15/55 [=======>......................] - ETA: 4s - loss: 0.7769 - accuracy: 0.9787['S', 'R', 'P']
16/55 [=======>......................] - ETA: 4s - loss: 0.7736 - accuracy: 0.9800['S', 'R', 'P']
17/55 [========>.....................] - ETA: 4s - loss: 0.7792 - accuracy: 0.9794['S', 'R', 'P']
18/55 [========>.....................] - ETA: 4s - loss: 0.7868 - accuracy: 0.9770['S', 'R', 'P']
19/55 [=========>....................] - ETA: 4s - loss: 0.7906 - accuracy: 0.9749['S', 'R', 'P']
20/55 [=========>....................] - ETA: 4s - loss: 0.7913 - accuracy: 0.9730['S', 'R', 'P']
21/55 [==========>...................] - ETA: 4s - loss: 0.7917 - accuracy: 0.9743['S', 'R', 'P']
22/55 [==========>...................] - ETA: 4s - loss: 0.7939 - accuracy: 0.9740['S', 'R', 'P']
23/55 [==========>...................] - ETA: 4s - loss: 0.7929 - accuracy: 0.9738['S', 'R', 'P']
24/55 [===========>..................] - ETA: 4s - loss: 0.7920 - accuracy: 0.9749['S', 'R', 'P']
25/55 [===========>..................] - ETA: 4s - loss: 0.7915 - accuracy: 0.9747['S', 'R', 'P']
26/55 [============>.................] - ETA: 3s - loss: 0.7926 - accuracy: 0.9732['S', 'R', 'P']
27/55 [============>.................] - ETA: 3s - loss: 0.7912 - accuracy: 0.9730['S', 'R', 'P']
28/55 [=============>................] - ETA: 3s - loss: 0.7902 - accuracy: 0.9729['S', 'R', 'P']
29/55 [=============>................] - ETA: 3s - loss: 0.7946 - accuracy: 0.9716['S', 'R', 'P']
30/55 [==============>...............] - ETA: 3s - loss: 0.7946 - accuracy: 0.9715['S', 'R', 'P']
31/55 [==============>...............] - ETA: 3s - loss: 0.7960 - accuracy: 0.9694['S', 'R', 'P']
32/55 [===============>..............] - ETA: 3s - loss: 0.7993 - accuracy: 0.9684['S', 'R', 'P']
33/55 [===============>..............] - ETA: 2s - loss: 0.8008 - accuracy: 0.9684['S', 'R', 'P']
34/55 [================>.............] - ETA: 2s - loss: 0.8000 - accuracy: 0.9694['S', 'R', 'P']
35/55 [================>.............] - ETA: 2s - loss: 0.8028 - accuracy: 0.9684['S', 'R', 'P']
36/55 [=================>............] - ETA: 2s - loss: 0.8024 - accuracy: 0.9684['S', 'R', 'P']
37/55 [=================>............] - ETA: 2s - loss: 0.8015 - accuracy: 0.9693['S', 'R', 'P']
38/55 [==================>...........] - ETA: 2s - loss: 0.7996 - accuracy: 0.9701['S', 'R', 'P']
39/55 [==================>...........] - ETA: 2s - loss: 0.8000 - accuracy: 0.9693['S', 'R', 'P']
40/55 [===================>..........] - ETA: 2s - loss: 0.7981 - accuracy: 0.9701['S', 'R', 'P']
41/55 [===================>..........] - ETA: 1s - loss: 0.7993 - accuracy: 0.9685['S', 'R', 'P']
42/55 [====================>.........] - ETA: 1s - loss: 0.8028 - accuracy: 0.9677['S', 'R', 'P']
43/55 [====================>.........] - ETA: 1s - loss: 0.8035 - accuracy: 0.9685['S', 'R', 'P']
44/55 [=====================>........] - ETA: 1s - loss: 0.8027 - accuracy: 0.9685['S', 'R', 'P']
45/55 [=====================>........] - ETA: 1s - loss: 0.8013 - accuracy: 0.9692['S', 'R', 'P']
46/55 [======================>.......] - ETA: 1s - loss: 0.7997 - accuracy: 0.9699['S', 'R', 'P']
47/55 [======================>.......] - ETA: 1s - loss: 0.8022 - accuracy: 0.9685['S', 'R', 'P']
48/55 [=======================>......] - ETA: 0s - loss: 0.8017 - accuracy: 0.9692['S', 'R', 'P']
```

