Performance Predictive Model for Deep Learning Models

Karthick Panner Selvam, Mats Brorsson
University of Luxembourg

karthick.pannerselvam@uni.lu
mats.brorsson@uni.lu

Deep Learning Everywhere

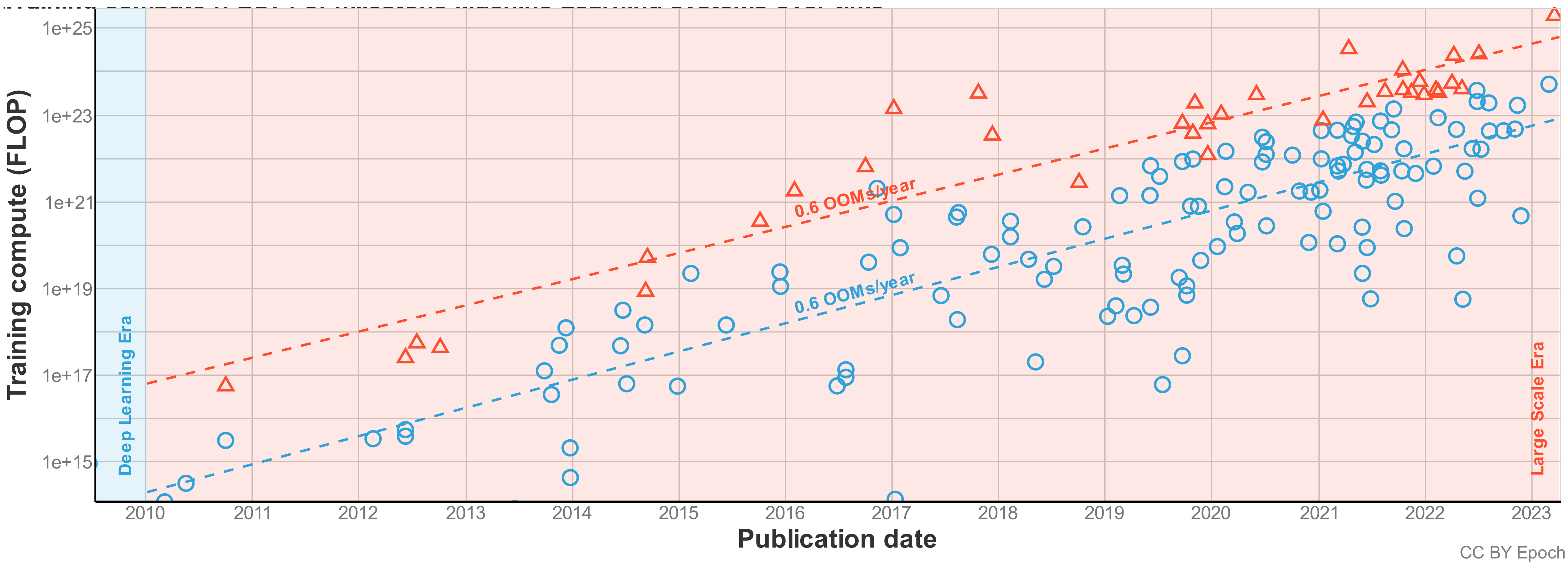
Computer Vision

Natural Language Processing

Robotics

Weather/Climate

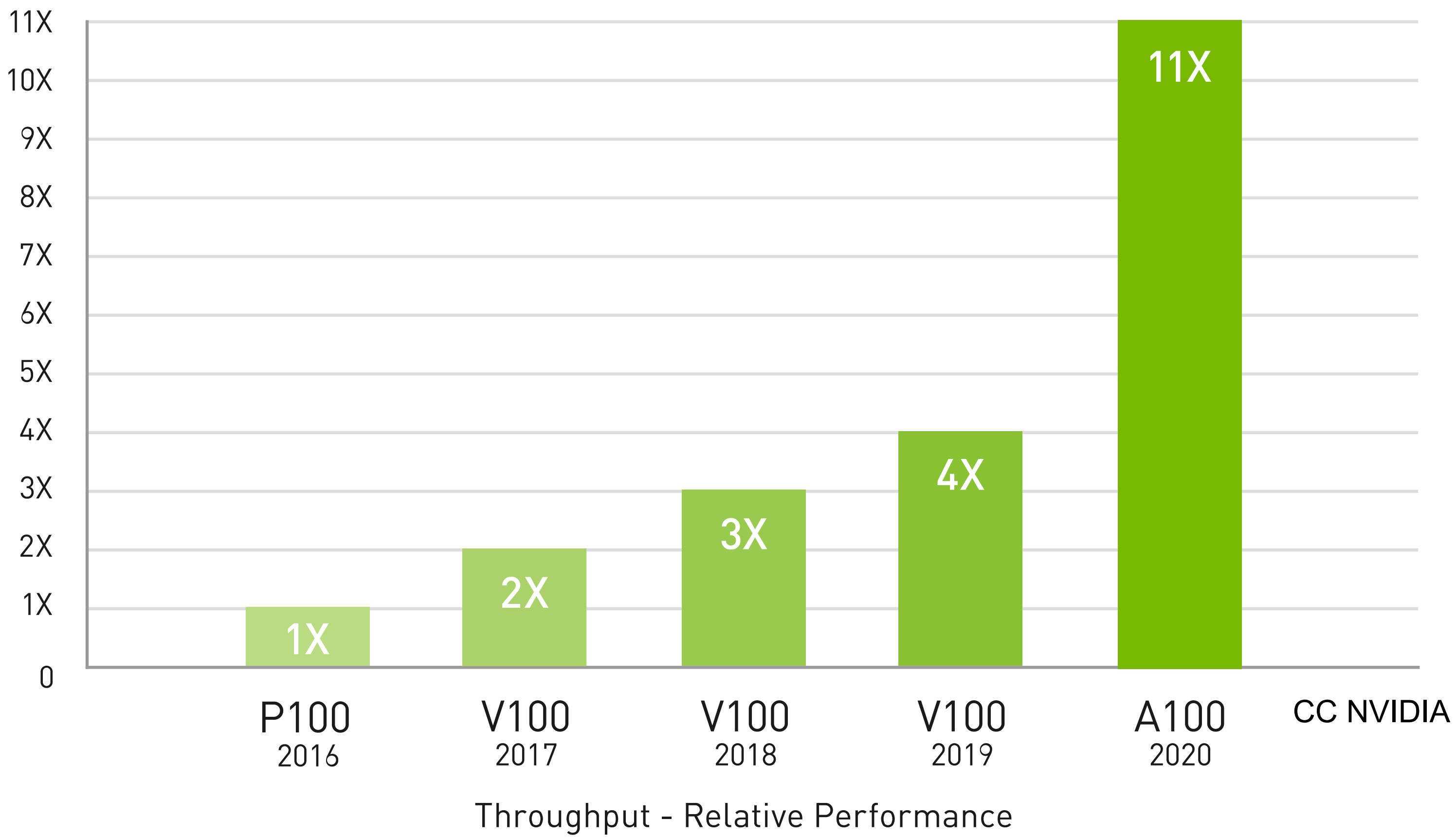
Model complexity steadily increases



Researchers focused on improving the efficiency of deep learning models.

Sevilla et.al. 'Compute Trends Across Three Eras of Machine Learning'. ArXiv [Cs.LG], 2022. arXiv. http://arxiv.org/abs/2202.05924.

Computing power also increases



Choosing the correct hardware can save much money for training and deployment.

