Scalability of Large Language Models

Dr. Nikos Bakas

December 18, 2025

Contents

Part 1

Scaling LLMs

1.1 GPU Monitoring Stack

1.1.1 NVML (NVIDIA Management Library)

- *Definition:* C API for monitoring and managing NVIDIA GPU state (utilization, memory, temperature, power, clocks, ECC, etc.); it is also the underlying library used by `nvidia-smi`.
- *Links:*
  - https://developer.nvidia.com/management-library-nvml
  - https://docs.nvidia.com/deploy/nvml-api/
- *Developed by:* NVIDIA.
- *Methods (API):*
  - `nvmlInit` — initialize NVML (must be called before queries).
  - `nvmlShutdown` — shut down NVML when finished.
  - `nvmlDeviceGetHandleByIndex` — get a handle to a specific GPU (by index).
  - `nvmlDeviceGetUtilizationRates` — get GPU and memory-controller utilization (% over the sample window).
  - `nvmlDeviceGetMemoryInfo` — get total/free/used GPU memory.
  - `nvmlDeviceGetTemperature` — get current GPU temperature (e.g., core temperature).
  - `nvmlDeviceGetPowerUsage` — get current power draw (typically in milliwatts).

1.1.2 nvidia-smi (NVIDIA System Management Interface)

- *Definition:* A command-line tool (built on NVML) to monitor/query and (with privileges) manage NVIDIA GPUs; supports machine-readable output (e.g., CSV/XML) for scripting.
- *Links:*
  - https://developer.nvidia.com/system-management-interface
  - https://docs.nvidia.com/deploy/nvidia-smi/
- *Developed by:* NVIDIA.
- *CLI Commands:*
  - `nvidia-smi -query-gpu=<fields> -format=csv,noheader` — query specific metrics (e.g., `utilization.gpu`, `utilization.memory`, `memory.used`, `memory.total`, `temperature.gpu`, `power.draw`) in script-friendly CSV.
  - `nvidia-smi -i <gpu> ...` — select which GPU to query (useful on multi-GPU systems).
  - `nvidia-smi -l <seconds> ...` / `nvidia-smi -loop-ms <ms> ...` — repeat queries at a fixed interval to log utilization/memory/temperature/power over time.
  - `nvidia-smi -f <logfile> ...` — write the output to a log file.
  - `nvidia-smi dmon` — live per-GPU monitoring including utilization, memory activity/usage, temperature, and power (depending on support).

1.1.3 CUDA (Compute Unified Device Architecture)

- *Definition:* NVIDIA's GPU computing platform + APIs (runtime + driver) that let software allocate GPU memory, launch kernels, and manage devices/streams to run general-purpose computation on NVIDIA GPUs.
- *Links:*
  - https://docs.nvidia.com/cuda/
  - https://docs.nvidia.com/cuda/cuda-runtime-api/index.html
  - https://docs.nvidia.com/cuda/cuda-programming-guide/01-introduction/cuda-platform.html
- *Developed by:* NVIDIA.
- *Methods (API):*
  - `cudaMemGetInfo` — query global free and total device memory (bytes).
  - `cudaGetDeviceProperties` — get device properties including total global memory (`totalGlobalMem`).
  - `cudaMalloc` / `cudaFree` — allocate/free memory on the GPU (device RAM).
  - `cudaMemcpy` — copy data between host RAM and GPU RAM (or device-to-device).

1.1.4 torch.cuda (PyTorch CUDA package)

- *Definition:* PyTorch's CUDA interface for allocating CUDA tensors and running GPU ops; also exposes some *monitoring* helpers (some are reported "as given by `nvidia-smi`").
- *Link:* https://docs.pytorch.org/docs/stable/cuda.html
- *Developed by:* The PyTorch open-source project (PyTorch Foundation under the Linux Foundation).
- *Methods you can use:*
  - `torch.cuda.memory_allocated` — bytes of GPU memory currently allocated by PyTorch (this process).
  - `torch.cuda.max_memory_allocated` — peak allocated GPU memory by PyTorch (this process).
  - `torch.cuda.reset_peak_memory_stats` — reset PyTorch peak-memory statistics.
  - `torch.cuda.memory.mem_get_info` — global free/total GPU memory (bytes) from CUDA (`cudaMemGetInfo`).
  - `torch.cuda.utilization` — GPU utilization (% over a sampling window), reported "as given by `nvidia-smi`".
  - `torch.cuda.memory_usage` — memory-controller utilization (% over a sampling window), reported "as given by `nvidia-smi`".

  - `torch.cuda.temperature` — GPU temperature (average over the past sample period), reported "as given by `nvidia-smi`".
  - `torch.cuda.power_draw` — GPU power draw (average over the past sample period), reported "as given by `nvidia-smi`".

- `torch.cuda` vs NVML:
  - PyTorch's core `torch.cuda` is not NVML-based. The stuff that makes tensors run on the GPU uses CUDA (driver/runtime), not NVML.
  - Some `torch.cuda` info is explicitly CUDA-API-based, not NVML. For example `torch.cuda.memory.mem_get_info` uses `cudaMemGetInfo`.
  - But some PyTorch monitoring helpers are effectively NVML-based. Example: `torch.cuda.utilization()` is documented as returning utilization "as given by `nvidia-smi`" (which is NVML-backed)

1.1.5 nvidia-ml-py (official NVML Python bindings distribution)

- *Definition:* The Python package distribution that wraps the NVML library and provides the `pynvml` module for GPU management/monitoring from Python (i.e., you `pip install nvidia-ml-py` but you `import pynvml`). The separate `pynvml` (https://pypi.org/project/pynvml/) PyPI project is deprecated and provides unofficial utilities.
- *Link:* https://pypi.org/project/nvidia-ml-py/
- *Developed by:* NVIDIA.
- *Methods you can use:*
  - `pynvml.nvmlInit()` — initialize NVML (must be called before queries).
  - `pynvml.nvmlDeviceGetHandleByIndex()` — select a GPU (by index) to query.
  - `pynvml.nvmlDeviceGetMemoryInfo()` — get total/free/used GPU memory.
  - `pynvml.nvmlDeviceGetUtilizationRates()` — get GPU and memory-controller utilization (% over a sampling window).
  - `pynvml.nvmlDeviceGetTemperature()` — get current GPU temperature (e.g., core temperature).
  - `pynvml.nvmlDeviceGetPowerUsage()` — get current GPU power draw (typically in milliwatts).

1.1.6 How software measures "hardware" telemetry (utilization, memory, temperature, power/energy)

The GPU has a **built-in controller** (a tiny processor in the GPU *hardware*) that runs **firmware** (small *software* inside the GPU). That controller and the GPU hardware provide telemetry such as temperature (typically from on-chip **sensors**), power (from board power monitoring / telemetry), and activity counters.

The **NVIDIA kernel driver** (privileged software in the operating system) talks to the GPU, collects these readings (often as averages over a short sampling window), and exposes them to normal programs through standard interfaces such as NVML (and tools built on it, like nvidia-smi). Some memory info can also come from CUDA runtime queries (e.g., global free/total memory).

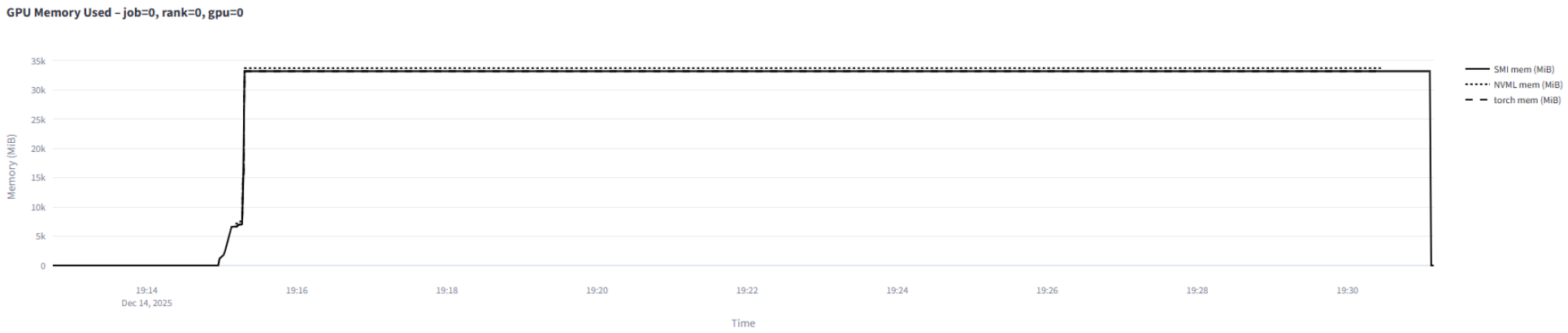
Part 2

GPU Utilization and Memory Usage

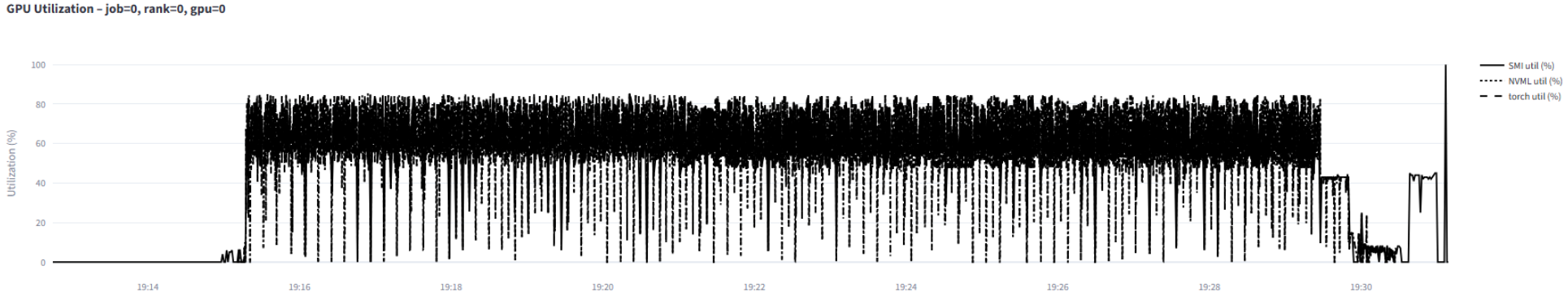
2.1 Task 0 - Model=meta-Llama-3.2-3B-Instruct

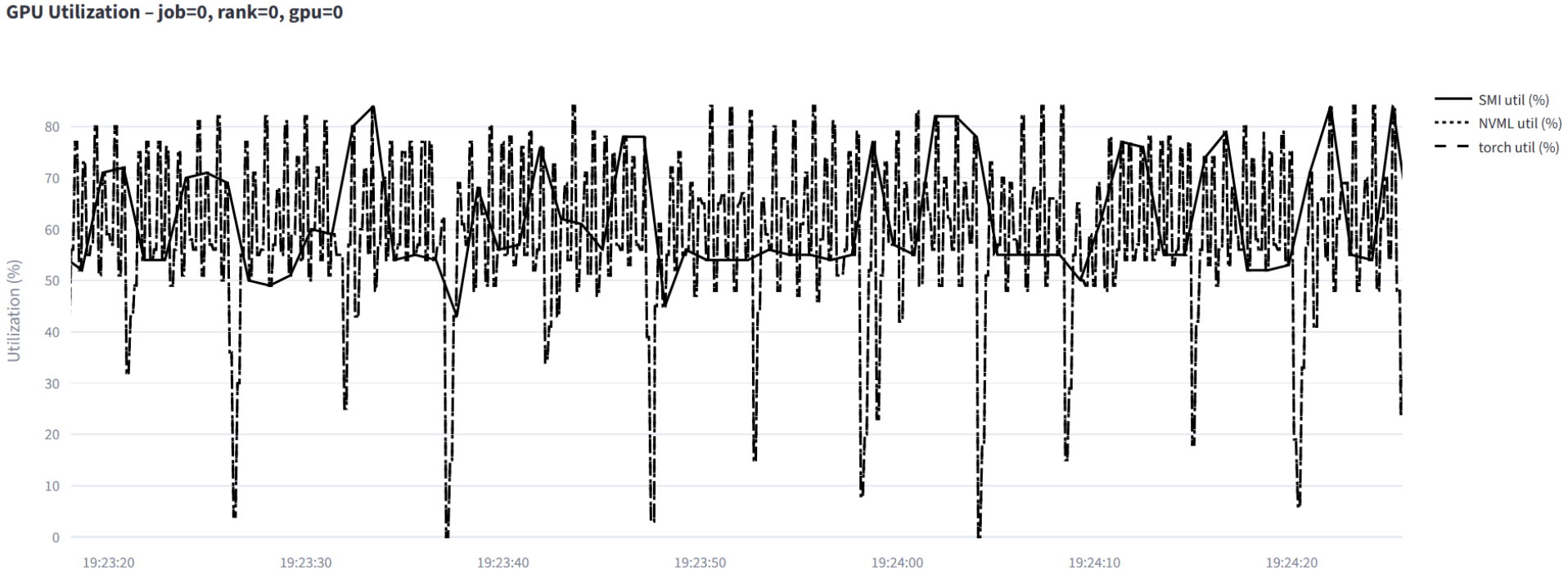
Batch=1, MaxLen=16, Epochs=1, GPUs=1, TrainLoss=1.549743680655956 4, ValLoss=1.29731655120849 6, TrainTime=917.2674236297609s

GPU Memory Used (MiB)



GPU Utilization (%)

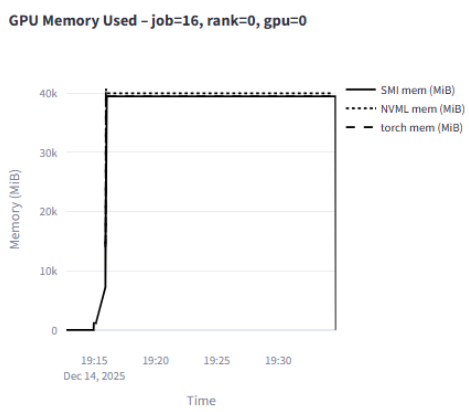


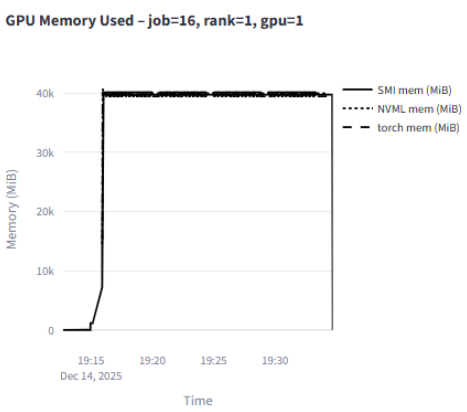

GPU Utilization – job=0, rank=0, gpu=0
SMI util (%)
NVML util (%)
torch util (%)
Utilization (%)
0
10
20
30
40
50
60
70
80
19:23:20
19:23:30
19:23:40
19:23:50
19:24:00
19:24:10
19:24:20

