```
2 import pandas as pd
  3 csv_path = './Data/csv'
 4 artist_train = pd.read_csv('./Data/csv/Artist/artist_train')
 5 # lets visualize one imag
 6 base_url = './Data/image/wikiart'
 1 # lets start creating data
 2 artist = './Data/csv/Artist'
 3 genre = './Data/csv/Genre'
 4 style = './Data/csv/Style
 6 data dir = './Data/csv'
 7 # artist_train = artist + '/artist_train'
 8 # artist_val = artist + '/artist_val'
 9 # artist class = artist + '/artist class.txt'
 11 # genre_train = genre + '/genre_train.csv'
 12 # genre_val = genre + '/genre_val.csv'
 13 # genre_class = genre + '/genre_class.txt'
 15 # style_train = style + '/style_train.csv'
 16 # style_val = style + '/style_val.csv'
 17 # style class = style + '/style class.txt'
 18
 19 artist_train_path = data_dir + '/artist_train.csv'
 20 artist_val_path = data_dir + '/artist_val.csv'
 21 artist_class_path = data_dir + '/artist_class.txt'
 23 genre_train_path = data_dir + '/genre_train.csv'
 24 genre_val_path = data_dir + '/genre_val.csv'
 25 genre_class_path = data_dir + '/genre_class.txt'
 27 style_train_path = data_dir + '/style_train.csv'
 28 style_val_path = data_dir + '/style_val.csv'
 29 style_class_path = data_dir + '/style_class.txt'
 30
 31
 33 artist_train = pd.read_csv(data_dir + '/artist_train.csv')
 34 artist_val = pd.read_csv(data_dir + '/artist_val.csv')
 35 artist_class = pd.read_csv(data_dir + '/artist_class.txt')
 37 genre_train = pd.read_csv(data_dir + '/genre_train.csv')
 38 genre_val = pd.read_csv(data_dir + '/genre_val.csv')
 39 genre class = pd.read csv(data dir + '/genre class.txt')
 40
 41 style_train = pd.read_csv(data_dir + '/style_train.csv')
 42 style val = pd.read csv(data dir + '/style val.csv')
 43 style_class = pd.read_csv(data_dir + '/style_class.txt')
 44
 45
 46
 47
 48 artist_train[:4]
path class no
         Realism/vincent-van-gogh_pine-trees-in-the-fen...
                                                           22
     1 Baroque/rembrandt_the-angel-appearing-to-the-s...
                                                           20
         Post Impressionism/paul-cezanne portrait-of-th...
                                                           16
     3
         Impressionism/pierre-auguste-renoir_young-girl...
                                                           17
  1 import os
  2 import pandas as pd
  3 import torch
  4 from torch.utils.data import Dataset
  5 from torchvision import transforms
  6 from PIL import Image
  7 from collections import defaultdict
  8 import random
  9 from tqdm import tqdm
```