```
49/55 [=========================>....] - ETA: 0s - loss: 0.8013 - accuracy: 0.9692['S', 'R', 'P']
50/55 [=========================>...] - ETA: 0s - loss: 0.8022 - accuracy: 0.9679['S', 'R', 'P']
51/55 [=========================>...] - ETA: 0s - loss: 0.8022 - accuracy: 0.9679['S', 'R', 'P']
52/55 [==========================>..] - ETA: 0s - loss: 0.8056 - accuracy: 0.9667['S', 'R', 'P']
53/55 [==========================>..] - ETA: 0s - loss: 0.8042 - accuracy: 0.9674['S', 'R', 'P']
54/55 [==========================>.] - ETA: 0s - loss: 0.8074 - accuracy: 0.9651['S', 'R', 'P']
55/55 [==============================] - ETA: 0s - loss: 0.8062 - accuracy: 0.9657['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
55/55 [==============================] - 9s 172ms/step - loss: 0.8062 - accuracy: 0.9657 - val_loss: 0.8728 - val_accuracy: 0.93
17
Epoch 19/20
['S', 'R', 'P']
 1/55 [..............................] - ETA: 8s - loss: 0.7248 - accuracy: 1.0000['S', 'R', 'P']
 2/55 [>.............................] - ETA: 5s - loss: 0.7466 - accuracy: 0.9844['S', 'R', 'P']
 3/55 [>.............................] - ETA: 5s - loss: 0.7368 - accuracy: 0.9896['S', 'R', 'P']
 4/55 [=>............................] - ETA: 5s - loss: 0.7368 - accuracy: 0.9922['S', 'R', 'P']
 5/55 [=>............................] - ETA: 6s - loss: 0.7426 - accuracy: 0.9875['S', 'R', 'P']
 6/55 [==>...........................] - ETA: 5s - loss: 0.7394 - accuracy: 0.9896['S', 'R', 'P']
 7/55 [==>...........................] - ETA: 5s - loss: 0.7454 - accuracy: 0.9866['S', 'R', 'P']
 8/55 [===>..........................] - ETA: 5s - loss: 0.7571 - accuracy: 0.9805['S', 'R', 'P']
 9/55 [===>..........................] - ETA: 5s - loss: 0.7601 - accuracy: 0.9792['S', 'R', 'P']
10/55 [====>.........................] - ETA: 5s - loss: 0.7544 - accuracy: 0.9812['S', 'R', 'P']
11/55 [=====>........................] - ETA: 5s - loss: 0.7506 - accuracy: 0.9830['S', 'R', 'P']
12/55 [=====>........................] - ETA: 5s - loss: 0.7565 - accuracy: 0.9792['S', 'R', 'P']
13/55 [======>.......................] - ETA: 5s - loss: 0.7724 - accuracy: 0.9760['S', 'R', 'P']
14/55 [======>.......................] - ETA: 4s - loss: 0.7781 - accuracy: 0.9710['S', 'R', 'P']
15/55 [=======>......................] - ETA: 4s - loss: 0.7754 - accuracy: 0.9708['S', 'R', 'P']
16/55 [=======>......................] - ETA: 4s - loss: 0.7780 - accuracy: 0.9688['S', 'R', 'P']
17/55 [========>.....................] - ETA: 4s - loss: 0.7732 - accuracy: 0.9706['S', 'R', 'P']
18/55 [========>.....................] - ETA: 4s - loss: 0.7780 - accuracy: 0.9688['S', 'R', 'P']
19/55 [=========>....................] - ETA: 4s - loss: 0.7802 - accuracy: 0.9655['S', 'R', 'P']
20/55 [=========>....................] - ETA: 4s - loss: 0.7847 - accuracy: 0.9641['S', 'R', 'P']
```