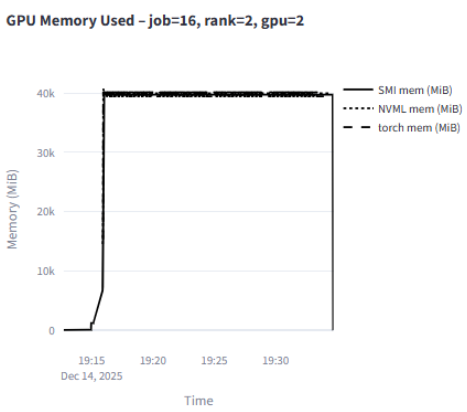
2.2 Task 16 - Model=meta-Llama-3.2-3B-Instruct

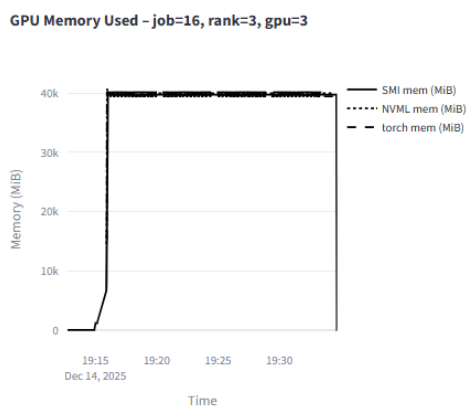
Batch=1, MaxLen=16, Epochs=4, GPUs=4, TrainLoss=1.224546863734722, ValLoss=0.9465489983558656, TrainTime=1109.7244033813477s

GPU Memory Used (MiB)

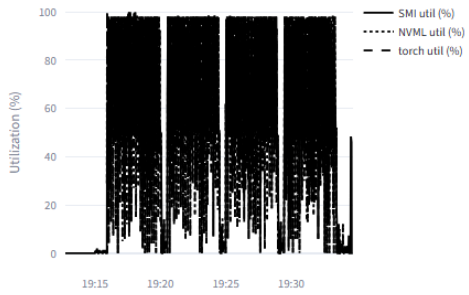


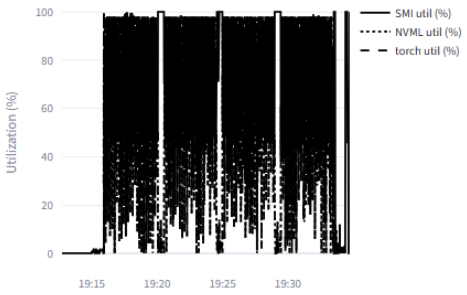


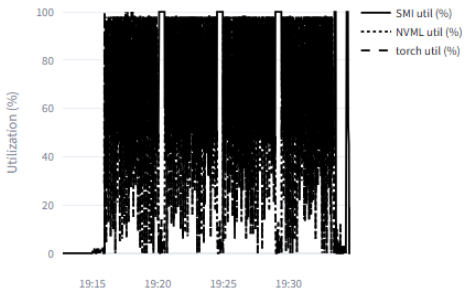


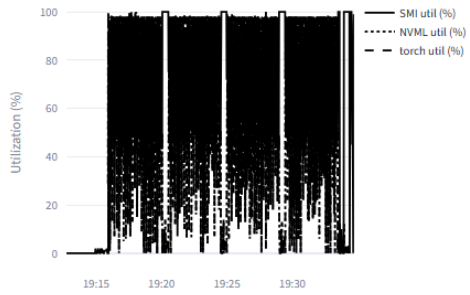


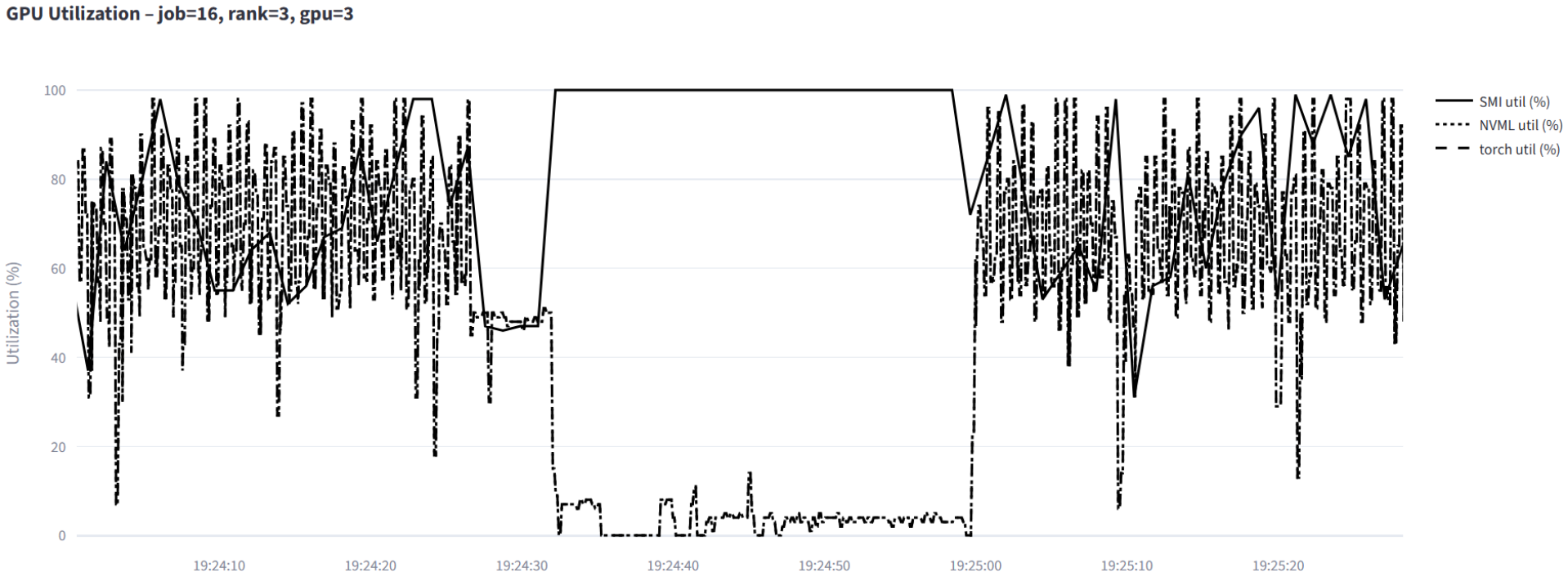
GPU Utilization (%)









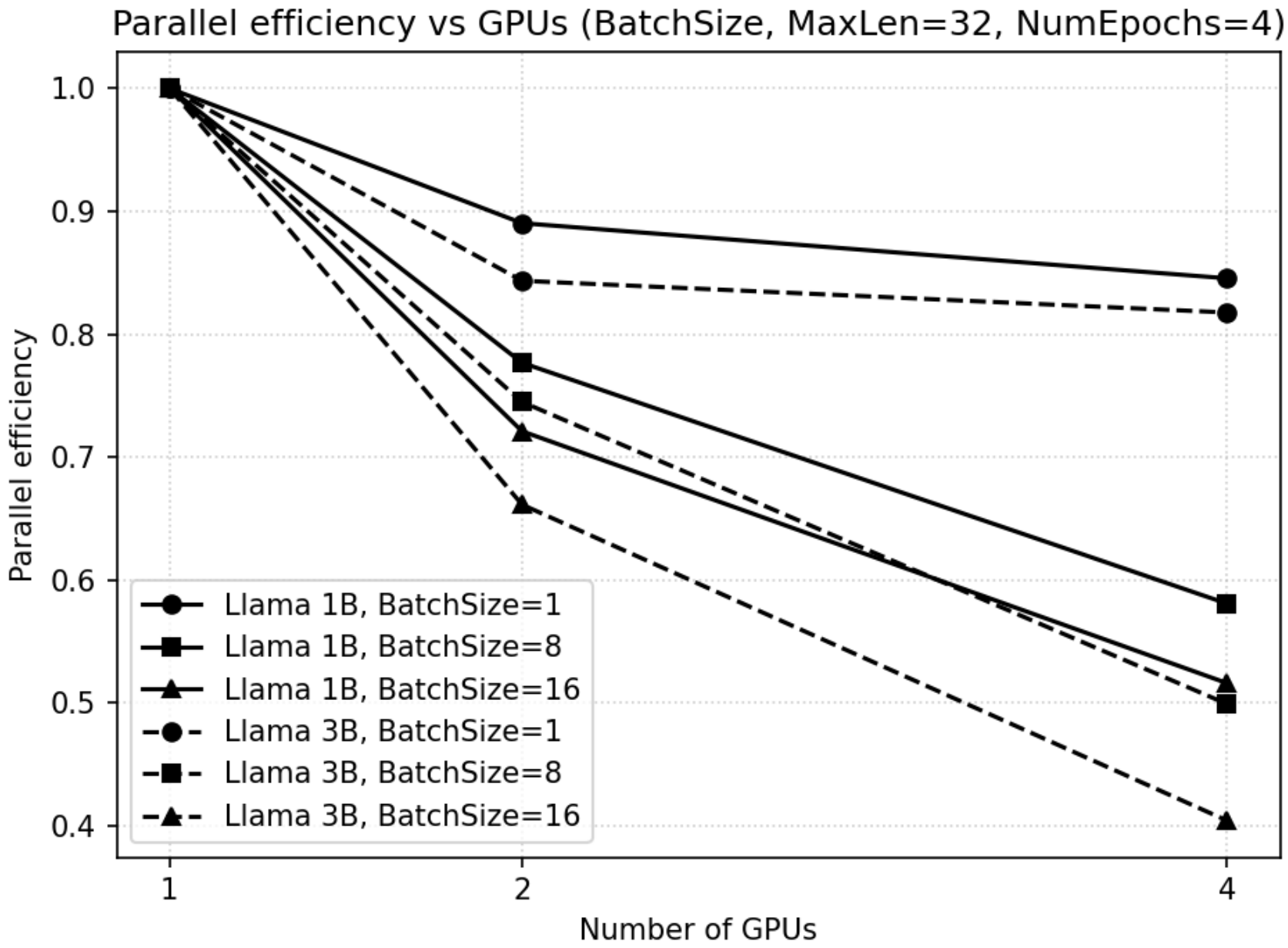

GPU Utilization – job=16, rank=3, gpu=3
Utilization (%)
100
80
60
40
20
0
19:24:10
19:24:20
19:24:30
19:24:40
19:24:50
19:25:00
19:25:10
19:25:20
SMI util (%)
NVML util (%)
torch util (%)

