```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pyarrow.parquet as pq

file_paths = [
    "data/QCDToGGQQ_IMGjet_RH1all_jet0_run0_n36272_LR.parquet",
    "data/QCDToGGQQ_IMGjet_RH1all_jet0_run1_n47540_LR.parquet",
    "data/QCDToGGQQ_IMGjet_RH1all_jet0_run2_n55494_LR.parquet"
]
```

Dataset Exploration

```python
df_list = []
for fp in file_paths:
    pf = pq.ParquetFile(fp)
    batch = next(pf.iter_batches(batch_size=100))
    df_list.append(batch.to_pandas())

df = pd.concat(df_list, ignore_index=True)

print("Dataset Columns:", df.columns)
print("Number of Samples:", len(df))
```

```
Dataset Columns: Index(['X_jets_LR', 'X_jets', 'pt', 'm0', 'y'],
dtype='object')
Number of Samples: 300
```

```python
df.describe()
```

```
               pt          m0           y
count  300.000000  300.000000  300.000000
mean   116.913870   20.956278    0.480000
std     25.567939    5.966501    0.500435
min     75.781281    7.815182    0.000000
25%     98.774593   16.781054    0.000000
50%    110.637585   20.031707    0.000000
75%    129.321957   23.521461    1.000000
max    238.406982   46.464977    1.000000
```

```python
df.head()
```

```
                                           X_jets_LR  \
0  [[[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0...
1  [[[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0...
2  [[[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0...
3  [[[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0...
4  [[[0.0, 0.0, 0.0, 0.0, 0.08212081342935562, 0....

                                              X_jets          pt
m0  \
0  [[[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0...  112.411095
21.098248
1  [[[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0...   95.220406
14.030600
2  [[[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0...   97.007317
17.728968
3  [[[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0...   82.490311
14.702741
4  [[[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.328483...  102.539238
19.456257
```

```
     y
0  0.0
1  1.0
2  1.0
3  0.0
4  0.0
```

```python
print(df["X_jets"][0].shape)

print(df["X_jets_LR"][0][0].shape)
print(df["X_jets_LR"][0][1].shape)
print(df["X_jets_LR"][0][2].shape)

print(df["X_jets"][0][0].shape)
print(df["X_jets"][0][1].shape)
print(df["X_jets"][0][2].shape)
```

```
(3,)
(64,)
(64,)
(64,)
(125,)
(125,)
(125,)
```

```python
print("Example HR image shape:", df["X_jets"][0].shape)
print("Example LR image shape:", df["X_jets_LR"][0].shape)
```

```
Example HR image shape: (3,)
Example LR image shape: (3,)
```

```python
print("Shape of X_jets[0]:", df["X_jets"][0].shape)
print("Shape of X_jets[0][0]:", df["X_jets"][0][0].shape)
```

```
Shape of X_jets[0]: (3,)
Shape of X_jets[0][0]: (125,)
```

```python
df['X_jets'] = df['X_jets'].apply(lambda x:
np.stack([np.stack(ch).astype(np.float32) for ch in x]))
```