NVIDIA Multi-Instance GPU

A100 GPU - 40GB

- 1 x 7g.40gb
  or
- 2 x 3g.20gb
  or
- 3 x 2g.10gb
  or
- 7 x 1g.5gb

To select ideal MIG profile

- 1 x 7g.40gb
  or
- 2 x 3g.20gb
  or
- 3 x 2g.10gb
  or
- 7 x 1g.5gb

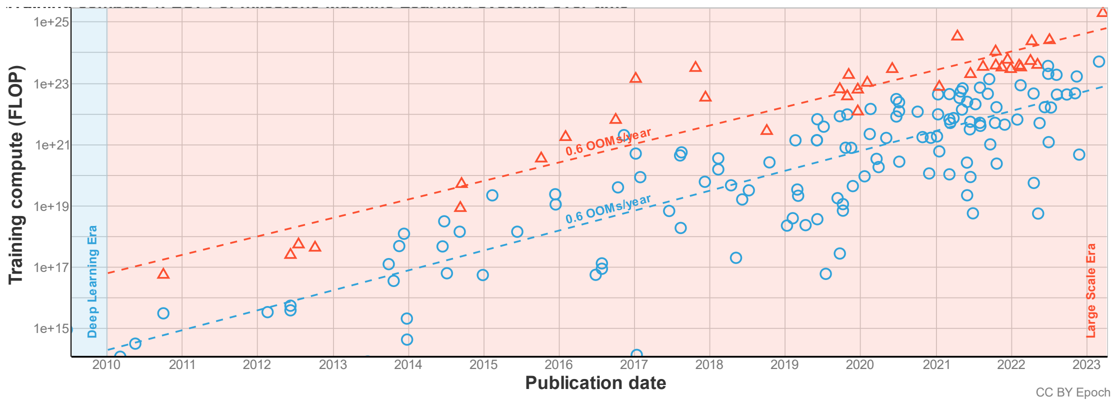
CC NVIDIA

A100 GPU - 40GB

If we already know…

- **Memory usage of the DL model**

Develop efficient DL models



CC BY Epoch

If we already know…

- **Model latency**
- **Power consumption**

while developing the DL model

Why not just directly measure it on GPU ?

Payment is required to access the GPU(s).

It’s tedious to replicate for multiple models.

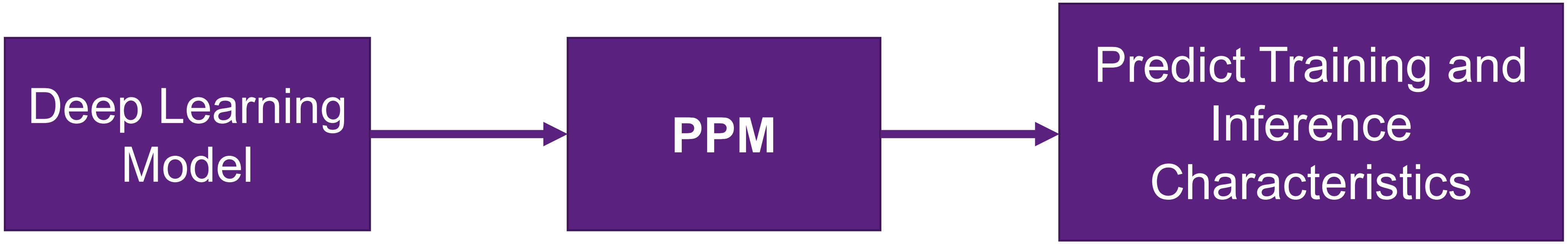
Performance Predictive Model

Deep Learning Model → **PPM** → Predict Training and Inference Characteristics

Predicted parameters help to

- Better resource allocation
- Cost saving
- Neural Architecture Search

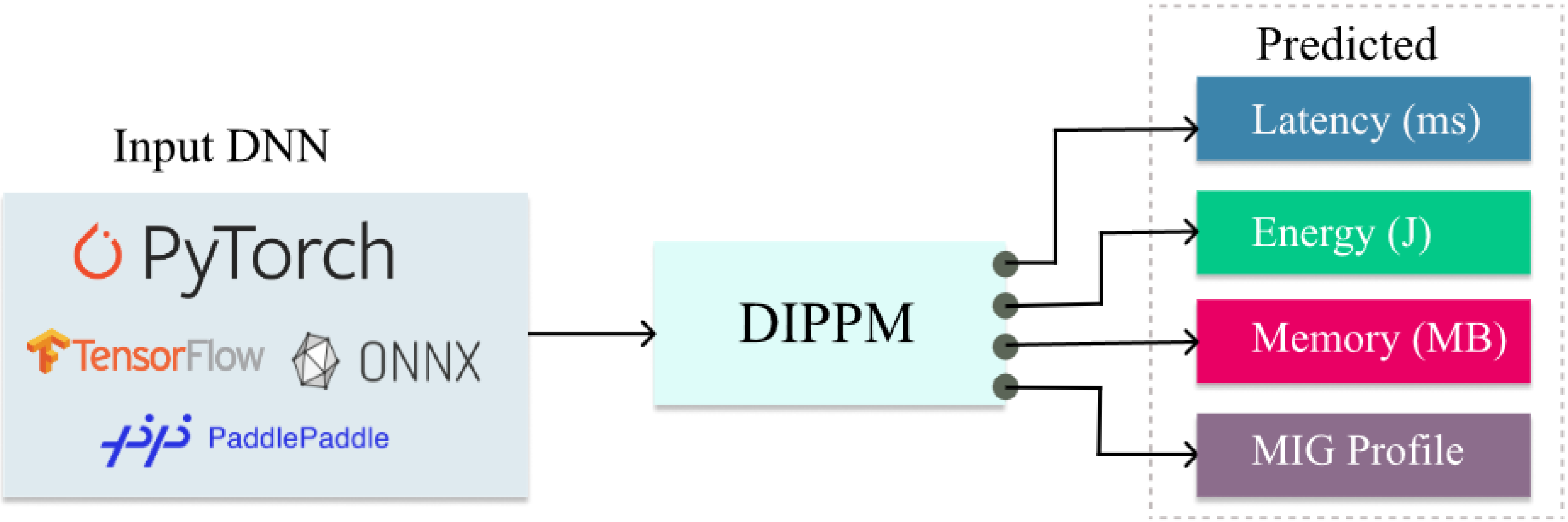
Performance Predictive Model



1. *DIPPM: a Deep Learning Inference Performance Predictive Model using Graph Neural Network* – EuroPAR 2023

2. *Can Semi-Supervised Learning Improve Prediction of Deep Learning Model Resource Consumption?* – NeurIPS 2023 MLSys workshop

