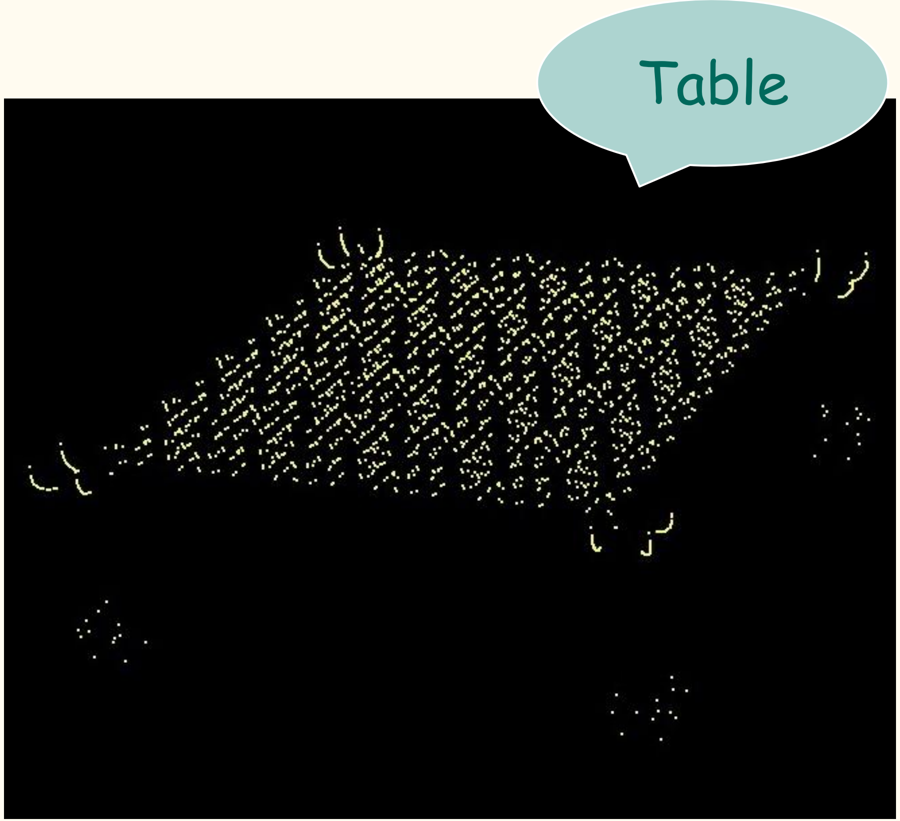
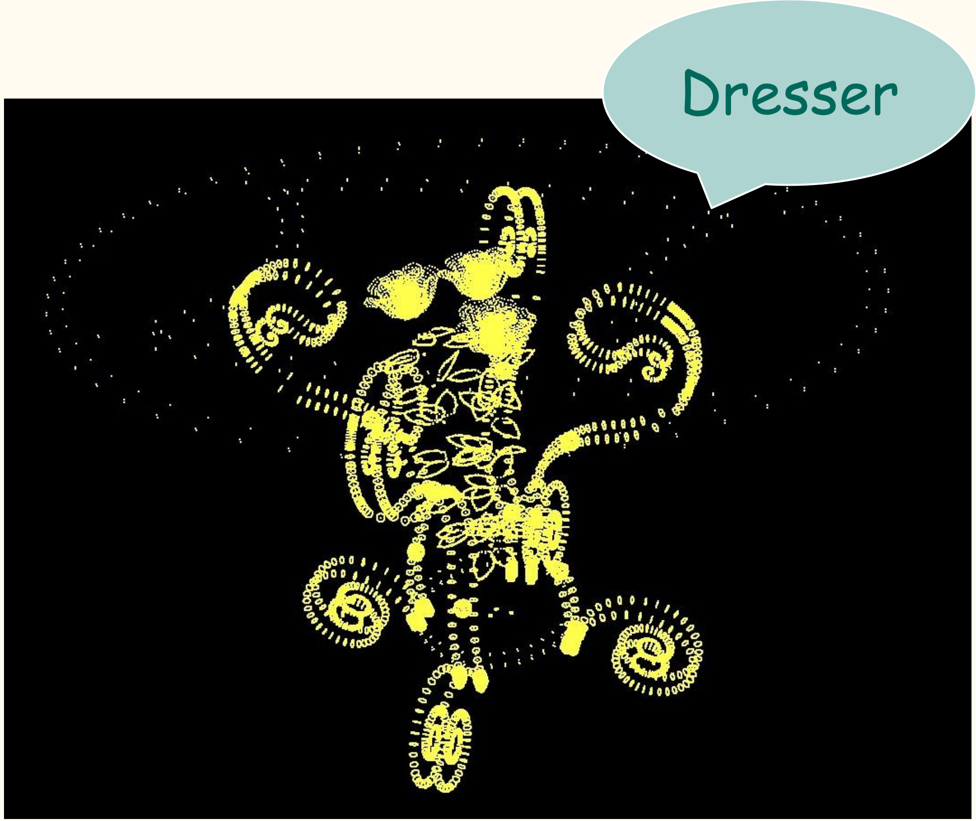
Point Cloud Classification Milestone 3