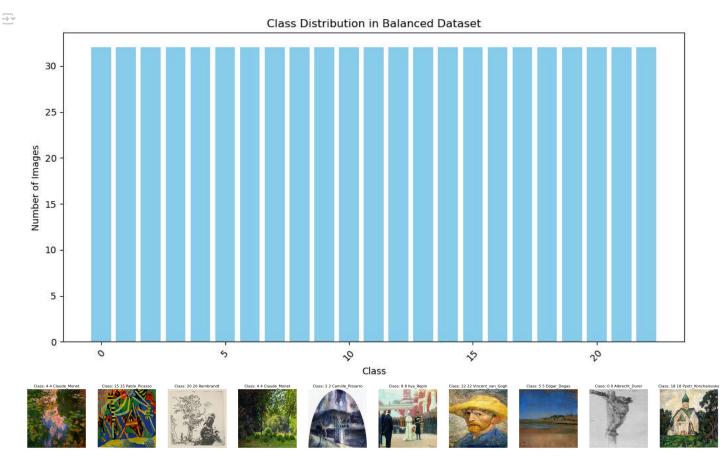


```
10 import matplotlib.pyplot as plt
11
12 # Define dataset class
13 class BalancedArtDataset(Dataset):
      def __init__(self, csv_file, img_dir, class_mapping, transform=None, images_per_class=32):
           self.data = pd.read_csv(csv_file)
15
16
           self.img dir = img dir
17
           self.class_mapping = class_mapping
          self.transform = transform
18
19
          self.images_per_class = images_per_class
20
21
           # Filter out missing images
22
           print("Filtering missing images...")
           self.data = self.data[self.data.iloc[:, 0].apply(lambda x: os.path.exists(os.path.join(img_dir, str(x))))]
23
25
          # Group images by class
26
           print("Grouping images by class...")
27
           self.class_images = defaultdict(list)
           for _, row in tqdm(self.data.iterrows(), total=len(self.data), desc="Processing rows"):
28
29
               self.class_images[row.iloc[1]].append(row)
30
           # Balance dataset with 32 images per class
31
32
          print("Balancing dataset...")
          self.final_data = []
33
           all_images = []
35
           for cls, images in tqdm(self.class_images.items(), total=len(self.class_images), desc="Processing classes"):
36
               if len(images) >= images per class:
37
                   selected_images = random.sample(images, images_per_class)
38
               else:
39
                   selected_images = images[:]
40
                   all_images.extend(images) # Store extra images for filling
41
               self.final_data.extend(selected_images)
42
           # Fill missing slots with extra images
43
           print("Filling missing slots...")
45
           needed_images = images_per_class * len(self.class_images) - len(self.final_data)
46
           if needed images > 0:
47
               self.final_data.extend(random.sample(all_images, min(needed_images, len(all_images))))
48
49
           # Shuffle dataset
50
           print("Shuffling dataset...")
51
           random.shuffle(self.final_data)
52
53
           # Count images per class
          self.class_counts = defaultdict(int)
55
           for row in self.final_data:
56
               self.class counts[row.iloc[1]] += 1
57
      def __len__(self):
58
59
           return len(self.final data)
60
61
      def __getitem__(self, idx):
62
           row = self.final_data[idx]
           img_path = os.path.join(self.img_dir, str(row.iloc[0]))
63
64
          label = row.iloc[1]
65
           image = Image.open(img_path).convert("RGB")
66
67
           if self.transform:
               image = self.transform(image)
68
69
70
          return image, label
71
      def visualize_class_distribution(self):
72
73
           plt.figure(figsize=(12, 6))
74
           plt.bar(self.class_counts.keys(), self.class_counts.values(), color='skyblue')
75
           plt.xlabel("Class")
           plt.ylabel("Number of Images")
76
77
          plt.title("Class Distribution in Balanced Dataset")
78
          plt.xticks(rotation=45)
79
           plt.show()
80
      def visualize_samples(self, num_samples=10):
81
82
           fig, axes = plt.subplots(1, num_samples, figsize=(40, 20))
                                                                                                      McAfee | WebAdvisor
83
           for i in range(num_samples):
84
               image, label = self.__getitem__(random.randint(0, len(self) - 1))
                                                                                                      We'll let you know if there's an issue.
85
               image = image.permute(1, 2, 0).numpy() # Convert to (H, W, C)
               image = (image * 0.5) + 0.5 # Unnormalize
```

```
87
                axes[i].imshow(image)
 88
                axes[i].set_title(f"Class: {label} {artist_class['class_name'][label]}")
 89
                axes[i].axis("off")
 90
            plt.show()
 91
 92
        # Function to compare artist and genre relationships
 93
 94
 95
 96 # Define transformations
 97 transform = transforms.Compose([
 98
       transforms.Resize((224, 224)),
 99
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])
100
101 ])
102
103 # Create balanced artist dataset
104 print("Creating balanced artist dataset...")
105 artist_balanced_dataset = BalancedArtDataset(artist_train_path, "./Data/image/wikiart", artist_class_path, transform=transform, images_
107 # Check dataset length
108 print("Artist balanced dataset size:", len(artist_balanced_dataset))
109
110 # Visualize class distribution
111 artist_balanced_dataset.visualize_class_distribution()
112
113 # Visualize sample images
114 artist_balanced_dataset.visualize_samples(num_samples=10)
115

→ Creating balanced artist dataset...
    Filtering missing images...
    Grouping images by class..
                                                                                    | 13344/13344 [00:02<00:00, 6538.13it/s]
    Processing rows: 100%
    Balancing dataset...
    Processing classes: 100%
                                                                                       23/23 [00:00<00:00, 11122.91it/s]
    Filling missing slots...
    Shuffling dataset...
    Artist balanced dataset size: 736
                                                    Class Distribution in Balanced Dataset
        30
        25
     Number of Images
        20
        10
         5
         0
                                                                   0
                                                                                           $
                                                                                                                    20
                                                                       Class
                                                                                                       McAfee | WebAdvisor
                                                                                                       Your download's being scanned.
```

