In [6]:
```
import os

wikiart_folder = "./wikiart/"

# Mapped style names to class indices
style_classes = {
    "Abstract_Expressionism": 0,
    "Action_painting": 1,
    "Analytical_Cubism": 2,
    "Art_Nouveau_Modern": 3,
    "Baroque": 4,
    "Color_Field_Painting": 5,
    "Contemporary_Realism": 6,
    "Cubism": 7,
    "Early_Renaissance": 8,
    "Expressionism": 9,
    "Fauvism": 10,
    "High_Renaissance": 11,
    "Impressionism": 12,
    "Mannerism_Late_Renaissance": 13,
    "Minimalism": 14,
    "Naive_Art_Primitivism": 15,
    "New_Realism": 16,
    "Northern_Renaissance": 17,
    "Pointillism": 18,
    "Pop_Art": 19,
    "Post_Impressionism": 20,
    "Realism": 21,
    "Rococo": 22,
    "Romanticism": 23,
    "Symbolism": 24,
    "Synthetic_Cubism": 25,
    "Ukiyo_e": 26
}

image_paths = []
image_labels = []

for style_name, class_index in style_classes.items():
    style_folder = os.path.join(wikiart_folder, style_name)
    for img_name in os.listdir(style_folder):
        img_path = os.path.join(style_folder, img_name)
        image_paths.append(img_path)
        image_labels.append(class_index)

print(f"Total images: {len(image_paths)}")
```

```
Total images: 81444
```

image preprocessing

In [7]:
```
import os

for img_path in image_paths:
    if not os.path.exists(img_path):
        print(f"File does not exist: {img_path}")
```

In [13]:
```
import os
import numpy as np
import pandas as pd
from PIL import Image
from tqdm import tqdm
import gc
```

In [8]:
```
img_size = (224, 224)             # Fixed size for resizing
batch_size = 1000                      # Number of images per batch
output_folder = "processed_batches_styles"
wikiart_base = "./wikiart/"
csv_folder = "./wikiart_csv"

os.makedirs(output_folder, exist_ok=True)

# here we demonstrate the training set; and then repeating similar steps for val
csv_file = os.path.join(csv_folder, "style_train.csv")

# csv rows are: [relative_image_path, class_index]
data = pd.read_csv(csv_file, header=None, names=["img_path", "class_index"])
image_paths = data["img_path"].tolist()
image_labels = data["class_index"].tolist()

def preprocess_image(img_path):
    #Load an image, convert to RGB, resize, and normalize.
    full_path = os.path.join(wikiart_base, img_path)
    try:
        with Image.open(full_path) as img:
            img = img.convert("RGB")
            img_resized = img.resize(img_size)
            img_array = np.array(img_resized, dtype=np.float32) / 255.0  # Norma
            return img_array
    except Exception as e:
        print(f"Failed to preprocess {full_path}: {e}")
        return None

def preprocess_in_batches(image_paths, image_labels, batch_size, output_folder):
    for batch_idx in tqdm(range(0, len(image_paths), batch_size), desc="Processi
        batch_paths = image_paths[batch_idx:batch_idx + batch_size]
        batch_labels = image_labels[batch_idx:batch_idx + batch_size]
        batch_images = []

        for img_path in batch_paths:
            img = preprocess_image(img_path)
            if img is not None:
                batch_images.append(img)

        # If an image fails, we remove its label as well
        if len(batch_images) == 0:
            print(f"Warning: No valid images in batch starting at index {batch_i
            continue

        batch_images_array = np.array(batch_images)

        batch_labels_array = np.array(batch_labels[:len(batch_images)], dtype=np

        batch_file = os.path.join(output_folder, f"batch_{batch_idx // batch_siz
        np.savez_compressed(batch_file, arr_0=batch_images_array, labels=batch_l
        print(f"Saved batch {batch_idx // batch_size + 1} with {len(batch_images
```

```
        # Free memory
        del batch_images, batch_images_array, batch_labels_array
        gc.collect()

preprocess_in_batches(image_paths, image_labels, batch_size, output_folder)

print("Batch processing with labels completed!")
```

```
Processing Batches:   2%|
| 1/58 [01:36<1:31:50, 96.67s/it]
Saved batch 1 with 1000 images.
Processing Batches:   3%|
| 2/58 [03:12<1:29:58, 96.41s/it]
Saved batch 2 with 1000 images.
Processing Batches:   5%|
| 3/58 [04:50<1:28:58, 97.06s/it]
Saved batch 3 with 1000 images.
Processing Batches:   7%|
| 4/58 [06:28<1:27:30, 97.23s/it]
Saved batch 4 with 1000 images.
C:\Users\KIIT\AppData\Roaming\Python\Python312\site-packages\PIL\Image.py:3368: D
ecompressionBombWarning: Image size (99962094 pixels) exceeds limit of 89478485 p
ixels, could be decompression bomb DOS attack.
  warnings.warn(
Processing Batches:   9%|
| 5/58 [08:03<1:25:11, 96.44s/it]
Saved batch 5 with 1000 images.
Processing Batches:  10%|
| 6/58 [09:25<1:19:16, 91.48s/it]
Saved batch 6 with 1000 images.
Processing Batches:  12%|
| 7/58 [10:37<1:12:33, 85.36s/it]
Saved batch 7 with 1000 images.
Processing Batches:  14%|
| 8/58 [11:48<1:07:16, 80.74s/it]
Saved batch 8 with 1000 images.
Processing Batches:  16%|
| 9/58 [12:57<1:02:48, 76.91s/it]
Saved batch 9 with 1000 images.
Processing Batches:  17%|
| 10/58 [14:05<59:28, 74.34s/it]
Saved batch 10 with 1000 images.
Processing Batches:  19%|
| 11/58 [15:14<56:55, 72.67s/it]
Saved batch 11 with 1000 images.
Processing Batches:  21%|
| 12/58 [16:24<55:02, 71.80s/it]
Saved batch 12 with 1000 images.
Processing Batches:  22%|
| 13/58 [17:32<53:05, 70.79s/it]
Saved batch 13 with 1000 images.
Processing Batches:  24%|
| 14/58 [18:41<51:22, 70.07s/it]
Saved batch 14 with 1000 images.
```

```
Processing Batches:  26%|███████████████
| 15/58 [19:51<50:13, 70.09s/it]
Saved batch 15 with 1000 images.
Processing Batches:  28%|████████████████
| 16/58 [21:00<48:55, 69.90s/it]
Saved batch 16 with 1000 images.
Processing Batches:  29%|████████████████▏
| 17/58 [22:11<47:47, 69.95s/it]
Saved batch 17 with 1000 images.
Processing Batches:  31%|█████████████████▏
| 18/58 [23:20<46:30, 69.76s/it]
Saved batch 18 with 1000 images.
C:\Users\KIIT\AppData\Roaming\Python\Python312\site-packages\PIL\Image.py:3368: D
ecompressionBombWarning: Image size (107327830 pixels) exceeds limit of 89478485
pixels, could be decompression bomb DOS attack.
  warnings.warn(
Processing Batches:  33%|██████████████████▏
| 19/58 [24:30<45:21, 69.77s/it]
Saved batch 19 with 1000 images.
Processing Batches:  34%|███████████████████▏
| 20/58 [25:40<44:12, 69.81s/it]
Saved batch 20 with 1000 images.
Processing Batches:  36%|████████████████████▏
| 21/58 [26:49<42:57, 69.65s/it]
Saved batch 21 with 1000 images.
Processing Batches:  38%|█████████████████████▏
| 22/58 [27:59<41:55, 69.87s/it]
Saved batch 22 with 1000 images.
Processing Batches:  40%|██████████████████████▏
| 23/58 [29:12<41:12, 70.66s/it]
Saved batch 23 with 1000 images.
Processing Batches:  41%|███████████████████████▏
| 24/58 [30:20<39:43, 70.10s/it]
Saved batch 24 with 1000 images.
Processing Batches:  43%|████████████████████████▏
| 25/58 [31:31<38:40, 70.32s/it]
Saved batch 25 with 1000 images.
Processing Batches:  45%|█████████████████████████▏
| 26/58 [32:41<37:27, 70.24s/it]
Saved batch 26 with 1000 images.
Processing Batches:  47%|██████████████████████████▏
| 27/58 [33:53<36:28, 70.58s/it]
Saved batch 27 with 1000 images.
Processing Batches:  48%|███████████████████████████▏
| 28/58 [35:01<34:54, 69.83s/it]
Saved batch 28 with 1000 images.
Processing Batches:  50%|████████████████████████████
| 29/58 [36:11<33:47, 69.91s/it]
Saved batch 29 with 1000 images.
Processing Batches:  52%|█████████████████████████████
| 30/58 [37:20<32:29, 69.64s/it]
Saved batch 30 with 1000 images.
Processing Batches:  53%|█████████████████████████████▏
| 31/58 [38:29<31:13, 69.39s/it]
Saved batch 31 with 1000 images.
```

```
Processing Batches:  55%|
| 32/58 [39:39<30:09, 69.59s/it]
Saved batch 32 with 1000 images.
Processing Batches:  57%|
| 33/58 [40:49<29:05, 69.81s/it]
Saved batch 33 with 1000 images.
Processing Batches:  59%|
| 34/58 [41:59<27:58, 69.95s/it]
Saved batch 34 with 1000 images.
Processing Batches:  60%|
| 35/58 [43:11<27:01, 70.52s/it]
Saved batch 35 with 1000 images.
Processing Batches:  62%|
| 36/58 [44:21<25:44, 70.22s/it]
Saved batch 36 with 1000 images.
Processing Batches:  64%|
| 37/58 [45:31<24:37, 70.37s/it]
Saved batch 37 with 1000 images.
Processing Batches:  66%|
| 38/58 [46:41<23:19, 69.98s/it]
Saved batch 38 with 1000 images.
Failed to preprocess ./wikiart/Baroque/rembrandt_woman-standing-with-raised-hand
s.jpg: [Errno 2] No such file or directory: 'C:\\Users\\KIIT\\Desktop\\UJJU\\PROJ
ECT\\GSOC(proposal)\\ArtExtract\\TASK-1\\wikiart\\Baroque\\rembrandt_woman-standi
ng-with-raised-hands.jpg'
Processing Batches:  67%|
| 39/58 [47:50<22:04, 69.69s/it]
Saved batch 39 with 999 images.
Processing Batches:  69%|
| 40/58 [49:00<20:59, 69.96s/it]
Saved batch 40 with 1000 images.
Processing Batches:  71%|
| 41/58 [50:09<19:45, 69.73s/it]
Saved batch 41 with 1000 images.
Processing Batches:  72%|
| 42/58 [51:19<18:32, 69.56s/it]
Saved batch 42 with 1000 images.
Processing Batches:  74%|
| 43/58 [52:28<17:24, 69.64s/it]
Saved batch 43 with 1000 images.
Processing Batches:  76%|
| 44/58 [53:39<16:17, 69.84s/it]
Saved batch 44 with 1000 images.
Failed to preprocess ./wikiart/Post_Impressionism/vincent-van-gogh_l-arlesienne-p
ortrait-of-madame-ginoux-1890.jpg: [Errno 2] No such file or directory: 'C:\\User
s\\KIIT\\Desktop\\UJJU\\PROJECT\\GSOC(proposal)\\ArtExtract\\TASK-1\\wikiart\\Pos
t_Impressionism\\vincent-van-gogh_l-arlesienne-portrait-of-madame-ginoux-1890.jp
g'
Processing Batches:  78%|
| 45/58 [54:51<15:16, 70.54s/it]
Saved batch 45 with 999 images.
Processing Batches:  79%|
| 46/58 [56:00<14:03, 70.27s/it]
Saved batch 46 with 1000 images.
```

```
Processing Batches:  81%|████████████████████████████████████████████████▌
| 47/58 [57:12<12:56, 70.57s/it]
Saved batch 47 with 1000 images.
Processing Batches:  83%|█████████████████████████████████████████████████▍
| 48/58 [58:21<11:41, 70.17s/it]
Saved batch 48 with 1000 images.
Saved batch 49 with 1000 images.
Processing Batches:  86%|██████████████████████████████████████████████████▎
| 50/58 [1:00:43<09:25, 70.69s/it]
Saved batch 50 with 1000 images.
Processing Batches:  88%|███████████████████████████████████████████████████▏
| 51/58 [1:01:52<08:10, 70.09s/it]
Saved batch 51 with 1000 images.
Processing Batches:  90%|████████████████████████████████████████████████████
| 52/58 [1:03:04<07:03, 70.60s/it]
Saved batch 52 with 1000 images.
Processing Batches:  91%|████████████████████████████████████████████████████▉
| 53/58 [1:04:13<05:51, 70.30s/it]
Saved batch 53 with 1000 images.
Processing Batches:  93%|█████████████████████████████████████████████████████▊
█     | 54/58 [1:05:23<04:40, 70.18s/it]
Saved batch 54 with 1000 images.
Processing Batches:  95%|██████████████████████████████████████████████████████▋
██    | 55/58 [1:06:31<03:28, 69.35s/it]
Saved batch 55 with 1000 images.
Processing Batches:  97%|███████████████████████████████████████████████████████▌
███   | 56/58 [1:07:41<02:19, 69.81s/it]
Saved batch 56 with 1000 images.
Processing Batches:  98%|████████████████████████████████████████████████████████▍
███▌ | 57/58 [1:08:50<01:09, 69.38s/it]
Saved batch 57 with 1000 images.
Processing Batches: 100%|█████████████████████████████████████████████████████████▎
████| 58/58 [1:08:52<00:00, 71.25s/it]
Saved batch 58 with 25 images.
Batch processing with labels completed!
```

In [9]:

```python
img_size = (224, 224)
batch_size = 1000
output_folder = "processed_batches_styles_val"
wikiart_base = "./wikiart/"
csv_folder = "./wikiart_csv"

os.makedirs(output_folder, exist_ok=True)

csv_file = os.path.join(csv_folder, "style_val.csv")
data = pd.read_csv(csv_file, header=None, names=["img_path", "class_index"])
image_paths = data["img_path"].tolist()
image_labels = data["class_index"].tolist()


def preprocess_image(img_path):
    """Load an image, convert to RGB, resize, and normalize."""
    full_path = os.path.join(wikiart_base, img_path)
    try:
        with Image.open(full_path) as img:
            img = img.convert("RGB")
```

```
            img_resized = img.resize(img_size)
            img_array = np.array(img_resized, dtype=np.float32) / 255.0
            return img_array
    except Exception as e:
        print(f"Failed to preprocess {full_path}: {e}")
        return None

def preprocess_in_batches(image_paths, image_labels, batch_size, output_folder):
    for batch_idx in tqdm(range(0, len(image_paths), batch_size), desc="Processi
        batch_paths = image_paths[batch_idx:batch_idx + batch_size]
        batch_labels = image_labels[batch_idx:batch_idx + batch_size]
        batch_images = []

        for img_path in batch_paths:
            img = preprocess_image(img_path)
            if img is not None:
                batch_images.append(img)


        if len(batch_images) == 0:
            print(f"Warning: No valid images in batch starting at index {batch_i
            continue

        batch_images_array = np.array(batch_images)
        batch_labels_array = np.array(batch_labels[:len(batch_images)], dtype=np

        batch_file = os.path.join(output_folder, f"batch_{batch_idx // batch_siz
        np.savez_compressed(batch_file, arr_0=batch_images_array, labels=batch_l
        print(f"Saved batch {batch_idx // batch_size + 1} with {len(batch_images

        del batch_images, batch_images_array, batch_labels_array
        gc.collect()

preprocess_in_batches(image_paths, image_labels, batch_size, output_folder)

print("Batch processing with labels completed!")
```

```
Processing Batches:   4%|███
| 1/25 [01:13<29:24, 73.50s/it]
Saved batch 1 with 1000 images.
Processing Batches:   8%|█████
| 2/25 [02:27<28:15, 73.71s/it]
Saved batch 2 with 1000 images.
Processing Batches:  12%|███████
| 3/25 [03:49<28:30, 77.74s/it]
Saved batch 3 with 1000 images.
Processing Batches:  16%|██████████
| 4/25 [05:28<30:08, 86.13s/it]
Saved batch 4 with 1000 images.
Processing Batches:  20%|████████████
| 5/25 [06:39<26:52, 80.62s/it]
Saved batch 5 with 1000 images.
Processing Batches:  24%|██████████████
| 6/25 [07:46<24:05, 76.06s/it]
Saved batch 6 with 1000 images.
Processing Batches:  28%|████████████████
| 7/25 [08:53<21:52, 72.92s/it]
Saved batch 7 with 1000 images.
```

```
Processing Batches:  32%|
| 8/25 [09:59<20:04, 70.84s/it]
Saved batch 8 with 1000 images.
Processing Batches:  36%|
| 9/25 [11:08<18:41, 70.07s/it]
Saved batch 9 with 1000 images.
Processing Batches:  40%|
| 10/25 [12:20<17:40, 70.70s/it]
Saved batch 10 with 1000 images.
Processing Batches:  44%|
| 11/25 [13:27<16:14, 69.62s/it]
Saved batch 11 with 1000 images.
Processing Batches:  48%|
| 12/25 [14:32<14:48, 68.34s/it]
Saved batch 12 with 1000 images.
Processing Batches:  52%|
| 13/25 [15:49<14:09, 70.82s/it]
Saved batch 13 with 1000 images.
Processing Batches:  56%|
| 14/25 [17:03<13:10, 71.87s/it]
Saved batch 14 with 1000 images.
Processing Batches:  60%|
| 15/25 [18:18<12:06, 72.69s/it]
Saved batch 15 with 1000 images.
Processing Batches:  64%|
| 16/25 [19:37<11:12, 74.69s/it]
Saved batch 16 with 1000 images.
Processing Batches:  68%|
| 17/25 [20:46<09:44, 73.10s/it]
Saved batch 17 with 1000 images.
Processing Batches:  72%|
| 18/25 [21:53<08:17, 71.08s/it]
Saved batch 18 with 1000 images.
Processing Batches:  76%|
| 19/25 [23:02<07:02, 70.37s/it]
Saved batch 19 with 1000 images.
Processing Batches:  80%|
| 20/25 [24:11<05:50, 70.06s/it]
Saved batch 20 with 1000 images.
Processing Batches:  84%|
| 21/25 [25:19<04:37, 69.39s/it]
Saved batch 21 with 1000 images.
Processing Batches:  88%|
| 22/25 [26:15<03:16, 65.49s/it]
Saved batch 22 with 1000 images.
Processing Batches:  92%|
     | 23/25 [27:25<02:13, 66.72s/it]
Saved batch 23 with 1000 images.
Processing Batches:  96%|
    | 24/25 [28:24<01:04, 64.52s/it]
Saved batch 24 with 1000 images.
Processing Batches: 100%|
| 25/25 [28:52<00:00, 69.31s/it]
Saved batch 25 with 421 images.
Batch processing with labels completed!
```

In [ ]:

In [12]:
```
import numpy as np

data = np.load("./processed_batches_styles/batch_1.npz")
print(data.files)

images = data["arr_0"]
labels = data["labels"]
print(images.shape, labels.shape)
```

```
['arr_0', 'labels']
(1000, 224, 224, 3) (1000,)
```

In [14]:
```
img_size = (224, 224)
batch_size = 1000
output_folder = "processed_batches_artists"
wikiart_base = "./wikiart/"
csv_folder = "./wikiart_csv"

os.makedirs(output_folder, exist_ok=True)

csv_file = os.path.join(csv_folder, "artist_train.csv")

data = pd.read_csv(csv_file, header=None, names=["img_path", "class_index"])
image_paths = data["img_path"].tolist()
image_labels = data["class_index"].tolist()

def preprocess_image(img_path):
    full_path = os.path.join(wikiart_base, img_path)
    try:
        with Image.open(full_path) as img:
            img = img.convert("RGB")
            img_resized = img.resize(img_size)
            img_array = np.array(img_resized, dtype=np.float32) / 255.0
            return img_array
    except Exception as e:
        print(f"Failed to preprocess {full_path}: {e}")
        return None

def preprocess_in_batches(image_paths, image_labels, batch_size, output_folder):
    for batch_idx in tqdm(range(0, len(image_paths), batch_size), desc="Processi
        batch_paths = image_paths[batch_idx:batch_idx + batch_size]
        batch_labels = image_labels[batch_idx:batch_idx + batch_size]
        batch_images = []

        for img_path in batch_paths:
            img = preprocess_image(img_path)
            if img is not None:
                batch_images.append(img)

        if len(batch_images) == 0:
            print(f"Warning: No valid images in batch starting at index {batch_i
            continue

        batch_images_array = np.array(batch_images)

        batch_labels_array = np.array(batch_labels[:len(batch_images)], dtype=np
```

```
        batch_file = os.path.join(output_folder, f"batch_{batch_idx // batch_siz
        np.savez_compressed(batch_file, arr_0=batch_images_array, labels=batch_l
        print(f"Saved batch {batch_idx // batch_size + 1} with {len(batch_images

        # Free memory
        del batch_images, batch_images_array, batch_labels_array
        gc.collect()

preprocess_in_batches(image_paths, image_labels, batch_size, output_folder)

print("Batch processing with labels completed!")
```

```
Processing Batches:   7%|
| 1/14 [01:11<15:32, 71.72s/it]
Saved batch 1 with 1000 images.
Processing Batches:  14%|
| 2/14 [02:09<12:43, 63.60s/it]
Saved batch 2 with 1000 images.
Processing Batches:  21%|
| 3/14 [03:09<11:18, 61.71s/it]
Saved batch 3 with 1000 images.
Processing Batches:  29%|
| 4/14 [04:07<10:04, 60.40s/it]
Saved batch 4 with 1000 images.
Processing Batches:  36%|
| 5/14 [05:06<08:57, 59.74s/it]
Saved batch 5 with 1000 images.
Processing Batches:  43%|
| 6/14 [06:05<07:55, 59.47s/it]
Saved batch 6 with 1000 images.
Processing Batches:  50%|
| 7/14 [07:03<06:53, 59.08s/it]
Saved batch 7 with 1000 images.
Processing Batches:  57%|
| 8/14 [08:02<05:53, 58.98s/it]
Saved batch 8 with 1000 images.
Processing Batches:  64%|
| 9/14 [09:00<04:54, 58.82s/it]
Saved batch 9 with 1000 images.
Failed to preprocess ./wikiart/Post_Impressionism/vincent-van-gogh_l-arlesienne-p
ortrait-of-madame-ginoux-1890.jpg: [Errno 2] No such file or directory: 'C:\\User
s\\KIIT\\Desktop\\UJJU\\PROJECT\\GSOC(proposal)\\ArtExtract\\TASK-1\\wikiart\\Pos
t_Impressionism\\vincent-van-gogh_l-arlesienne-portrait-of-madame-ginoux-1890.jp
g'
Processing Batches:  71%|
| 10/14 [09:58<03:54, 58.67s/it]
Saved batch 10 with 999 images.
Processing Batches:  79%|
| 11/14 [10:57<02:55, 58.58s/it]
Saved batch 11 with 1000 images.
Processing Batches:  86%|
| 12/14 [11:59<01:59, 59.66s/it]
Saved batch 12 with 1000 images.
Processing Batches:  93%|
     | 13/14 [13:00<01:00, 60.19s/it]
```

```
Saved batch 13 with 1000 images.
Failed to preprocess ./wikiart/Baroque/rembrandt_woman-standing-with-raised-hand
s.jpg: [Errno 2] No such file or directory: 'C:\\Users\\KIIT\\Desktop\\UJJU\\PROJ
ECT\\GSOC(proposal)\\ArtExtract\\TASK-1\\wikiart\\Baroque\\rembrandt_woman-standi
ng-with-raised-hands.jpg'
Processing Batches: 100%|████████████████████████████████████████████████
██████| 14/14 [13:20<00:00, 57.19s/it]
Saved batch 14 with 345 images.
Batch processing with labels completed!
```

