dasya.dk rad.itu.dk dff.dk itu.dk

GPUMemNet: GPU Memory Usage Estimation for Efficient Resource Management for Deep Learning Training

Ehsan Yousefzadeh-Asl-Miandoab (***ehyo@itu.dk***)

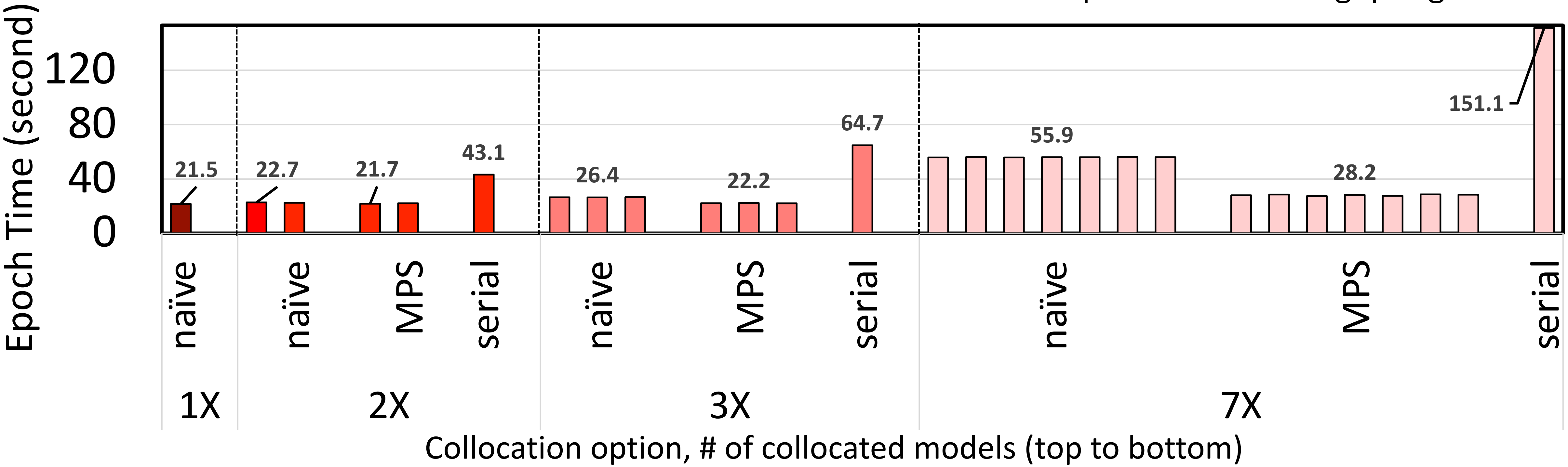
Collaborators: Reza Karimzadeh, Bulat Ibragimov*, Pınar Tözün*

IT University of Copenhagen, *University of Copenhagen

GPU Underutilization

- GPUs are underutilized ✷.
  - **Energy-inefficient** & **waste of hardware resources**
- Collocation can be beneficial ✣.
  - ResNet26 on Cifar10, batch size = 32

Up to **~5.35x** throughput gain



✷ Jeon, Myeongjae, et al. "**Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads.**" USENIX Annual Technical Conference. 2019.
✷ Yanjie Gao et al. **"An Empirical Study on Low GPU Utilization of Deep Learning Jobs."** ICSE 2024.
✣ Ties Robroek, Ehsan Yousefzadeh-Asl-Miandoab, and Pınar Tözün. "**An analysis of collocation on GPUs for deep learning training.**" *EuroMLSys 2024*

Need for GPU Memory Estimation

- Collocation comes with challenges:
  - Out-of-memory (OOM) Crash
  - Resource-interference can degrade performance (can be harsher than being serialized)

Having an estimation for GPU Memory ~= More Reliable Collocation

Estimating GPU Memory is challenging!

- Optimizations
  - Applied by Default:
    - Activation reuse, dynamic memory management
  - May be enabled by the user:
    - Layer fusion, gradient checkpointing, mixed precision, etc.

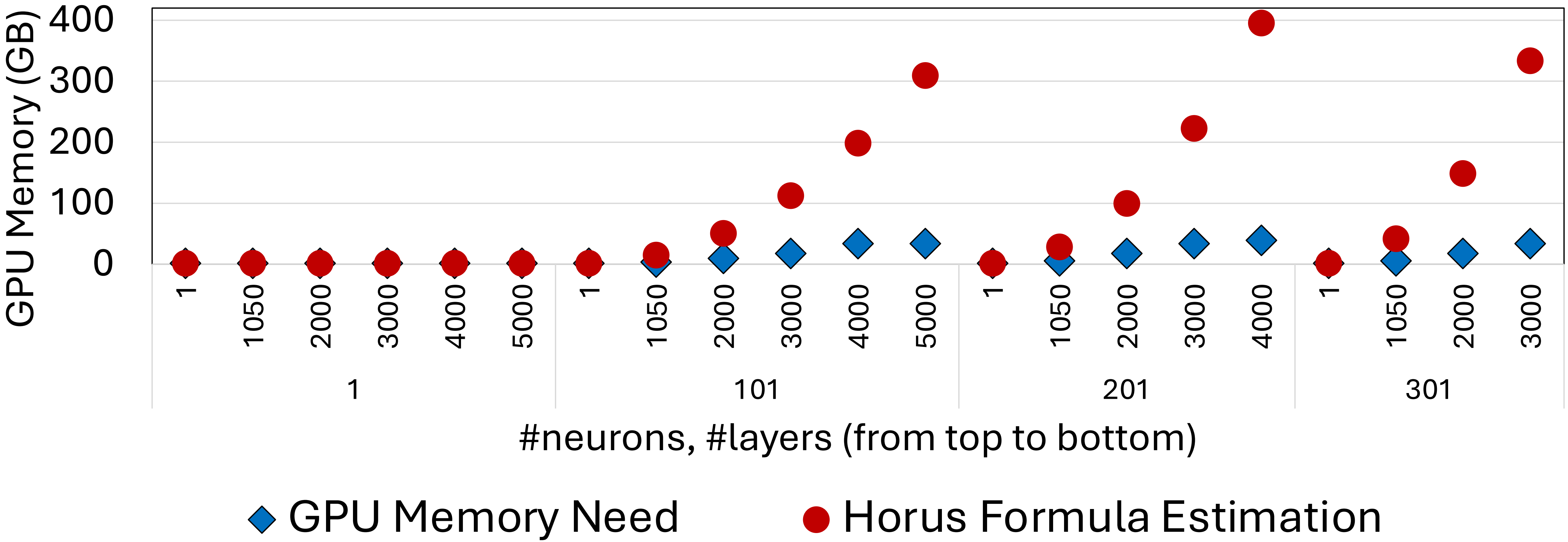
These introduce levels of unpredictability to GPU memory estimation.