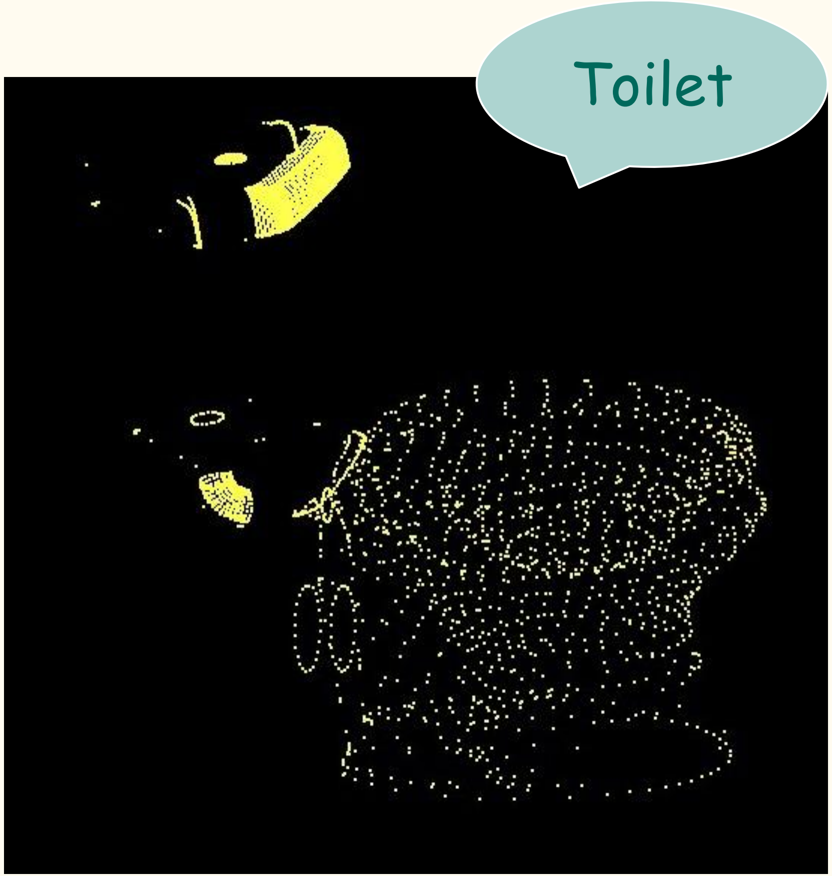
Disha Jindal, Kushagra Goel and Saket Karve