In [15]:

```python
img_size = (224, 224)
batch_size = 1000
output_folder = "processed_batches_artists_val"
wikiart_base = "./wikiart/"
csv_folder = "./wikiart_csv"

os.makedirs(output_folder, exist_ok=True)

csv_file = os.path.join(csv_folder, "artist_val.csv")

data = pd.read_csv(csv_file, header=None, names=["img_path", "class_index"])
image_paths = data["img_path"].tolist()
image_labels = data["class_index"].tolist()


def preprocess_image(img_path):

    full_path = os.path.join(wikiart_base, img_path)
    try:
        with Image.open(full_path) as img:
            img = img.convert("RGB")
            img_resized = img.resize(img_size)
            img_array = np.array(img_resized, dtype=np.float32) / 255.0  # Norma
            return img_array
    except Exception as e:
        print(f"Failed to preprocess {full_path}: {e}")
        return None

def preprocess_in_batches(image_paths, image_labels, batch_size, output_folder):
    for batch_idx in tqdm(range(0, len(image_paths), batch_size), desc="Processi
        batch_paths = image_paths[batch_idx:batch_idx + batch_size]
        batch_labels = image_labels[batch_idx:batch_idx + batch_size]
        batch_images = []

        for img_path in batch_paths:
            img = preprocess_image(img_path)
            if img is not None:
                batch_images.append(img)

        if len(batch_images) == 0:
            print(f"Warning: No valid images in batch starting at index {batch_i
            continue

        batch_images_array = np.array(batch_images)

        batch_labels_array = np.array(batch_labels[:len(batch_images)], dtype=np

        batch_file = os.path.join(output_folder, f"batch_{batch_idx // batch_siz
        np.savez_compressed(batch_file, arr_0=batch_images_array, labels=batch_l
```

```
        print(f"Saved batch {batch_idx // batch_size + 1} with {len(batch_images

        # Free memory
        del batch_images, batch_images_array, batch_labels_array
        gc.collect()

preprocess_in_batches(image_paths, image_labels, batch_size, output_folder)

print("Batch processing with labels completed!")
```

```
Processing Batches:  17%|
| 1/6 [01:06<05:31, 66.32s/it]
Saved batch 1 with 1000 images.
Processing Batches:  33%|
| 2/6 [02:03<04:04, 61.16s/it]
Saved batch 2 with 1000 images.
Processing Batches:  50%|
| 3/6 [02:53<02:47, 55.94s/it]
Saved batch 3 with 1000 images.
Processing Batches:  67%|
| 4/6 [03:51<01:53, 56.77s/it]
Saved batch 4 with 1000 images.
Processing Batches:  83%|
| 5/6 [04:48<00:56, 56.87s/it]
Saved batch 5 with 1000 images.
Processing Batches: 100%|
| 6/6 [05:30<00:00, 55.15s/it]
Saved batch 6 with 706 images.
Batch processing with labels completed!
```

In [16]:
```
import numpy as np

data = np.load("./processed_batches_artists/batch_1.npz")
print(data.files

images = data["arr_0"]
labels = data["labels"]
print(images.shape, labels.shape)
```

```
['arr_0', 'labels']
(1000, 224, 224, 3) (1000,)
```

In [17]:
```
img_size = (224, 224)
batch_size = 1000
output_folder = "processed_batches_genres"
wikiart_base = "./wikiart/"
csv_folder = "./wikiart_csv"

os.makedirs(output_folder, exist_ok=True)

csv_file = os.path.join(csv_folder, "genre_train.csv")
data = pd.read_csv(csv_file, header=None, names=["img_path", "class_index"])
image_paths = data["img_path"].tolist()
image_labels = data["class_index"].tolist()

def preprocess_image(img_path):
    """Load an image, convert to RGB, resize, and normalize."""
    full_path = os.path.join(wikiart_base, img_path)
    try:
```

```
        with Image.open(full_path) as img:
            img = img.convert("RGB")
            img_resized = img.resize(img_size)
            img_array = np.array(img_resized, dtype=np.float32) / 255.0  # Norma
            return img_array
    except Exception as e:
        print(f"Failed to preprocess {full_path}: {e}")
        return None

def preprocess_in_batches(image_paths, image_labels, batch_size, output_folder):
    for batch_idx in tqdm(range(0, len(image_paths), batch_size), desc="Processi
        batch_paths = image_paths[batch_idx:batch_idx + batch_size]
        batch_labels = image_labels[batch_idx:batch_idx + batch_size]
        batch_images = []

        for img_path in batch_paths:
            img = preprocess_image(img_path)
            if img is not None:
                batch_images.append(img)

        if len(batch_images) == 0:
            print(f"Warning: No valid images in batch starting at index {batch_i
            continue

        batch_images_array = np.array(batch_images)

        batch_labels_array = np.array(batch_labels[:len(batch_images)], dtype=np

        batch_file = os.path.join(output_folder, f"batch_{batch_idx // batch_siz
        np.savez_compressed(batch_file, arr_0=batch_images_array, labels=batch_l
        print(f"Saved batch {batch_idx // batch_size + 1} with {len(batch_images

        # Free memory
        del batch_images, batch_images_array, batch_labels_array
        gc.collect()

preprocess_in_batches(image_paths, image_labels, batch_size, output_folder)

print("Batch processing with labels completed!")
```

```
Processing Batches:   2%|▉
| 1/46 [00:59<44:49, 59.77s/it]
Saved batch 1 with 1000 images.
Processing Batches:   4%|██
| 2/46 [01:59<43:39, 59.54s/it]
Saved batch 2 with 1000 images.
Processing Batches:   7%|███
| 3/46 [02:57<42:16, 59.00s/it]
Saved batch 3 with 1000 images.
Failed to preprocess ./wikiart/Baroque/rembrandt_woman-standing-with-raised-hand
s.jpg: [Errno 2] No such file or directory: 'C:\\Users\\KIIT\\Desktop\\UJJU\\PROJ
ECT\\GSOC(proposal)\\ArtExtract\\TASK-1\\wikiart\\Baroque\\rembrandt_woman-standi
ng-with-raised-hands.jpg'
Processing Batches:   9%|████
| 4/46 [03:57<41:38, 59.49s/it]
Saved batch 4 with 999 images.
Processing Batches:  11%|████▍
| 5/46 [04:57<40:41, 59.55s/it]
Saved batch 5 with 1000 images.
```

```
Processing Batches:  13%|████████
| 6/46 [05:57<39:51, 59.78s/it]
Saved batch 6 with 1000 images.
Processing Batches:  15%|█████████
| 7/46 [06:56<38:37, 59.42s/it]
Saved batch 7 with 1000 images.
Processing Batches:  17%|██████████
| 8/46 [07:54<37:27, 59.14s/it]
Saved batch 8 with 1000 images.
Processing Batches:  20%|███████████
| 9/46 [08:53<36:20, 58.94s/it]
Saved batch 9 with 1000 images.
Processing Batches:  22%|████████████
| 10/46 [09:53<35:30, 59.18s/it]
Saved batch 10 with 1000 images.
Processing Batches:  24%|█████████████
| 11/46 [10:52<34:36, 59.32s/it]
Saved batch 11 with 1000 images.
Processing Batches:  26%|██████████████
| 12/46 [11:51<33:28, 59.06s/it]
Saved batch 12 with 1000 images.
Processing Batches:  28%|███████████████
| 13/46 [12:50<32:35, 59.26s/it]
Saved batch 13 with 1000 images.
Processing Batches:  30%|████████████████
| 14/46 [13:50<31:36, 59.25s/it]
Saved batch 14 with 1000 images.
Processing Batches:  33%|██████████████████
| 15/46 [14:49<30:35, 59.20s/it]
Saved batch 15 with 1000 images.
Processing Batches:  35%|███████████████████
| 16/46 [15:48<29:39, 59.32s/it]
Saved batch 16 with 1000 images.
Saved batch 17 with 1000 images.
Processing Batches:  37%|████████████████████
| 17/46 [17:02<30:47, 63.72s/it]
Saved batch 18 with 1000 images.
Processing Batches:  41%|██████████████████████
| 19/46 [19:44<32:32, 72.33s/it]
Saved batch 19 with 1000 images.
Processing Batches:  43%|███████████████████████
| 20/46 [21:04<32:23, 74.77s/it]
Saved batch 20 with 1000 images.
Processing Batches:  46%|████████████████████████
| 21/46 [22:23<31:40, 76.02s/it]
Saved batch 21 with 1000 images.
Processing Batches:  48%|█████████████████████████
| 22/46 [23:43<30:49, 77.06s/it]
Saved batch 22 with 1000 images.
Processing Batches:  50%|██████████████████████████
| 23/46 [25:05<30:11, 78.77s/it]
Saved batch 23 with 1000 images.
Processing Batches:  52%|███████████████████████████
| 24/46 [26:27<29:11, 79.63s/it]
Saved batch 24 with 1000 images.
```

```
C:\Users\KIIT\AppData\Roaming\Python\Python312\site-packages\PIL\Image.py:3368: D
ecompressionBombWarning: Image size (99962094 pixels) exceeds limit of 89478485 p
ixels, could be decompression bomb DOS attack.
  warnings.warn(
Processing Batches:  54%|
| 25/46 [27:49<28:07, 80.34s/it]
Saved batch 25 with 1000 images.
Processing Batches:  57%|
| 26/46 [29:10<26:53, 80.65s/it]
Saved batch 26 with 1000 images.
Processing Batches:  59%|
| 27/46 [30:32<25:40, 81.07s/it]
Saved batch 27 with 1000 images.
Processing Batches:  61%|
| 28/46 [31:56<24:30, 81.68s/it]
Saved batch 28 with 1000 images.
Processing Batches:  63%|
| 29/46 [33:17<23:09, 81.74s/it]
Saved batch 29 with 1000 images.
Processing Batches:  65%|
| 30/46 [34:39<21:45, 81.59s/it]
Saved batch 30 with 1000 images.
Processing Batches:  67%|
| 31/46 [36:01<20:25, 81.68s/it]
Saved batch 31 with 1000 images.
Processing Batches:  70%|
| 32/46 [37:21<18:59, 81.40s/it]
Saved batch 32 with 1000 images.
Processing Batches:  72%|
| 33/46 [38:45<17:49, 82.24s/it]
Saved batch 33 with 1000 images.
Processing Batches:  74%|
| 34/46 [40:07<16:23, 81.96s/it]
Saved batch 34 with 1000 images.
Processing Batches:  76%|
| 35/46 [41:30<15:05, 82.33s/it]
Saved batch 35 with 1000 images.
C:\Users\KIIT\AppData\Roaming\Python\Python312\site-packages\PIL\Image.py:3368: D
ecompressionBombWarning: Image size (107327830 pixels) exceeds limit of 89478485
pixels, could be decompression bomb DOS attack.
  warnings.warn(
Saved batch 36 with 1000 images.
Processing Batches:  80%|
| 37/46 [44:11<12:13, 81.49s/it]
Saved batch 37 with 1000 images.
Processing Batches:  83%|
| 38/46 [45:36<10:58, 82.30s/it]
Saved batch 38 with 1000 images.
Saved batch 39 with 1000 images.
Processing Batches:  87%|
| 40/46 [48:20<08:13, 82.24s/it]
Saved batch 40 with 1000 images.
Processing Batches:  89%|
▌       | 41/46 [49:45<06:55, 83.14s/it]
Saved batch 41 with 1000 images.
```

```
Processing Batches:  91%|████████████████████████████████████████████
▌     | 42/46 [51:08<05:32, 83.02s/it]
Saved batch 42 with 1000 images.
Processing Batches:  93%|████████████████████████████████████████████
██    | 43/46 [52:28<04:06, 82.13s/it]
Saved batch 43 with 1000 images.
Processing Batches:  96%|████████████████████████████████████████████
███   | 44/46 [53:48<02:42, 81.35s/it]
Saved batch 44 with 1000 images.
Processing Batches:  98%|████████████████████████████████████████████
████▌ | 45/46 [55:07<01:20, 80.88s/it]
Saved batch 45 with 1000 images.
Failed to preprocess ./wikiart/Post_Impressionism/vincent-van-gogh_l-arlesienne-p
ortrait-of-madame-ginoux-1890.jpg: [Errno 2] No such file or directory: 'C:\\User
s\\KIIT\\Desktop\\UJJU\\PROJECT\\GSOC(proposal)\\ArtExtract\\TASK-1\\wikiart\\Pos
t_Impressionism\\vincent-van-gogh_l-arlesienne-portrait-of-madame-ginoux-1890.jp
g'
Processing Batches: 100%|████████████████████████████████████████████
██████| 46/46 [55:49<00:00, 72.81s/it]
Saved batch 46 with 502 images.
Batch processing with labels completed!
```

In [18]:
```python
img_size = (224, 224)
batch_size = 1000
output_folder = "processed_batches_genres_val"
wikiart_base = "./wikiart/"
csv_folder = "./wikiart_csv"

os.makedirs(output_folder, exist_ok=True)

csv_file = os.path.join(csv_folder, "genre_val.csv")
data = pd.read_csv(csv_file, header=None, names=["img_path", "class_index"])
image_paths = data["img_path"].tolist()
image_labels = data["class_index"].tolist()

def preprocess_image(img_path):
    """Load an image, convert to RGB, resize, and normalize."""
    full_path = os.path.join(wikiart_base, img_path)
    try:
        with Image.open(full_path) as img:
            img = img.convert("RGB")
            img_resized = img.resize(img_size)
            img_array = np.array(img_resized, dtype=np.float32) / 255.0  # Norma
            return img_array
    except Exception as e:
        print(f"Failed to preprocess {full_path}: {e}")
        return None

def preprocess_in_batches(image_paths, image_labels, batch_size, output_folder):
    for batch_idx in tqdm(range(0, len(image_paths), batch_size), desc="Processi
        batch_paths = image_paths[batch_idx:batch_idx + batch_size]
        batch_labels = image_labels[batch_idx:batch_idx + batch_size]
        batch_images = []

        for img_path in batch_paths:
            img = preprocess_image(img_path)
            if img is not None:
                batch_images.append(img)
```

```
        if len(batch_images) == 0:
            print(f"Warning: No valid images in batch starting at index {batch_i
            continue

        batch_images_array = np.array(batch_images)
        batch_labels_array = np.array(batch_labels[:len(batch_images)], dtype=np

        batch_file = os.path.join(output_folder, f"batch_{batch_idx // batch_siz
        np.savez_compressed(batch_file, arr_0=batch_images_array, labels=batch_l
        print(f"Saved batch {batch_idx // batch_size + 1} with {len(batch_images

        # Free memory
        del batch_images, batch_images_array, batch_labels_array
        gc.collect()

preprocess_in_batches(image_paths, image_labels, batch_size, output_folder)

print("Batch processing with labels completed!")
```

```
Processing Batches:   5%|███
| 1/20 [01:26<27:27, 86.71s/it]
Saved batch 1 with 1000 images.
Processing Batches:  10%|██████
| 2/20 [02:52<25:54, 86.35s/it]
Saved batch 2 with 1000 images.
Processing Batches:  15%|█████████
| 3/20 [04:19<24:32, 86.63s/it]
Saved batch 3 with 1000 images.
Processing Batches:  20%|████████████
| 4/20 [05:47<23:11, 86.98s/it]
Saved batch 4 with 1000 images.
Processing Batches:  25%|███████████████
| 5/20 [07:13<21:42, 86.81s/it]
Saved batch 5 with 1000 images.
Saved batch 6 with 1000 images.
Processing Batches:  35%|█████████████████████
| 7/20 [10:01<18:27, 85.21s/it]
Saved batch 7 with 1000 images.
Processing Batches:  40%|████████████████████████
| 8/20 [11:26<17:01, 85.10s/it]
Saved batch 8 with 1000 images.
Processing Batches:  45%|███████████████████████████
| 9/20 [12:53<15:43, 85.75s/it]
Saved batch 9 with 1000 images.
Processing Batches:  50%|██████████████████████████████
| 10/20 [14:18<14:15, 85.54s/it]
Saved batch 10 with 1000 images.
Processing Batches:  55%|█████████████████████████████████
| 11/20 [15:43<12:48, 85.43s/it]
Saved batch 11 with 1000 images.
Processing Batches:  60%|████████████████████████████████████
| 12/20 [17:05<11:15, 84.38s/it]
Saved batch 12 with 1000 images.
Processing Batches:  65%|███████████████████████████████████████
| 13/20 [18:13<09:15, 79.30s/it]
Saved batch 13 with 1000 images.
```

```
Processing Batches:  70%|███████████████████████████████████▌
| 14/20 [19:21<07:36, 76.09s/it]
Saved batch 14 with 1000 images.
Processing Batches:  75%|██████████████████████████████████████
| 15/20 [20:41<06:26, 77.32s/it]
Saved batch 15 with 1000 images.
Processing Batches:  80%|████████████████████████████████████████
| 16/20 [22:02<05:13, 78.46s/it]
Saved batch 16 with 1000 images.
Processing Batches:  85%|███████████████████████████████████████████
| 17/20 [23:22<03:56, 78.80s/it]
Saved batch 17 with 1000 images.
Processing Batches:  90%|████████████████████████████████████████████
█       | 18/20 [24:42<02:38, 79.08s/it]
Saved batch 18 with 1000 images.
Processing Batches:  95%|████████████████████████████████████████████
███    | 19/20 [25:41<01:13, 73.23s/it]
Saved batch 19 with 1000 images.
Processing Batches: 100%|████████████████████████████████████████████
██████| 20/20 [26:11<00:00, 78.57s/it]
Saved batch 20 with 492 images.
Batch processing with labels completed!
```

In [19]:
```
import numpy as np

data = np.load("./processed_batches_genres/batch_1.npz")
print(data.files)

images = data["arr_0"]
labels = data["labels"]
print(images.shape, labels.shape)
```

```
['arr_0', 'labels']
(1000, 224, 224, 3) (1000,)
```

training

In [2]:
```
import os
import glob
import numpy as np
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import Input, GlobalAveragePooling2D, BatchNormaliz
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau
from tqdm.keras import TqdmCallback
```

In [1]:
```
import os
import glob
import numpy as np
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import Input, Reshape, Bidirectional, LSTM, Dense,
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import ModelCheckpoint
from tqdm.keras import TqdmCallback
```

```
TRAIN_FOLDER = "processed_batches_styles"
VAL_FOLDER = "processed_batches_styles_val"
IMG_SHAPE = (224, 224, 3)
NUM_CLASSES = 27
BATCH_SIZE = 1000
NEW_BATCH_SIZE = 100
EPOCHS = 5

SAVE_DIR = "saved_models"
os.makedirs(SAVE_DIR, exist_ok=True)
BEST_MODEL_PATH = os.path.join(SAVE_DIR, "best_style_model.keras")
FINAL_MODEL_PATH = os.path.join(SAVE_DIR, "final_style_model.keras")


def npz_generator(folder):
    """
    Generator that yields (images, labels) for each npz file in the folder
    If the npz file has fewer than the expected batch size images, it yields the
    """
    files = sorted(glob.glob(os.path.join(folder, "*.npz")))
    for f in files:
        data = np.load(f)
        imgs = data['arr_0']    # Shape: (N, 224, 224, 3)
        labels = data['labels'] # Shape: (N,)
        current_batch_size = imgs.shape[0]
        if current_batch_size != BATCH_SIZE:
            print(f"Warning: File {f} has only {current_batch_size} images (expe
        yield imgs, labels

# each yielded item has shape (N, 224, 224, 3) with n roughly batch size.
train_dataset = tf.data.Dataset.from_generator(
    lambda: npz_generator(TRAIN_FOLDER),
    output_types=(tf.float32, tf.int32),
    output_shapes=((None, IMG_SHAPE[0], IMG_SHAPE[1], IMG_SHAPE[2]), (None,))
)

val_dataset = tf.data.Dataset.from_generator(
    lambda: npz_generator(VAL_FOLDER),
    output_types=(tf.float32, tf.int32),
    output_shapes=((None, IMG_SHAPE[0], IMG_SHAPE[1], IMG_SHAPE[2]), (None,))
)

# Shuffle the training dataset at the "big batch" level
train_dataset = train_dataset.shuffle(100)

# Now "unbatch" these big batches into individual samples and then re-batch them
train_dataset = train_dataset.unbatch().batch(NEW_BATCH_SIZE)
val_dataset = val_dataset.unbatch().batch(NEW_BATCH_SIZE)

# Calculate steps per epoch: originally, each npz file had ~BATCH_SIZE images.
# Total images in train = number of npz files * BATCH_SIZE (approx.).
num_train_files = len(glob.glob(os.path.join(TRAIN_FOLDER, "*.npz")))
num_val_files   = len(glob.glob(os.path.join(VAL_FOLDER, "*.npz")))

steps_per_epoch = (num_train_files * BATCH_SIZE) // NEW_BATCH_SIZE
validation_steps = (num_val_files * BATCH_SIZE) // NEW_BATCH_SIZE


inputs = Input(shape=IMG_SHAPE)
```

```
# Pre-trained ResNet50 (without top) for feature extraction
base_model = ResNet50(weights='imagenet', include_top=False, input_tensor=inputs
x = base_model.output                          # Typically (7,7,2048) for a 224x2
x = Reshape((7 * 7, x.shape[-1]))(x)             # Reshape the output into a seq
x = Bidirectional(LSTM(256, return_sequences=False))(x)          # Add a Bidirec
x = Dropout(0.5)(x)

outputs = Dense(NUM_CLASSES, activation='softmax')(x)              # Final Dense
model = Model(inputs=inputs, outputs=outputs)
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.summary()

checkpoint_cb = ModelCheckpoint(
    BEST_MODEL_PATH,
    monitor='val_accuracy',
    save_best_only=True,
    mode='max',
    verbose=1
)

history = model.fit(
    train_dataset,
    steps_per_epoch=steps_per_epoch,
    epochs=EPOCHS,
    validation_data=val_dataset,
    validation_steps=validation_steps,
    callbacks=[checkpoint_cb, TqdmCallback(verbose=1)]
)

model.save(FINAL_MODEL_PATH)
print(f"Final model saved to: {FINAL_MODEL_PATH}")
```

```
WARNING:tensorflow:From C:\Users\KIIT\AppData\Local\Temp\ipykernel_23452\36794001
78.py:49: calling DatasetV2.from_generator (from tensorflow.python.data.ops.datas
et_ops) with output_types is deprecated and will be removed in a future version.
Instructions for updating:
Use output_signature instead
WARNING:tensorflow:From C:\Users\KIIT\AppData\Local\Temp\ipykernel_23452\36794001
78.py:49: calling DatasetV2.from_generator (from tensorflow.python.data.ops.datas
et_ops) with output_shapes is deprecated and will be removed in a future version.
Instructions for updating:
Use output_signature instead
Model: "functional"
```