Deep Learning Inference Performance Predictive Model

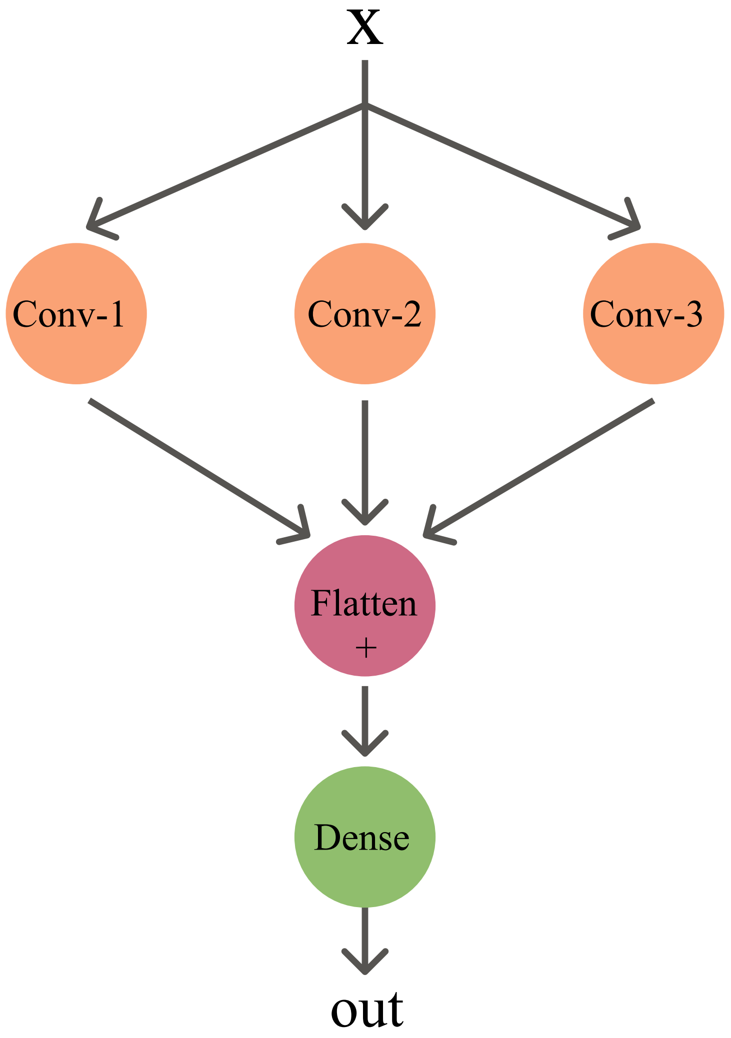


DIPPM predicts Inference characteristics and MIG profile
without running it on target hardware

Background

1. DL Computational Graph
2. Graph Neural Network

Background – DL Computational Graph



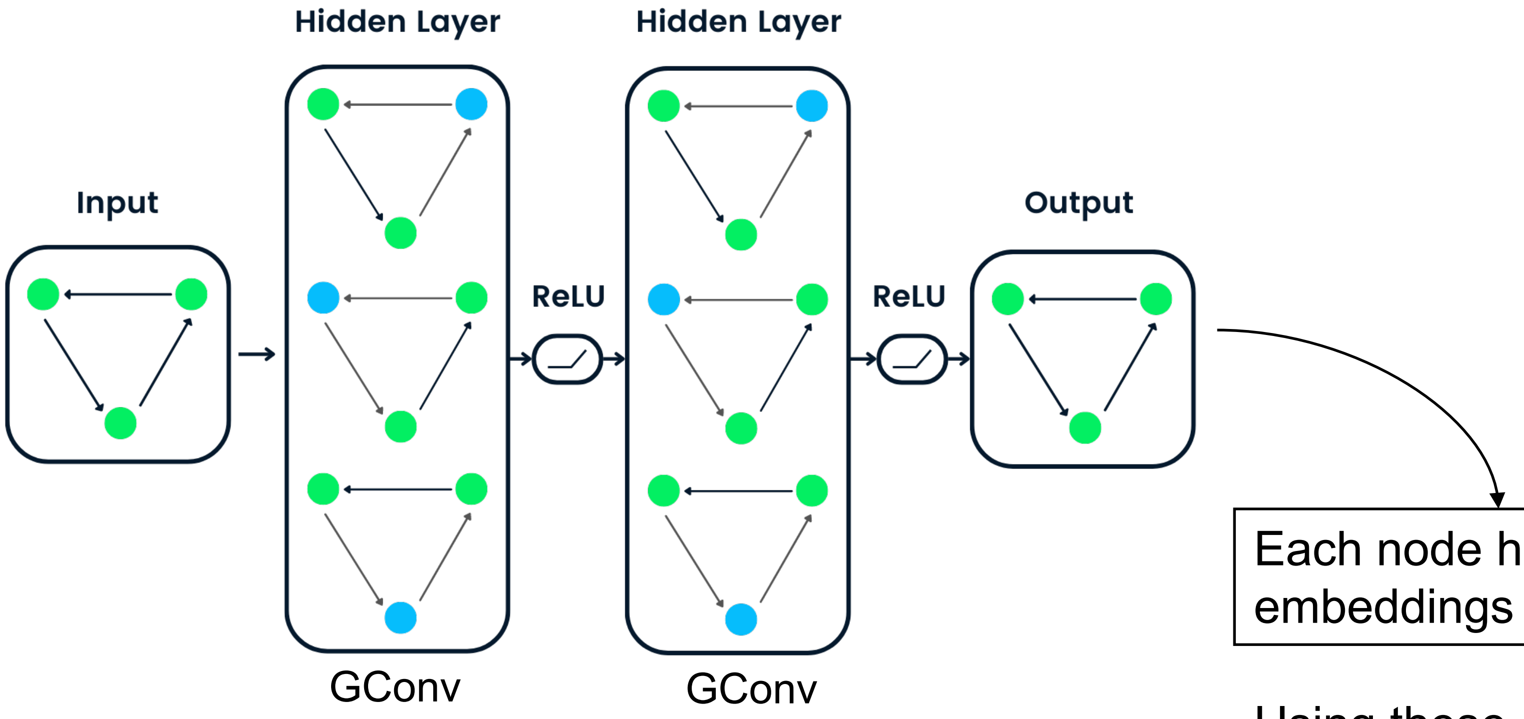
Simple CNN

Nodes (V) = Mathematical operators

Edges (E) = Data flow between nodes

$$\mathcal{G} = (V, E)$$

Background – Graph Neural Network



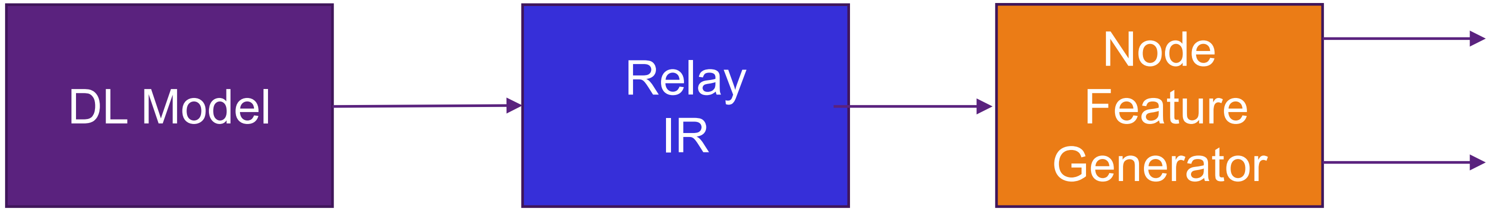
Each node has node embeddings

Using these embeddings
- Graph level task
- Node level task

cc datacamp

How to represent the DL model as input?