Existing Estimators

1. Analytical
    1. Horus formula (***TPDS ’21***)
    2. DNNMem, Microsoft’s Analytical work (***ESEC/FSE ’20***)
    3. LLMem (***IJCAI ’24***)

Evaluating Horus Formula

Up to ~395GB misestimation



Horus Overestimates and limits Collocation Potentials!

Existing Estimators

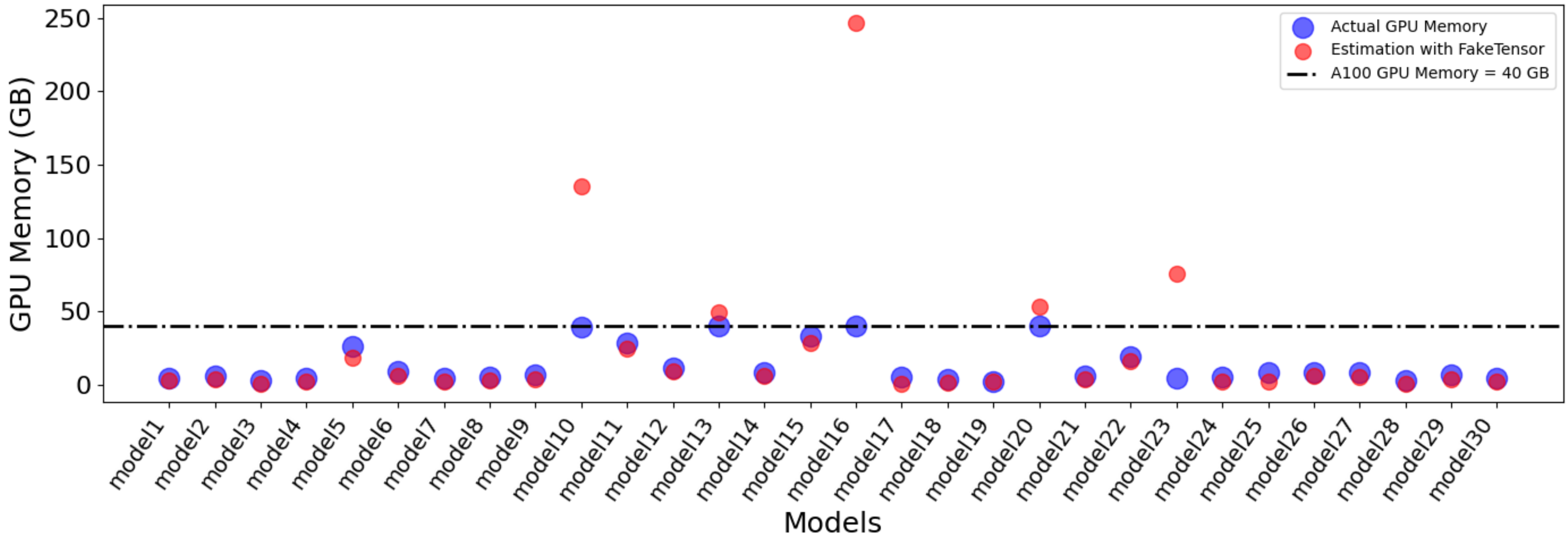
1. Analytical
    1. Horus formula (***TPDS ’21***)
    2. DNNMem, Microsoft’s Analytical work (***ESEC/FSE ’20***)
    3. LLMem (***IJCAI ’24***)

2. Libraries
    1. Fake Tensor
    2. DeepSpeed

https://pytorch.org/docs/stable/torch.compiler_fake_tensor.html
https://deepspeed.readthedocs.io/en/latest/memory.html

Evaluating Fake Tensor

Huge Misestimations . E.g., 500GB diff



Fake Tensor misestimates, causing OOMs and limiting the collocation potential!

Existing Estimators

1. Analytical
    1. Horus formula (***TPDS ’21***)
    2. DNNMem, Microsoft’s Analytical work (***ESEC/FSE ’20***)
    3. LLMem (***IJCAI ’24***)

2. Libraries
    1. Fake Tensor
    2. DeepSpeed

3. ML-based approach
    - DNNPerf , graph neural networks (***ICSE-SEIP ‘23***)

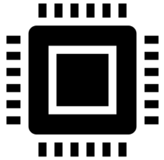
Challenges of Using ML for Estimation

Dataset

MLP	CNN	Transformer
MLP1	CNN1	Transformer1
MLP2	CNN2	Transformer2
MLP3	CNN3	Transformer3
MLP4	CNN4	Transformer4
MLP5	CNN5	Transformer5
…	…	…