Part 3

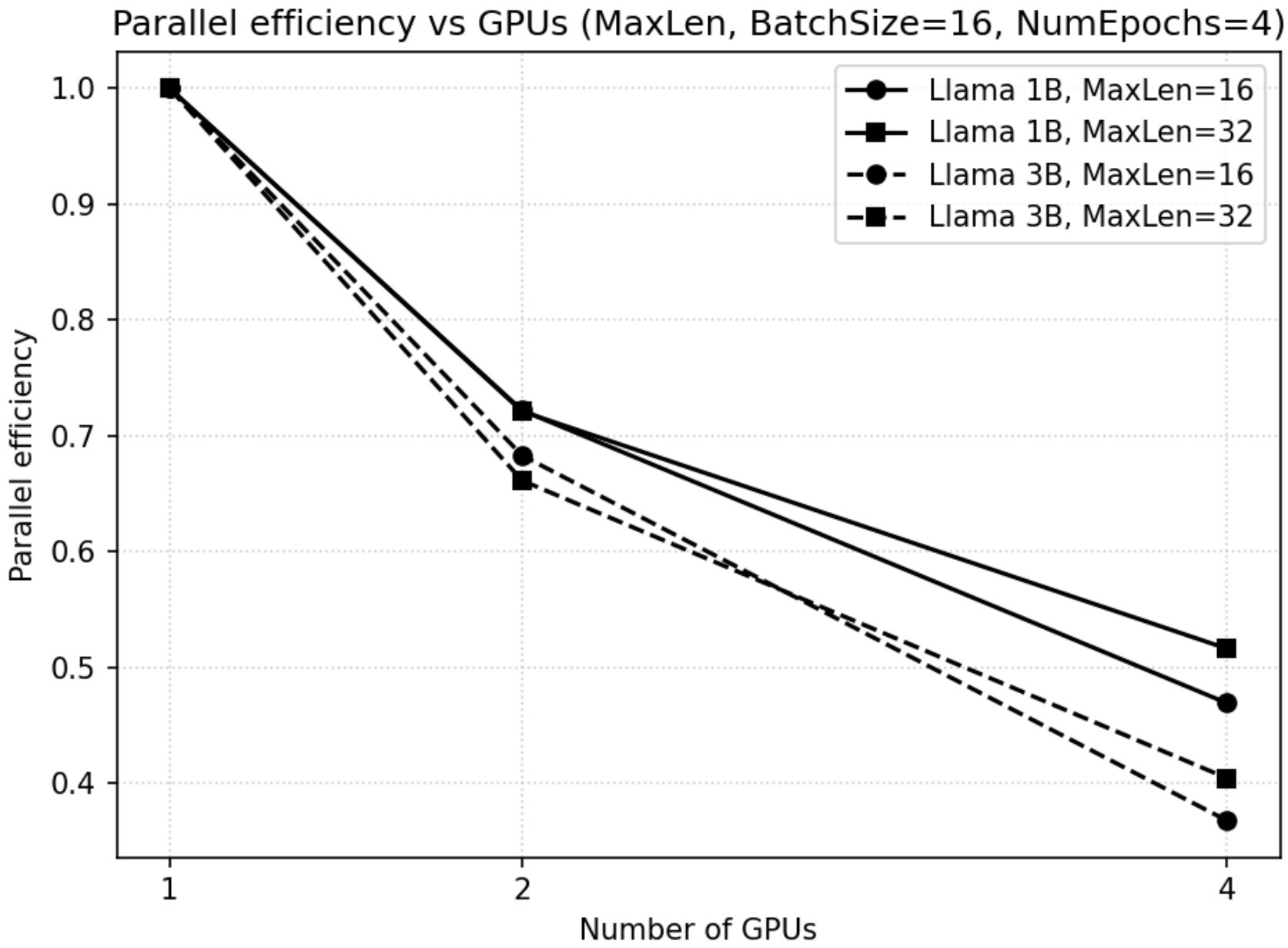
Scalability Plots

3.1 Indiccative Plots

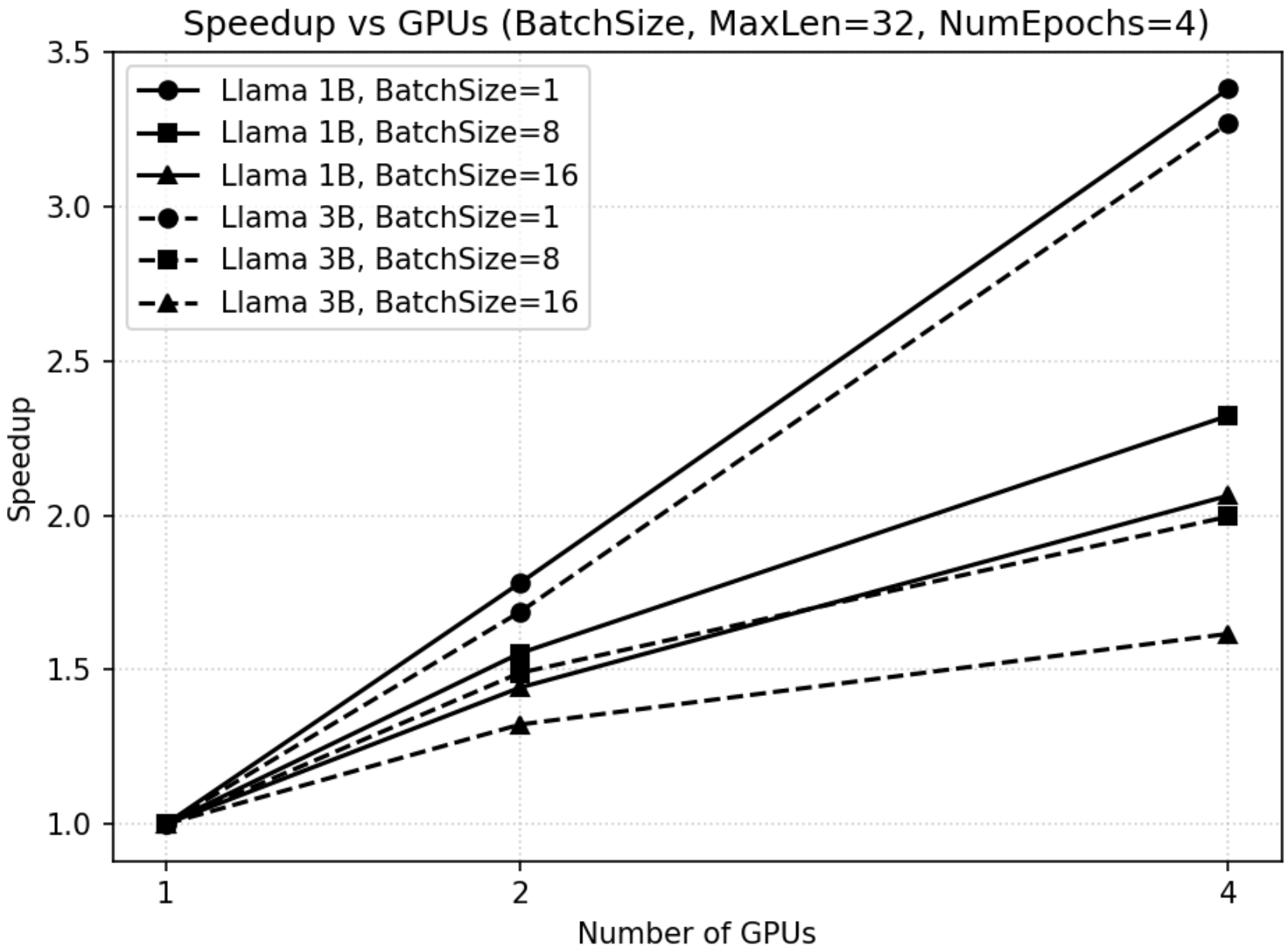
3.1.1 Efficiency vs Batch Size



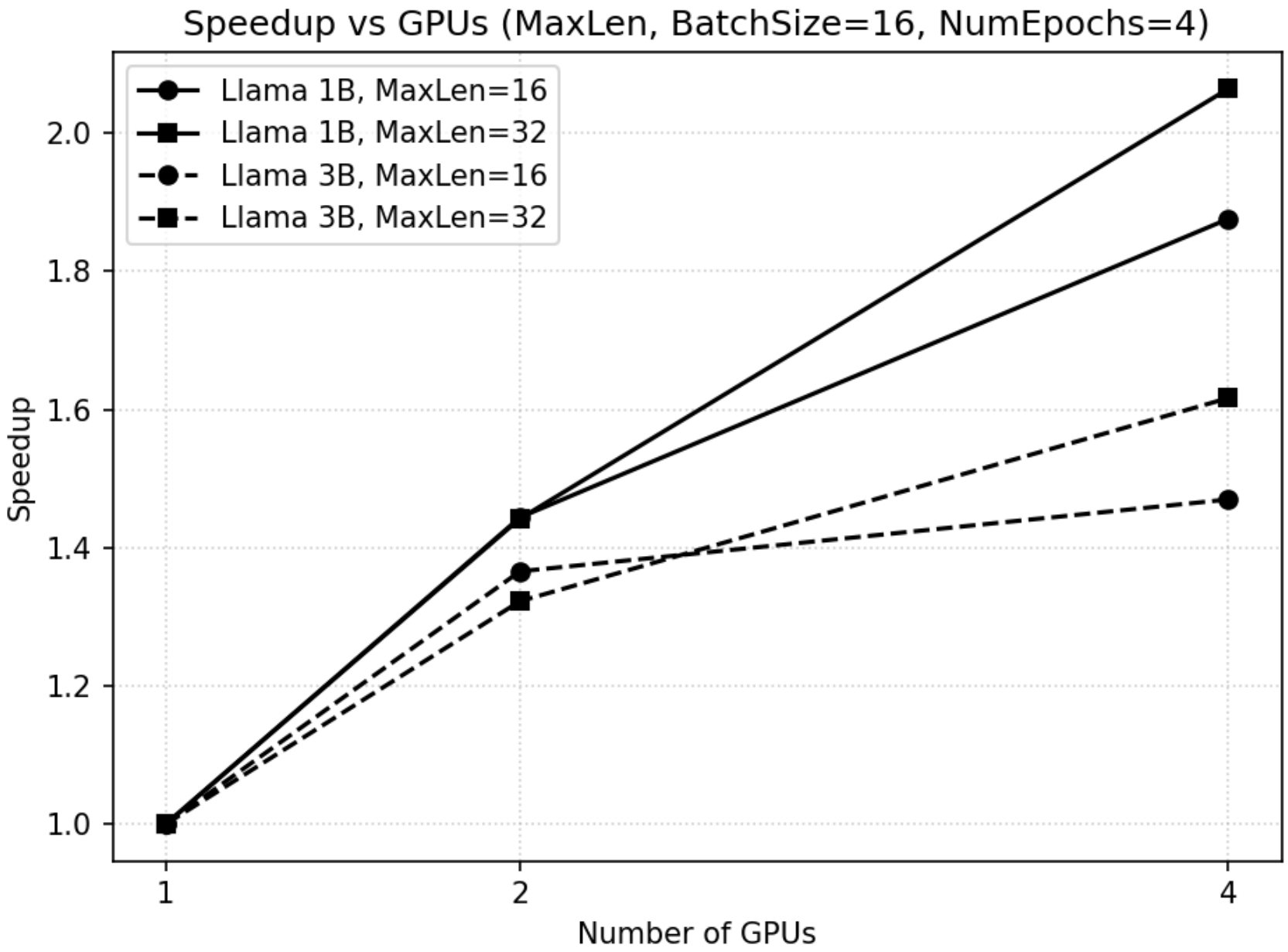
3.1.2 Efficiency vs Max Length



3.1.3 speedup vs Batch size

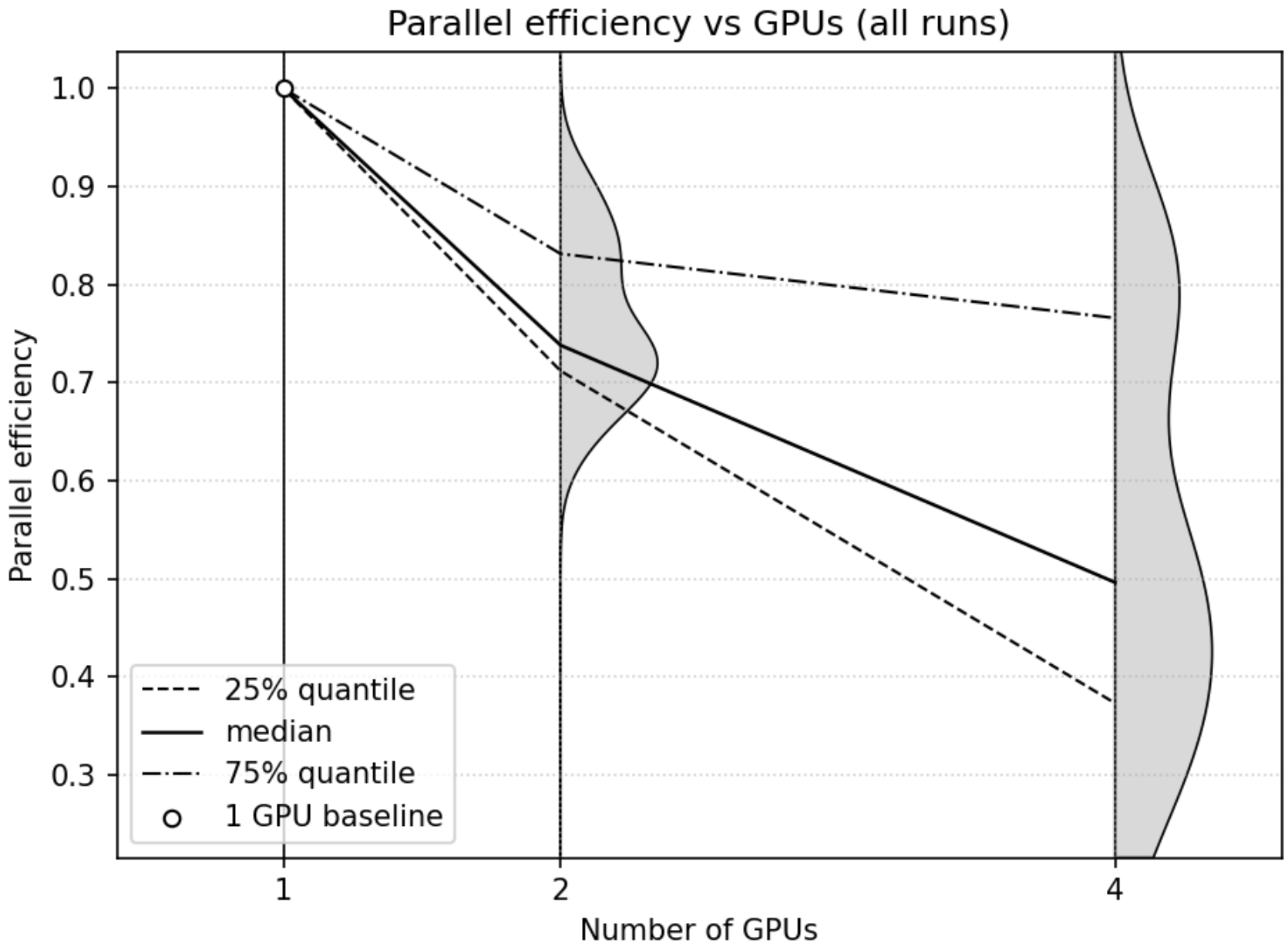


3.1.4 speedup vs Max Length

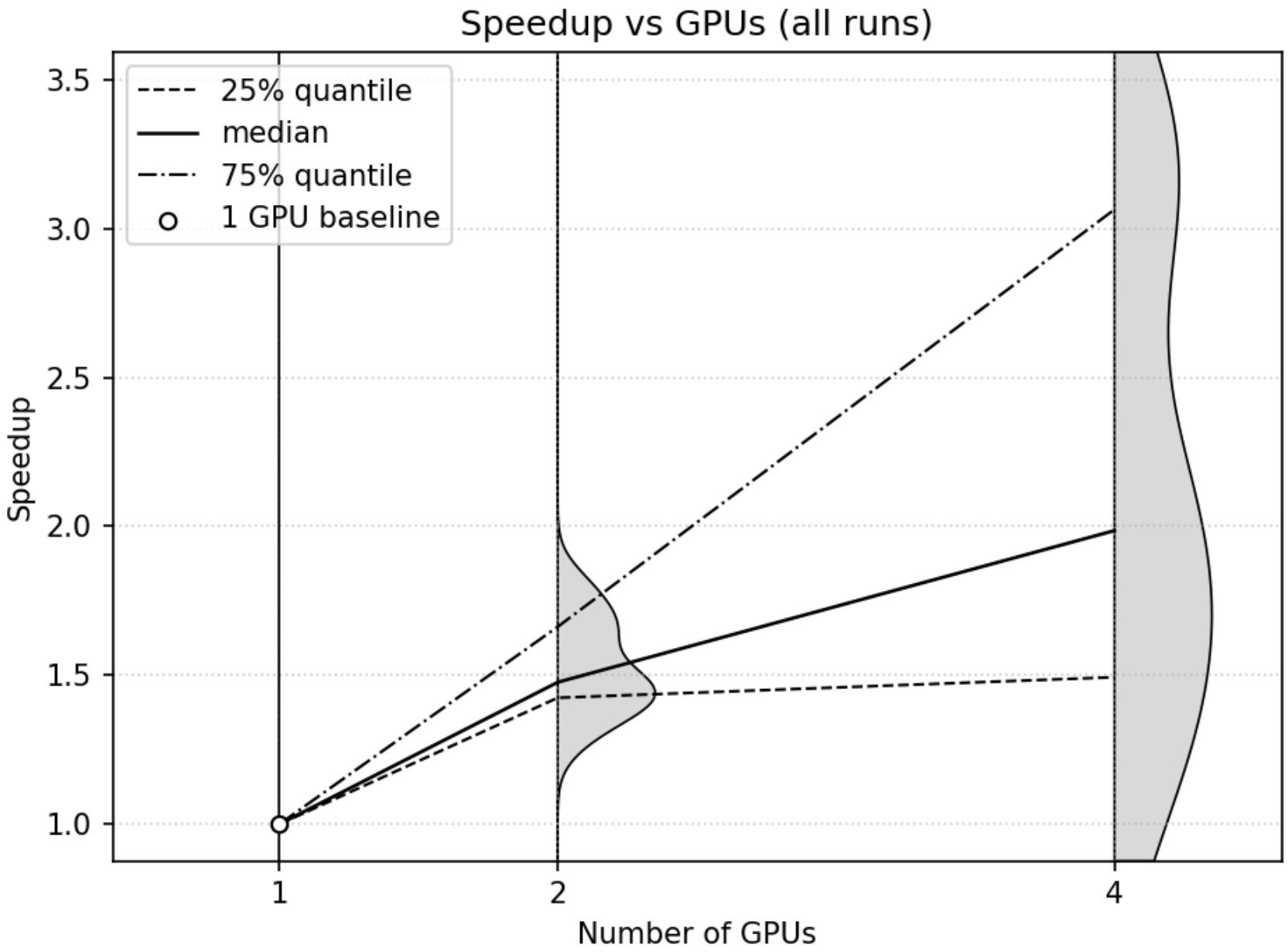


3.2 Aggregated Plots

3.2.1 Efficiency Distributions



3.2.2 Speedup Distributions

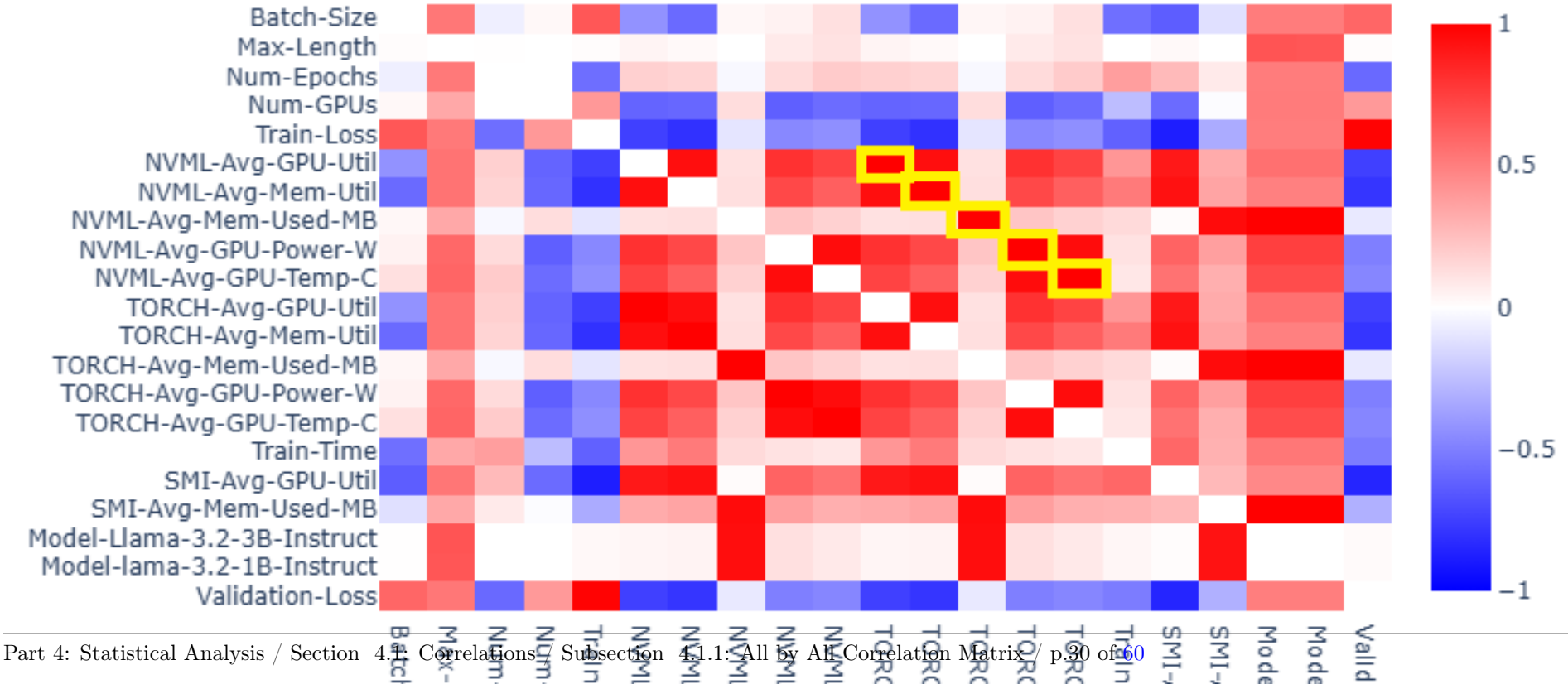


Part 4

Statistical Analysis

4.1 Correlations

4.1.1 All by All Correlation Matrix



4.1.2 All by All Relationships Top

Variable-i	Variable-j	Value	Metric
NVML-Avg-Mem-Used-MB	Model-Llama-3.2-3B-Instruct	1.00e+00	F1Score
TORCH-Avg-Mem-Used-MB	Model-lama-3.2-1B-Instruct	1.00e+00	F1Score
SMI-Avg-Mem-Used-MB	Model-lama-3.2-1B-Instruct	1.00e+00	F1Score
SMI-Avg-Mem-Used-MB	Model-Llama-3.2-3B-Instruct	1.00e+00	F1Score
NVML-Avg-Mem-Used-MB	Model-lama-3.2-1B-Instruct	1.00e+00	F1Score
TORCH-Avg-Mem-Used-MB	Model-Llama-3.2-3B-Instruct	1.00e+00	F1Score
NVML-Avg-Mem-Used-MB	TORCH-Avg-Mem-Used-MB	1.00e+00	Pearson R
NVML-Avg-GPU-Temp-C	TORCH-Avg-GPU-Temp-C	1.00e+00	Pearson R
NVML-Avg-GPU-Util	TORCH-Avg-GPU-Util	1.00e+00	Pearson R
NVML-Avg-Mem-Util	TORCH-Avg-Mem-Util	1.00e+00	Pearson R
NVML-Avg-GPU-Power-W	TORCH-Avg-GPU-Power-W	1.00e+00	Pearson R
Train-Loss	Validation-Loss	9.89e-01	Pearson R
NVML-Avg-Mem-Used-MB	SMI-Avg-Mem-Used-MB	9.56e-01	Pearson R
TORCH-Avg-Mem-Used-MB	SMI-Avg-Mem-Used-MB	9.56e-01	Pearson R
TORCH-Avg-GPU-Power-W	TORCH-Avg-GPU-Temp-C	9.54e-01	Pearson R
