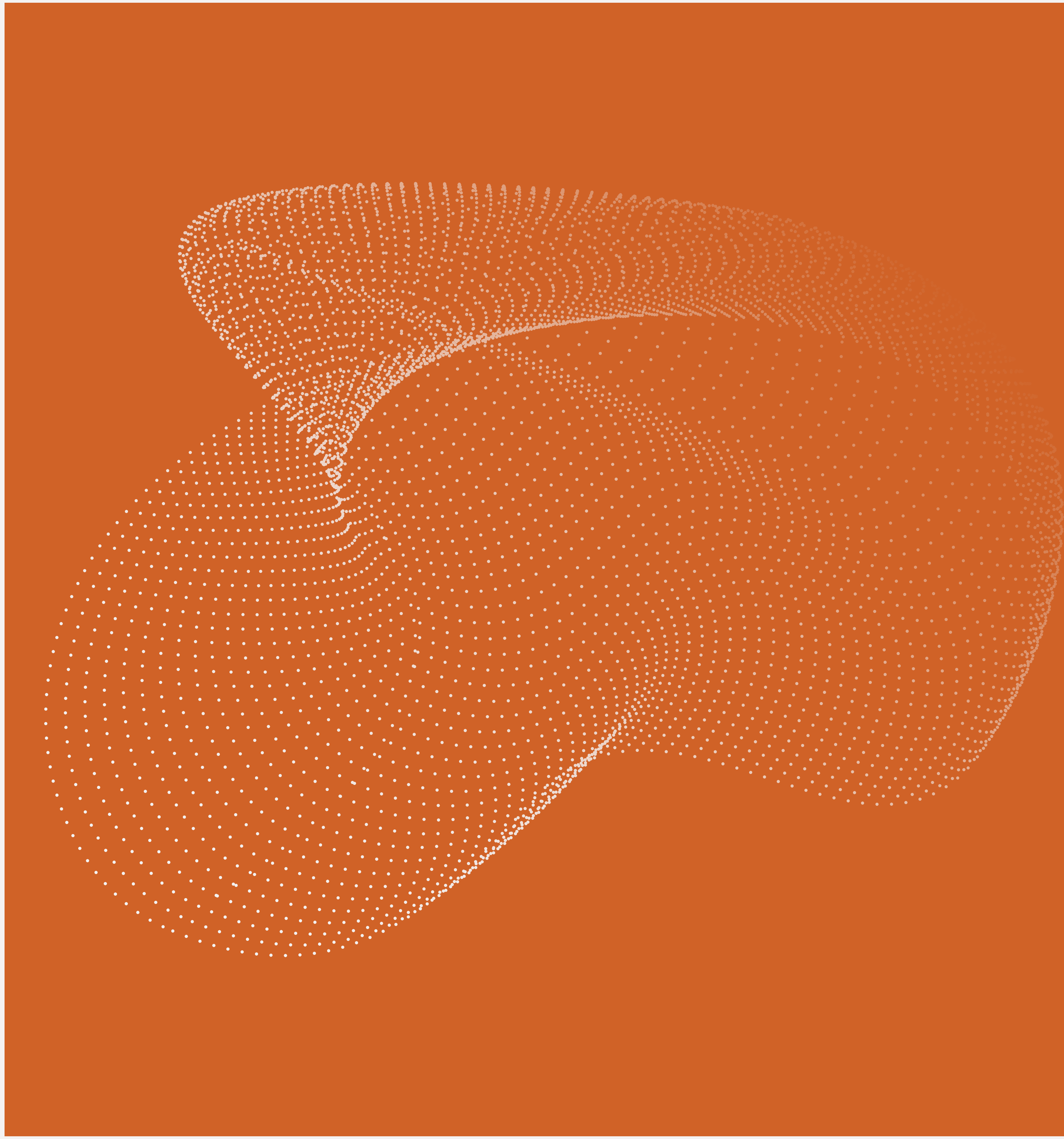
TPU Runtime Prediction Using GNNs

Valeria, Corrina, Amelia,
Eduardo & Pablo

Agenda

01 - Introduction
02 - Data
03 - Model Architecture
04 - Model Evaluation
05 - Results
07 - Conclusion