Models runs

Execution & Monitoring GPU

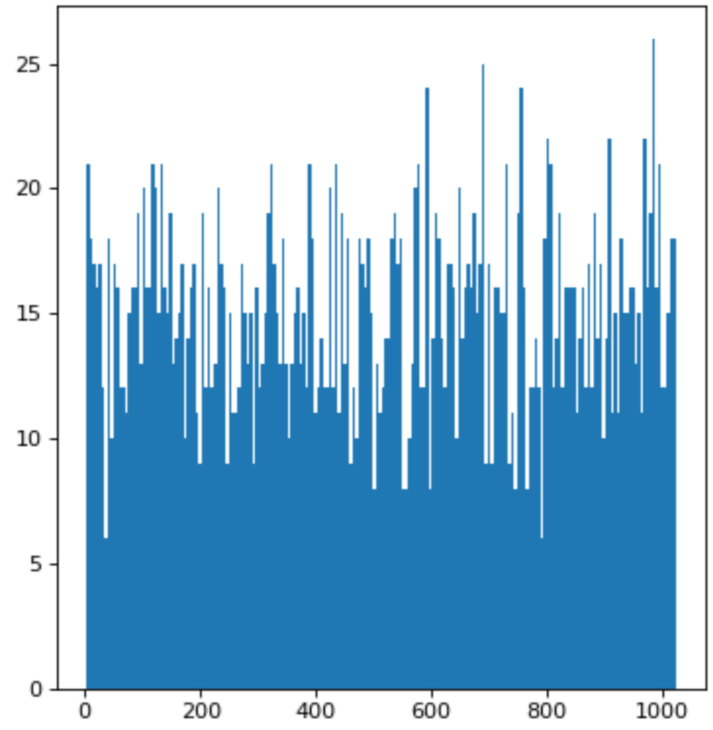


MLP **CNN** **Transformer**

Dataset

Dataset Building

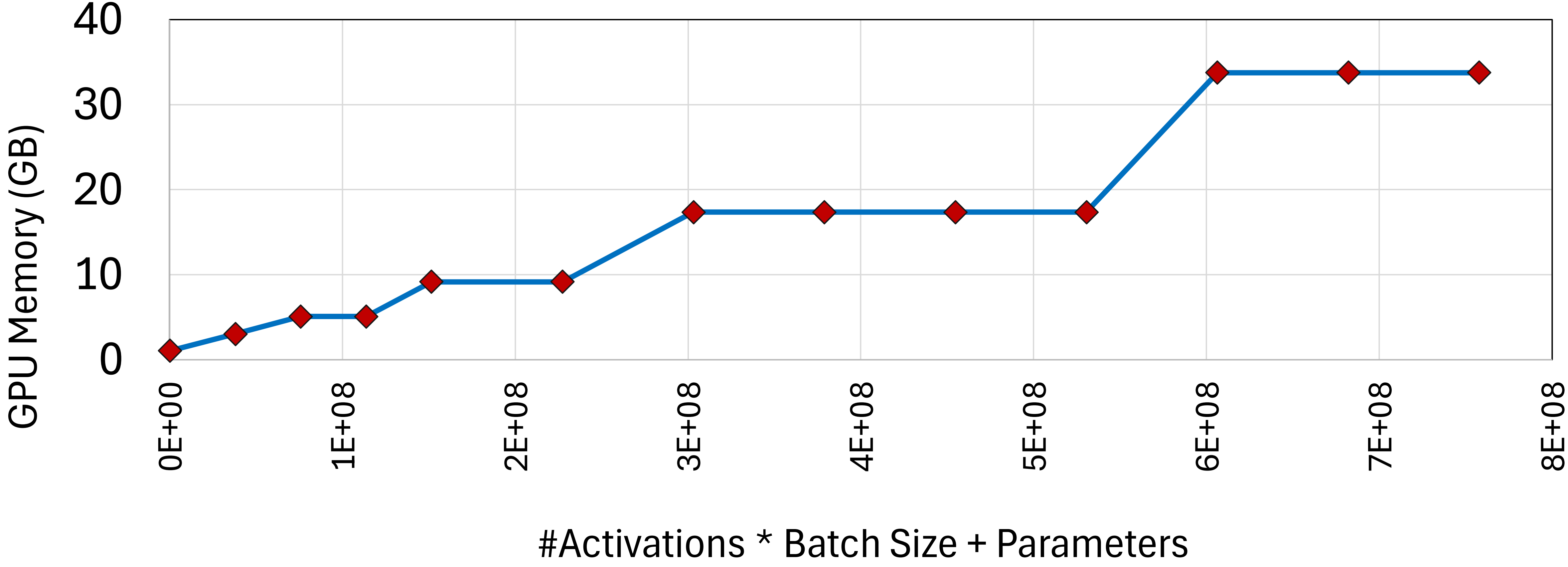
- Representativeness of the key input features
- Uniform feature distribution
- Different architectures (pyramid, uniform, etc.)
- Different layer types (including batch norm, dropout, etc.)
- Varying input and output dimensions



Batch Size

Challenges of Using ML for Estimation

- Regression/ Classification

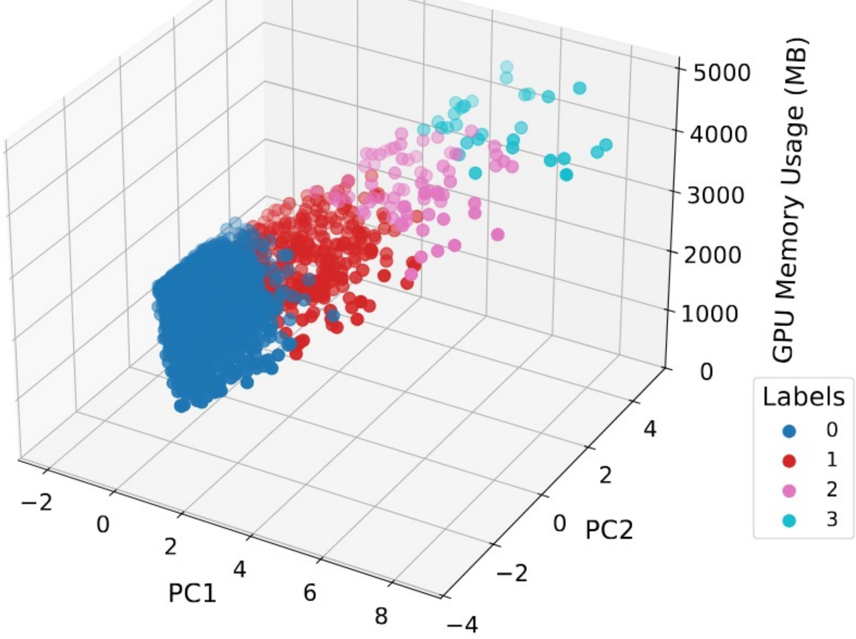


Different MLP model configurations show staircase growth pattern, suitable for classification!