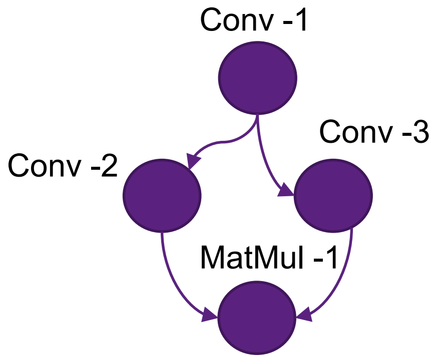
Where X is the Shape of
[*Number of Nodes, Number of features*]



X (Node Feature Matrix)

A (Adjacency Matrix)

TensorFlow, PyTorch…



$A[i][j] = 1$ If directed edge

$A[i][j] = 0$ Otherwise

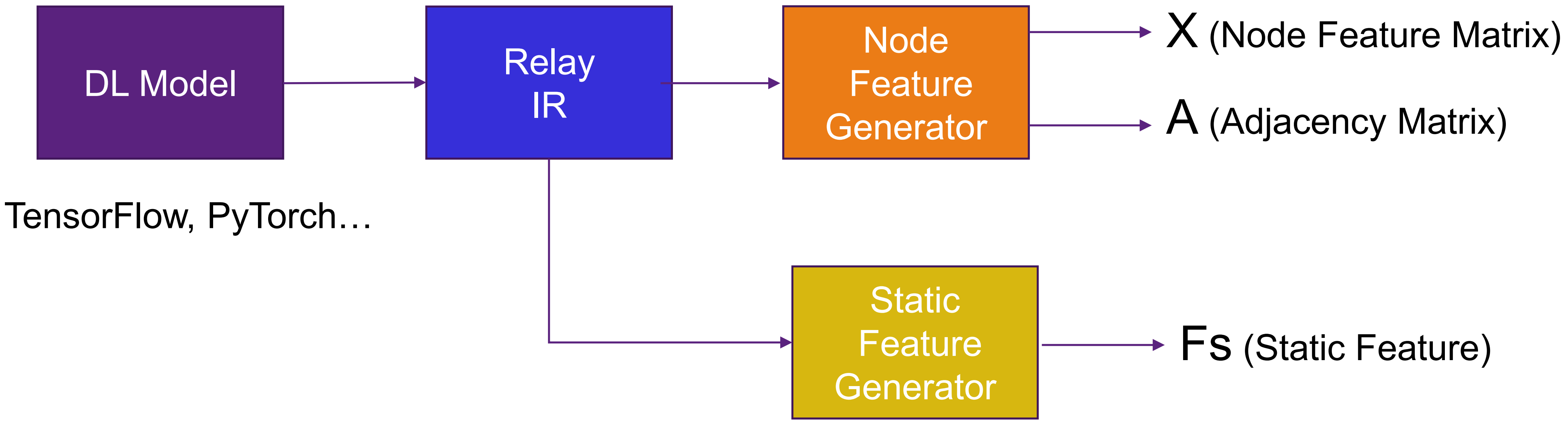
Node Features:

$$\mathcal{F}_{node} \leftarrow \mathcal{F}_{oh} \oplus \mathcal{F}_{attr} \oplus \mathcal{F}_{shape}$$

Length of Node Features: 32

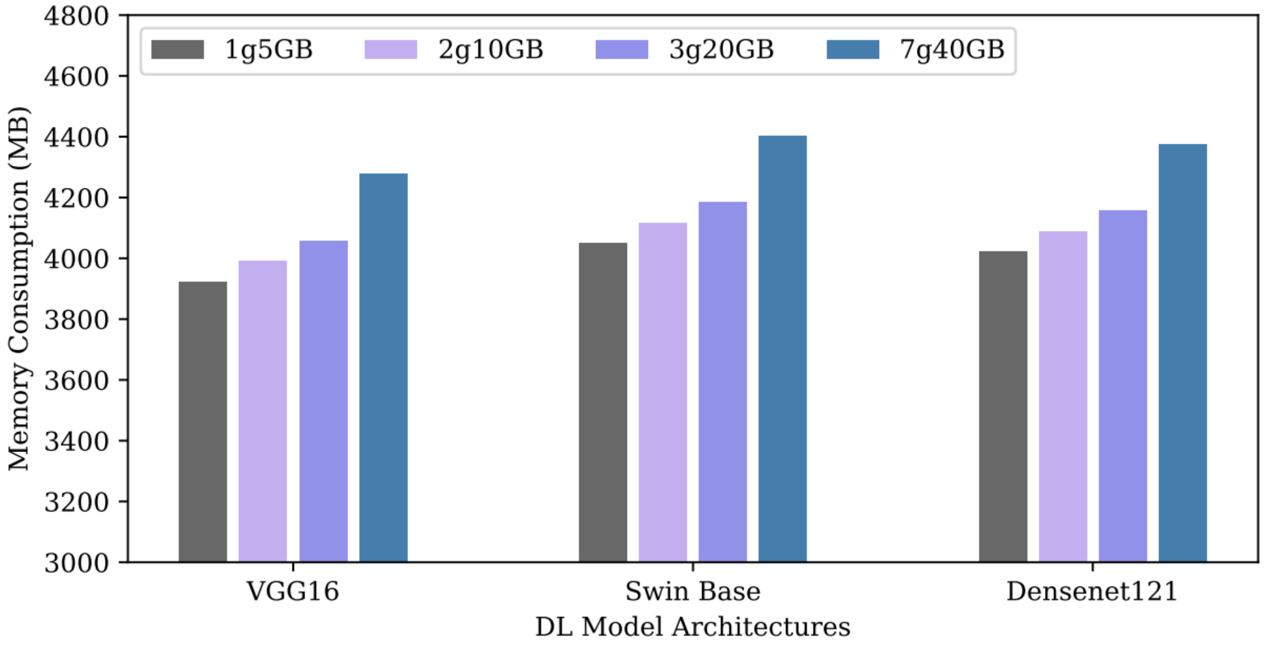
How to represent the DL model as input?

Where X is the Shape of
[*Number of Nodes, Number of features*]



$$\mathcal{F}_s \leftarrow \mathcal{F}_{mac} \oplus \mathcal{F}_{batch} \oplus \mathcal{F}_{Tconv} \oplus \mathcal{F}_{Tdense} \oplus \mathcal{F}_{Trelu}$$