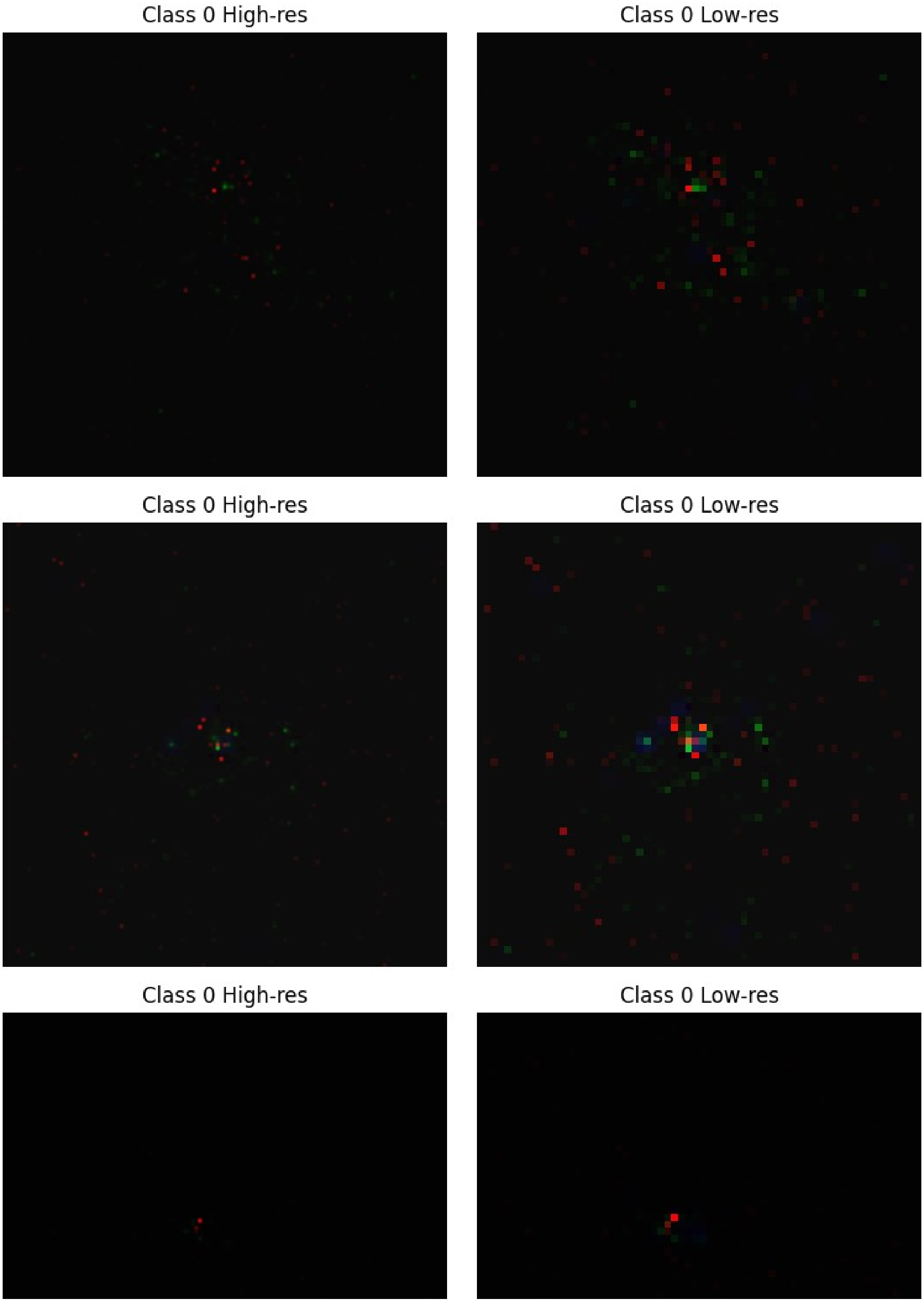
```
df['X_jets_LR'] = df['X_jets_LR'].apply(lambda x:
np.stack([np.stack(ch).astype(np.float32) for ch in x]))

print("High-res image shape:", df['X_jets'].iloc[0].shape)
print("Low-res image shape:", df['X_jets_LR'].iloc[0].shape)
```

```
High-res image shape: (3, 125, 125)
Low-res image shape: (3, 64, 64)
```