Layer (type)	Output Shape	Para
input_layer (InputLayer)	(None, 224, 224, 3)	
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	
conv1_conv (Conv2D)	(None, 112, 112, 64)	9,
conv1_bn (BatchNormalization)	(None, 112, 112, 64)	
conv1_relu (Activation)	(None, 112, 112, 64)	
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4,
conv2_block1_1_bn (BatchNormalization)	(None, 56, 56, 64)	
conv2_block1_1_relu (Activation)	(None, 56, 56, 64)	
conv2_block1_2_conv (Conv2D)	(None, 56, 56, 64)	36,
conv2_block1_2_bn (BatchNormalization)	(None, 56, 56, 64)	
conv2_block1_2_relu (Activation)	(None, 56, 56, 64)	
conv2_block1_0_conv (Conv2D)	(None, 56, 56, 256)	16,
conv2_block1_3_conv (Conv2D)	(None, 56, 56, 256)	16,
conv2_block1_0_bn (BatchNormalization)	(None, 56, 56, 256)	1,
conv2_block1_3_bn (BatchNormalization)	(None, 56, 56, 256)	1,
conv2_block1_add (Add)	(None, 56, 56, 256)	
conv2_block1_out (Activation)	(None, 56, 56, 256)	
conv2_block2_1_conv (Conv2D)	(None, 56, 56, 64)	16,
conv2_block2_1_bn (BatchNormalization)	(None, 56, 56, 64)	
conv2_block2_1_relu (Activation)	(None, 56, 56, 64)	
conv2_block2_2_conv (Conv2D)	(None, 56, 56, 64)	36,
conv2_block2_2_bn (BatchNormalization)	(None, 56, 56, 64)	

conv2_block2_2_relu (Activation)	(None, 56, 56, 64)	
conv2_block2_3_conv (Conv2D)	(None, 56, 56, 256)	16,
conv2_block2_3_bn (BatchNormalization)	(None, 56, 56, 256)	1,
conv2_block2_add (Add)	(None, 56, 56, 256)	
conv2_block2_out (Activation)	(None, 56, 56, 256)	
conv2_block3_1_conv (Conv2D)	(None, 56, 56, 64)	16,
conv2_block3_1_bn (BatchNormalization)	(None, 56, 56, 64)	
conv2_block3_1_relu (Activation)	(None, 56, 56, 64)	
conv2_block3_2_conv (Conv2D)	(None, 56, 56, 64)	36,
conv2_block3_2_bn (BatchNormalization)	(None, 56, 56, 64)	
conv2_block3_2_relu (Activation)	(None, 56, 56, 64)	
conv2_block3_3_conv (Conv2D)	(None, 56, 56, 256)	16,
conv2_block3_3_bn (BatchNormalization)	(None, 56, 56, 256)	1,
conv2_block3_add (Add)	(None, 56, 56, 256)	
conv2_block3_out (Activation)	(None, 56, 56, 256)	
conv3_block1_1_conv (Conv2D)	(None, 28, 28, 128)	32,
conv3_block1_1_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block1_1_relu (Activation)	(None, 28, 28, 128)	
conv3_block1_2_conv (Conv2D)	(None, 28, 28, 128)	147,
conv3_block1_2_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block1_2_relu (Activation)	(None, 28, 28, 128)	
conv3_block1_0_conv (Conv2D)	(None, 28, 28, 512)	131,
conv3_block1_3_conv (Conv2D)	(None, 28, 28, 512)	66,

conv3_block1_0_bn (BatchNormalization)	(None, 28, 28, 512)	2,
conv3_block1_3_bn (BatchNormalization)	(None, 28, 28, 512)	2,
conv3_block1_add (Add)	(None, 28, 28, 512)	
conv3_block1_out (Activation)	(None, 28, 28, 512)	
conv3_block2_1_conv (Conv2D)	(None, 28, 28, 128)	65,
conv3_block2_1_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block2_1_relu (Activation)	(None, 28, 28, 128)	
conv3_block2_2_conv (Conv2D)	(None, 28, 28, 128)	147,
conv3_block2_2_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block2_2_relu (Activation)	(None, 28, 28, 128)	
conv3_block2_3_conv (Conv2D)	(None, 28, 28, 512)	66,
conv3_block2_3_bn (BatchNormalization)	(None, 28, 28, 512)	2,
conv3_block2_add (Add)	(None, 28, 28, 512)	
conv3_block2_out (Activation)	(None, 28, 28, 512)	
conv3_block3_1_conv (Conv2D)	(None, 28, 28, 128)	65,
conv3_block3_1_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block3_1_relu (Activation)	(None, 28, 28, 128)	
conv3_block3_2_conv (Conv2D)	(None, 28, 28, 128)	147,
conv3_block3_2_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block3_2_relu (Activation)	(None, 28, 28, 128)	
conv3_block3_3_conv (Conv2D)	(None, 28, 28, 512)	66,
conv3_block3_3_bn (BatchNormalization)	(None, 28, 28, 512)	2,
conv3_block3_add (Add)	(None, 28, 28, 512)	

conv3_block3_out (Activation)	(None, 28, 28, 512)	
conv3_block4_1_conv (Conv2D)	(None, 28, 28, 128)	65,
conv3_block4_1_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block4_1_relu (Activation)	(None, 28, 28, 128)	
conv3_block4_2_conv (Conv2D)	(None, 28, 28, 128)	147,
conv3_block4_2_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block4_2_relu (Activation)	(None, 28, 28, 128)	
conv3_block4_3_conv (Conv2D)	(None, 28, 28, 512)	66,
conv3_block4_3_bn (BatchNormalization)	(None, 28, 28, 512)	2,
conv3_block4_add (Add)	(None, 28, 28, 512)	
conv3_block4_out (Activation)	(None, 28, 28, 512)	
conv4_block1_1_conv (Conv2D)	(None, 14, 14, 256)	131,
conv4_block1_1_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block1_1_relu (Activation)	(None, 14, 14, 256)	
conv4_block1_2_conv (Conv2D)	(None, 14, 14, 256)	590,
conv4_block1_2_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block1_2_relu (Activation)	(None, 14, 14, 256)	
conv4_block1_0_conv (Conv2D)	(None, 14, 14, 1024)	525,
conv4_block1_3_conv (Conv2D)	(None, 14, 14, 1024)	263,
conv4_block1_0_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block1_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block1_add (Add)	(None, 14, 14, 1024)	
conv4_block1_out (Activation)	(None, 14, 14, 1024)	

conv4_block2_1_conv (Conv2D)	(None, 14, 14, 256)	262,
conv4_block2_1_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block2_1_relu (Activation)	(None, 14, 14, 256)	
conv4_block2_2_conv (Conv2D)	(None, 14, 14, 256)	590,
conv4_block2_2_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block2_2_relu (Activation)	(None, 14, 14, 256)	
conv4_block2_3_conv (Conv2D)	(None, 14, 14, 1024)	263,
conv4_block2_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block2_add (Add)	(None, 14, 14, 1024)	
conv4_block2_out (Activation)	(None, 14, 14, 1024)	
conv4_block3_1_conv (Conv2D)	(None, 14, 14, 256)	262,
conv4_block3_1_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block3_1_relu (Activation)	(None, 14, 14, 256)	
conv4_block3_2_conv (Conv2D)	(None, 14, 14, 256)	590,
conv4_block3_2_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block3_2_relu (Activation)	(None, 14, 14, 256)	
conv4_block3_3_conv (Conv2D)	(None, 14, 14, 1024)	263,
conv4_block3_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block3_add (Add)	(None, 14, 14, 1024)	
conv4_block3_out (Activation)	(None, 14, 14, 1024)	
conv4_block4_1_conv (Conv2D)	(None, 14, 14, 256)	262,
conv4_block4_1_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block4_1_relu (Activation)	(None, 14, 14, 256)	

conv4_block4_2_conv (Conv2D)	(None, 14, 14, 256)	590,
conv4_block4_2_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block4_2_relu (Activation)	(None, 14, 14, 256)	
conv4_block4_3_conv (Conv2D)	(None, 14, 14, 1024)	263,
conv4_block4_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block4_add (Add)	(None, 14, 14, 1024)	
conv4_block4_out (Activation)	(None, 14, 14, 1024)	
conv4_block5_1_conv (Conv2D)	(None, 14, 14, 256)	262,
conv4_block5_1_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block5_1_relu (Activation)	(None, 14, 14, 256)	
conv4_block5_2_conv (Conv2D)	(None, 14, 14, 256)	590,
conv4_block5_2_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block5_2_relu (Activation)	(None, 14, 14, 256)	
conv4_block5_3_conv (Conv2D)	(None, 14, 14, 1024)	263,
conv4_block5_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block5_add (Add)	(None, 14, 14, 1024)	
conv4_block5_out (Activation)	(None, 14, 14, 1024)	
conv4_block6_1_conv (Conv2D)	(None, 14, 14, 256)	262,
conv4_block6_1_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block6_1_relu (Activation)	(None, 14, 14, 256)	
conv4_block6_2_conv (Conv2D)	(None, 14, 14, 256)	590,
conv4_block6_2_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block6_2_relu (Activation)	(None, 14, 14, 256)	

conv4_block6_3_conv (Conv2D)	(None, 14, 14, 1024)	263,
conv4_block6_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block6_add (Add)	(None, 14, 14, 1024)	
conv4_block6_out (Activation)	(None, 14, 14, 1024)	
conv5_block1_1_conv (Conv2D)	(None, 7, 7, 512)	524,
conv5_block1_1_bn (BatchNormalization)	(None, 7, 7, 512)	2,
conv5_block1_1_relu (Activation)	(None, 7, 7, 512)	
conv5_block1_2_conv (Conv2D)	(None, 7, 7, 512)	2,359,
conv5_block1_2_bn (BatchNormalization)	(None, 7, 7, 512)	2,
conv5_block1_2_relu (Activation)	(None, 7, 7, 512)	
conv5_block1_0_conv (Conv2D)	(None, 7, 7, 2048)	2,099,
conv5_block1_3_conv (Conv2D)	(None, 7, 7, 2048)	1,050,
conv5_block1_0_bn (BatchNormalization)	(None, 7, 7, 2048)	8,
conv5_block1_3_bn (BatchNormalization)	(None, 7, 7, 2048)	8,
conv5_block1_add (Add)	(None, 7, 7, 2048)	
conv5_block1_out (Activation)	(None, 7, 7, 2048)	
conv5_block2_1_conv (Conv2D)	(None, 7, 7, 512)	1,049,
conv5_block2_1_bn (BatchNormalization)	(None, 7, 7, 512)	2,
conv5_block2_1_relu (Activation)	(None, 7, 7, 512)	
conv5_block2_2_conv (Conv2D)	(None, 7, 7, 512)	2,359,
conv5_block2_2_bn (BatchNormalization)	(None, 7, 7, 512)	2,
conv5_block2_2_relu (Activation)	(None, 7, 7, 512)	
conv5_block2_3_conv (Conv2D)	(None, 7, 7, 2048)	1,050,
conv5_block2_3_bn	(None, 7, 7, 2048)	8,

(BatchNormalization)		
conv5_block2_add (Add)	(None, 7, 7, 2048)	
conv5_block2_out (Activation)	(None, 7, 7, 2048)	
conv5_block3_1_conv (Conv2D)	(None, 7, 7, 512)	1,049,
conv5_block3_1_bn (BatchNormalization)	(None, 7, 7, 512)	2,
conv5_block3_1_relu (Activation)	(None, 7, 7, 512)	
conv5_block3_2_conv (Conv2D)	(None, 7, 7, 512)	2,359,
conv5_block3_2_bn (BatchNormalization)	(None, 7, 7, 512)	2,
conv5_block3_2_relu (Activation)	(None, 7, 7, 512)	
conv5_block3_3_conv (Conv2D)	(None, 7, 7, 2048)	1,050,
conv5_block3_3_bn (BatchNormalization)	(None, 7, 7, 2048)	8,
conv5_block3_add (Add)	(None, 7, 7, 2048)	
conv5_block3_out (Activation)	(None, 7, 7, 2048)	
reshape (Reshape)	(None, 49, 2048)	
bidirectional (Bidirectional)	(None, 512)	4,720,
dropout (Dropout)	(None, 512)	
dense (Dense)	(None, 27)	13,

Total params: 28,322,203 (108.04 MB)

Trainable params: 28,269,083 (107.84 MB)

Non-trainable params: 53,120 (207.50 KB)

```
0epoch [00:00, ?epoch/s]
0batch [00:00, ?batch/s]
Epoch 1/5
Warning: File processed_batches_styles\batch_38.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles\batch_44.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles\batch_57.npz has only 25 images (expected
1000). Yielding partial batch.
571/580 ━━━━━━━━━━━━━━━━━━━ 1:43 11s/step - accuracy: 0.2685 - loss: 2.3685
```

```
C:\Users\KIIT\AppData\Roaming\Python\Python312\site-packages\keras\src\trainers\epoch_iterator.py:151: UserWarning: Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batches. You may need to use the `.repeat()` function when building your dataset.
  self._interrupted_warning()
```

```
Warning: File processed_batches_styles_val\batch_24.npz has only 421 images (expe
cted 1000). Yielding partial batch.

Epoch 1: val_accuracy improved from -inf to 0.02064, saving model to saved_models
\best_style_model.keras
580/580 ━━━━━━━━━━━━━━━━━━━━ 7402s 12s/step - accuracy: 0.2692 - loss: 2.3659 - v
al_accuracy: 0.0206 - val_loss: 6.2398
Epoch 2/5
Warning: File processed_batches_styles\batch_38.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles\batch_44.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles\batch_57.npz has only 25 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles_val\batch_24.npz has only 421 images (expe
cted 1000). Yielding partial batch.

Epoch 2: val_accuracy improved from 0.02064 to 0.35285, saving model to saved_mod
els\best_style_model.keras
580/580 ━━━━━━━━━━━━━━━━━━━━ 7402s 12s/step - accuracy: 0.3796 - loss: 1.9335 - v
al_accuracy: 0.3529 - val_loss: 2.0101
Epoch 3/5
Warning: File processed_batches_styles\batch_38.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles\batch_44.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles\batch_57.npz has only 25 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles_val\batch_24.npz has only 421 images (expe
cted 1000). Yielding partial batch.

Epoch 3: val_accuracy did not improve from 0.35285
580/580 ━━━━━━━━━━━━━━━━━━━━ 7377s 12s/step - accuracy: 0.4350 - loss: 1.7310 - v
al_accuracy: 0.2746 - val_loss: 2.4080
Epoch 4/5
Warning: File processed_batches_styles\batch_38.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles\batch_44.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles\batch_57.npz has only 25 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles_val\batch_24.npz has only 421 images (expe
cted 1000). Yielding partial batch.

Epoch 4: val_accuracy did not improve from 0.35285
580/580 ━━━━━━━━━━━━━━━━━━━━ 7383s 12s/step - accuracy: 0.4662 - loss: 1.6449 - v
al_accuracy: 0.3075 - val_loss: 2.2563
Epoch 5/5
Warning: File processed_batches_styles\batch_38.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles\batch_44.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles\batch_57.npz has only 25 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles_val\batch_24.npz has only 421 images (expe
cted 1000). Yielding partial batch.

Epoch 5: val_accuracy did not improve from 0.35285
580/580 ━━━━━━━━━━━━━━━━━━━━ 7334s 12s/step - accuracy: 0.4823 - loss: 1.6209 - v
```

```
al_accuracy: 0.3492 - val_loss: 2.1665
Final model saved to: saved_models\final_style_model.keras
```

In [ ]:

In [2]:
```python
import os
import glob
import numpy as np
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import Input, Reshape, Bidirectional, LSTM, Dense,
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import ModelCheckpoint
from tqdm.keras import TqdmCallback

TRAIN_FOLDER = "processed_batches_artists"
VAL_FOLDER = "processed_batches_artists_val"
IMG_SHAPE = (224, 224, 3)
NUM_CLASSES = 23
BATCH_SIZE = 1000
NEW_BATCH_SIZE = 100
EPOCHS = 5

SAVE_DIR = "saved_models"
os.makedirs(SAVE_DIR, exist_ok=True)
BEST_MODEL_PATH = os.path.join(SAVE_DIR, "best_artist_model.keras")
FINAL_MODEL_PATH = os.path.join(SAVE_DIR, "final_artist_model.keras")


def npz_generator(folder):
    """
    Generator that yields (images, labels) for each npz file in the folder.
    If the npz file has fewer than the expected BATCH_SIZE images, it yields the
    """
    files = sorted(glob.glob(os.path.join(folder, "*.npz")))
    for f in files:
        data = np.load(f)
        imgs = data['arr_0']
        labels = data['labels']
        current_batch_size = imgs.shape[0]
        if current_batch_size != BATCH_SIZE:
            print(f"Warning: File {f} has only {current_batch_size} images (expe
        yield imgs, labels


train_dataset = tf.data.Dataset.from_generator(
    lambda: npz_generator(TRAIN_FOLDER),
    output_types=(tf.float32, tf.int32),
    output_shapes=((None, IMG_SHAPE[0], IMG_SHAPE[1], IMG_SHAPE[2]), (None,))
)

val_dataset = tf.data.Dataset.from_generator(
    lambda: npz_generator(VAL_FOLDER),
    output_types=(tf.float32, tf.int32),
    output_shapes=((None, IMG_SHAPE[0], IMG_SHAPE[1], IMG_SHAPE[2]), (None,))
)

train_dataset = train_dataset.shuffle(100)

train_dataset = train_dataset.unbatch().batch(NEW_BATCH_SIZE)
```

```
val_dataset = val_dataset.unbatch().batch(NEW_BATCH_SIZE)

num_train_files = len(glob.glob(os.path.join(TRAIN_FOLDER, "*.npz")))
num_val_files   = len(glob.glob(os.path.join(VAL_FOLDER, "*.npz")))

steps_per_epoch = (num_train_files * BATCH_SIZE) // NEW_BATCH_SIZE
validation_steps = (num_val_files * BATCH_SIZE) // NEW_BATCH_SIZE


inputs = Input(shape=IMG_SHAPE)
base_model = ResNet50(weights='imagenet', include_top=False, input_tensor=inputs
x = base_model.output
x = Reshape((7 * 7, x.shape[-1]))(x)
x = Bidirectional(LSTM(256, return_sequences=False))(x)
x = Dropout(0.5)(x)
outputs = Dense(NUM_CLASSES, activation='softmax')(x)
model = Model(inputs=inputs, outputs=outputs)
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.summary()

checkpoint_cb = ModelCheckpoint(
    BEST_MODEL_PATH,
    monitor='val_accuracy',
    save_best_only=True,
    mode='max',
    verbose=1
)

history = model.fit(
    train_dataset,
    steps_per_epoch=steps_per_epoch,
    epochs=EPOCHS,
    validation_data=val_dataset,
    validation_steps=validation_steps,
    callbacks=[checkpoint_cb, TqdmCallback(verbose=1)]
)

model.save(FINAL_MODEL_PATH)
print(f"Final model saved to: {FINAL_MODEL_PATH}")
```

Model: "functional_1"

Layer (type)	Output Shape	Para
input_layer_1 (InputLayer)	(None, 224, 224, 3)	
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	
conv1_conv (Conv2D)	(None, 112, 112, 64)	9,
conv1_bn (BatchNormalization)	(None, 112, 112, 64)	
conv1_relu (Activation)	(None, 112, 112, 64)	
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4,
conv2_block1_1_bn (BatchNormalization)	(None, 56, 56, 64)	
conv2_block1_1_relu (Activation)	(None, 56, 56, 64)	
conv2_block1_2_conv (Conv2D)	(None, 56, 56, 64)	36,
conv2_block1_2_bn (BatchNormalization)	(None, 56, 56, 64)	
conv2_block1_2_relu (Activation)	(None, 56, 56, 64)	
conv2_block1_0_conv (Conv2D)	(None, 56, 56, 256)	16,
conv2_block1_3_conv (Conv2D)	(None, 56, 56, 256)	16,
conv2_block1_0_bn (BatchNormalization)	(None, 56, 56, 256)	1,
conv2_block1_3_bn (BatchNormalization)	(None, 56, 56, 256)	1,
conv2_block1_add (Add)	(None, 56, 56, 256)	
conv2_block1_out (Activation)	(None, 56, 56, 256)	
conv2_block2_1_conv (Conv2D)	(None, 56, 56, 64)	16,
conv2_block2_1_bn (BatchNormalization)	(None, 56, 56, 64)	
conv2_block2_1_relu (Activation)	(None, 56, 56, 64)	
conv2_block2_2_conv (Conv2D)	(None, 56, 56, 64)	36,
conv2_block2_2_bn (BatchNormalization)	(None, 56, 56, 64)	

conv2_block2_2_relu (Activation)	(None, 56, 56, 64)	
conv2_block2_3_conv (Conv2D)	(None, 56, 56, 256)	16,
conv2_block2_3_bn (BatchNormalization)	(None, 56, 56, 256)	1,
conv2_block2_add (Add)	(None, 56, 56, 256)	
conv2_block2_out (Activation)	(None, 56, 56, 256)	
conv2_block3_1_conv (Conv2D)	(None, 56, 56, 64)	16,
conv2_block3_1_bn (BatchNormalization)	(None, 56, 56, 64)	
conv2_block3_1_relu (Activation)	(None, 56, 56, 64)	
conv2_block3_2_conv (Conv2D)	(None, 56, 56, 64)	36,
conv2_block3_2_bn (BatchNormalization)	(None, 56, 56, 64)	
conv2_block3_2_relu (Activation)	(None, 56, 56, 64)	
conv2_block3_3_conv (Conv2D)	(None, 56, 56, 256)	16,
conv2_block3_3_bn (BatchNormalization)	(None, 56, 56, 256)	1,
conv2_block3_add (Add)	(None, 56, 56, 256)	
conv2_block3_out (Activation)	(None, 56, 56, 256)	
conv3_block1_1_conv (Conv2D)	(None, 28, 28, 128)	32,
conv3_block1_1_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block1_1_relu (Activation)	(None, 28, 28, 128)	
conv3_block1_2_conv (Conv2D)	(None, 28, 28, 128)	147,
conv3_block1_2_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block1_2_relu (Activation)	(None, 28, 28, 128)	
conv3_block1_0_conv (Conv2D)	(None, 28, 28, 512)	131,
conv3_block1_3_conv (Conv2D)	(None, 28, 28, 512)	66,

