uw3tsfgyj

November 28, 2024

1 Cat and Dog Dataset

The **Cat and Dog Dataset** is a labeled dataset of over **10,000 images**, divided into training and testing subsets, designed for binary classification tasks to distinguish between cats and dogs. It is ideal for beginner and advanced deep learning projects, particularly for image classification using convolutional neural networks (CNNs).

1.1 Key Features:

- **Number of Images**: 10,000 images in total.
- **File Format**: All images are in `.jpg` format.
- **Folder Structure**:
  - **Training Set**: Contains separate folders for `cats` and `dogs`.
  - **Test Set**: Contains separate folders for `cats` and `dogs`.
- **Dataset Size**: 228.46 MB.

1.2 Dataset URL: Cat and Dog Dataset on Kaggle

2 1

[46]:
```
# General imports
import os
import time
import warnings
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image

# TensorFlow/Keras imports
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,␣
↪Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical

# Scikit-learn imports
from sklearn.model_selection import train_test_split
```

```
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE, LocallyLinearEmbedding, MDS
from sklearn.mixture import GaussianMixture
from sklearn.metrics import confusion_matrix, accuracy_score, silhouette_score
from sklearn.cluster import KMeans

# Visualization utilities
from matplotlib.offsetbox import OffsetImage, AnnotationBbox

# Suppress warnings
warnings.filterwarnings('ignore')
```

[47]:
```
import os
from PIL import Image

# Define a mapping for the classes (cats and dogs)
class_mapping = {
    "cats": "Cat",
    "dogs": "Dog"
}

# Path to your dataset (Update the path as per your dataset's location)
dataset_path = "./training_set/training_set"  # Replace with your actual␣
↪dataset path

# Data storage
data = []

# Function to resize, downsample, and convert to grayscale
def process_image(img_path, size=(100, 100), grayscale=False):
    try:
        # Open the image and convert to RGB (color image)
        img = Image.open(img_path).convert("RGB")

        # Resize the image to the target size
        img = img.resize(size)

        # Optionally convert to grayscale
        if grayscale:
            img = img.convert('L')  # Convert to grayscale ('L' mode)

        return img
    except Exception as e:
        print(f"Error processing image {img_path}: {e}")
        return None
```