Visualize Dataset

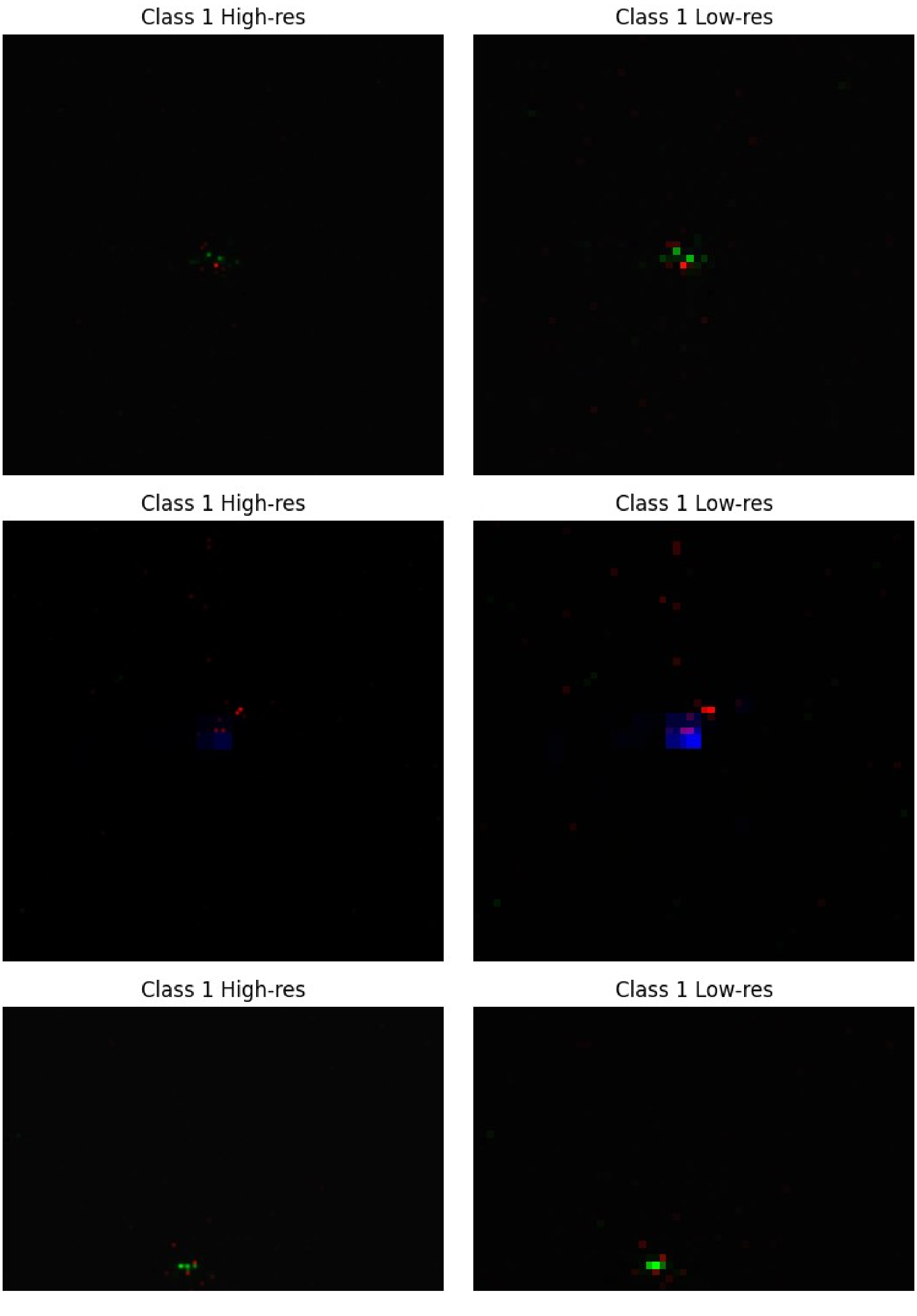
```
def normalize_image(img):
    """
    Normalize image data to the range [0, 1].
    """
    img = np.array(img, dtype=np.float32)
    min_val = img.min()
    max_val = img.max()
    if max_val - min_val > 0:
        return (img - min_val) / (max_val - min_val)
    else:
        return img

def prepare_image(img):
    """
    Transpose from channel-first (3, H, W) to channel-last (H, W, 3)
    and normalize the image data to [0,1] for proper display.
    """
    img = np.transpose(img, (1, 2, 0))
    return normalize_image(img)

def plot_class_images(df_class, class_label):
    """
    For each sample in the DataFrame, plot the high-res and low-res
images side by side.
    """
    n_samples = len(df_class)
    fig, axes = plt.subplots(n_samples, 2, figsize=(8, 4 * n_samples))

    if n_samples == 1:
        axes = np.expand_dims(axes, axis=0)

    for i, (_, row) in enumerate(df_class.iterrows()):
        hr_img = prepare_image(row['X_jets'])
        lr_img = prepare_image(row['X_jets_LR'])

        axes[i, 0].imshow(hr_img)
        axes[i, 0].set_title(f'Class {class_label} High-res')
        axes[i, 0].axis('off')
        axes[i, 1].imshow(lr_img)
        axes[i, 1].set_title(f'Class {class_label} Low-res')
        axes[i, 1].axis('off')

    plt.tight_layout()
    plt.show()

num_samples = 5
class0 = df[df['y'] == 0].sample(num_samples, random_state=69)
class1 = df[df['y'] == 1].sample(num_samples, random_state=69)

print("Class 0 Samples:")
plot_class_images(class0, class_label=0)
```

```
Class 0 Samples:
```


Class 0 High-res
Class 0 Low-res
Class 0 High-res
Class 0 Low-res
Class 0 High-res
Class 0 Low-res

```python
print("Class 1 Samples:")
plot_class_images(class1, class_label=1)
```

```
Class 1 Samples:
```