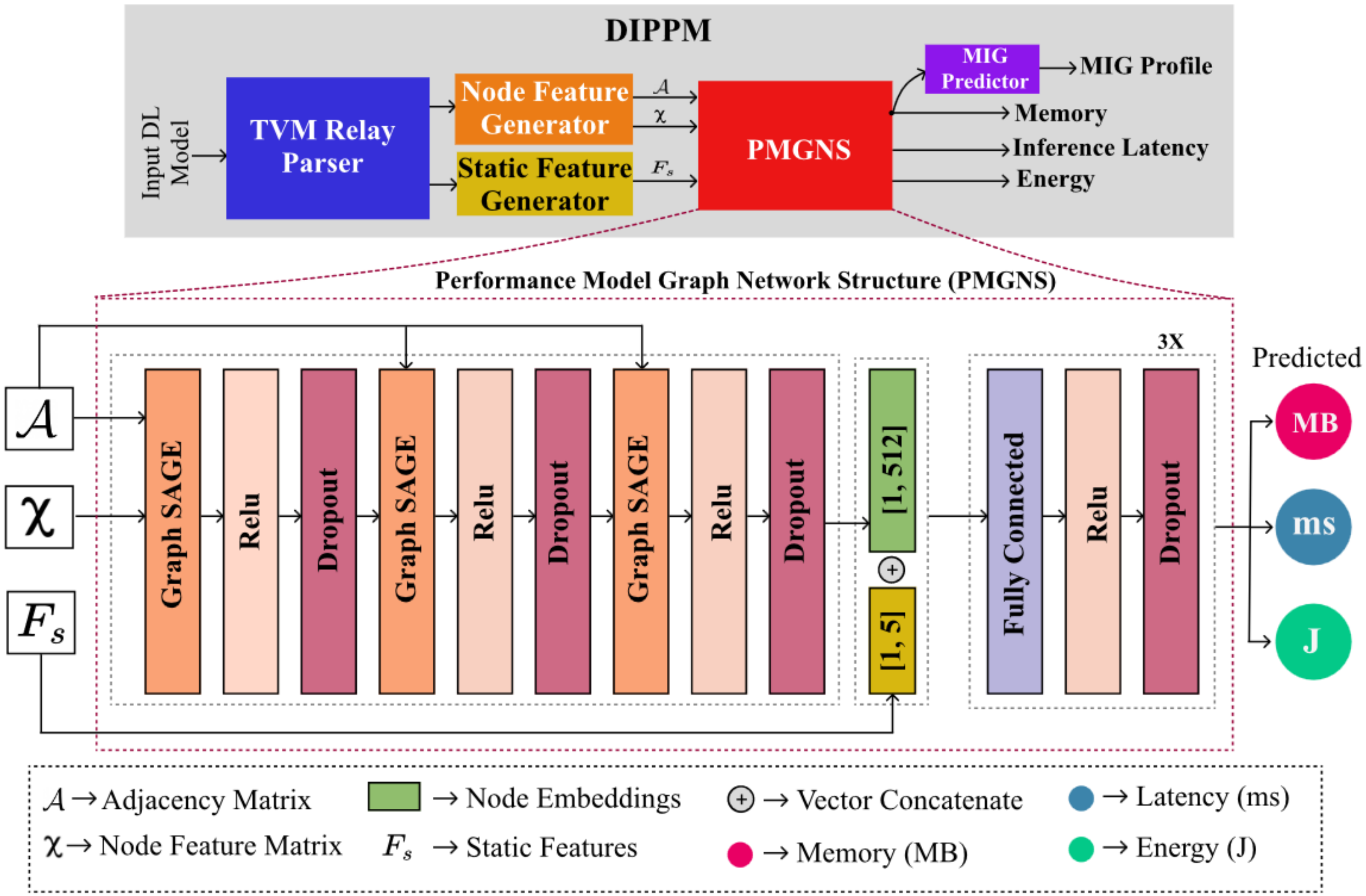
DIPPM: MIG Predictor



$$\mathrm{MIG}(\alpha) = \begin{cases} \text{1g.5gb}, & \text{if } 0gb < \alpha < \text{5gb} \\ \text{2g.10gb}, & \text{if } \text{5gb} < \alpha < \text{10gb} \\ \text{3g.20gb}, & \text{if } \text{10gb} < \alpha < \text{20gb} \\ \text{7g.40gb}, & \text{if } \text{20gb} < \alpha < \text{40gb} \\ \text{None}, & \text{otherwise} \end{cases}$$

The memory consumption is always the highest when running on the 7g.40gb MIG profile. So, we claim that predicted memory will be an upper bound.

DIPPM: a Deep Learning Inference Performance Predictive Model using Graph Neural Network



DIPPM Architecture

DIPPM Dataset

- We used NVIDIA A100 GPU to collect the dataset. From 10 different model families, a total of **10508** graphs were collected.

- We used NVML and CUDA API to measure *Inference time*, *Memory,* and *Energy.*

Each graph contains

1. Node Features Matrix
2. Adjanceny Matrix
3. Target variable
4. Static Features

DIPPM: Results

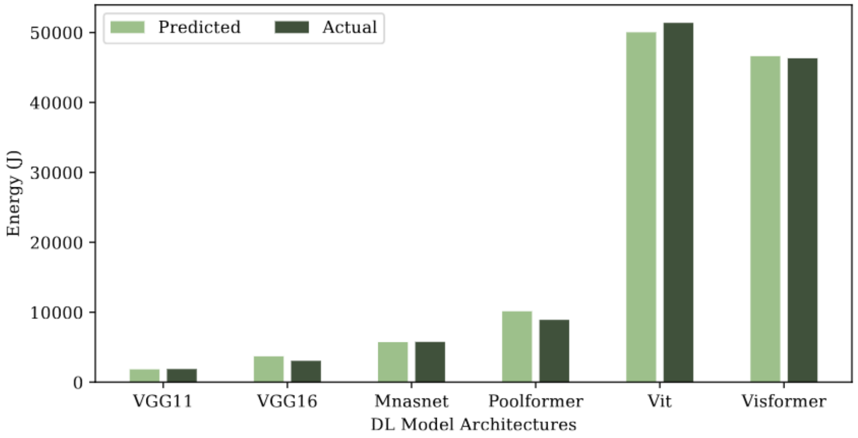
Model	Training Loss	Validation Loss
GAT	0.4966	0.3793
GCN	0.2122	0.1776
GIN	0.4880	0.3939
MLP	0.3714	0.3874
(Ours) GraphSAGE	0.1824	0.1587

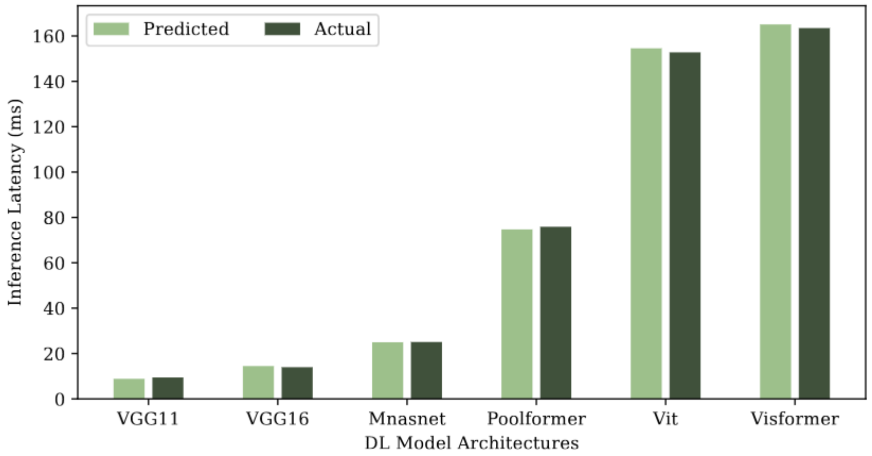
In comparison with different GNN algorithms and MLP, we trained for ten epochs.

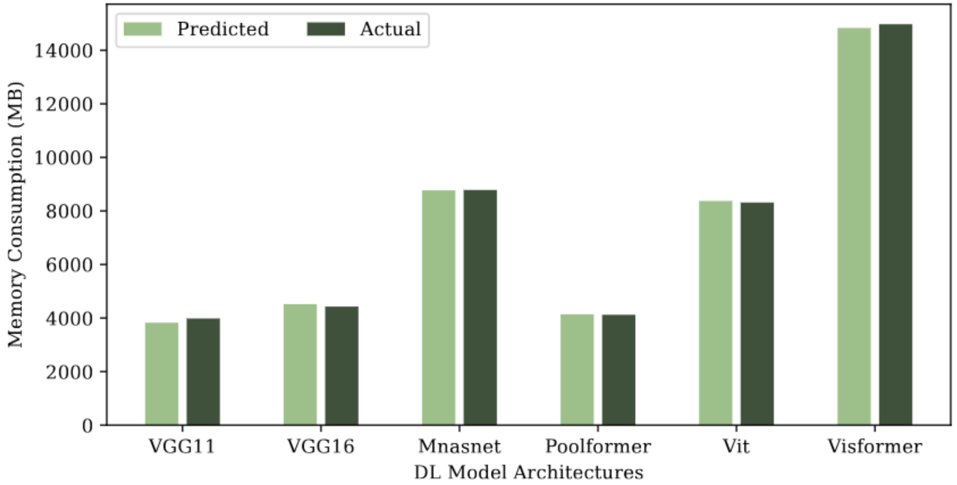
The results indicate that DIPPM with graphSAGE performs significantly better than other variants.