```
21/55 [==========>...................] - ETA: 4s - loss: 0.7856 - accuracy: 0.9658['S', 'R', 'P']
22/55 [==========>...................] - ETA: 3s - loss: 0.7827 - accuracy: 0.9659['S', 'R', 'P']
23/55 [==========>...................] - ETA: 3s - loss: 0.7857 - accuracy: 0.9633['S', 'R', 'P']
24/55 [===========>..................] - ETA: 3s - loss: 0.7882 - accuracy: 0.9609['S', 'R', 'P']
25/55 [===========>..................] - ETA: 3s - loss: 0.7845 - accuracy: 0.9625['S', 'R', 'P']
26/55 [============>.................] - ETA: 3s - loss: 0.7824 - accuracy: 0.9627['S', 'R', 'P']
27/55 [============>.................] - ETA: 3s - loss: 0.7890 - accuracy: 0.9606['S', 'R', 'P']
28/55 [=============>................] - ETA: 3s - loss: 0.7907 - accuracy: 0.9587['S', 'R', 'P']
29/55 [=============>................] - ETA: 3s - loss: 0.7983 - accuracy: 0.9558['S', 'R', 'P']
30/55 [==============>...............] - ETA: 2s - loss: 0.7996 - accuracy: 0.9563['S', 'R', 'P']
31/55 [==============>...............] - ETA: 2s - loss: 0.7968 - accuracy: 0.9577['S', 'R', 'P']
32/55 [===============>..............] - ETA: 2s - loss: 0.7963 - accuracy: 0.9580['S', 'R', 'P']
33/55 [===============>..............] - ETA: 2s - loss: 0.7945 - accuracy: 0.9583['S', 'R', 'P']
34/55 [================>.............] - ETA: 2s - loss: 0.7974 - accuracy: 0.9577['S', 'R', 'P']
35/55 [================>.............] - ETA: 2s - loss: 0.7967 - accuracy: 0.9580['S', 'R', 'P']
36/55 [=================>............] - ETA: 2s - loss: 0.7989 - accuracy: 0.9575['S', 'R', 'P']
37/55 [=================>............] - ETA: 2s - loss: 0.7982 - accuracy: 0.9578['S', 'R', 'P']
38/55 [==================>...........] - ETA: 2s - loss: 0.7977 - accuracy: 0.9581['S', 'R', 'P']
39/55 [==================>...........] - ETA: 1s - loss: 0.7986 - accuracy: 0.9583['S', 'R', 'P']
40/55 [===================>..........] - ETA: 1s - loss: 0.7992 - accuracy: 0.9578['S', 'R', 'P']
41/55 [===================>..........] - ETA: 1s - loss: 0.8011 - accuracy: 0.9570['S', 'R', 'P']
42/55 [====================>.........] - ETA: 1s - loss: 0.7992 - accuracy: 0.9580['S', 'R', 'P']
43/55 [====================>.........] - ETA: 1s - loss: 0.8009 - accuracy: 0.9582['S', 'R', 'P']
44/55 [=====================>........] - ETA: 1s - loss: 0.8031 - accuracy: 0.9585['S', 'R', 'P']
45/55 [=====================>........] - ETA: 1s - loss: 0.8026 - accuracy: 0.9587['S', 'R', 'P']
46/55 [======================>.......] - ETA: 1s - loss: 0.8013 - accuracy: 0.9589['S', 'R', 'P']
47/55 [======================>.......] - ETA: 0s - loss: 0.8005 - accuracy: 0.9591['S', 'R', 'P']
48/55 [=======================>......] - ETA: 0s - loss: 0.7998 - accuracy: 0.9600['S', 'R', 'P']
49/55 [=======================>......] - ETA: 0s - loss: 0.7995 - accuracy: 0.9602['S', 'R', 'P']
50/55 [========================>.....] - ETA: 0s - loss: 0.7980 - accuracy: 0.9610['S', 'R', 'P']
51/55 [========================>.....] - ETA: 0s - loss: 0.7972 - accuracy: 0.9611['S', 'R', 'P']
52/55 [=========================>....] - ETA: 0s - loss: 0.7993 - accuracy: 0.9607['S', 'R', 'P']
53/55 [=========================>....] - ETA: 0s - loss: 0.7989 - accuracy: 0.9614['S', 'R', 'P']
54/55 [==========================>...] - ETA: 0s - loss: 0.7984 - accuracy: 0.9621['S', 'R', 'P']
55/55 [==============================] - ETA: 0s - loss: 0.7970 - accuracy: 0.9628['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
```