Class 1 High-res

Class 1 Low-res

Class 1 High-res

Class 1 Low-res

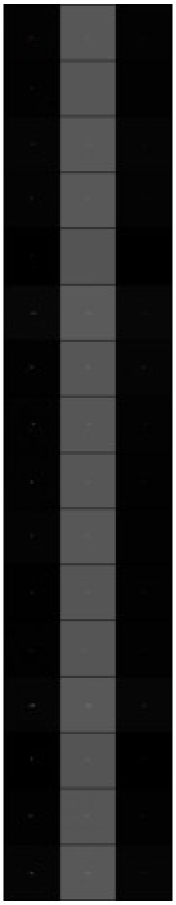
Class 1 High-res

Class 1 Low-res

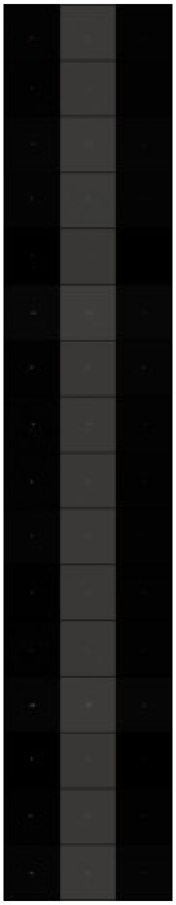
Validation result Visualization

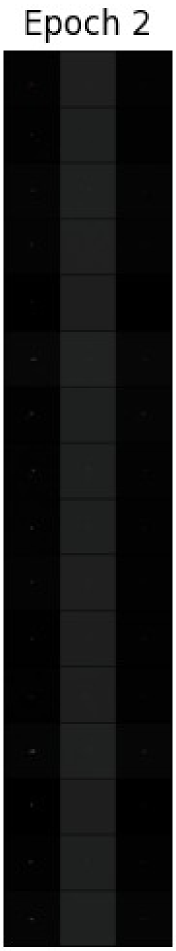
```
import matplotlib.image as mpimg
import os
import glob

eval_dir = "images/training"
val_images = sorted(glob.glob(os.path.join(eval_dir,
"epoch_*_val.png")))

if not val_images:
    print("No evaluation images found in the directory.")
else:
    for img_path in val_images:
        epoch_num = os.path.basename(img_path).split('_')[1]
        image = mpimg.imread(img_path)

        plt.figure(figsize=(6, 6))
        plt.imshow(image)
        plt.title(f"Epoch {epoch_num}")
        plt.axis('off')
        plt.show()
```