Truncated Table. Total rows = 210.			

4.2 Descriptive Statistics (Train Set)

Variables	mean	median	std	min	max	skewness	kurtosis
Batch-Size	8.22e+00	8.00e+00	5.95e+00	1.00e+00	1.60e+01	1.15e-01	-1.40e+00
Max-Length	2.43e+01	3.20e+01	7.99e+00	1.60e+01	3.20e+01	-7.41e-02	-1.99e+00
Num-Epochs	2.31e+00	2.00e+00	1.23e+00	1.00e+00	4.00e+00	4.24e-01	-1.44e+00
Num-GPUs	2.36e+00	2.00e+00	1.26e+00	1.00e+00	4.00e+00	3.40e-01	-1.55e+00
Train-Loss	2.02e+00	1.89e+00	6.23e-01	1.11e+00	3.70e+00	5.99e-01	-4.50e-01
NVML-Avg-GPU-Util	4.87e+01	5.21e+01	1.08e+01	2.09e+01	6.50e+01	-8.51e-01	1.48e-01
NVML-Avg-Mem-Util	3.08e+01	3.27e+01	8.31e+00	1.12e+01	4.35e+01	-6.00e-01	-2.47e-01
NVML-Avg-Mem-Used-MB	2.73e+04	3.30e+04	1.11e+04	1.38e+04	4.04e+04	4.28e-02	-1.84e+00
NVML-Avg-GPU-Power-W	1.51e+02	1.49e+02	2.52e+01	9.71e+01	2.24e+02	3.80e-01	4.77e-01
NVML-Avg-GPU-Temp-C	4.52e+01	4.50e+01	1.98e+00	4.05e+01	5.14e+01	5.46e-01	7.36e-01
TORCH-Avg-GPU-Util	4.87e+01	5.21e+01	1.08e+01	2.09e+01	6.50e+01	-8.50e-01	1.46e-01
TORCH-Avg-Mem-Util	3.08e+01	3.27e+01	8.31e+00	1.12e+01	4.35e+01	-6.00e-01	-2.46e-01
TORCH-Avg-Mem-Used-MB	2.67e+04	3.25e+04	1.11e+04	1.33e+04	3.99e+04	4.27e-02	-1.84e+00
TORCH-Avg-GPU-Power-W	1.51e+02	1.49e+02	2.52e+01	9.71e+01	2.24e+02	3.80e-01	4.80e-01
TORCH-Avg-GPU-Temp-C	4.52e+01	4.50e+01	1.98e+00	4.05e+01	5.14e+01	5.46e-01	7.35e-01
Truncated Table. Total rows = 21.							

4.3 Model Llama 3.2 3B Instruct vs TORCH Avg Mem Used MB

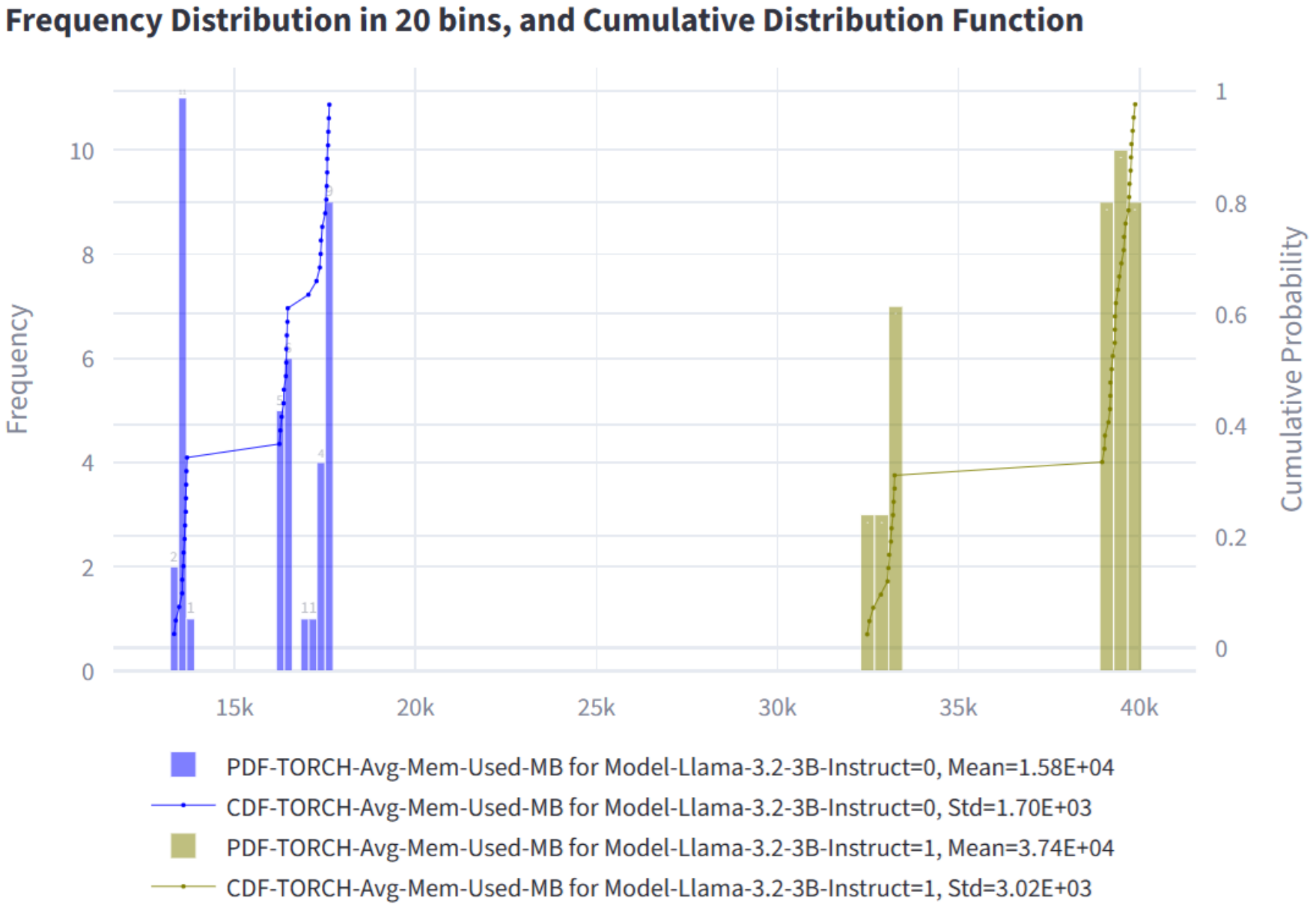
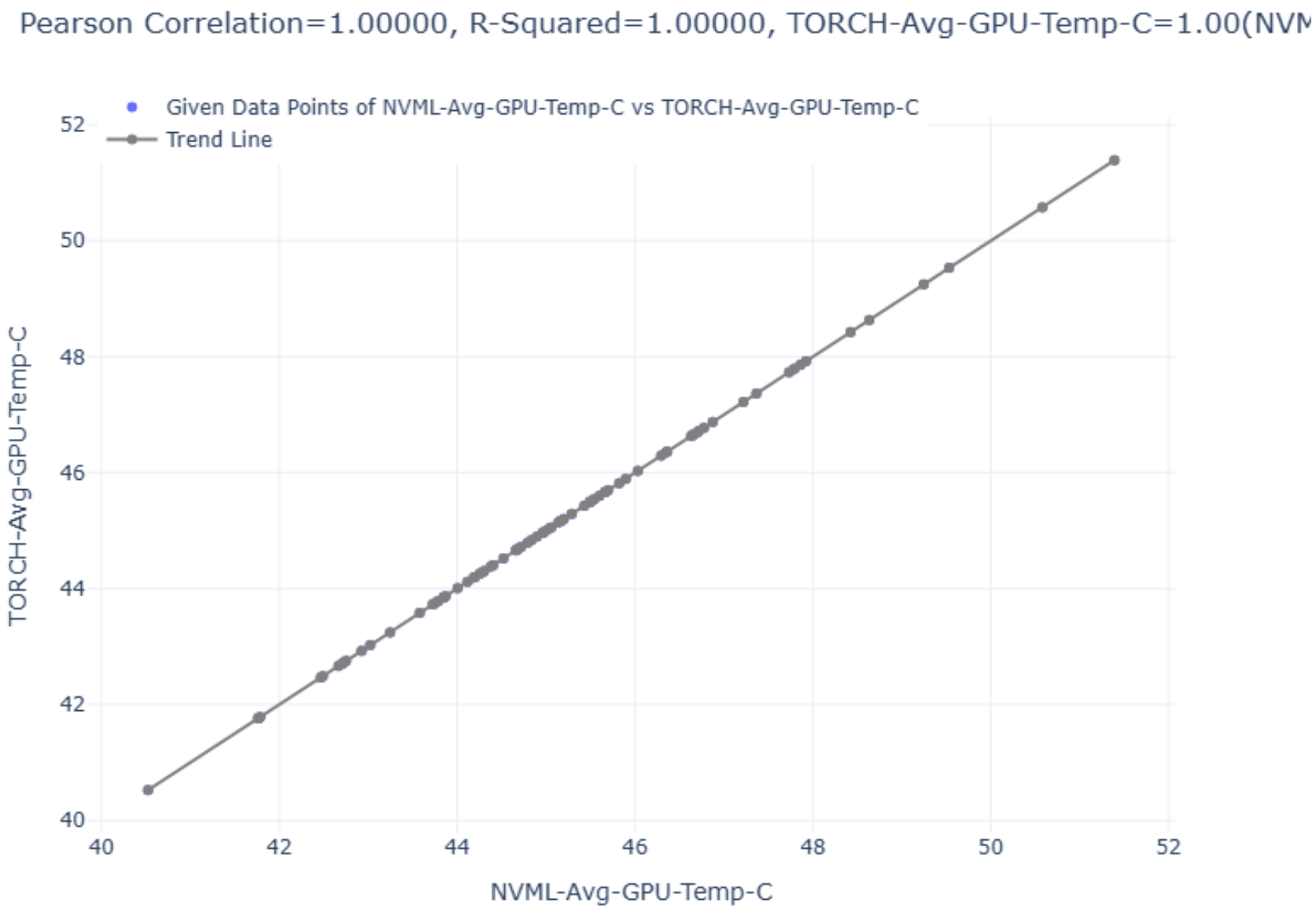


