The Iby and Aladar Fleischman
Faculty of Engineering
Tel Aviv University

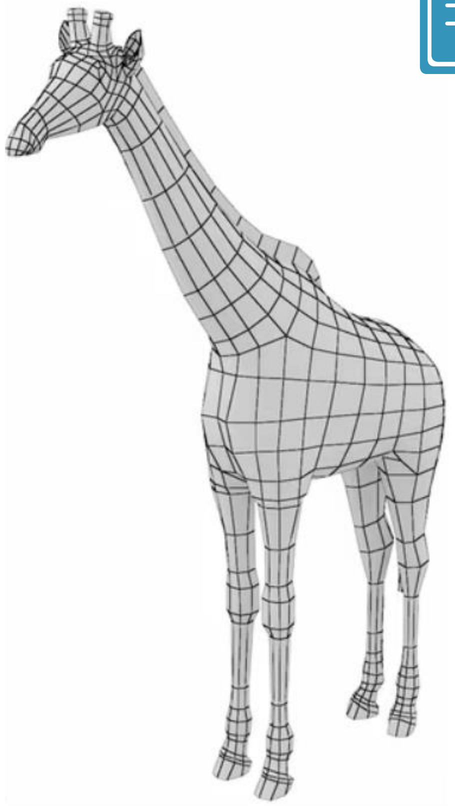
QuadMeshCNN

Omri Levi

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

Agenda

- **Introduction**
  - Mesh and other 3D data representations
  - Goal and motivation of this work
  - Previous work - MeshCNN
- **Quad MeshCNN** - method
- **Applications**
  - Classification model
  - QuadZoo5 dataset
- **Conclusions and future work**



Introduction

Basics, motivation and goal

CNNs on 2D images

- Shown an outstanding performance in classification and segmentation.

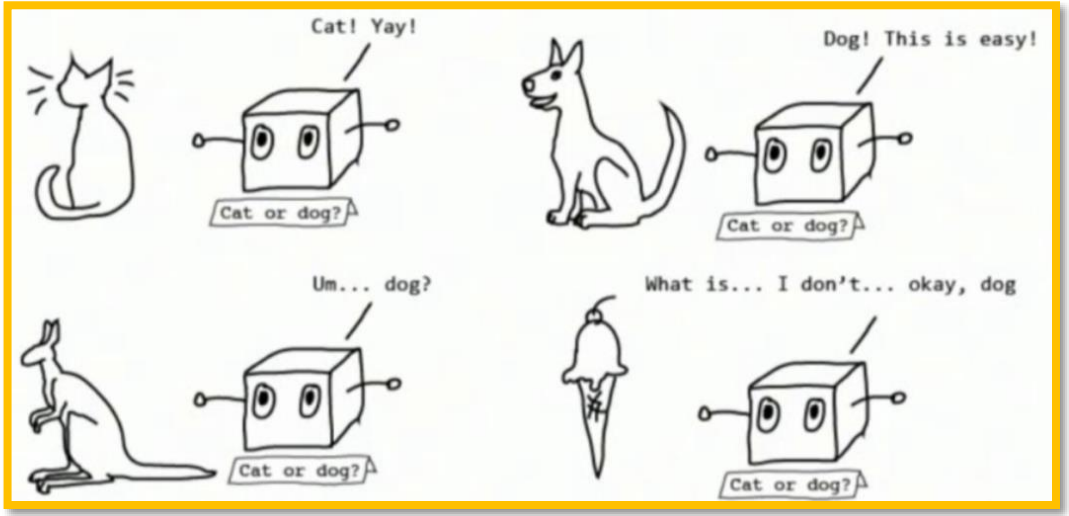
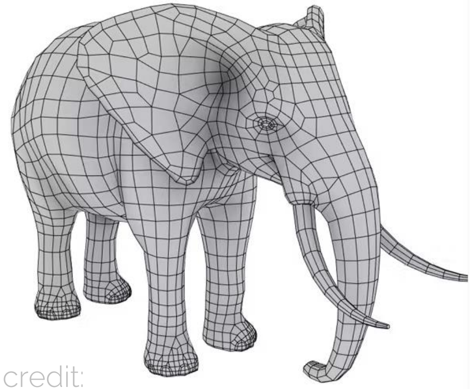
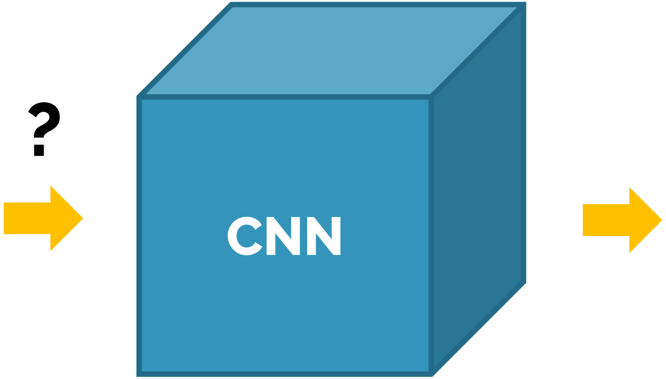