Introduction

Challenges for DLM Training Today

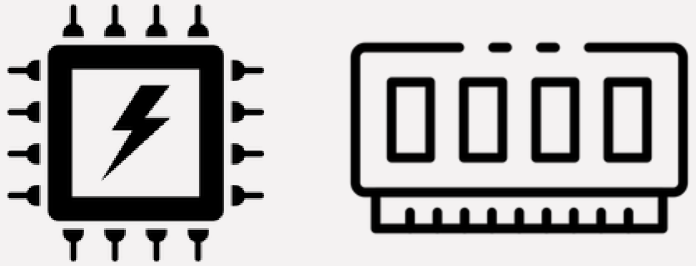
Hardware

Runtime

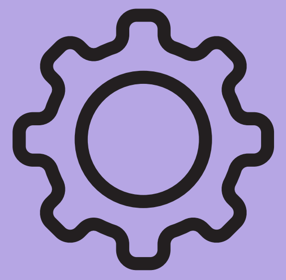
Challenges for DLM Training Today

Hardware

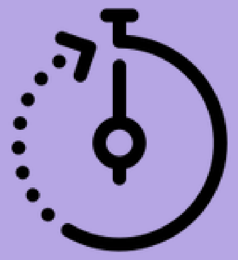
Runtime



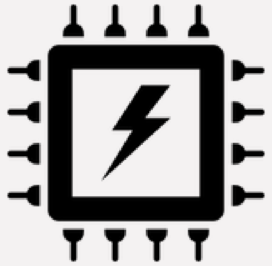
Challenges for DLM Training Today

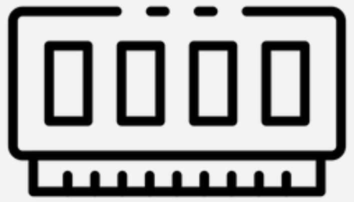


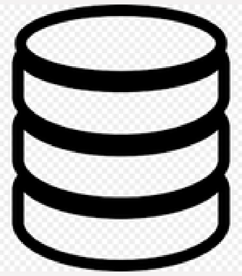
Hardware



Runtime







How can we effectively measure DLM runtime on complex hardware?

Kaggle Competition Details

Problem

Machine learning models experience slow performance.

Objective

Optimize model compilation by adjusting compiler configurations affecting model speed.

Methodology

Train DLM to predict runtimes using provided data (graphs).

Expected Outcome

Predict the runtime of test dataset graphs and its configurations.

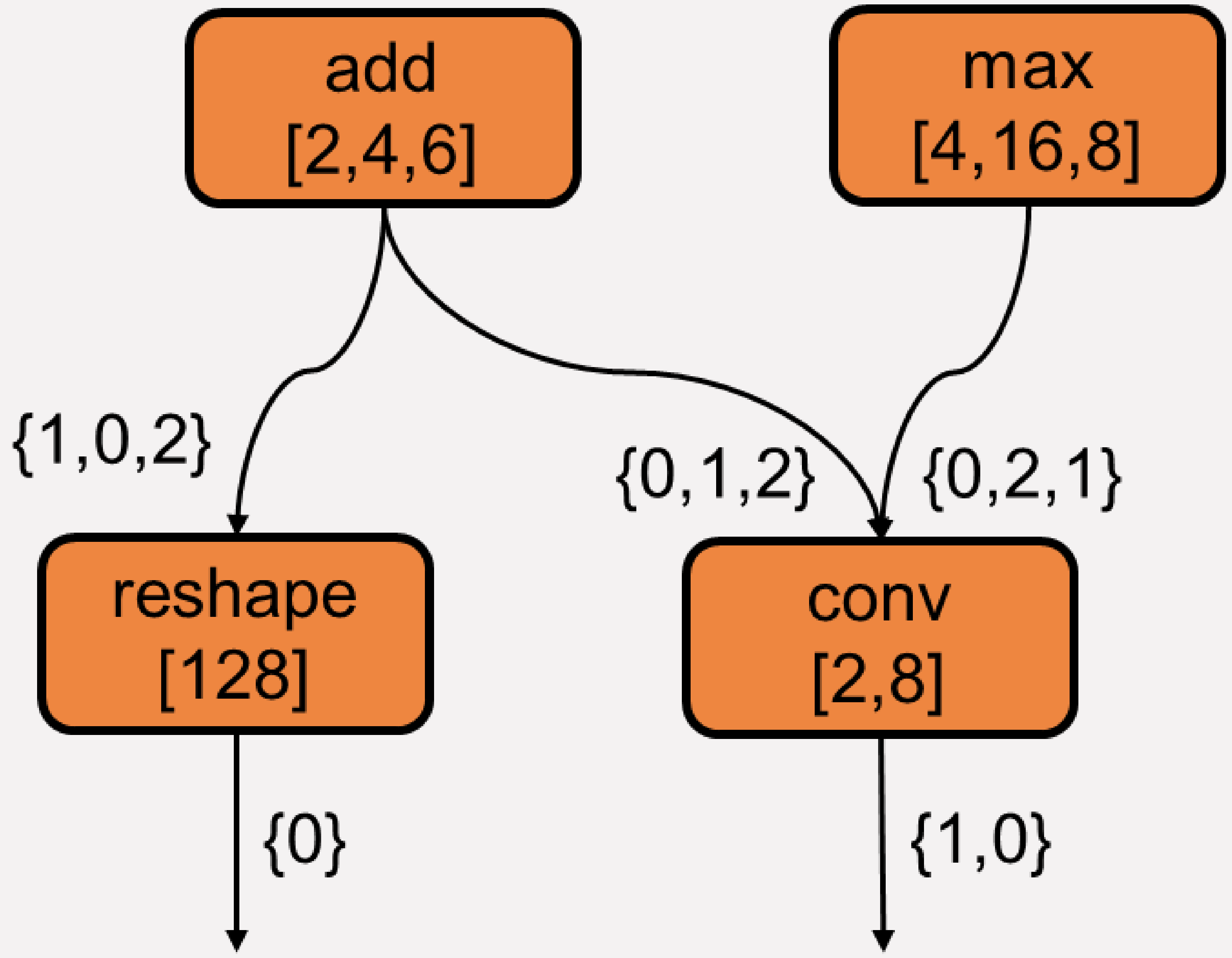
Leveraging **GNNs** for Runtime Prediction on TPUs