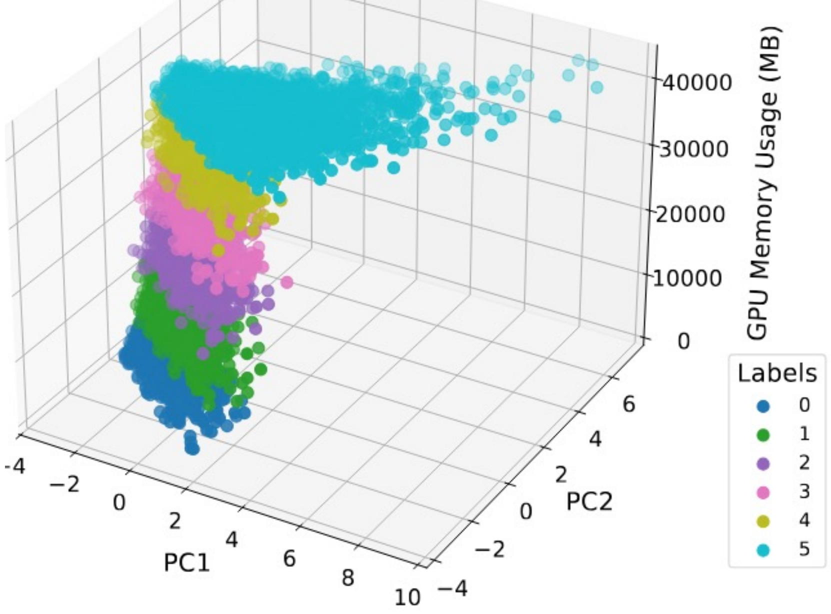
Classification Formulation

- Discretize data into same-GB classes
    - e.g., 0-1GB (1), 1GB-2GB (2), …
- Looked into the data through PCA and t-SNE
    - The classes are observable!