[48]:

```python
# Iterate through the subdirectories for cats and dogs
for class_name in class_mapping:
    class_folder = os.path.join(dataset_path, class_name)  # Folder for each␣
 ↪class (cats/dogs)

    if os.path.exists(class_folder):
        count = 0
        for img_file in os.listdir(class_folder):
            if img_file.endswith(".jpg"):  # Ensure it's an image file
                # Build the full image path
                img_path = os.path.join(class_folder, img_file)

                # Process the image (resize, grayscale)
                img = process_image(img_path)

                if img:  # If image is processed successfully
                    # Store the image and its metadata
                    data.append({
                        "Image": img,
                        "Class": class_mapping[class_name],  # 'Cat' or 'Dog'
                        "Filename": img_file
                    })
                    print(f"Loaded: {img_file} | Class:␣
 ↪{class_mapping[class_name]}")
                    count += 1

                # Stop after 200 images from each class
                if count >= 200:
                    break
    else:
        print(f"Class folder {class_name} does not exist.")
```

```
Loaded: cat.1.jpg | Class: Cat
Loaded: cat.10.jpg | Class: Cat
Loaded: cat.100.jpg | Class: Cat
Loaded: cat.1000.jpg | Class: Cat
Loaded: cat.1001.jpg | Class: Cat
Loaded: cat.1002.jpg | Class: Cat
Loaded: cat.1003.jpg | Class: Cat
Loaded: cat.1004.jpg | Class: Cat
Loaded: cat.1005.jpg | Class: Cat
Loaded: cat.1006.jpg | Class: Cat
Loaded: cat.1007.jpg | Class: Cat
Loaded: cat.1008.jpg | Class: Cat
Loaded: cat.1009.jpg | Class: Cat
Loaded: cat.101.jpg | Class: Cat
Loaded: cat.1010.jpg | Class: Cat
```

```
Loaded: cat.1011.jpg | Class: Cat
Loaded: cat.1012.jpg | Class: Cat
Loaded: cat.1013.jpg | Class: Cat
Loaded: cat.1014.jpg | Class: Cat
Loaded: cat.1015.jpg | Class: Cat
Loaded: cat.1016.jpg | Class: Cat
Loaded: cat.1017.jpg | Class: Cat
Loaded: cat.1018.jpg | Class: Cat
Loaded: cat.1019.jpg | Class: Cat
Loaded: cat.102.jpg | Class: Cat
Loaded: cat.1020.jpg | Class: Cat
Loaded: cat.1021.jpg | Class: Cat
Loaded: cat.1022.jpg | Class: Cat
Loaded: cat.1023.jpg | Class: Cat
Loaded: cat.1024.jpg | Class: Cat
Loaded: cat.1025.jpg | Class: Cat
Loaded: cat.1026.jpg | Class: Cat
Loaded: cat.1027.jpg | Class: Cat
Loaded: cat.1028.jpg | Class: Cat
Loaded: cat.1029.jpg | Class: Cat
Loaded: cat.103.jpg | Class: Cat
Loaded: cat.1030.jpg | Class: Cat
Loaded: cat.1031.jpg | Class: Cat
Loaded: cat.1032.jpg | Class: Cat
Loaded: cat.1033.jpg | Class: Cat
Loaded: cat.1034.jpg | Class: Cat
Loaded: cat.1035.jpg | Class: Cat
Loaded: cat.1036.jpg | Class: Cat
Loaded: cat.1037.jpg | Class: Cat
Loaded: cat.1038.jpg | Class: Cat
Loaded: cat.1039.jpg | Class: Cat
Loaded: cat.104.jpg | Class: Cat
Loaded: cat.1040.jpg | Class: Cat
Loaded: cat.1041.jpg | Class: Cat
Loaded: cat.1042.jpg | Class: Cat
Loaded: cat.1043.jpg | Class: Cat
Loaded: cat.1044.jpg | Class: Cat
Loaded: cat.1045.jpg | Class: Cat
Loaded: cat.1046.jpg | Class: Cat
Loaded: cat.1047.jpg | Class: Cat
Loaded: cat.1048.jpg | Class: Cat
Loaded: cat.1049.jpg | Class: Cat
Loaded: cat.105.jpg | Class: Cat
Loaded: cat.1050.jpg | Class: Cat
Loaded: cat.1051.jpg | Class: Cat
Loaded: cat.1052.jpg | Class: Cat
Loaded: cat.1053.jpg | Class: Cat
Loaded: cat.1054.jpg | Class: Cat
```

```
Loaded: cat.1055.jpg | Class: Cat
Loaded: cat.1056.jpg | Class: Cat
Loaded: cat.1057.jpg | Class: Cat
Loaded: cat.1058.jpg | Class: Cat
Loaded: cat.1059.jpg | Class: Cat
Loaded: cat.106.jpg | Class: Cat
Loaded: cat.1060.jpg | Class: Cat
Loaded: cat.1061.jpg | Class: Cat
Loaded: cat.1062.jpg | Class: Cat
Loaded: cat.1063.jpg | Class: Cat
Loaded: cat.1064.jpg | Class: Cat
Loaded: cat.1065.jpg | Class: Cat
Loaded: cat.1066.jpg | Class: Cat
Loaded: cat.1067.jpg | Class: Cat
Loaded: cat.1068.jpg | Class: Cat
Loaded: cat.1069.jpg | Class: Cat
Loaded: cat.107.jpg | Class: Cat
Loaded: cat.1070.jpg | Class: Cat
Loaded: cat.1071.jpg | Class: Cat
Loaded: cat.1072.jpg | Class: Cat
Loaded: cat.1073.jpg | Class: Cat
Loaded: cat.1074.jpg | Class: Cat
Loaded: cat.1075.jpg | Class: Cat
Loaded: cat.1076.jpg | Class: Cat
Loaded: cat.1077.jpg | Class: Cat
Loaded: cat.1078.jpg | Class: Cat
Loaded: cat.1079.jpg | Class: Cat
Loaded: cat.108.jpg | Class: Cat
Loaded: cat.1080.jpg | Class: Cat
Loaded: cat.1081.jpg | Class: Cat
Loaded: cat.1082.jpg | Class: Cat
Loaded: cat.1083.jpg | Class: Cat
Loaded: cat.1084.jpg | Class: Cat
Loaded: cat.1085.jpg | Class: Cat
Loaded: cat.1086.jpg | Class: Cat
Loaded: cat.1087.jpg | Class: Cat
Loaded: cat.1088.jpg | Class: Cat
Loaded: cat.1089.jpg | Class: Cat
Loaded: cat.109.jpg | Class: Cat
Loaded: cat.1090.jpg | Class: Cat
Loaded: cat.1091.jpg | Class: Cat
Loaded: cat.1092.jpg | Class: Cat
Loaded: cat.1093.jpg | Class: Cat
Loaded: cat.1094.jpg | Class: Cat
Loaded: cat.1095.jpg | Class: Cat
Loaded: cat.1096.jpg | Class: Cat
Loaded: cat.1097.jpg | Class: Cat
Loaded: cat.1098.jpg | Class: Cat
```

```
Loaded: cat.1099.jpg | Class: Cat
Loaded: cat.11.jpg | Class: Cat
Loaded: cat.110.jpg | Class: Cat
Loaded: cat.1100.jpg | Class: Cat
Loaded: cat.1101.jpg | Class: Cat
Loaded: cat.1102.jpg | Class: Cat
Loaded: cat.1103.jpg | Class: Cat
Loaded: cat.1104.jpg | Class: Cat
Loaded: cat.1105.jpg | Class: Cat
Loaded: cat.1106.jpg | Class: Cat
Loaded: cat.1107.jpg | Class: Cat
Loaded: cat.1108.jpg | Class: Cat
Loaded: cat.1109.jpg | Class: Cat
Loaded: cat.111.jpg | Class: Cat
Loaded: cat.1110.jpg | Class: Cat
Loaded: cat.1111.jpg | Class: Cat
Loaded: cat.1112.jpg | Class: Cat
Loaded: cat.1113.jpg | Class: Cat
Loaded: cat.1114.jpg | Class: Cat
Loaded: cat.1115.jpg | Class: Cat
Loaded: cat.1116.jpg | Class: Cat
Loaded: cat.1117.jpg | Class: Cat
Loaded: cat.1118.jpg | Class: Cat
Loaded: cat.1119.jpg | Class: Cat
Loaded: cat.112.jpg | Class: Cat
Loaded: cat.1120.jpg | Class: Cat
Loaded: cat.1121.jpg | Class: Cat
Loaded: cat.1122.jpg | Class: Cat
Loaded: cat.1123.jpg | Class: Cat
Loaded: cat.1124.jpg | Class: Cat
Loaded: cat.1125.jpg | Class: Cat
Loaded: cat.1126.jpg | Class: Cat
Loaded: cat.1127.jpg | Class: Cat
Loaded: cat.1128.jpg | Class: Cat
Loaded: cat.1129.jpg | Class: Cat
Loaded: cat.113.jpg | Class: Cat
Loaded: cat.1130.jpg | Class: Cat
Loaded: cat.1131.jpg | Class: Cat
Loaded: cat.1132.jpg | Class: Cat
Loaded: cat.1133.jpg | Class: Cat
Loaded: cat.1134.jpg | Class: Cat
Loaded: cat.1135.jpg | Class: Cat
Loaded: cat.1136.jpg | Class: Cat
Loaded: cat.1137.jpg | Class: Cat
Loaded: cat.1138.jpg | Class: Cat
Loaded: cat.1139.jpg | Class: Cat
Loaded: cat.114.jpg | Class: Cat
Loaded: cat.1140.jpg | Class: Cat
```

```
Loaded: cat.1141.jpg | Class: Cat
Loaded: cat.1142.jpg | Class: Cat
Loaded: cat.1143.jpg | Class: Cat
Loaded: cat.1144.jpg | Class: Cat
Loaded: cat.1145.jpg | Class: Cat
Loaded: cat.1146.jpg | Class: Cat
Loaded: cat.1147.jpg | Class: Cat
Loaded: cat.1148.jpg | Class: Cat
Loaded: cat.1149.jpg | Class: Cat
Loaded: cat.115.jpg | Class: Cat
Loaded: cat.1150.jpg | Class: Cat
Loaded: cat.1151.jpg | Class: Cat
Loaded: cat.1152.jpg | Class: Cat
Loaded: cat.1153.jpg | Class: Cat
Loaded: cat.1154.jpg | Class: Cat
Loaded: cat.1155.jpg | Class: Cat
Loaded: cat.1156.jpg | Class: Cat
Loaded: cat.1157.jpg | Class: Cat
Loaded: cat.1158.jpg | Class: Cat
Loaded: cat.1159.jpg | Class: Cat
Loaded: cat.116.jpg | Class: Cat
Loaded: cat.1160.jpg | Class: Cat
Loaded: cat.1161.jpg | Class: Cat
Loaded: cat.1162.jpg | Class: Cat
Loaded: cat.1163.jpg | Class: Cat
Loaded: cat.1164.jpg | Class: Cat
Loaded: cat.1165.jpg | Class: Cat
Loaded: cat.1166.jpg | Class: Cat
Loaded: cat.1167.jpg | Class: Cat
Loaded: cat.1168.jpg | Class: Cat
Loaded: cat.1169.jpg | Class: Cat
Loaded: cat.117.jpg | Class: Cat
Loaded: cat.1170.jpg | Class: Cat
Loaded: cat.1171.jpg | Class: Cat
Loaded: cat.1172.jpg | Class: Cat
Loaded: cat.1173.jpg | Class: Cat
Loaded: cat.1174.jpg | Class: Cat
Loaded: cat.1175.jpg | Class: Cat
Loaded: cat.1176.jpg | Class: Cat
Loaded: cat.1177.jpg | Class: Cat
Loaded: cat.1178.jpg | Class: Cat
Loaded: dog.1.jpg | Class: Dog
Loaded: dog.10.jpg | Class: Dog
Loaded: dog.100.jpg | Class: Dog
Loaded: dog.1000.jpg | Class: Dog
Loaded: dog.1001.jpg | Class: Dog
Loaded: dog.1002.jpg | Class: Dog
Loaded: dog.1003.jpg | Class: Dog
```

```
Loaded: dog.1004.jpg | Class: Dog
Loaded: dog.1005.jpg | Class: Dog
Loaded: dog.1006.jpg | Class: Dog
Loaded: dog.1007.jpg | Class: Dog
Loaded: dog.1008.jpg | Class: Dog
Loaded: dog.1009.jpg | Class: Dog
Loaded: dog.101.jpg | Class: Dog
Loaded: dog.1010.jpg | Class: Dog
Loaded: dog.1011.jpg | Class: Dog
Loaded: dog.1012.jpg | Class: Dog
Loaded: dog.1013.jpg | Class: Dog
Loaded: dog.1014.jpg | Class: Dog
Loaded: dog.1015.jpg | Class: Dog
Loaded: dog.1016.jpg | Class: Dog
Loaded: dog.1017.jpg | Class: Dog
Loaded: dog.1018.jpg | Class: Dog
Loaded: dog.1019.jpg | Class: Dog
Loaded: dog.102.jpg | Class: Dog
Loaded: dog.1020.jpg | Class: Dog
Loaded: dog.1021.jpg | Class: Dog
Loaded: dog.1022.jpg | Class: Dog
Loaded: dog.1023.jpg | Class: Dog
Loaded: dog.1024.jpg | Class: Dog
Loaded: dog.1025.jpg | Class: Dog
Loaded: dog.1026.jpg | Class: Dog
Loaded: dog.1027.jpg | Class: Dog
Loaded: dog.1028.jpg | Class: Dog
Loaded: dog.1029.jpg | Class: Dog
Loaded: dog.103.jpg | Class: Dog
Loaded: dog.1030.jpg | Class: Dog
Loaded: dog.1031.jpg | Class: Dog
Loaded: dog.1032.jpg | Class: Dog
Loaded: dog.1033.jpg | Class: Dog
Loaded: dog.1034.jpg | Class: Dog
Loaded: dog.1035.jpg | Class: Dog
Loaded: dog.1036.jpg | Class: Dog
Loaded: dog.1037.jpg | Class: Dog
Loaded: dog.1038.jpg | Class: Dog
Loaded: dog.1039.jpg | Class: Dog
Loaded: dog.104.jpg | Class: Dog
Loaded: dog.1040.jpg | Class: Dog
Loaded: dog.1041.jpg | Class: Dog
Loaded: dog.1042.jpg | Class: Dog
Loaded: dog.1043.jpg | Class: Dog
Loaded: dog.1044.jpg | Class: Dog
Loaded: dog.1045.jpg | Class: Dog
Loaded: dog.1046.jpg | Class: Dog
Loaded: dog.1047.jpg | Class: Dog
```

```
Loaded: dog.1048.jpg | Class: Dog
Loaded: dog.1049.jpg | Class: Dog
Loaded: dog.105.jpg | Class: Dog
Loaded: dog.1050.jpg | Class: Dog
Loaded: dog.1051.jpg | Class: Dog
Loaded: dog.1052.jpg | Class: Dog
Loaded: dog.1053.jpg | Class: Dog
Loaded: dog.1054.jpg | Class: Dog
Loaded: dog.1055.jpg | Class: Dog
Loaded: dog.1056.jpg | Class: Dog
Loaded: dog.1057.jpg | Class: Dog
Loaded: dog.1058.jpg | Class: Dog
Loaded: dog.1059.jpg | Class: Dog
Loaded: dog.106.jpg | Class: Dog
Loaded: dog.1060.jpg | Class: Dog
Loaded: dog.1061.jpg | Class: Dog
Loaded: dog.1062.jpg | Class: Dog
Loaded: dog.1063.jpg | Class: Dog
Loaded: dog.1064.jpg | Class: Dog
Loaded: dog.1065.jpg | Class: Dog
Loaded: dog.1066.jpg | Class: Dog
Loaded: dog.1067.jpg | Class: Dog
Loaded: dog.1068.jpg | Class: Dog
Loaded: dog.1069.jpg | Class: Dog
Loaded: dog.107.jpg | Class: Dog
Loaded: dog.1070.jpg | Class: Dog
Loaded: dog.1071.jpg | Class: Dog
Loaded: dog.1072.jpg | Class: Dog
Loaded: dog.1073.jpg | Class: Dog
Loaded: dog.1074.jpg | Class: Dog
Loaded: dog.1075.jpg | Class: Dog
Loaded: dog.1076.jpg | Class: Dog
Loaded: dog.1077.jpg | Class: Dog
Loaded: dog.1078.jpg | Class: Dog
Loaded: dog.1079.jpg | Class: Dog
Loaded: dog.108.jpg | Class: Dog
Loaded: dog.1080.jpg | Class: Dog
Loaded: dog.1081.jpg | Class: Dog
Loaded: dog.1082.jpg | Class: Dog
Loaded: dog.1083.jpg | Class: Dog
Loaded: dog.1084.jpg | Class: Dog
Loaded: dog.1085.jpg | Class: Dog
Loaded: dog.1086.jpg | Class: Dog
Loaded: dog.1087.jpg | Class: Dog
Loaded: dog.1088.jpg | Class: Dog
Loaded: dog.1089.jpg | Class: Dog
Loaded: dog.109.jpg | Class: Dog
Loaded: dog.1090.jpg | Class: Dog
```

```
Loaded: dog.1091.jpg | Class: Dog
Loaded: dog.1092.jpg | Class: Dog
Loaded: dog.1093.jpg | Class: Dog
Loaded: dog.1094.jpg | Class: Dog
Loaded: dog.1095.jpg | Class: Dog
Loaded: dog.1096.jpg | Class: Dog
Loaded: dog.1097.jpg | Class: Dog
Loaded: dog.1098.jpg | Class: Dog
Loaded: dog.1099.jpg | Class: Dog
Loaded: dog.11.jpg | Class: Dog
Loaded: dog.110.jpg | Class: Dog
Loaded: dog.1100.jpg | Class: Dog
Loaded: dog.1101.jpg | Class: Dog
Loaded: dog.1102.jpg | Class: Dog
Loaded: dog.1103.jpg | Class: Dog
Loaded: dog.1104.jpg | Class: Dog
Loaded: dog.1105.jpg | Class: Dog
Loaded: dog.1106.jpg | Class: Dog
Loaded: dog.1107.jpg | Class: Dog
Loaded: dog.1108.jpg | Class: Dog
Loaded: dog.1109.jpg | Class: Dog
Loaded: dog.111.jpg | Class: Dog
Loaded: dog.1110.jpg | Class: Dog
Loaded: dog.1111.jpg | Class: Dog
Loaded: dog.1112.jpg | Class: Dog
Loaded: dog.1113.jpg | Class: Dog
Loaded: dog.1114.jpg | Class: Dog
Loaded: dog.1115.jpg | Class: Dog
Loaded: dog.1116.jpg | Class: Dog
Loaded: dog.1117.jpg | Class: Dog
Loaded: dog.1118.jpg | Class: Dog
Loaded: dog.1119.jpg | Class: Dog
Loaded: dog.112.jpg | Class: Dog
Loaded: dog.1120.jpg | Class: Dog
Loaded: dog.1121.jpg | Class: Dog
Loaded: dog.1122.jpg | Class: Dog
Loaded: dog.1123.jpg | Class: Dog
Loaded: dog.1124.jpg | Class: Dog
Loaded: dog.1125.jpg | Class: Dog
Loaded: dog.1126.jpg | Class: Dog
Loaded: dog.1127.jpg | Class: Dog
Loaded: dog.1128.jpg | Class: Dog
Loaded: dog.1129.jpg | Class: Dog
Loaded: dog.113.jpg | Class: Dog
Loaded: dog.1130.jpg | Class: Dog
Loaded: dog.1131.jpg | Class: Dog
Loaded: dog.1132.jpg | Class: Dog
Loaded: dog.1133.jpg | Class: Dog
```

```
Loaded: dog.1134.jpg | Class: Dog
Loaded: dog.1135.jpg | Class: Dog
Loaded: dog.1136.jpg | Class: Dog
Loaded: dog.1137.jpg | Class: Dog
Loaded: dog.1138.jpg | Class: Dog
Loaded: dog.1139.jpg | Class: Dog
Loaded: dog.114.jpg | Class: Dog
Loaded: dog.1140.jpg | Class: Dog
Loaded: dog.1141.jpg | Class: Dog
Loaded: dog.1142.jpg | Class: Dog
Loaded: dog.1143.jpg | Class: Dog
Loaded: dog.1144.jpg | Class: Dog
Loaded: dog.1145.jpg | Class: Dog
Loaded: dog.1146.jpg | Class: Dog
Loaded: dog.1147.jpg | Class: Dog
Loaded: dog.1148.jpg | Class: Dog
Loaded: dog.1149.jpg | Class: Dog
Loaded: dog.115.jpg | Class: Dog
Loaded: dog.1150.jpg | Class: Dog
Loaded: dog.1151.jpg | Class: Dog
Loaded: dog.1152.jpg | Class: Dog
Loaded: dog.1153.jpg | Class: Dog
Loaded: dog.1154.jpg | Class: Dog
Loaded: dog.1155.jpg | Class: Dog
Loaded: dog.1156.jpg | Class: Dog
Loaded: dog.1157.jpg | Class: Dog
Loaded: dog.1158.jpg | Class: Dog
Loaded: dog.1159.jpg | Class: Dog
Loaded: dog.116.jpg | Class: Dog
Loaded: dog.1160.jpg | Class: Dog
Loaded: dog.1161.jpg | Class: Dog
Loaded: dog.1162.jpg | Class: Dog
Loaded: dog.1163.jpg | Class: Dog
Loaded: dog.1164.jpg | Class: Dog
Loaded: dog.1165.jpg | Class: Dog
Loaded: dog.1166.jpg | Class: Dog
Loaded: dog.1167.jpg | Class: Dog
Loaded: dog.1168.jpg | Class: Dog
Loaded: dog.1169.jpg | Class: Dog
Loaded: dog.117.jpg | Class: Dog
Loaded: dog.1170.jpg | Class: Dog
Loaded: dog.1171.jpg | Class: Dog
Loaded: dog.1172.jpg | Class: Dog
Loaded: dog.1173.jpg | Class: Dog
Loaded: dog.1174.jpg | Class: Dog
Loaded: dog.1175.jpg | Class: Dog
Loaded: dog.1176.jpg | Class: Dog
Loaded: dog.1177.jpg | Class: Dog
```

```
Loaded: dog.1178.jpg | Class: Dog
```

[49]:
```python
# Verify that images have been successfully loaded
print(f"Total images processed: {len(data)}")
```

```
Total images processed: 400
```

[50]:
```python
cat_dog_df = pd.DataFrame(data)
cat_dog_df.head()
```

[50]:
```
                                               Image Class      Filename
0  <PIL.Image.Image image mode=RGB size=100x100 a…   Cat     cat.1.jpg
1  <PIL.Image.Image image mode=RGB size=100x100 a…   Cat    cat.10.jpg
2  <PIL.Image.Image image mode=RGB size=100x100 a…   Cat   cat.100.jpg
3  <PIL.Image.Image image mode=RGB size=100x100 a…   Cat  cat.1000.jpg
4  <PIL.Image.Image image mode=RGB size=100x100 a…   Cat  cat.1001.jpg
```