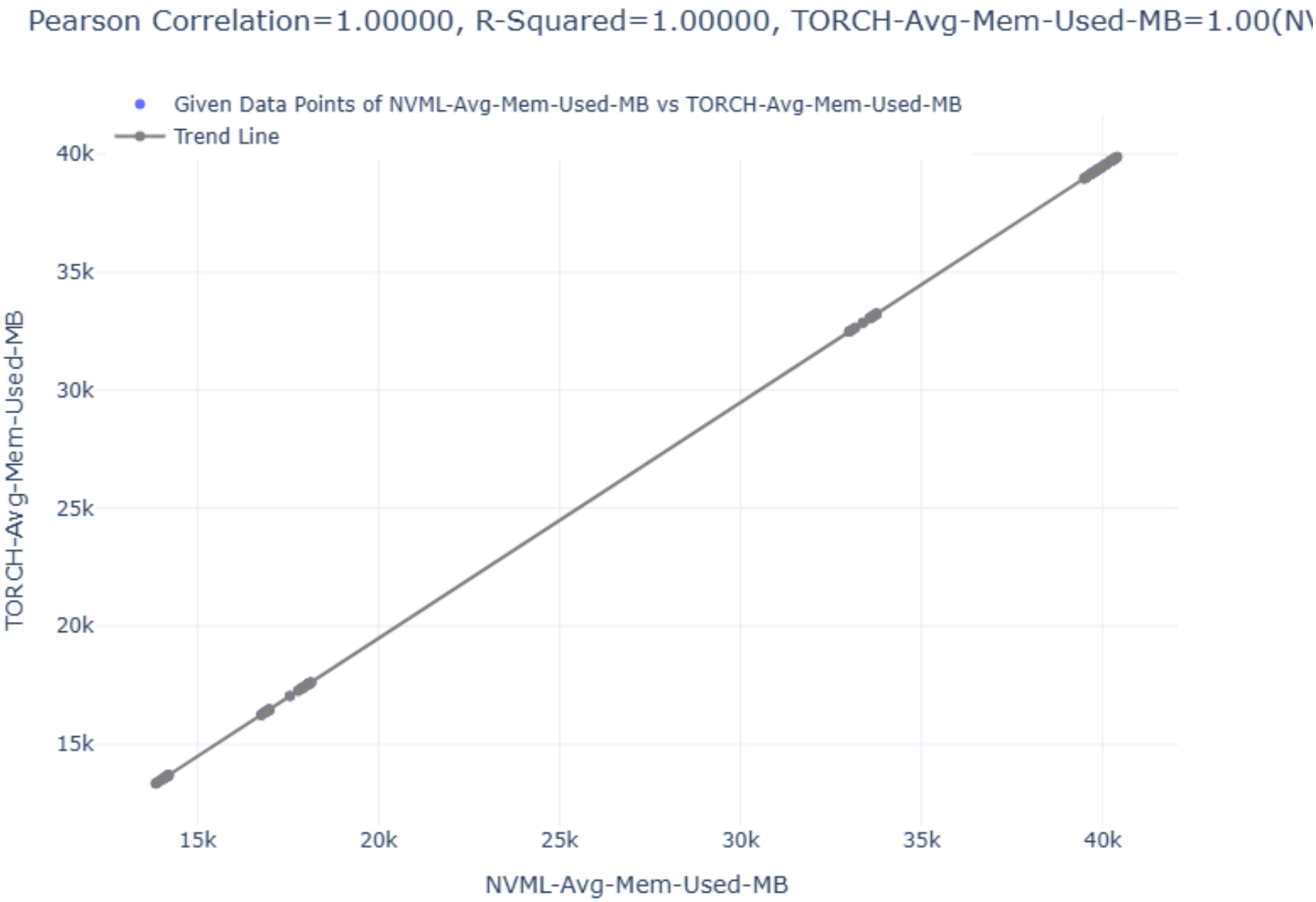
Figure 4.1: $\eta^2 = 0.95080$ for Model Llama 3.2 3B Instruct vs TORCH Avg Mem Used MB