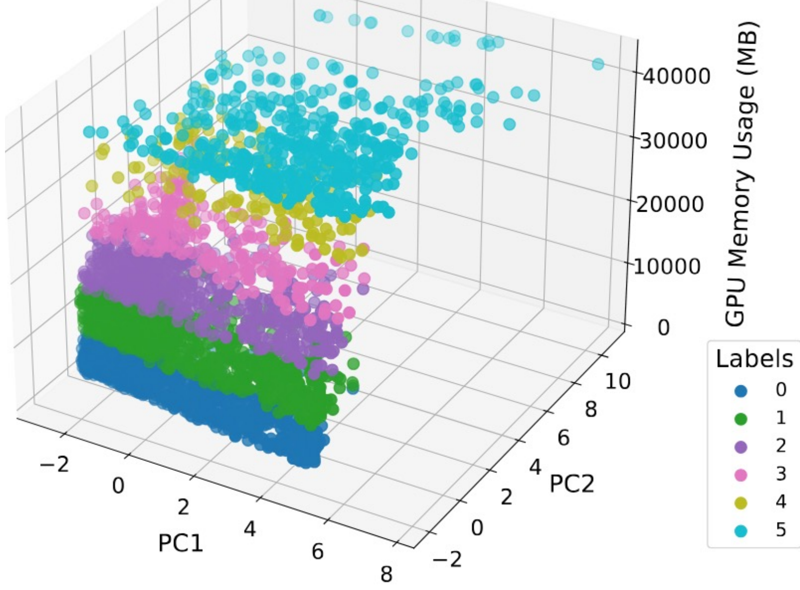
MLP



CNN

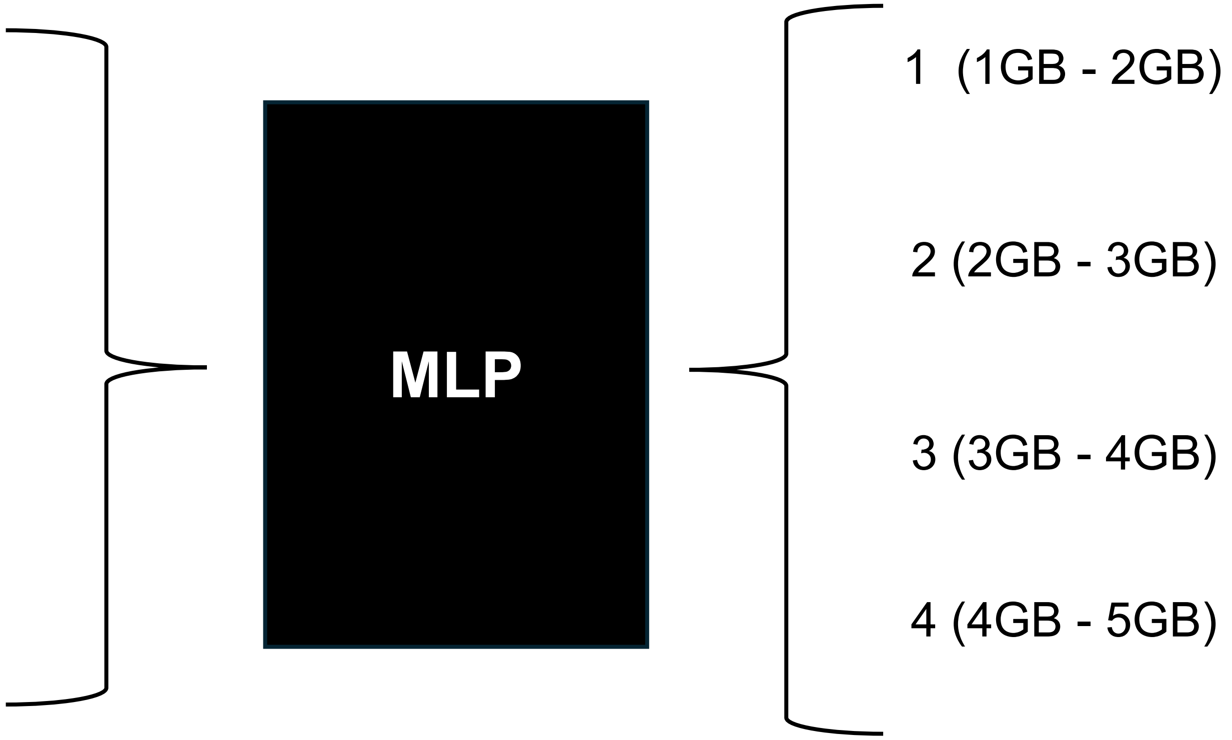


Transformer



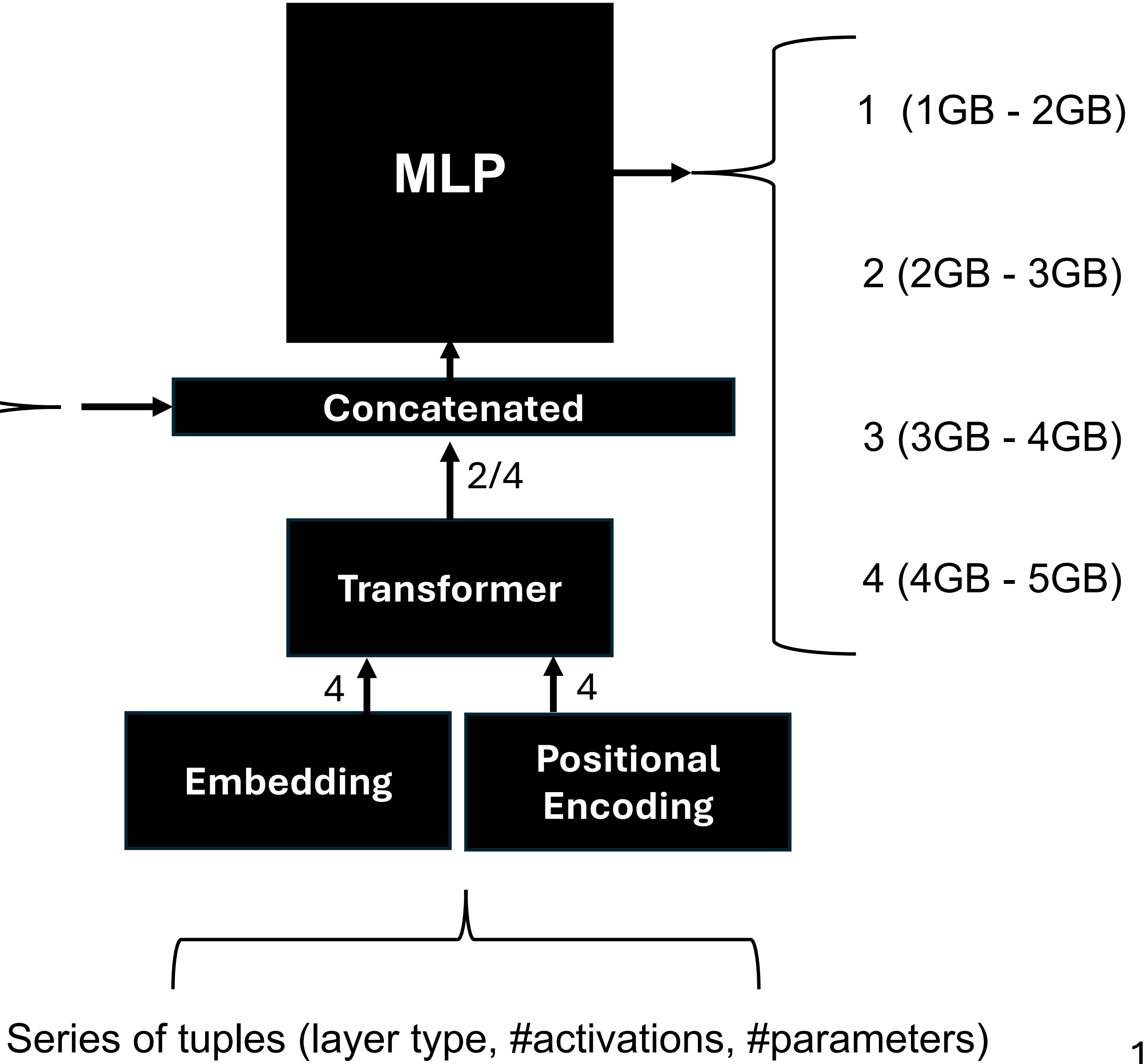
Memory Estimator - MLP-based

- #linear layers
- #batch normalization layers
- #dropout layers
- batch size
- #parameters
- #activations
- activation encoding cos/sin



Memory Estimator - Transformer-based

- #linear layers
- #batch normalization layers
- #dropout layers
- batch size
- #parameters
- #activations
- activation encoding cos/sin



Results

Dataset	Estimator	Class Range Size	#Classes	Accuracy
MLP	MLP	1GB	5	0.95
	MLP	2GB	4	0.97
	Transformer	1GB	5	0.97
	Transformer	2GB	4	0.98