```
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
55/55 [==============================] - 8s 148ms/step - loss: 0.7970 - accuracy: 0.9628 - val_loss: 0.8433 - val_accuracy: 0.93
39
Epoch 20/20
['S', 'R', 'P']
 1/55 [..............................] - ETA: 7s - loss: 0.7302 - accuracy: 1.0000['S', 'R', 'P']
 2/55 [>.............................] - ETA: 6s - loss: 0.7875 - accuracy: 0.9531['S', 'R', 'P']
 3/55 [>.............................] - ETA: 6s - loss: 0.7747 - accuracy: 0.9688['S', 'R', 'P']
 4/55 [=>............................] - ETA: 5s - loss: 0.7593 - accuracy: 0.9766['S', 'R', 'P']
 5/55 [=>............................] - ETA: 5s - loss: 0.7725 - accuracy: 0.9750['S', 'R', 'P']
 6/55 [==>...........................] - ETA: 5s - loss: 0.7802 - accuracy: 0.9740['S', 'R', 'P']
 7/55 [==>...........................] - ETA: 5s - loss: 0.7694 - accuracy: 0.9777['S', 'R', 'P']
 8/55 [===>..........................] - ETA: 5s - loss: 0.7697 - accuracy: 0.9805['S', 'R', 'P']
 9/55 [===>..........................] - ETA: 5s - loss: 0.7811 - accuracy: 0.9722['S', 'R', 'P']
10/55 [====>.........................] - ETA: 5s - loss: 0.7780 - accuracy: 0.9719['S', 'R', 'P']
11/55 [=====>........................] - ETA: 5s - loss: 0.7741 - accuracy: 0.9716['S', 'R', 'P']
12/55 [=====>........................] - ETA: 5s - loss: 0.7709 - accuracy: 0.9740['S', 'R', 'P']
13/55 [======>.......................] - ETA: 5s - loss: 0.7644 - accuracy: 0.9760['S', 'R', 'P']
14/55 [======>.......................] - ETA: 5s - loss: 0.7639 - accuracy: 0.9754['S', 'R', 'P']
15/55 [=======>......................] - ETA: 5s - loss: 0.7629 - accuracy: 0.9750['S', 'R', 'P']
16/55 [=======>......................] - ETA: 4s - loss: 0.7742 - accuracy: 0.9727['S', 'R', 'P']
17/55 [========>.....................] - ETA: 4s - loss: 0.7756 - accuracy: 0.9706['S', 'R', 'P']
18/55 [========>.....................] - ETA: 4s - loss: 0.7736 - accuracy: 0.9705['S', 'R', 'P']
19/55 [=========>....................] - ETA: 4s - loss: 0.7689 - accuracy: 0.9720['S', 'R', 'P']
20/55 [=========>....................] - ETA: 4s - loss: 0.7651 - accuracy: 0.9734['S', 'R', 'P']
21/55 [==========>...................] - ETA: 4s - loss: 0.7621 - accuracy: 0.9747['S', 'R', 'P']
22/55 [==========>...................] - ETA: 4s - loss: 0.7593 - accuracy: 0.9759['S', 'R', 'P']
23/55 [==========>...................] - ETA: 4s - loss: 0.7651 - accuracy: 0.9728['S', 'R', 'P']
24/55 [===========>..................] - ETA: 4s - loss: 0.7645 - accuracy: 0.9727['S', 'R', 'P']
25/55 [===========>..................] - ETA: 4s - loss: 0.7619 - accuracy: 0.9737['S', 'R', 'P']
26/55 [============>.................] - ETA: 3s - loss: 0.7623 - accuracy: 0.9748['S', 'R', 'P']
27/55 [============>.................] - ETA: 3s - loss: 0.7644 - accuracy: 0.9734['S', 'R', 'P']
28/55 [=============>................] - ETA: 3s - loss: 0.7635 - accuracy: 0.9732['S', 'R', 'P']
29/55 [=============>................] - ETA: 3s - loss: 0.7606 - accuracy: 0.9741['S', 'R', 'P']
30/55 [==============>...............] - ETA: 3s - loss: 0.7632 - accuracy: 0.9729['S', 'R', 'P']
31/55 [==============>...............] - ETA: 3s - loss: 0.7658 - accuracy: 0.9728['S', 'R', 'P']
32/55 [===============>..............] - ETA: 3s - loss: 0.7645 - accuracy: 0.9736['S', 'R', 'P']
33/55 [===============>..............] - ETA: 3s - loss: 0.7640 - accuracy: 0.9735['S', 'R', 'P']
34/55 [================>.............] - ETA: 2s - loss: 0.7614 - accuracy: 0.9743['S', 'R', 'P']
35/55 [================>.............] - ETA: 2s - loss: 0.7609 - accuracy: 0.9732['S', 'R', 'P']
36/55 [================>.............] - ETA: 2s - loss: 0.7608 - accuracy: 0.9731['S', 'R', 'P']
```