conv3_block1_0_bn (BatchNormalization)	(None, 28, 28, 512)	2,
conv3_block1_3_bn (BatchNormalization)	(None, 28, 28, 512)	2,
conv3_block1_add (Add)	(None, 28, 28, 512)	
conv3_block1_out (Activation)	(None, 28, 28, 512)	
conv3_block2_1_conv (Conv2D)	(None, 28, 28, 128)	65,
conv3_block2_1_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block2_1_relu (Activation)	(None, 28, 28, 128)	
conv3_block2_2_conv (Conv2D)	(None, 28, 28, 128)	147,
conv3_block2_2_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block2_2_relu (Activation)	(None, 28, 28, 128)	
conv3_block2_3_conv (Conv2D)	(None, 28, 28, 512)	66,
conv3_block2_3_bn (BatchNormalization)	(None, 28, 28, 512)	2,
conv3_block2_add (Add)	(None, 28, 28, 512)	
conv3_block2_out (Activation)	(None, 28, 28, 512)	
conv3_block3_1_conv (Conv2D)	(None, 28, 28, 128)	65,
conv3_block3_1_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block3_1_relu (Activation)	(None, 28, 28, 128)	
conv3_block3_2_conv (Conv2D)	(None, 28, 28, 128)	147,
conv3_block3_2_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block3_2_relu (Activation)	(None, 28, 28, 128)	
conv3_block3_3_conv (Conv2D)	(None, 28, 28, 512)	66,
conv3_block3_3_bn (BatchNormalization)	(None, 28, 28, 512)	2,
conv3_block3_add (Add)	(None, 28, 28, 512)	

conv3_block3_out (Activation)	(None, 28, 28, 512)	
conv3_block4_1_conv (Conv2D)	(None, 28, 28, 128)	65,
conv3_block4_1_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block4_1_relu (Activation)	(None, 28, 28, 128)	
conv3_block4_2_conv (Conv2D)	(None, 28, 28, 128)	147,
conv3_block4_2_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block4_2_relu (Activation)	(None, 28, 28, 128)	
conv3_block4_3_conv (Conv2D)	(None, 28, 28, 512)	66,
conv3_block4_3_bn (BatchNormalization)	(None, 28, 28, 512)	2,
conv3_block4_add (Add)	(None, 28, 28, 512)	
conv3_block4_out (Activation)	(None, 28, 28, 512)	
conv4_block1_1_conv (Conv2D)	(None, 14, 14, 256)	131,
conv4_block1_1_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block1_1_relu (Activation)	(None, 14, 14, 256)	
conv4_block1_2_conv (Conv2D)	(None, 14, 14, 256)	590,
conv4_block1_2_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block1_2_relu (Activation)	(None, 14, 14, 256)	
conv4_block1_0_conv (Conv2D)	(None, 14, 14, 1024)	525,
conv4_block1_3_conv (Conv2D)	(None, 14, 14, 1024)	263,
conv4_block1_0_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block1_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block1_add (Add)	(None, 14, 14, 1024)	
conv4_block1_out (Activation)	(None, 14, 14, 1024)	

conv4_block2_1_conv (Conv2D)	(None, 14, 14, 256)	262,
conv4_block2_1_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block2_1_relu (Activation)	(None, 14, 14, 256)	
conv4_block2_2_conv (Conv2D)	(None, 14, 14, 256)	590,
conv4_block2_2_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block2_2_relu (Activation)	(None, 14, 14, 256)	
conv4_block2_3_conv (Conv2D)	(None, 14, 14, 1024)	263,
conv4_block2_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block2_add (Add)	(None, 14, 14, 1024)	
conv4_block2_out (Activation)	(None, 14, 14, 1024)	
conv4_block3_1_conv (Conv2D)	(None, 14, 14, 256)	262,
conv4_block3_1_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block3_1_relu (Activation)	(None, 14, 14, 256)	
conv4_block3_2_conv (Conv2D)	(None, 14, 14, 256)	590,
conv4_block3_2_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block3_2_relu (Activation)	(None, 14, 14, 256)	
conv4_block3_3_conv (Conv2D)	(None, 14, 14, 1024)	263,
conv4_block3_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block3_add (Add)	(None, 14, 14, 1024)	
conv4_block3_out (Activation)	(None, 14, 14, 1024)	
conv4_block4_1_conv (Conv2D)	(None, 14, 14, 256)	262,
conv4_block4_1_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block4_1_relu (Activation)	(None, 14, 14, 256)	

conv4_block4_2_conv (Conv2D)	(None, 14, 14, 256)	590,
conv4_block4_2_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block4_2_relu (Activation)	(None, 14, 14, 256)	
conv4_block4_3_conv (Conv2D)	(None, 14, 14, 1024)	263,
conv4_block4_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block4_add (Add)	(None, 14, 14, 1024)	
conv4_block4_out (Activation)	(None, 14, 14, 1024)	
conv4_block5_1_conv (Conv2D)	(None, 14, 14, 256)	262,
conv4_block5_1_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block5_1_relu (Activation)	(None, 14, 14, 256)	
conv4_block5_2_conv (Conv2D)	(None, 14, 14, 256)	590,
conv4_block5_2_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block5_2_relu (Activation)	(None, 14, 14, 256)	
conv4_block5_3_conv (Conv2D)	(None, 14, 14, 1024)	263,
conv4_block5_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block5_add (Add)	(None, 14, 14, 1024)	
conv4_block5_out (Activation)	(None, 14, 14, 1024)	
conv4_block6_1_conv (Conv2D)	(None, 14, 14, 256)	262,
conv4_block6_1_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block6_1_relu (Activation)	(None, 14, 14, 256)	
conv4_block6_2_conv (Conv2D)	(None, 14, 14, 256)	590,
conv4_block6_2_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block6_2_relu (Activation)	(None, 14, 14, 256)	

conv4_block6_3_conv (Conv2D)	(None, 14, 14, 1024)	263,
conv4_block6_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block6_add (Add)	(None, 14, 14, 1024)	
conv4_block6_out (Activation)	(None, 14, 14, 1024)	
conv5_block1_1_conv (Conv2D)	(None, 7, 7, 512)	524,
conv5_block1_1_bn (BatchNormalization)	(None, 7, 7, 512)	2,
conv5_block1_1_relu (Activation)	(None, 7, 7, 512)	
conv5_block1_2_conv (Conv2D)	(None, 7, 7, 512)	2,359,
conv5_block1_2_bn (BatchNormalization)	(None, 7, 7, 512)	2,
conv5_block1_2_relu (Activation)	(None, 7, 7, 512)	
conv5_block1_0_conv (Conv2D)	(None, 7, 7, 2048)	2,099,
conv5_block1_3_conv (Conv2D)	(None, 7, 7, 2048)	1,050,
conv5_block1_0_bn (BatchNormalization)	(None, 7, 7, 2048)	8,
conv5_block1_3_bn (BatchNormalization)	(None, 7, 7, 2048)	8,
conv5_block1_add (Add)	(None, 7, 7, 2048)	
conv5_block1_out (Activation)	(None, 7, 7, 2048)	
conv5_block2_1_conv (Conv2D)	(None, 7, 7, 512)	1,049,
conv5_block2_1_bn (BatchNormalization)	(None, 7, 7, 512)	2,
conv5_block2_1_relu (Activation)	(None, 7, 7, 512)	
conv5_block2_2_conv (Conv2D)	(None, 7, 7, 512)	2,359,
conv5_block2_2_bn (BatchNormalization)	(None, 7, 7, 512)	2,
conv5_block2_2_relu (Activation)	(None, 7, 7, 512)	
conv5_block2_3_conv (Conv2D)	(None, 7, 7, 2048)	1,050,
conv5_block2_3_bn	(None, 7, 7, 2048)	8,

(BatchNormalization)		
conv5_block2_add (Add)	(None, 7, 7, 2048)	
conv5_block2_out (Activation)	(None, 7, 7, 2048)	
conv5_block3_1_conv (Conv2D)	(None, 7, 7, 512)	1,049,
conv5_block3_1_bn (BatchNormalization)	(None, 7, 7, 512)	2,
conv5_block3_1_relu (Activation)	(None, 7, 7, 512)	
conv5_block3_2_conv (Conv2D)	(None, 7, 7, 512)	2,359,
conv5_block3_2_bn (BatchNormalization)	(None, 7, 7, 512)	2,
conv5_block3_2_relu (Activation)	(None, 7, 7, 512)	
conv5_block3_3_conv (Conv2D)	(None, 7, 7, 2048)	1,050,
conv5_block3_3_bn (BatchNormalization)	(None, 7, 7, 2048)	8,
conv5_block3_add (Add)	(None, 7, 7, 2048)	
conv5_block3_out (Activation)	(None, 7, 7, 2048)	
reshape_1 (Reshape)	(None, 49, 2048)	
bidirectional_1 (Bidirectional)	(None, 512)	4,720,
dropout_1 (Dropout)	(None, 512)	
dense_1 (Dense)	(None, 23)	11,

Total params: 28,320,151 (108.03 MB)

Trainable params: 28,267,031 (107.83 MB)

Non-trainable params: 53,120 (207.50 KB)

0epoch [00:00, ?epoch/s]
0batch [00:00, ?batch/s]

```
Epoch 1/5
Warning: File processed_batches_artists\batch_13.npz has only 345 images (expecte
d 1000). Yielding partial batch.
Warning: File processed_batches_artists\batch_9.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_artists_val\batch_5.npz has only 706 images (expe
cted 1000). Yielding partial batch.

Epoch 1: val_accuracy improved from -inf to 0.09551, saving model to saved_models
\best_artist_model.keras
140/140 ━━━━━━━━━━━━━━━━━━━━ 1781s 12s/step - accuracy: 0.3719 - loss: 2.1889 - v
al_accuracy: 0.0955 - val_loss: 4.6661
Epoch 2/5
Warning: File processed_batches_artists\batch_13.npz has only 345 images (expecte
d 1000). Yielding partial batch.
Warning: File processed_batches_artists\batch_9.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_artists_val\batch_5.npz has only 706 images (expe
cted 1000). Yielding partial batch.

Epoch 2: val_accuracy did not improve from 0.09551
140/140 ━━━━━━━━━━━━━━━━━━━━ 1746s 12s/step - accuracy: 0.5458 - loss: 1.6012 - v
al_accuracy: 0.0408 - val_loss: 4.5888
Epoch 3/5
Warning: File processed_batches_artists\batch_13.npz has only 345 images (expecte
d 1000). Yielding partial batch.
Warning: File processed_batches_artists\batch_9.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_artists_val\batch_5.npz has only 706 images (expe
cted 1000). Yielding partial batch.

Epoch 3: val_accuracy did not improve from 0.09551
140/140 ━━━━━━━━━━━━━━━━━━━━ 1748s 12s/step - accuracy: 0.6169 - loss: 1.3909 - v
al_accuracy: 0.0408 - val_loss: 4.8301
Epoch 4/5
Warning: File processed_batches_artists\batch_13.npz has only 345 images (expecte
d 1000). Yielding partial batch.
Warning: File processed_batches_artists\batch_9.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_artists_val\batch_5.npz has only 706 images (expe
cted 1000). Yielding partial batch.

Epoch 4: val_accuracy did not improve from 0.09551
140/140 ━━━━━━━━━━━━━━━━━━━━ 1712s 12s/step - accuracy: 0.7618 - loss: 0.8301 - v
al_accuracy: 0.0613 - val_loss: 4.4296
Epoch 5/5
Warning: File processed_batches_artists\batch_13.npz has only 345 images (expecte
d 1000). Yielding partial batch.
Warning: File processed_batches_artists\batch_9.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_artists_val\batch_5.npz has only 706 images (expe
cted 1000). Yielding partial batch.

Epoch 5: val_accuracy improved from 0.09551 to 0.10060, saving model to saved_mod
els\best_artist_model.keras
140/140 ━━━━━━━━━━━━━━━━━━━━ 1706s 12s/step - accuracy: 0.6946 - loss: 1.1358 - v
al_accuracy: 0.1006 - val_loss: 5.4637
Final model saved to: saved_models\final_artist_model.keras
```

In [ ]:

In [ ]:
```
import os
import glob
import numpy as np
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import Input, Reshape, Bidirectional, LSTM, Dense,
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import ModelCheckpoint
from tqdm.keras import TqdmCallback

TRAIN_FOLDER = "processed_batches_genres"
VAL_FOLDER = "processed_batches_genres_val"
IMG_SHAPE = (224, 224, 3)
NUM_CLASSES = 10
BATCH_SIZE = 1000
NEW_BATCH_SIZE = 100
EPOCHS = 5

SAVE_DIR = "saved_models"
os.makedirs(SAVE_DIR, exist_ok=True)
BEST_MODEL_PATH = os.path.join(SAVE_DIR, "best_genre_model.keras")
FINAL_MODEL_PATH = os.path.join(SAVE_DIR, "final_genre_model.keras")


def npz_generator(folder):
    """
    Generator that yields (images, labels) for each npz file in the folder.
    If the npz file has fewer than the expected BATCH_SIZE images, it yields the
    """
    files = sorted(glob.glob(os.path.join(folder, "*.npz")))
    for f in files:
        data = np.load(f)
        imgs = data['arr_0']
        labels = data['labels']
        current_batch_size = imgs.shape[0]
        if current_batch_size != BATCH_SIZE:
            print(f"Warning: File {f} has only {current_batch_size} images (expe
        yield imgs, labels

train_dataset = tf.data.Dataset.from_generator(
    lambda: npz_generator(TRAIN_FOLDER),
    output_types=(tf.float32, tf.int32),
    output_shapes=((None, IMG_SHAPE[0], IMG_SHAPE[1], IMG_SHAPE[2]), (None,))
)

val_dataset = tf.data.Dataset.from_generator(
    lambda: npz_generator(VAL_FOLDER),
    output_types=(tf.float32, tf.int32),
    output_shapes=((None, IMG_SHAPE[0], IMG_SHAPE[1], IMG_SHAPE[2]), (None,))
)

train_dataset = train_dataset.shuffle(100)

train_dataset = train_dataset.unbatch().batch(NEW_BATCH_SIZE)
val_dataset = val_dataset.unbatch().batch(NEW_BATCH_SIZE)

num_train_files = len(glob.glob(os.path.join(TRAIN_FOLDER, "*.npz")))
```

```
num_val_files   = len(glob.glob(os.path.join(VAL_FOLDER, "*.npz")))

steps_per_epoch = (num_train_files * BATCH_SIZE) // NEW_BATCH_SIZE
validation_steps = (num_val_files * BATCH_SIZE) // NEW_BATCH_SIZE

inputs = Input(shape=IMG_SHAPE)
base_model = ResNet50(weights='imagenet', include_top=False, input_tensor=inputs
x = base_model.output
x = Reshape((7 * 7, x.shape[-1]))(x)
x = Bidirectional(LSTM(256, return_sequences=False))(x)
x = Dropout(0.5)(x)
outputs = Dense(NUM_CLASSES, activation='softmax')(x)
model = Model(inputs=inputs, outputs=outputs)
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.summary()

checkpoint_cb = ModelCheckpoint(
    BEST_MODEL_PATH,
    monitor='val_accuracy',
    save_best_only=True,
    mode='max',
    verbose=1
)

history = model.fit(
    train_dataset,
    steps_per_epoch=steps_per_epoch,
    epochs=EPOCHS,
    validation_data=val_dataset,
    validation_steps=validation_steps,
    callbacks=[checkpoint_cb, TqdmCallback(verbose=1)]
)

model.save(FINAL_MODEL_PATH)
print(f"Final model saved to: {FINAL_MODEL_PATH}")
```

Model: "functional_2"

Layer (type)	Output Shape	Para
input_layer_2 (InputLayer)	(None, 224, 224, 3)	
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	
conv1_conv (Conv2D)	(None, 112, 112, 64)	9,
conv1_bn (BatchNormalization)	(None, 112, 112, 64)	
conv1_relu (Activation)	(None, 112, 112, 64)	
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4,
conv2_block1_1_bn (BatchNormalization)	(None, 56, 56, 64)	
conv2_block1_1_relu (Activation)	(None, 56, 56, 64)	
conv2_block1_2_conv (Conv2D)	(None, 56, 56, 64)	36,
conv2_block1_2_bn (BatchNormalization)	(None, 56, 56, 64)	
conv2_block1_2_relu (Activation)	(None, 56, 56, 64)	
conv2_block1_0_conv (Conv2D)	(None, 56, 56, 256)	16,
conv2_block1_3_conv (Conv2D)	(None, 56, 56, 256)	16,
conv2_block1_0_bn (BatchNormalization)	(None, 56, 56, 256)	1,
conv2_block1_3_bn (BatchNormalization)	(None, 56, 56, 256)	1,
conv2_block1_add (Add)	(None, 56, 56, 256)	
conv2_block1_out (Activation)	(None, 56, 56, 256)	
conv2_block2_1_conv (Conv2D)	(None, 56, 56, 64)	16,
conv2_block2_1_bn (BatchNormalization)	(None, 56, 56, 64)	
conv2_block2_1_relu (Activation)	(None, 56, 56, 64)	
conv2_block2_2_conv (Conv2D)	(None, 56, 56, 64)	36,
conv2_block2_2_bn (BatchNormalization)	(None, 56, 56, 64)	

conv2_block2_2_relu (Activation)	(None, 56, 56, 64)	
conv2_block2_3_conv (Conv2D)	(None, 56, 56, 256)	16,
conv2_block2_3_bn (BatchNormalization)	(None, 56, 56, 256)	1,
conv2_block2_add (Add)	(None, 56, 56, 256)	
conv2_block2_out (Activation)	(None, 56, 56, 256)	
conv2_block3_1_conv (Conv2D)	(None, 56, 56, 64)	16,
conv2_block3_1_bn (BatchNormalization)	(None, 56, 56, 64)	
conv2_block3_1_relu (Activation)	(None, 56, 56, 64)	
conv2_block3_2_conv (Conv2D)	(None, 56, 56, 64)	36,
conv2_block3_2_bn (BatchNormalization)	(None, 56, 56, 64)	
conv2_block3_2_relu (Activation)	(None, 56, 56, 64)	
conv2_block3_3_conv (Conv2D)	(None, 56, 56, 256)	16,
conv2_block3_3_bn (BatchNormalization)	(None, 56, 56, 256)	1,
conv2_block3_add (Add)	(None, 56, 56, 256)	
conv2_block3_out (Activation)	(None, 56, 56, 256)	
conv3_block1_1_conv (Conv2D)	(None, 28, 28, 128)	32,
conv3_block1_1_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block1_1_relu (Activation)	(None, 28, 28, 128)	
conv3_block1_2_conv (Conv2D)	(None, 28, 28, 128)	147,
conv3_block1_2_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block1_2_relu (Activation)	(None, 28, 28, 128)	
conv3_block1_0_conv (Conv2D)	(None, 28, 28, 512)	131,
conv3_block1_3_conv (Conv2D)	(None, 28, 28, 512)	66,

conv3_block1_0_bn (BatchNormalization)	(None, 28, 28, 512)	2,
conv3_block1_3_bn (BatchNormalization)	(None, 28, 28, 512)	2,
conv3_block1_add (Add)	(None, 28, 28, 512)	
conv3_block1_out (Activation)	(None, 28, 28, 512)	
conv3_block2_1_conv (Conv2D)	(None, 28, 28, 128)	65,
conv3_block2_1_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block2_1_relu (Activation)	(None, 28, 28, 128)	
conv3_block2_2_conv (Conv2D)	(None, 28, 28, 128)	147,
conv3_block2_2_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block2_2_relu (Activation)	(None, 28, 28, 128)	
conv3_block2_3_conv (Conv2D)	(None, 28, 28, 512)	66,
conv3_block2_3_bn (BatchNormalization)	(None, 28, 28, 512)	2,
conv3_block2_add (Add)	(None, 28, 28, 512)	
conv3_block2_out (Activation)	(None, 28, 28, 512)	
conv3_block3_1_conv (Conv2D)	(None, 28, 28, 128)	65,
conv3_block3_1_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block3_1_relu (Activation)	(None, 28, 28, 128)	
conv3_block3_2_conv (Conv2D)	(None, 28, 28, 128)	147,
conv3_block3_2_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block3_2_relu (Activation)	(None, 28, 28, 128)	
conv3_block3_3_conv (Conv2D)	(None, 28, 28, 512)	66,
conv3_block3_3_bn (BatchNormalization)	(None, 28, 28, 512)	2,
conv3_block3_add (Add)	(None, 28, 28, 512)	

conv3_block3_out (Activation)	(None, 28, 28, 512)	
conv3_block4_1_conv (Conv2D)	(None, 28, 28, 128)	65,
conv3_block4_1_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block4_1_relu (Activation)	(None, 28, 28, 128)	
conv3_block4_2_conv (Conv2D)	(None, 28, 28, 128)	147,
conv3_block4_2_bn (BatchNormalization)	(None, 28, 28, 128)	
conv3_block4_2_relu (Activation)	(None, 28, 28, 128)	
conv3_block4_3_conv (Conv2D)	(None, 28, 28, 512)	66,
conv3_block4_3_bn (BatchNormalization)	(None, 28, 28, 512)	2,
conv3_block4_add (Add)	(None, 28, 28, 512)	
conv3_block4_out (Activation)	(None, 28, 28, 512)	
conv4_block1_1_conv (Conv2D)	(None, 14, 14, 256)	131,
conv4_block1_1_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block1_1_relu (Activation)	(None, 14, 14, 256)	
conv4_block1_2_conv (Conv2D)	(None, 14, 14, 256)	590,
conv4_block1_2_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block1_2_relu (Activation)	(None, 14, 14, 256)	
conv4_block1_0_conv (Conv2D)	(None, 14, 14, 1024)	525,
conv4_block1_3_conv (Conv2D)	(None, 14, 14, 1024)	263,
conv4_block1_0_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block1_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block1_add (Add)	(None, 14, 14, 1024)	
conv4_block1_out (Activation)	(None, 14, 14, 1024)	

conv4_block2_1_conv (Conv2D)	(None, 14, 14, 256)	262,
conv4_block2_1_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block2_1_relu (Activation)	(None, 14, 14, 256)	
conv4_block2_2_conv (Conv2D)	(None, 14, 14, 256)	590,
conv4_block2_2_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block2_2_relu (Activation)	(None, 14, 14, 256)	
conv4_block2_3_conv (Conv2D)	(None, 14, 14, 1024)	263,
conv4_block2_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block2_add (Add)	(None, 14, 14, 1024)	
conv4_block2_out (Activation)	(None, 14, 14, 1024)	
conv4_block3_1_conv (Conv2D)	(None, 14, 14, 256)	262,
conv4_block3_1_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block3_1_relu (Activation)	(None, 14, 14, 256)	
conv4_block3_2_conv (Conv2D)	(None, 14, 14, 256)	590,
conv4_block3_2_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block3_2_relu (Activation)	(None, 14, 14, 256)	
conv4_block3_3_conv (Conv2D)	(None, 14, 14, 1024)	263,
conv4_block3_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block3_add (Add)	(None, 14, 14, 1024)	
conv4_block3_out (Activation)	(None, 14, 14, 1024)	
conv4_block4_1_conv (Conv2D)	(None, 14, 14, 256)	262,
conv4_block4_1_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block4_1_relu (Activation)	(None, 14, 14, 256)	

conv4_block4_2_conv (Conv2D)	(None, 14, 14, 256)	590,
conv4_block4_2_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block4_2_relu (Activation)	(None, 14, 14, 256)	
conv4_block4_3_conv (Conv2D)	(None, 14, 14, 1024)	263,
conv4_block4_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block4_add (Add)	(None, 14, 14, 1024)	
conv4_block4_out (Activation)	(None, 14, 14, 1024)	
conv4_block5_1_conv (Conv2D)	(None, 14, 14, 256)	262,
conv4_block5_1_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block5_1_relu (Activation)	(None, 14, 14, 256)	
conv4_block5_2_conv (Conv2D)	(None, 14, 14, 256)	590,
conv4_block5_2_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block5_2_relu (Activation)	(None, 14, 14, 256)	
conv4_block5_3_conv (Conv2D)	(None, 14, 14, 1024)	263,
conv4_block5_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block5_add (Add)	(None, 14, 14, 1024)	
conv4_block5_out (Activation)	(None, 14, 14, 1024)	
conv4_block6_1_conv (Conv2D)	(None, 14, 14, 256)	262,
conv4_block6_1_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block6_1_relu (Activation)	(None, 14, 14, 256)	
conv4_block6_2_conv (Conv2D)	(None, 14, 14, 256)	590,
conv4_block6_2_bn (BatchNormalization)	(None, 14, 14, 256)	1,
conv4_block6_2_relu (Activation)	(None, 14, 14, 256)	

conv4_block6_3_conv (Conv2D)	(None, 14, 14, 1024)	263,
conv4_block6_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4,
conv4_block6_add (Add)	(None, 14, 14, 1024)	
conv4_block6_out (Activation)	(None, 14, 14, 1024)	
conv5_block1_1_conv (Conv2D)	(None, 7, 7, 512)	524,
conv5_block1_1_bn (BatchNormalization)	(None, 7, 7, 512)	2,
conv5_block1_1_relu (Activation)	(None, 7, 7, 512)	
conv5_block1_2_conv (Conv2D)	(None, 7, 7, 512)	2,359,
conv5_block1_2_bn (BatchNormalization)	(None, 7, 7, 512)	2,
conv5_block1_2_relu (Activation)	(None, 7, 7, 512)	
conv5_block1_0_conv (Conv2D)	(None, 7, 7, 2048)	2,099,
conv5_block1_3_conv (Conv2D)	(None, 7, 7, 2048)	1,050,
conv5_block1_0_bn (BatchNormalization)	(None, 7, 7, 2048)	8,
conv5_block1_3_bn (BatchNormalization)	(None, 7, 7, 2048)	8,
conv5_block1_add (Add)	(None, 7, 7, 2048)	
conv5_block1_out (Activation)	(None, 7, 7, 2048)	
conv5_block2_1_conv (Conv2D)	(None, 7, 7, 512)	1,049,
conv5_block2_1_bn (BatchNormalization)	(None, 7, 7, 512)	2,
conv5_block2_1_relu (Activation)	(None, 7, 7, 512)	
conv5_block2_2_conv (Conv2D)	(None, 7, 7, 512)	2,359,
conv5_block2_2_bn (BatchNormalization)	(None, 7, 7, 512)	2,
conv5_block2_2_relu (Activation)	(None, 7, 7, 512)	
conv5_block2_3_conv (Conv2D)	(None, 7, 7, 2048)	1,050,
conv5_block2_3_bn	(None, 7, 7, 2048)	8,