Epoch 0

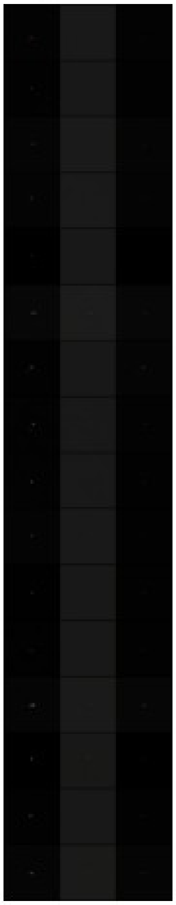


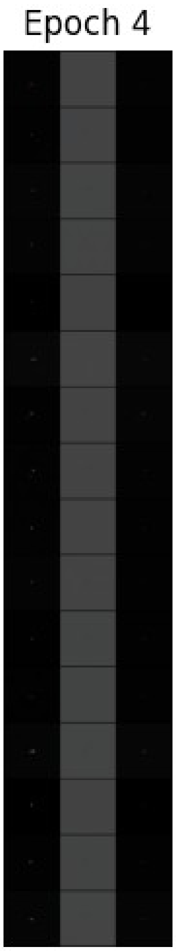
Epoch 1




Epoch 2

Epoch 3





Epoch 4

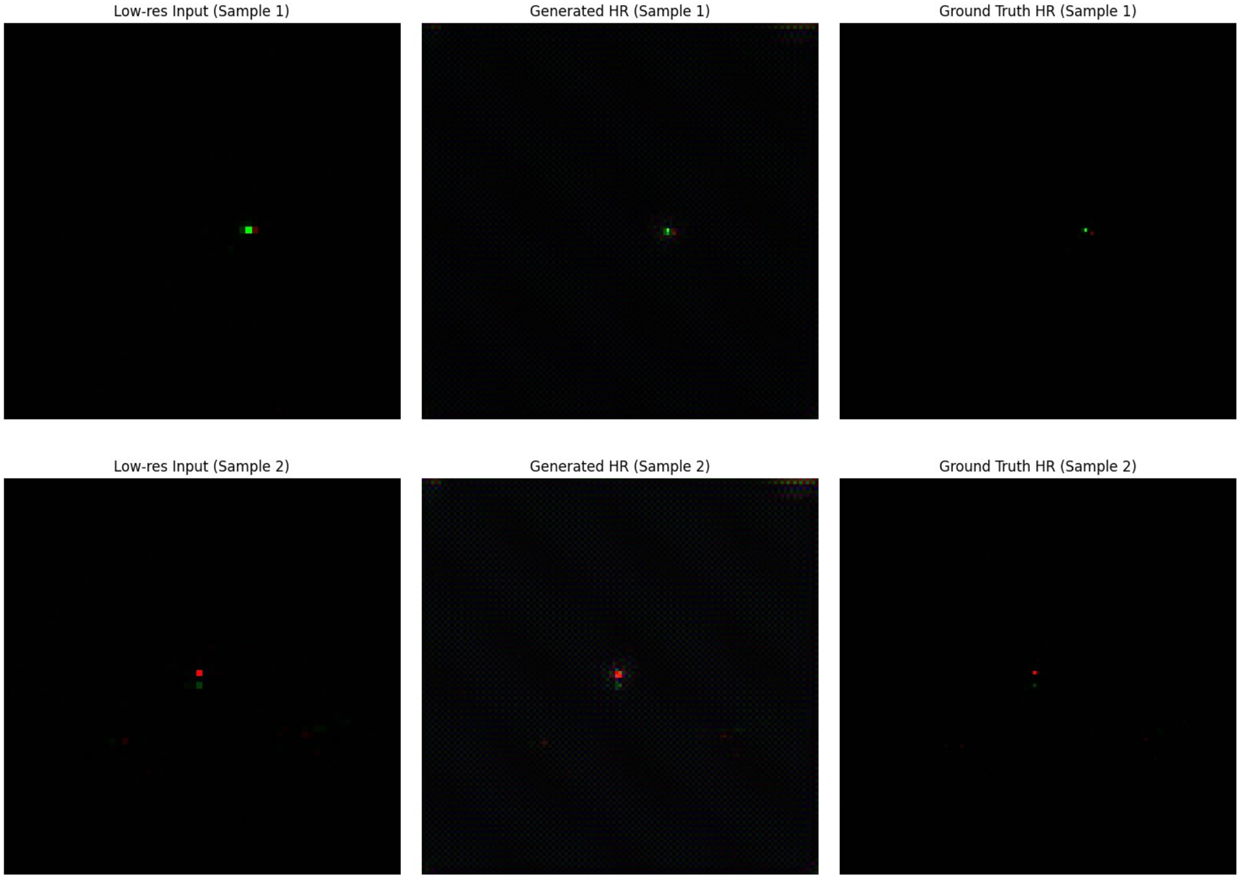
Test Trained model

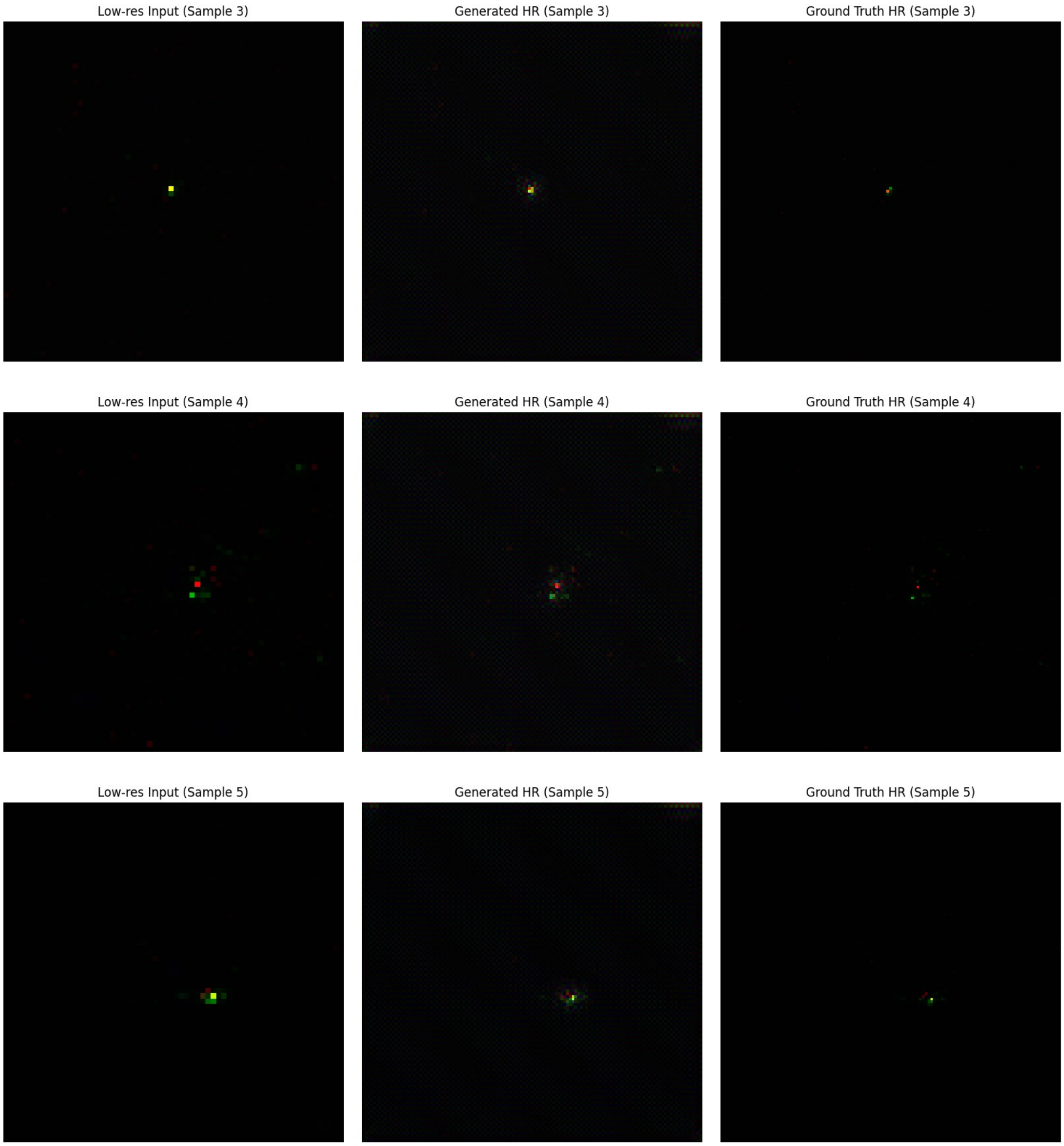
```
import models
import torch
import torch.nn.functional as F
from models import GeneratorRRDB

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using device: {device}")
```

```
Using device: cuda
```

```
hr_height, hr_width = 125, 125

def exact_resize(x):
    return F.interpolate(x, size=(hr_height, hr_width),
mode='bicubic', align_corners=True)

num_samples = 10
samples = df.sample(num_samples, random_state=101)
```

```python
lr_images = []
hr_images = []
for idx, row in samples.iterrows():
    lr_img = torch.from_numpy(row['X_jets_LR']).unsqueeze(0)  # shape:
(1, 3, 64, 64)
    hr_img = torch.from_numpy(row['X_jets']).unsqueeze(0)       #
shape: (1, 3, 125, 125)

    # Normalize images to [-1, 1]
    lr_img = (lr_img - lr_img.min()) / (lr_img.max() - lr_img.min()) *
2 - 1
    hr_img = (hr_img - hr_img.min()) / (hr_img.max() - hr_img.min()) *
2 - 1

    lr_images.append(lr_img)
    hr_images.append(hr_img)

lr_batch = torch.cat(lr_images, dim=0).to(device)
hr_batch = torch.cat(hr_images, dim=0).to(device)

with torch.no_grad():