```
37/55 [===================>.........] - ETA: 2s - loss: 0.7602 - accuracy: 0.9738['S', 'R', 'P']
38/55 [===================>.........] - ETA: 2s - loss: 0.7609 - accuracy: 0.9737['S', 'R', 'P']
39/55 [====================>........] - ETA: 2s - loss: 0.7587 - accuracy: 0.9744['S', 'R', 'P']
40/55 [====================>........] - ETA: 2s - loss: 0.7579 - accuracy: 0.9750['S', 'R', 'P']
41/55 [=====================>.......] - ETA: 1s - loss: 0.7567 - accuracy: 0.9748['S', 'R', 'P']
42/55 [=====================>.......] - ETA: 1s - loss: 0.7549 - accuracy: 0.9754['S', 'R', 'P']
43/55 [======================>......] - ETA: 1s - loss: 0.7557 - accuracy: 0.9746['S', 'R', 'P']
44/55 [======================>......] - ETA: 1s - loss: 0.7543 - accuracy: 0.9744['S', 'R', 'P']
45/55 [======================>......] - ETA: 1s - loss: 0.7530 - accuracy: 0.9750['S', 'R', 'P']
46/55 [=======================>.....] - ETA: 1s - loss: 0.7516 - accuracy: 0.9749['S', 'R', 'P']
47/55 [=======================>.....] - ETA: 1s - loss: 0.7501 - accuracy: 0.9754['S', 'R', 'P']
48/55 [========================>....] - ETA: 0s - loss: 0.7507 - accuracy: 0.9746['S', 'R', 'P']
49/55 [========================>....] - ETA: 0s - loss: 0.7518 - accuracy: 0.9732['S', 'R', 'P']
50/55 [=========================>...] - ETA: 0s - loss: 0.7533 - accuracy: 0.9725['S', 'R', 'P']
51/55 [=========================>...] - ETA: 0s - loss: 0.7545 - accuracy: 0.9722['S', 'R', 'P']
52/55 [==========================>..] - ETA: 0s - loss: 0.7530 - accuracy: 0.9728['S', 'R', 'P']
53/55 [==========================>..] - ETA: 0s - loss: 0.7525 - accuracy: 0.9727['S', 'R', 'P']
54/55 [==========================>.] - ETA: 0s - loss: 0.7541 - accuracy: 0.9715['S', 'R', 'P']
55/55 [==============================] - ETA: 0s - loss: 0.7550 - accuracy: 0.9708['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
['S', 'R', 'P']
55/55 [==============================] - 9s 172ms/step - loss: 0.7550 - accuracy: 0.9708 - val_loss: 0.8996 - val_accuracy: 0.93
17
['S', 'R', 'P']
['S', 'R', 'P']
 1/14 [=>............................] - ETA: 1s - loss: 0.9064 - accuracy: 0.9062['S', 'R', 'P']
 2/14 [===>..........................] - ETA: 1s - loss: 0.9111 - accuracy: 0.9219['S', 'R', 'P']
 3/14 [=====>........................] - ETA: 1s - loss: 0.9573 - accuracy: 0.9062['S', 'R', 'P']
 4/14 [=======>......................] - ETA: 1s - loss: 0.9214 - accuracy: 0.9219['S', 'R', 'P']
 5/14 [=========>....................] - ETA: 1s - loss: 0.9664 - accuracy: 0.9062['S', 'R', 'P']
 6/14 [===========>..................] - ETA: 0s - loss: 0.9423 - accuracy: 0.9115['S', 'R', 'P']
 7/14 [==============>...............] - ETA: 0s - loss: 0.9569 - accuracy: 0.9107['S', 'R', 'P']
 8/14 [================>.............] - ETA: 0s - loss: 0.9345 - accuracy: 0.9180['S', 'R', 'P']
```