After 150 epochs, we achieved 1.9% MAPE on our test dataset.

DIPPM: Results







Results show that DIPPM predictions are close to the actual predictions.

DIPPM Usability

example.py M ×

example.py > ...

```python
import dippm
import torchvision

model = torchvision.models.vgg16(pretrained=True)
model.eval()

#current dippm supports only A100 GPU
out  = dippm.predict(model, batch=8, input="3,244,244", device="A100")

print("====================================")
print("Predicted Memory {0} MB, Energy {1} J, Latency {2} ms, MIG {3}".format(*out))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1

```
(multi) karthick@UNIDBP9NG3:~/work/dippm$
```

An example code demonstrating the utilization of DIPPM for performance prediction of a VGG16 DL model with a batch size of 8.

DIPPM: Summary

We developed a novel performance model to predict the *Inference characteristics* and *MIG profile* from a given input DL model from *various frameworks*.

TraPPM

<u>Motivation:</u>

Most prior studies, including DIPPM, utilized supervised techniques for performance prediction, *<u>neglecting the vast pool of unlabelled DL model data</u>*.

<u>Our innovative approach</u>, TraPPM, bridges this gap using a semi-supervised learning paradigm, enhancing prediction accuracy by harnessing unlabelled data.

Train Unsupervised GAE with unlabelled data → Use GAE to generate embedding for labelled data → Train the supervised GNN along with GAE embedding

Can Semi-Supervised Learning Improve Prediction of Deep Learning Model Resource Consumption? – NeurIPS 2023 MLSys workshop

TraPPM Dataset

- We used NVIDIA A100 and V100 to collect the dataset. From 11 different model families.
- We used NVML and CUDA API to measure *Training step time*, *Memory,* and *Power usage.*

Each graph contains

1. Node Features Matrix
2. Adjanceny Matrix
3. Target variable *(only for supervised)*
4. Static Features

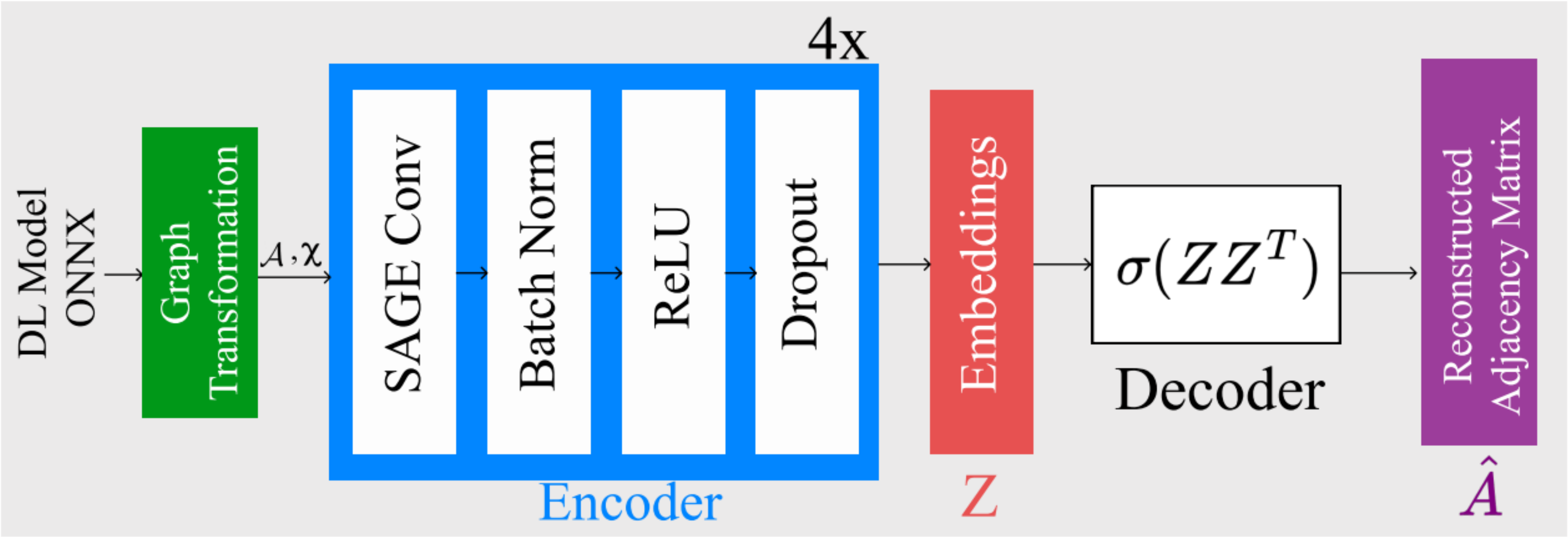
$OneHot(Op_v)$	I_v	O_v	Mac_v	P_v	M_v
98	6	6	1	1	1

The graph's nodes are augmented with node features, each consisting of 113 elements.

