Galaxy Classifier

```
# Install necessary libraries

!pip install -Uqq fastai gradio bing-image-downloader
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 51.3/51.3 MB 14.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 322.2/322.2 kB 12.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 95.2/95.2 kB 6.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.3/11.3 MB 42.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 72.0/72.0 kB 3.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 2.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 32.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 17.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 11.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 5.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 10.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 7.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 7.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 58.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.3/62.3 kB 4.9 MB/s eta 0:00:00
```

```
#Import required libraries
import fastai
from fastai.vision.all import *
import gradio as gr
from bing_image_downloader import downloader
from pathlib import Path
import shutil
```

Image Downloading

```
def download_images_bing(search_term, dest_dir, max_images = 150):
  """
  Downloads images from Bing Image Search and saves them to the specified directory.
  """
  downloader.download(search_term,
                      limit = max_images,
                      output_dir = dest_dir,
                      adult_filter_off= False,
                      force_replace = False)
```

Galaxy Categories

```
galaxy_types = ['spiral galaxy', 'elliptical galaxy', 'irregular galaxy']
path = Path('galaxies')
path.mkdir(exist_ok=True) #Create main directory if it does not exist
```

Download images for each galaxy type

```
for galaxy_type in galaxy_types:
  dest = path/galaxy_type.replace(" ","_") # Replace spaces for better path handling
  dest.mkdir(exist_ok=True)
  download_images_bing(galaxy_type, dest.as_posix(), max_images = 200)
```

Show hidden output

Data Preparation

```
# Get all downloaded image file paths
fns = get_image_files(path)
print(f"Total images found: {len(fns)}")
```

```
Total images found: 595
```

```
#Verify Image integrity
failed = verify_images(fns)
if len(failed) > 0:
  print(f"Removing {len(failed)} corrupted images.")
  for img in failed:
    img.unlink()
else:
  print("All images are valid.")
```

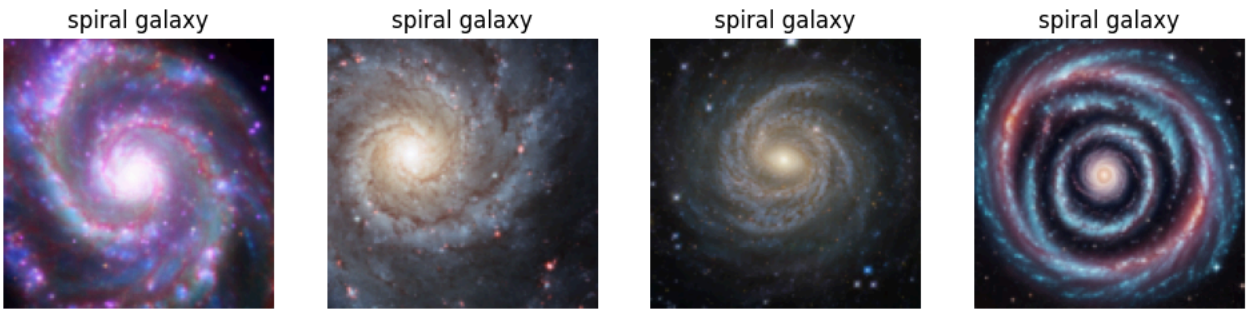
```
All images are valid.
```

Data Loaders for Model Training

```
def get_data_loaders(path, batch_size=64, img_size=128):
  """
    Creates a DataBlock pipeline with image transformations and loads data.
  """
  galaxies = DataBlock(
      blocks = (ImageBlock, CategoryBlock),
      get_items = get_image_files,
      splitter = RandomSplitter(valid_pct=0.2, seed=42),
      get_y = parent_label,
      item_tfms = Resize(img_size),
      batch_tfms=aug_transforms(size=img_size,
                                min_scale=0.5,
                                flip_vert=True,
                                max_rotate=30.0,
                                max_zoom=1.1,
                                max_lighting=0.2,
                                max_warp=0.2)
  )
  return galaxies.dataloaders(path, bs=batch_size)

dls = get_data_loaders(path)
```