[51]:
```python
cat_dog_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Image     400 non-null    object
 1   Class     400 non-null    object
 2   Filename  400 non-null    object
dtypes: object(3)
memory usage: 9.5+ KB
```

[52]:
```python
cat_dog_df.shape
```

[52]:
```
(400, 3)
```

[53]:
```python
# Flatten images and create a NumPy array
cat_dog_df['Image_Array'] = cat_dog_df['Image'].apply(lambda img: np.array(img).
 ↪flatten())
cat_dog_df.head()
```

[53]:
```
                                               Image Class      Filename  \
0  <PIL.Image.Image image mode=RGB size=100x100 a…   Cat     cat.1.jpg
1  <PIL.Image.Image image mode=RGB size=100x100 a…   Cat    cat.10.jpg
2  <PIL.Image.Image image mode=RGB size=100x100 a…   Cat   cat.100.jpg
3  <PIL.Image.Image image mode=RGB size=100x100 a…   Cat  cat.1000.jpg
4  <PIL.Image.Image image mode=RGB size=100x100 a…   Cat  cat.1001.jpg

                                         Image_Array
0  [40, 45, 42, 40, 44, 46, 47, 50, 57, 44, 48, 5…
```

```
1   [29, 34, 43, 26, 31, 42, 40, 43, 58, 41, 46, 5…
2   [221, 222, 217, 223, 224, 219, 223, 224, 218, …
3   [140, 112, 75, 143, 115, 78, 149, 121, 84, 120…
4   [54, 50, 9, 59, 46, 19, 62, 47, 14, 56, 45, 15…
```

[54]:
```python
print("No of pixels in each image:",len(cat_dog_df['Image_Array'][0]))
cat_dog_df['Image_Array'][0]
```

```
No of pixels in each image: 30000
```

[54]:
```
array([40, 45, 42, …, 45, 33, 25], dtype=uint8)
```

[55]:
```python
# Stack all the flattened images into a 2D NumPy array
image_data = np.stack(cat_dog_df['Image_Array'].values)

# Normalize the image data (optional but recommended)
image_data = image_data / 255.0
```

[56]:
```python
# Apply PCA to reduce the dimensionality of the image data
pca = PCA()
pca.fit(image_data)

# Calculate the number of components required to preserve 90% of the variance
explained_variance = np.cumsum(pca.explained_variance_ratio_)
components_needed = np.argmax(explained_variance >= 0.90) + 1

print(f"Number of components to preserve 90% variance: {components_needed}")
```

```
Number of components to preserve 90% variance: 121
```

[57]:
```python
# Reduce the image data to the selected number of components
pca = PCA(n_components=components_needed)
reduced_data = pca.fit_transform(image_data)

# Create column names for the PCA components
reduced_columns = [f"PCA_Component_{i}" for i in range(components_needed)]

# Add the PCA components back to the DataFrame
cat_dog_df[reduced_columns] = reduced_data

cat_dog_df.head()
```

[57]:
```
                                               Image Class      Filename  \
0  <PIL.Image.Image image mode=RGB size=100x100 a…   Cat     cat.1.jpg
1  <PIL.Image.Image image mode=RGB size=100x100 a…   Cat    cat.10.jpg
2  <PIL.Image.Image image mode=RGB size=100x100 a…   Cat   cat.100.jpg
3  <PIL.Image.Image image mode=RGB size=100x100 a…   Cat  cat.1000.jpg
```

```
4  <PIL.Image.Image image mode=RGB size=100x100 a…   Cat  cat.1001.jpg

                                       Image_Array  PCA_Component_0  \
0  [40, 45, 42, 40, 44, 46, 47, 50, 57, 44, 48, 5…       -30.256248
1  [29, 34, 43, 26, 31, 42, 40, 43, 58, 41, 46, 5…        -7.244256
2  [221, 222, 217, 223, 224, 219, 223, 224, 218, …        39.295936
3  [140, 112, 75, 143, 115, 78, 149, 121, 84, 120…       -16.772328
4  [54, 50, 9, 59, 46, 19, 62, 47, 14, 56, 45, 15…         8.113674

   PCA_Component_1  PCA_Component_2  PCA_Component_3  PCA_Component_4  \
0        -5.830454        -6.679201         3.263976        -0.737821
1         5.522854        -7.267380         8.770166        13.004716
2        -7.658990         3.318373         3.445404       -12.931039
3        16.972485       -10.146222        -9.976377        -9.416429
4         8.018668        -7.917145        -3.073646        -2.139626

   PCA_Component_5  …  PCA_Component_111  PCA_Component_112  \
0        -7.383711  …           0.842703          -1.172835
1         3.174100  …           1.635040           1.358336
2        10.594014  …           0.152513           1.724625
3       -17.320315  …          -0.331758           1.805941
4        -2.214704  …          -0.412273           1.288186

   PCA_Component_113  PCA_Component_114  PCA_Component_115  PCA_Component_116  \
0           0.645978          -0.585119           0.143256           0.274923
1          -0.385905           1.401584          -0.015800           0.675318
2           1.894523          -0.559815          -0.226759           0.783015
3           0.781750           0.081736          -0.055631           0.137769
4           0.462400           2.813101          -0.668271          -0.745204

   PCA_Component_117  PCA_Component_118  PCA_Component_119  PCA_Component_120
0          -0.597906           0.456654          -0.565541          -0.620346
1           0.013697           3.051938          -0.262150          -0.707740
2          -2.284796          -1.418917          -0.113111          -1.754841
3           4.202649          -0.025462           2.196081          -1.379367
4           0.164160           3.019058          -2.833628          -1.396560

[5 rows x 125 columns]
```

2.0.1 Summary

To determine the number of components required to preserve 90% of the variance using Principal Component Analysis (PCA) on the images in the cat and dog dataset, the following steps were followed:

1. **Data Preprocessing**:
    - We are reading only 200 images of each class to ensure that we are not encountering any issues with memory and time

- Images were resized to a uniform dimension of (100, 100) pixels to ensure no issues with the memory/running time.

2. **Image Flattening**:
   - Each image was flattened into a 1D vector, transforming the image data into a format suitable for PCA.
3. **Normalization**:
   - The pixel values of the images were normalized by dividing by 255. This scales the pixel values to a range of 0 to 1, ensuring that each feature (pixel) contributes equally to the PCA process.
4. **PCA Application**:
   - PCA was applied to the normalized image data, and the cumulative explained variance was computed for each principal component.
5. **Result**:
   - The number of principal components required to preserve 90% of the total variance in the dataset was found to be **116**.

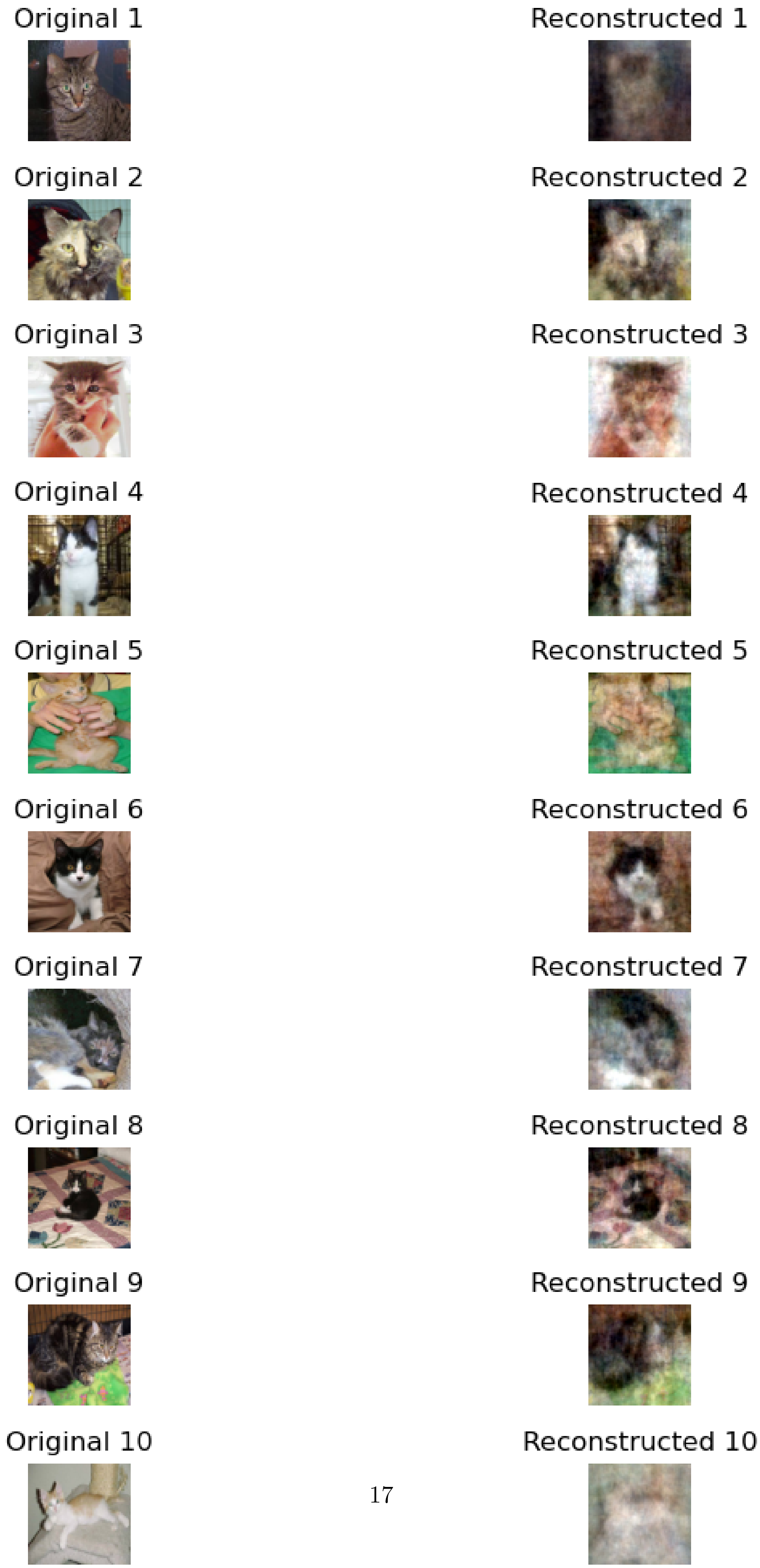
Thus, **121 principal components** are needed to retain 90% of the variance in the '360 Rocks' image dataset after applying PCA.

3 2

[58]:
```python
# Number of images to display
num_images = 10

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

# Select 10 random images for display (or just use the first 10)
for i in range(num_images):
    # Reshape the original image back to its 2D form (64x64x3 for this example)
    original_image = image_data[i].reshape(100, 100, 3)

    # Reconstruct the image using PCA
    reconstructed_image_flat = pca.inverse_transform(reduced_data[i])

    # Reshape the reconstructed image back to (64, 64, 3)
    reconstructed_image = reconstructed_image_flat.reshape(100, 100, 3)

    # Display original image
    plt.subplot(num_images, 2, 2 * i + 1)
    plt.imshow(original_image)
    plt.title(f"Original {i+1}")
    plt.axis('off')

    # Display reconstructed image
    plt.subplot(num_images, 2, 2 * i + 2)
    plt.imshow(reconstructed_image)
    plt.title(f"Reconstructed {i+1}")
```

```
    plt.axis('off')

plt.tight_layout()

plt.show()
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.026344231026005693..0.602391307492034].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.146364426585181..1.0559651075825798].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[0.010606131238229122..1.1483355274240026].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.12700106079004814..1.0774082899489914].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.10609422929987056..0.967406701714502].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.1407724575866775..0.9320270805674691].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.12293471645549292..1.1454409390944673].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.1869464164612536..1.1051531538939414].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.1279065321592141..0.9170753324135477].
```


Original 1
Reconstructed 1
Original 2
Reconstructed 2
Original 3
Reconstructed 3
Original 4
Reconstructed 4
Original 5
Reconstructed 5
Original 6
Reconstructed 6
Original 7
Reconstructed 7
Original 8
Reconstructed 8
Original 9
Reconstructed 9
Original 10
Reconstructed 10