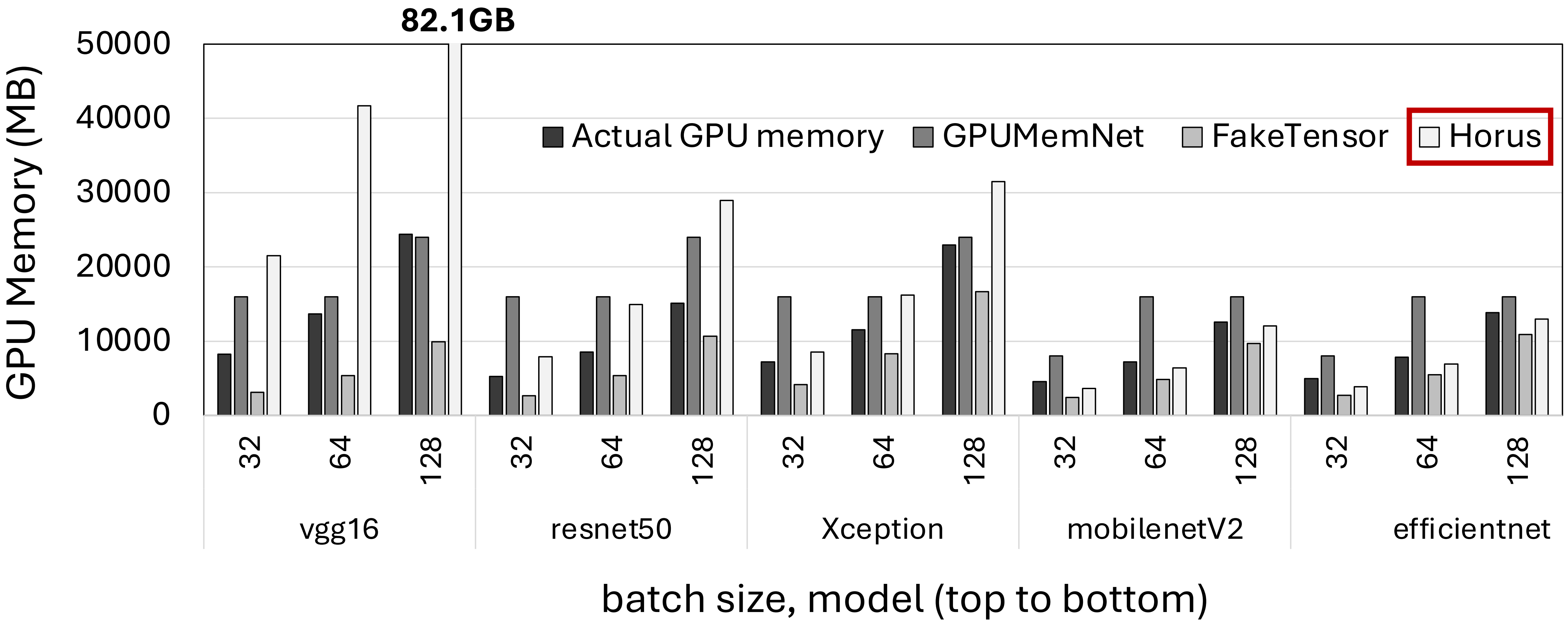
Results

Dataset	Estimator	Class Range Size	#Classes	Accuracy
MLP	MLP	1GB	5	0.95
	MLP	2GB	4	0.97
	Transformer	1GB	5	0.97
	Transformer	2GB	4	0.98
CNN	MLP	8GB	6	0.82
	Transformer	8GB	6	0.81

Results

Dataset	Estimator	Class Range Size	#Classes	Accuracy
MLP	MLP	1GB	5	0.95
	MLP	2GB	4	0.97
	Transformer	1GB	5	0.97
	Transformer	2GB	4	0.98
CNN	MLP	8GB	6	0.82
	Transformer	8GB	6	0.81
Transformer	MLP	8GB	6	0.87
	Transformer	8GB	6	0.85