- **The data**: structured in a way that represents a graph with n nodes and m edges.
- **GNNs**: designed to work with graph-structured data.
- **Proven Success:** *A Learned Performance Model for Tensor Processing Units*

Data

Graph Representation of Deep Learning Models

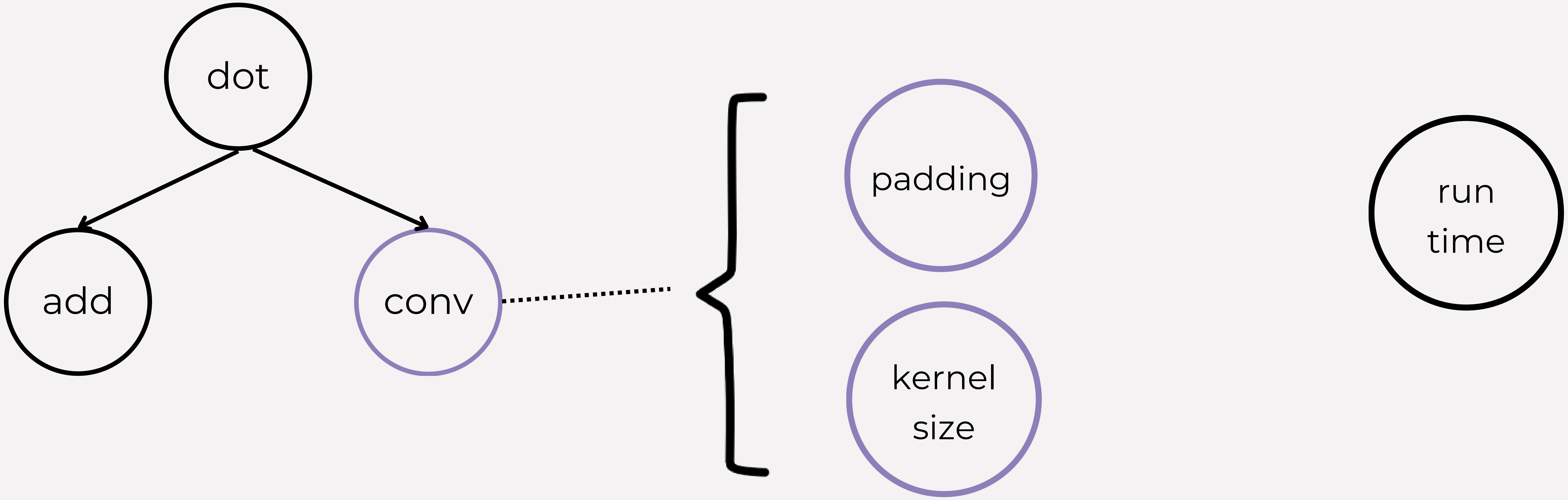
- **Nodes:**
  - Tensor operations
    - **Features:** Configurations for the *operation*
- **Edges:**
  - Tensors flowing through the nodes
    - **Features:** Configurations for *physical storage*



Compiler Configurations for an AlexNet RNN

(10 layers
256 hidden units
4 attention heads)

Operation Nodes (19,541) **Config Nodes** (1,778) **Context Nodes** (1,778)



Edge Sets hold directed adjacency as list of pairs of indices:

- Actual computation graph
    - Random subset (used for drop-out)
- Correspondence between configurations and operations

Dataset Description

Dataset Files (.npz) :

- Each file stores a graph (> 20 MB)

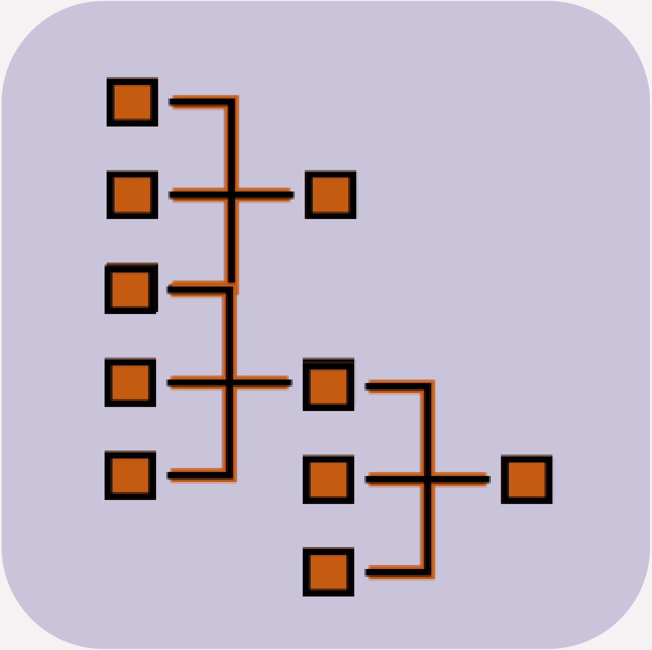
Each graph is specific to:

1) Model Type:

