Mesorasi: Architecture Support for Point Cloud Analytics via Delayed-Aggregation

https://github.com/horizon-research/Efficient-Deep-Learning-for-Point-Clouds

Yu Feng, Boyuan Tian, Tiancheng Xu
with Paul Whatmough (Arm Research) and Yuhao Zhu

Department of Computer Science
University of Rochester
http://horizon-lab.org
Deep Learning on Point Clouds

Classification

Segmentation

Detection

SLAM
Deep Learning on Point Clouds

**Performance**

**Energy-efficiency**

---

**PointNet:** Deep Learning on Point Sets for 3D Classification and Segmentation

**PointNet++:** Deep Hierarchical Feature Learning on Point Sets in a Metric Space

**OctNet:** Learning Deep 3D Representations at High Resolutions

**FoldingNet:** Point Cloud Auto-encoder via Deep Grid Deformation

**A Network Architecture for Point Cloud Classification via Automatic Depth Images Generation**

**3D Graph Neural Networks for RGBD Semantic Segmentation**

**SGPNet:** Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation

**SpiderCNN:** Deep Learning on Point Sets with Parameterized Convolutional Filters

**SqueezeSeg:** Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud

---

**Deep Continuous Fusion for Multi-Sensor 3D Object Detection**

**PointPillars:** Fast Encoders for Object Detection from Point Clouds

**Deep Feature Voting for 3D Object Detection in Point Clouds**

**Detection and Tracking of Small Objects in Sparse 3D Laser Range Data**

**MVX-Net:** Multimodal VoxelNet for 3D Object Detection

**PointPainting:** Sequential Fusion for 3D Object Detection

**TANet:** Robust 3D Object Detection from Point Clouds with Triple Attention

---

**SLICloud:** Semantic Segmentation of 3D Point Clouds

[http://segcloud.stanford.edu](http://segcloud.stanford.edu)
Deep Learning on Point Clouds

Can we use existing neural network accelerators on point cloud workloads?

PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space

OctNet: Learning Deep 3D Representations at High Resolutions

Multi-View 3D Object Detection Network for Autonomous Driving

Vote3Deep: Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks

PIXOR: Real-time 3D Object Detection from Point Clouds

3D Graph Neural Networks for RGBD Semantic Segmentation

SEGCloud: Semantic Segmentation of 3D Point Clouds

SGPN: Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation

Spider-CNN: Deep Learning on Point Sets with Parameterized Convolutional Filters

SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud

Detection and Tracking of Small Objects in Sparse 3D Laser Range Data

MVX-Net: Multimodal VoxelNet for 3D Object Detection

PointPainting: Sequential Fusion for 3D Object Detection

TANet: Robust 3D Object Detection from Point Clouds with Triple Attention

Can we use existing neural network accelerators on point cloud workloads?
Key Operators

Neighbor Search

Aggregation

P1: {P2, P3, …}
P3: {P8, P7, …}
... P6: {P8, P5, …}

Feature Computation

Mat Mul 1 ReLU
Mat Mul 2 ReLU
Mat Mul 3 ReLU
Key Operators

Neighbor Search

Aggregation

Feature Computation

P1: {P2, P3, ....}
P3: {P8, P7, ....}
... 
P6: {P8, P5, ....}

Mat Mul 1 ReLU

Mat Mul 2 ReLU

Mat Mul 3 ReLU
Neighbor Search in Point Clouds
Neighbor Search in Point Clouds

P1:
\{ P0, P2, P3, P4, P5, P6 \}

P8:
\{ P5, P6, P7, P9, P10, P11 \}
Neighbor Search in conventional DNN

Regular DNNs don’t need explicit neighbor search

Pixels: inherently regular
Points: irregularly scattered
Key Operators

Neighbor Search

Aggregation

Feature Computation

P1: \{P2, P3, \ldots\}
P3: \{P8, P7, \ldots\}
\ldots
P6: \{P8, P5, \ldots\}

- Mat Mul 1
- ReLU
- Mat Mul 2
- ReLU
- Mat Mul 3
- ReLU
Aggregation

Points: feature vectors

Neighbor Index Table

P1: { P0, P2, P3, P4, P5, P6 }

P8: { P5, P6, P7, P9, P10, P11 }

P1:

Feature Matrix

A
Aggregation

Points: feature vectors

Neighbor Index Table

P1:
{ P0, P2, P3, P4, P5, P6 }

P8:
{ P5, P6, P7, P9, P10, P11 }

P0

P1: Feature Matrix

A
Aggregation

Points: feature vectors

Neighbor Index Table

**P1:**
{ P0, P2, P3, P4, P5, P6 }

**P8:**
{ P5, P6, P7, P9, P10, P11 }

P0 - P1
Aggregation

Points: feature vectors

Neighbor Index Table

P1:
\{ P0, P2, P3, P4, P5, P6 \}

P8:
\{ P5, P6, P7, P9, P10, P11 \}

P1: Feature Matrix

P0 - P1
P2 - P1
P3 - P1
P4 - P1
P5 - P1
P6 - P1
Aggregation

Points: feature vectors

Neighbor Index Table

**P1:**
{ P0, P2, P3, P4, P5, P6 }

**P8:**
{ P5, P6, P7, P9, P10, P11 }

P1: Feature Matrix

P8: Feature Matrix

A

P0 - P1
P2 - P1
P3 - P1
P4 - P1
P5 - P1
P6 - P1
P5 - P8
P6 - P8
P7 - P8
P9 - P8
P10 - P8
P11 - P8
Key Operators

Neighbor Search

Aggregation

Feature Computation

P1: \{P2, P3, \ldots\}
P3: \{P8, P7, \ldots\}
\ldots
P6: \{P8, P5, \ldots\}

Mat Mul 1 \rightarrow ReLU

Mat Mul 2 \rightarrow ReLU

Mat Mul 3 \rightarrow ReLU
Feature Computation

**P1**: Feature Matrix
- P0 - P1
- P2 - P1
- P3 - P1
- P4 - P1
- P5 - P1
- P6 - P1
- P7 - P8
- P6 - P8
- P5 - P8
- P4 - P8
- P3 - P8
- P2 - P8
- P1 - P8

**P8**: Feature Matrix
- P0 - P1
- P2 - P1
- P3 - P1
- P4 - P1
- P5 - P1
- P6 - P1
- P7 - P8
- P6 - P8
- P5 - P8
- P4 - P8
- P3 - P8
- P2 - P8
- P1 - P8

**P1’**: New Feature Matrix

**P8’**: New Feature Matrix
Feature Computation

**P1**: Feature Matrix
- P0 - P1
- P2 - P1
- P3 - P1
- P4 - P1
- P5 - P1
- P6 - P1
- P5 - P8
- P6 - P8
- P7 - P8
- P9 - P8
- P10 - P8
- P11 - P8

**P8**: Feature Matrix
- P5 - P8
- P6 - P8
- P7 - P8
- P9 - P8
- P10 - P8
- P11 - P8

**P1'**: New Feature Matrix
- P0'
- P2'
- P3'
- P4'
- P5'
- P6'

**P8'**: New Feature Matrix
- P5'
- P6'
- P7'
- P9'
- P10'
- P11'

**Reduction**

**P1'**
- New Feature Vector

**P8'**
- New Feature Vector
Point Cloud Network Layer

- **Neighbor Search**
  - Points: \( P_1, P_2, P_3, P_4, P_5, P_6 \)
  - M-dimension feature vectors
  - Neighbor index table:
    - \( P_1: \{P_2, P_3\} \)
    - \( P_5: \{P_3, P_6\} \)
  - N neighbors (\( N = 2 \))
  - \( P_1, P_5 \): centroid points