4 3A

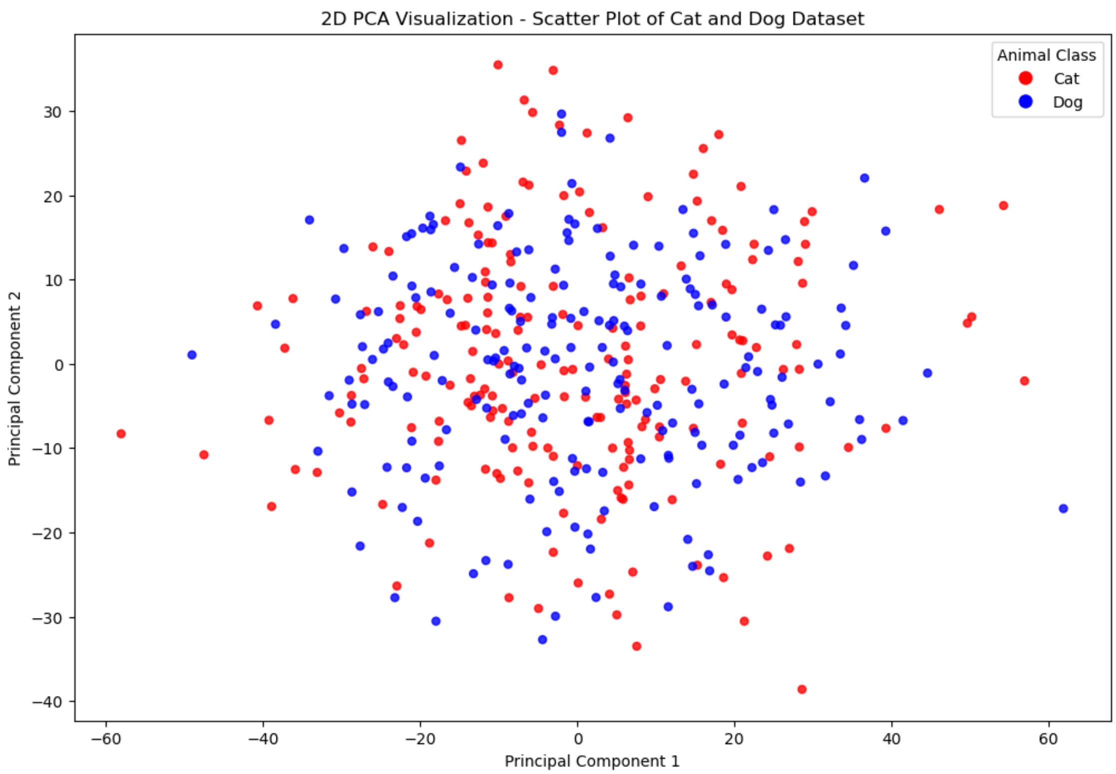
[59]:
```
# Apply PCA to reduce the dimensionality to 2
pca_2d = PCA(n_components=2)

# Fit PCA on the flattened image data (image_data contains images reshaped to␣
↪1D arrays)
reduced_data_2d = pca_2d.fit_transform(image_data)

# Calculate the total variance explained by the first two components
explained_variance = np.sum(pca_2d.explained_variance_ratio_)

# Print the explained variance
print(f"Variance explained by the first 2 principal components:␣
↪{explained_variance * 100:.2f}%")
```

```
Variance explained by the first 2 principal components: 28.53%
```

5 3B

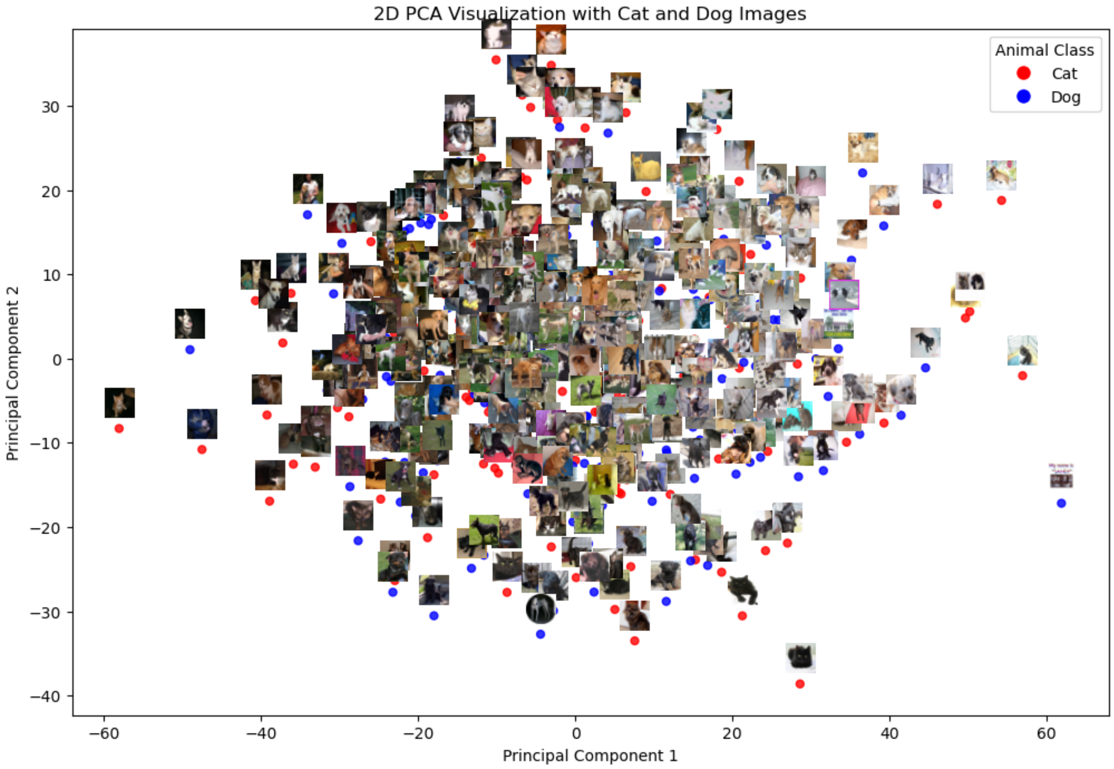
[60]:
```
# Define color mapping for categories (cats and dogs)
color_mapping = {
    "Cat": "red",
    "Dog": "blue"
}

# Add a column for colors in the dataframe based on the class
cat_dog_df['Color'] = cat_dog_df['Class'].map(color_mapping)

# Create the scatter plot
plt.figure(figsize=(12, 8))
scatter = plt.scatter(
    cat_dog_df['PCA_Component_0'],
    cat_dog_df['PCA_Component_1'],
    c=cat_dog_df['Color'],
    label=cat_dog_df['Class'],
    s=25, alpha=0.8
)

# Add legend
legend_elements = [
    plt.Line2D([0], [0], marker='o', color='w', label=key, markersize=10,␣
↪markerfacecolor=color)
```

```
    for key, color in color_mapping.items()
]
plt.legend(handles=legend_elements, title="Animal Class", loc="best")

# Add labels and title
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.title("2D PCA Visualization - Scatter Plot of Cat and Dog Dataset")

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



[61]:
```
# Define color mapping for categories (cats and dogs)
color_mapping = {
    "Cat": "red",
    "Dog": "blue"
}

# Assuming cat_dog_df already contains the necessary PCA components and image␣
↪data
cat_dog_df['Color'] = cat_dog_df['Class'].map(color_mapping)
```

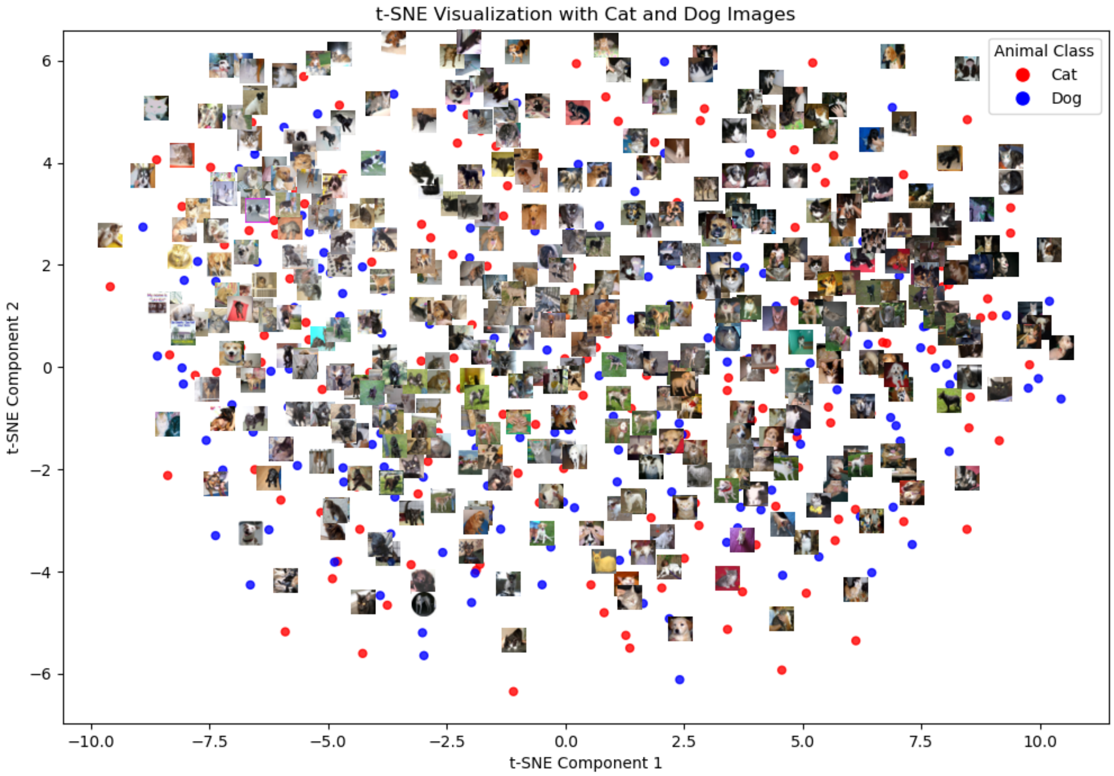
```python
# Create the scatter plot
fig, ax = plt.subplots(figsize=(12, 8))

# Plot the dots (scatter points)
scatter = ax.scatter(
    cat_dog_df['PCA_Component_0'],
    cat_dog_df['PCA_Component_1'],
    c=cat_dog_df['Color'],
    s=25, alpha=0.8
)

# Add legend
legend_elements = [
    plt.Line2D([0], [0], marker='o', color='w', label=key, markersize=10, markerfacecolor=color)
    for key, color in color_mapping.items()
]
ax.legend(handles=legend_elements, title="Animal Class", loc="best")

# Define the vertical offset for the images
vertical_offset = 3  # Adjust this value as needed

# Overlay cat and dog images above or below the dots
for i, row in cat_dog_df.iterrows():
    try:
        # Resize the image for visualization
        image = np.array(row['Image'].resize((24, 24)))  # Resize for better fit
        im = OffsetImage(image, zoom=0.8, alpha=1.0)  # Adjust zoom level for visibility
        # Position the image above the dot
        ab = AnnotationBbox(
            im,
            (row['PCA_Component_0'], row['PCA_Component_1'] + vertical_offset), # Offset y-coordinate (adjust + or - for above/below)
            frameon=False
        )
        ax.add_artist(ab)
    except Exception as e:
        print(f"Error adding image for {row['Filename']}: {e}")

# Set axis labels and title
ax.set_xlabel("Principal Component 1")
ax.set_ylabel("Principal Component 2")
ax.set_title("2D PCA Visualization with Cat and Dog Images")

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



[62]:

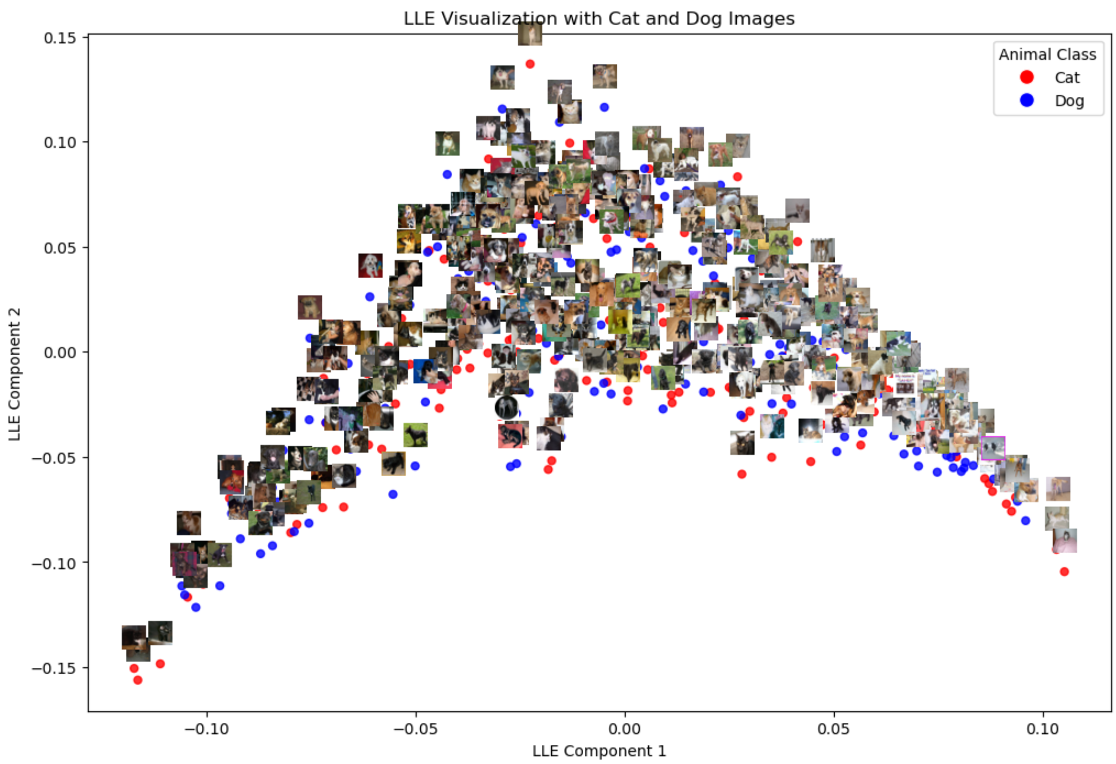
```
# Reduce the data to 2D using t-SNE
reduced_data_2d_TSNE = TSNE(n_components=2, random_state=42).
 ↪fit_transform(image_data)  # Replace image_data with your image data array

# Define color mapping for categories (Cats and Dogs)
color_mapping = {
    "Cat": "red",
    "Dog": "blue"
}

# Add color to the dataframe for mapping
cat_dog_df['Color'] = cat_dog_df['Class'].map(color_mapping)

# Create the scatter plot
fig, ax = plt.subplots(figsize=(12, 8))

# Add scatter points
scatter = ax.scatter(
    reduced_data_2d_TSNE[:, 0],
    reduced_data_2d_TSNE[:, 1],
    c=cat_dog_df['Color'],
    s=25,
```

```
    alpha=0.8,
)

# Add legend
legend_elements = [
    plt.Line2D([0], [0], marker='o', color='w', label=key, markersize=10,␣
 ↪markerfacecolor=color)
    for key, color in color_mapping.items()
]
ax.legend(handles=legend_elements, title="Animal Class", loc="best")

# Define the vertical offset for the images
vertical_offset = 1  # Adjust this value as needed

# Add images to the scatter plot
for i, row in cat_dog_df.iterrows():
    try:
        # Resize the image for visualization
        image = np.array(row['Image'].resize((24, 24)))  # Resize images to fit␣
 ↪better in the plot
        im = OffsetImage(image, zoom=0.65, alpha=1.0)  # Adjust zoom level as␣
 ↪necessary
        # Position the image above the dot
        ab = AnnotationBbox(
            im,
            (reduced_data_2d_TSNE[i, 0], reduced_data_2d_TSNE[i, 1] +␣
 ↪vertical_offset),
            frameon=False
        )
        ax.add_artist(ab)
    except Exception as e:
        print(f"Error adding image for {row['Filename']}: {e}")

# Set axis labels and title
ax.set_xlabel("t-SNE Component 1")
ax.set_ylabel("t-SNE Component 2")
ax.set_title("t-SNE Visualization with Cat and Dog Images")

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



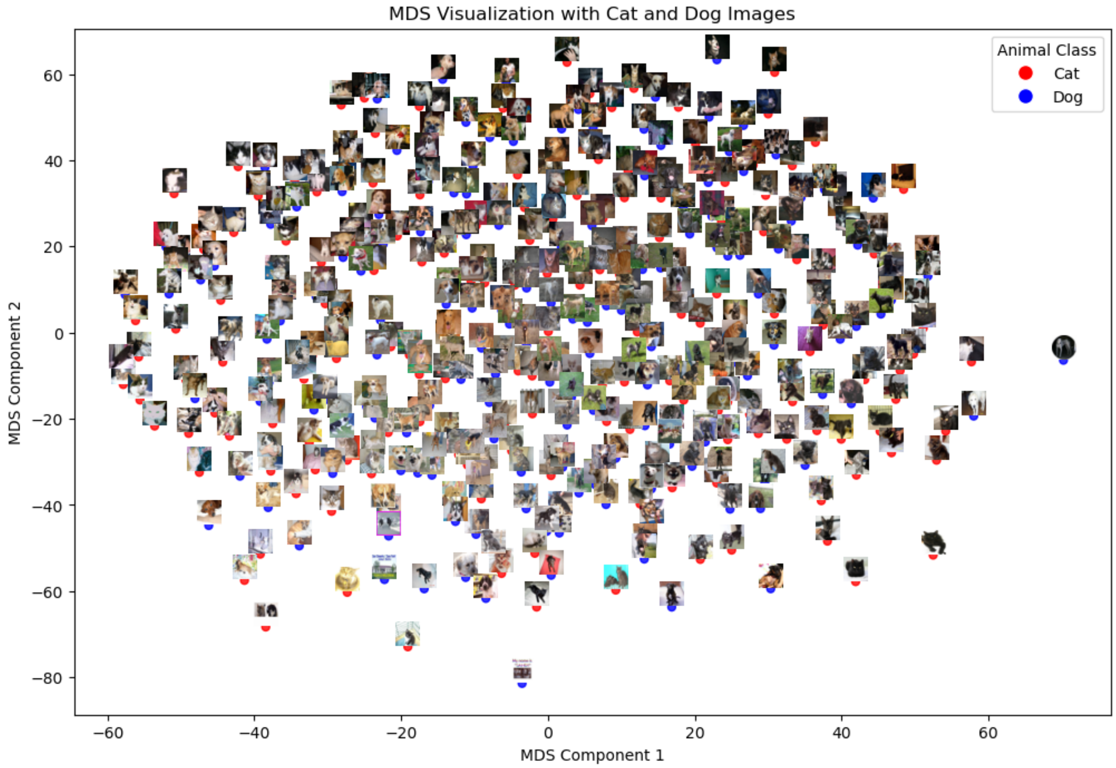
[63]:
```
# Reduce the data to 2D using Locally Linear Embedding (LLE)
reduced_data_2d_LLE = LocallyLinearEmbedding(n_components=2, n_neighbors=10).
↪fit_transform(image_data)  # Replace image_data with your image data array

# Define color mapping for categories (Cats and Dogs)
color_mapping = {
    "Cat": "red",
    "Dog": "blue"
}

# Add color to the dataframe for mapping
cat_dog_df['Color'] = cat_dog_df['Class'].map(color_mapping)

# Create the scatter plot
fig, ax = plt.subplots(figsize=(12, 8))

# Add scatter points
scatter = ax.scatter(
    reduced_data_2d_LLE[:, 0],
    reduced_data_2d_LLE[:, 1],
    c=cat_dog_df['Color'],
    s=25,
```

```
    alpha=0.8,
)

# Add legend
legend_elements = [
    plt.Line2D([0], [0], marker='o', color='w', label=key, markersize=10,␣
↪markerfacecolor=color)
    for key, color in color_mapping.items()
]
ax.legend(handles=legend_elements, title="Animal Class", loc="best")

# Calculate a dynamic vertical offset based on the y-range
y_range = reduced_data_2d_LLE[:, 1].max() - reduced_data_2d_LLE[:, 1].min()
vertical_offset = 0.05 * y_range  # Use 5% of the y-range as offset

# Add images to the scatter plot
for i, row in cat_dog_df.iterrows():
    try:
        # Resize the image for visualization
        image = np.array(row['Image'].resize((24, 24)))  # Resize images to fit␣
↪better in the plot
        im = OffsetImage(image, zoom=0.65, alpha=1.0)  # Adjust zoom level as␣
↪necessary
        # Position the image above the dot
        ab = AnnotationBbox(
            im,
            (reduced_data_2d_LLE[i, 0], reduced_data_2d_LLE[i, 1] +␣
↪vertical_offset),
            frameon=False
        )
        ax.add_artist(ab)
    except Exception as e:
        print(f"Error adding image for {row['Filename']}: {e}")

# Set axis labels and title
ax.set_xlabel("LLE Component 1")
ax.set_ylabel("LLE Component 2")
ax.set_title("LLE Visualization with Cat and Dog Images")

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



[64]:
```
reduced_data_2d_MDS = MDS(n_components=2, random_state=42).
 ↪fit_transform(image_data)

fig, ax = plt.subplots(figsize=(12, 8))

# Add scatter points
scatter = ax.scatter(
    reduced_data_2d_MDS[:, 0],
    reduced_data_2d_MDS[:, 1],
    c=cat_dog_df['Color'],
    s=25,
    alpha=0.8,
)

# Add legend
legend_elements = [
    plt.Line2D([0], [0], marker='o', color='w', label=key, markersize=10,␣
 ↪markerfacecolor=color)
    for key, color in color_mapping.items()
]
ax.legend(handles=legend_elements, title="Animal Class", loc="best")
```

```
# Define the vertical offset for the images
vertical_offset = 3  # Adjust this value as needed

# Add images to the scatter plot
for i, row in cat_dog_df.iterrows():
    try:
        # Resize the image for visualization
        image = np.array(row['Image'].resize((24, 24)))  # Resize images to fit␣
↪better in the plot
        im = OffsetImage(image, zoom=0.65, alpha=1.0)  # Adjust zoom level as␣
↪necessary
        # Position the image above the dot
        ab = AnnotationBbox(
            im,
            (reduced_data_2d_MDS[i, 0], reduced_data_2d_MDS[i, 1] +␣
↪vertical_offset),
            frameon=False
        )
        ax.add_artist(ab)
    except Exception as e:
        print(f"Error adding image for {row['Filename']}: {e}")

# Set axis labels and title
ax.set_xlabel("MDS Component 1")
ax.set_ylabel("MDS Component 2")
ax.set_title("MDS Visualization with Cat and Dog Images")

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



6 3C

6.1 Observations

Cluster Formation All four techniques (PCA, t-SNE, LLE, and MDS) successfully formed distinct clusters for cat and dog images, indicating that the model has learned meaningful representations.

Nonlinear Separability The data is not linearly separable, as evidenced by the complex shapes of the clusters. t-SNE, LLE, and MDS, which are better suited for capturing nonlinear relationships, have produced more visually appealing and informative visualizations compared to PCA.

Local Structure Preservation t-SNE and LLE excel at preserving local structure, meaning that similar images are clustered together. This is evident in the tight clusters formed by these techniques.

Global Structure Preservation MDS is designed to preserve global structure, ensuring that distant points in the original high-dimensional space remain distant in the low-dimensional embedding.

Overlapping Regions and Outliers All techniques exhibit some degree of overlap between clusters and outliers. This could be attributed to inherent similarities between certain cat and dog breeds, variations in image quality, or potential misclassifications in the dataset.

Choosing the Right Technique The choice of visualization technique depends on the specific goals of the analysis:

- **PCA**: Suitable for understanding linear relationships and reducing dimensionality. However, it may not be ideal for complex, nonlinear data.
- **t-SNE**: Excellent for visualizing nonlinear relationships and preserving local structure. However, it can be computationally expensive and might not preserve global structure well.
- **LLE**: Similar to t-SNE, LLE is good at preserving local structure but can struggle with global structure.
- **MDS**: Good for preserving global structure, but it might not be as effective as t-SNE and LLE in capturing local structure.

Which Method to Use?

PCA would be the better choice for this dataset and analysis due to the following reasons:

1. **Variance Explanation:** PCA efficiently captures a large proportion of the variance in the data, making it an excellent tool for dimensionality reduction.
2. **Interpretability:** PCA provides linear combinations of features, allowing for easy interpretation of the reduced dimensions.
3. **Computational Efficiency:** It is computationally faster than t-SNE and LLE, making it suitable for large datasets.
4. **Global Structure Representation:** PCA excels at preserving global patterns in the data, which is critical for understanding overall relationships between data points.

Additional Recommendations

- **t-SNE or LLE:** Use if the goal is to gain deeper insights into local relationships or non-linear clusters for visualization.
- **MDS:** Use to interpret data relationships based on pairwise distances but not for non-linear relationships.

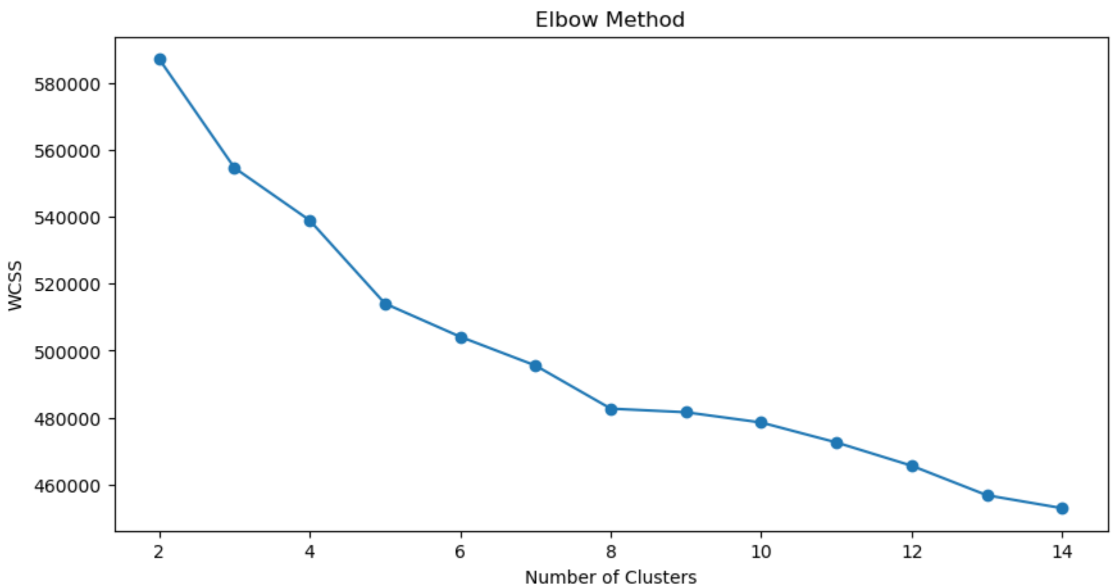
By starting with **PCA**, you can capture the major variance and global patterns efficiently, and then complement the analysis with t-SNE or LLE for more detailed local cluster insights.

7 5A

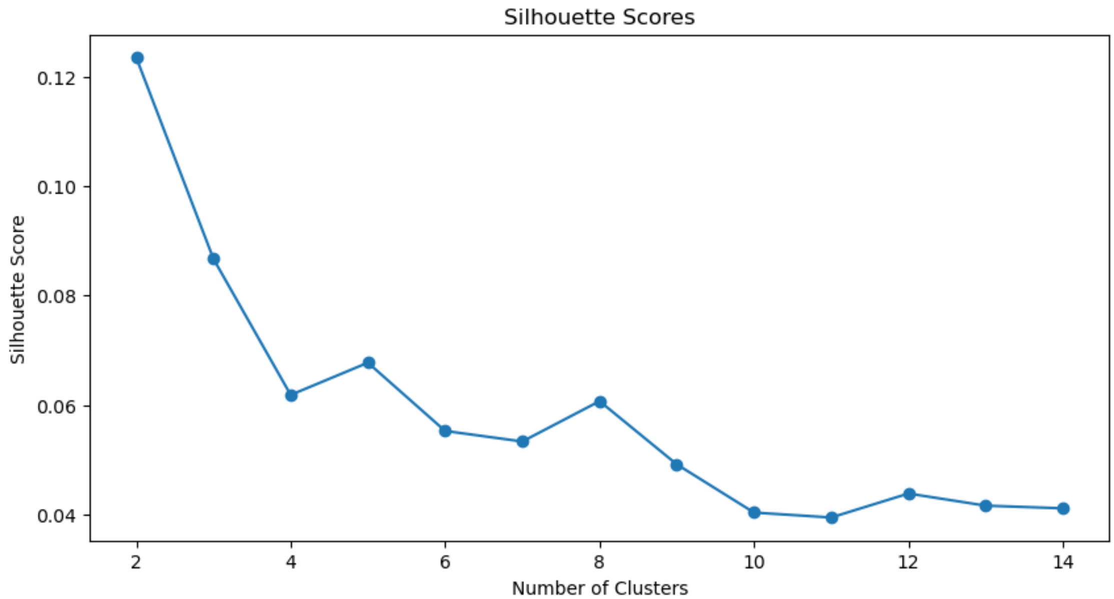
[65]:
```
wcss = []
silhouette_scores = []
range_clusters = range(2, 15)