We'll let you know if there's an issue.



```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
5 # Visualize artist-genre relationship
6 def visualize_artist_genre_relationship(artist_csv, genre_csv):
      # Load datasets
8
      artist_data = pd.read_csv(artist_csv)
9
      genre_data = pd.read_csv(genre_csv)
10
11
      # Merge datasets on image path
12
      merged_data = artist_data.merge(genre_data, on="path")
13
14
      # Extract artist and genre relationships
15
      artist_genre_counts = merged_data.groupby(["class_no_x", "class_no_y"]).size().unstack(fill_value=0)
16
17
      # Plot heatmap
18
      plt.figure(figsize=(12, 8))
      sns.heatmap(artist_genre_counts, cmap="Blues", annot=True, fmt="d")
19
20
      plt.xlabel("Genre")
      plt.ylabel("Artist")
21
      plt.title("Artist-Genre Relationship")
22
23
      plt.xticks(rotation=45, ha="right")
      plt.yticks(rotation=0)
24
                                                                                                      McAfee | WebAdvisor
25
      plt.show()
26
                                                                                                       Your download's being scanned.
27 # File paths
                                                                                                      We'll let you know if there's an issue.
28 df_artist_path = artist_train_path # Replace with actual artist CSV path
```

 $\overline{\Rightarrow}$

```
29 df_genre_path = genre_train_path  # Replace with actual genre CSV path
30
31 # Run visualization
32 visualize_artist_genre_relationship(df_artist_path, df_genre_path)
33
```

Artist-Genre Relationship 0 -1 -2 -3 -4 -5 -- 400 6 -7 -8 -9 -- 300 10 -11 -12 -13 -14 -15 -16 -17 -18 -- 100 19 -20 -21 -22 -- 0 A Genre

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
5 # Visualize artist-genre relationship
6 def visualize_artist_genre_relationship(artist_csv, genre_csv, artist_class_txt, genre_class_txt):
7
      # Load datasets
8
      artist_data = pd.read_csv(artist_csv)
9
      genre_data = pd.read_csv(genre_csv)
10
      # Load artist and genre class data from text files and assign index as class no
11
12
      artist_classes = pd.read_csv(artist_class_txt, header=None, names=["artist_name"])
13
      genre_classes = pd.read_csv(genre_class_txt, header=None, names=["genre_name"])
14
15
      # Assign class_no starting from 0
16
      artist_classes.reset_index(inplace=True)
      genre_classes.reset_index(inplace=True)
17
18
19
      # Rename index column to class_no
20
      artist_classes.rename(columns={"index": "class_no"}, inplace=True)
      genre_classes.rename(columns={"index": "class_no"}, inplace=True)
21
22
23
      # Merge datasets on image path
24
      merged_data = artist_data.merge(genre_data, on="path")
25
26
      # Convert class_no_x and class_no_y to integers
27
      merged_data["class_no_x"] = merged_data["class_no_x"].astype(int)
28
      merged_data["class_no_y"] = merged_data["class_no_y"].astype(int)
                                                                                                       McAfee | WebAdvisor
                                                                                                                                      X
29
30
      # Merge to replace class_no with actual names
                                                                                                       Your download's being scanned.
31
      merged_data = merged_data.merge(artist_classes, left_on="class_no_x", right_on="cl
                                                                                                       We'll let you know if there's an issue.
      merged_data = merged_data.merge(genre_classes, left_on="class_no_y", right_on="class_
```