```
 9/14 [==================>............] - ETA: 0s - loss: 0.9290 - accuracy: 0.9236['S', 'R', 'P']
10/14 [====================>..........] - ETA: 0s - loss: 0.9105 - accuracy: 0.9281['S', 'R', 'P']
11/14 [======================>........] - ETA: 0s - loss: 0.9183 - accuracy: 0.9290['S', 'R', 'P']
12/14 [========================>.....] - ETA: 0s - loss: 0.9180 - accuracy: 0.9271['S', 'R', 'P']
13/14 [==========================>...] - ETA: 0s - loss: 0.9108 - accuracy: 0.9279['S', 'R', 'P']
14/14 [==============================] - 2s 118ms/step - loss: 0.8996 - accuracy: 0.9317
Validation Loss: 0.8995972275733948, Validation Accuracy: 0.9316628575325012
```

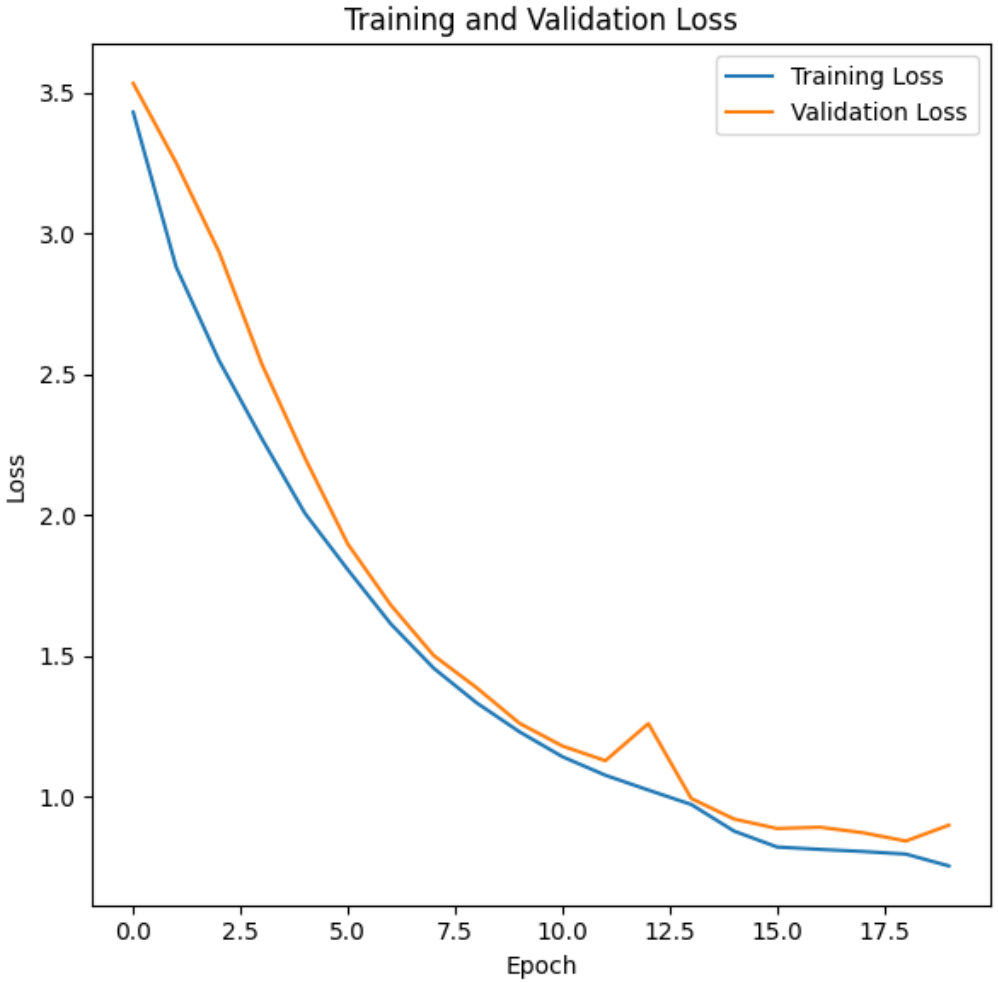
(A) Report the training time (use code to do this).

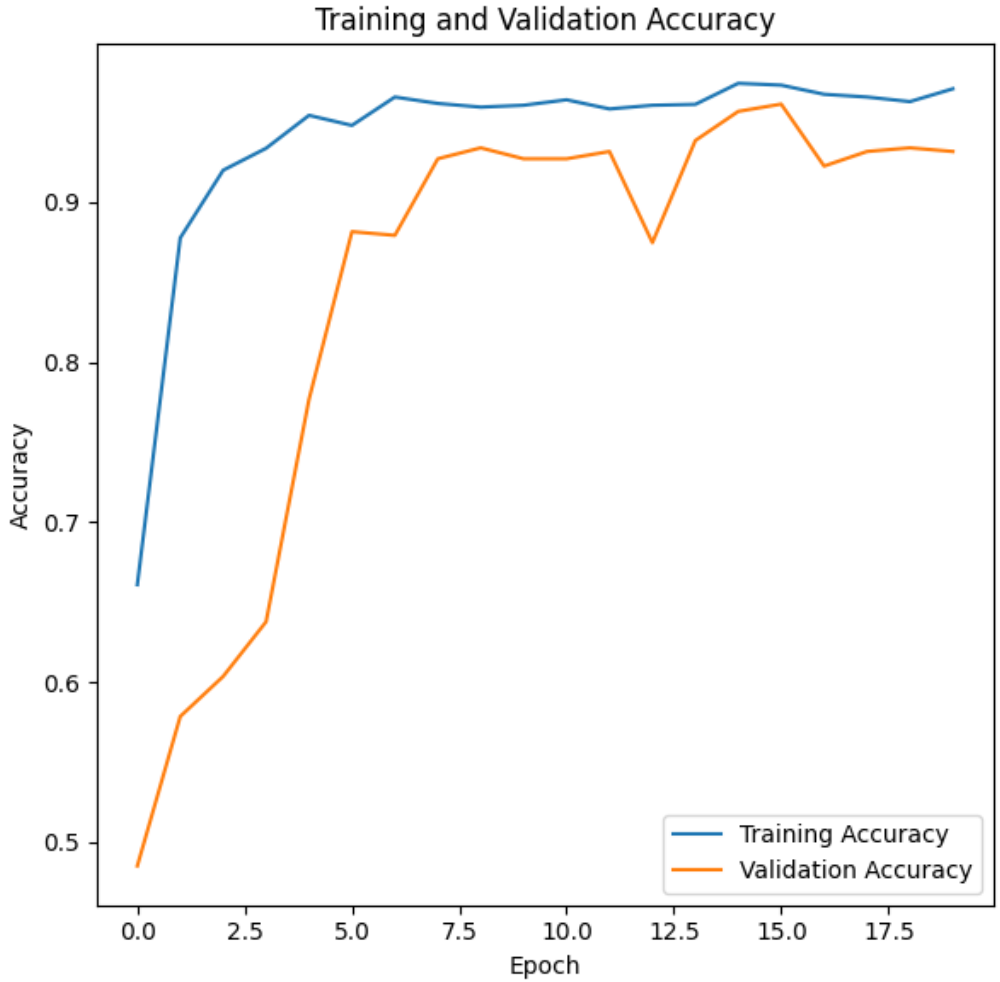
In [59]:
```
# Calculating and reporting the training time
training_time_seconds = end_time - start_time
print(f"The time it took to train the model (in seconds) is: {training_time_seconds:.2f} seconds")
```

