Loading Packages

```
In [1]: import os
        import torch
        from utils.visualize       import visualize_images_from_classes
        from utils.datasets        import returnDataLoader
        from utils.model_execution import runFMA, runDLA
        from utils.analysis        import print_analysis
```

```
/home/sysop/programming/python/work/HistoART/.venv/lib/python3.12/site-packages/tqdm/auto.py:21: TqdmWarning: IProgress not found. Please update jupyter and ipywidget
s. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

```
In [2]: if torch.cuda.is_available():
            device = torch.device("cuda")
        elif torch.xpu.is_available():
            device = torch.device("xpu")
        elif torch.mps.is_available():
            device = torch.device("mps")
        else:
            device = torch.device("cpu")

        print(f"Using device: {device}")
```

```
Using device: xpu
```

Load Patches

```
In [3]: while True:
            data_dir = input("Please enter the dataset directory, for example: ./data/: ") or "./data/DeepFocus/"

            if not os.path.isdir(data_dir):
                print(f"Directory '{data_dir}' does not exist. Please try again.\n")
                continue
            print("\nDataset directory confirmed.\n")

            break

        classes            = []
        artifact_free      = input("Please enter the name of the artifact free folder (e.g., artifact_free): ").strip()
        artifact_free_path = os.path.join(data_dir, artifact_free)

        if artifact_free and os.path.isdir(artifact_free_path):
            classes.append(artifact_free)
            print(f"Folder '{artifact_free_path}' exists and was added as artifact free.")
        else:
            if artifact_free:
                print(f"Folder '{artifact_free_path}' does not exist. Please check the name.")

        artifact      = input("Please enter the name of the artifact folder (e.g., artifact): ").strip()
        artifact_path = os.path.join(data_dir, artifact)

        if artifact and os.path.isdir(artifact_path):
            classes.append(artifact)
            print(f"Folder '{artifact_path}' exists and was added as artifact.")
        else:
            if artifact:
                print(f"Folder '{artifact_path}' does not exist. Please check the name.")

        if classes:
            print("\nDataset loaded successfully.")
        else:
            print("\nNo valid class folders were provided. Defaulting to sample classes.")
            classes = ['artifact_free', 'blur']
```

```
Dataset directory confirmed.


No valid class folders were provided. Defaulting to sample classes.
```

Random visualization of different classes

```
In [ ]: visualize_images_from_classes(data_dir, classes)
```

Prepare data for classification

```
In [4]: dataloader = returnDataLoader(data_dir, classes)
        print("Total dataset size (samples): ", len(dataloader.dataset))
```

```
Total dataset size (samples):  204000
```

Classify images with different models

FMA

```
In [5]: runFMA(dataloader, device, './models/fma_binary_blur.pth')
```

```
/home/sysop/programming/python/work/HistoART/utils/model_execution.py:71: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default valu
e), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See http
s://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for more details). In a future release, the default value for `weights_only` will be flipped to `
True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are expli
citly allowlisted by the user via `torch.serialization.add_safe_globals`. We recommend you start setting `weights_only=True` for any use case where you don't have full
control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.
  uni_model.load_state_dict(torch.load(model, map_location=device))
Epoch 1 [test]: 100%|██████████| 12750/12750 [1:30:47<00:00,  2.34it/s]
```

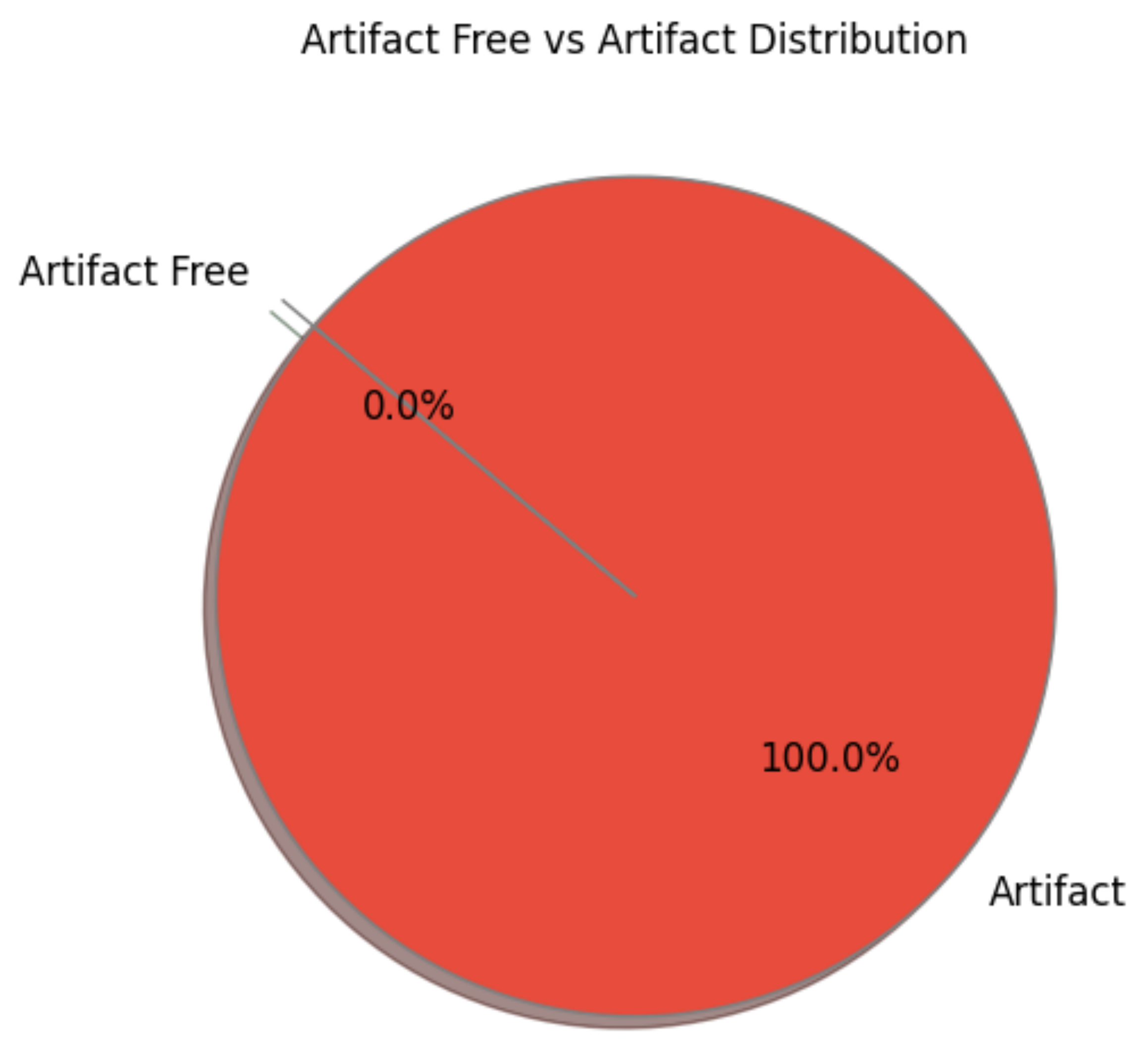
DLA

```
In [ ]: runDLA(dataloader, device, './models/dla_binary.pth')
```

KBA

Analysis

```
In [6]: print_analysis('./results/fma_results.csv')
```



```
In [ ]: print_analysis('./results/dla_results.csv')
```

```
In [ ]:
```