    gen_hr = generator(lr_batch)
    gen_hr = exact_resize(gen_hr)

def tensor_to_image(tensor):
    """
    Convert a tensor image in range [-1,1] to a numpy image in [0,1].
    """
    tensor = tensor.cpu().detach()
    tensor = (tensor + 1) / 2
    tensor = torch.clamp(tensor, 0, 1)
    return tensor.numpy()

lr_images_np = [tensor_to_image(img) for img in lr_batch]
gen_hr_images_np = [tensor_to_image(img) for img in gen_hr]
hr_images_np = [tensor_to_image(img) for img in hr_batch]

def plot_comparison(lr_img, gen_hr_img, true_hr_img, title_suffix=""):
    """
    Plot the low-resolution image, the generated high-resolution
image, and the ground truth high-resolution image.
    """
    fig, axes = plt.subplots(1, 3, figsize=(15, 5))
    axes[0].imshow(prepare_image(lr_img))
    axes[0].set_title("Low-res Input " + title_suffix)
    axes[0].axis('off')

    axes[1].imshow(prepare_image(gen_hr_img))
    axes[1].set_title("Generated HR " + title_suffix)
    axes[1].axis('off')
    axes[2].imshow(prepare_image(true_hr_img))
    axes[2].set_title("Ground Truth HR " + title_suffix)
    axes[2].axis('off')

    plt.tight_layout()
    plt.show()

for i in range(num_samples):
    plot_comparison(lr_images_np[i], gen_hr_images_np[i],
hr_images_np[i], title_suffix=f"(Sample {i+1})")
```