(BatchNormalization)		
conv5_block2_add (Add)	(None, 7, 7, 2048)	
conv5_block2_out (Activation)	(None, 7, 7, 2048)	
conv5_block3_1_conv (Conv2D)	(None, 7, 7, 512)	1,049,
conv5_block3_1_bn (BatchNormalization)	(None, 7, 7, 512)	2,
conv5_block3_1_relu (Activation)	(None, 7, 7, 512)	
conv5_block3_2_conv (Conv2D)	(None, 7, 7, 512)	2,359,
conv5_block3_2_bn (BatchNormalization)	(None, 7, 7, 512)	2,
conv5_block3_2_relu (Activation)	(None, 7, 7, 512)	
conv5_block3_3_conv (Conv2D)	(None, 7, 7, 2048)	1,050,
conv5_block3_3_bn (BatchNormalization)	(None, 7, 7, 2048)	8,
conv5_block3_add (Add)	(None, 7, 7, 2048)	
conv5_block3_out (Activation)	(None, 7, 7, 2048)	
reshape_2 (Reshape)	(None, 49, 2048)	
bidirectional_2 (Bidirectional)	(None, 512)	4,720,
dropout_2 (Dropout)	(None, 512)	
dense_2 (Dense)	(None, 10)	5,

Total params: 28,313,482 (108.01 MB)

Trainable params: 28,260,362 (107.80 MB)

Non-trainable params: 53,120 (207.50 KB)

0epoch [00:00, ?epoch/s]
0batch [00:00, ?batch/s]

```
Epoch 1/5
Warning: File processed_batches_genres\batch_3.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_genres\batch_45.npz has only 502 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_genres_val\batch_19.npz has only 492 images (expe
cted 1000). Yielding partial batch.

Epoch 1: val_accuracy improved from -inf to 0.07654, saving model to saved_models
\best_genre_model.keras
460/460 ━━━━━━━━━━━━━━━━━━━━ 6208s 13s/step - accuracy: 0.5240 - loss: 1.4178 - v
al_accuracy: 0.0765 - val_loss: 4.1509
Epoch 2/5
Warning: File processed_batches_genres\batch_3.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_genres\batch_45.npz has only 502 images (expected
1000). Yielding partial batch.
378/460 ━━━━━━━━━━━━━━━━━━━━ 16:11 12s/step - accuracy: 0.6284 - loss: 1.0961
```

In [ ]:

correcting the overfitted model, as the val_accuracy is very low compared to batch accuracy

In [6]:
```
TRAIN_FOLDER = "processed_batches_artists"
VAL_FOLDER = "processed_batches_artists_val"
IMG_SHAPE = (224, 224, 3)
NUM_CLASSES = 23
BATCH_SIZE = 1000
NEW_BATCH_SIZE = 100
EPOCHS = 5

SAVE_DIR = "saved_models"
os.makedirs(SAVE_DIR, exist_ok=True)
BEST_MODEL_PATH = os.path.join(SAVE_DIR, "best_artist_model.keras")
FINAL_MODEL_PATH = os.path.join(SAVE_DIR, "final_artist_model.keras")
INITIAL_WEIGHTS_PATH = os.path.join(SAVE_DIR, "best_artist_model.keras")


def npz_generator(folder):
    """
    Generator that yields (images, labels) from each npz file.
    If a file has fewer than the expected BATCH_SIZE images, it yields a partial
    """
    files = sorted(glob.glob(os.path.join(folder, "*.npz")))
    for f in files:
        data = np.load(f)
        imgs = data['arr_0']
        labels = data['labels']
        current_batch_size = imgs.shape[0]
        if current_batch_size != BATCH_SIZE:
            print(f"Warning: File {f} has only {current_batch_size} images (expe
        yield imgs, labels

train_dataset = tf.data.Dataset.from_generator(
    lambda: npz_generator(TRAIN_FOLDER),
    output_types=(tf.float32, tf.int32),
    output_shapes=((None, IMG_SHAPE[0], IMG_SHAPE[1], IMG_SHAPE[2]), (None,))
)
```

```
val_dataset = tf.data.Dataset.from_generator(
    lambda: npz_generator(VAL_FOLDER),
    output_types=(tf.float32, tf.int32),
    output_shapes=((None, IMG_SHAPE[0], IMG_SHAPE[1], IMG_SHAPE[2]), (None,))
)

train_dataset = train_dataset.shuffle(100).unbatch().batch(NEW_BATCH_SIZE)
val_dataset = val_dataset.unbatch().batch(NEW_BATCH_SIZE)

num_train_files = len(glob.glob(os.path.join(TRAIN_FOLDER, "*.npz")))
num_val_files = len(glob.glob(os.path.join(VAL_FOLDER, "*.npz")))
steps_per_epoch = (num_train_files * BATCH_SIZE) // NEW_BATCH_SIZE
validation_steps = (num_val_files * BATCH_SIZE) // NEW_BATCH_SIZE

inputs = Input(shape=IMG_SHAPE)

if os.path.exists(INITIAL_WEIGHTS_PATH):
    print(f"Found pre-trained weights at {INITIAL_WEIGHTS_PATH}.")
    base_model = ResNet50(weights='imagenet', include_top=False, input_shape=IMG
else:
    print("No pre-trained model found; using fresh ResNet50 with ImageNet weight
    base_model = ResNet50(weights='imagenet', include_top=False, input_shape=IMG

# new classifier head
x = base_model(inputs)
x = GlobalAveragePooling2D()(x)
x = BatchNormalization()(x)
x = Dropout(0.75)(x)
outputs = Dense(NUM_CLASSES, activation='softmax', kernel_regularizer=tf.keras.r

model = Model(inputs=inputs, outputs=outputs)

# Compiling with a lower learning rate to help fine tuning
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.summary()

checkpoint_cb = ModelCheckpoint(
    BEST_MODEL_PATH,
    monitor='val_accuracy',
    save_best_only=True,
    mode='max',
    verbose=1
)
reduce_lr_cb = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=2, ver

history = model.fit(
    train_dataset,
    steps_per_epoch=steps_per_epoch,
    epochs=EPOCHS,
    validation_data=val_dataset,
    validation_steps=validation_steps,
    callbacks=[checkpoint_cb, reduce_lr_cb, TqdmCallback(verbose=1)]
)

model.save(FINAL_MODEL_PATH)
print(f"Final model saved to: {FINAL_MODEL_PATH}")
```

```
Found pre-trained weights at saved_models\best_artist_model.keras.
```

Model: "functional_2"

Layer (type)	Output Shape	
input_layer_6 (InputLayer)	(None, 224, 224, 3)	
resnet50 (Functional)	(None, 7, 7, 2048)	
global_average_pooling2d_2 (GlobalAveragePooling2D)	(None, 2048)	
batch_normalization_2 (BatchNormalization)	(None, 2048)	
dropout_2 (Dropout)	(None, 2048)	
dense_2 (Dense)	(None, 23)	

Total params: 23,643,031 (90.19 MB)

Trainable params: 23,585,815 (89.97 MB)

Non-trainable params: 57,216 (223.50 KB)

0epoch [00:00, ?epoch/s]
0batch [00:00, ?batch/s]

```
Epoch 1/5
Warning: File processed_batches_artists\batch_13.npz has only 345 images (expecte
d 1000). Yielding partial batch.
Warning: File processed_batches_artists\batch_9.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_artists_val\batch_5.npz has only 706 images (expe
cted 1000). Yielding partial batch.

Epoch 1: val_accuracy improved from -inf to 0.09551, saving model to saved_models
\best_artist_model.keras
140/140 ━━━━━━━━━━━━━━━━━━━━ 1683s 11s/step - accuracy: 0.2422 - loss: 8.2306 - v
al_accuracy: 0.0955 - val_loss: 8.2328 - learning_rate: 1.0000e-04
Epoch 2/5
Warning: File processed_batches_artists\batch_13.npz has only 345 images (expecte
d 1000). Yielding partial batch.
Warning: File processed_batches_artists\batch_9.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_artists_val\batch_5.npz has only 706 images (expe
cted 1000). Yielding partial batch.

Epoch 2: val_accuracy did not improve from 0.09551
140/140 ━━━━━━━━━━━━━━━━━━━━ 1626s 11s/step - accuracy: 0.5519 - loss: 5.5671 - v
al_accuracy: 0.0955 - val_loss: 10.7166 - learning_rate: 1.0000e-04
Epoch 3/5
Warning: File processed_batches_artists\batch_13.npz has only 345 images (expecte
d 1000). Yielding partial batch.
Warning: File processed_batches_artists\batch_9.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_artists_val\batch_5.npz has only 706 images (expe
cted 1000). Yielding partial batch.

Epoch 3: val_accuracy improved from 0.09551 to 0.09586, saving model to saved_mod
els\best_artist_model.keras

Epoch 3: ReduceLROnPlateau reducing learning rate to 4.999999873689376e-05.
140/140 ━━━━━━━━━━━━━━━━━━━━ 1624s 11s/step - accuracy: 0.7165 - loss: 4.1959 - v
al_accuracy: 0.0959 - val_loss: 8.4703 - learning_rate: 1.0000e-04
Epoch 4/5
Warning: File processed_batches_artists\batch_13.npz has only 345 images (expecte
d 1000). Yielding partial batch.
Warning: File processed_batches_artists\batch_9.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_artists_val\batch_5.npz has only 706 images (expe
cted 1000). Yielding partial batch.

Epoch 4: val_accuracy improved from 0.09586 to 0.09709, saving model to saved_mod
els\best_artist_model.keras
140/140 ━━━━━━━━━━━━━━━━━━━━ 1918s 13s/step - accuracy: 0.8688 - loss: 2.9660 - v
al_accuracy: 0.0971 - val_loss: 7.4612 - learning_rate: 5.0000e-05
Epoch 5/5
Warning: File processed_batches_artists\batch_13.npz has only 345 images (expecte
d 1000). Yielding partial batch.
Warning: File processed_batches_artists\batch_9.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_artists_val\batch_5.npz has only 706 images (expe
cted 1000). Yielding partial batch.

Epoch 5: val_accuracy improved from 0.09709 to 0.12723, saving model to saved_mod
els\best_artist_model.keras
140/140 ━━━━━━━━━━━━━━━━━━━━ 1655s 12s/step - accuracy: 0.8819 - loss: 2.7051 - v
```

```
al_accuracy: 0.1272 - val_loss: 8.6164 - learning_rate: 5.0000e-05
Final model saved to: saved_models\final_artist_model.keras
```

In [ ]:

In [7]:
```
TRAIN_FOLDER = "processed_batches_genres"
VAL_FOLDER = "processed_batches_genres_val"
IMG_SHAPE = (224, 224, 3)
NUM_CLASSES = 10
BATCH_SIZE = 1000
NEW_BATCH_SIZE = 100
EPOCHS = 5

SAVE_DIR = "saved_models"
os.makedirs(SAVE_DIR, exist_ok=True)
BEST_MODEL_PATH = os.path.join(SAVE_DIR, "best_genre_model.keras")
FINAL_MODEL_PATH = os.path.join(SAVE_DIR, "final_genre_model.keras")
INITIAL_WEIGHTS_PATH = os.path.join(SAVE_DIR, "best_genre_model.keras")


def npz_generator(folder):
    """
    Generator that yields (images, labels) from each npz file.
    If a file has fewer than the expected BATCH_SIZE images, it yields a partial
    """
    files = sorted(glob.glob(os.path.join(folder, "*.npz")))
    for f in files:
        data = np.load(f)
        imgs = data['arr_0']
        labels = data['labels']
        current_batch_size = imgs.shape[0]
        if current_batch_size != BATCH_SIZE:
            print(f"Warning: File {f} has only {current_batch_size} images (expe
        yield imgs, labels

train_dataset = tf.data.Dataset.from_generator(
    lambda: npz_generator(TRAIN_FOLDER),
    output_types=(tf.float32, tf.int32),
    output_shapes=((None, IMG_SHAPE[0], IMG_SHAPE[1], IMG_SHAPE[2]), (None,))
)
val_dataset = tf.data.Dataset.from_generator(
    lambda: npz_generator(VAL_FOLDER),
    output_types=(tf.float32, tf.int32),
    output_shapes=((None, IMG_SHAPE[0], IMG_SHAPE[1], IMG_SHAPE[2]), (None,))
)


train_dataset = train_dataset.shuffle(100).unbatch().batch(NEW_BATCH_SIZE)
val_dataset = val_dataset.unbatch().batch(NEW_BATCH_SIZE)

num_train_files = len(glob.glob(os.path.join(TRAIN_FOLDER, "*.npz")))
num_val_files = len(glob.glob(os.path.join(VAL_FOLDER, "*.npz")))
steps_per_epoch = (num_train_files * BATCH_SIZE) // NEW_BATCH_SIZE
validation_steps = (num_val_files * BATCH_SIZE) // NEW_BATCH_SIZE

inputs = Input(shape=IMG_SHAPE)

if os.path.exists(INITIAL_WEIGHTS_PATH):
    print(f"Found pre-trained weights at {INITIAL_WEIGHTS_PATH}.")
    base_model = ResNet50(weights='imagenet', include_top=False, input_shape=IMG
```

```
else:
    print("No pre-trained model found; using fresh ResNet50 with ImageNet weight
    base_model = ResNet50(weights='imagenet', include_top=False, input_shape=IMG

x = base_model(inputs)
x = GlobalAveragePooling2D()(x)
x = BatchNormalization()(x)
x = Dropout(0.75)(x)
outputs = Dense(NUM_CLASSES, activation='softmax', kernel_regularizer=tf.keras.r

model = Model(inputs=inputs, outputs=outputs)

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.summary()

checkpoint_cb = ModelCheckpoint(
    BEST_MODEL_PATH,
    monitor='val_accuracy',
    save_best_only=True,
    mode='max',
    verbose=1
)
reduce_lr_cb = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=2, ver

history = model.fit(
    train_dataset,
    steps_per_epoch=steps_per_epoch,
    epochs=EPOCHS,
    validation_data=val_dataset,
    validation_steps=validation_steps,
    callbacks=[checkpoint_cb, reduce_lr_cb, TqdmCallback(verbose=1)]
)

model.save(FINAL_MODEL_PATH)
print(f"Final model saved to: {FINAL_MODEL_PATH}")
```

```
Found pre-trained weights at saved_models\best_genre_model.keras.
```

Model: "functional_3"

Layer (type)	Output Shape	
input_layer_8 (InputLayer)	(None, 224, 224, 3)	
resnet50 (Functional)	(None, 7, 7, 2048)	
global_average_pooling2d_3 (GlobalAveragePooling2D)	(None, 2048)	
batch_normalization_3 (BatchNormalization)	(None, 2048)	
dropout_3 (Dropout)	(None, 2048)	
dense_3 (Dense)	(None, 10)	

Total params: 23,616,394 (90.09 MB)

Trainable params: 23,559,178 (89.87 MB)

Non-trainable params: 57,216 (223.50 KB)

0epoch [00:00, ?epoch/s]
0batch [00:00, ?batch/s]

```
Epoch 1/5
Warning: File processed_batches_genres\batch_3.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_genres\batch_45.npz has only 502 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_genres_val\batch_19.npz has only 492 images (expe
cted 1000). Yielding partial batch.

Epoch 1: val_accuracy improved from -inf to 0.07926, saving model to saved_models
\best_genre_model.keras
460/460 ━━━━━━━━━━━━━━━━━━━━ 5692s 12s/step - accuracy: 0.4980 - loss: 3.9001 - v
al_accuracy: 0.0793 - val_loss: 42.5889 - learning_rate: 1.0000e-04
Epoch 2/5
Warning: File processed_batches_genres\batch_3.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_genres\batch_45.npz has only 502 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_genres_val\batch_19.npz has only 492 images (expe
cted 1000). Yielding partial batch.

Epoch 2: val_accuracy improved from 0.07926 to 0.64914, saving model to saved_mod
els\best_genre_model.keras
460/460 ━━━━━━━━━━━━━━━━━━━━ 5726s 12s/step - accuracy: 0.7426 - loss: 1.8785 - v
al_accuracy: 0.6491 - val_loss: 2.0057 - learning_rate: 1.0000e-04
Epoch 3/5
Warning: File processed_batches_genres\batch_3.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_genres\batch_45.npz has only 502 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_genres_val\batch_19.npz has only 492 images (expe
cted 1000). Yielding partial batch.

Epoch 3: val_accuracy improved from 0.64914 to 0.75195, saving model to saved_mod
els\best_genre_model.keras
460/460 ━━━━━━━━━━━━━━━━━━━━ 5764s 12s/step - accuracy: 0.7978 - loss: 1.2794 - v
al_accuracy: 0.7519 - val_loss: 1.0920 - learning_rate: 1.0000e-04
Epoch 4/5
Warning: File processed_batches_genres\batch_3.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_genres\batch_45.npz has only 502 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_genres_val\batch_19.npz has only 492 images (expe
cted 1000). Yielding partial batch.

Epoch 4: val_accuracy did not improve from 0.75195
460/460 ━━━━━━━━━━━━━━━━━━━━ 5605s 12s/step - accuracy: 0.8681 - loss: 0.7876 - v
al_accuracy: 0.7217 - val_loss: 1.0882 - learning_rate: 1.0000e-04
Epoch 5/5
Warning: File processed_batches_genres\batch_3.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_genres\batch_45.npz has only 502 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_genres_val\batch_19.npz has only 492 images (expe
cted 1000). Yielding partial batch.

Epoch 5: val_accuracy did not improve from 0.75195
460/460 ━━━━━━━━━━━━━━━━━━━━ 5627s 12s/step - accuracy: 0.9271 - loss: 0.4263 - v
al_accuracy: 0.7324 - val_loss: 1.0560 - learning_rate: 1.0000e-04
Final model saved to: saved_models\final_genre_model.keras
```

In [ ]:

In [3]:
```
TRAIN_FOLDER = "processed_batches_styles"
VAL_FOLDER = "processed_batches_styles_val"
IMG_SHAPE = (224, 224, 3)
NUM_CLASSES = 27
BATCH_SIZE = 1000
NEW_BATCH_SIZE = 100
EPOCHS = 3

SAVE_DIR = "saved_models"
os.makedirs(SAVE_DIR, exist_ok=True)
BEST_MODEL_PATH = os.path.join(SAVE_DIR, "best_style_model.keras")
FINAL_MODEL_PATH = os.path.join(SAVE_DIR, "final_style_model.keras")
INITIAL_WEIGHTS_PATH = os.path.join(SAVE_DIR, "best_style_model.keras")

def npz_generator(folder):
    """
    Generator that yields (images, labels) from each npz file.
    If a file has fewer than the expected BATCH_SIZE images, it yields a partial
    """
    files = sorted(glob.glob(os.path.join(folder, "*.npz")))
    for f in files:
        data = np.load(f)
        imgs = data['arr_0']
        labels = data['labels']
        current_batch_size = imgs.shape[0]
        if current_batch_size != BATCH_SIZE:
            print(f"Warning: File {f} has only {current_batch_size} images (expe
        yield imgs, labels

train_dataset = tf.data.Dataset.from_generator(
    lambda: npz_generator(TRAIN_FOLDER),
    output_types=(tf.float32, tf.int32),
    output_shapes=((None, IMG_SHAPE[0], IMG_SHAPE[1], IMG_SHAPE[2]), (None,))
)
val_dataset = tf.data.Dataset.from_generator(
    lambda: npz_generator(VAL_FOLDER),
    output_types=(tf.float32, tf.int32),
    output_shapes=((None, IMG_SHAPE[0], IMG_SHAPE[1], IMG_SHAPE[2]), (None,))
)

train_dataset = train_dataset.shuffle(100).unbatch().batch(NEW_BATCH_SIZE)
val_dataset = val_dataset.unbatch().batch(NEW_BATCH_SIZE)

num_train_files = len(glob.glob(os.path.join(TRAIN_FOLDER, "*.npz")))
num_val_files = len(glob.glob(os.path.join(VAL_FOLDER, "*.npz")))
steps_per_epoch = (num_train_files * BATCH_SIZE) // NEW_BATCH_SIZE
validation_steps = (num_val_files * BATCH_SIZE) // NEW_BATCH_SIZE

inputs = Input(shape=IMG_SHAPE)

if os.path.exists(INITIAL_WEIGHTS_PATH):
    print(f"Found pre-trained weights at {INITIAL_WEIGHTS_PATH}.")
    base_model = ResNet50(weights='imagenet', include_top=False, input_shape=IMG
else:
    print("No pre-trained model found; using fresh ResNet50 with ImageNet weight
    base_model = ResNet50(weights='imagenet', include_top=False, input_shape=IMG
```

```
x = base_model(inputs)
x = GlobalAveragePooling2D()(x)
x = BatchNormalization()(x)
x = Dropout(0.75)(x)
outputs = Dense(NUM_CLASSES, activation='softmax', kernel_regularizer=tf.keras.r

model = Model(inputs=inputs, outputs=outputs)

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.summary()

checkpoint_cb = ModelCheckpoint(
    BEST_MODEL_PATH,
    monitor='val_accuracy',
    save_best_only=True,
    mode='max',
    verbose=1
)
reduce_lr_cb = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=2, ver

history = model.fit(
    train_dataset,
    steps_per_epoch=steps_per_epoch,
    epochs=EPOCHS,
    validation_data=val_dataset,
    validation_steps=validation_steps,
    callbacks=[checkpoint_cb, reduce_lr_cb, TqdmCallback(verbose=1)]
)

model.save(FINAL_MODEL_PATH)
print(f"Final model saved to: {FINAL_MODEL_PATH}")
```

```
WARNING:tensorflow:From C:\Users\KIIT\AppData\Local\Temp\ipykernel_5692\215119679
3.py:39: calling DatasetV2.from_generator (from tensorflow.python.data.ops.datase
t_ops) with output_types is deprecated and will be removed in a future version.
Instructions for updating:
Use output_signature instead
WARNING:tensorflow:From C:\Users\KIIT\AppData\Local\Temp\ipykernel_5692\215119679
3.py:39: calling DatasetV2.from_generator (from tensorflow.python.data.ops.datase
t_ops) with output_shapes is deprecated and will be removed in a future version.
Instructions for updating:
Use output_signature instead
Found pre-trained weights at saved_models\best_style_model.keras.
Model: "functional"
```