for k in range_clusters:
    kmeans = KMeans(n_clusters=k, random_state=42)
    cluster_labels = kmeans.fit_predict(reduced_data)
```

```
    wcss.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(reduced_data, cluster_labels))
```

[66]:
```
# Elbow Method Plot
plt.figure(figsize=(10, 5))
plt.plot(range_clusters, wcss, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```



[67]:
```
# Silhouette Scores Plot
plt.figure(figsize=(10, 5))
plt.plot(range_clusters, silhouette_scores, marker='o')
plt.title('Silhouette Scores')
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')
plt.show()
```



7.1 Clustering Analysis: Inertia and Silhouette Scores

7.1.1 Inertia (Within-Cluster Sum of Squares)

Inertia measures the sum of squared distances from data points to their cluster centers. Lower inertia indicates better clustering. As the number of clusters increases, inertia decreases, but the rate of improvement slows after 2 clusters. This suggests 2 clusters might be optimal, as further increases yield diminishing returns.

7.1.2 Silhouette Scores

The silhouette score, which measures cluster cohesion and separation, is relatively low, with the highest score at **0.123**. This indicates weak cluster separation, even though inertia is improving. As the number of clusters increases, the silhouette score decreases, suggesting adding more clusters might not improve the clustering quality.

7.1.3 Insights and Recommendations

- **Inertia** suggests that **2 clusters** may be optimal based on the elbow method.
- The low **silhouette scores** indicate that the clustering could be improved, possibly by experimenting with different algorithms (e.g., **DBSCAN**, **Gaussian Mixture Models**) or revisiting feature engineering.

7.1.4 Conclusion

While 2 clusters seem optimal based on inertia, the low silhouette scores suggest further refinement is needed for better cluster separation.

8 5B

[68]:
```
# Extract PCA components for clustering
pca_columns = [col for col in cat_dog_df.columns if "PCA_Component" in col]
pca_data = cat_dog_df[pca_columns].values

# Ground truth labels from the 'Class' column
true_labels = cat_dog_df['Class'].astype('category').cat.codes  # Convert to␣
↪numeric codes
```

[69]:
```
# Perform K-Means clustering
kmeans = KMeans(n_clusters=2, random_state=42)
predicted_labels = kmeans.fit_predict(pca_data)
```

[70]:
```
# Confusion matrix to match clusters with ground truth classes
conf_matrix = confusion_matrix(true_labels, predicted_labels)
conf_matrix
```

[70]:
```
array([[ 87, 113],
       [ 94, 106]], dtype=int64)
```

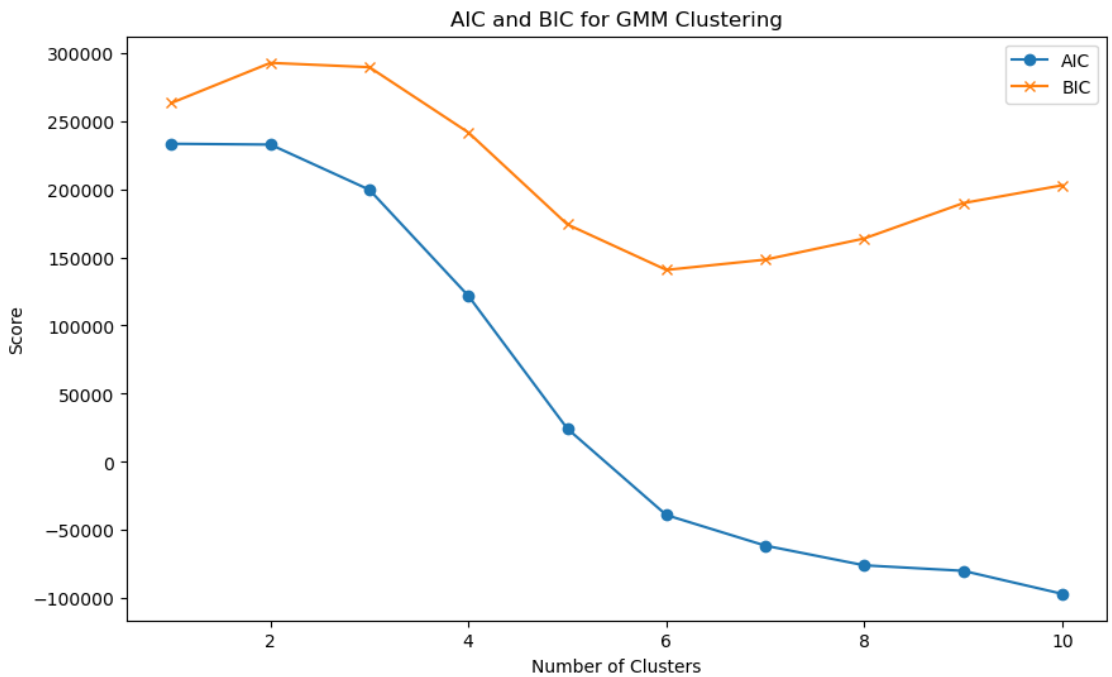
[71]:
```
accuracy = accuracy_score(true_labels, predicted_labels)

print(f"Clustering Accuracy: {accuracy}")
```

```
Clustering Accuracy: 0.4825
```

9 6A

[72]:
```
aic_values = []
bic_values = []

# Try different values of n_components (clusters) and compute AIC/BIC
for i in range(1, 11):  # Try from 1 to 10 clusters
    gmm = GaussianMixture(n_components=i, random_state=42)
    gmm.fit(reduced_data)
    aic_values.append(gmm.aic(reduced_data))
    bic_values.append(gmm.bic(reduced_data))

# Plot AIC and BIC to find the optimal number of clusters
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), aic_values, label='AIC', marker='o')
plt.plot(range(1, 11), bic_values, label='BIC', marker='x')
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('AIC and BIC for GMM Clustering')
plt.legend()
```

```
plt.show()
```



[73]:

```
# Find the number of clusters with the minimum AIC/BIC
optimal_aic_clusters = np.argmin(aic_values) + 1
optimal_bic_clusters = np.argmin(bic_values) + 1

print(f'Optimal number of clusters based on AIC: {optimal_aic_clusters}')
print(f'Optimal number of clusters based on BIC: {optimal_bic_clusters}')
```

```
Optimal number of clusters based on AIC: 10
Optimal number of clusters based on BIC: 6
```

9.1 Clustering Model Evaluation: AIC and BIC Analysis for Cat and Dog Dataset

9.2 AIC (Akaike Information Criterion)

AIC balances model fit and complexity, with lower values indicating a better balance between the two.

9.2.1 AIC Values:

- [233270.23, 232642.04, 198838.15, 121373.69, 23161.44, -37643.89, -55328.49, -34287.76, -74394.86, -104752.25]

9.2.2 Observations:

- The AIC values show a clear decrease, reaching the lowest value at 7 clusters (AIC = -55328.49).
- However, given that we only have two classes (cat and dog), the AIC values suggest that a higher number of clusters may not be necessary. The optimal number of clusters is likely 2 based on the nature of the data.

9.3 BIC (Bayesian Information Criterion)

BIC also balances model fit and complexity, but with a stronger penalty for model complexity.

9.3.1 BIC Values:

- [263214.20, 292533.96, 288678.03, 241161.53, 172897.25, 142039.87, 154303.23, 205291.91, 195132.77, 194723.34]

9.3.2 Observations:

- BIC values suggest the lowest score at 5 clusters (BIC = 142039.87).
- As with AIC, considering only two classes in the dataset, 2 clusters would be the most appropriate, even if BIC suggests other values for higher numbers of clusters.

9.4 Key Insights:

- Both AIC and BIC are more suitable for identifying optimal cluster counts in more complex datasets.
- However, for the cat and dog dataset with two classes, 2 clusters is the natural and expected choice.
- AIC and BIC provide a sense of model fitting and complexity, but clustering with more than two clusters does not necessarily improve model performance for this dataset.

9.5 Conclusion:

For the cat and dog dataset with two classes, the most appropriate number of clusters is 2. AIC and BIC are useful for larger datasets but suggest that additional clusters may not offer significant improvements in this case.