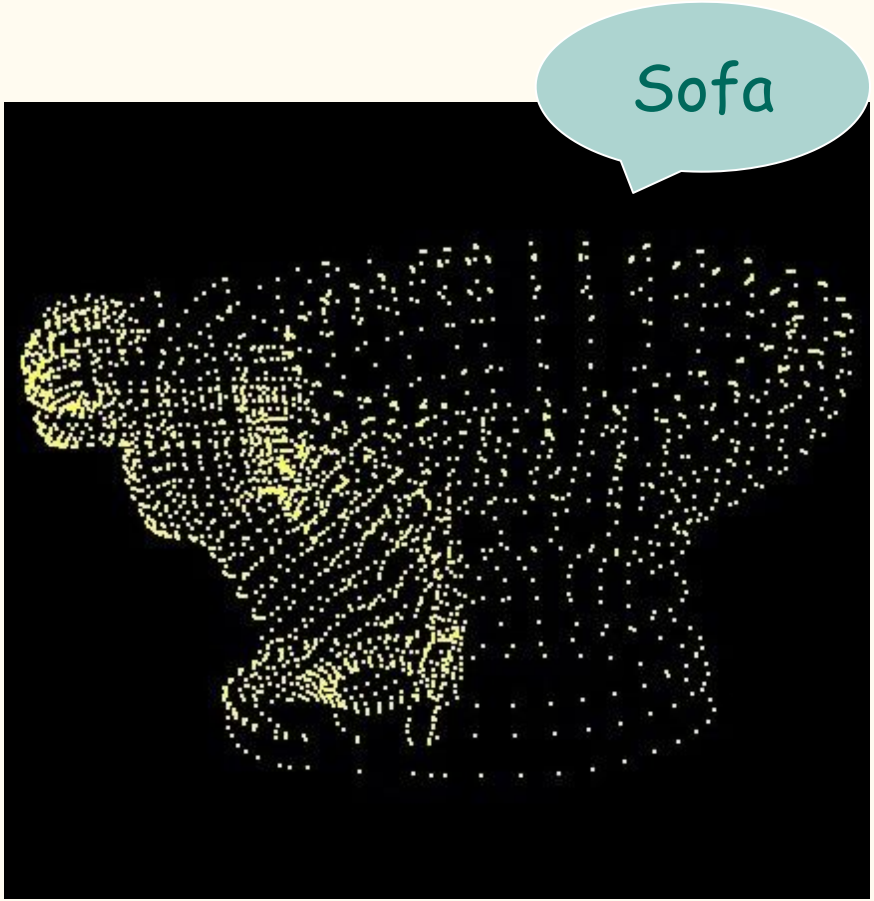
PREDICTIONS



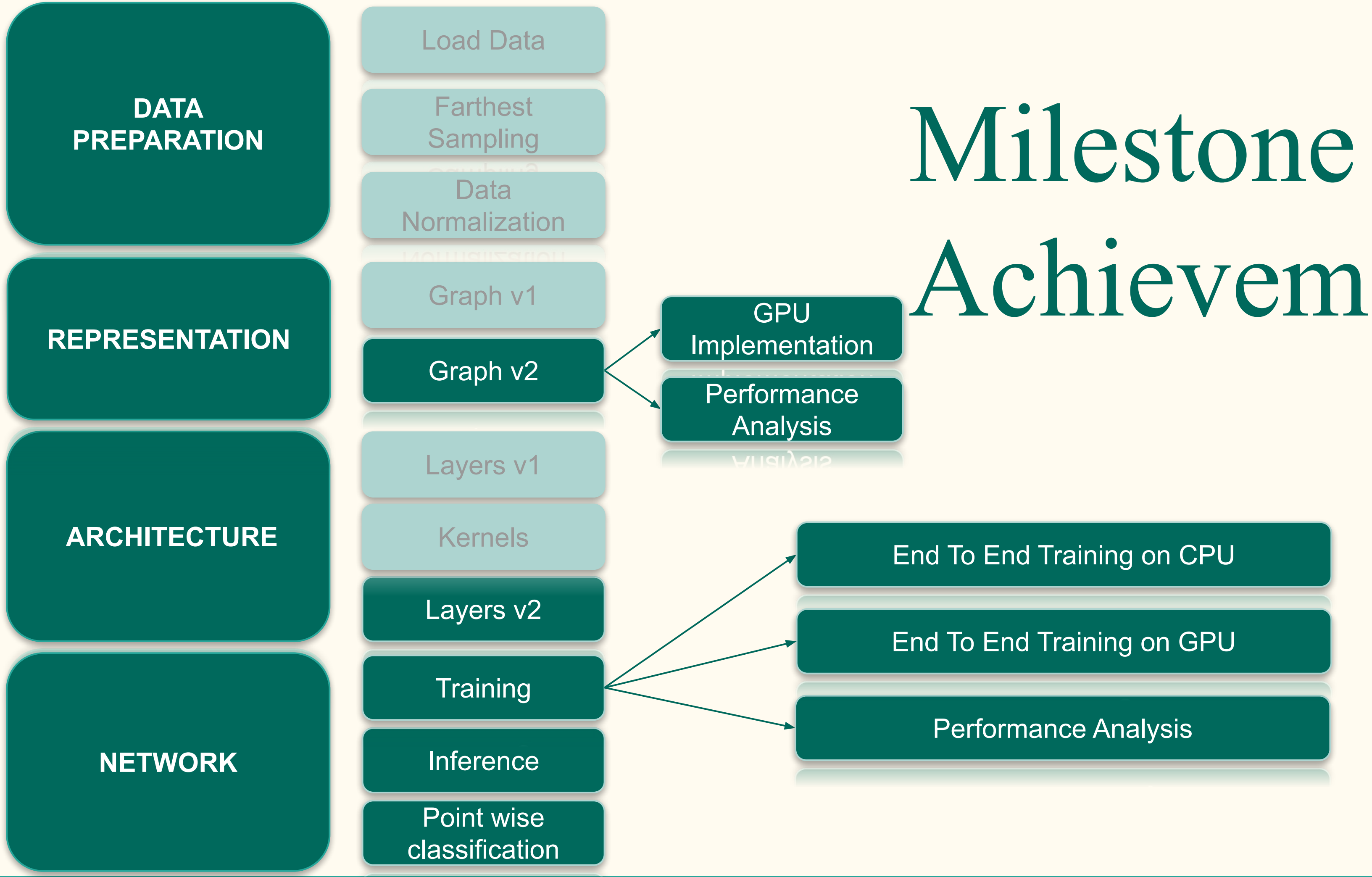


PREDICTIONS





Milestone 3 Achievements



DATA PREPARATION

Load Data

Farthest Sampling

Data Normalization

REPRESENTATION

Graph v1

Graph v2

GPU Implementation

Performance Analysis

ARCHITECTURE

Layers v1

Kernels

Layers v2

NETWORK

Training

End To End Training on CPU

End To End Training on GPU

Performance Analysis

Inference

Point wise classification

DATA PREPARATION

- Load Data
- Farthest Sampling
- Data Normalization

REPRESENTATION

- Graph v1
- **Graph v2**

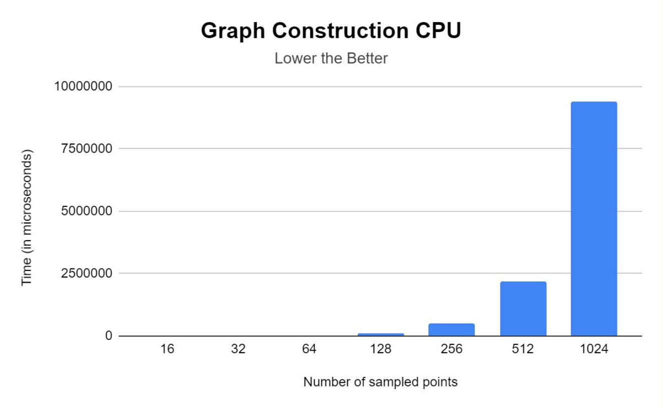
ARCHITECTURE

- Layers v1
- Kernels
- Layers v2