- **Aggregation**
  - Neighbors feature matrix (NFM):
    - \( P_1: \begin{bmatrix} P_2 - P_1 \\ P_3 - P_1 \end{bmatrix} \)
    - \( P_5: \begin{bmatrix} P_6 - P_5 \\ P_3 - P_5 \end{bmatrix} \)
  - N \times M

- **Feature Computation**
  - MLP + Reduction:
    - Mat Mul 1 \( \rightarrow \) ReLU
    - Mat Mul 2 \( \rightarrow \) ReLU
    - Mat Mul 3 \( \rightarrow \) ReLU
    - Max Pool

- **MLP + Reduction**

- **Each centroid point has a NFM**
Can We Use Existing DNN Accelerators?

- No. They are not enough.
  - Neighbor Search
  - Aggregation
Can We Use Existing DNN Accelerators?

- No. They are not enough.
  - Neighbor Search
  - Aggregation
Point Cloud Network Layer

Points

P1, P2, P3, P4, P5, P6

M-dimension feature vectors

Neighbor Search

Neighbor Index Table

P1: {P2, P3}
P5: {P3, P6}

N neighbors (N == 2)
P1, P5: centroid points

Feature Computation

Neighbor Feature Matrix (NFM)

M'-dimension features

MLP + Reduction

Mat Mul 1 → ReLU → Mat Mul 2 → ReLU → Mat Mul 3 → ReLU → Max Pool

Aggregation

P1: P2 - P1, P3 - P1
P5: P6 - P5, P3 - P5

N x M

Each centroid point has a NFM

17
Optimization

Each centroid point has a NFM. The Neighbor Feature Matrix (NFM) consists of N x M P1 to P6 features. Features are computed through a series of operations:

1. Mat Mul 1
2. ReLU
3. Mat Mul 2
4. ReLU
5. Mat Mul 3
6. ReLU
7. Max Pool

This process results in MLP + Reduction and Feature Computation. The output includes P1' and P5', representing M'-dimension features.
Each centroid point has a NFM:

\[ \text{P1': P2 - P1} \]
\[ \text{P3 - P1} \]

\[ \text{N x M} \]

\[ \text{P5': P6 - P5} \]
\[ \text{P3 - P5} \]

\[ \text{N x M} \]

Optimization:

\[ \text{Mat Mul (P3 - P1, Kernel weights)} \]
\[ \text{Mat Mul (P3, Kernel weights)} \]
\[ \text{Mat Mul (P1, Kernel weights)} \]

\[ = \]

\[ \text{Mat Mul (P3 - P1, Kernel weights)} - \text{Mat Mul (P3, Kernel weights)} + \text{Mat Mul (P1, Kernel weights)} \]

\[ \text{MLP + Reduction} \]

\[ \text{Max Pool} \]

\[ \text{Neighbor Feature Matrix (NFM)} \]

\[ \text{Feature Computation} \]

\[ \text{M'-dimension features} \]
Optimization

- Each centroid point has a NFM

\[
\text{Mat Mul (} \ P_3 - P_1 \ \text{)} \quad \text{Kernel weights (} \ P_3 \ \text{)} \quad \text{Mat Mul (} \ P_1 \ \text{)} \quad \text{Kernel weights (} \ P_1 \ \text{)}
\]

\[
\begin{align*}
\text{Mat Mul (} \ P_3 \ \text{)} & \quad \text{N x M} \\
\text{Mat Mul (} \ P_1 \ \text{)} & \quad \text{N x M}
\end{align*}
\]

\[=\]

Benefit:

- Effectively introduces reuse opportunities
- Each point is used ~30 times
- Reduces up to 90% MAC (Multiply-Accumulate) operations
- Dependency elimination

\[
\text{Feature comp. (} \ P_3 - P_1 \ \text{)} \quad \text{Kernel weights (} \ P_3 \ \text{)} \quad \text{N x M}
\]

\[
\text{Mat Mul (} \ P_1 \ \text{)} \quad \text{Kernel weights (} \ P_1 \ \text{)} \quad \text{Mat Mul (} \ P_3 \ \text{)} \quad \text{Kernel weights (} \ P_3 \ \text{)} \quad \text{Mat Mul (} \ P_3 \ \text{)} \quad \text{Kernel weights (} \ P_3 \ \text{)} \quad \text{Mat Mul (} \ P_1 \ \text{)} \quad \text{Kernel weights (} \ P_1 \ \text{)}
\]

\[
\begin{align*}
\text{Mat Mul (} \ P_3 \ \text{)} & \quad \text{N x M} \\
\text{Mat Mul (} \ P_1 \ \text{)} & \quad \text{N x M}
\end{align*}
\]

Each point is used ~30 times
**Delayed Aggregation**

- **Points**
  - $P_1$
  - $P_2$
  - $P_3$
  - $P_4$
  - $P_5$
  - $P_6$

- **Neighbor Search**
  - Neighbor Index Table

- **Feature Compute**
  - Point Feature Table

- **Aggregation**
  - Neighbor Feature Matrix (NFM)

- **Reduction**
  - $P'_1$
  - $P'_5$

- **M'-dimension features**

M-dimension feature vectors:

- Mat Mul 1
- ReLU
- Mat Mul 2
- ReLU
- Mat Mul 3
- ReLU

**MLP**
Bottleneck Shift

Execution Time Distribution of PointNet++
Mesorasi: Point Cloud Acceleration Frame

Algorithm

Delayed Aggregation

Hardware

Aggregation Acceleration
Aggregation Operation

Neighbor Index Table

P1: \{P_2, P_3, P_{11}, P_{23}, P_{31}, P_{39}, \ldots\}

P3: \{P_5, P_7, P_{16}, P_{19}, P_{38}, P_{41}, \ldots\}

\ldots

P_{99}: \{P_{16}, P_{71}, P_{96}, P_{119}, P_{128}, P_{142}, \ldots\}

N neighbors

Point Feature Table

Neighbor Feature Matrix
Aggregation Operation

Neighbor Index Table

P1: \{P_2, P_3, P_{11}, P_{23}, P_{31}, P_{39}, \ldots\}

P3: \{P_5, P_7, P_{16}, P_{19}, P_{38}, P_{41}, \ldots\}

\ldots

P_{99}: \{P_{16}, P_{71}, P_{96}, P_{119}, P_{128}, P_{142}, \ldots\}

N neighbors

Point Feature Table

Neighbor Feature Matrix
Aggregation Unit

 Neighbor Index Table

 Point Feature Table (PFT)

900: {832, 987, ...}
....
6: {18, 25, 34, ...}
3: {8, 17, 6, ...}
1: {2, 17, 9, ...}

Address Generation
Aggregation Unit

Neighbor Index Table

900: {832, 987, ...}

....

6: {18, 25, 34, ...}

3: {8, 17, 6, ...}

1: {2, 17, 9, ...}

Point Feature Table (PFT)

Bank B

... 

Bank 3

Bank 2

Bank 1

Address Generation

B-ported, B-banked; No Crossbar
Aggregation Unit

Neighbor Index Table

Point Feature Table (PFT)

Bank B

Bank 3

Bank 2

Bank 1

Address Generation

B-ported, B-banked; No Crossbar

Reduction (Max)

Shift Registers

Bank 1

Bank 2

Bank 3

Bank B

Point Feature Table (PFT)
Aggregation Unit

Neighbor Index Table
- 900: \{832, 987, \ldots\}
- 6: \{18, 25, 34, \ldots\}
- 3: \{8, 17, 6, \ldots\}
- 1: \{2, 17, 9, \ldots\}

Address Generation

Shift Registers
- Reduction (Max)
- MUX
- (Store centroid's feature vector)