Family	Unsupervised	Supervised	
		A100	V100
densenet	838	466	27
efficientnet	1370	566	44
mnasnet	7208	795	64
mobilenet	2449	1613	123
poolformer	601	377	36
resnet	1805	821	56
swin	787	421	36
vgg	6171	937	61
visformer	237	235	17
convnext	1530	439	27
vit	2057	866	52
Total	**25053**	**7536**	**543**

TraPPM: Unsupervised Learning

Graph Auto Encoder

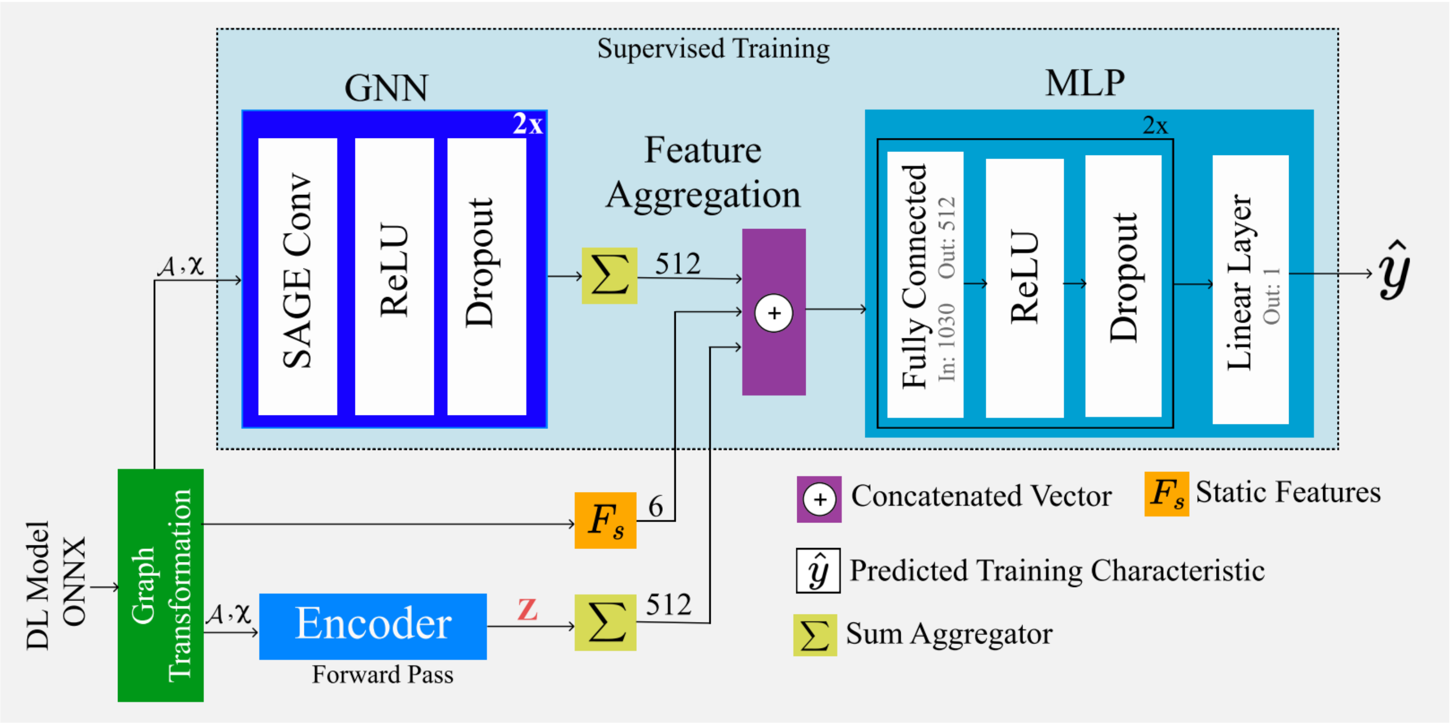


Training Graph Auto Encoder to minimize reconstruction loss of unlabelled DL model graphs

$$L_{\mathrm{BCE}} = -\log(\hat{A}(z, i_{\mathrm{pos}}, j_{\mathrm{pos}}) + \epsilon) - \log(1 - \hat{A}(z, i_{\mathrm{neg}}, j_{\mathrm{neg}}) + \epsilon)$$

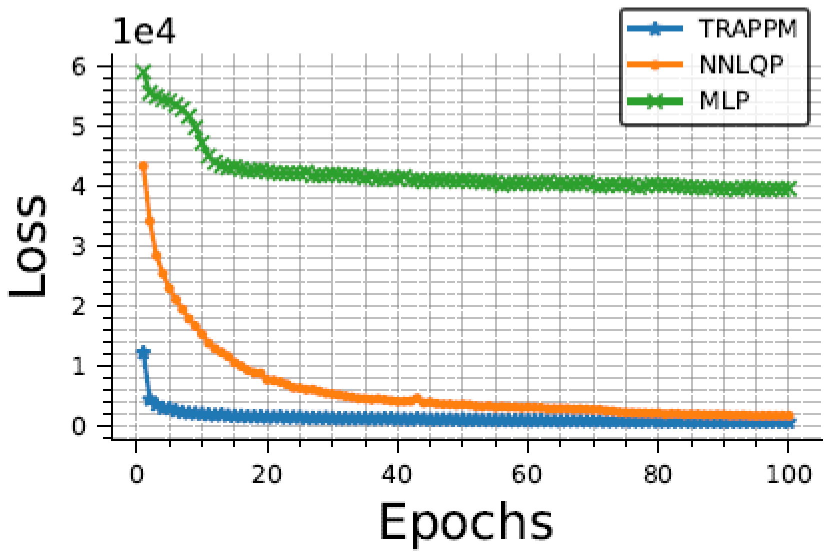
TraPPM: Supervised Learning

Supervised GNN

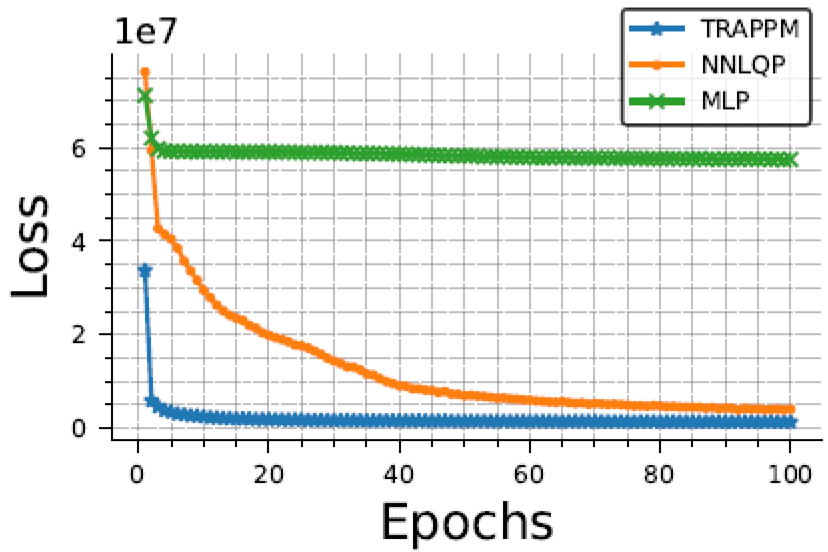


Training a GNN regressor using MSE loss to minimize the actual y vs. predicted y.

TraPPM: Results



(a) Step Time (ms)



(b) Memory Usage (MB)

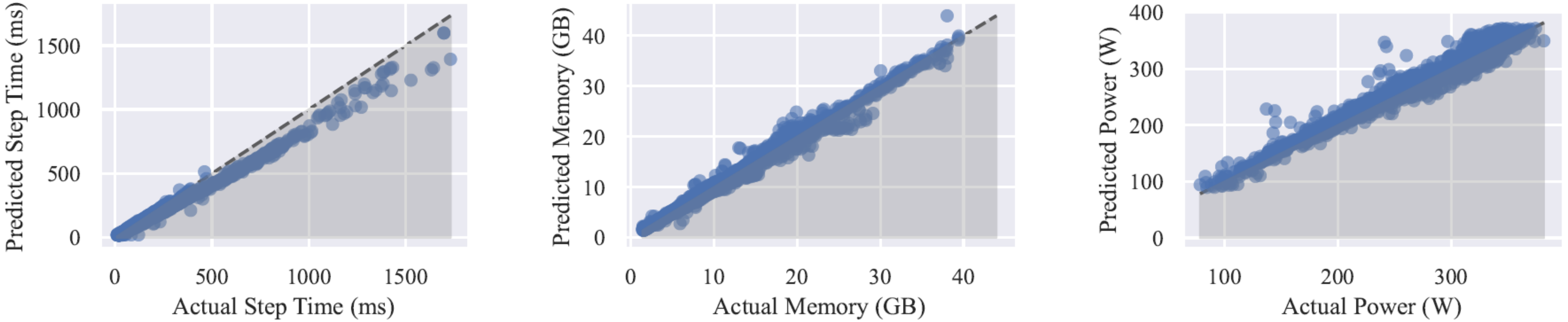
***Epoch vs Loss plot** comparing the convergence rates of TraPPM, NNLQP, and MLP. TraPPM showcases rapid convergence due to its ability to leverage unsupervised learning from unlabeled data.*

TraPPM: Results

Model	Memory Usage (MB)		Step Time (ms)	
	MAPE	RMSE	MAPE	RMSE
TraPPM	**4.92%**	**910.34**	**9.51%**	**23.23**
NNLQP	8.29%	1688.18	14.47%	37.02
MLP	85.01%	8045.68	134.07%	188.36
GBoost	16.10%	2971.52	16.98%	54.54

Average Performance Comparison of TraPPM with Baseline Models. The lower the value, the higher the accuracy.

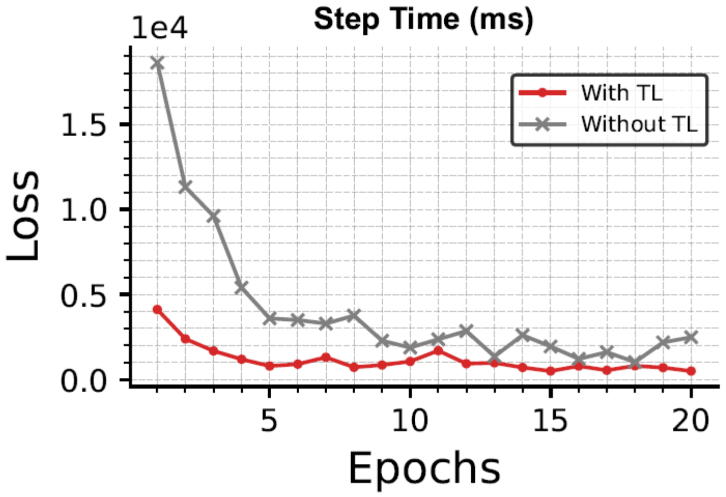
TraPPM: Results

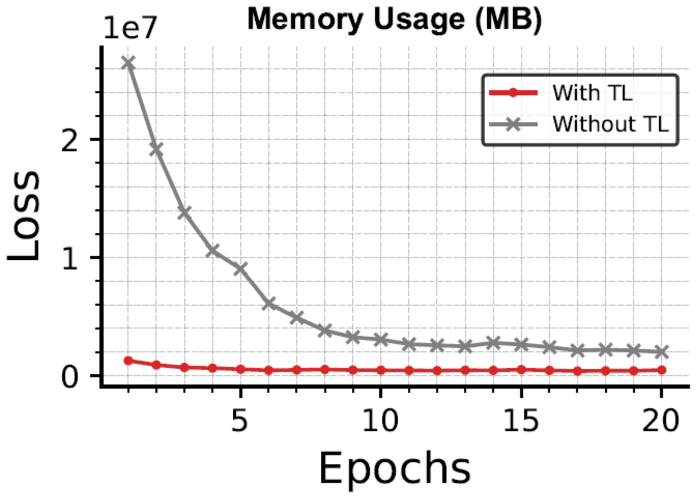


Comparison of actual values with predictions from TraPPM on the test set

TraPPM: Transfer Learning Results

Epoch vs. Loss plot demonstrating TraPPM's enhanced convergence through transfer learning.





Target variable	Metric	With TL	Without TL
Step Time	MAPE	**19.13%**	28.24%
	RMSE	**20.05 ms**	44.59 ms
Memory Usage	MAPE	**11.22%**	28.49%
	RMSE	**603.03 MB**	1176.90 MB

V100 Prediction results on the test dataset using with and without TL

TraPPM Usability