4.4 NVML vs TORCH measurements

4.4.1 NVML Avg GPU Temperature vs TORCH Avg GPU Temperature

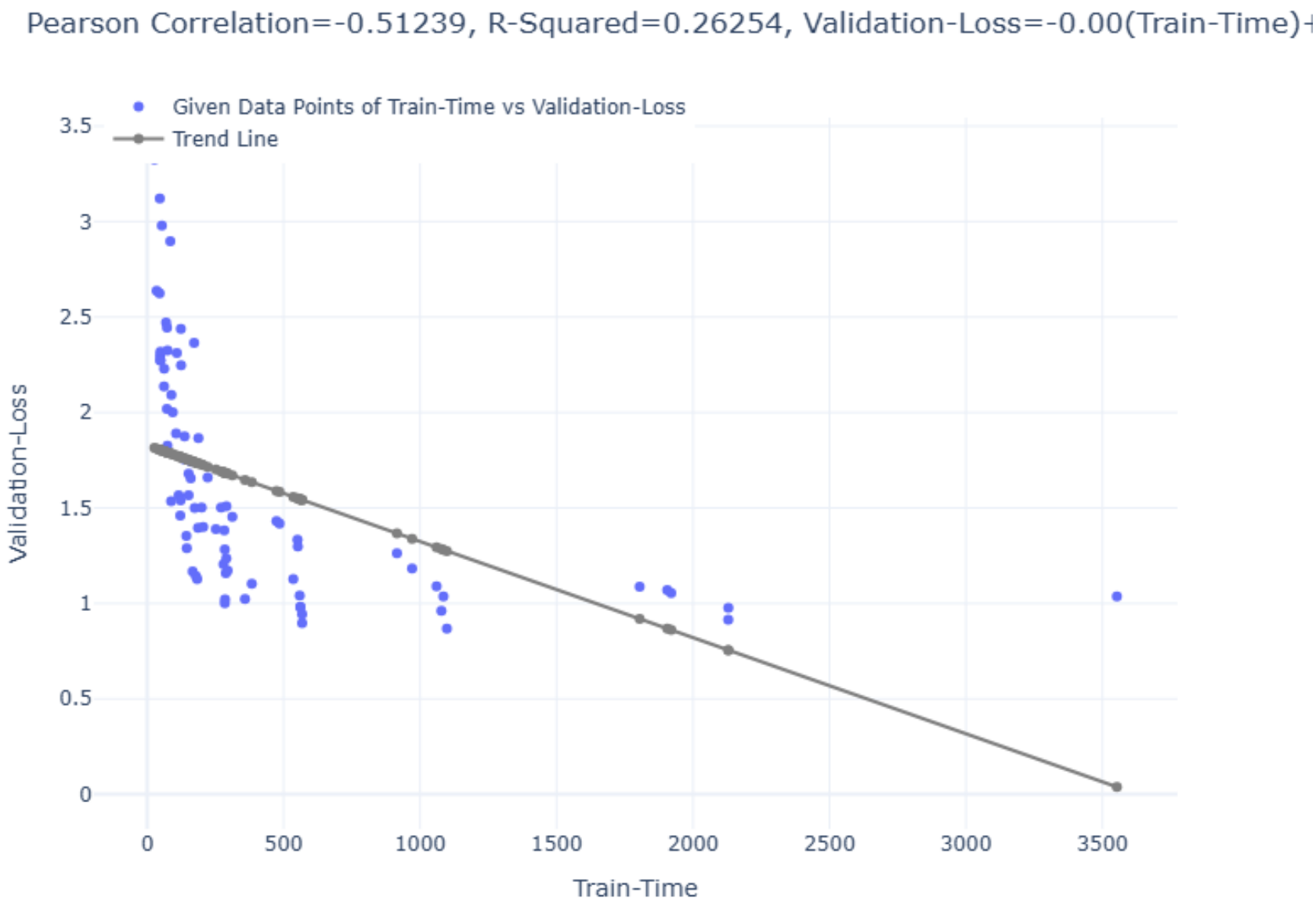


4.4.2 NVML Avg Mem Used MB vs TORCH Avg Mem Used MB



4.5 Hyperparameters vs Validation Loss

4.5.1 Train Time vs Validation Loss



4.5.2 Num Epochs vs Validation Loss

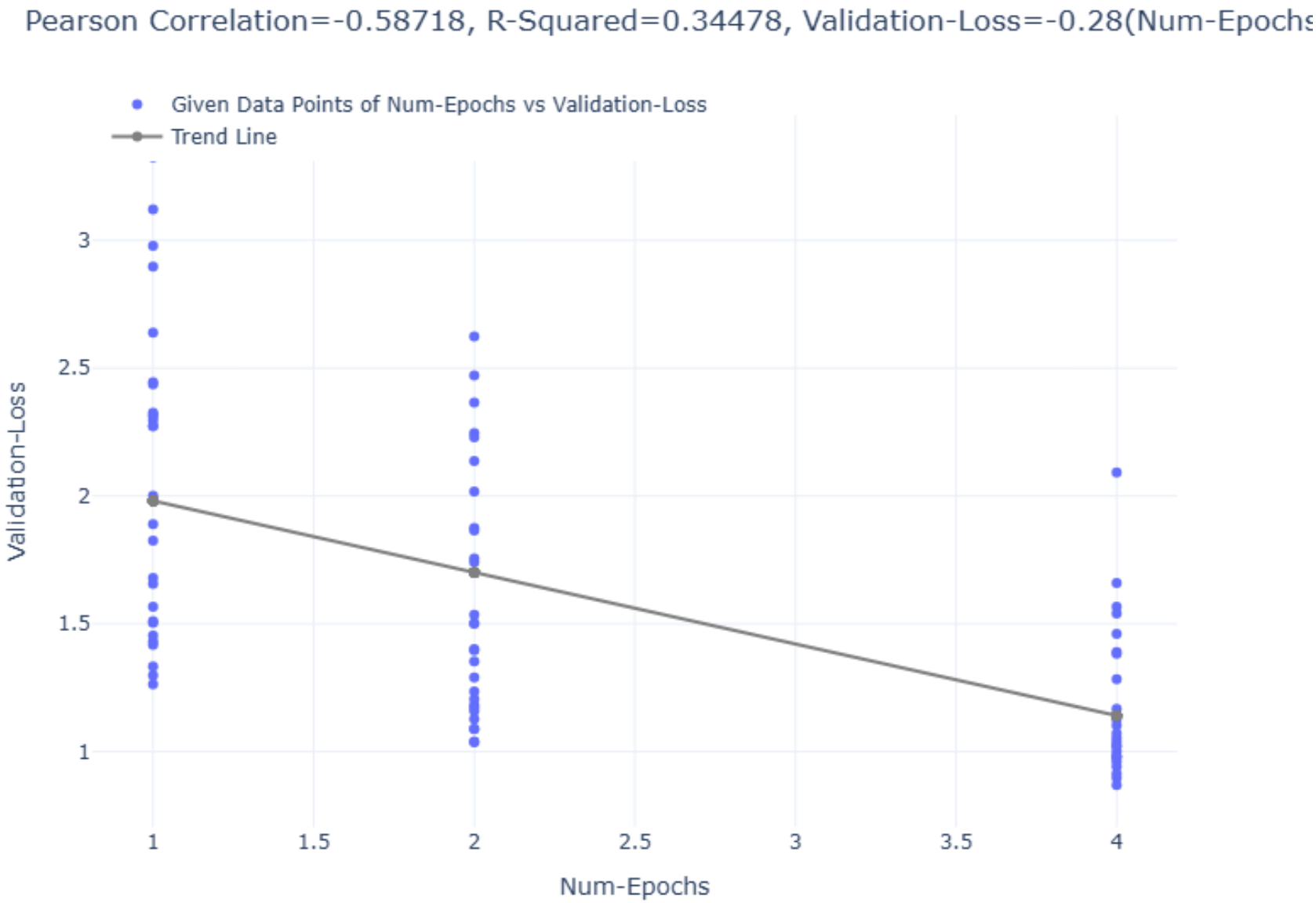
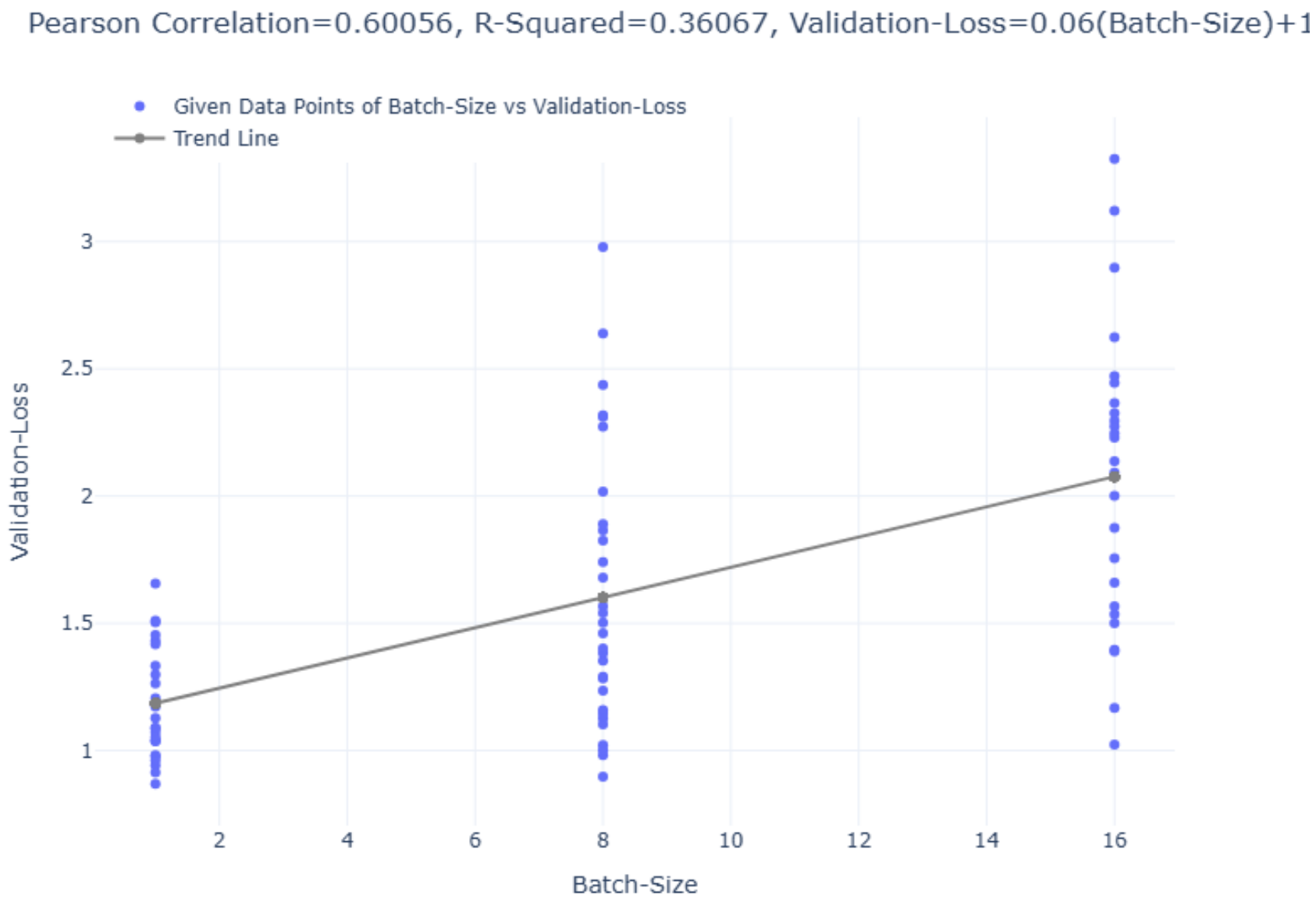
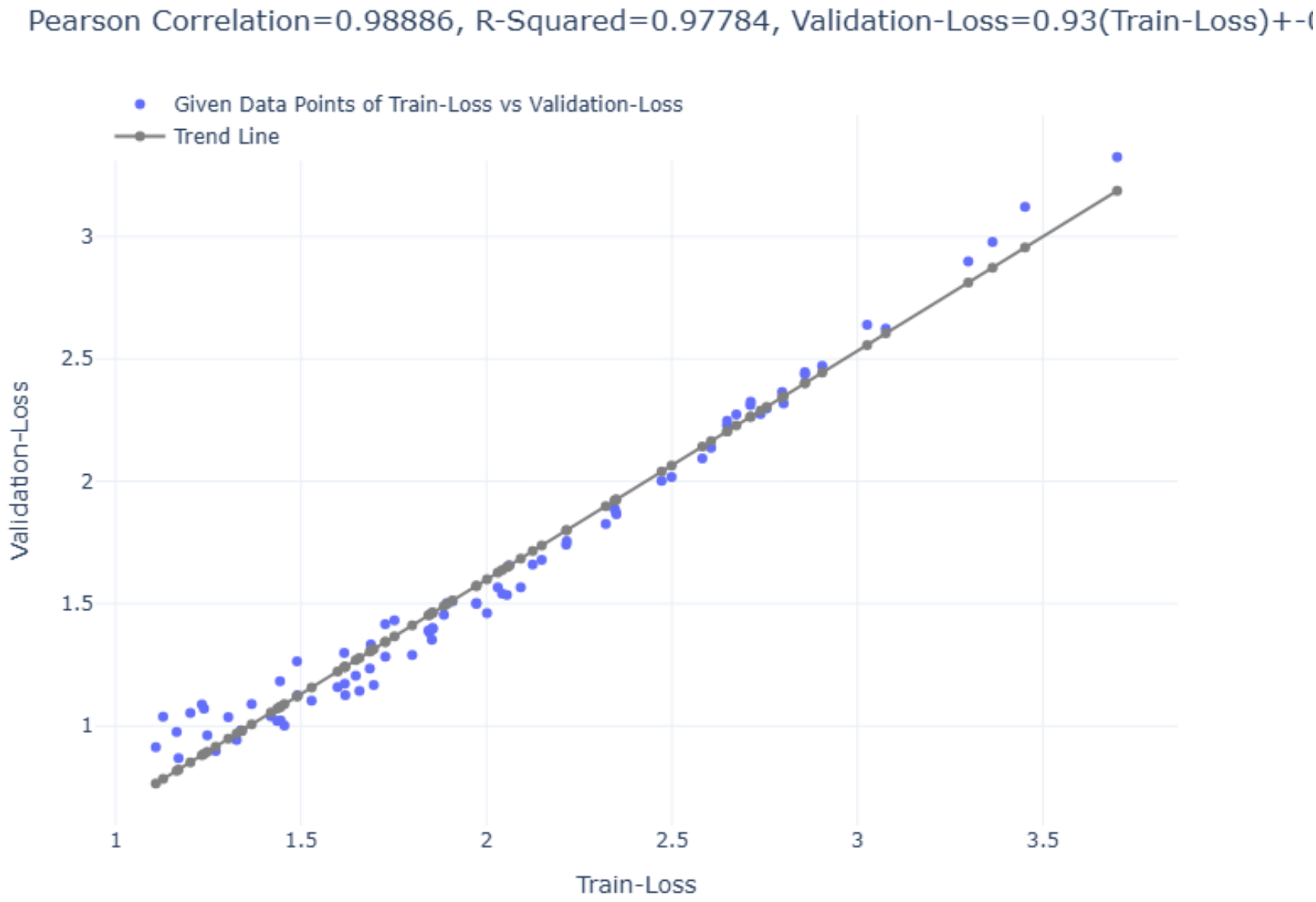


Figure 4.2: 059NegativePearson FeatureNum Epochs vs Target Validation Loss.html.json