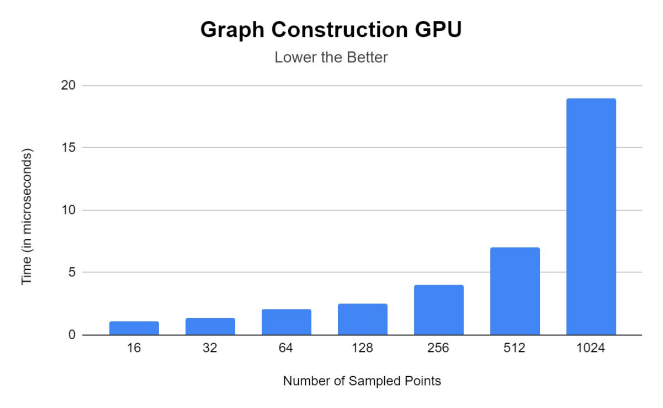
NETWORK

- Training
- Inference
- Point wise classification

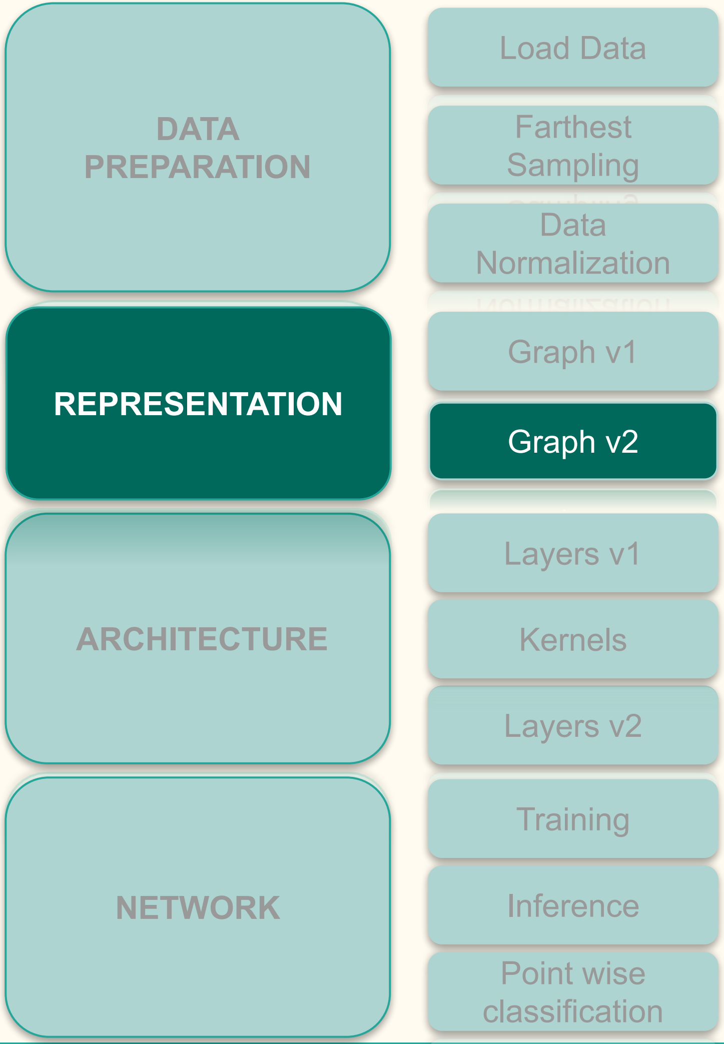
GRAPH CONSTRUCTION


Graph Construction CPU

Lower the Better


Graph Construction GPU

Lower the Better



DATA PREPARATION

- Load Data
- Farthest Sampling
- Data Normalization

REPRESENTATION

- Graph v1
- **Graph v2**

ARCHITECTURE

- Layers v1
- Kernels
- Layers v2

NETWORK

- Training
- Inference
- Point wise classification

GRAPH CONSTRUCTION

N	CPU Time(µs)	GPU Time(µs)	Factor
16	1920	1.107	~1734
32	6836	1.338	~5109
64	25968	2.058	~12618
128	111756	2.5	~44702
256	489553	4	~122388
512	2169774	7	~309967
1024	9409076	19	~495214