```python
import trappm

trappm.predict("resnet101_32.onnx")
```

TraPPM Performance Prediction Report

GPU Metrics	A100 GPU Prediction
Train Memory	6633.54 Mb
Train Power	266.5 W
Train Step Time	58.52 ms
Inference Step Time	15.47 ms

Code: https://github.com/karthickai/trappm

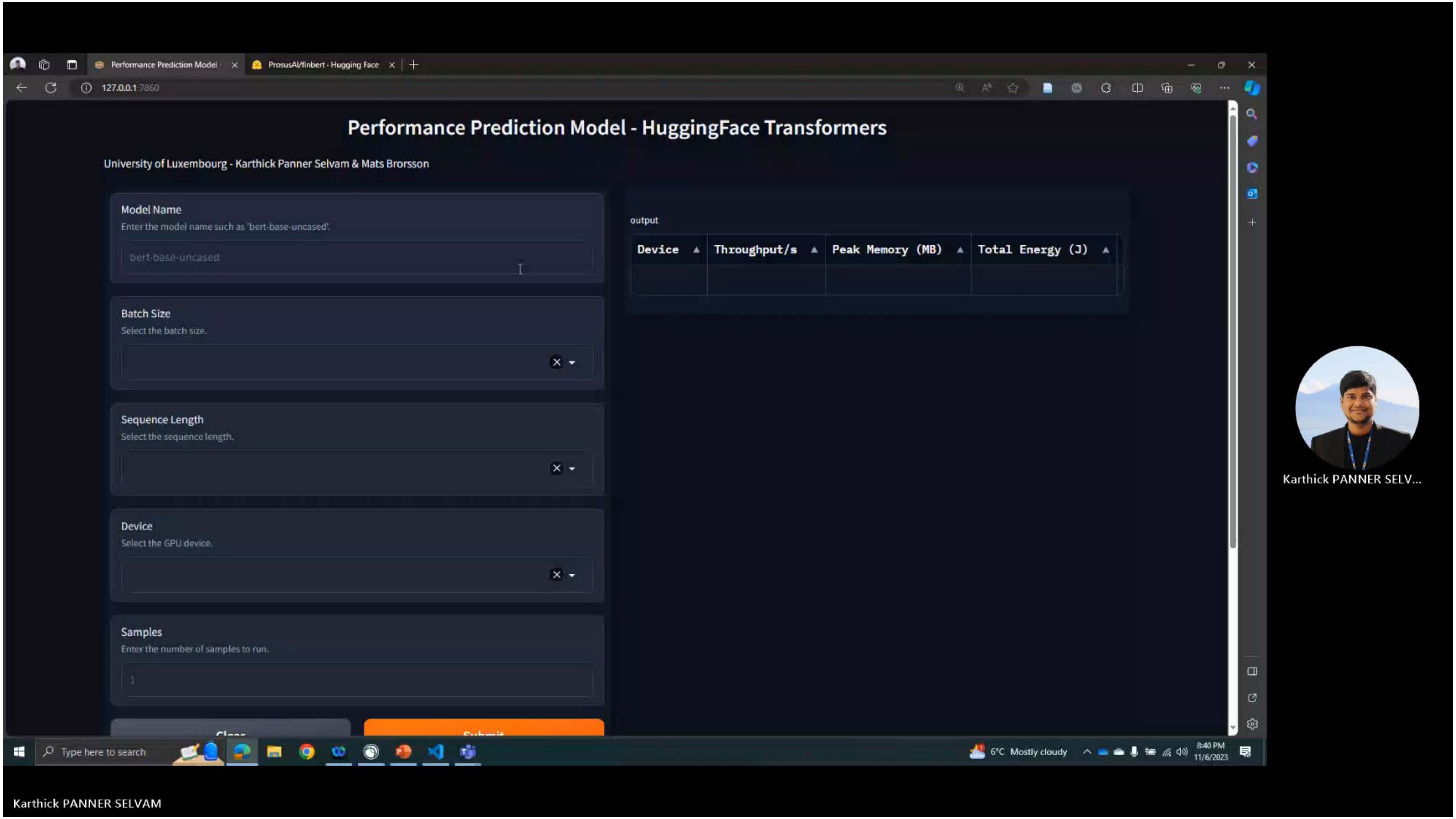
Summary

In DIPPM[1], we developed a novel performance model to predict the Inference characteristics and MIG profile.

In TraPPM[2], we utilized semi-supervised learning to use unlabeled data to enhance performance accuracy.

1. *DIPPM: a Deep Learning Inference Performance Predictive Model using Graph Neural Network* – EuroPAR 2023

2. *Can Semi-Supervised Learning Improve Prediction of Deep Learning Model Resource Consumption?* – NeurIPS 2023 MLSys workshop



Performance Transformer – Ongoing research

Thank You