Point Feature Table (PFT)
- Bank B
- Bank 3
- Bank 2
- Bank 1

B-ported, B-banked; No Crossbar
Aggregation Unit

Neighbor Index Table

<table>
<thead>
<tr>
<th>900: {832, 987, …}</th>
</tr>
</thead>
<tbody>
<tr>
<td>…</td>
</tr>
<tr>
<td>6: {18, 25, 34, …}</td>
</tr>
<tr>
<td>3: {8, 17, 6, …}</td>
</tr>
<tr>
<td>1: {2, 17, 9, …}</td>
</tr>
</tbody>
</table>

Address Generation

Point Feature Table (PFT)

<table>
<thead>
<tr>
<th>Bank B</th>
</tr>
</thead>
<tbody>
<tr>
<td>…</td>
</tr>
<tr>
<td>Bank 3</td>
</tr>
<tr>
<td>Bank 2</td>
</tr>
<tr>
<td>Bank 1</td>
</tr>
</tbody>
</table>

B-ported, B-banked; No Crossbar

Shift Registers

<table>
<thead>
<tr>
<th>Reduction (Max)</th>
</tr>
</thead>
<tbody>
<tr>
<td>…</td>
</tr>
<tr>
<td>MUX</td>
</tr>
<tr>
<td>…</td>
</tr>
</tbody>
</table>

(Store centroid's feature vector)

Global Buffer

Bank 1

Bank 2

Bank 3
Overall Hardware Design

DNN Accelerator (NPU)

Feature Extraction
+
Aggregation

CPU

GPU
(Neighbor Search)

DRAM

Input Point Cloud
MLP Kernel Weights
MLP Intermediate Activations
Neighbor Index Table
Overall Hardware Design

DNN Accelerator (NPU)
- Aggregation Logic
- Point Feature Buffer
- Neighbor Index Buffer
  - Global Buffer (Weights /FMaps)
  - Reduction (Max)
  - Systolic MAC Unit Array
    - BN/ReLU/MaxPool
    - MCU
    - MCU

CPU

GPU (Neighbor Search)

DRAM
- Input Point Cloud
- MLP Kernel Weights
- MLP Intermediate Activations
- Neighbor Index Table
Overall Hardware Design

DNN Accelerator (NPU)

- Aggregation Logic
- Reduction (Max)
- Systolic MAC Unit Array
- Global Buffer (Weights / FMaps)
- BN/ReLU/MaxPool
- MCU

CPU

With 3.8% area overhead to the NPU

DRAM

- Input Point Cloud
- MLP Kernel Weights
- MLP Intermediate Activations
- Neighbor Index Table

GPU (Neighbor Search)
Experimental Setup

Three Point Cloud Applications:
▷ Object Classification, Object Segmentation, and Object Detection

Datasets:
▷ ModelNet40, ShapeNet, and KITTI dataset

Models:
▷ Classification: PointNet++ (c), DGCNN (c), LDGCNN, DensePoint
▷ Segmentation: PointNet++ (s), DGCNN (s)
▷ Detection: F-PointNet
Accuracy Comparison

- **PointNet++ (c)**
- **PointNet++ (s)**
- **DGCNN (c)**
- **DGCNN (s)**
- **F-PointNet**
- **LDGCNN**
- **DensePoint**

Accuracy (%)

- **Original**
- **Delayed Aggr.**
Hardware Performance Evaluation

Hardware Baseline:
▷ A generic NPU + GPU SoC.

Variants:
▷ Mesorasi-SW: delayed-aggregation without AU support.
▷ Mesorasi-HW: delayed-aggregation with AU support.
Speedup

- PointNet++ (c)
- PointNet++ (s)
- DGCNN (c)
- DGCNN (s)
- F-PointNet
- LDGCNN
- DensePoint

**Mesorasi-SW**
- 2.2

**Mesorasi-HW**
- 1.3
- 3.6
- AVG.: 1.9
Energy Savings

![Energy Savings Chart](chart.png)
Delayed-aggregation decouples neighbor search with feature computation and significantly reduces the overall workload.

Hardware support further maximizes the effectiveness of delayed-aggregation.

https://github.com/horizon-research/Efficient-Deep-Learning-for-Point-Clouds