Layer (type)	Output Shape	
input_layer (InputLayer)	(None, 224, 224, 3)	
resnet50 (Functional)	(None, 7, 7, 2048)	
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	
batch_normalization (BatchNormalization)	(None, 2048)	
dropout (Dropout)	(None, 2048)	
dense (Dense)	(None, 27)	

Total params: 23,651,227 (90.22 MB)

Trainable params: 23,594,011 (90.00 MB)

Non-trainable params: 57,216 (223.50 KB)

```
0epoch [00:00, ?epoch/s]
0batch [00:00, ?batch/s]
Epoch 1/3
Warning: File processed_batches_styles\batch_38.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles\batch_44.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles\batch_57.npz has only 25 images (expected
1000). Yielding partial batch.
571/580 ━━━━━━━━━━━━━━━━━━━━ 1:41 11s/step - accuracy: 0.2448 - loss: 7.9882
C:\Users\KIIT\AppData\Roaming\Python\Python312\site-packages\keras\src\trainers\e
poch_iterator.py:151: UserWarning: Your input ran out of data; interrupting train
ing. Make sure that your dataset or generator can generate at least `steps_per_ep
och * epochs` batches. You may need to use the `.repeat()` function when building
your dataset.
  self._interrupted_warning()
```

```
Warning: File processed_batches_styles_val\batch_24.npz has only 421 images (expe
cted 1000). Yielding partial batch.

Epoch 1: val_accuracy improved from -inf to 0.14905, saving model to saved_models
\best_style_model.keras
580/580 ━━━━━━━━━━━━━━━━━━━━ 7314s 12s/step - accuracy: 0.2460 - loss: 7.9637 - v
al_accuracy: 0.1491 - val_loss: 5.2471 - learning_rate: 1.0000e-04
Epoch 2/3
Warning: File processed_batches_styles\batch_38.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles\batch_44.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles\batch_57.npz has only 25 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles_val\batch_24.npz has only 421 images (expe
cted 1000). Yielding partial batch.

Epoch 2: val_accuracy improved from 0.14905 to 0.49380, saving model to saved_mod
els\best_style_model.keras
580/580 ━━━━━━━━━━━━━━━━━━━━ 7038s 12s/step - accuracy: 0.4579 - loss: 3.7593 - v
al_accuracy: 0.4938 - val_loss: 2.6024 - learning_rate: 1.0000e-04
Epoch 3/3
Warning: File processed_batches_styles\batch_38.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles\batch_44.npz has only 999 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles\batch_57.npz has only 25 images (expected
1000). Yielding partial batch.
Warning: File processed_batches_styles_val\batch_24.npz has only 421 images (expe
cted 1000). Yielding partial batch.

Epoch 3: val_accuracy improved from 0.49380 to 0.55866, saving model to saved_mod
els\best_style_model.keras
580/580 ━━━━━━━━━━━━━━━━━━━━ 7021s 12s/step - accuracy: 0.5765 - loss: 2.1271 - v
al_accuracy: 0.5587 - val_loss: 1.8417 - learning_rate: 1.0000e-04
Final model saved to: saved_models\final_style_model.keras
```

In [ ]:

In [ ]:

evaluation and outlier detection

In [8]:
```python
import os
import glob
import numpy as np
import tensorflow as tf
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.cluster import KMeans
from sklearn.ensemble import IsolationForest
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
import seaborn as sns

IMG_SHAPE = (224, 224, 3)
BATCH_SIZE = 1000
NEW_BATCH_SIZE = 64
ARTISTS_FOLDER = "processed_batches_artists_val"
```

```
GENRES_FOLDER = "processed_batches_genres_val"
STYLES_FOLDER = "processed_batches_styles_val"

def npz_generator(folder, batch_size=1000):
    """
    Generator that processes `.npz` files and yields batches of (images, labels)
    including partial batches when encountered.
    Args:
        folder: Path to the folder containing `.npz` files.
        batch_size: Expected number of images per `.npz` file.
    Yields:
        imgs: Batch of images (shape: [N, height, width, channels]).
        labels: Batch of corresponding labels (shape: [N]).
    """
    files = sorted(glob.glob(os.path.join(folder, "*.npz")))
    for f in files:
        data = np.load(f)
        imgs = data['arr_0']
        labels = data['labels']
        current_batch_size = imgs.shape[0]

        if current_batch_size < batch_size:
            print(f"Warning: File {f} has only {current_batch_size} images (expe

        yield imgs, labels

def create_dataset(folder):
    """
    Create a TensorFlow dataset from the given folder.
    Args:
        folder: Path to the folder containing `.npz` files.
    Returns:
        tf.data.Dataset object
    """
    dataset = tf.data.Dataset.from_generator(
        lambda: npz_generator(folder),
        output_types=(tf.float32, tf.int32),
        output_shapes=((None, IMG_SHAPE[0], IMG_SHAPE[1], IMG_SHAPE[2]), (None,)
    )
    dataset = dataset.unbatch().batch(NEW_BATCH_SIZE)  # Unbatch and re-batch to
    dataset = dataset.prefetch(tf.data.experimental.AUTOTUNE)  # Prefetch data f
    return dataset

artists_val_dataset = create_dataset(ARTISTS_FOLDER)
genres_val_dataset = create_dataset(GENRES_FOLDER)
styles_val_dataset = create_dataset(STYLES_FOLDER)

def extract_embeddings(model, dataset):
    """
    Extract embeddings (penultimate layer outputs) from the given dataset using
    Args:
        model: The trained Keras model.
        dataset: Dataset to extract embeddings from.
    Returns:
        embeddings: Feature embeddings.
        labels: Corresponding labels.
    """
    embeddings = []
    labels = []
    for batch_imgs, batch_labels in dataset:
```

```
        batch_embeddings = model.predict(batch_imgs)
        embeddings.append(batch_embeddings)
        labels.append(batch_labels.numpy())
    embeddings = np.concatenate(embeddings, axis=0)
    labels = np.concatenate(labels, axis=0)
    return embeddings, labels

def evaluate_and_detect_outliers(model, dataset, task_name, num_classes):
    """
    Perform classification evaluation and outlier detection for a specific task.
    Args:
        model: Trained Keras model.
        dataset: Validation dataset.
        task_name: Name of the task (e.g., 'Artists', 'Genres', 'Styles').
        num_classes: Number of unique classes for the task.
    """
    print(f"\n--- Task: {task_name} ---")

    embeddings, labels = extract_embeddings(model, dataset)
    preds = model.predict(dataset.map(lambda x, y: x))  # Only feed images to th
    pred_classes = np.argmax(preds, axis=1)

    print("\nClassification Report:")
    print(classification_report(labels, pred_classes))

    print("\nConfusion Matrix:")
    cm = confusion_matrix(labels, pred_classes)
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
    plt.xlabel("Predicted")
    plt.ylabel("True")
    plt.title(f"Confusion Matrix for {task_name}")
    plt.show()

    print("\nOutlier Detection:")

    # Option : distance from Centroids
    class_centroids = {}
    for c in np.unique(labels):
        class_embeddings = embeddings[labels == c]
        class_centroids[c] = np.mean(class_embeddings, axis=0)

    distances = [np.linalg.norm(embeddings[i] - class_centroids[labels[i]]) for
    outlier_threshold = np.percentile(distances, 95)
    outliers = np.where(distances > outlier_threshold)
    print(f"Outliers based on distances from centroids: {outliers}")

    # Option : clustering (k-means)
    kmeans = KMeans(n_clusters=num_classes).fit(embeddings)
    cluster_distances = kmeans.transform(embeddings)
    outliers_kmeans = np.where(np.min(cluster_distances, axis=1) > np.percentile
    print(f"Outliers based on k-means clustering: {outliers_kmeans}")

    # Option : isolation Forest
    iso_forest = IsolationForest(contamination=0.05)
    outlier_predictions = iso_forest.fit_predict(embeddings)
    outliers_isolation_forest = np.where(outlier_predictions == -1)
    print(f"Outliers detected by Isolation Forest: {outliers_isolation_forest}")

    # Visualization of Outliers
    tsne = TSNE(n_components=2, random_state=42)
```

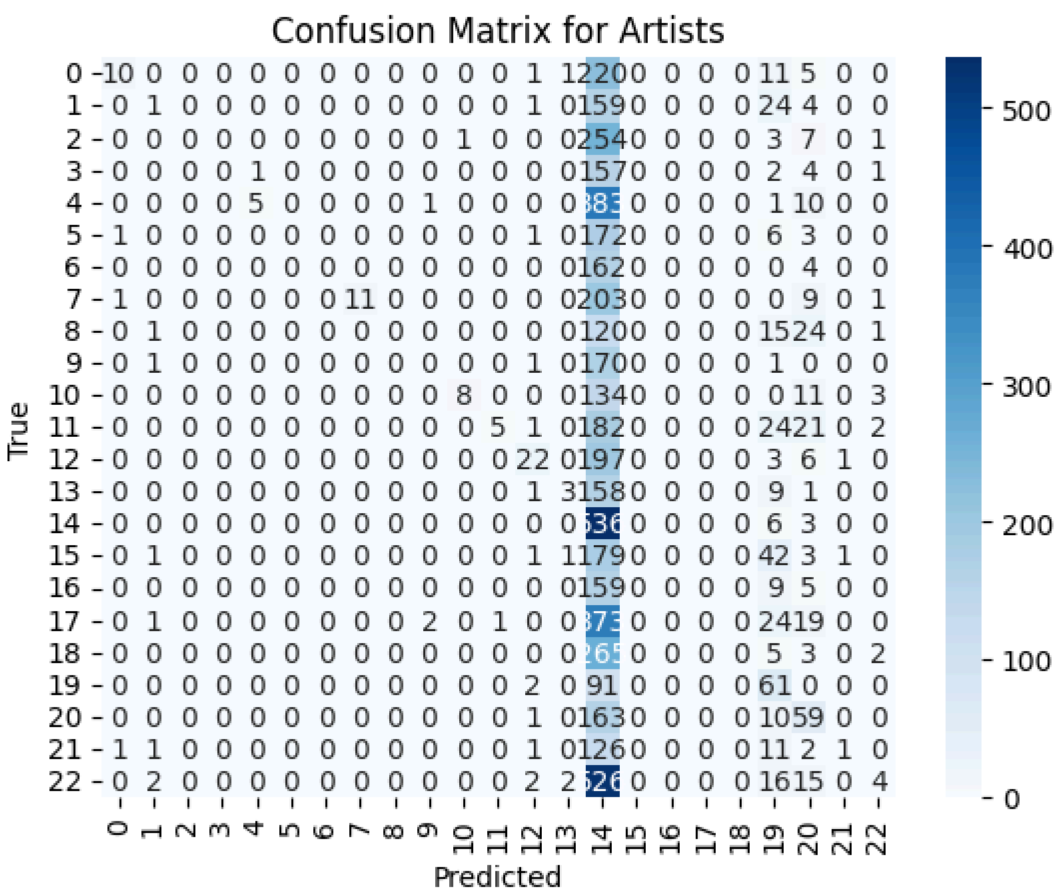
```
    reduced_embeddings = tsne.fit_transform(embeddings)

    plt.figure(figsize=(10, 8))
    plt.scatter(reduced_embeddings[:, 0], reduced_embeddings[:, 1], c=labels, cm
    plt.scatter(reduced_embeddings[outliers, 0], reduced_embeddings[outliers, 1]
    plt.scatter(reduced_embeddings[outliers_kmeans, 0], reduced_embeddings[outli
    plt.scatter(reduced_embeddings[outliers_isolation_forest, 0], reduced_embedd
    plt.legend()
    plt.title(f"t-SNE Visualization of Embeddings with Outliers for {task_name}"
    plt.show()

artists_model = tf.keras.models.load_model("./saved_models/best_artist_model.ker
genres_model = tf.keras.models.load_model("./saved_models/best_genre_model.keras
styles_model = tf.keras.models.load_model("./saved_models/best_style_model.keras


tasks = [
    {
        "name": "Artists",
        "dataset": artists_val_dataset,
        "num_classes": 23,
        "model": artists_model
    },
    {
        "name": "Genres",
        "dataset": genres_val_dataset,
        "num_classes": 10,
        "model": genres_model
    },
    {
        "name": "Styles",
        "dataset": styles_val_dataset,
        "num_classes": 27,
        "model": styles_model
    }
]

for task in tasks:
    evaluate_and_detect_outliers(
        model=task["model"],
        dataset=task["dataset"],
        task_name=task["name"],
        num_classes=task["num_classes"]
    )
```

```
--- Task: Artists ---
2/2 ━━━━━━━━━━━━━━━━━━━━ 4s 974ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 973ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 976ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 1s/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 946ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 902ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 939ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 963ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 959ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 970ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 966ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 964ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 931ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 914ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 978ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 815ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 793ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 838ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 834ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 831ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 851ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 828ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 828ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 868ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 850ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 886ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 878ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 853ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 879ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 950ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 903ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 799ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 819ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 781ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 775ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 820ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 801ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 789ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 768ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 823ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 822ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 879ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 918ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 893ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 816ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 796ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 793ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 794ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 798ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 786ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 815ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 848ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 872ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 867ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 834ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 842ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 842ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 841ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 889ms/step
```

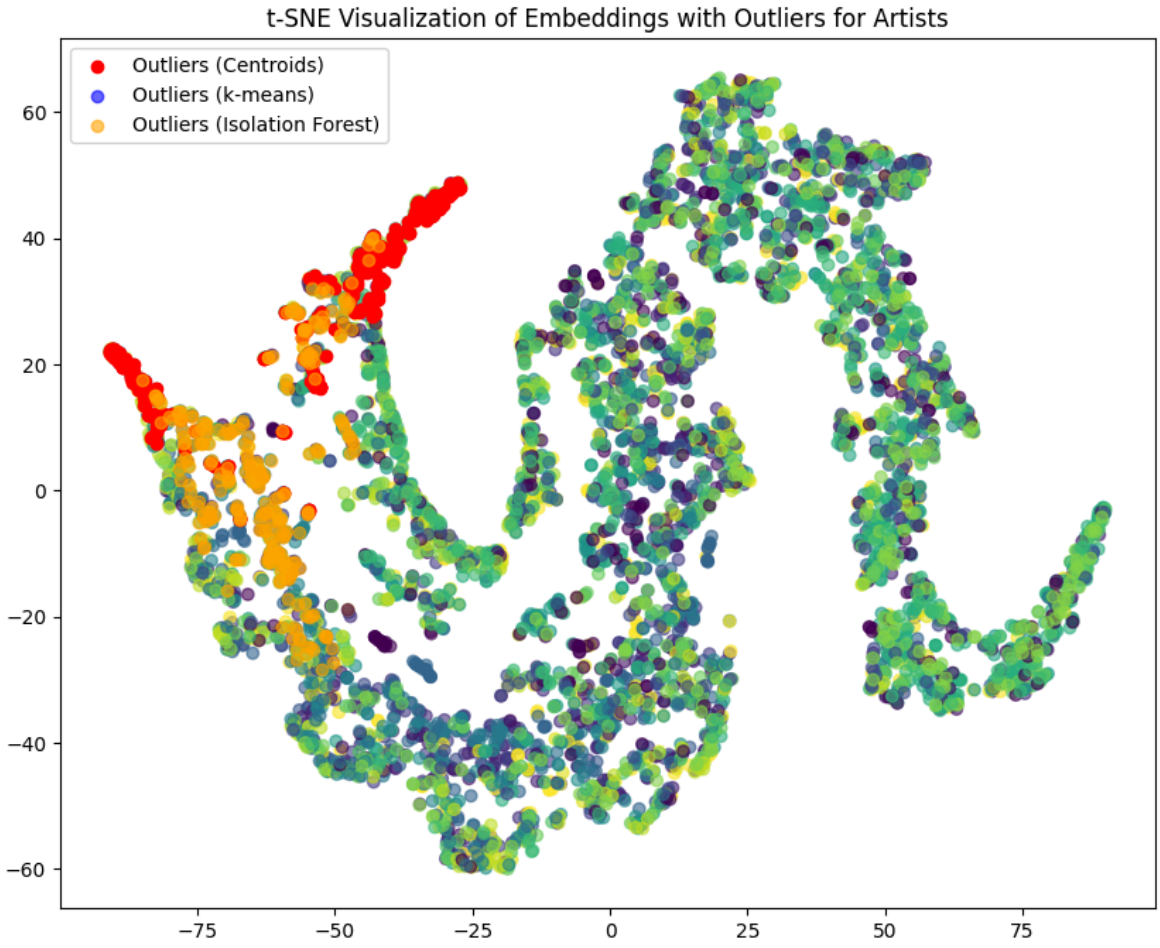
```
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 949ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 913ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 822ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 810ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 824ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 819ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 807ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 817ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 866ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 806ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 812ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 801ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 885ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 838ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 808ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 948ms/step
Warning: File processed_batches_artists_val\batch_5.npz has only 706 images (expe
cted 1000). Yielding partial batch.
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 919ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 801ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 840ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 825ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 849ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 791ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 824ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 855ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 822ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 812ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 816ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 798ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 809ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 807ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
Warning: File processed_batches_artists_val\batch_5.npz has only 706 images (expe
cted 1000). Yielding partial batch.
90/90 ━━━━━━━━━━━━━━━━━━━━ 140s 2s/step

Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.04      0.08       248
           1       0.12      0.01      0.01       189
           2       0.00      0.00      0.00       266
           3       0.00      0.00      0.00       165
           4       0.83      0.01      0.02       400
           5       0.00      0.00      0.00       183
           6       0.00      0.00      0.00       166
           7       1.00      0.05      0.09       225
           8       0.00      0.00      0.00       161
           9       0.00      0.00      0.00       173
          10       0.89      0.05      0.10       156
          11       0.83      0.02      0.04       235
          12       0.63      0.10      0.17       229
          13       0.43      0.02      0.03       172
          14       0.11      0.98      0.19       545
          15       0.00      0.00      0.00       228
          16       0.00      0.00      0.00       173
          17       0.00      0.00      0.00       420
          18       0.00      0.00      0.00       275
          19       0.22      0.40      0.28       154
```

```
          20       0.27      0.25      0.26       233
          21       0.33      0.01      0.01       143
          22       0.27      0.01      0.01       567

    accuracy                           0.13      5706
   macro avg       0.29      0.08      0.06      5706
weighted avg       0.29      0.13      0.06      5706


Confusion Matrix:
```

```
C:\Users\KIIT\AppData\Roaming\Python\Python312\site-packages\keras\src\trainers\epoch_iterator.py:151: UserWarning: Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batches. You may need to use the `.repeat()` function when building your dataset.
  self._interrupted_warning()
C:\Users\KIIT\AppData\Roaming\Python\Python312\site-packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\KIIT\AppData\Roaming\Python\Python312\site-packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\KIIT\AppData\Roaming\Python\Python312\site-packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```



