Section 1

AI-Based Traffic Flow Prediction: HPC Final Project Pitch

5-Slide Presentation Outline

Section 2

Slide 1: Problem & Impact

Urban Traffic Congestion: A €100B Problem

The Challenge: - Traffic congestion costs European cities **€100B+ annually** - Average commuter loses **40+ hours/year** to traffic - Transportation accounts for **25% of CO2 emissions** - Urban populations growing **1.5% annually**

Why Current Solutions Fail: - Traditional physics models: Too slow for real-time - Simple ML models: Ignore road network structure - Single-GPU training: Can't handle city-scale data

Our Goal: Build a **real-time traffic prediction digital twin** using distributed deep learning on HPC systems

```
Current: Hours to predict    →    Target: Seconds to pr
Small networks (50 sensors)  →    City-scale (2000+ sen
```

Section 3

Slide 2: Approach & Prototype

Diffusion Convolutional Recurrent Neural Network (DCRNN)

Architecture:

```
Input: 1hr          Diffusion          Output:
of traffic         Conv + GRU        5-min
data                (2 layers)          prediction

  207 sensors         Road network
  12 timesteps        as graph
```

Distributed Training with PyTorch DDP: - NCCL backend for GPU-to-GPU communication - DistributedSampler for data sharding - Mixed-precision (BF16) for speed - Apptainer container for reproducibility

Dataset: METR-LA - 207 highway sensors in Los Angeles - 4 months of

Section 4

Slide 3: Scaling & Profiling Results

Performance at Scale

Strong Scaling (Fixed Dataset):

Nodes	Time (s)	Speedup	Efficiency
1	120.0	1.00×	100%
2	62.4	1.92×	96%
4	33.1	3.63×	91%
8	19.4	6.19×	82%

Key Metrics: | Metric | Value | |——–|——-| | Throughput (8 nodes) | 1,600 samples/sec | | GPU Utilization | 84% average | | MAE | 3.52 mph | | RMSE | 5.18 mph |

Bottleneck Analysis:

```
72% Compute
18% Data Load
 7% Comm
```

Section 5

Slide 4: EuroHPC Target & Resource Ask

Scaling to LUMI-G

Why LUMI-G? - 2,978 GPU nodes (11,912 AMD MI250X GPUs) - Slingshot-11 interconnect (200 Gbps) - Ideal for distributed GNN training

Scaling Goal:

```
Current:  8 nodes  → 1,600 samples/sec
Target:  64 nodes  → 10,000+ samples/sec (enabling real-time)
```

Resource Request: 512 GPU-node-hours

Experiment	Nodes	Hours	GPU-node-hours
Strong scaling	1-64	2	160
Weak scaling	1-64	2	160
Optimization	8-16	4	80
Science runs	64	4	80
Buffer	-	-	32
Total			**512**

Section 6

Slide 5: Risks, Milestones & Support Needed

Project Timeline (6 Months)

`M1-M2`	`Porting (HIP/ROCm)`
`M2-M3`	`Optimization`
`M3-M4`	`Scaling experiments`
`M4-M5`	`Science runs`
`M5-M6`	`Documentation`

Key Milestones: - **M2**: Validated code on LUMI-G (single node) - **M4**: 64-node scaling with >75% efficiency - **M6**: Technical report + open-source release

Risks & Mitigations:

Risk	Probability	Mitigation
Porting delays	Medium	Start with hipify, Leonardo backup
Scaling issues	Low	Gradient compression, async comm
Queue delays	Medium	Flexible scheduling

Summary

What We Built: - Distributed DCRNN for traffic prediction - 82% efficiency on 8 GPU nodes - Containerized, reproducible framework

What We'll Deliver: - Scaling benchmarks on EuroHPC - Open-source code and models - Publication-ready results

The Impact: - Real-time traffic digital twins - Reduced congestion and emissions - Template for GNN training at scale

Thank you! Questions?

Backup Slides

B1: DCRNN Architecture Details

```
class DCRNN(nn.Module):
    def __init__(self, input_dim=1, hidden_dim=64,
                 num_nodes=207, num_layers=2):
        # Diffusion conv captures spatial dependencies
        # GRU captures temporal dynamics
        self.cells = nn.ModuleList([
            DCGRUCell(input_dim, hidden_dim, num_nodes),
            DCGRUCell(hidden_dim, hidden_dim, num_nodes)
        ])
        self.output_layer = nn.Linear(hidden_dim, 1)
```

B2: Dataset Details

Dataset	Sensors	Timesteps	Size
METR-LA	207	34,272	34 MB