Low-res Input (Sample 3)

Generated HR (Sample 3)

Ground Truth HR (Sample 3)

Low-res Input (Sample 4)

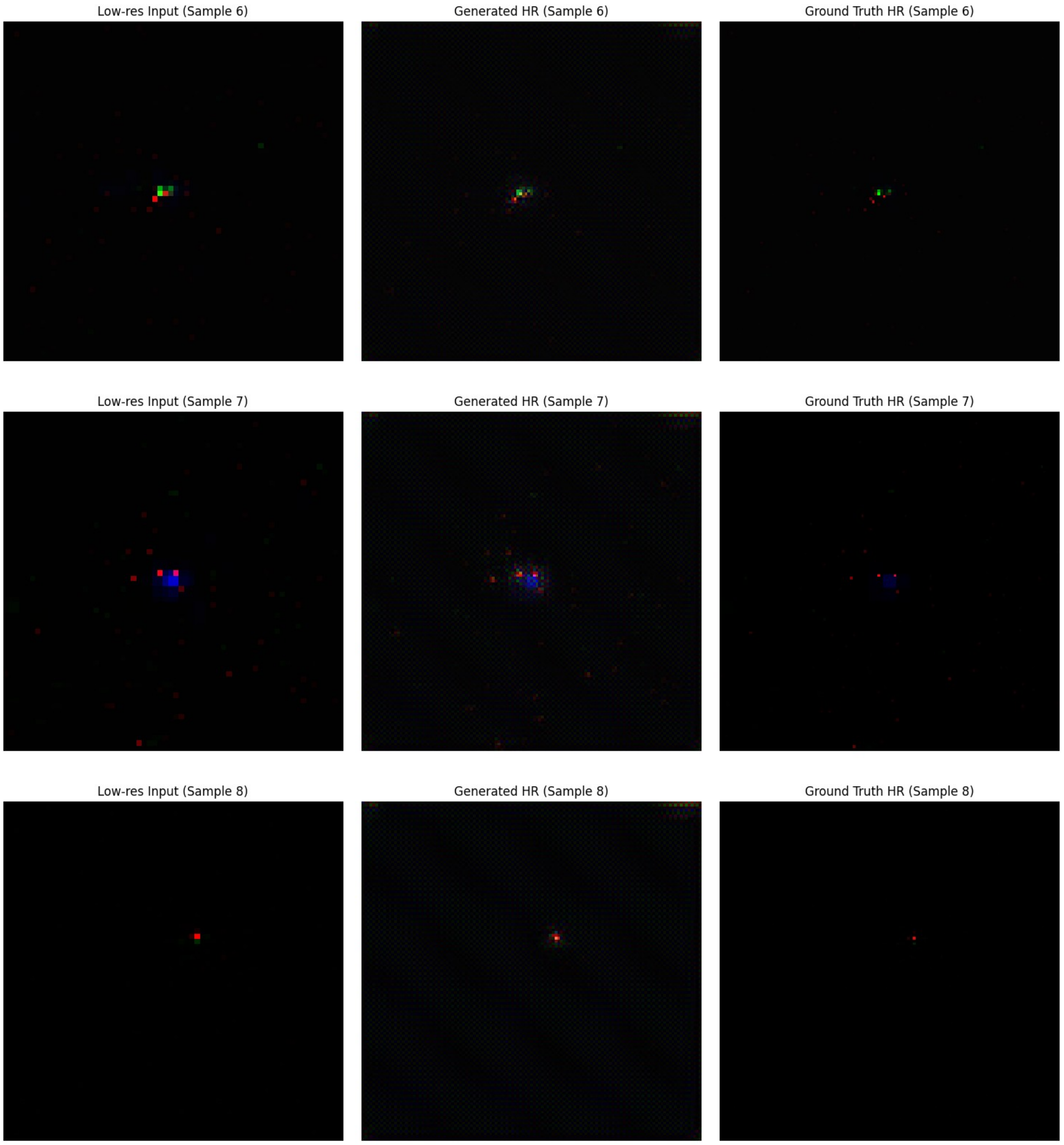
Generated HR (Sample 4)