```
Outlier Detection:
Outliers based on distances from centroids: (array([  23,  184,  204,  230,  266,
327,  355,  385,  414,  455,  550,
        605,  648,  774,  866,  886,  919,  960,  992,  997, 1029, 1072,
       1101, 1103, 1174, 1175, 1180, 1181, 1189, 1202, 1225, 1254, 1260,
       1282, 1294, 1307, 1372, 1396, 1400, 1404, 1433, 1438, 1457, 1521,
       1550, 1558, 1684, 1703, 1820, 1838, 1862, 1884, 1889, 1932, 1937,
       2037, 2126, 2144, 2149, 2155, 2172, 2173, 2182, 2200, 2209, 2255,
       2270, 2285, 2300, 2314, 2322, 2323, 2325, 2326, 2334, 2351, 2382,
       2383, 2389, 2392, 2396, 2405, 2408, 2413, 2425, 2457, 2491, 2496,
       2502, 2521, 2615, 2618, 2622, 2635, 2645, 2651, 2669, 2674, 2702,
       2710, 2758, 2762, 2764, 2777, 2786, 2789, 2795, 2801, 2839, 2841,
       2856, 2867, 2872, 2889, 2897, 2907, 2937, 3044, 3081, 3179, 3241,
       3253, 3420, 3451, 3452, 3458, 3459, 3463, 3466, 3478, 3486, 3492,
       3515, 3522, 3537, 3558, 3560, 3572, 3576, 3617, 3619, 3623, 3675,
       3706, 3712, 3722, 3747, 3753, 3759, 3767, 3776, 3777, 3857, 3899,
       3900, 3964, 4001, 4009, 4011, 4018, 4019, 4021, 4023, 4025, 4026,
       4045, 4048, 4134, 4138, 4139, 4150, 4152, 4191, 4204, 4206, 4214,
       4229, 4235, 4240, 4242, 4246, 4265, 4275, 4277, 4278, 4294, 4305,
       4321, 4325, 4327, 4339, 4355, 4359, 4365, 4383, 4396, 4408, 4425,
       4436, 4440, 4465, 4518, 4524, 4525, 4531, 4532, 4548, 4614, 4648,
       4676, 4684, 4691, 4708, 4738, 4762, 4764, 4776, 4780, 4802, 4817,
       4841, 4860, 4862, 4881, 4887, 4895, 4904, 4919, 4922, 4926, 4942,
       4945, 4953, 4978, 4980, 4982, 4984, 5019, 5027, 5036, 5040, 5041,
       5061, 5062, 5112, 5136, 5148, 5195, 5303, 5338, 5354, 5382, 5384,
       5398, 5477, 5498, 5514, 5522, 5527, 5535, 5548, 5551, 5563, 5573,
       5576, 5579, 5584, 5585, 5594, 5601, 5602, 5607, 5617, 5624, 5629,
       5636, 5640, 5657, 5666, 5669, 5675, 5676, 5682, 5684, 5697, 5705],
      dtype=int64),)
Outliers based on k-means clustering: (array([], dtype=int64),)
Outliers detected by Isolation Forest: (array([  11,   14,   26,   32,   35,   3
9,   45,   52,   73,  106,  111,
        118,  122,  129,  145,  170,  184,  190,  204,  210,  219,  228,
        231,  265,  294,  306,  307,  312,  314,  337,  353,  357,  362,
        389,  421,  428,  442,  446,  462,  478,  495,  516,  527,  539,
        554,  566,  574,  612,  690,  691,  733,  774,  789,  795,  817,
        825,  853,  855,  859,  866,  885,  915,  950,  960, 1005, 1026,
       1072, 1100, 1112, 1174, 1180, 1226, 1236, 1252, 1276, 1286, 1334,
       1344, 1386, 1431, 1448, 1469, 1483, 1528, 1585, 1628, 1643, 1651,
       1659, 1684, 1687, 1693, 1703, 1801, 1802, 1827, 1846, 1862, 1872,
       1879, 1887, 1891, 1930, 1933, 1944, 1980, 2127, 2144, 2149, 2167,
       2182, 2251, 2253, 2322, 2361, 2390, 2502, 2521, 2643, 2667, 2672,
       2680, 2681, 2693, 2698, 2714, 2736, 2750, 2751, 2753, 2755, 2756,
       2758, 2767, 2794, 2803, 2805, 2813, 2817, 2820, 2826, 2829, 2838,
       2841, 2843, 2844, 2850, 2852, 2861, 2862, 2865, 2871, 2876, 2880,
       2883, 2887, 3300, 3459, 3473, 3476, 3478, 3486, 3502, 3557, 3560,
       3583, 3595, 3631, 3639, 3654, 3663, 3685, 3700, 3746, 3753, 3769,
       3806, 3813, 3829, 3833, 3881, 3927, 3931, 3936, 3960, 3970, 3984,
       3989, 4000, 4004, 4018, 4031, 4035, 4038, 4047, 4053, 4084, 4087,
       4100, 4107, 4108, 4110, 4114, 4122, 4129, 4143, 4148, 4160, 4179,
       4192, 4193, 4194, 4231, 4247, 4260, 4262, 4280, 4282, 4314, 4324,
       4356, 4360, 4372, 4401, 4429, 4441, 4456, 4470, 4479, 4481, 4483,
       4492, 4511, 4516, 4519, 4520, 4521, 4527, 4531, 4537, 4548, 4550,
       4551, 4554, 4558, 4564, 4570, 4578, 4596, 4603, 4604, 4609, 4678,
       4710, 4776, 4802, 4841, 4871, 4877, 4907, 4913, 4929, 4930, 4933,
       4953, 4964, 4978, 4997, 5010, 5012, 5017, 5018, 5027, 5040, 5041,
       5240, 5255, 5303, 5324, 5446, 5478, 5624, 5625, 5635, 5671, 5693],
      dtype=int64),)
```


t-SNE Visualization of Embeddings with Outliers for Artists
Outliers (Centroids)
Outliers (k-means)
Outliers (Isolation Forest)
60
40
20
0
−20
−40
−60
−75
−50
−25
0
25
50
75

```
--- Task: Genres ---
2/2 ━━━━━━━━━━━━━━━━━━━━ 4s 805ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 815ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 816ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 812ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 794ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 800ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 802ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 812ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 779ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 806ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 812ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 765ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 785ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 774ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 833ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 778ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 771ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 796ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 785ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 799ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 786ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 783ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 785ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 786ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 799ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 792ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 786ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 857ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 865ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 779ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 796ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 793ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 786ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 789ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 792ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 779ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 813ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 802ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 810ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 833ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 857ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 841ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 832ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 772ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 854ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 809ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 801ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 788ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 826ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 838ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 830ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 834ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 830ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 836ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 827ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 834ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 843ms/step
```

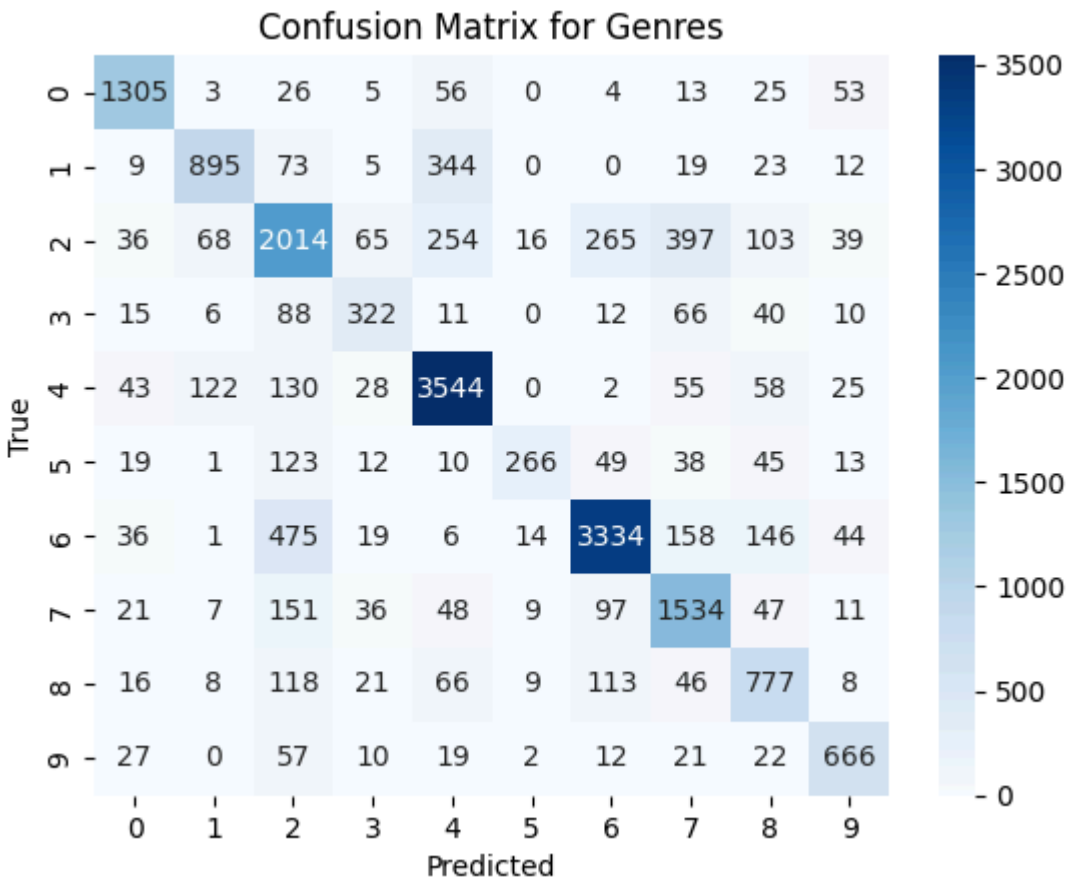
```
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 855ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 932ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 835ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 835ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 794ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 807ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 785ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 775ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 822ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 783ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 779ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 776ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 768ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 765ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 770ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 839ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 844ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 768ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 767ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 806ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 804ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 822ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 814ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 821ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 777ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 779ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 829ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 817ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 799ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 781ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 834ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 824ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 795ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 769ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 761ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 767ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 765ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 762ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 774ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 798ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 789ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 834ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 829ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 825ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 792ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 788ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 836ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 859ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 857ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 781ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 802ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 791ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 774ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 762ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 759ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 771ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 784ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 778ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 781ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 768ms/step
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 753ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 767ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 810ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 834ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 789ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 762ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 756ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 767ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 756ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 751ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 770ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 758ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 774ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 766ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 767ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 764ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 799ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 822ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 831ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 810ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 780ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 767ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 796ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 773ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 773ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 819ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 846ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 839ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 860ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 835ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 815ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 784ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 782ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 829ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 877ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 857ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 791ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 791ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 809ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 814ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 799ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 815ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 783ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 775ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 771ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 794ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 800ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 825ms/step
Warning: File processed_batches_genres_val\batch_19.npz has only 492 images (expe
cted 1000). Yielding partial batch.
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 800ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 792ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 763ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 785ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 781ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 770ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 769ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 822ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 846ms/step
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 799ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 789ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 799ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 803ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 782ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 782ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 779ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 806ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 801ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 786ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 770ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 795ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 759ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 773ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 809ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 842ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 808ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 780ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 769ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 791ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 777ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 766ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 774ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 791ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 766ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 782ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 781ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 779ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 768ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 829ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 818ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 831ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 777ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 773ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 772ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 783ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 759ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 772ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 777ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 775ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 786ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 780ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 779ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 781ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 782ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 825ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 848ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 852ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 806ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 811ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 793ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 796ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 789ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 811ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 822ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 765ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 781ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 779ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 776ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 773ms/step
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 784ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 835ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 837ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 832ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 780ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 774ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 784ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 788ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 777ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 805ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 779ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 802ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 806ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 784ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 816ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 828ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 859ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 855ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 804ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 785ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 815ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 797ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 803ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 787ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 797ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 787ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 791ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 762ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 780ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 774ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 771ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 819ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 832ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 817ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 777ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 781ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 778ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 774ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 784ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 774ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 778ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 784ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 764ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 787ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 776ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 778ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 820ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 848ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 805ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 773ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 784ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 780ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 767ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 784ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 769ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 788ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 778ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 777ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 773ms/step
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 778ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 783ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 766ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 784ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 776ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 780ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 779ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 3s 2s/step
Warning: File processed_batches_genres_val\batch_19.npz has only 492 images (expe
cted 1000). Yielding partial batch.
305/305 ━━━━━━━━━━━━━━━━━━━━ 468s 2s/step

Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.88      0.87      1490
           1       0.81      0.65      0.72      1380
           2       0.62      0.62      0.62      3257
           3       0.62      0.56      0.59       570
           4       0.81      0.88      0.85      4007
           5       0.84      0.46      0.60       576
           6       0.86      0.79      0.82      4233
           7       0.65      0.78      0.71      1961
           8       0.60      0.66      0.63      1182
           9       0.76      0.80      0.78       836

    accuracy                           0.75     19492
   macro avg       0.74      0.71      0.72     19492
weighted avg       0.76      0.75      0.75     19492


Confusion Matrix:
C:\Users\KIIT\AppData\Roaming\Python\Python312\site-packages\keras\src\trainers\e
poch_iterator.py:151: UserWarning: Your input ran out of data; interrupting train
ing. Make sure that your dataset or generator can generate at least `steps_per_ep
och * epochs` batches. You may need to use the `.repeat()` function when building
your dataset.
  self._interrupted_warning()
```

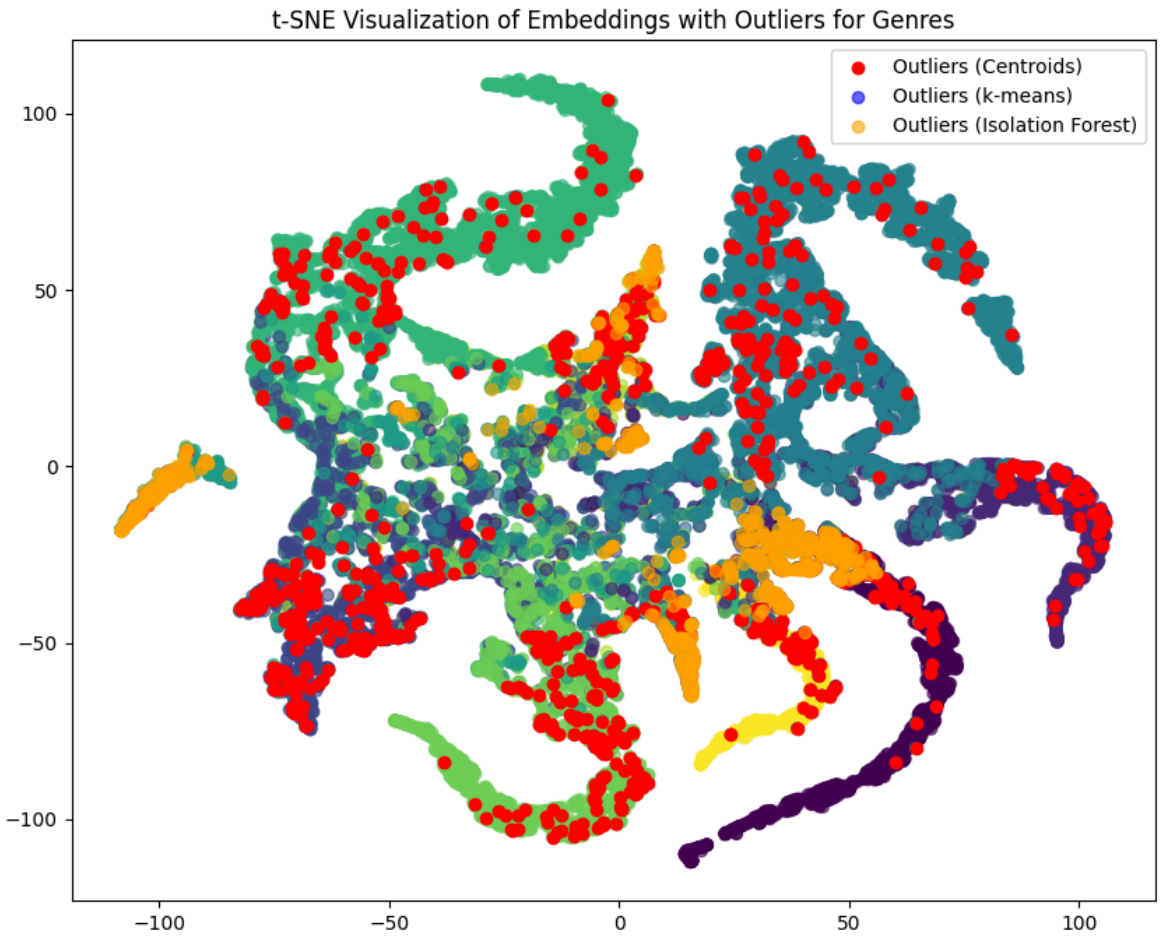

Confusion Matrix for Genres
True
Predicted
0 1305 3 26 5 56 0 4 13 25 53
1 9 895 73 5 344 0 0 19 23 12
2 36 68 2014 65 254 16 265 397 103 39
3 15 6 88 322 11 0 12 66 40 10
4 43 122 130 28 3544 0 2 55 58 25
5 19 1 123 12 10 266 49 38 45 13
6 36 1 475 19 6 14 3334 158 146 44
7 21 7 151 36 48 9 97 1534 47 11
8 16 8 118 21 66 9 113 46 777 8
9 27 0 57 10 19 2 12 21 22 666
0 1 2 3 4 5 6 7 8 9
3500 3000 2500 2000 1500 1000 500 0

```
Outlier Detection:
Outliers based on distances from centroids: (array([   40,    41,    57,    86,
112,   127,   132,   134,   139,
         149,   154,   170,   181,   189,   237,   253,   274,   281,
         285,   302,   303,   336,   357,   414,   429,   436,   462,
         464,   475,   486,   488,   494,   498,   505,   506,   554,
         560,   590,   629,   668,   674,   675,   701,   720,   734,
         739,   777,   794,   804,   811,   871,   892,   898,   917,
         923,   961,   982,   995,  1016,  1022,  1051,  1054,  1065,
        1083,  1098,  1133,  1134,  1142,  1144,  1145,  1147,  1149,
        1151,  1163,  1182,  1184,  1189,  1222,  1242,  1243,  1262,
        1324,  1339,  1344,  1352,  1356,  1359,  1369,  1379,  1400,
        1429,  1452,  1496,  1500,  1575,  1586,  1591,  1598,  1630,
        1669,  1685,  1709,  1733,  1749,  1750,  1786,  1805,  1832,
        1837,  1843,  1844,  1849,  1890,  1903,  1918,  1990,  1993,
        2001,  2026,  2047,  2063,  2078,  2091,  2094,  2115,  2126,
        2188,  2197,  2231,  2232,  2240,  2408,  2433,  2467,  2497,
        2499,  2530,  2564,  2570,  2616,  2624,  2643,  2711,  2721,
        2735,  2751,  2781,  2811,  2832,  2861,  2900,  3073,  3094,
        3160,  3180,  3184,  3187,  3211,  3225,  3230,  3241,  3306,
        3313,  3327,  3331,  3361,  3366,  3398,  3423,  3465,  3466,
        3477,  3502,  3503,  3510,  3529,  3560,  3563,  3570,  3603,
        3653,  3682,  3692,  3719,  3727,  3750,  3761,  3781,  3809,
        3832,  3842,  3860,  3901,  3911,  3918,  4019,  4050,  4064,
        4077,  4080,  4082,  4092,  4104,  4106,  4127,  4128,  4137,
        4145,  4150,  4158,  4166,  4167,  4169,  4170,  4213,  4220,
        4222,  4227,  4243,  4267,  4276,  4288,  4290,  4348,  4357,
        4379,  4383,  4387,  4399,  4421,  4427,  4469,  4490,  4500,
        4519,  4520,  4534,  4562,  4603,  4609,  4631,  4638,  4654,
        4666,  4669,  4680,  4723,  4737,  4762,  4768,  4771,  4772,
        4778,  4779,  4808,  4822,  4865,  4881,  4915,  5014,  5015,
        5032,  5043,  5067,  5080,  5107,  5126,  5148,  5163,  5164,
        5201,  5222,  5289,  5307,  5340,  5358,  5365,  5375,  5390,
        5451,  5512,  5528,  5542,  5555,  5607,  5616,  5660,  5683,
        5692,  5722,  5753,  5759,  5767,  5769,  5834,  5869,  5885,
        5890,  5894,  5905,  5916,  5917,  5927,  5947,  5989,  5994,
        6013,  6023,  6037,  6048,  6070,  6077,  6079,  6114,  6125,
        6135,  6138,  6155,  6189,  6192,  6194,  6209,  6233,  6240,
        6254,  6283,  6288,  6321,  6340,  6368,  6369,  6401,  6406,
        6430,  6442,  6452,  6456,  6470,  6472,  6484,  6550,  6612,
        6614,  6637,  6645,  6685,  6718,  6723,  6736,  6737,  6766,
        6767,  6781,  6784,  6831,  6846,  6867,  6868,  6888,  6896,
        6902,  6930,  6932,  6943,  6948,  6965,  6987,  7011,  7021,
        7027,  7051,  7079,  7093,  7134,  7157,  7172,  7186,  7210,
        7216,  7217,  7222,  7246,  7288,  7316,  7331,  7341,  7359,
        7383,  7394,  7404,  7411,  7444,  7463,  7470,  7481,  7486,
        7492,  7493,  7533,  7561,  7566,  7581,  7602,  7609,  7613,
        7635,  7649,  7659,  7685,  7690,  7700,  7702,  7707,  7720,
        7728,  7734,  7788,  7790,  7793,  7807,  7835,  7861,  7870,
        7883,  7891,  7927,  7971,  7984,  8002,  8011,  8015,  8031,
        8056,  8065,  8068,  8071,  8081,  8101,  8137,  8149,  8155,
        8156,  8166,  8182,  8217,  8222,  8228,  8229,  8232,  8257,
        8260,  8267,  8272,  8290,  8304,  8306,  8323,  8325,  8358,
        8359,  8525,  8540,  8559,  8560,  8568,  8607,  8635,  8640,
        8667,  8676,  8684,  8726,  8732,  8738,  8748,  8759,  8799,
        8828,  8836,  8848,  8851,  8866,  8883,  8906,  8910,  8916,
        8929,  8945,  8949,  8957,  8961,  8984,  9007,  9014,  9039,
        9076,  9092,  9128,  9145,  9165,  9167,  9170,  9202,  9216,
        9257,  9275,  9284,  9311,  9313,  9363,  9375,  9379,  9394,
        9399,  9401,  9415,  9427,  9437,  9456,  9504,  9517,  9537,
```

```
 9597,  9626,  9646,  9649,  9677,  9690,  9762,  9769,  9790,
 9809,  9813,  9820,  9821,  9833,  9842,  9843,  9846,  9850,
 9853,  9861,  9879,  9894,  9901,  9914,  9925,  9927,  9960,
 9970,  9986, 10039, 10041, 10056, 10063, 10117, 10122, 10158,
10163, 10183, 10184, 10189, 10190, 10205, 10210, 10220, 10313,
10329, 10425, 10437, 10464, 10467, 10479, 10549, 10553, 10613,
10641, 10652, 10658, 10677, 10684, 10692, 10695, 10705, 10707,
10712, 10729, 10737, 10776, 10787, 10794, 10801, 10814, 10823,
10842, 10891, 10897, 10924, 10939, 10971, 11003, 11015, 11023,
11048, 11054, 11074, 11126, 11130, 11139, 11188, 11198, 11230,
11235, 11244, 11263, 11273, 11284, 11302, 11337, 11387, 11393,
11442, 11444, 11486, 11496, 11500, 11503, 11510, 11512, 11515,
11521, 11541, 11560, 11568, 11586, 11595, 11599, 11600, 11625,
11665, 11677, 11679, 11681, 11683, 11716, 11737, 11753, 11755,
11775, 11776, 11788, 11824, 11827, 11890, 11898, 11903, 11904,
11914, 11935, 11947, 11951, 11954, 11956, 11976, 12048, 12071,
12086, 12088, 12090, 12095, 12125, 12154, 12182, 12197, 12207,
12292, 12354, 12384, 12407, 12425, 12444, 12449, 12450, 12481,
12502, 12514, 12541, 12551, 12552, 12596, 12606, 12638, 12647,
12664, 12704, 12708, 12713, 12721, 12773, 12798, 12805, 12806,
12821, 12928, 12960, 13008, 13040, 13042, 13068, 13069, 13083,
13097, 13137, 13138, 13154, 13164, 13178, 13182, 13200, 13232,
13233, 13245, 13257, 13265, 13273, 13299, 13332, 13335, 13366,
13396, 13399, 13407, 13411, 13442, 13461, 13479, 13530, 13543,
13561, 13594, 13597, 13606, 13631, 13652, 13668, 13671, 13676,
13697, 13714, 13729, 13781, 13791, 13818, 13823, 13831, 13867,
13872, 13882, 13890, 13901, 13916, 13971, 14000, 14032, 14065,
14068, 14070, 14079, 14111, 14128, 14129, 14174, 14211, 14213,
14215, 14253, 14282, 14298, 14300, 14365, 14368, 14379, 14381,
14417, 14441, 14487, 14494, 14495, 14506, 14525, 14529, 14535,
14541, 14580, 14597, 14610, 14638, 14641, 14644, 14645, 14664,
14686, 14693, 14702, 14713, 14715, 14732, 14741, 14818, 14830,
14868, 14874, 14875, 14916, 14943, 14944, 14992, 15003, 15037,
15065, 15096, 15128, 15151, 15185, 15202, 15240, 15245, 15246,
15267, 15275, 15401, 15414, 15422, 15443, 15452, 15502, 15577,
15582, 15688, 15695, 15850, 15989, 16020, 16042, 16048, 16237,
16342, 16389, 16427, 16432, 16455, 16501, 16594, 16605, 16638,
16645, 16675, 16687, 16732, 16756, 16758, 16764, 16771, 16809,
16831, 16858, 16861, 16913, 16970, 16974, 16983, 16984, 16992,
16999, 17009, 17087, 17104, 17131, 17144, 17154, 17182, 17226,
17233, 17242, 17262, 17267, 17279, 17344, 17369, 17384, 17385,
17401, 17418, 17451, 17486, 17528, 17546, 17612, 17631, 17665,
17671, 17681, 17687, 17688, 17726, 17779, 17802, 17803, 17810,
17853, 17867, 17891, 18015, 18024, 18037, 18047, 18056, 18085,
18102, 18134, 18154, 18191, 18199, 18227, 18283, 18288, 18307,
18327, 18348, 18356, 18362, 18391, 18408, 18439, 18458, 18496,
18516, 18528, 18557, 18579, 18589, 18591, 18592, 18643, 18653,
18654, 18714, 18772, 18834, 18859, 18949, 18960, 18963, 18991,
19009, 19032, 19106, 19110, 19140, 19182, 19225, 19284, 19285,
19310, 19337, 19358, 19369, 19373, 19380, 19385, 19395, 19458,
19466, 19471, 19478], dtype=int64),)
Outliers based on k-means clustering: (array([], dtype=int64),)
Outliers detected by Isolation Forest: (array([   83,    93,   149,   150,   171,
235,   323,   393,   446,
  477,   486,   627,   705,   710,   777,   854,   928,   966,
  993,  1111,  1134,  1297,  1317,  1322,  1324,  1446,  1461,
 1575,  1599,  1634,  1737,  1911,  1918,  1922,  1997,  2188,
 2240,  2331,  2832,  2968,  3146,  3193,  3194,  3206,  3209,
 3211,  3241,  3246,  3247,  3265,  3350,  3358,  3387,  3442,
 3450,  3458,  3480,  3481,  3486,  3492,  3505,  3512,  3542,
```