```
33
34
       # Drop unnecessary columns to prevent conflicts
35
       merged_data.drop(columns=["class_no", "class_no_artist", "class_no_genre"], errors="ignore", inplace=True)
36
37
       # Extract artist and genre relationships
       artist_genre_counts = merged_data.groupby(["artist_name", "genre_name"]).size().unstack(fill_value=0)
38
39
40
       # Plot heatmap
41
       plt.figure(figsize=(12, 8))
42
       sns.heatmap(artist_genre_counts, cmap="Blues", annot=True, fmt="d")
43
       plt.xlabel("Genre")
44
       plt.ylabel("Artist")
45
       plt.title("Artist-Genre Relationship")
46
       plt.xticks(rotation=45, ha="right")
47
       plt.yticks(rotation=0)
48
       plt.show()
49
50 # File paths
51 df_artist_path = artist_train_path # Replace with actual artist CSV path
52 df_genre_path = genre_train_path  # Replace with actual genre CSV path
53 df_artist_class_txt = artist_class_path # Artist class text file (each line is a name)
54 df_genre_class_txt = genre_class_path
                                            # Genre class text file (each line is a name)
56 # Run visualization
57 visualize_artist_genre_relationship(df_artist_path, df_genre_path, df_artist_class_txt, df_genre_class_txt)
58
<del>_</del>
```

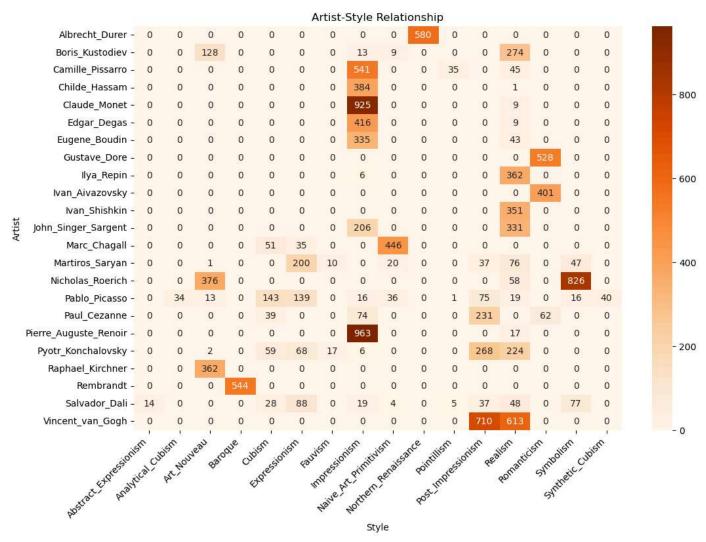
Artist-Genre Relationship Albrecht Durer -Boris_Kustodiev -Camille Pissarro -Childe Hassam -Claude_Monet -Edgar_Degas -Eugene_Boudin -Gustave Dore -Ilya_Repin -Ivan_Aivazovsky -- 300 Ivan Shishkin -John_Singer_Sargent -Marc Chagall -Martiros_Saryan -Nicholas_Roerich -- 200 Pablo Picasso -Paul_Cezanne -Pierre Auguste Renoir -Pyotr_Konchalovsky -- 100 Raphael_Kirchner -Rembrandt -Salvador_Dali -Vincent van Gogh -seetch and sudy - 0 nude Painting Genre

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 # Visualize artist-style relationship
6 def visualize_artist_style_relationship(artist_csv, style_csv, artist_class_txt, style)
7 # Load datasets
We'll let you know if there's an issue.
```

```
8
      artist_data = pd.read_csv(artist_csv)
9
      style data = pd.read csv(style csv)
10
      # Load artist and style class data from text files and assign index as class_no
11
      artist_classes = pd.read_csv(artist_class_txt, header=None, names=["artist_name"])
12
13
      style_classes = pd.read_csv(style_class_txt, header=None, names=["style_name"])
15
      # Assign class_no starting from 0
      artist classes.reset index(inplace=True)
16
17
      style_classes.reset_index(inplace=True)
18
19
      # Rename index column to class no
20
      artist_classes.rename(columns={"index": "class_no"}, inplace=True)
      style_classes.rename(columns={"index": "class_no"}, inplace=True)
21
22
23
      # Merge datasets on image path
24
      merged data = artist data.merge(style data, on="path")
25
26
      # Convert class_no_x and class_no_y to integers
27
      merged_data["class_no_x"] = merged_data["class_no_x"].astype(int)
28
      merged_data["class_no_y"] = merged_data["class_no_y"].astype(int)
29
30
      # Merge to replace class_no with actual names
31
      merged_data = merged_data.merge(artist_classes, left_on="class_no_x", right_on="class_no", how="left", suffixes=("", "_artist"))
32
      merged_data = merged_data.merge(style_classes, left_on="class_no_y", right_on="class_no", how="left", suffixes=("", "_style"))
33
34
      # Drop unnecessary columns to prevent conflicts
35
      merged_data.drop(columns=["class_no", "class_no_artist", "class_no_style"], errors="ignore", inplace=True)
36
37
       # Extract artist and style relationships
      artist_style_counts = merged_data.groupby(["artist_name", "style_name"]).size().unstack(fill_value=0)
38
39
40
      # Plot heatmap
41
      plt.figure(figsize=(12, 8))
42
      sns.heatmap(artist_style_counts, cmap="Oranges", annot=True, fmt="d")
43
      plt.xlabel("Style")
44
      plt.ylabel("Artist")
45
      plt.title("Artist-Style Relationship")
46
      plt.xticks(rotation=45, ha="right")
47
      plt.yticks(rotation=0)
48
      plt.show()
49
50 # File paths
51 df_artist_path = artist_train_path # Replace with actual artist CSV path
52 df_style_path = style_train_path  # Replace with actual style CSV path
53 df_artist_class_txt = artist_class_path # Artist class text file (each line is a name)
54 df_style_class_txt = style_class_path  # Style class text file (each line is a name)
56 # Run visualization
57 visualize artist style relationship(df artist path, df style path, df artist class txt, df style class txt)
58
```