4.5.3 Batch Size vs Validation Loss

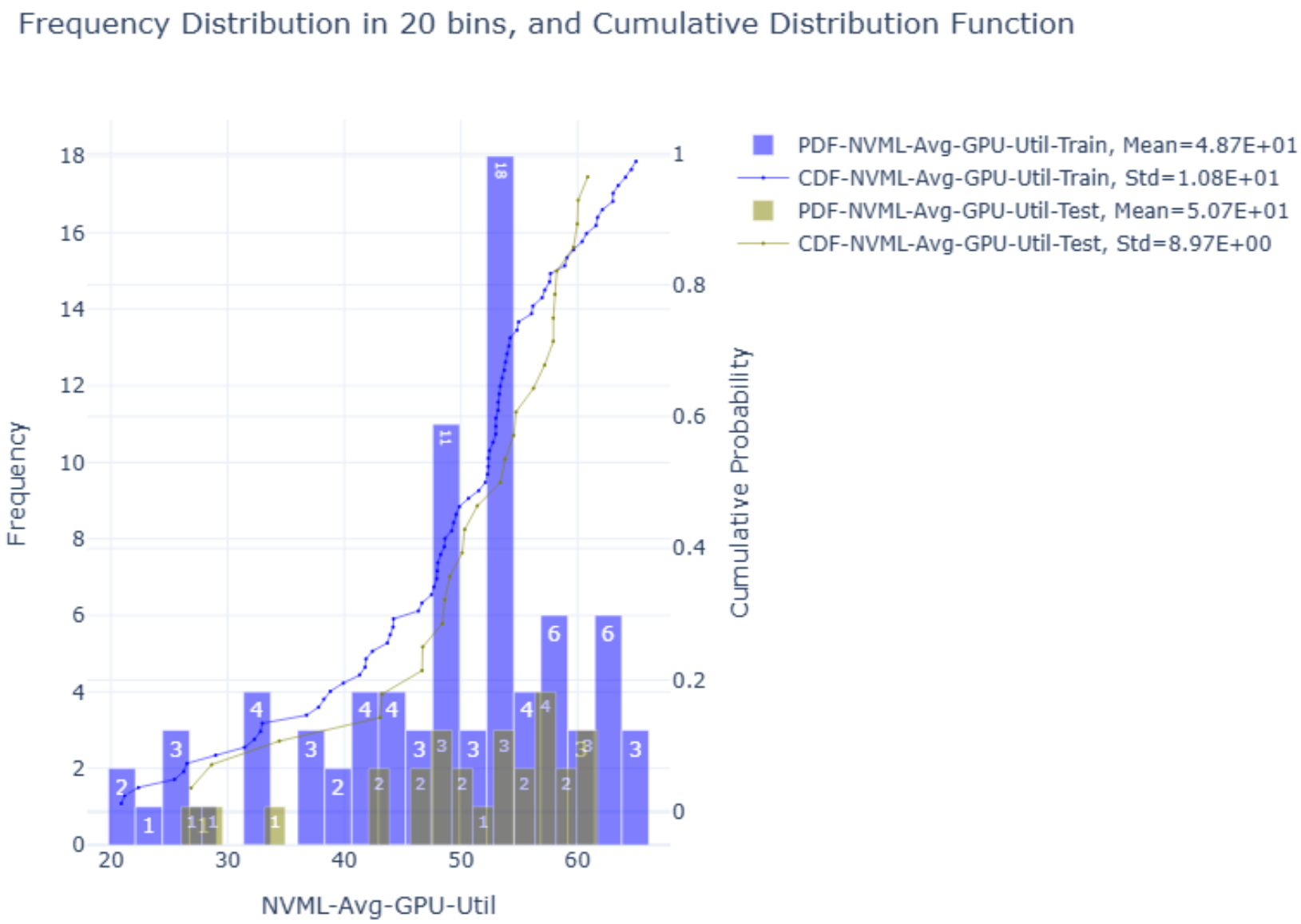


4.5.4 Train Loss vs Validation Loss

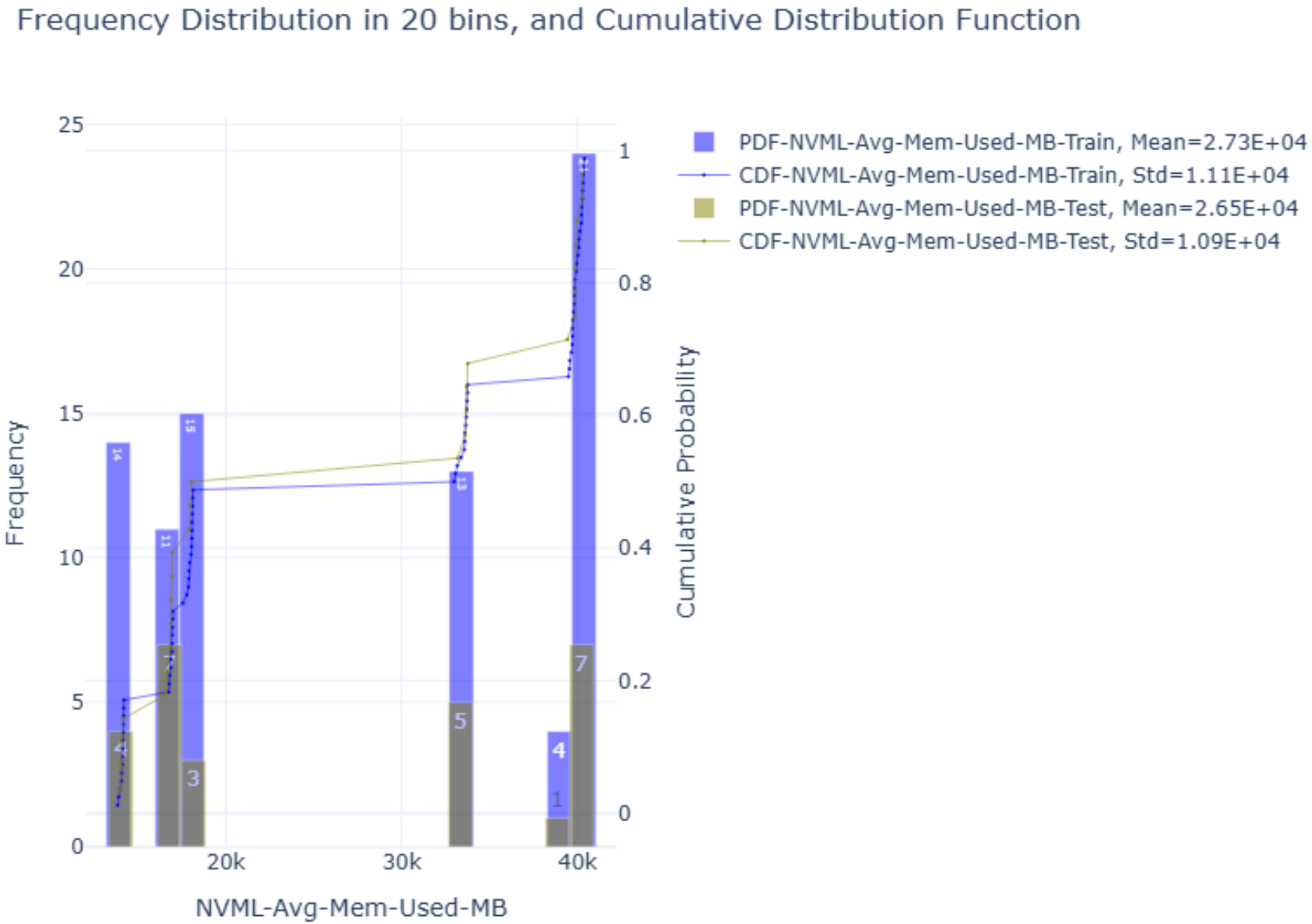


4.6 Distributions

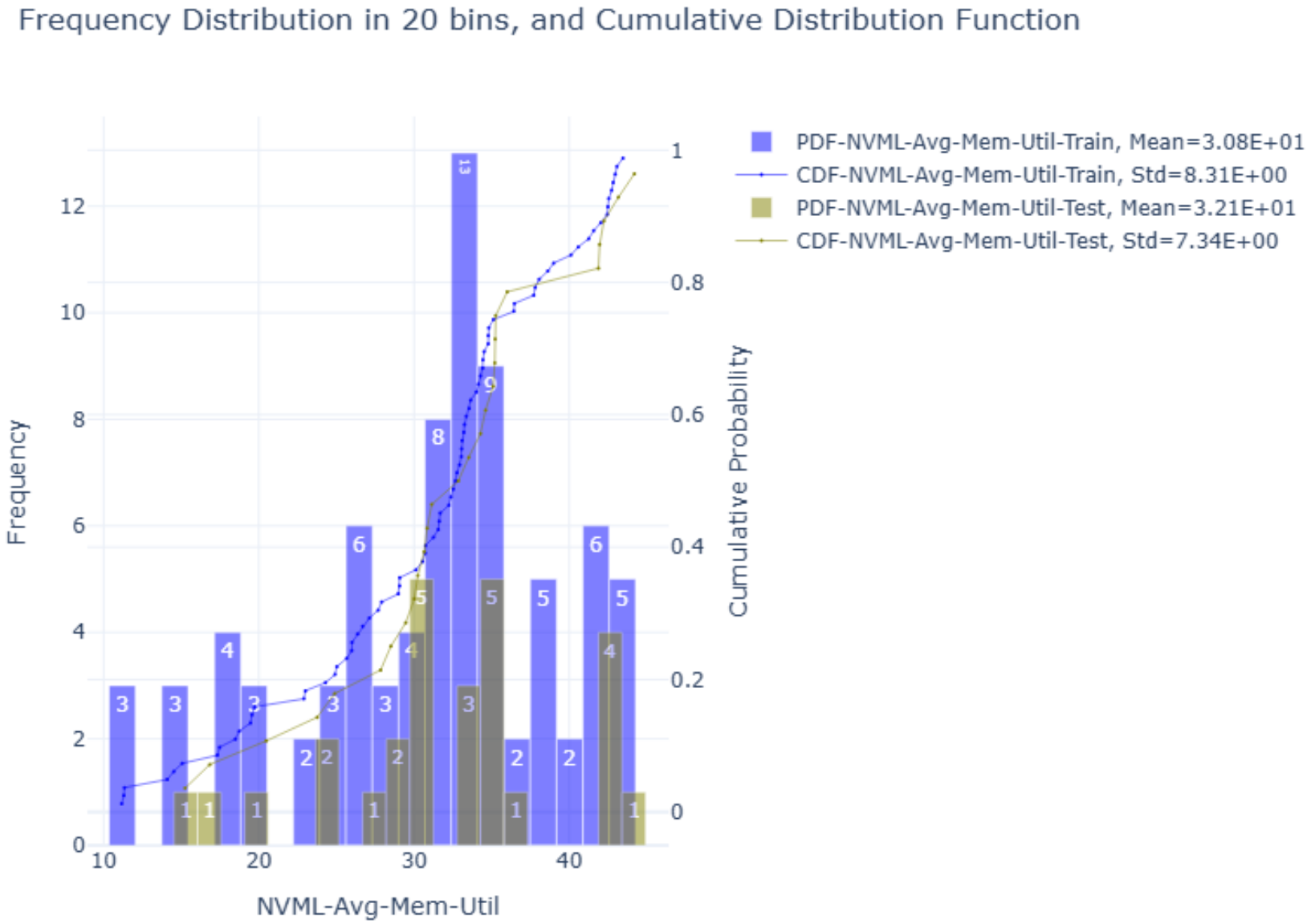
4.6.1 NVML Avg GPU Util



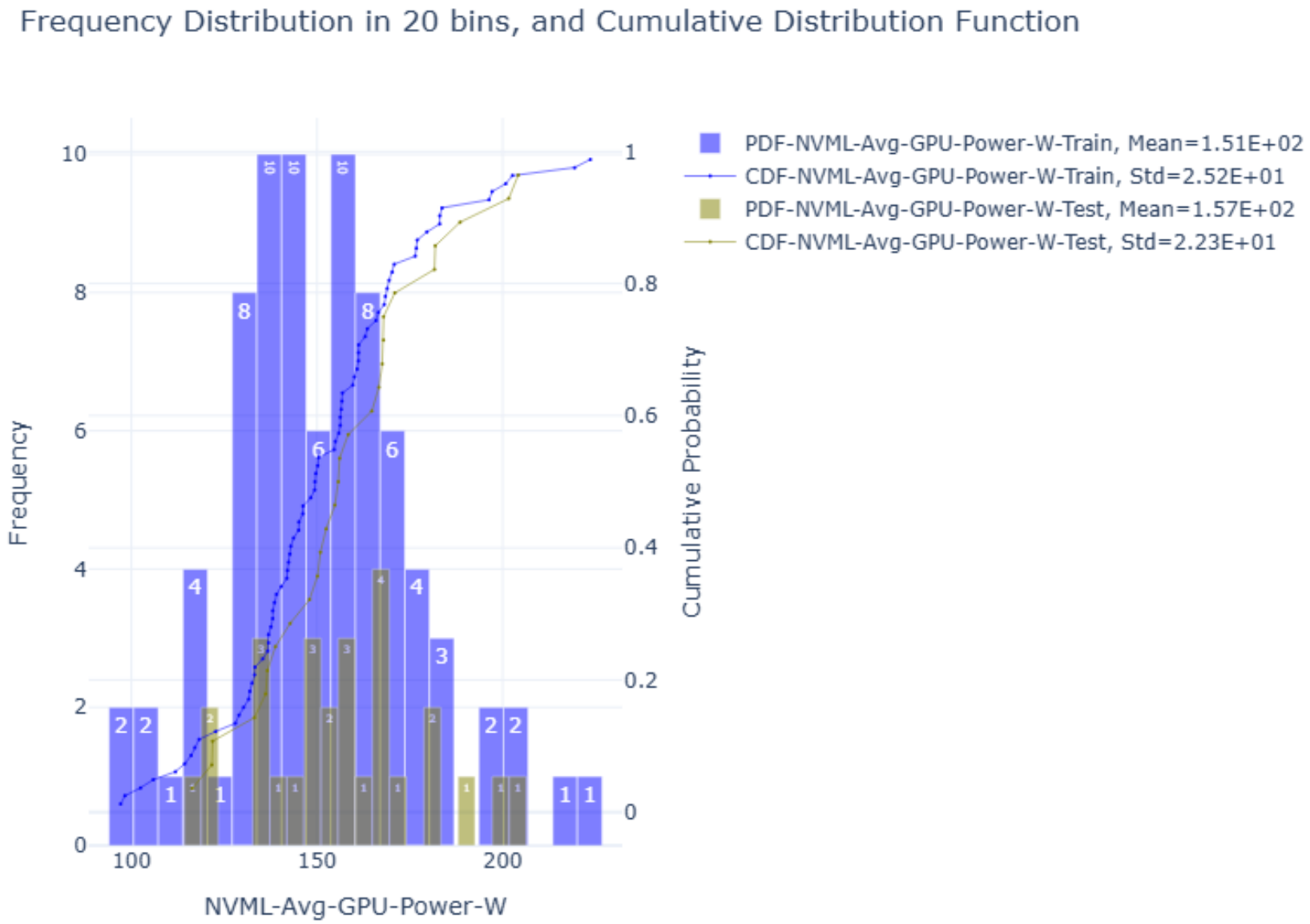
4.6.2 NVML Avg Mem Used MB



4.6.3 NVML Avg Mem Util

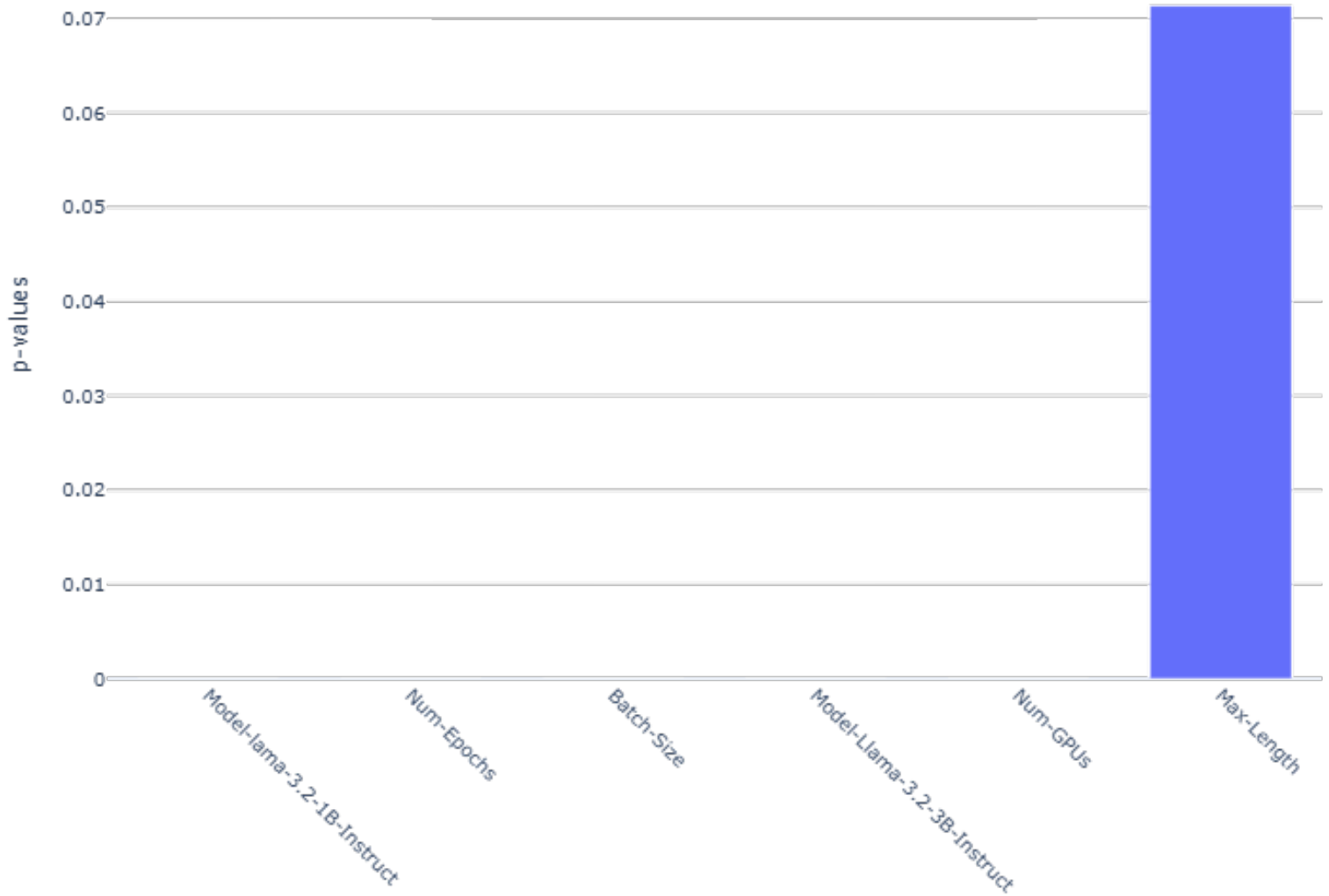


4.6.4 NVML Avg GPU Power W

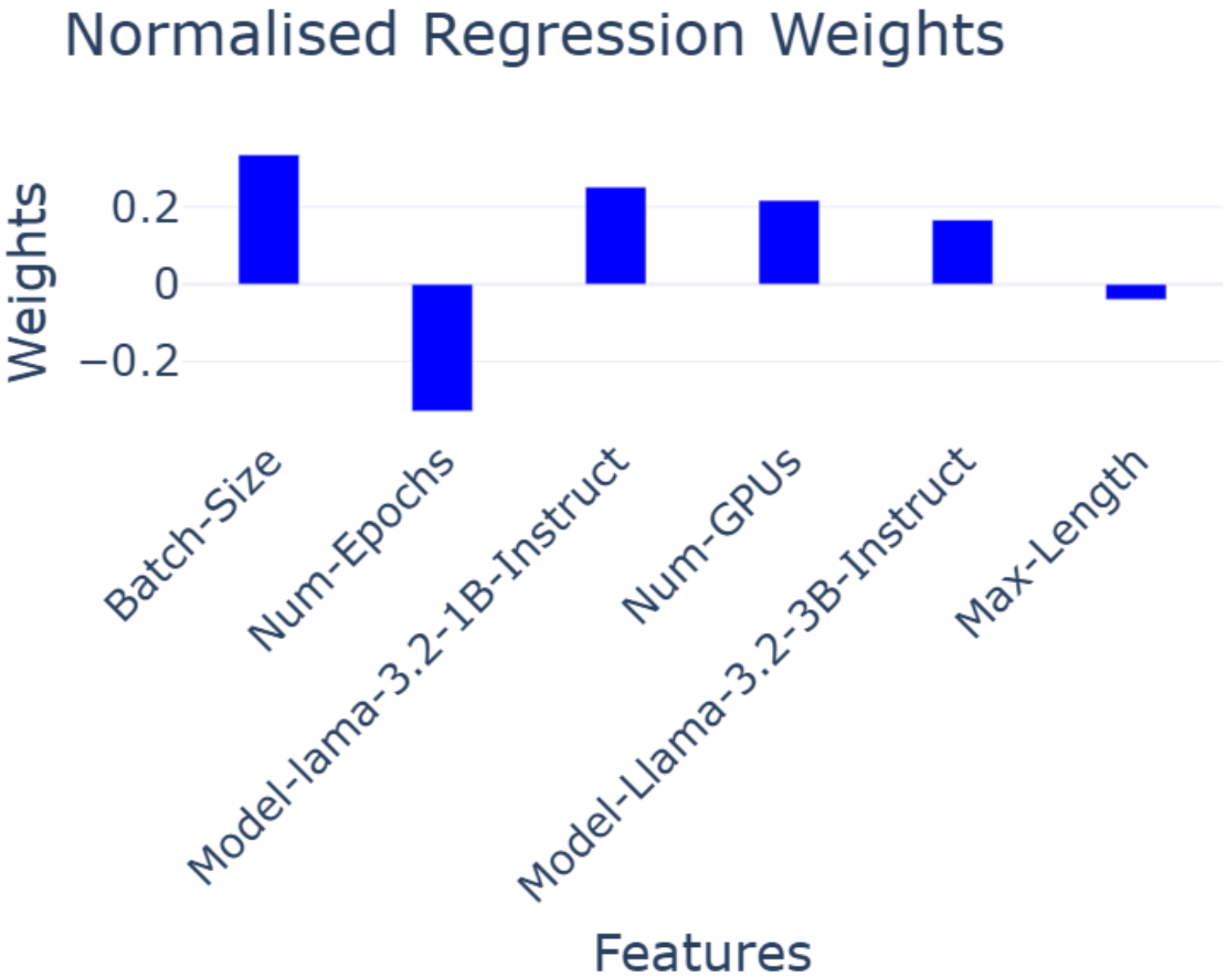


4.7 Linear Regression

4.7.1 p Values for Target Validation Loss

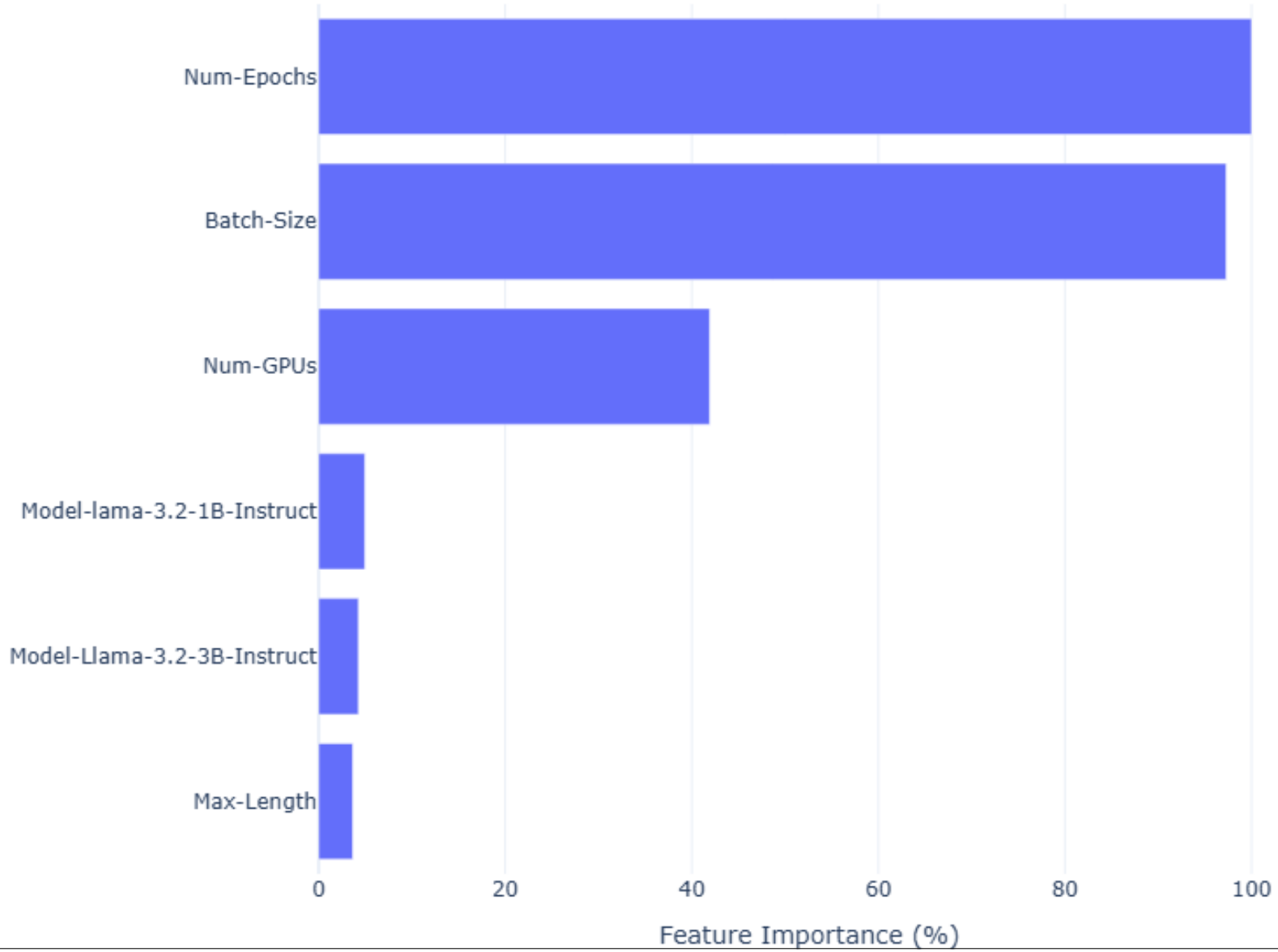


4.7.2 Normalised Regression Weights for Target Validation Loss



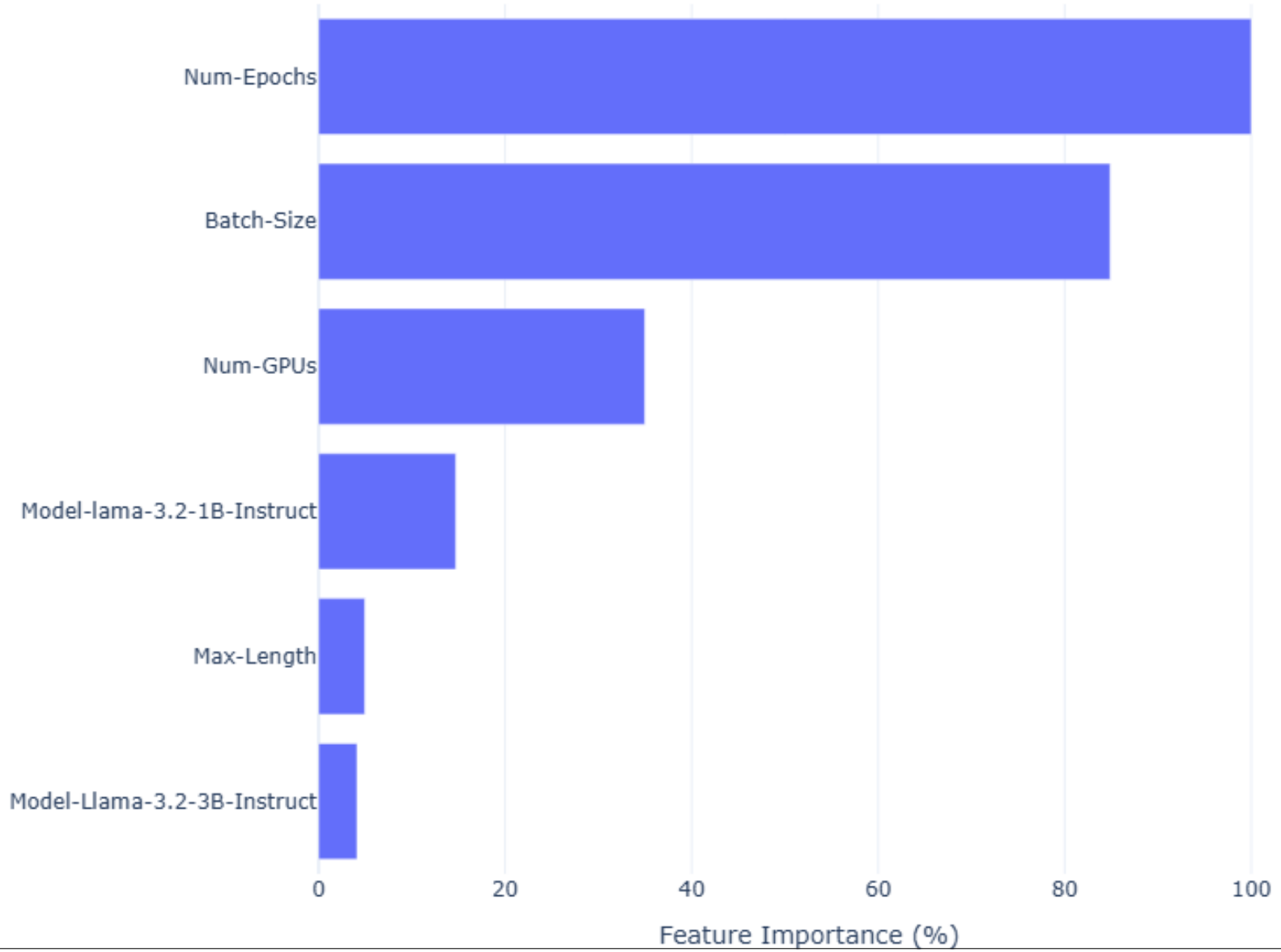
4.8 Random Forests

4.8.1 Feature Importances with Random Forests



4.9 XGBoost

4.9.1 Feature Importances with XGBoost



Part 5

Results

Below is a comparison among the base and fine-tuned models for the various questions. The results are indicative and the model is not assumed for production.

Question: What is Pharos AI Factory?

1. **Answer (Base model)**: Pharos AI Factory is a proprietary artificial intelligence (AI) technology developed by the company Neso Group. It is a cloud-based platform designed to support businesses in their digital transformation journey.
2. **Answer (Fine-tuned model)**: Pharos AI Factory is a Greek government initiative to establish Greece as a hub for artificial intelligence (AI) and machine learning (ML), providing access to cutting-edge technology, expertise, and infrastructure. The project aims to promote the development of an AI ecosystem that can support the country's digital transformation goals, including the creation of innovative startups, research and development activities, and the integration of AI into various sectors such as healthcare, finance, education, and public services.

Question: What is the significance and benefits of the Pharos AI Factory project?

1. **Answer (Base model)**: The Pharos AI Factory is a collaborative initiative aimed at developing a robust, scalable, and secure artificial intelligence (AI) platform that can be used for various applications. Here are some key aspects and benefits:

2. **Answer (Fine-tuned model)**: The Pharos AI Factory is a flagship initiative of Greece's Hellenic Ministry of Digital Governance, in partnership with the European Union. It serves as an essential tool for fostering innovation and digital transformation across various sectors by providing access to cutting-edge artificial intelligence (AI) technologies, data, and expertise. By doing so, it aims to bridge the gap between technological advancements and societal needs, thereby creating a robust foundation for sustainable economic growth and competitiveness.

Question: How is Pharos AI Factory connected to DAEDALUS?

1. **Answer (Base model)**: Pharos AI Factory (PAF) and DAEDALUS are two distinct entities. However, they share some commonalities in their development process.