DATA PREPARATION

- Load Data
- Farthest Sampling
- Data Normalization

REPRESENTATION

- Graph v1
- Graph v2

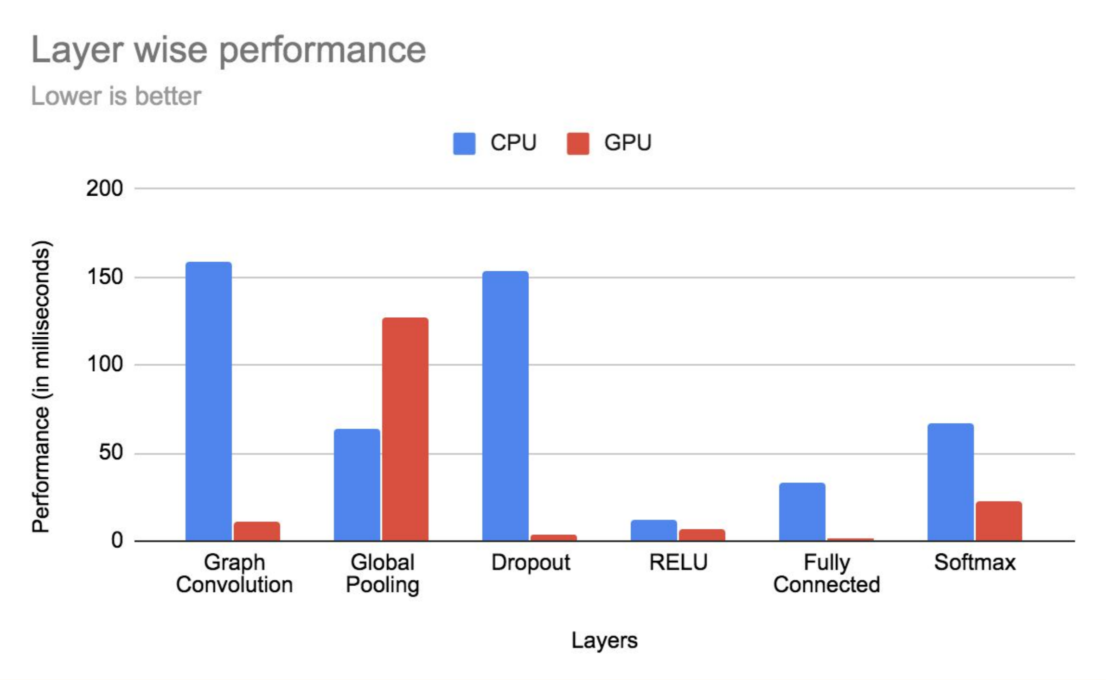
ARCHITECTURE

- Layers v1
- Kernels
- **Layers v2**

NETWORK

- Training
- Inference
- Point wise classification

Layer wise Performance Comparison



Layer wise performance

Lower is better

DATA PREPARATION

- Load Data
- Farthest Sampling
- Data Normalization

REPRESENTATION

- Graph v1
- Graph v2

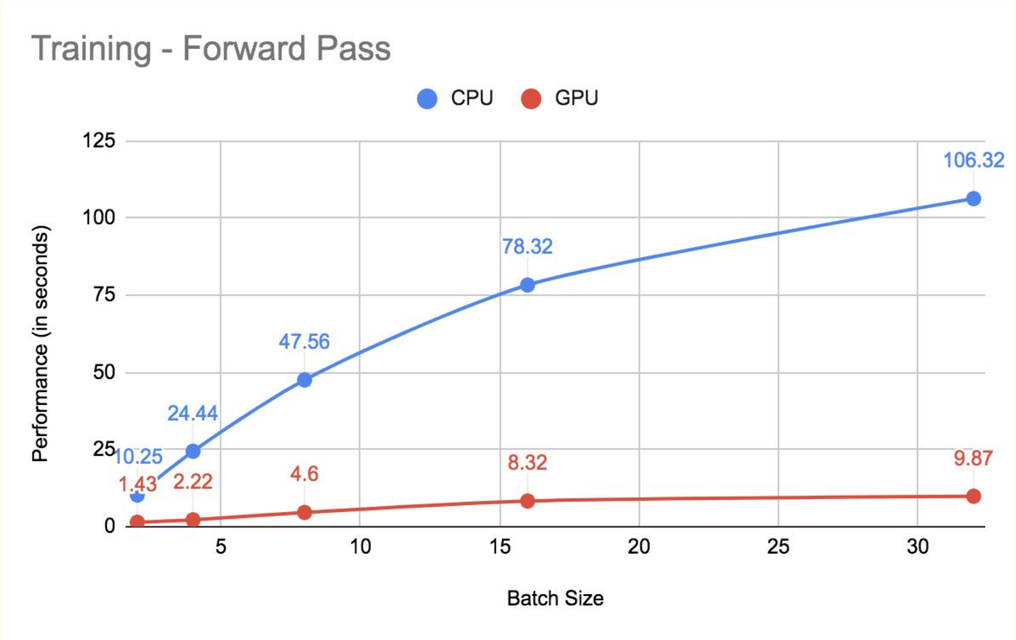
ARCHITECTURE

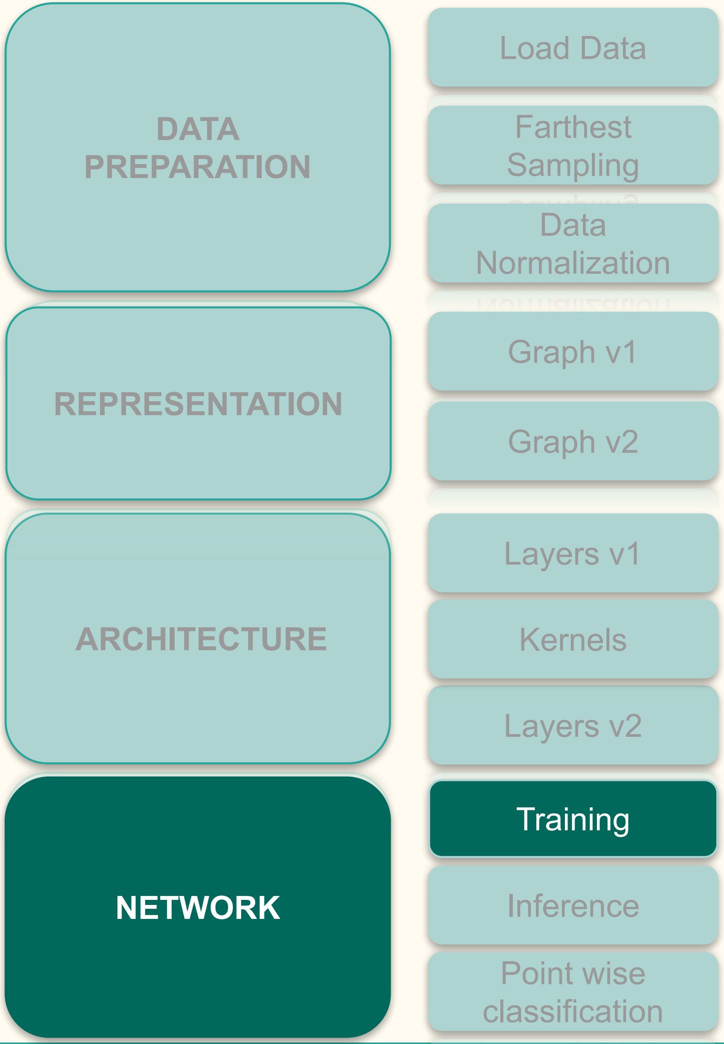
- Layers v1
- Kernels
- Layers v2

NETWORK

- Training
- Inference
- Point wise classification

Training Time Per Epoch





DATA PREPARATION

- Load Data
- Farthest Sampling
- Data Normalization

REPRESENTATION

- Graph v1
- Graph v2

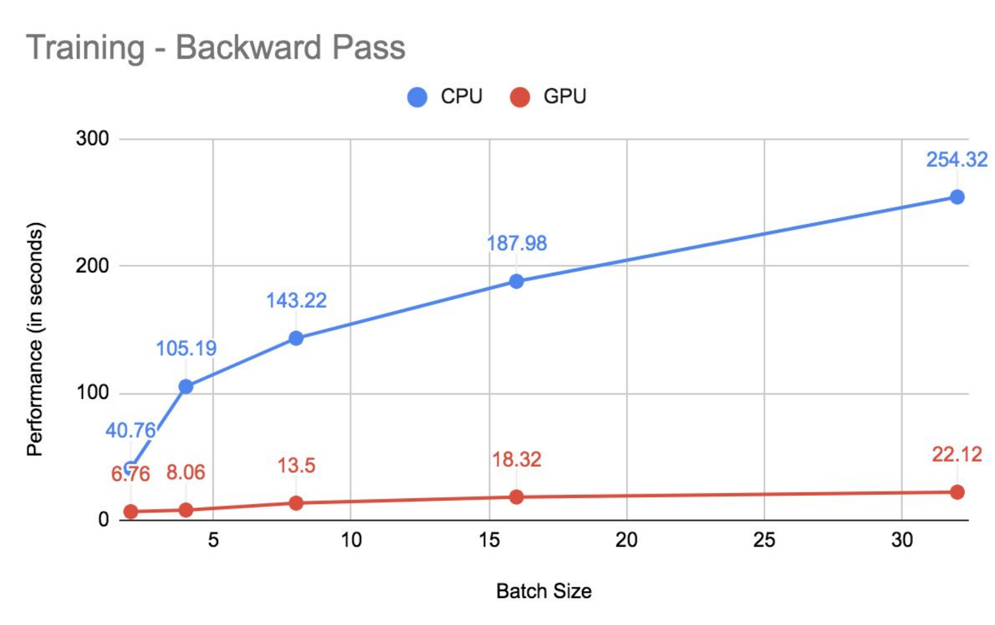
ARCHITECTURE

- Layers v1
- Kernels
- Layers v2

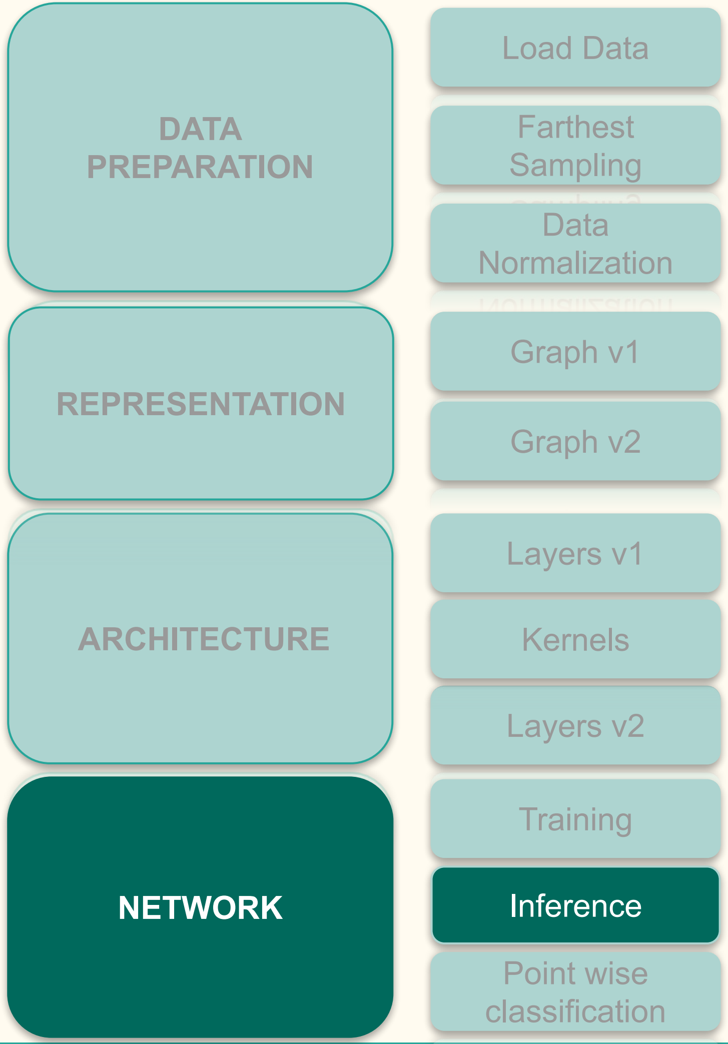
NETWORK

- **Training**
- Inference
- Point wise classification

Training Time Per Epoch



Training - Backward Pass



DATA PREPARATION

Load Data

Farthest Sampling

Data Normalization

REPRESENTATION

Graph v1

Graph v2

ARCHITECTURE

Layers v1

Kernels

Layers v2

NETWORK

Training

Inference

Point wise classification