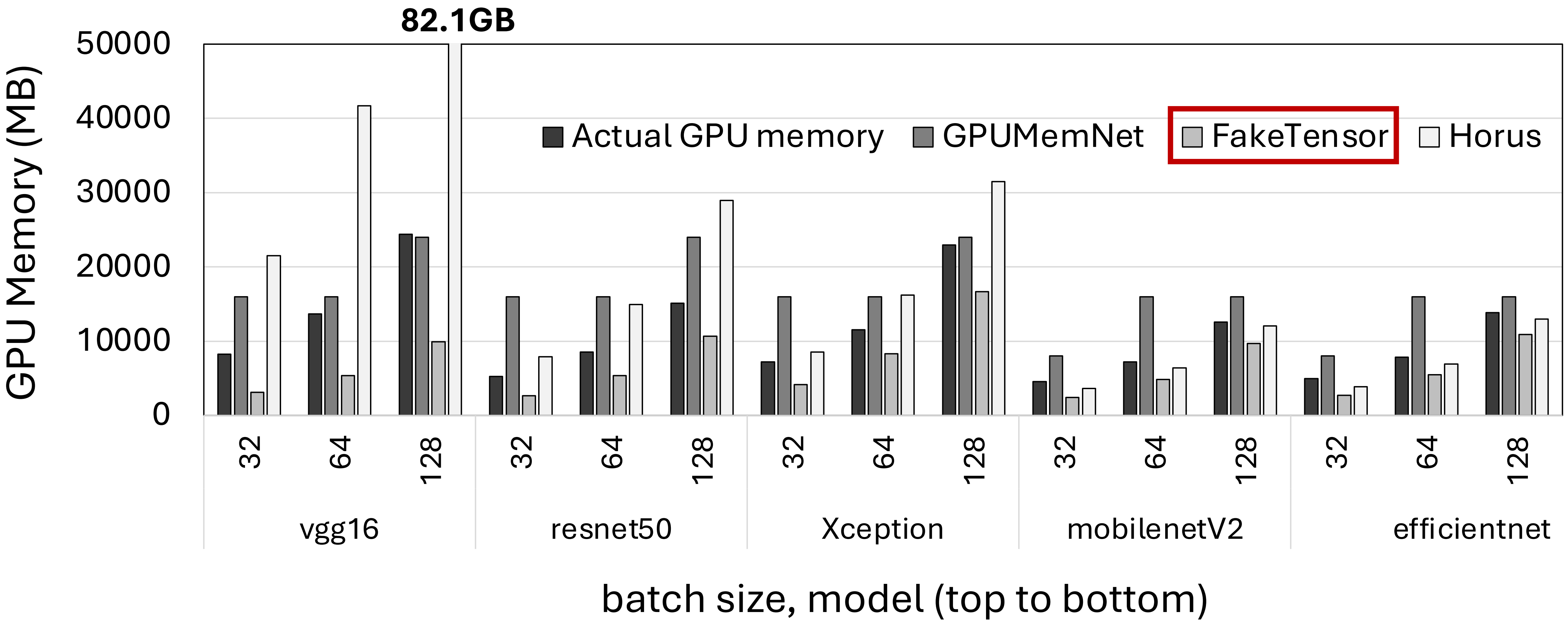
Unseen real-world models



Horus for mobilenetV2 and efficient underestimates ~= OOM crashes

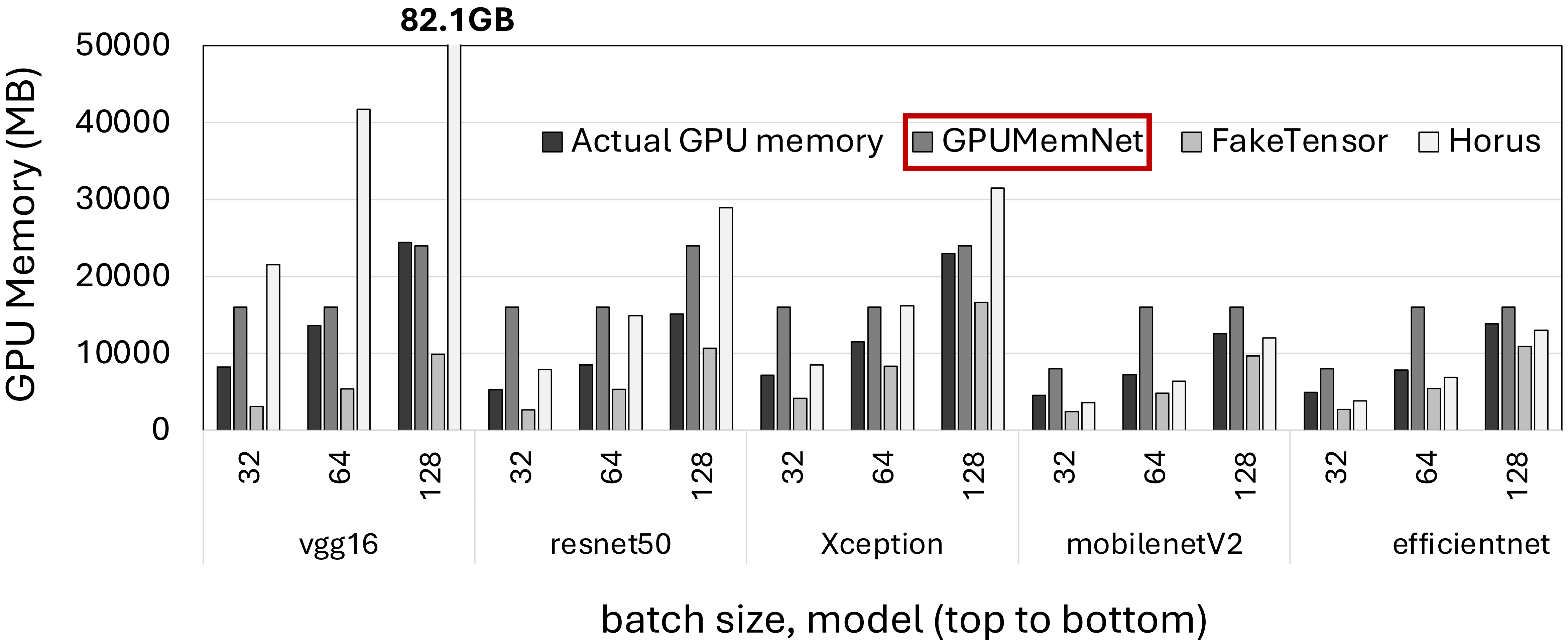
Horus for vgg and resnet50 and Xception overestimates ~= Wastes Optimization Potential

Unseen real-world models



Fake Tensor always underestimates ~= OOM crashes

Unseen real-world models



GPUMemNet closest to actual GPU memory!
Almost never underestimates, preventing OOMs.

Conclusion

- GPUs are underutilized.
- Collocation can be an opportunity.
- GPU memory estimation is needed for more reliable collocation
- GPUMemNet
  - Dataset
  - Tools for extending the dataset

Thanks ☺

Backup Slides

Using Lightweight ML!