Show sample batch

```
dls.valid.show_batch(max_n=4, nrows=1)
```



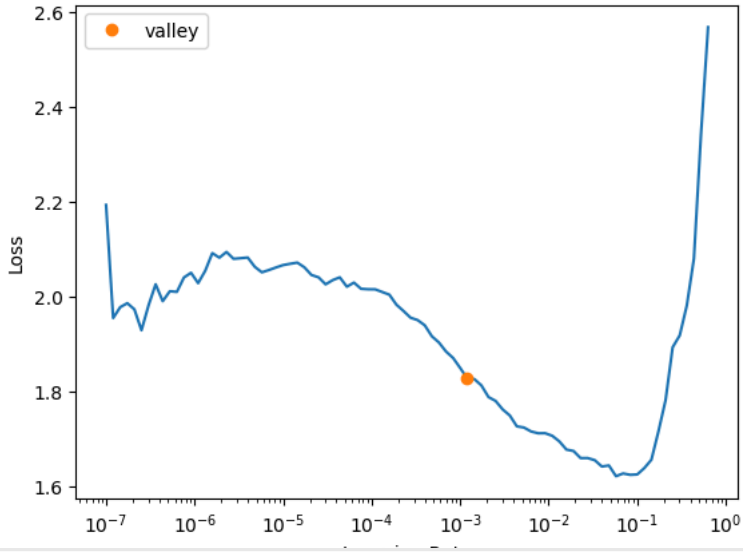
Model Training

```
learn = vision_learner(dls, resnet18, metrics=accuracy, cbs=MixUp())

# Find the optimal learning rate
lr_min = learn.lr_find().valley  # Extract the best LR

# Fine-tune using the suggested learning rate
learn.fine_tune(5, base_lr=lr_min)
```

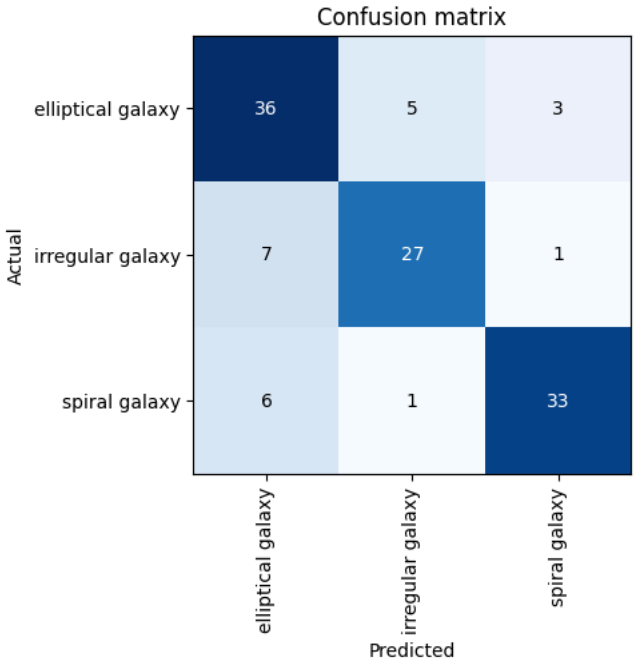
```
Downloading: "https://download.pytorch.org/models/resnet18-f37072fd.pth" to /root/.cache/torch/hub/checkpoints/resnet18-f370
100%|██████████| 44.7M/44.7M [00:00<00:00, 143MB/s]
```

epoch	train_loss	valid_loss	accuracy	time
0	1.704931	1.575493	0.588235	01:00

epoch	train_loss	valid_loss	accuracy	time
0	1.404471	1.021714	0.621849	01:13
1	1.273312	0.758092	0.722689	01:16
2	1.228914	0.575777	0.789916	01:13
3	1.142259	0.533123	0.806723	01:15
4	1.132436	0.535144	0.806723	01:15




```
# Analyze model performance
interp = ClassificationInterpretation.from_learner(learn)
interp.plot_confusion_matrix(figsize=(5,5), dpi=100)  # Display confusion matrix
interp.plot_top_losses(5, nrows=1, figsize=(17,10))
for ax in plt.gcf().axes:
    ax.set_title(ax.get_title(), fontsize=8, rotation=45, ha='right')
    plt.tight_layout()
```



Prediction/Actual/Loss/Probability


```
# --- Image Cleaning ---
from fastai.vision.widgets import ImageClassifierCleaner # Import ImageClassifierCleaner

# Provide a tool to clean misclassified images
cleaner = ImageClassifierCleaner(learn)
cleaner
```