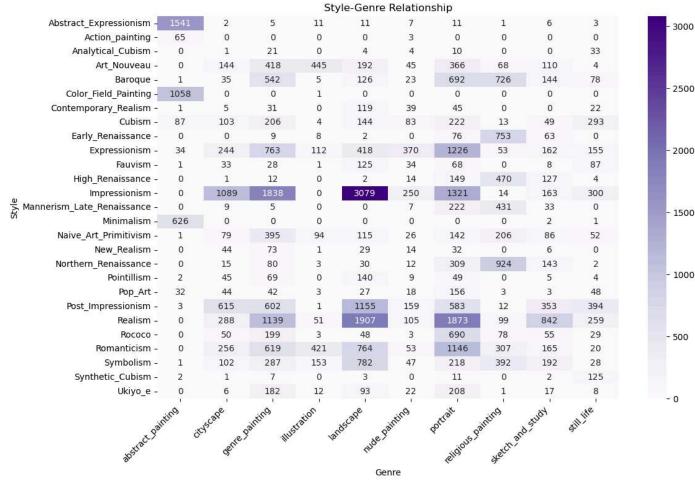


```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
5 # Visualize style-genre relationship
6 def visualize_style_genre_relationship(style_csv, genre_csv, style_class_txt, genre_class_txt):
      # Load datasets
8
      style data = pd.read csv(style csv)
9
      genre_data = pd.read_csv(genre_csv)
10
11
      # Load style and genre class data from text files and assign index as class_no
12
      style_classes = pd.read_csv(style_class_txt, header=None, names=["style_name"])
13
      genre_classes = pd.read_csv(genre_class_txt, header=None, names=["genre_name"])
14
15
      # Assign class_no starting from 0
16
      style_classes.reset_index(inplace=True)
17
      genre_classes.reset_index(inplace=True)
18
19
      # Rename index column to class_no
      style_classes.rename(columns={"index": "class_no"}, inplace=True)
20
21
      genre_classes.rename(columns={"index": "class_no"}, inplace=True)
22
23
      # Merge datasets on image path
24
      merged_data = style_data.merge(genre_data, on="path")
25
26
      # Convert class_no_x and class_no_y to integers
      merged_data["class_no_x"] = merged_data["class_no_x"].astype(int)
27
      merged_data["class_no_y"] = merged_data["class_no_y"].astype(int)
28
                                                                                                       McAfee | WebAdvisor
29
                                                                                                       Your download's being scanned.
30
      # Merge to replace class_no with actual names
      merged data = merged data.merge(style classes, left on="class no x", right on="cla
                                                                                                       We'll let you know if there's an issue.
31
      merged_data = merged_data.merge(genre_classes, left_on="class_no_y", right_on="class_
```

1

1

```
33
34
       # Drop unnecessary columns to prevent conflicts
35
       merged_data.drop(columns=["class_no", "class_no_style", "class_no_genre"], errors="ignore", inplace=True)
36
37
       # Extract style and genre relationships
       style_genre_counts = merged_data.groupby(["style_name", "genre_name"]).size().unstack(fill_value=0)
38
39
40
       # Plot heatmap
41
       plt.figure(figsize=(12, 8))
42
       sns.heatmap(style_genre_counts, cmap="Purples", annot=True, fmt="d")
43
       plt.xlabel("Genre")
44
       plt.ylabel("Style")
45
       plt.title("Style-Genre Relationship")
46
       plt.xticks(rotation=45, ha="right")
47
       plt.yticks(rotation=0)
48
       plt.show()
49
50 # File paths
51 df_style_path = style_train_path # Replace with actual style CSV path
52 df_genre_path = genre_train_path # Replace with actual genre CSV path
53 df_style_class_txt = style_class_path  # Style class text file (each line is a name)
54 df_genre_class_txt = genre_class_path  # Genre class text file (each line is a name)
56 # Run visualization
57 visualize_style_genre_relationship(df_style_path, df_genre_path, df_style_class_txt, df_genre_class_txt)
58
<del>_</del>
```



Your download's being scanned.
We'll let you know if there's an issue.

4/6/25, 6:01 PM