- No data to train on → Let's build a dataset

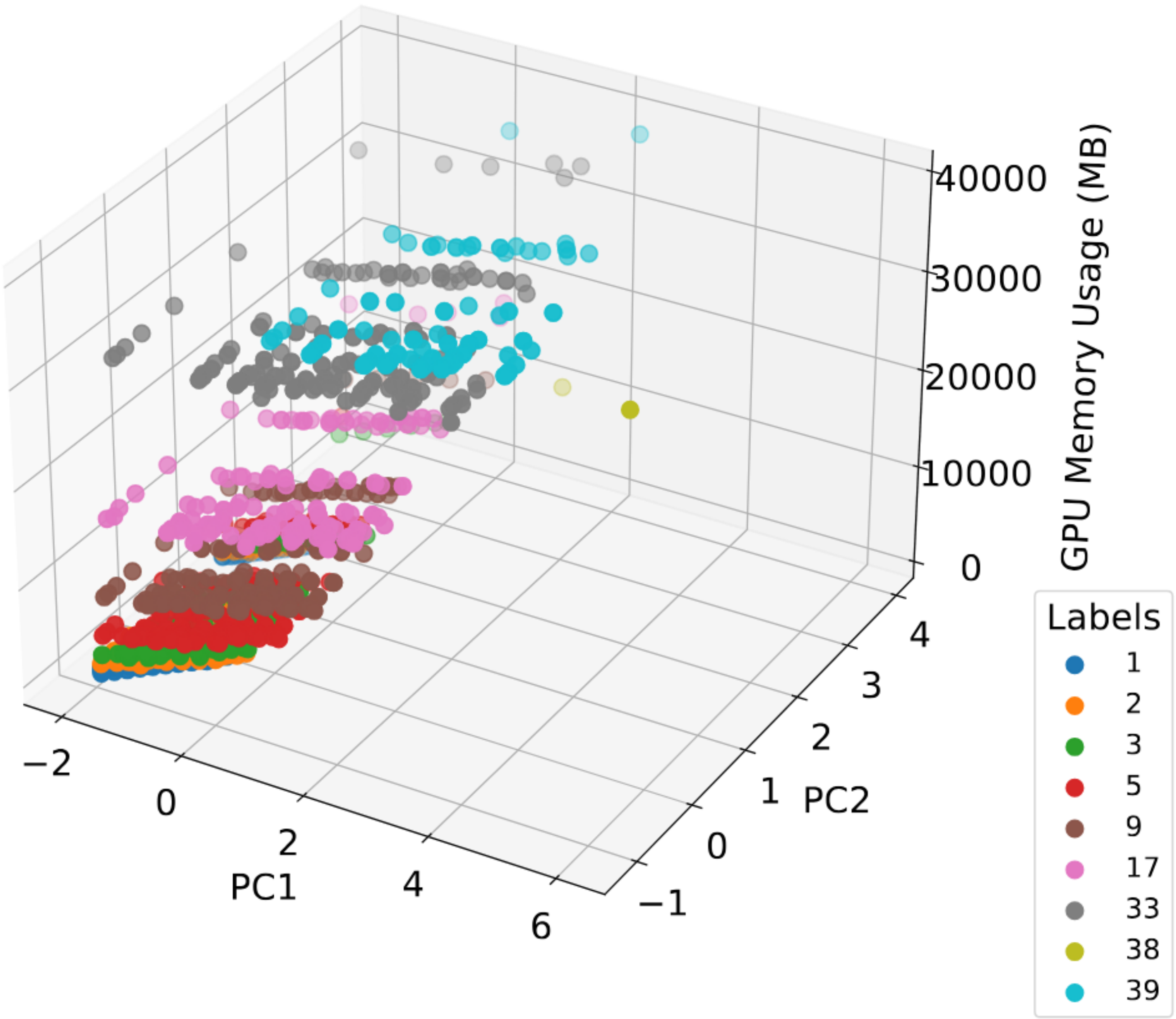
- **Regression**
- Good results on tree-based models
  - **ExtraTreeRegressor** with a +\− ~ 1.2GB error margin
- On unseen data, no reliable
  - feature importance
    - #layers=0.0152, batch_size=0.0144, #parameters=0.699, and #activations=0.271
- Trained an MLP (different loss functions)
  - No convergence!
  - The staircase growth pattern, non-one-to-one function! (**non-identifiability**)

Formulating as Classification

- Labeling data points (0-1GB (1), 1GB-2GB (2), 3GB-4GB (3), …)
- Looked into the data through PCA and t-SNE

- Trained an MLP and observed that the pattern in the data can be learned!

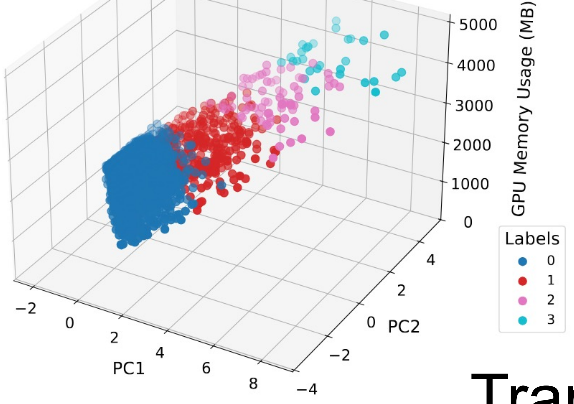
Accuracy	Precision	Recall	F1-Score
0.6909	0.6485	0.6909	0.6520



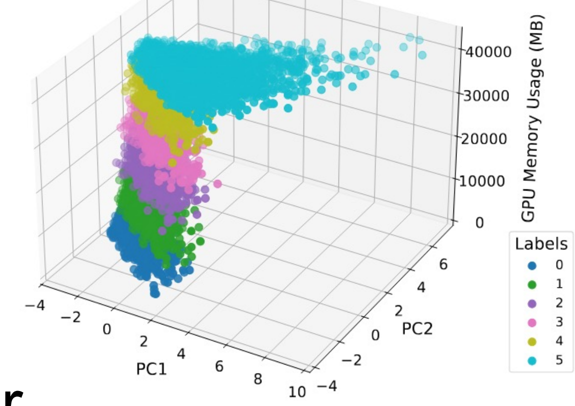
Patterns in the data become easier to detect.

PCA

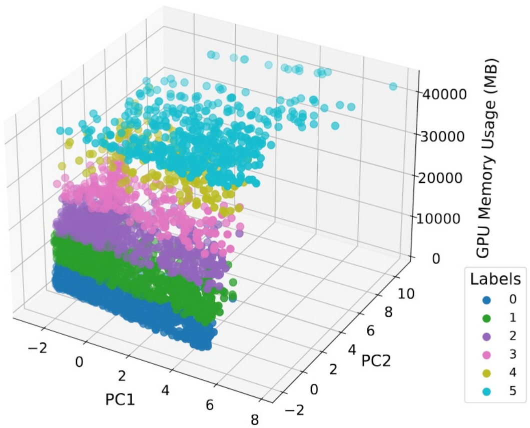
MLP



CNN



Transformer



Obvious data patterns in the data subsets!