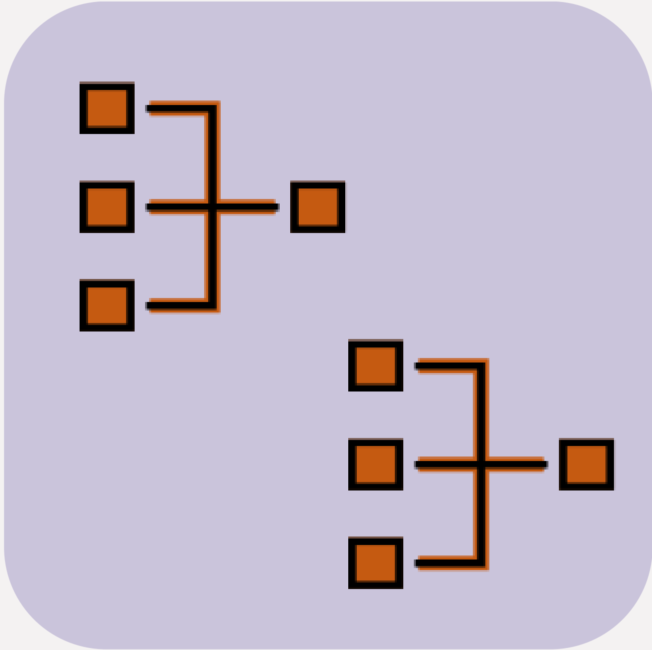
- XLA (AlexNet, Inception, ResNet, etc.)
- NLP (BERT, Electra, ALBERT, etc.)

2) Compiler Optimizations

- "Tile": Dividing tensors into smaller parts
- "Layout": Memory arrangement of tensors
  - **3) Search Strategy**
    - Default
    - Random



Layout

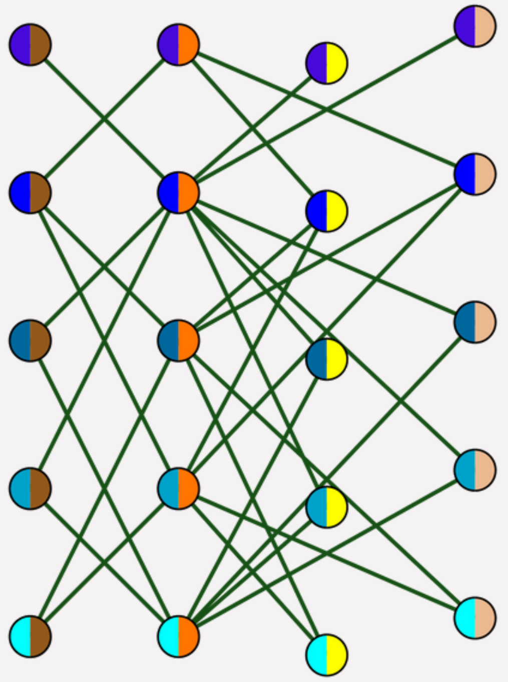


Tile

Dataset Sizes for Each Model

Layout Datasets

Model Type	Search strategy	Test Graphs	Train Graphs	Val Graphs
XLA	Default	8	61	7
	Random	8	69	7
NLP	Default	17	179	20
	Random	17	77	20