Time required for getting predictions for a single point cloud

CPU	GPU	Factor
982 ms	111 ms	~9

DATA PREPARATION

- Load Data
- Farthest Sampling
- Data Normalization

REPRESENTATION

- Graph v1
- Graph v2

ARCHITECTURE

- Layers v1
- Kernels
- Layers v2

NETWORK

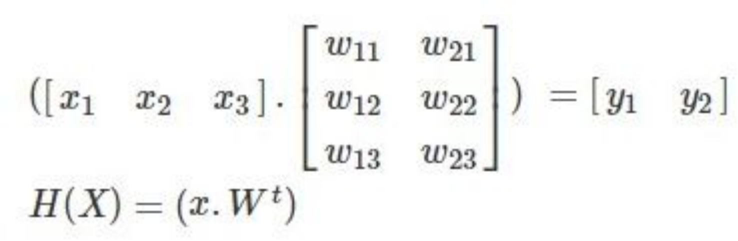
- Training
- Inference
- **Point wise classification**

Time required for getting predictions for a single point cloud

CPU	GPU	Factor
982 ms	111 ms	~9

Fully Connected Layer - Improvement

- Added bias term to the Fully Connected Layer.

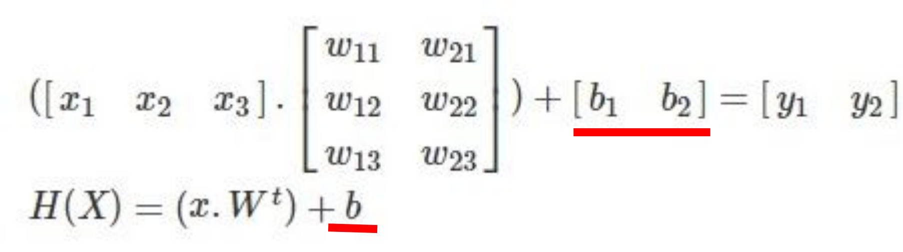


$$\left(\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} . \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix}\right) = \begin{bmatrix} y_1 & y_2 \end{bmatrix}$$

$$H(X) = (x.W^t)$$

Fully Connected Layer - Improvement

- Added bias term to the Fully Connected Layer.



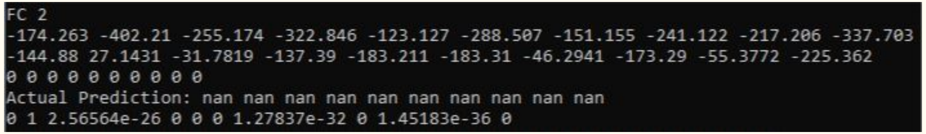
$$([x_1 \quad x_2 \quad x_3] . \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix}) + \underline{[b_1 \quad b_2]} = [y_1 \quad y_2]$$

$$H(X) = (x.W^t) + \underline{b}$$

L2 Regularization

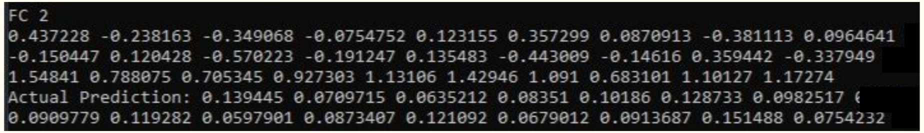
- Added option for L2 regularization.
- L2 regularization prevents parameters from going to extreme values.

