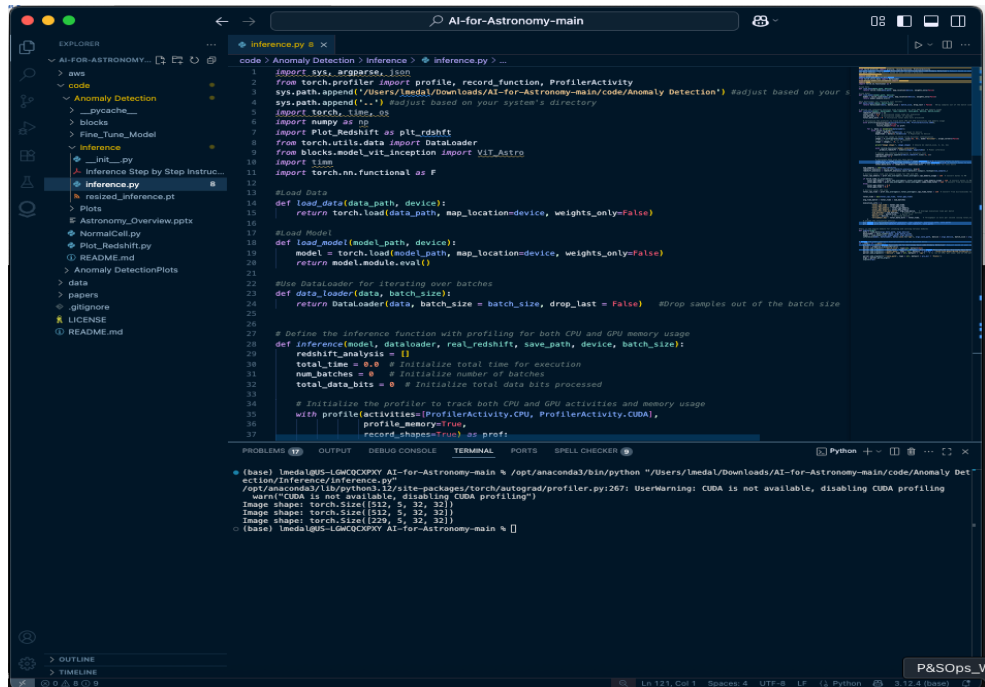


Step 3: Astronomy Inference

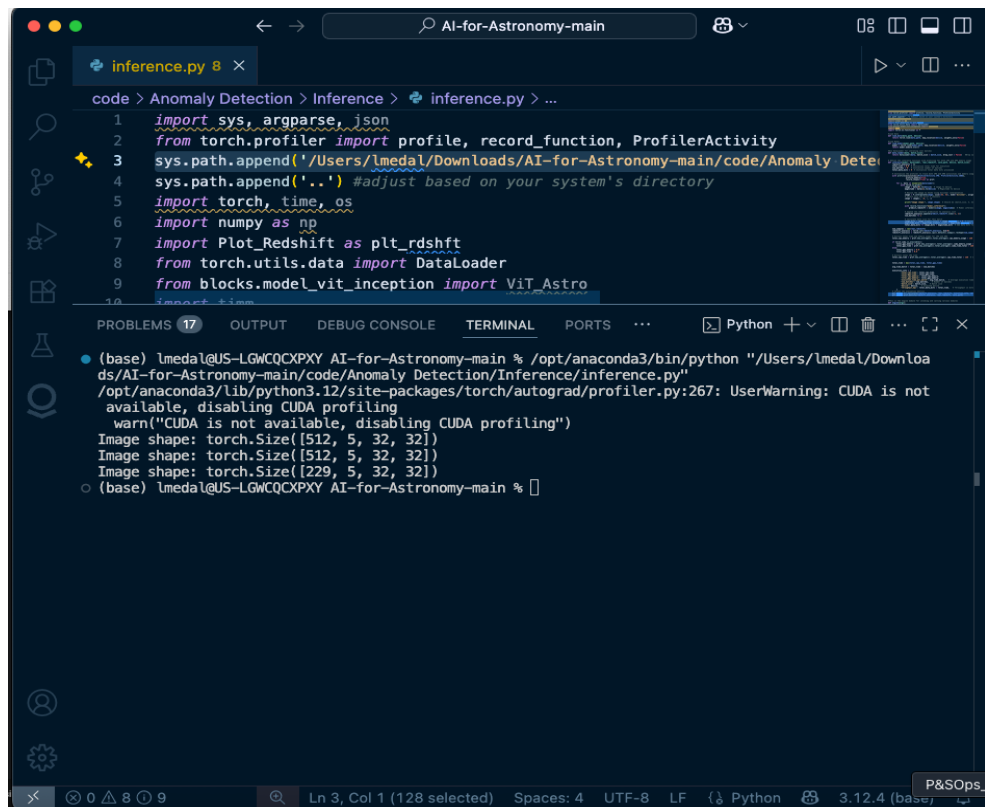
Team 5: Lionel Medal and Vicky Singh

Repository Cloning (Local VSCode)



```
code > Anomaly Detection > Inference > inference.py > ...
1 import sys, argparse, json
2 from torch.profiler import profile, record_function, ProfilerActivity
3 sys.path.append('/Users/lmedal/Downloads/AI-for-Astronomy-main/code/Anomaly Detection') #adjust based on your system's directory
4 sys.path.append('.') #adjust based on your system's directory
5 import torch, time, os
6 import numpy as np
7 import Plot_Redshift as plt_rdshft
8 from torch.utils.data import DataLoader
9 from blocks.model_vit_inception import ViT_Astro
10 import time
11 import torch.nn.functional as F
12
13 #Load Data
14 def load_data(data_path, device):
15     return torch.load(data_path, map_location=device, weights_only=False)
16
17 #Load Model
18 def load_model(model_path, device):
19     model = torch.load(model_path, map_location=device, weights_only=False)
20     return model.module.eval()
21
22 #Use DataLoader for iterating over batches
23 def data_loader(data, batch_size):
24     return DataLoader(data, batch_size=batch_size, drop_last=False) #Drop samples out of the batch size
25
26
27 # Define the inference function with profiling for both CPU and GPU memory usage
28 def inference(model, dataloader, real_redshift, save_path, device, batch_size):
29     redshift_analysis = {}
30     total_time = 0.0 # Initialize total time for execution
31     num_batches = 0 # Initialize number of batches
32     total_data_bits = 0 # Initialize total data bits processed
33
34     # Initialize the profiler to track both CPU and GPU activities and memory usage
35     with profile(activities=[ProfilerActivity.CPU, ProfilerActivity.CUDA],
36                 profile_memory=True,
37                 record_shapes=True) as prof:
38         for i, data in enumerate(dataloader):
39             start_time = time.time()
40             # Perform inference
41             output = model(data.to(device))
42             # Calculate redshift
43             redshift = plt_rdshft(output)
44             # Save results
45             torch.save(output, save_path + f'batch_{i}.pt')
46             total_time += time.time() - start_time
47             num_batches += 1
48             total_data_bits += data.size()[0] * data.size()[1] * data.size()[2] * data.size()[3] * 4
```

File Path Updates in 'inference.py' (Downloads Folder Highlighted)



```
code > Anomaly Detection > Inference > inference.py > ...
1 import sys, argparse, json
2 from torch.profiler import profile, record_function, ProfilerActivity
3 sys.path.append('/Users/lmedal/Downloads/AI-for-Astronomy-main/code/Anomaly Detection') #adjust based on your system's directory
4 sys.path.append('.') #adjust based on your system's directory
5 import torch, time, os
6 import numpy as np
7 import Plot_Redshift as plt_rdshft
8 from torch.utils.data import DataLoader
9 from blocks.model_vit_inception import ViT_Astro
10 import time
11 import torch.nn.functional as F
12
13 #Load Data
14 def load_data(data_path, device):
15     return torch.load(data_path, map_location=device, weights_only=False)
16
17 #Load Model
18 def load_model(model_path, device):
19     model = torch.load(model_path, map_location=device, weights_only=False)
20     return model.module.eval()
21
22 #Use DataLoader for iterating over batches
23 def data_loader(data, batch_size):
24     return DataLoader(data, batch_size=batch_size, drop_last=False) #Drop samples out of the batch size
25
26
27 # Define the inference function with profiling for both CPU and GPU memory usage
28 def inference(model, dataloader, real_redshift, save_path, device, batch_size):
29     redshift_analysis = {}
30     total_time = 0.0 # Initialize total time for execution
31     num_batches = 0 # Initialize number of batches
32     total_data_bits = 0 # Initialize total data bits processed
33
34     # Initialize the profiler to track both CPU and GPU activities and memory usage
35     with profile(activities=[ProfilerActivity.CPU, ProfilerActivity.CUDA],
36                 profile_memory=True,
37                 record_shapes=True) as prof:
38         for i, data in enumerate(dataloader):
39             start_time = time.time()
40             # Perform inference
41             output = model(data.to(device))
42             # Calculate redshift
43             redshift = plt_rdshft(output)
44             # Save results
45             torch.save(output, save_path + f'batch_{i}.pt')
46             total_time += time.time() - start_time
47             num_batches += 1
48             total_data_bits += data.size()[0] * data.size()[1] * data.size()[2] * data.size()[3] * 4
```

Documentation of Execution Time

The inference process was executed on a CPU using a batch size of 512. According to the Results.json output, the total CPU processing time across all batches was 9.56 seconds, with an average execution time per batch of 3.19 seconds. No GPU resources were utilized during this run, with the reported total GPU time being 0.0 seconds. The system achieved a throughput of approximately 21.5 million bits per second, demonstrating efficient performance for CPU-based execution under the given configuration.

Output Files

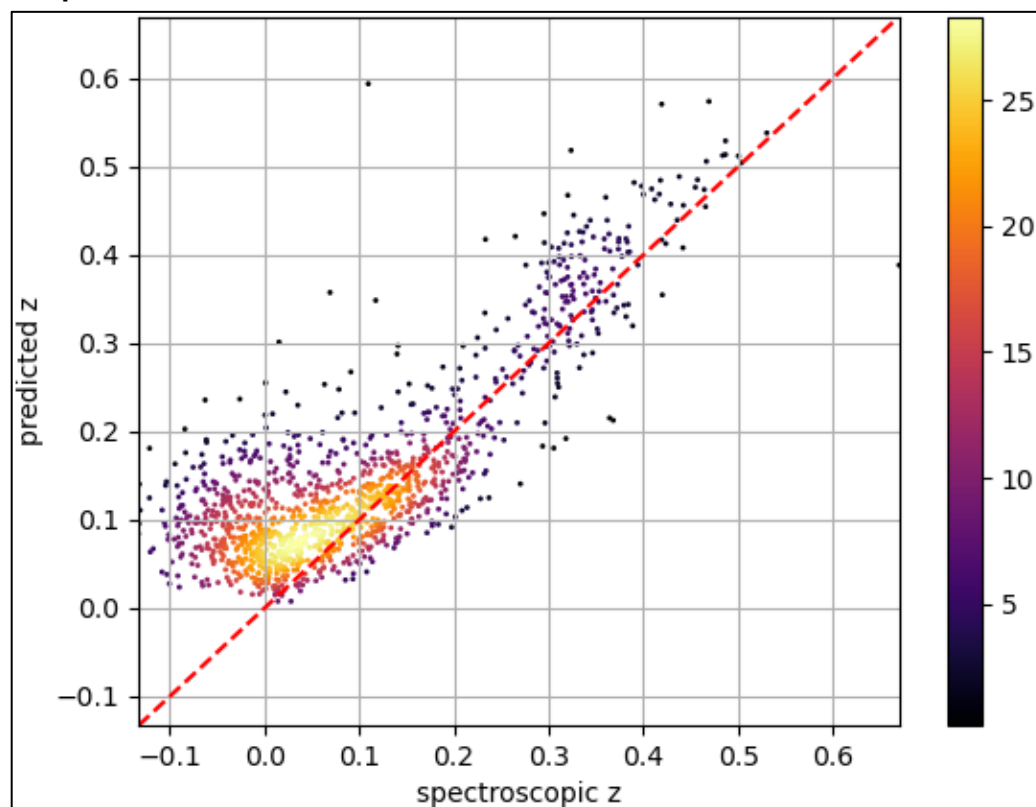


Figure 1. Inference Output File

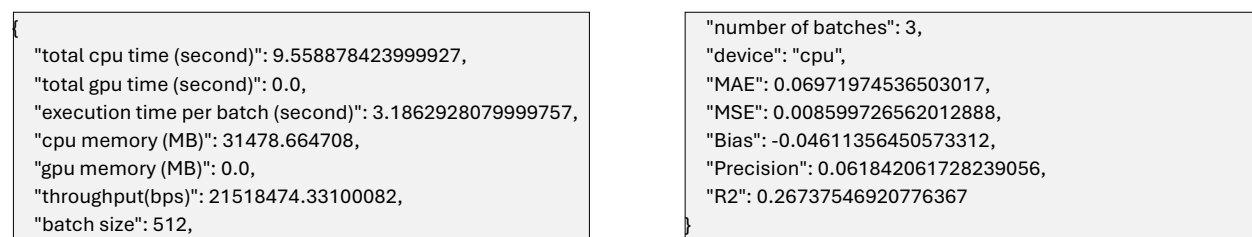


Figure 2. Results JSON Output File

Analysis of Inference Performance

- **Mean Absolute Error (MAE): 0.0697**
Indicates the average absolute deviation between predicted and actual redshift values.
- **Mean Squared Error (MSE): 0.0086**
Measures the average of the squared errors, placing greater emphasis on larger deviations.

- **Bias:** -0.046
Reflects a slight tendency of the model to underpredict redshift values.
- **Precision (NMAD):** 0.0618
Suggests a moderate spread in the prediction errors, based on the normalized median absolute deviation.
- **R² Score:** 0.26
Indicates that approximately 26.7% of the variance in true redshift values is explained by the model. This relatively low score highlights room for further optimization or model refinement.

Deployment Options Comparison

Batch Size	Total Time (s)	Memory (MB)	Throughput (bps)	No. of Batches	Device
128	10.78	31,186.56	19,082,229	10	CPU
256	10.07	31,433.14	20,419,591	5	CPU
512	9.56	31,478.66	21,518,474	3	CPU
1024	9.87	31,651.86	20,834,646	2	CPU

Troubleshooting and Resolutions

Several issues were encountered and resolved during implementation:

- **Input Shape Mismatch**
The model was configured to accept images with 5 channels and a resolution of 32×32, while the input data initially contained 64 channels and a different size. This was addressed by resizing the images to 32×32 and selecting only the first 5 channels for processing.
- **CUDA/GPU Profiling Errors**
Running on a CPU-only system caused errors when the code attempted to access GPU attributes. The script was updated to check for CUDA availability and default to zero for GPU-related metrics when unavailable.
- **FileNotFoundError for Output Directory**
The script initially failed when the specified output directory did not exist. This was resolved by adding logic to create the directory automatically before writing output files.
- **Path and Environment Adjustments**
File paths within inference.py were modified to align with the local environment. Additionally, any missing Python dependencies were installed to ensure successful execution.