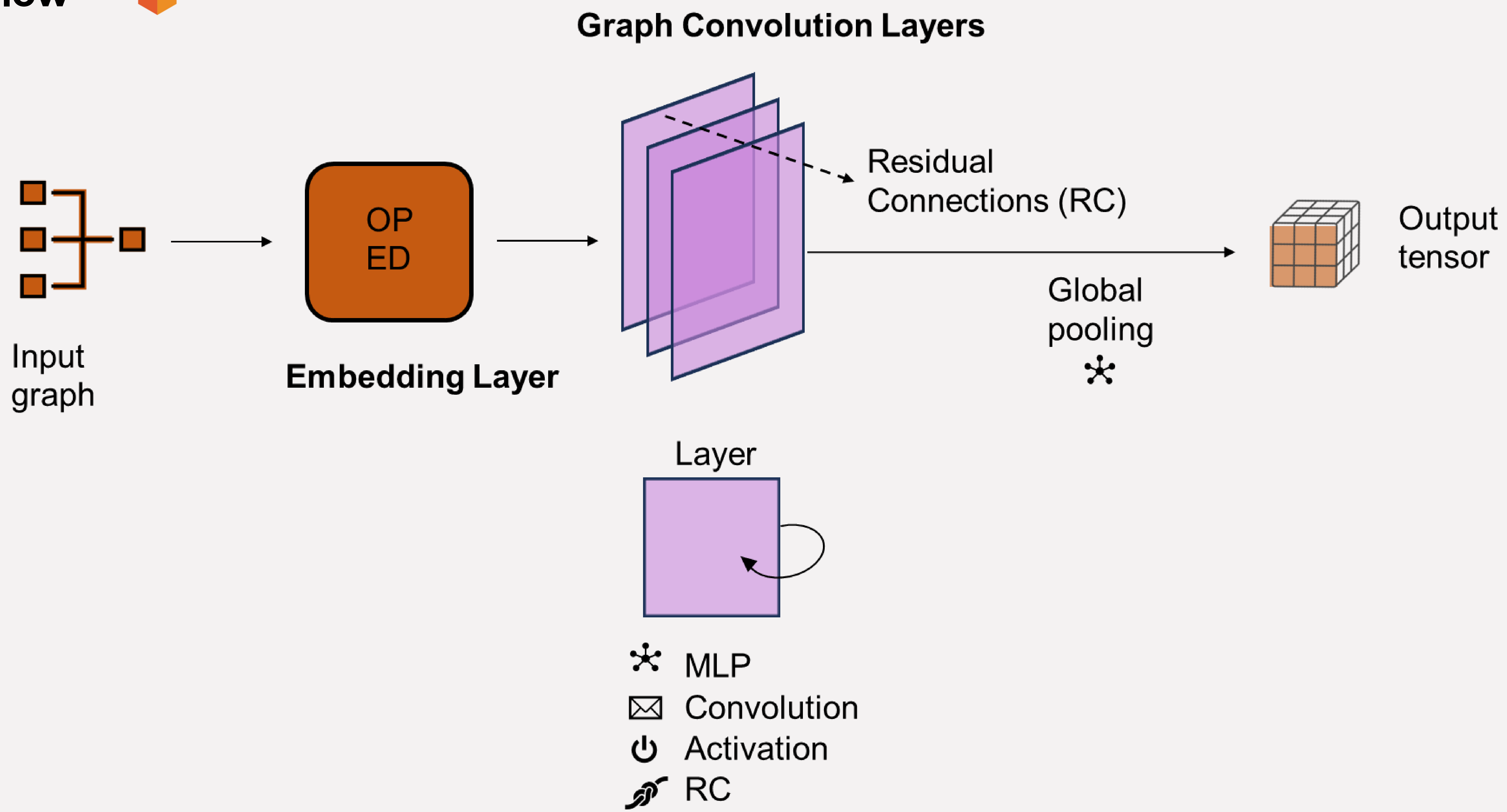
Tiles Datasets

Model Type	Test Graphs	Train Graphs	Val Graphs
XLA	832	5716	676

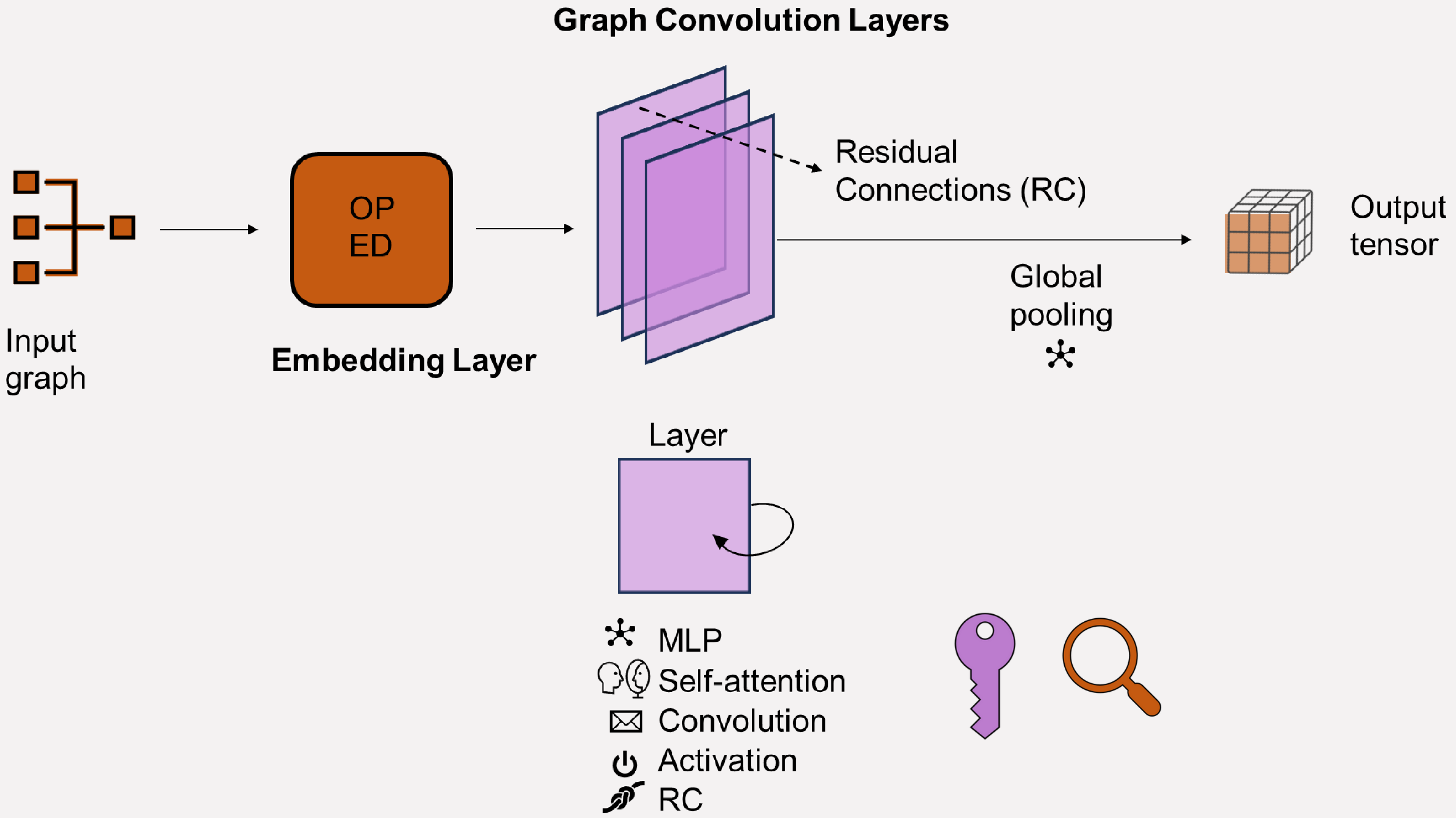
Model Architecture

GNN

TensorFlow



GNN + Self-Attention



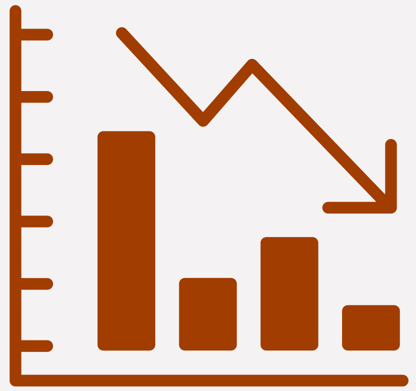
Training Process

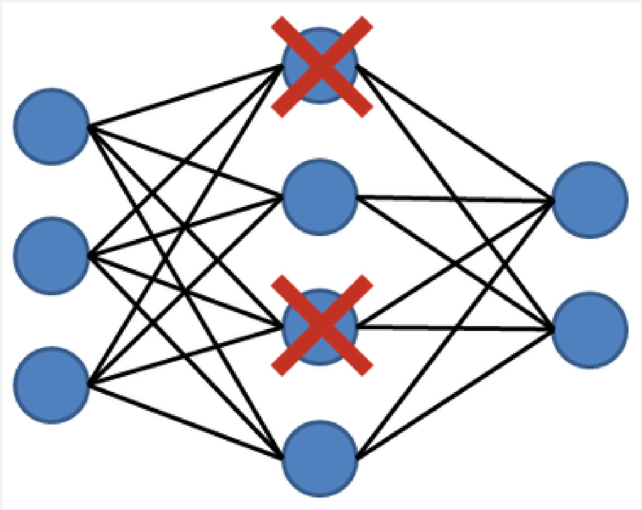
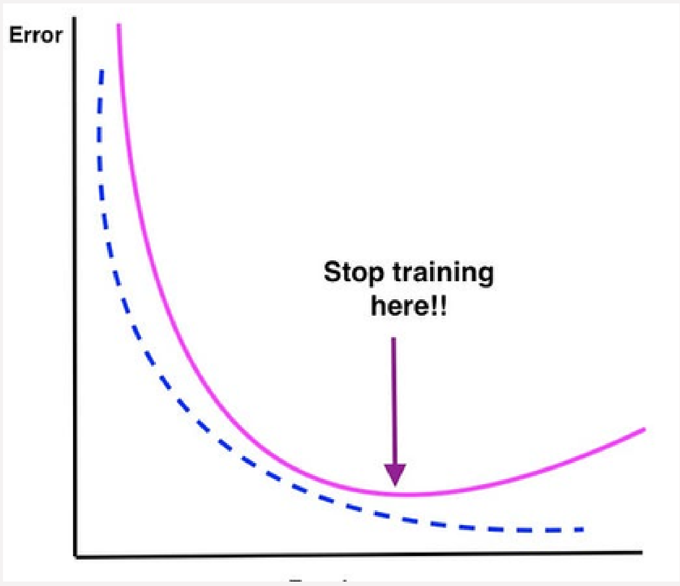
Loss Function and Optimizer

- Listwise Maximum Likelihood Estimation (ListMLE) loss function
- Adam optimizer

Overfitting Prevention Techniques

- Segment dropout: Random edge sampling during training
- Early stopping based on validation metrics



Model Evaluation

Experimentation

Project Setup

- Google Collab:
  CPU/TPU: 107.7 GB
  RAM: 12.7 GB

- Kaggle:
  GPU: 15.9 GB
  CPU: 29 GB,
  RAM: 29 GB

Hyperparameter tuning

- Max nodes
- Max configurations
- Batch size
- Epochs
- Early stopping

Evaluation Performance

Competition evaluation

Test set

Avg metrics :

- Tiles:

$$2 - \frac{\min_{i \in K} y_i}{\min_{i \in A} y_i}$$

- Layout:

  Kendal Tau Correlation

Team evaluation

Training- validation set

- OPA:

  Configuration rankings

- ListMLE loss:

  Output graph

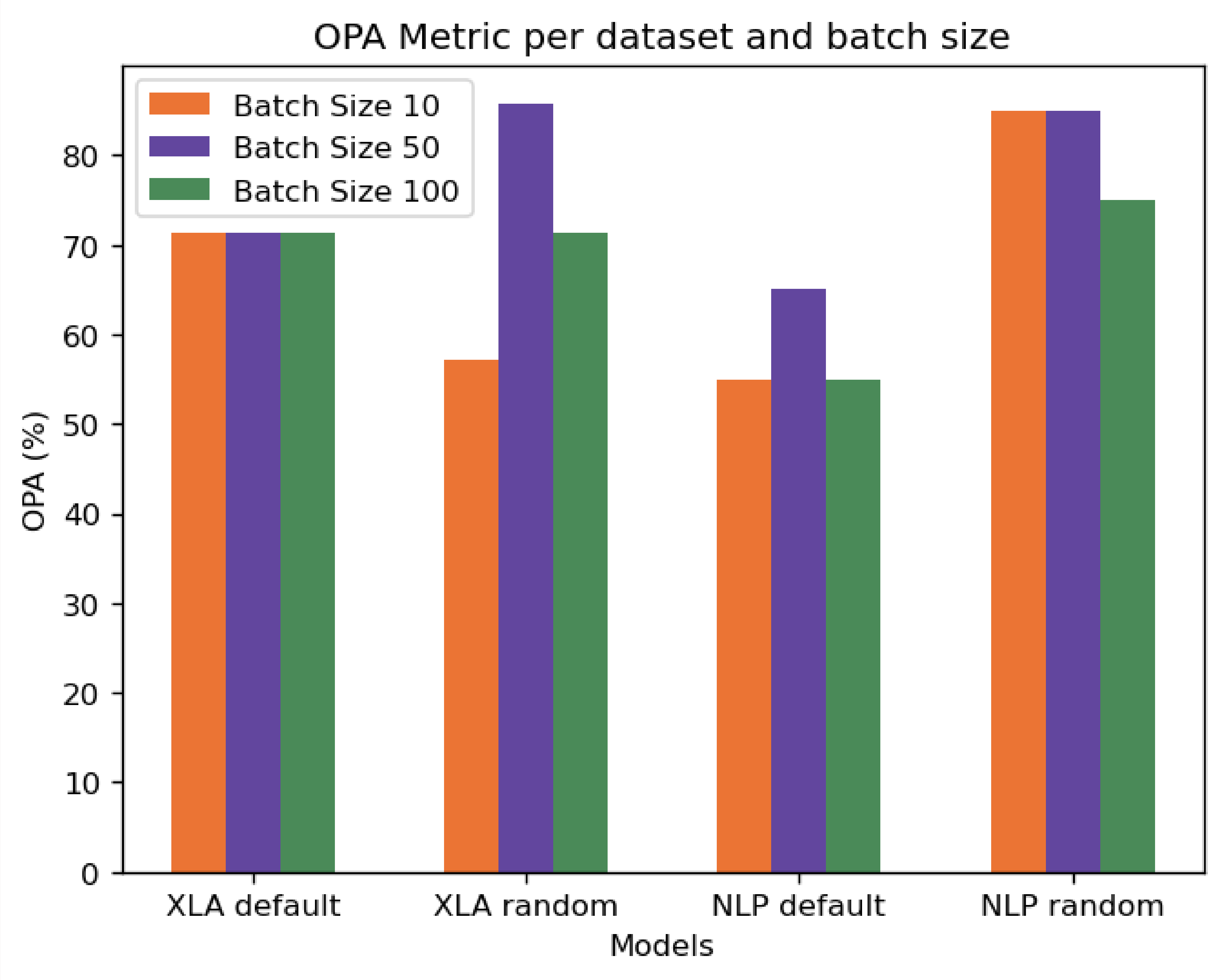
Results

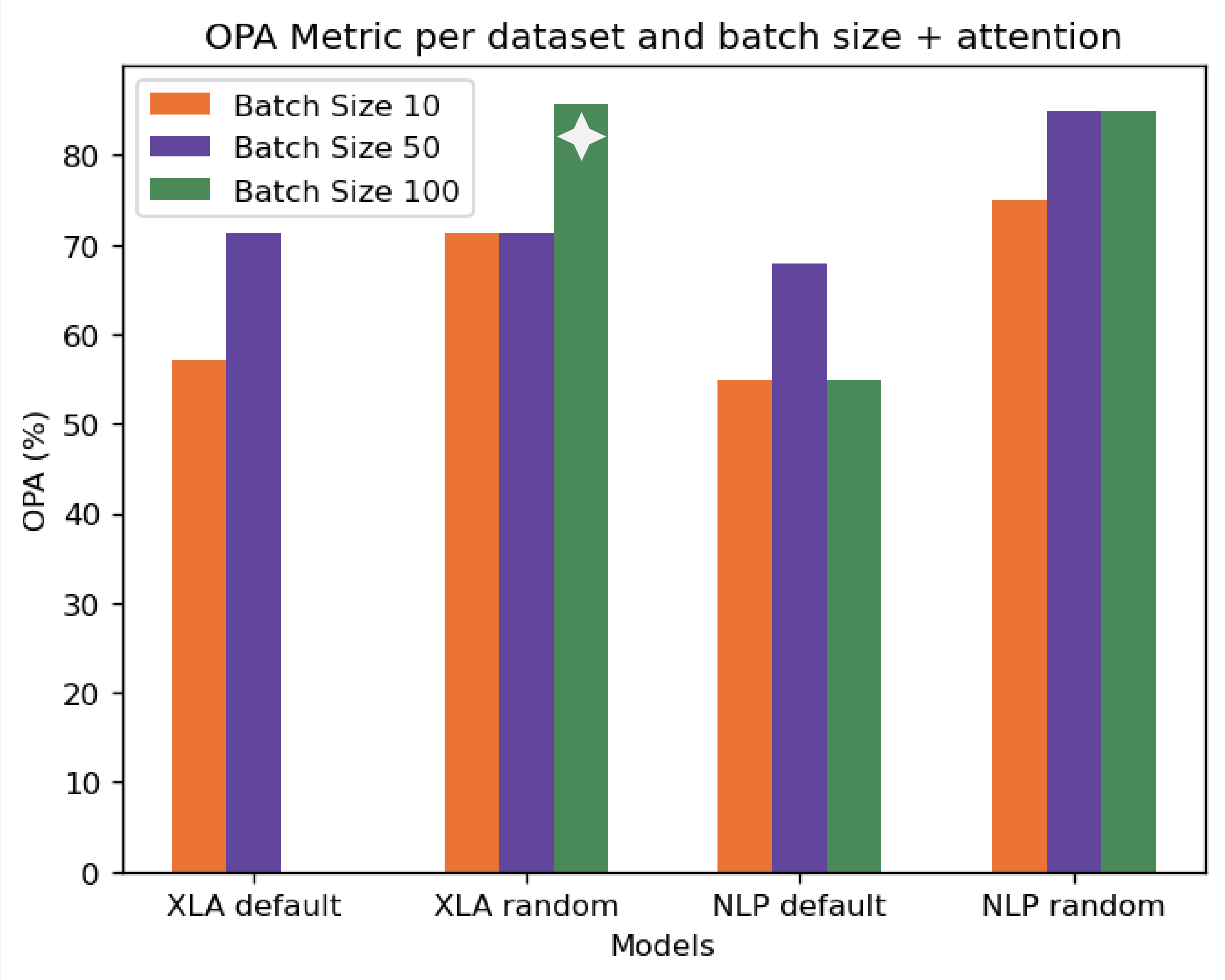
Hyperparameter Tuning

Hyperparameters	Crashed	Passed
Max nodes	1000	100
Max configurations	500	20
Batch size	200	100, 50, 10
Epochs	-	1 to 6
Early stopping	-	2 to 4

Training: 20 - 40 minutes
Testing: 2 - 4 hours

Models' Performances over Datasets





Best OPA: 85%

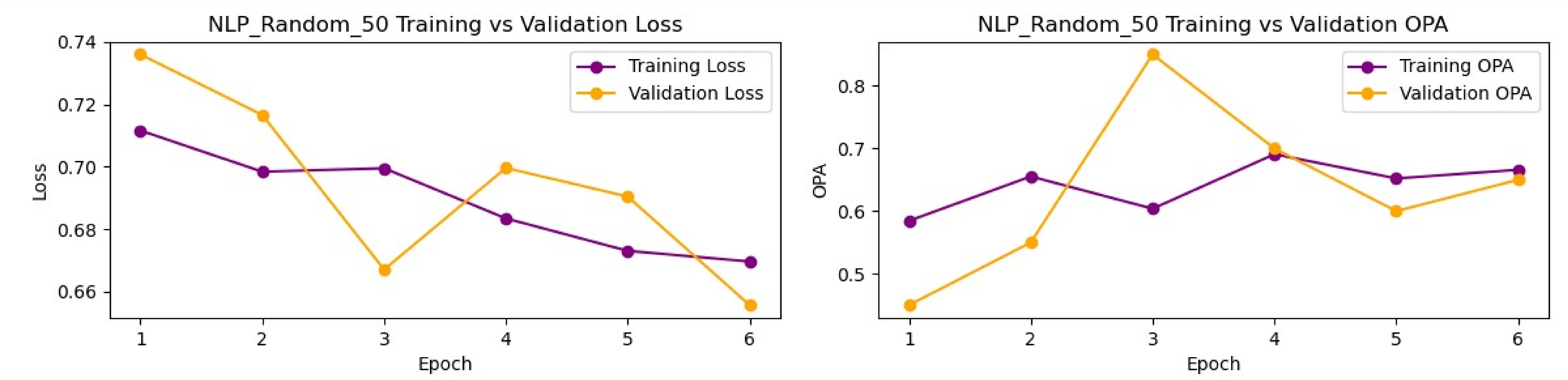
Best avg OPA: **76.79%**

Best OPA: **85.71%**

Best avg OPA: 73.97%

Models' Loss Behavior

Better OPA (~20%) and loss with testing than training and validation (underfit)



Loss and OPA of NLP Random with 50 batches using attention

Competition Results

hyperparameters	private avg metric
1 epoch 10 batches	**23.18%**
6 epochs 10 batches	23.06%
6 epochs 50 batches	22.17%
6 epochs 100 batches	17.58%

239 / 616

Conclusions

- **Understanding Data**
  - Understanding the data was a big part of the project
  - We had a time constraint; learned about graphs
- **Hardware Limitations**
  - Lack of RAM limited the our ability to experiment; free cloud services were also lacking resources
- **Best Methods**
  - Adding attention to the models yielded the best results, and the competition winner had the same approach
- **Kaggle Competition Style**
  - Submitting results and comparing performance instead of deep exploration and writing a paper
- **Future Work**
  - Obtain more resources and experiment with the parameters we were unable to use

Thanks

References:

Article: Kaufman, S., Phothilimthana, P., Zhou, Y., Mendis, C., Roy, S., Sabne, A., & Burrows, M. (2021). A learned performance model for tensor processing units. Proceedings of Machine Learning and Systems, 3, 387-400.

Competition repository: Kaggle Starter Code: https://github.com/google-research-datasets/tpu_graphs

Appendix

Top-Performing Solutions:

- Architecture:
    - Attention
    - SAGEconv (GNN)
- Data Reduction:
    - Reducing graphs structures by learning its statistics
    - Dijkstra’s Algorithm to reduce nodes structures
    - Extract values from graphs and do MLP over GNN