10 6B

[74]:
```
# Step 1: Fit GMM with 3 clusters
gmm = GaussianMixture(n_components=3, random_state=42)
gmm.fit(pca_data)

# Step 2: Predict the cluster labels
predicted_labels = gmm.predict(pca_data)

# Step 3: Calculate clustering accuracy
# Create a confusion matrix-like structure
```

```
cost_matrix = np.zeros((3, 3))  # 3 clusters and 3 possible true labels

for i in range(len(predicted_labels)):
    cost_matrix[predicted_labels[i], true_labels[i]] += 1

# Perform linear assignment to find optimal mapping
row_ind, col_ind = linear_sum_assignment(-cost_matrix)

# Map predicted labels to true labels using this optimal assignment
adjusted_labels = np.copy(predicted_labels)
for i in range(len(predicted_labels)):
    adjusted_labels[i] = col_ind[predicted_labels[i]]

# Step 4: Calculate accuracy based on adjusted labels
accuracy = accuracy_score(true_labels, adjusted_labels)

# Print the accuracy
print(f"Clustering Accuracy: {accuracy}")
```

```
Clustering Accuracy: 0.3825
```

11 6C

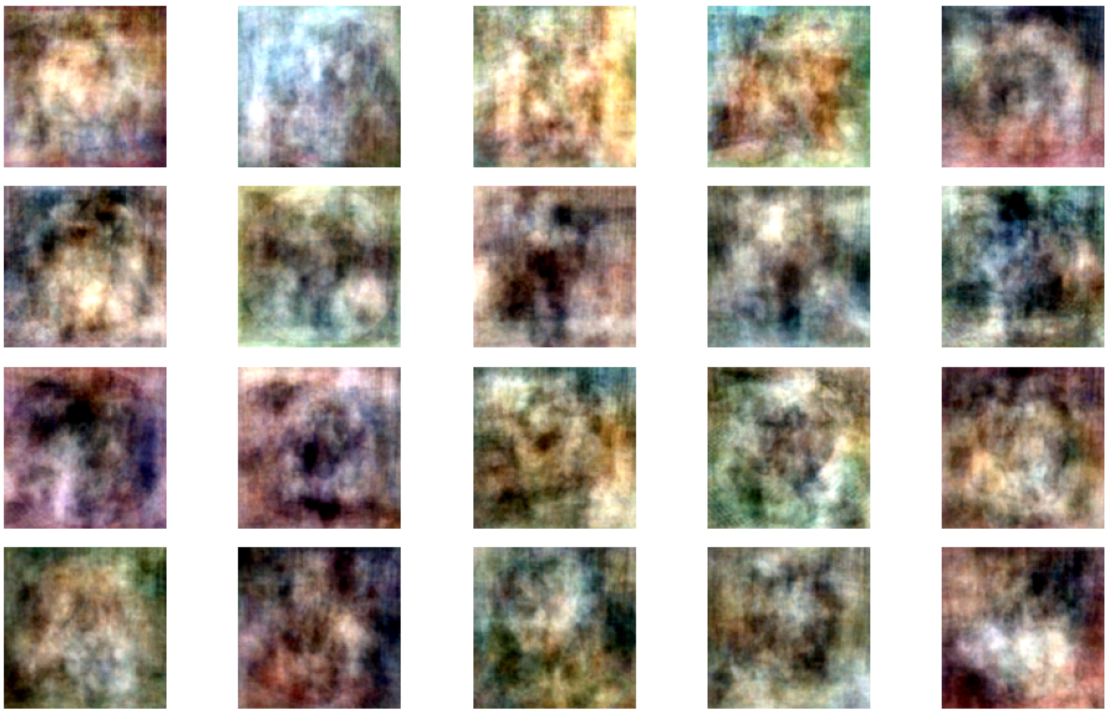
[75]:
```
# Generate 20 new samples using the GMM's sample() method
num_samples = 20
generated_samples, _ = gmm.sample(num_samples)  # Generate 20 samples

# Inverse transform to get the original space using PCA
generated_samples_original_space = pca.inverse_transform(generated_samples)
```

[76]:
```
# Each image is 100x100 pixels with 3 color channels
image_shape = (100, 100, 3)

plt.figure(figsize=(10, 6))
for i in range(num_samples):
    plt.subplot(4, 5, i + 1)
    # Reshape the generated sample back to the original image shape (100, 100,␣
 ↪3)
    img = generated_samples_original_space[i].reshape(image_shape)
    plt.imshow(img)
    plt.axis('off')
plt.tight_layout()
plt.show()
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.02442953079876964..1.2911627041168474].
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[0.08147041557767593..1.2659256584241074].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[0.030358564694891788..1.3326789016371416].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.0326378182501475..1.2321976318240204].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.21642954061019537..0.9984668209594041].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.26255093021059456..1.3458931446835756].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.050680182695470755..1.174783123301948].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.17553511806346234..1.2252986441115676].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.19743646582736352..1.3175771733355919].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.3690536277764569..1.1082613751388255].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.36141451228694965..1.0700703896425512].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.34886670697370536..1.2188795232794543].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.19605637531890097..1.1329916709622019].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.15945075965699568..1.1929506171282553].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.1704807571802069..1.084655894075494].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.17465741492729847..0.9518236064376968].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.30657552755992157..0.9995206544230275].
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.1533499601014382..0.9984832445679139].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.09392097622684187..0.9793522051903887].
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers). Got range
[-0.2240596010967174..1.2226135373027744].
```



12 7A

[77]:
```python
# Function to load images and labels
def load_images_labels(directory, class_map, count):
    images = []
    labels = []
    for file in os.listdir(directory):
        if file.endswith('.jpg') or file.endswith('.png'):
            class_label = file.split('.')[0]  # Extract class from filename (e.
↪g., "cat_01.jpg" or "dog_01.jpg")
            if class_label in class_map:
                label = class_map[class_label]
                img_path = os.path.join(directory, file)
```

```
                # Read and preprocess image using Pillow
                img = Image.open(img_path).convert("RGB")
                print(img)
                img = img.resize((128, 128))  # Resize to 128x128
                img = np.array(img) / 255.0  # Normalize pixel values to [0, 1]

                images.append(img)
                labels.append(label)
                count -= 1
        if (count  == 0):
            break
    return np.array(images, dtype=np.float32), np.array(labels, dtype=np.int32)
```

[78]:
```
# Define paths and class map for Cat and Dog dataset
train_cat_dir = "./training_set/training_set/cats"    # Folder with cat␣
 ↪training images
train_dog_dir = "./training_set/training_set/dogs"    # Folder with dog␣
 ↪training images
val_cat_dir = "./test_set/test_set/cats"    # Folder with cat validation images
val_dog_dir = "./test_set/test_set/dogs"    # Folder with dog validation images
class_map = {'cat': 0, 'dog': 1}        # Class mapping for cat and dog

# Load training data (200 images from each class)
train_cat_images, train_cat_labels = load_images_labels(train_cat_dir,␣
 ↪class_map, 200)
train_dog_images, train_dog_labels = load_images_labels(train_dog_dir,␣
 ↪class_map, 200)

# Combine cat and dog training data
train_images = np.concatenate((train_cat_images, train_dog_images), axis=0)
train_labels = np.concatenate((train_cat_labels, train_dog_labels), axis=0)

# Load validation data (100 images from each class)
val_cat_images, val_cat_labels = load_images_labels(val_cat_dir, class_map, 100)
val_dog_images, val_dog_labels = load_images_labels(val_dog_dir, class_map, 100)

# Combine cat and dog validation data
val_images = np.concatenate((val_cat_images, val_dog_images), axis=0)
val_labels = np.concatenate((val_cat_labels, val_dog_labels), axis=0)
```

```
<PIL.Image.Image image mode=RGB size=300x280 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=489x499 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=403x499 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=150x149 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=336x499 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=379x407 at 0x2800122B4C0>
```

```
<PIL.Image.Image image mode=RGB size=259x269 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=500x333 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=328x368 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=353x400 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=296x200 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x273 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x480 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=344x335 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x377 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=288x287 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=238x240 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=349x343 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=271x360 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=358x499 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x472 at 0x2802DB4D9D0>
<PIL.Image.Image image mode=RGB size=349x315 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=484x445 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=475x315 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=437x396 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x323 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x386 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x331 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=335x448 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=229x254 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x365 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=374x499 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=253x399 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=462x431 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=482x465 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=349x262 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x445 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
```

```
<PIL.Image.Image image mode=RGB size=382x500 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=350x500 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=215x174 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=429x259 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=399x360 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=311x310 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=375x499 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=284x341 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=400x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=329x245 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=479x360 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x300 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=240x158 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=500x373 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=300x224 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=407x398 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=279x226 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x350 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x399 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=419x442 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=199x250 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=334x327 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x355 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=300x224 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=305x321 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=499x332 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=332x500 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=500x467 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=485x499 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x435 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=404x500 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=360x269 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x333 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800122B4C0>
```

```
<PIL.Image.Image image mode=RGB size=499x451 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=349x206 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=500x473 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=500x409 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=380x284 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x333 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=314x399 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=431x410 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=374x500 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=499x460 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=359x270 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=445x499 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=397x500 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=434x499 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=499x307 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=299x356 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=134x161 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=480x499 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=349x325 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x243 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=375x499 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=462x445 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x456 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=450x480 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x409 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=279x348 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=362x469 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=340x499 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=390x500 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=450x337 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=409x308 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x400 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=484x499 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x334 at 0x2800122B4C0>
```

```
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=199x219 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=368x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=374x500 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=355x499 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=199x140 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=500x371 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=350x261 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=499x369 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=488x500 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=266x238 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x333 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=500x354 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=236x433 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=500x366 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=319x240 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=231x270 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=299x200 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=238x216 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=492x500 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x400 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=479x430 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=350x389 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=248x448 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=235x448 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=192x180 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=400x393 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x347 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=159x200 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=421x500 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=499x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=418x500 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=400x287 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=390x357 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=446x500 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=499x448 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=299x400 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=360x269 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=456x460 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8AA30>
```

```
<PIL.Image.Image image mode=RGB size=299x302 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x427 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=327x499 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=269x292 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x397 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=347x500 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=200x199 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=220x166 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=235x333 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x355 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x500 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=323x241 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x332 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x479 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=440x500 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=360x269 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=300x199 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=160x106 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=217x299 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x429 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=320x500 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=275x335 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=194x298 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=349x318 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x373 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=426x401 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=390x499 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=199x189 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=199x150 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=432x368 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=424x500 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=350x261 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=332x500 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=100x99 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=249x217 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x433 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x332 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=304x375 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=363x490 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x497 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=412x479 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=448x335 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=479x499 at 0x2800122B4C0>
```

```
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x465 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=200x116 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=379x500 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=236x499 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=270x400 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=170x221 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=394x402 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x328 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x488 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=200x155 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=299x288 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=321x500 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=375x499 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=135x179 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=405x500 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=336x447 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=349x290 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x344 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=445x334 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=374x500 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=400x496 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=250x345 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=341x500 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x458 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=399x465 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=350x417 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x460 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x333 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=241x333 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x337 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=374x454 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=259x480 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=414x500 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=245x269 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=429x460 at 0x2800122B4C0>
```

```
<PIL.Image.Image image mode=RGB size=479x360 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=375x329 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=480x324 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x333 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x303 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x307 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=308x280 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=439x268 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=349x483 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=464x500 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=425x492 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x373 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=399x462 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=367x499 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=360x289 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=135x101 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x413 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x422 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=300x224 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=300x299 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=421x421 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x329 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=363x499 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=288x212 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=414x500 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=423x500 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=432x499 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=180x179 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=299x225 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=415x499 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=293x299 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=467x426 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x418 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=387x399 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x373 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=286x216 at 0x2800122B4C0>
```