Before L2 Regularization



```
FC 2
-174.263 -402.21 -255.174 -322.846 -123.127 -288.507 -151.155 -241.122 -217.206 -337.703
-144.88 27.1431 -31.7819 -137.39 -183.211 -183.31 -46.2941 -173.29 -55.3772 -225.362
0 0 0 0 0 0 0 0 0 0
Actual Prediction: nan nan nan nan nan nan nan nan nan nan
0 1 2.56564e-26 0 0 0 1.27837e-32 0 1.45183e-36 0
```

After L2 Regularization



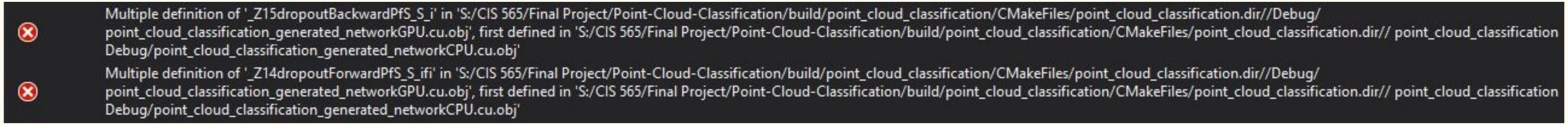
```
FC 2
0.437228 -0.238163 -0.349068 -0.0754752 0.123155 0.357299 0.0870913 -0.381113 0.0964641
-0.150447 0.120428 -0.570223 -0.191247 0.135483 -0.443009 -0.14616 0.359442 -0.337949
1.54841 0.788075 0.705345 0.927303 1.13106 1.42946 1.091 0.683101 1.10127 1.17274
Actual Prediction: 0.139445 0.0709715 0.0635212 0.08351 0.10186 0.128733 0.0982517
0.0909779 0.119282 0.0597901 0.0873407 0.121092 0.0679012 0.0913687 0.151488 0.0754232
```

Challenges

Bug Squash and Overcoming Shortfalls

- Precision Issue
  - Prescaled values by 1000 before operation and then rescaled back to original value.
  - Better than using double, which made our kernel very slow

- Multiple Instances error
  - Redesigned class declaration structure for all layers to fix copies of same function due multiple object files.