2. **Answer (Fine-tuned model)**: Pharos AI Factory is a Greek initiative that supports the development and deployment of Artificial Intelligence (AI) technologies, with its main goal being to empower businesses and public sector organizations in their transformation towards digitalization. Pharos AI Factory has strong connections to DAEDALUS, which stands for Data Analytics and Decision Support for European Agencies. The connection between the two initiatives enables users from both sides to collaborate on projects related to data management, analytics, and decision-making. By leveraging these collaborations, users can access valuable resources, expertise, and support, contributing to the development of more effective AI solutions across various sectors.

Question: Which groups will benefit from Pharos AI Factory?

1. **Answer (Base model)**: These are typically the ones that have a high need for customized solutions, advanced analytics, and data-driven insights.
2. **Answer (Fine-tuned model)**: Private companies, public administrations and research organizations can all participate in the program. They can access resources and tools to develop their AI capabilities, contribute to the development of EU-wide AI strategies, improve their competitiveness and enhance innovation. Pharos is open to the private sector, including startups, SMEs, as well as larger enterprises. It also has a strong focus on the public sector, providing valuable expertise, support, and opportunities for them to become more AI-competent, leading to better decision-making, improved service quality, and enhanced competitiveness.

Question: What is the connection between AI and HPC in Pharos?

1. **Answer (Base model)**: Pharos is a type of autonomous underwater vehicle (AUV) that can collect data from the ocean floor. It uses artificial intelligence (AI) to process and analyze this data, which is then used to inform decision-making about marine conservation and management. In Pharos, AI plays a critical role in collecting, processing, and analyzing large amounts of sensor data from the ocean floor. This data includes information about water temperature, salinity, chemistry, and other physical properties of the seafloor. Here are some ways in which AI connects with High-Performance Computing (HPC): 1. **Data analysis**: AI algorithms are applied
2. **Answer (Fine-tuned model)**: Pharos, a Greek AI Factory, is designed to support Greece's vision of becoming a global leader in Artificial Intelligence. As part of this initiative, Pharos aims to create an ecosystem that fosters innovation, inclusivity, and excellence. It includes several components, including AI Factories, which are equipped with High-Performance Computing (HPC) infrastructure. In Pharos, HPC plays a critical role in accelerating AI development, deployment, and analysis. By leveraging HPC resources, researchers and developers can run complex simulations, generate high-quality data, and train machine learning models more efficiently. This enables them to tackle challenging problems