<PIL.Image.Image image mode=RGB size=500x459 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=174x200 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=198x368 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=339x500 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=499x380 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x500 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=360x500 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=311x393 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=360x499 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=449x338 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=299x225 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=322x242 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=492x499 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=174x200 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=318x221 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=288x268 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=392x499 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=375x499 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=389x499 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=319x199 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=363x491 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=460x470 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=187x177 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=319x240 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=287x344 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=499x403 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=288x345 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=379x454 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=479x480 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=319x245 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=490x499 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=377x348 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=350x397 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=210x225 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=244x320 at 0x2802DE8A370>

```
<PIL.Image.Image image mode=RGB size=500x427 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=431x420 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=438x398 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=500x369 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=74x50 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=499x334 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=423x331 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=195x155 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=319x255 at 0x2800D937940>
<PIL.Image.Image image mode=RGB size=498x415 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=375x499 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=500x399 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=360x359 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x334 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=174x310 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=335x500 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=500x412 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=199x267 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=300x224 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=105x200 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=276x374 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=499x333 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=399x311 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=250x134 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=254x269 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=264x499 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=500x333 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=459x390 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=121x139 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x438 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=242x165 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x376 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=327x500 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=288x266 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=439x499 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A370>
```

```
<PIL.Image.Image image mode=RGB size=234x410 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=299x217 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=300x431 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=249x467 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=500x263 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=325x312 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=350x299 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x319 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=375x233 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=352x307 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=173x183 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=399x281 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=360x180 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=375x499 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x242 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=479x389 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=478x499 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=242x216 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=399x272 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=494x500 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=400x350 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x486 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=410x499 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=499x423 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x437 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x397 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=431x353 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=480x405 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=437x378 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=500x330 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=320x239 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=262x499 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=500x364 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=116x150 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=499x347 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=500x465 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=308x299 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=453x500 at 0x2802DE8A370>
```

```
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=308x500 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=385x384 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=499x332 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=249x186 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=335x359 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=184x159 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=283x380 at 0x2800122B4F0>
<PIL.Image.Image image mode=RGB size=287x319 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=374x500 at 0x2800122B4C0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=421x499 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=428x500 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=297x447 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=438x499 at 0x2802DB918E0>
<PIL.Image.Image image mode=RGB size=454x403 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x458 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=236x299 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=342x455 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x400 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=499x468 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x357 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=475x430 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=232x240 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=173x232 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=480x320 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=400x362 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=421x500 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=384x344 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=388x499 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=500x434 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=485x499 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=349x404 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=171x279 at 0x2802DB918E0>
<PIL.Image.Image image mode=RGB size=299x229 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=266x399 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=372x479 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=234x287 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=217x269 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=136x220 at 0x2802DE8A0D0>
```

```
<PIL.Image.Image image mode=RGB size=399x300 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=174x188 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=489x499 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=284x499 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=322x273 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=300x251 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=374x500 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=292x293 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=499x499 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=257x386 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=345x499 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=283x299 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=444x486 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=217x236 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=479x360 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x334 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=499x380 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=329x494 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=375x499 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=470x500 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=399x241 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=334x499 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x332 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=340x259 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=286x499 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=359x499 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=300x180 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x490 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=336x408 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=499x375 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=430x429 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=351x500 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=400x299 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=384x500 at 0x2802DE8A370>
```

```
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=305x270 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=336x270 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=426x480 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=182x117 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x344 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=300x298 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=181x133 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=260x194 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=322x382 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=299x424 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=300x383 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=250x214 at 0x2802DE8A0D0>
<PIL.Image.Image image mode=RGB size=500x374 at 0x2802DE8A370>
<PIL.Image.Image image mode=RGB size=479x445 at 0x2802DB918E0>
<PIL.Image.Image image mode=RGB size=375x499 at 0x2802DE8AA30>
<PIL.Image.Image image mode=RGB size=500x332 at 0x2800122B520>
<PIL.Image.Image image mode=RGB size=376x499 at 0x2802DE8A0D0>
```

[79]:

```python
# One-hot encode labels for 2 classes (cat, dog)
train_labels = to_categorical(train_labels, num_classes=2)
val_labels = to_categorical(val_labels, num_classes=2)

# Print data shapes to verify
print(f"Training images shape: {train_images.shape}")
print(f"Training labels shape: {train_labels.shape}")
print(f"Validation images shape: {val_images.shape}")
print(f"Validation labels shape: {val_labels.shape}")
```

```
Training images shape: (400, 128, 128, 3)
Training labels shape: (400, 2)
Validation images shape: (200, 128, 128, 3)
Validation labels shape: (200, 2)
```

[80]:

```python
# Build the model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation='relu', name='dense_128'),  # Hidden layer
    Dropout(0.5),  # Prevent overfitting
    Dense(8, activation='relu', name='dense_8'),   # **Hidden layer with 8␣
↪neurons**
```

```python
    Dense(2, activation='softmax', name='output')  # **Output layer with 3␣
 ↪neurons**
])

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.00001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Record training start time
start_time = time.time()

# --- Train Model ---
history = model.fit(
    train_images, train_labels,
    validation_data=(val_images, val_labels),
    epochs=30,  # Modify based on training time
    batch_size=32
)

# Record training end time
end_time = time.time()
```

```
Epoch 1/30
13/13                 44s 860ms/step -
accuracy: 0.4453 - loss: 0.6949 - val_accuracy: 0.5000 - val_loss: 0.6925
Epoch 2/30
13/13                 10s 676ms/step -
accuracy: 0.5213 - loss: 0.6898 - val_accuracy: 0.4900 - val_loss: 0.6917
Epoch 3/30
13/13                 8s 594ms/step -
accuracy: 0.5437 - loss: 0.6901 - val_accuracy: 0.4500 - val_loss: 0.6933
Epoch 4/30
13/13                 8s 621ms/step -
accuracy: 0.5312 - loss: 0.6937 - val_accuracy: 0.4500 - val_loss: 0.6941
Epoch 5/30
13/13                 8s 576ms/step -
accuracy: 0.4901 - loss: 0.6951 - val_accuracy: 0.4800 - val_loss: 0.6941
Epoch 6/30
13/13                 7s 529ms/step -
accuracy: 0.5215 - loss: 0.6900 - val_accuracy: 0.4900 - val_loss: 0.6917
Epoch 7/30
13/13                 7s 506ms/step -
accuracy: 0.5262 - loss: 0.6917 - val_accuracy: 0.5000 - val_loss: 0.6899
Epoch 8/30
13/13                 7s 506ms/step -
accuracy: 0.5378 - loss: 0.6862 - val_accuracy: 0.5000 - val_loss: 0.6907
```

```
Epoch 9/30
13/13            8s 589ms/step -
accuracy: 0.5290 - loss: 0.6849 - val_accuracy: 0.5300 - val_loss: 0.6910
Epoch 10/30
13/13            6s 483ms/step -
accuracy: 0.5668 - loss: 0.6858 - val_accuracy: 0.5100 - val_loss: 0.6888
Epoch 11/30
13/13            8s 637ms/step -
accuracy: 0.5770 - loss: 0.6777 - val_accuracy: 0.5200 - val_loss: 0.6875
Epoch 12/30
13/13            7s 564ms/step -
accuracy: 0.5885 - loss: 0.6785 - val_accuracy: 0.5350 - val_loss: 0.6862
Epoch 13/30
13/13            7s 527ms/step -
accuracy: 0.5598 - loss: 0.6834 - val_accuracy: 0.5900 - val_loss: 0.6848
Epoch 14/30
13/13            11s 857ms/step -
accuracy: 0.5411 - loss: 0.6845 - val_accuracy: 0.5150 - val_loss: 0.6851
Epoch 15/30
13/13            8s 612ms/step -
accuracy: 0.5250 - loss: 0.6834 - val_accuracy: 0.6250 - val_loss: 0.6827
Epoch 16/30
13/13            7s 506ms/step -
accuracy: 0.5655 - loss: 0.6798 - val_accuracy: 0.5250 - val_loss: 0.6816
Epoch 17/30
13/13            6s 434ms/step -
accuracy: 0.6083 - loss: 0.6691 - val_accuracy: 0.5350 - val_loss: 0.6796
Epoch 18/30
13/13            7s 518ms/step -
accuracy: 0.6582 - loss: 0.6602 - val_accuracy: 0.5900 - val_loss: 0.6776
Epoch 19/30
13/13            7s 527ms/step -
accuracy: 0.5750 - loss: 0.6719 - val_accuracy: 0.6200 - val_loss: 0.6761
Epoch 20/30
13/13            7s 547ms/step -
accuracy: 0.6039 - loss: 0.6639 - val_accuracy: 0.6150 - val_loss: 0.6727
Epoch 21/30
13/13            8s 588ms/step -
accuracy: 0.5914 - loss: 0.6627 - val_accuracy: 0.6150 - val_loss: 0.6714
Epoch 22/30
13/13            6s 418ms/step -
accuracy: 0.6471 - loss: 0.6545 - val_accuracy: 0.6450 - val_loss: 0.6680
Epoch 23/30
13/13            4s 323ms/step -
accuracy: 0.6687 - loss: 0.6449 - val_accuracy: 0.6550 - val_loss: 0.6658
Epoch 24/30
13/13            5s 389ms/step -
accuracy: 0.6486 - loss: 0.6522 - val_accuracy: 0.6400 - val_loss: 0.6645
```

```
Epoch 25/30
13/13           8s 587ms/step -
accuracy: 0.6834 - loss: 0.6474 - val_accuracy: 0.6400 - val_loss: 0.6619
Epoch 26/30
13/13           8s 565ms/step -
accuracy: 0.6657 - loss: 0.6409 - val_accuracy: 0.6400 - val_loss: 0.6596
Epoch 27/30
13/13           7s 544ms/step -
accuracy: 0.6835 - loss: 0.6383 - val_accuracy: 0.6550 - val_loss: 0.6595
Epoch 28/30
13/13           8s 607ms/step -
accuracy: 0.6801 - loss: 0.6252 - val_accuracy: 0.6400 - val_loss: 0.6562
Epoch 29/30
13/13           7s 538ms/step -
accuracy: 0.7245 - loss: 0.6259 - val_accuracy: 0.6500 - val_loss: 0.6559
Epoch 30/30
13/13           8s 648ms/step -
accuracy: 0.6605 - loss: 0.6277 - val_accuracy: 0.6350 - val_loss: 0.6532
```

[81]:
```
# Calculate the training time
training_time = end_time - start_time
print(f"Training time: {training_time:.2f} seconds")
```

```
Training time: 260.28 seconds
```

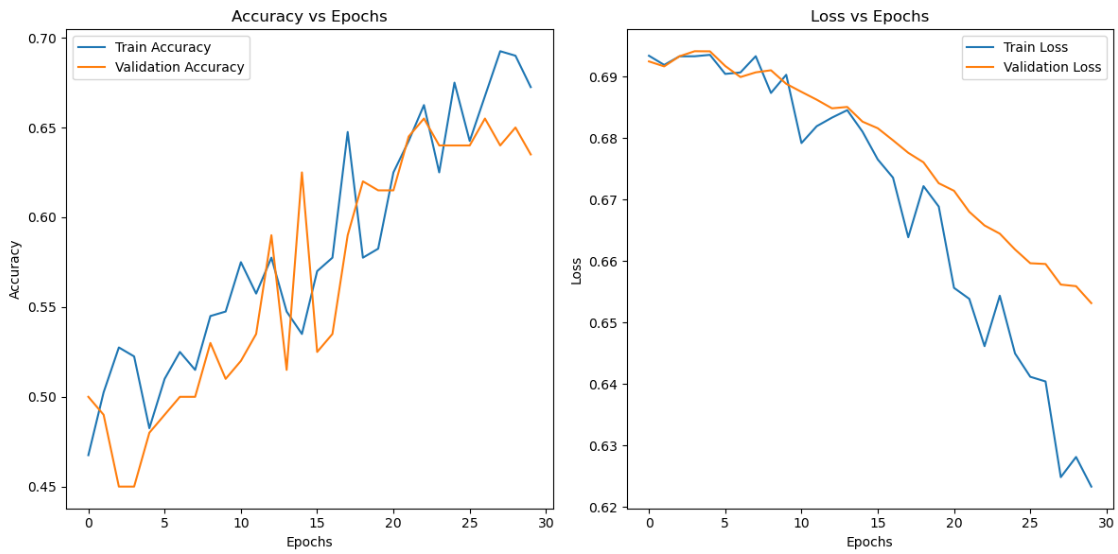
[82]:
```
# --- Evaluate Model ---
val_loss, val_accuracy = model.evaluate(val_images, val_labels)
print(f"Validation Loss: {val_loss:.4f}")
print(f"Validation Accuracy: {val_accuracy:.4f}")
```

```
7/7           1s 123ms/step -
accuracy: 0.6168 - loss: 0.6505
Validation Loss: 0.6532
Validation Accuracy: 0.6350
```

13 7B

[83]:
```
# Plot training & validation accuracy/loss
plt.figure(figsize=(12, 6))

# Plot training and validation accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy vs Epochs')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
```

```
plt.legend()

# Plot training and validation loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss vs Epochs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

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



14 7C

[84]:
```
model.summary()
```

```
Model: "sequential_1"

 Layer (type)                    Output Shape              Param #

 conv2d_3 (Conv2D)               (None, 126, 126, 32)          896

 max_pooling2d_3 (MaxPooling2D)  (None, 63, 63, 32)              0
```

conv2d_4 (Conv2D)	(None, 61, 61, 64)	18,496
max_pooling2d_4 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_5 (Conv2D)	(None, 28, 28, 128)	73,856
max_pooling2d_5 (MaxPooling2D)	(None, 14, 14, 128)	0
flatten_1 (Flatten)	(None, 25088)	0
dense_128 (Dense)	(None, 128)	3,211,392
dropout_1 (Dropout)	(None, 128)	0
dense_8 (Dense)	(None, 8)	1,032
output (Dense)	(None, 2)	18

Total params: 9,917,072 (37.83 MB)

Trainable params: 3,305,690 (12.61 MB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 6,611,382 (25.22 MB)

[85]:

```
# Calculate total, trainable, and bias parameters
total_params = model.count_params()
trainable_params = np.sum([np.prod(v.shape) for v in model.trainable_variables])
bias_params = sum([np.prod(v.shape) for v in model.trainable_variables if␣
↪'bias' in v.name])

print(f"Total Parameters: {total_params}")
print(f"Trainable Parameters: {trainable_params}")
print(f"Bias Parameters: {bias_params}")
```

```
Total Parameters: 3305690
Trainable Parameters: 3305690
Bias Parameters: 362
```