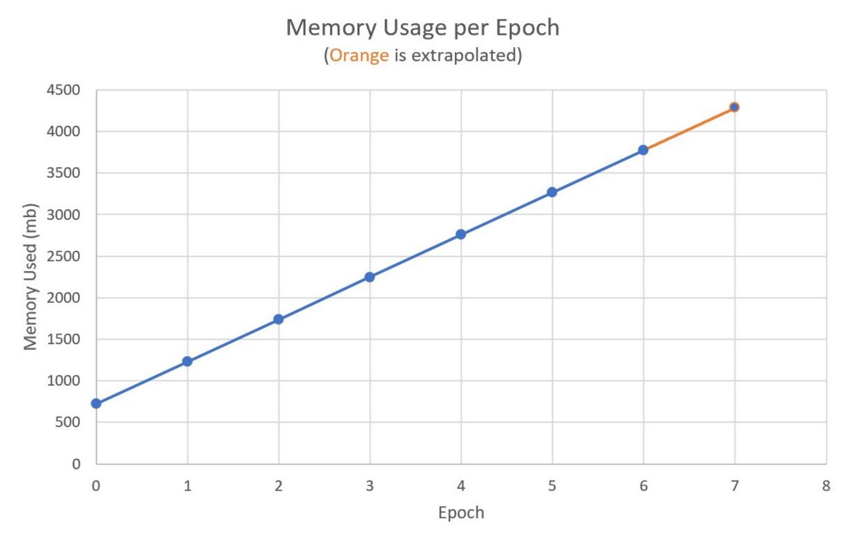


```
Multiple definition of '_Z15dropoutBackwardPfS_S_i' in 'S:/CIS 565/Final Project/Point-Cloud-Classification/build/point_cloud_classification/CMakeFiles/point_cloud_classification.dir//Debug/
point_cloud_classification_generated_networkGPU.cu.obj', first defined in 'S:/CIS 565/Final Project/Point-Cloud-Classification/build/point_cloud_classification/CMakeFiles/point_cloud_classification.dir// point_cloud_classification
Debug/point_cloud_classification_generated_networkCPU.cu.obj'
Multiple definition of '_Z14dropoutForwardPfS_S_ifi' in 'S:/CIS 565/Final Project/Point-Cloud-Classification/build/point_cloud_classification/CMakeFiles/point_cloud_classification.dir//Debug/
point_cloud_classification_generated_networkGPU.cu.obj', first defined in 'S:/CIS 565/Final Project/Point-Cloud-Classification/build/point_cloud_classification/CMakeFiles/point_cloud_classification.dir// point_cloud_classification
Debug/point_cloud_classification_generated_networkCPU.cu.obj'
```

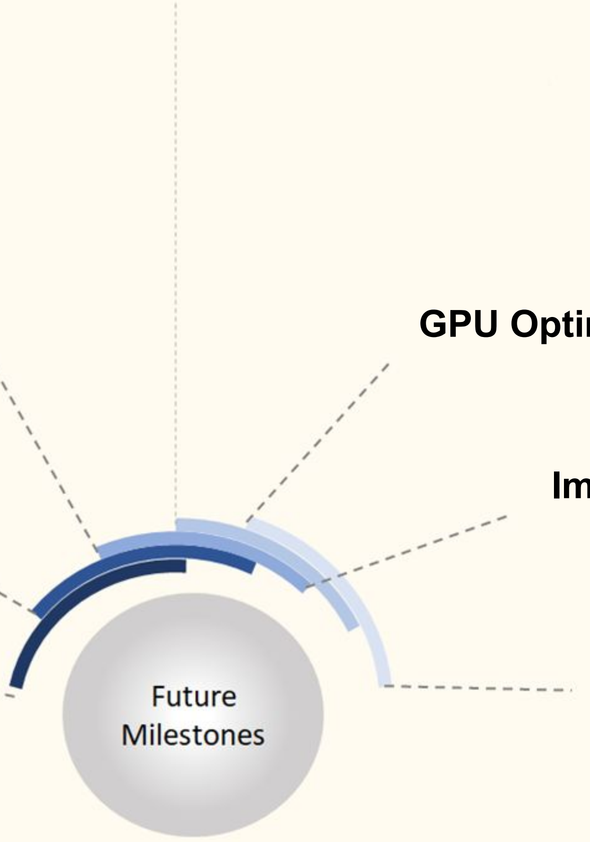
Memory Optimization

Memory Optimizations

- Worked on optimizing memory used in forward and backward pass.
- Working on optimizing memory used to store local context for gradient calculation.



Project Roadmap



Milestone 3

K-D Tree

End-to-end Training on GPU

Complete Kernels

Final Week

GPU Optimizations

Improve Model Performance

Final Presentation

Thank You