Ground Truth HR (Sample 4)

Low-res Input (Sample 5)

Generated HR (Sample 5)

Ground Truth HR (Sample 5)



Low-res Input (Sample 6)

Generated HR (Sample 6)

Ground Truth HR (Sample 6)

Low-res Input (Sample 7)

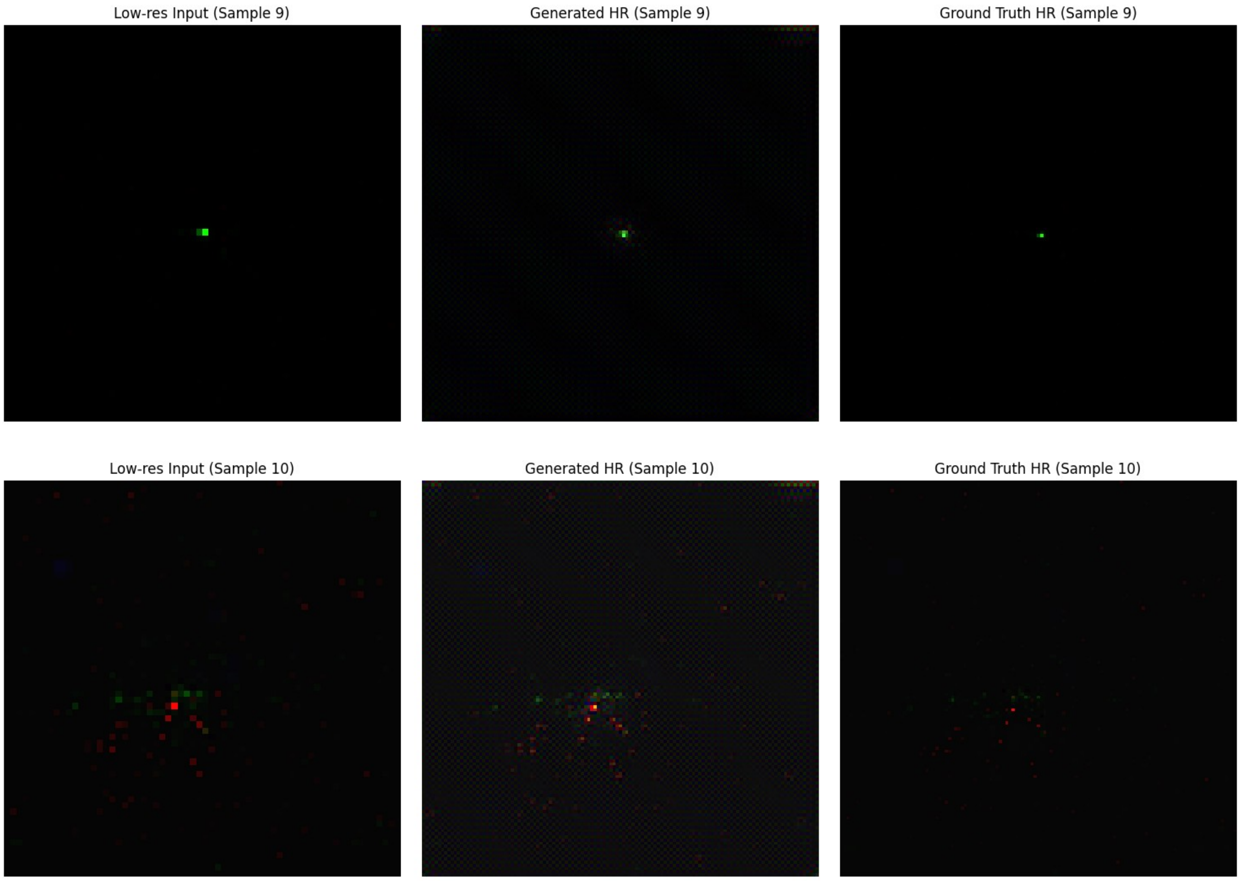
Generated HR (Sample 7)

Ground Truth HR (Sample 7)

Low-res Input (Sample 8)

Generated HR (Sample 8)

Ground Truth HR (Sample 8)



Low-res Input (Sample 9)

Generated HR (Sample 9)

Ground Truth HR (Sample 9)

Low-res Input (Sample 10)

Generated HR (Sample 10)

Ground Truth HR (Sample 10)