```
 3545,  3563,  3580,  3586,  3593,  3601,  3606,  3623,  3626,
 3632,  3633,  3640,  3670,  3693,  3713,  3740,  3779,  3781,
 3790,  3798,  3799,  3832,  3847,  3856,  3868,  3874,  3879,
 3889,  3894,  3919,  3922,  3925,  3942,  3987,  3988,  4014,
 4019,  4023,  4027,  4028,  4031,  4038,  4055,  4062,  4075,
 4104,  4105,  4120,  4136,  4139,  4187,  4202,  4236,  4244,
 4263,  4280,  4289,  4295,  4315,  4319,  4321,  4326,  4334,
 4340,  4368,  4374,  4386,  4389,  4397,  4404,  4411,  4417,
 4426,  4450,  4453,  4462,  4471,  4488,  4504,  4532,  4555,
 4570,  4575,  4584,  4588,  4602,  4608,  4612,  4628,  4675,
 4708,  4713,  4719,  4721,  4736,  4738,  4758,  4759,  4760,
 4777,  4791,  4795,  4800,  4828,  4837,  4846,  4857,  4868,
 4878,  4879,  4884,  4890,  4896,  4906,  4912,  4943,  4951,
 4952,  4970,  4984,  4999,  5008,  5049,  5061,  5072,  5073,
 5075,  5076,  5077,  5079,  5081,  5083,  5085,  5106,  5115,
 5122,  5133,  5140,  5141,  5145,  5147,  5196,  5204,  5210,
 5212,  5213,  5214,  5215,  5217,  5218,  5220,  5222,  5223,
 5225,  5230,  5235,  5238,  5243,  5244,  5248,  5251,  5253,
 5255,  5258,  5259,  5261,  5262,  5267,  5269,  5270,  5271,
 5274,  5280,  5284,  5300,  5301,  5303,  5309,  5310,  5312,
 5313,  5314,  5316,  5317,  5320,  5324,  5325,  5326,  5327,
 5331,  5333,  5338,  5339,  5342,  5343,  5344,  5345,  5348,
 5361,  5362,  5363,  5367,  5368,  5369,  5377,  5380,  5381,
 5384,  5385,  5386,  5387,  5388,  5389,  5391,  5398,  5404,
 5406,  5407,  5408,  5410,  5412,  5415,  5416,  5420,  5427,
 5429,  5433,  5437,  5441,  5444,  5445,  5446,  5450,  5452,
 5456,  5458,  5459,  5466,  5469,  5470,  5472,  5473,  5477,
 5478,  5479,  5480,  5484,  5485,  5488,  5490,  5491,  5496,
 5497,  5503,  5505,  5511,  5513,  5515,  5516,  5518,  5534,
 5538,  5540,  5543,  5545,  5548,  5552,  5557,  5558,  5559,
 5560,  5564,  5565,  5568,  5569,  5570,  5592,  5593,  5594,
 5595,  5596,  5598,  5601,  5603,  5609,  5610,  5617,  5621,
 5622,  5628,  5632,  5633,  5636,  5637,  5643,  5645,  5648,
 5650,  5652,  5657,  5658,  5659,  5666,  5669,  5670,  5673,
 5678,  5681,  5685,  5686,  5693,  5697,  5709,  5710,  5712,
 5717,  5720,  5721,  5724,  5737,  5738,  5740,  5742,  5747,
 5750,  5752,  5768,  5854,  5966,  6000,  6192,  6231,  6508,
 6515,  6653,  6736,  6776,  6943,  6948,  6949,  7055,  7088,
 7123,  7194,  7217,  7250,  7330,  7400,  7720,  7744,  7860,
 7984,  8015,  8156,  8266,  8358,  8437,  8559,  8667,  8759,
 8762,  8800,  8809,  8844,  8857,  8914,  8959,  8966,  9170,
 9252,  9437,  9455,  9497,  9504,  9570,  9762,  9790,  9830,
 9843,  9850,  9915, 10002, 10011, 10014, 10015, 10016, 10019,
10024, 10025, 10027, 10032, 10052, 10070, 10076, 10082, 10087,
10106, 10111, 10112, 10128, 10136, 10142, 10147, 10156, 10164,
10176, 10192, 10209, 10213, 10221, 10223, 10241, 10250, 10258,
10260, 10266, 10268, 10269, 10281, 10287, 10291, 10296, 10306,
10310, 10312, 10322, 10324, 10343, 10367, 10370, 10380, 10381,
10382, 10387, 10388, 10402, 10404, 10419, 10431, 10433, 10436,
10442, 10446, 10452, 10460, 10470, 10487, 10492, 10494, 10501,
10511, 10520, 10521, 10532, 10543, 10544, 10545, 10549, 10553,
10554, 10562, 10577, 10586, 10588, 10594, 10595, 10601, 10605,
10619, 10629, 10633, 10639, 10648, 10656, 10668, 10684, 10690,
10691, 10697, 10710, 10711, 10714, 10717, 10724, 10734, 10738,
10740, 10746, 10747, 10754, 10757, 10764, 10770, 10791, 10794,
10796, 10802, 10804, 10805, 10814, 10815, 10819, 10831, 10837,
10845, 10853, 10855, 10860, 10862, 10863, 10876, 10878, 10879,
10882, 10889, 10892, 10893, 10900, 10903, 10908, 10909, 10913,
10916, 10917, 10923, 10924, 10931, 10936, 10941, 10955, 10958,
10968, 10970, 10972, 10973, 10974, 10975, 10976, 10983, 10988,
```

```
11005, 11018, 11027, 11040, 11041, 11044, 11051, 11057, 11065,
11067, 11089, 11093, 11101, 11122, 11127, 11134, 11152, 11154,
11159, 11160, 11161, 11167, 11168, 11174, 11180, 11187, 11190,
11195, 11211, 11223, 11225, 11226, 11231, 11235, 11250, 11256,
11266, 11267, 11288, 11300, 11301, 11320, 11327, 11335, 11346,
11350, 11357, 11364, 11376, 11379, 11386, 11388, 11389, 11397,
11400, 11412, 11414, 11418, 11419, 11424, 11426, 11430, 11431,
11435, 11443, 11444, 11446, 11447, 11449, 11458, 11466, 11470,
11477, 11482, 11483, 11484, 11487, 11490, 11521, 11541, 11904,
11914, 11970, 12413, 12444, 12551, 12563, 12606, 12774, 12901,
12908, 12996, 13051, 13063, 13097, 13296, 13320, 13335, 13366,
13384, 13502, 13538, 13781, 13791, 13840, 13854, 13867, 13872,
13882, 13951, 14102, 14128, 14190, 14218, 14260, 14295, 14378,
14381, 14420, 14529, 14648, 14718, 14730, 14733, 14736, 14738,
14739, 14740, 14743, 14745, 14747, 14750, 14752, 14753, 14756,
14757, 14762, 14764, 14766, 14770, 14771, 14774, 14775, 14776,
14777, 14779, 14780, 14781, 14783, 14786, 14787, 14788, 14789,
14791, 14794, 14797, 14798, 14801, 14803, 14808, 14812, 14813,
14816, 14819, 14821, 14825, 14827, 14828, 14829, 14836, 14839,
14840, 14841, 14844, 14850, 14855, 14860, 14866, 14870, 14872,
14873, 14877, 14880, 14883, 14888, 14895, 14896, 14898, 14901,
14903, 14905, 14906, 14907, 14917, 14918, 14919, 14923, 14926,
14927, 14928, 14931, 14933, 14934, 14939, 14941, 14946, 14947,
14948, 14949, 14950, 14952, 14954, 14957, 14961, 14962, 14964,
14966, 14971, 14973, 14976, 14977, 14981, 14989, 14991, 14992,
14993, 14996, 14999, 15001, 15009, 15012, 15014, 15015, 15018,
15022, 15026, 15027, 15029, 15030, 15031, 15034, 15036, 15045,
15051, 15052, 15058, 15059, 15061, 15062, 15063, 15072, 15074,
15075, 15078, 15079, 15080, 15082, 15086, 15087, 15088, 15090,
15091, 15092, 15098, 15099, 15101, 15106, 15107, 15111, 15113,
15116, 15118, 15119, 15120, 15121, 15123, 15124, 15129, 15130,
15131, 15135, 15136, 15138, 15140, 15142, 15144, 15145, 15149,
15151, 15154, 15155, 15156, 15157, 15161, 15162, 15163, 15166,
15171, 15174, 15176, 15177, 15189, 15197, 15198, 15199, 15202,
15203, 15204, 15205, 15206, 15208, 15209, 15211, 15218, 15219,
15220, 15221, 15227, 15228, 15230, 15232, 15235, 15237, 15238,
15239, 15243, 15252, 15253, 15254, 15258, 15263, 15269, 15270,
15274, 15276, 15278, 15280, 15281, 15282, 15283, 15443, 15510,
15689, 15782, 15889, 16404, 16416, 16475, 16730, 16771, 16861,
16929, 17154, 17182, 17459, 17546, 17671, 17687, 18015, 18134,
18362, 18469, 18471, 18766, 18859, 19026, 19089, 19106, 19250,
19310, 19373, 19416], dtype=int64),)
```


t-SNE Visualization of Embeddings with Outliers for Genres
Outliers (Centroids)
Outliers (k-means)
Outliers (Isolation Forest)
100
50
0
−50
−100
−100
−50
0
50
100

```
--- Task: Styles ---
2/2 ━━━━━━━━━━━━━━━━━━━━ 3s 765ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 785ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 833ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 856ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 812ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 867ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 817ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 852ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 811ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 844ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 797ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 803ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 807ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 893ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 807ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 809ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 808ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 800ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 799ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 798ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 801ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 792ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 799ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 793ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 801ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 802ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 805ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 864ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 849ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 794ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 777ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 795ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 794ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 801ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 777ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 803ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 786ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 787ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 826ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 884ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 827ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 823ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 788ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 792ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 789ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 804ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 797ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 776ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 800ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 788ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 804ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 788ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 831ms/step
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 845ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 833ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 804ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 799ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 803ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 800ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 794ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 799ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 802ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 797ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 794ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 800ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 781ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 808ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 797ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 832ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 832ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 851ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 786ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 794ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 786ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 785ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 791ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 789ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 800ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 781ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 780ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 798ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 837ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 857ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 833ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 793ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 806ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 815ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 813ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 827ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 819ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 831ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 814ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 813ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 822ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 818ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 798ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 782ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 857ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 856ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 857ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 795ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 825ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 817ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 814ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 820ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 823ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 815ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 825ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 833ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 822ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 827ms/step
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 815ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 798ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 850ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 855ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 842ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 826ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 820ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 797ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 822ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 829ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 834ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 823ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 800ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 789ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 935ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 861ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 799ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 803ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 810ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 833ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 819ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 832ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 826ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 839ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 828ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 835ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 821ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 824ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 814ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 819ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 843ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 835ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 943ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 788ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 803ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 802ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 829ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 823ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 817ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 817ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 828ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 849ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 827ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 809ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 800ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 850ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 879ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 810ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 787ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 800ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 785ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 795ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 833ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 828ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 783ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 824ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 879ms/step
```

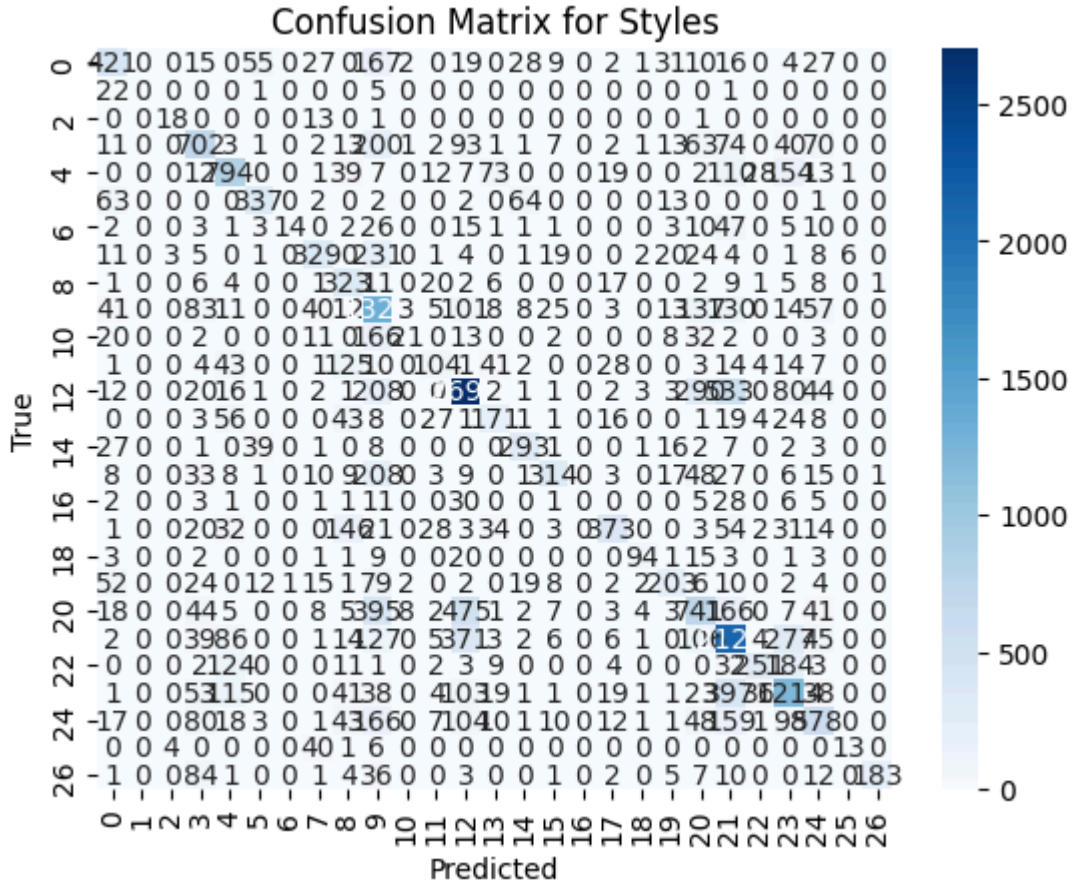
```
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 859ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 861ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 817ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 795ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 855ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 904ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 889ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 899ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 849ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 836ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 813ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 843ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 875ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 833ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 829ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 824ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 855ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 843ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 849ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 797ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 861ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 905ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 848ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 846ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 803ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 828ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 825ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 861ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 833ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 811ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 833ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 842ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 816ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 818ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 811ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 870ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 882ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 848ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 820ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 805ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 800ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 864ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 847ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 828ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 794ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 772ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 839ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 763ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 770ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 788ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 840ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 898ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 862ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 778ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 756ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 770ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 793ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 805ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 784ms/step
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 807ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 805ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 784ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 786ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 801ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 841ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 823ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 868ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 852ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 796ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 825ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 834ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 844ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 832ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 804ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 808ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 838ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 806ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 808ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 836ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 833ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 814ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 869ms/step
Warning: File processed_batches_styles_val\batch_24.npz has only 421 images (expe
cted 1000). Yielding partial batch.
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 824ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 792ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 786ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 781ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 792ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 787ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 868ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 822ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 865ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 804ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 773ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 816ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 793ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 784ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 793ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 766ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 773ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 784ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 809ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 786ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 785ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 834ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 830ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 864ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 773ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 770ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 776ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 767ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 781ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 802ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 799ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 798ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 795ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 779ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 767ms/step
```

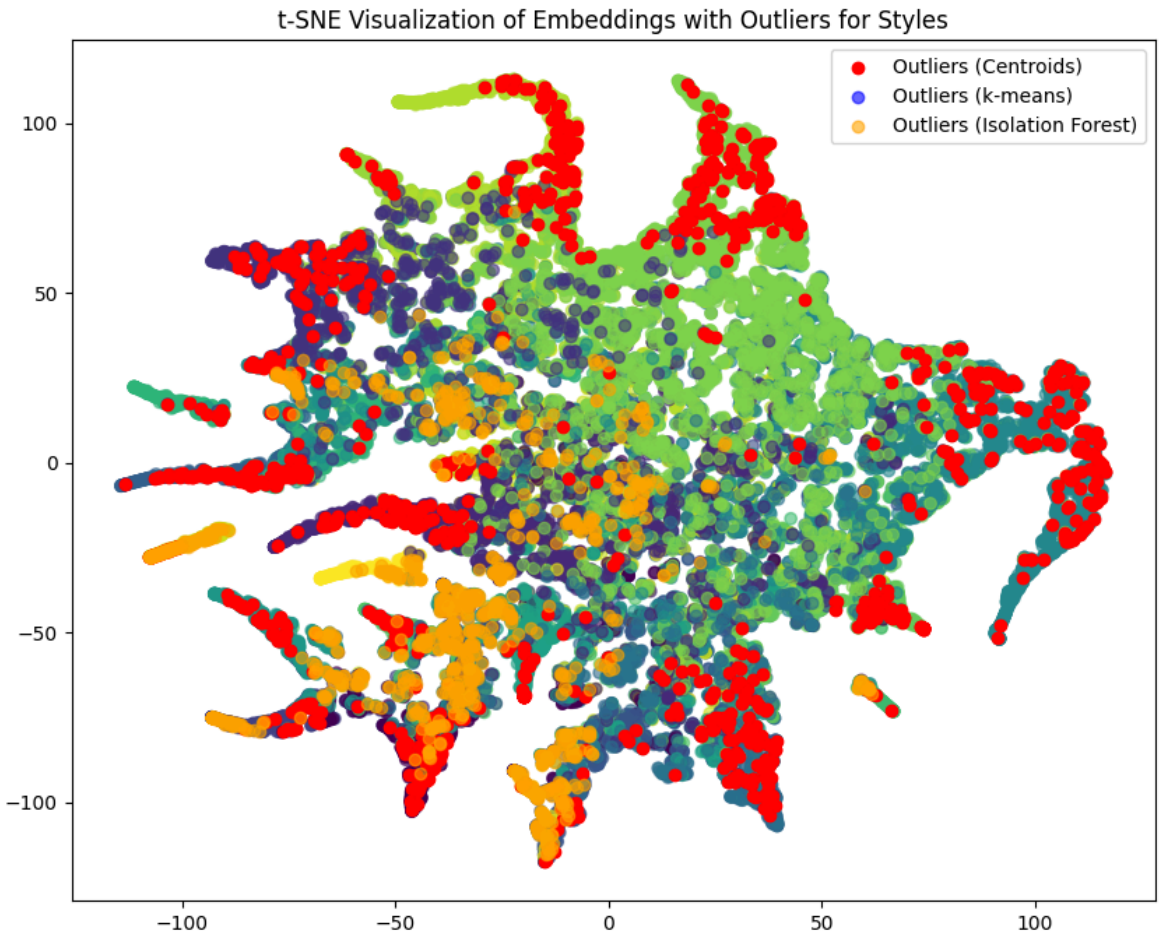
```
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 807ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 819ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 835ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 855ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 817ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 775ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 823ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 772ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 823ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 796ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 805ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 795ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 807ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 784ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 776ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 775ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 775ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 802ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 838ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 882ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 880ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 779ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 786ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 800ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 783ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 764ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 780ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 767ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 765ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 769ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 801ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 772ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 836ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 872ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 846ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 799ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 772ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 772ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 775ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 781ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 789ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 770ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 771ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 787ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 787ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 803ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 770ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 851ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 907ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 876ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 821ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 814ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 770ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 768ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 817ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 818ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 821ms/step
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 811ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 809ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 793ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 793ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 836ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 843ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 885ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 787ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 810ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 813ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 806ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 789ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 802ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 799ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 790ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 798ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 809ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 793ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 801ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 800ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 798ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 801ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 810ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 802ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 3s 2s/step
Warning: File processed_batches_styles_val\batch_24.npz has only 421 images (expe
cted 1000). Yielding partial batch.
382/382 ━━━━━━━━━━━━━━━━━━━━ 586s 2s/step
C:\Users\KIIT\AppData\Roaming\Python\Python312\site-packages\keras\src\trainers\e
poch_iterator.py:151: UserWarning: Your input ran out of data; interrupting train
ing. Make sure that your dataset or generator can generate at least `steps_per_ep
och * epochs` batches. You may need to use the `.repeat()` function when building
your dataset.
  self._interrupted_warning()
C:\Users\KIIT\AppData\Roaming\Python\Python312\site-packages\sklearn\metrics\_cla
ssification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being s
et to 0.0 in labels with no predicted samples. Use `zero_division` parameter to c
ontrol this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\KIIT\AppData\Roaming\Python\Python312\site-packages\sklearn\metrics\_cla
ssification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being s
et to 0.0 in labels with no predicted samples. Use `zero_division` parameter to c
ontrol this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\KIIT\AppData\Roaming\Python\Python312\site-packages\sklearn\metrics\_cla
ssification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being s
et to 0.0 in labels with no predicted samples. Use `zero_division` parameter to c
ontrol this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.57      0.50      0.54       834
           1       0.00      0.00      0.00        29
           2       0.72      0.55      0.62        33
           3       0.57      0.54      0.55      1300
           4       0.60      0.62      0.61      1272
           5       0.74      0.70      0.72       484
           6       0.93      0.10      0.18       144
           7       0.65      0.49      0.56       670
           8       0.39      0.77      0.52       417
           9       0.38      0.66      0.48      2020
          10       0.57      0.07      0.13       280
          11       0.47      0.26      0.33       402
          12       0.66      0.69      0.67      3918
          13       0.45      0.45      0.45       383
          14       0.69      0.73      0.71       401
          15       0.75      0.44      0.55       721
          16       0.00      0.00      0.00        94
          17       0.73      0.49      0.58       765
          18       0.85      0.61      0.71       153
          19       0.58      0.46      0.51       444
          20       0.47      0.38      0.42      1935
          21       0.53      0.66      0.59      3219
          22       0.76      0.40      0.52       626
          23       0.56      0.58      0.57      2105
          24       0.57      0.43      0.49      1358
          25       0.65      0.20      0.31        64
          26       0.99      0.52      0.68       350

    accuracy                           0.56     24421
   macro avg       0.59      0.46      0.48     24421
weighted avg       0.58      0.56      0.55     24421


Confusion Matrix:
```




```
Outlier Detection:
Outliers based on distances from centroids: (array([    1,    58,    72, ..., 243
79, 24397, 24406], dtype=int64),)
Outliers based on k-means clustering: (array([], dtype=int64),)
Outliers detected by Isolation Forest: (array([  218,   278,   443, ..., 24237, 2
4322, 24366], dtype=int64),)
```


t-SNE Visualization of Embeddings with Outliers for Styles
Outliers (Centroids)
Outliers (k-means)
Outliers (Isolation Forest)
100
50
0
−50
−100
−100
−50
0
50
100

In [ ]: