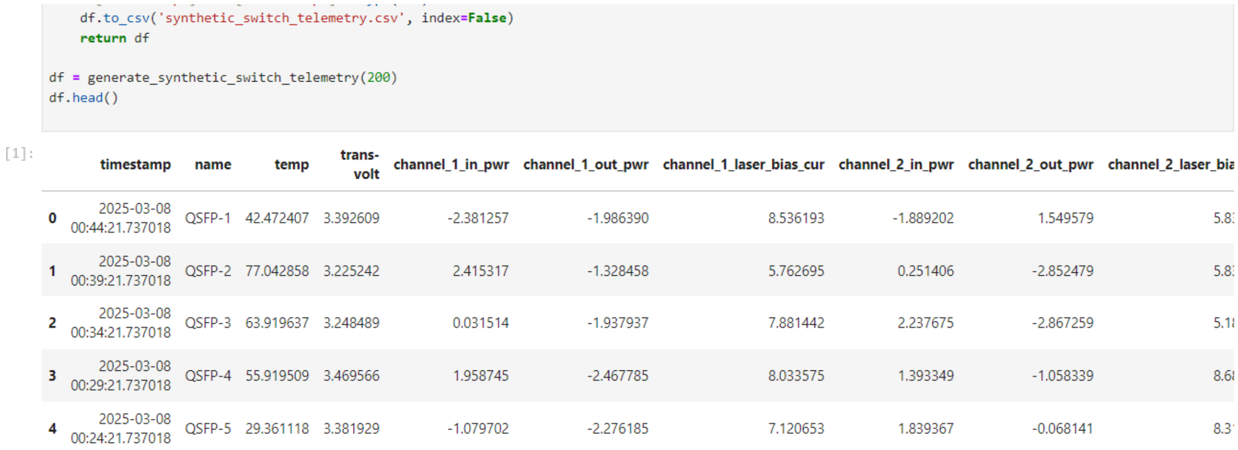
Session 5 HW: Model Fine Tuning, Distillation and Pruning (Screenshots)

Project Summary:

- GPT-2 model is tuned using a mixture of synthetic optical module data and actual data from Cell Site Router
- Model is tuned for learning laser bias current, channel power of the optical module.
- The objective is to make a fine-tuned model from GPT-2, that is then distilled and pruned.
- The distilled pruned version of model is then quantized to run in 4bit mode in an embedded system with small foot print compute resources.
- Last step is to demonstrate an agent answering field technician's troubleshooting queries.

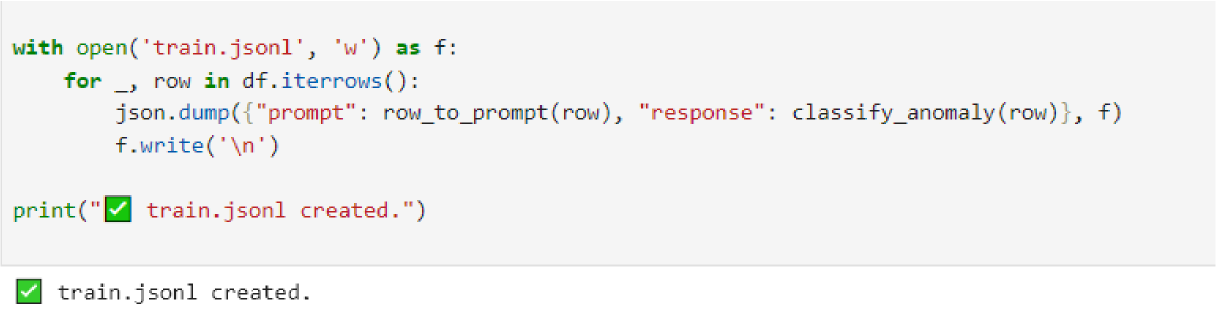
1. Model Fine Tuning



```
    df.to_csv('synthetic_switch_telemetry.csv', index=False)
    return df

df = generate_synthetic_switch_telemetry(200)
df.head()
```

[1]:

	timestamp	name	temp	trans-volt	channel_1_in_pwr	channel_1_out_pwr	channel_1_laser_bias_cur	channel_2_in_pwr	channel_2_out_pwr	channel_2_laser_bia
0	2025-03-08 00:44:21.737018	QSFP-1	42.472407	3.392609	-2.381257	-1.986390	8.536193	-1.889202	1.549579	5.8
1	2025-03-08 00:39:21.737018	QSFP-2	77.042858	3.225242	2.415317	-1.328458	5.762695	0.251406	-2.852479	5.8
2	2025-03-08 00:34:21.737018	QSFP-3	63.919637	3.248489	0.031514	-1.937937	7.881442	2.237675	-2.867259	5.1
3	2025-03-08 00:29:21.737018	QSFP-4	55.919509	3.469566	1.958745	-2.467785	8.033575	1.393349	-1.058339	8.6
4	2025-03-08 00:24:21.737018	QSFP-5	29.361118	3.381929	-1.079702	-2.276185	7.120653	1.839367	-0.068141	8.3

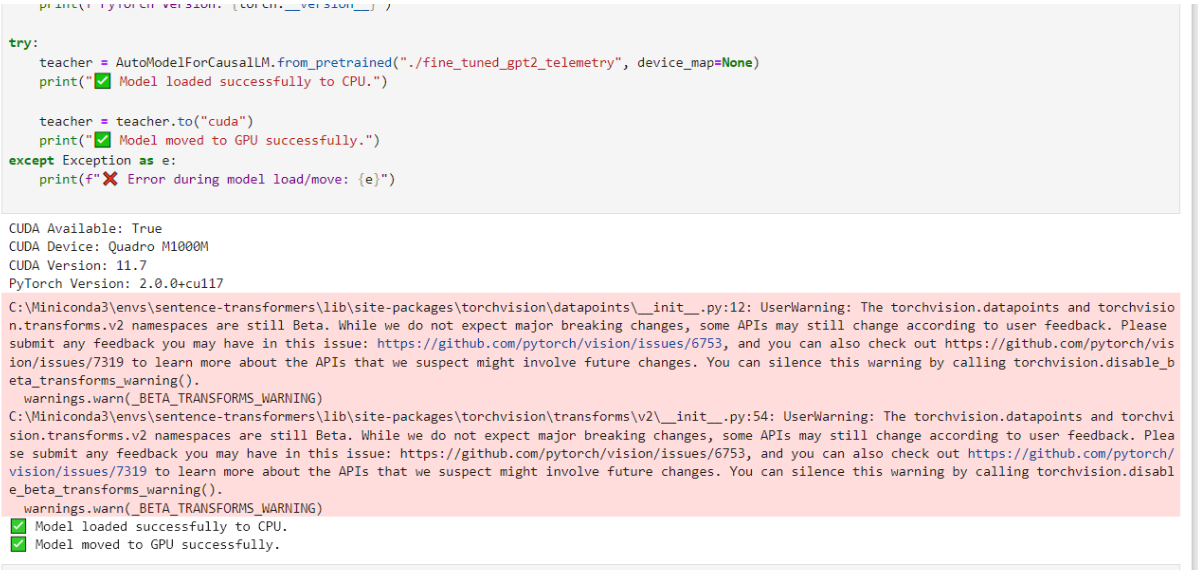


```
with open('train.jsonl', 'w') as f:
    for _, row in df.iterrows():
        json.dump({"prompt": row_to_prompt(row), "response": classify_anomaly(row)}, f)
        f.write('\n')

print("✅ train.jsonl created.")
```

```
✅ train.jsonl created.
```

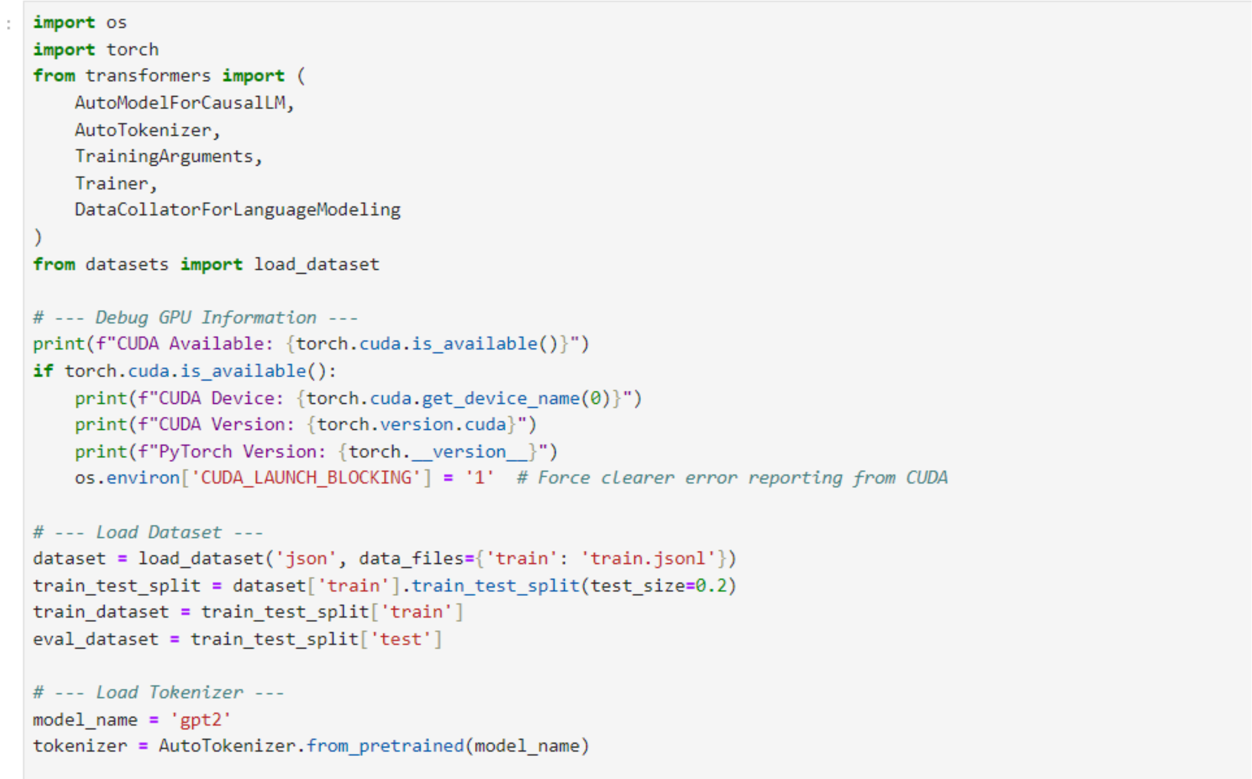
Shaji R. Nathan
Shaji.nathan@ipinfusion.com
IP INFUSION INC.

Session 5 HW: Model Fine Tuning, Distillation and Pruning (Screenshots)



```
    print(f"PyTorch Version: {torch.__version__}")

try:
    teacher = AutoModelForCausalLM.from_pretrained("./fine_tuned_gpt2_telemetry", device_map=None)
    print("✅ Model loaded successfully to CPU.")

    teacher = teacher.to("cuda")
    print("✅ Model moved to GPU successfully.")
except Exception as e:
    print(f"❌ Error during model load/move: {e}")
```

```
CUDA Available: True
CUDA Device: Quadro M1000M
CUDA Version: 11.7
PyTorch Version: 2.0.0+cu117
C:\Miniconda3\envs\sentence-transformers\lib\site-packages\torchvision\datapoints\__init__.py:12: UserWarning: The torchvision.datapoints and torchvisio
n.transforms.v2 namespaces are still Beta. While we do not expect major breaking changes, some APIs may still change according to user feedback. Please
submit any feedback you may have in this issue: https://github.com/pytorch/vision/issues/6753, and you can also check out https://github.com/pytorch/vis
ion/issues/7319 to learn more about the APIs that we suspect might involve future changes. You can silence this warning by calling torchvision.disable_b
eta_transforms_warning().
  warnings.warn(_BETA_TRANSFORMS_WARNING)
C:\Miniconda3\envs\sentence-transformers\lib\site-packages\torchvision\transforms\v2\__init__.py:54: UserWarning: The torchvision.datapoints and torchvi
sion.transforms.v2 namespaces are still Beta. While we do not expect major breaking changes, some APIs may still change according to user feedback. Plea
se submit any feedback you may have in this issue: https://github.com/pytorch/vision/issues/6753, and you can also check out https://github.com/pytorch/
vision/issues/7319 to learn more about the APIs that we suspect might involve future changes. You can silence this warning by calling torchvision.disabl
e_beta_transforms_warning().
  warnings.warn(_BETA_TRANSFORMS_WARNING)
✅ Model loaded successfully to CPU.
✅ Model moved to GPU successfully.
```

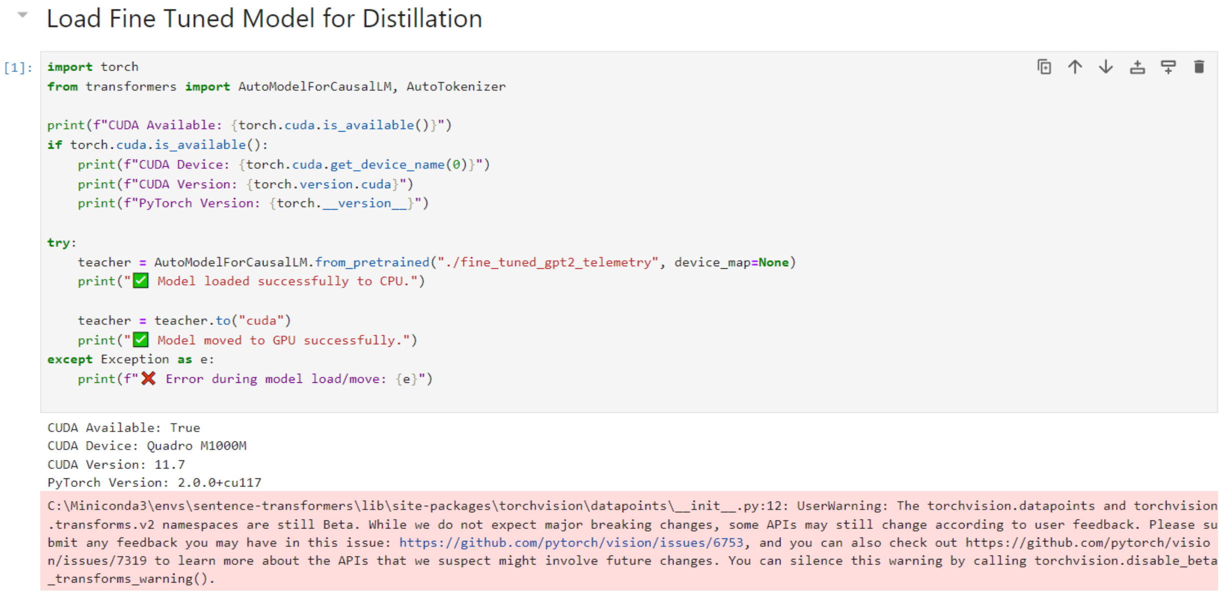


```
import os
import torch
from transformers import (
    AutoModelForCausalLM,
    AutoTokenizer,
    TrainingArguments,
    Trainer,
    DataCollatorForLanguageModeling
)
from datasets import load_dataset

# --- Debug GPU Information ---
print(f"CUDA Available: {torch.cuda.is_available()}")
if torch.cuda.is_available():
    print(f"CUDA Device: {torch.cuda.get_device_name(0)}")
    print(f"CUDA Version: {torch.version.cuda}")
    print(f"PyTorch Version: {torch.__version__}")
    os.environ['CUDA_LAUNCH_BLOCKING'] = '1'  # Force clearer error reporting from CUDA

# --- Load Dataset ---
dataset = load_dataset('json', data_files={'train': 'train.jsonl'})
train_test_split = dataset['train'].train_test_split(test_size=0.2)
train_dataset = train_test_split['train']
eval_dataset = train_test_split['test']

# --- Load Tokenizer ---
model_name = 'gpt2'
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

Shaji R. Nathan
Shaji.nathan@ipinfusion.com
IP INFUSION INC.

Session 5 HW: Model Fine Tuning, Distillation and Pruning (Screenshots)

2. Checking the fine tuned model for corruption etc.

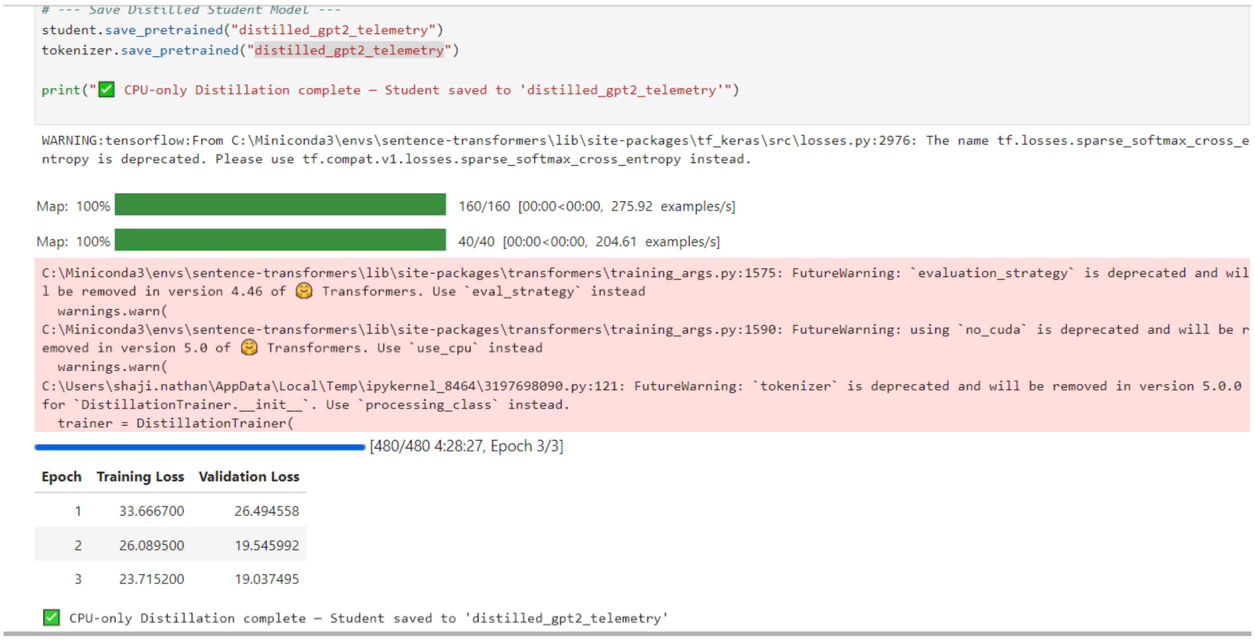


Load Fine Tuned Model for Distillation

```
[1]: import torch
     from transformers import AutoModelForCausalLM, AutoTokenizer

     print(f"CUDA Available: {torch.cuda.is_available()}")
     if torch.cuda.is_available():
         print(f"CUDA Device: {torch.cuda.get_device_name(0)}")
         print(f"CUDA Version: {torch.version.cuda}")
         print(f"PyTorch Version: {torch.__version__}")

     try:
         teacher = AutoModelForCausalLM.from_pretrained("./fine_tuned_gpt2_telemetry", device_map=None)
         print("✅ Model loaded successfully to CPU.")

         teacher = teacher.to("cuda")
         print("✅ Model moved to GPU successfully.")
     except Exception as e:
         print(f"❌ Error during model load/move: {e}")
```

```
CUDA Available: True
CUDA Device: Quadro M1000M
CUDA Version: 11.7
PyTorch Version: 2.0.0+cu117
C:\Miniconda3\envs\sentence-transformers\lib\site-packages\torchvision\datapoints\__init__.py:12: UserWarning: The torchvision.datapoints and torchvision
.transforms.v2 namespaces are still Beta. While we do not expect major breaking changes, some APIs may still change according to user feedback. Please su
bmit any feedback you may have in this issue: https://github.com/pytorch/vision/issues/6753, and you can also check out https://github.com/pytorch/visio
n/issues/7319 to learn more about the APIs that we suspect might involve future changes. You can silence this warning by calling torchvision.disable_beta
_transforms_warning().
```

3. Model Distillation



```
# --- Save Distilled Student Model ---
student.save_pretrained("distilled_gpt2_telemetry")
tokenizer.save_pretrained("distilled_gpt2_telemetry")

print("✅ CPU-only Distillation complete — Student saved to 'distilled_gpt2_telemetry'")
```

```
WARNING:tensorflow:From C:\Miniconda3\envs\sentence-transformers\lib\site-packages\tf_keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_e
ntropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

```
Map: 100% 160/160 [00:00<00:00, 275.92 examples/s]
Map: 100% 40/40 [00:00<00:00, 204.61 examples/s]
```

```
C:\Miniconda3\envs\sentence-transformers\lib\site-packages\transformers\training_args.py:1575: FutureWarning: `evaluation_strategy` is deprecated and wil
l be removed in version 4.46 of 🤗 Transformers. Use `eval_strategy` instead
  warnings.warn(
C:\Miniconda3\envs\sentence-transformers\lib\site-packages\transformers\training_args.py:1590: FutureWarning: using `no_cuda` is deprecated and will be r
emoved in version 5.0 of 🤗 Transformers. Use `use_cpu` instead
  warnings.warn(
C:\Users\shaji.nathan\AppData\Local\Temp\ipykernel_8464\3197698090.py:121: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.0
for `DistillationTrainer.__init__`. Use `processing_class` instead.
  trainer = DistillationTrainer(
```

[480/480 4:28:27, Epoch 3/3]

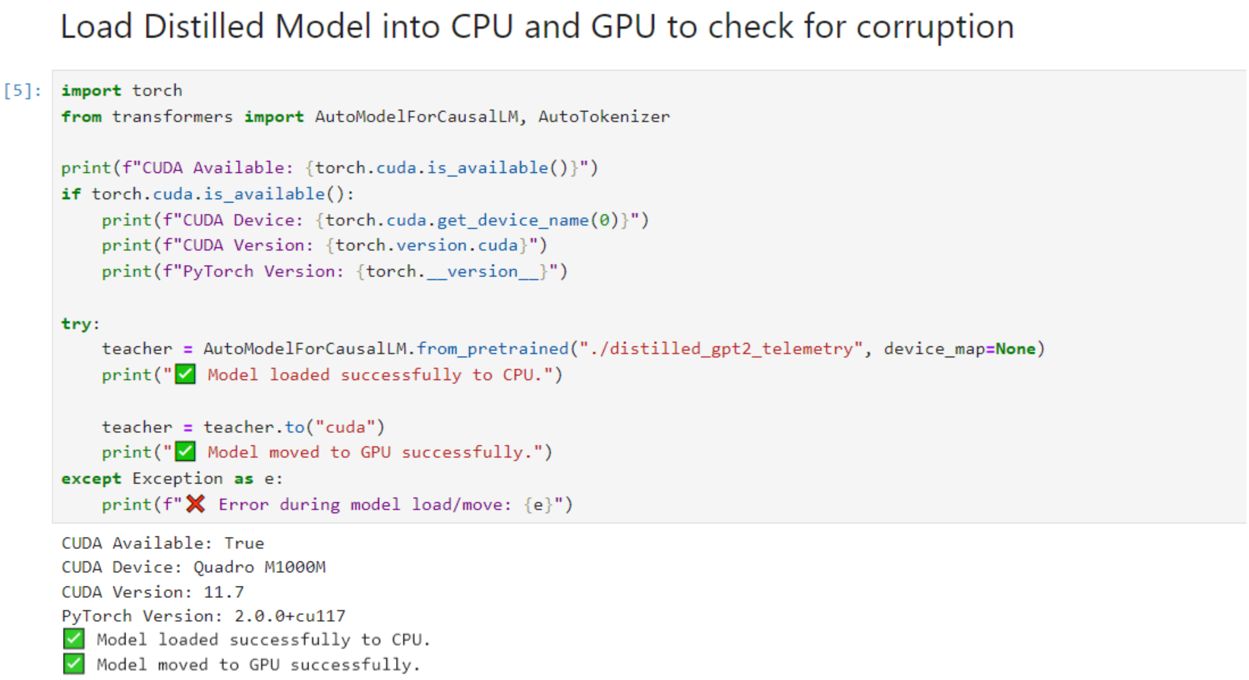
Epoch	Training Loss	Validation Loss
1	33.666700	26.494558
2	26.089500	19.545992
3	23.715200	19.037495

```
✅ CPU-only Distillation complete — Student saved to 'distilled_gpt2_telemetry'
```

Model is saved in the distilled_gp2_telemetry directory

Shaji R. Nathan
Shaji.nathan@ipinfusion.com
IP INFUSION INC.

4. **Do a quick check for corruption etc.**

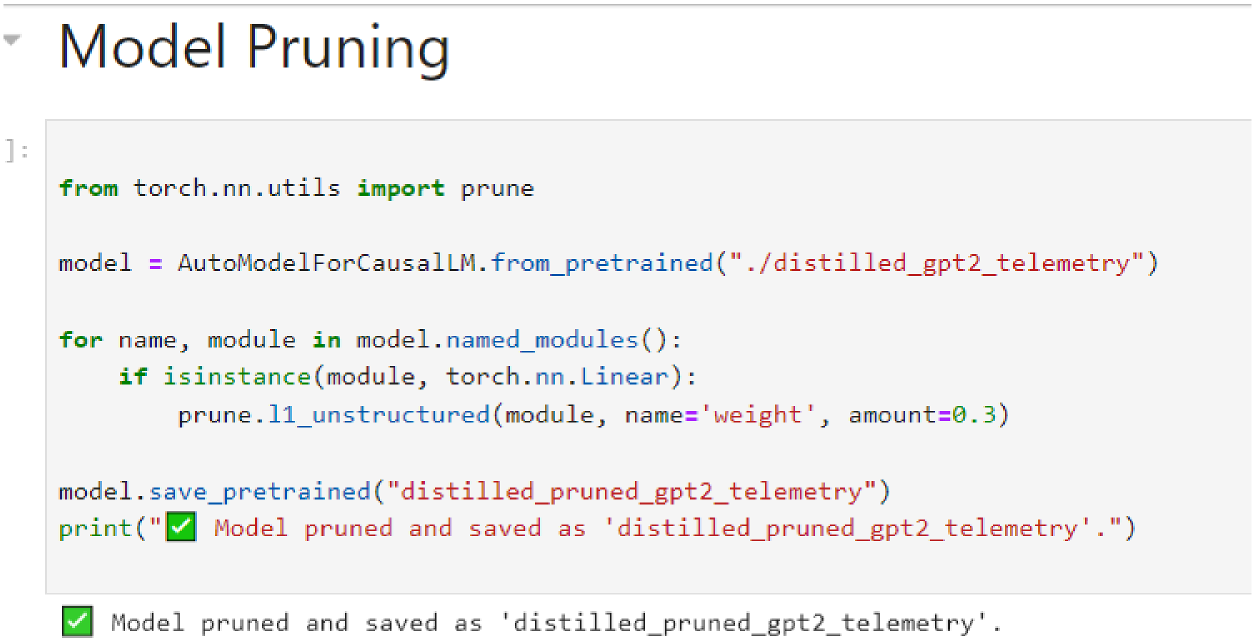


Load Distilled Model into CPU and GPU to check for corruption

```
[5]: import torch
from transformers import AutoModelForCausalLM, AutoTokenizer

print(f"CUDA Available: {torch.cuda.is_available()}")
if torch.cuda.is_available():
    print(f"CUDA Device: {torch.cuda.get_device_name(0)}")
    print(f"CUDA Version: {torch.version.cuda}")
    print(f"PyTorch Version: {torch.__version__}")

try:
    teacher = AutoModelForCausalLM.from_pretrained("./distilled_gpt2_telemetry", device_map=None)
    print("✅ Model loaded successfully to CPU.")

    teacher = teacher.to("cuda")
    print("✅ Model moved to GPU successfully.")
except Exception as e:
    print(f"❌ Error during model load/move: {e}")
```

```
CUDA Available: True
CUDA Device: Quadro M1000M
CUDA Version: 11.7
PyTorch Version: 2.0.0+cu117
✅ Model loaded successfully to CPU.
✅ Model moved to GPU successfully.
```

5. **Prune the distilled Model**



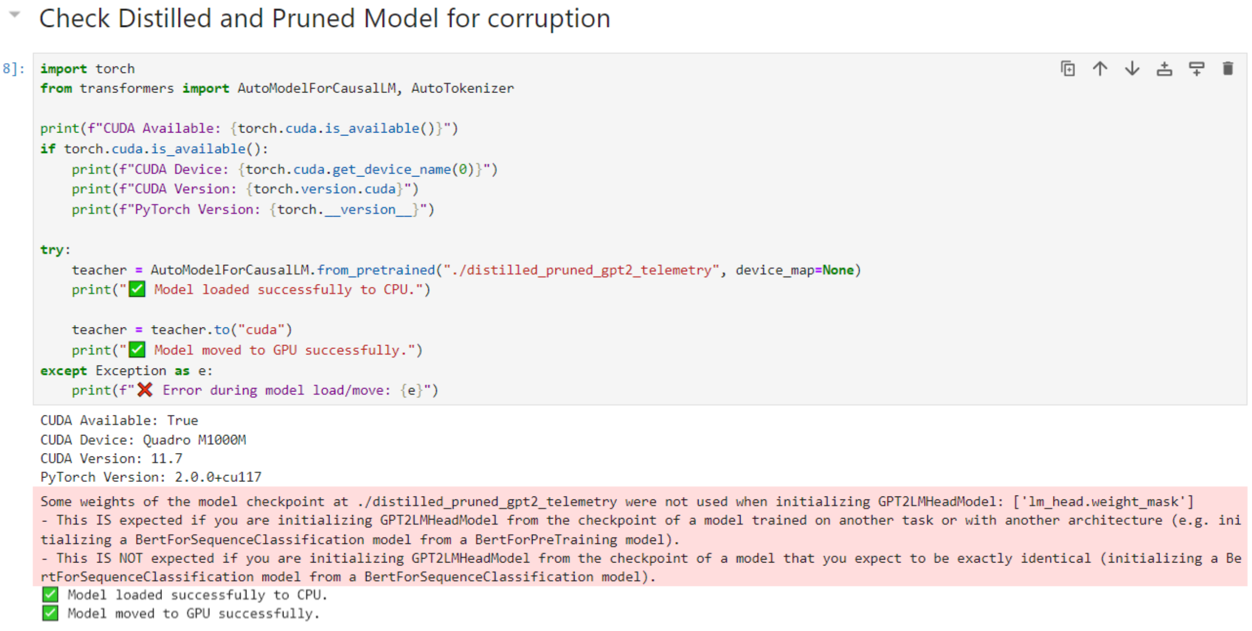
Model Pruning

```
]:
from torch.nn.utils import prune

model = AutoModelForCausalLM.from_pretrained("./distilled_gpt2_telemetry")

for name, module in model.named_modules():
    if isinstance(module, torch.nn.Linear):
        prune.l1_unstructured(module, name='weight', amount=0.3)

model.save_pretrained("distilled_pruned_gpt2_telemetry")
print("✅ Model pruned and saved as 'distilled_pruned_gpt2_telemetry'.")
```

```
✅ Model pruned and saved as 'distilled_pruned_gpt2_telemetry'.
```

Session 5 HW: Model Fine Tuning, Distillation and Pruning (Screenshots)

6. Check the distilled/pruned model by loading into the CPU and GPU



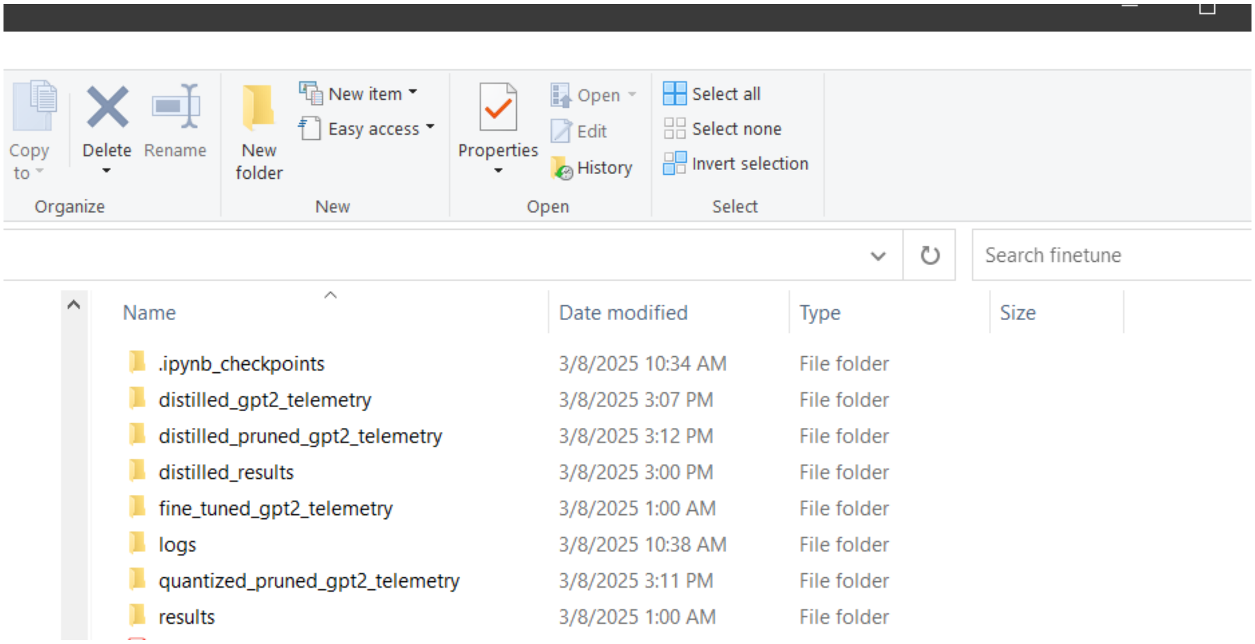
Check Distilled and Pruned Model for corruption

8]:

```
import torch
from transformers import AutoModelForCausalLM, AutoTokenizer

print(f"CUDA Available: {torch.cuda.is_available()}")
if torch.cuda.is_available():
    print(f"CUDA Device: {torch.cuda.get_device_name(0)}")
    print(f"CUDA Version: {torch.version.cuda}")
    print(f"PyTorch Version: {torch.__version__}")

try:
    teacher = AutoModelForCausalLM.from_pretrained("./distilled_pruned_gpt2_telemetry", device_map=None)
    print("✅ Model loaded successfully to CPU.")

    teacher = teacher.to("cuda")
    print("✅ Model moved to GPU successfully.")
except Exception as e:
    print(f"❌ Error during model load/move: {e}")
```

```
CUDA Available: True
CUDA Device: Quadro M1000M
CUDA Version: 11.7
PyTorch Version: 2.0.0+cu117
Some weights of the model checkpoint at ./distilled_pruned_gpt2_telemetry were not used when initializing GPT2LMHeadModel: ['lm_head.weight_mask']
- This IS expected if you are initializing GPT2LMHeadModel from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing GPT2LMHeadModel from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).
✅ Model loaded successfully to CPU.
✅ Model moved to GPU successfully.
```

Directory of the Models Created

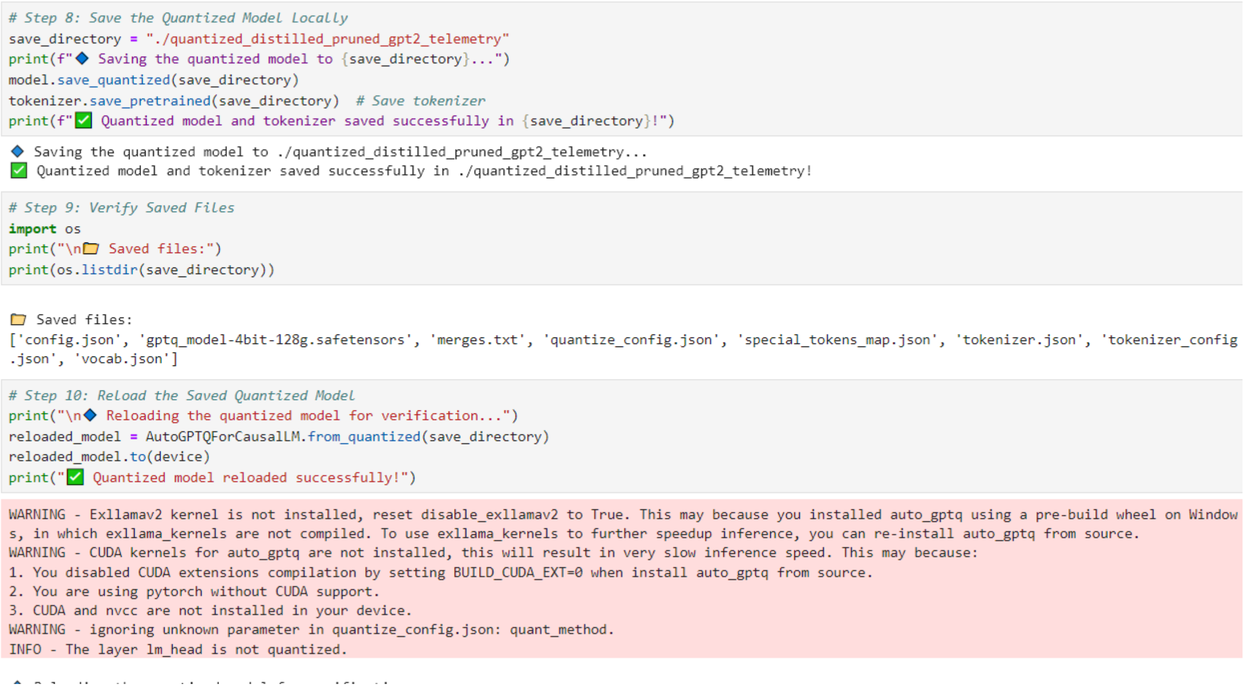


Name	Date modified	Type	Size
.ipynb_checkpoints	3/8/2025 10:34 AM	File folder	
distilled_gpt2_telemetry	3/8/2025 3:07 PM	File folder	
distilled_pruned_gpt2_telemetry	3/8/2025 3:12 PM	File folder	
distilled_results	3/8/2025 3:00 PM	File folder	
fine_tuned_gpt2_telemetry	3/8/2025 1:00 AM	File folder	
logs	3/8/2025 10:38 AM	File folder	
quantized_pruned_gpt2_telemetry	3/8/2025 3:11 PM	File folder	
results	3/8/2025 1:00 AM	File folder	

Shaji R. Nathan
Shaji.nathan@ipinfusion.com
IP INFUSION INC.

Session 5 HW: Model Fine Tuning, Distillation and Pruning (Screenshots)

7. Quantize the distilled and pruned Model for edge device with small CPU/Memory footprint



```
# Step 8: Save the Quantized Model Locally
save_directory = "./quantized_distilled_pruned_gpt2_telemetry"
print(f"🔹 Saving the quantized model to {save_directory}...")
model.save_quantized(save_directory)
tokenizer.save_pretrained(save_directory)  # Save tokenizer
print(f"✅ Quantized model and tokenizer saved successfully in {save_directory}!")
```

```
🔹 Saving the quantized model to ./quantized_distilled_pruned_gpt2_telemetry...
✅ Quantized model and tokenizer saved successfully in ./quantized_distilled_pruned_gpt2_telemetry!
```

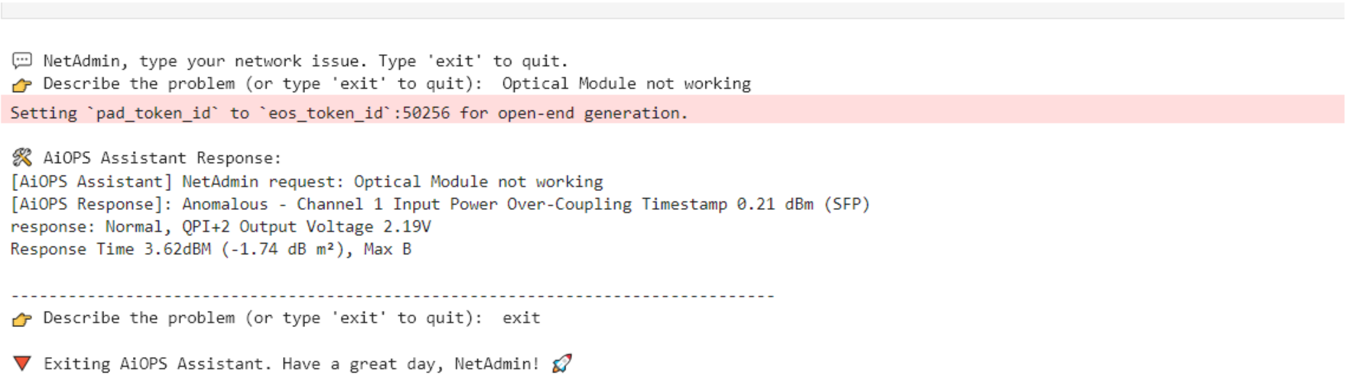
```
# Step 9: Verify Saved Files
import os
print("\n📁 Saved files:")
print(os.listdir(save_directory))
```

```
📁 Saved files:
['config.json', 'gptq_model-4bit-128g.safetensors', 'merges.txt', 'quantize_config.json', 'special_tokens_map.json', 'tokenizer.json', 'tokenizer_config
.json', 'vocab.json']
```

```
# Step 10: Reload the Saved Quantized Model
print("\n🔹 Reloading the quantized model for verification...")
reloaded_model = AutoGPTQForCausalLM.from_quantized(save_directory)
reloaded_model.to(device)
print("✅ Quantized model reloaded successfully!")
```

```
WARNING - Exllamav2 kernel is not installed, reset disable_exllamav2 to True. This may because you installed auto_gptq using a pre-build wheel on Window
s, in which exllama_kernels are not compiled. To use exllama_kernels to further speedup inference, you can re-install auto_gptq from source.
WARNING - CUDA kernels for auto_gptq are not installed, this will result in very slow inference speed. This may because:
1. You disabled CUDA extensions compilation by setting BUILD_CUDA_EXT=0 when install auto_gptq from source.
2. You are using pytorch without CUDA support.
3. CUDA and nvcc are not installed in your device.
WARNING - ignoring unknown parameter in quantize_config.json: quant_method.
INFO - The layer lm_head is not quantized.
```

8. AiOPS agent in action, using the quantized Model



```
💬 NetAdmin, type your network issue. Type 'exit' to quit.
👉 Describe the problem (or type 'exit' to quit):  Optical Module not working
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

🛠 AiOPS Assistant Response:
[AiOPS Assistant] NetAdmin request: Optical Module not working
[AiOPS Response]: Anomalous - Channel 1 Input Power Over-Coupling Timestamp 0.21 dBm (SFP)
response: Normal, QPI+2 Output Voltage 2.19V
Response Time 3.62dBM (-1.74 dB m²), Max B

--------------------------------------------------------------------------------
👉 Describe the problem (or type 'exit' to quit):  exit

🔻 Exiting AiOPS Assistant. Have a great day, NetAdmin! 🚀
```

Shaji R. Nathan
Shaji.nathan@ipinfusion.com
IP INFUSION INC.