```
The time it took to train the model (in seconds) is: 193.41 seconds
```

(B) Plot training and validation loss and accuracy as a function of training epochs.

In [60]:
```
# Plot training and validation loss
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

# Plot training and validation accuracy
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

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





Answer -

1. As seen from the graphs above -

- The training and validation loss decreases as more epochs are trained.
- The training and validation accuracy of the model increases as more models are trained.

2. As seen from the above graph, the model overfits the data a bit. Overfitting could be resolved by further tuning the hyperparameters.

(C) How many parameters does the network have? How many of those parameters are bias parameters?

In [61]:
```
model.summary()
print("="*100)

total_parameters = model.count_params()
print(f'\nTotal number of parameters: {total_parameters}')

# Extracting and displaying total bias parameters
total_bias_parameters = 0
for layer in model.layers:
    if hasattr(layer, 'get_weights'):
        weights = layer.get_weights()
        if len(weights) > 1:  # Check if the layer has bias parameters
            total_bias_parameters += weights[1].size

print(f"Total number of Bias Parameters: {total_bias_parameters}")
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_2 (Flatten)         (None, 180000)            0

 batch_normalization_6 (Bat  (None, 180000)            720000
 chNormalization)

 dense_6 (Dense)             (None, 100)               18000100

 batch_normalization_7 (Bat  (None, 100)               400
 chNormalization)

 dense_7 (Dense)             (None, 50)                5050

 batch_normalization_8 (Bat  (None, 50)                200
 chNormalization)

 intermediate_layer (Dense)  (None, 8)                 408

 dense_8 (Dense)             (None, 3)                 27

=================================================================
Total params: 18726185 (71.43 MB)
Trainable params: 18365885 (70.06 MB)
Non-trainable params: 360300 (1.37 MB)
_________________________________________________________________
================================================================================================================

Total number of parameters: 18726185
Total number of Bias Parameters: 180311
```

In [62]:

```python
# Displaying the number of Bias Parameters per layer
from tensorflow.keras.models import load_model

print(f"For every layer the model has the following bias parameters - ")
# Iterate through layers and print bias parameters
for layer in model.layers:
    if hasattr(layer, 'get_weights') and len(layer.get_weights()) > 0:
        all_weights = layer.get_weights()
        biases = all_weights[-1]  # Last element is assumed to be biases
```

```
        if biases is not None:
            print(f'\tLayer: {layer.name}, Bias parameters: {biases.shape[0]}')
```

```
For every layer the model has the following bias parameters -
        Layer: batch_normalization_6, Bias parameters: 180000
        Layer: dense_6, Bias parameters: 100
        Layer: batch_normalization_7, Bias parameters: 100
        Layer: dense_7, Bias parameters: 50
        Layer: batch_normalization_8, Bias parameters: 50
        Layer: intermediate_layer, Bias parameters: 8
        Layer: dense_8, Bias parameters: 3
```

(D) - Not required for Personal Dataset