Image credit:
https://www.linkedin.com/pulse/why-your-machine-learning-project-might-fail-how-avoid-ben-gutkovich

Deep learning on 3D data

- Many representations exist
- 2D input data CNNs are hard to adjust





Output label:

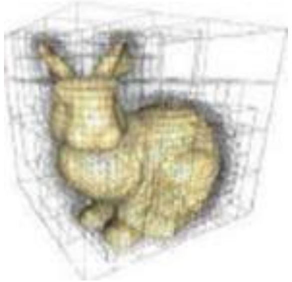
Elephant

Image credit:
https://3docean.net/item/elephant-base-mesh-low-poly/14341536

3D shapes representations

Multi-view data

Octree



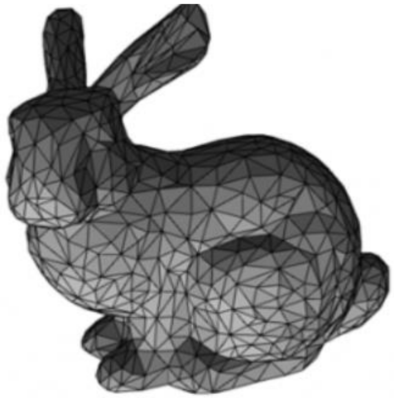
Voxels

3D descriptors

Graph

Point cloud

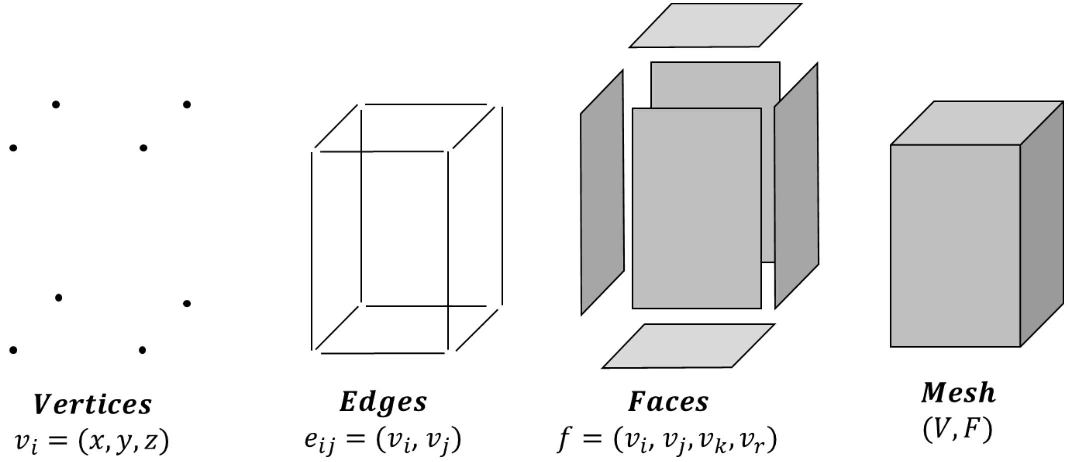
Polygon Mesh



Images from: Gezawa et al. 2020

Polygon Mesh

- Represents 2D surfaces
- Defined by vertices and faces (or edges)



Polygon Mesh

- Efficient representation
- Non-uniform

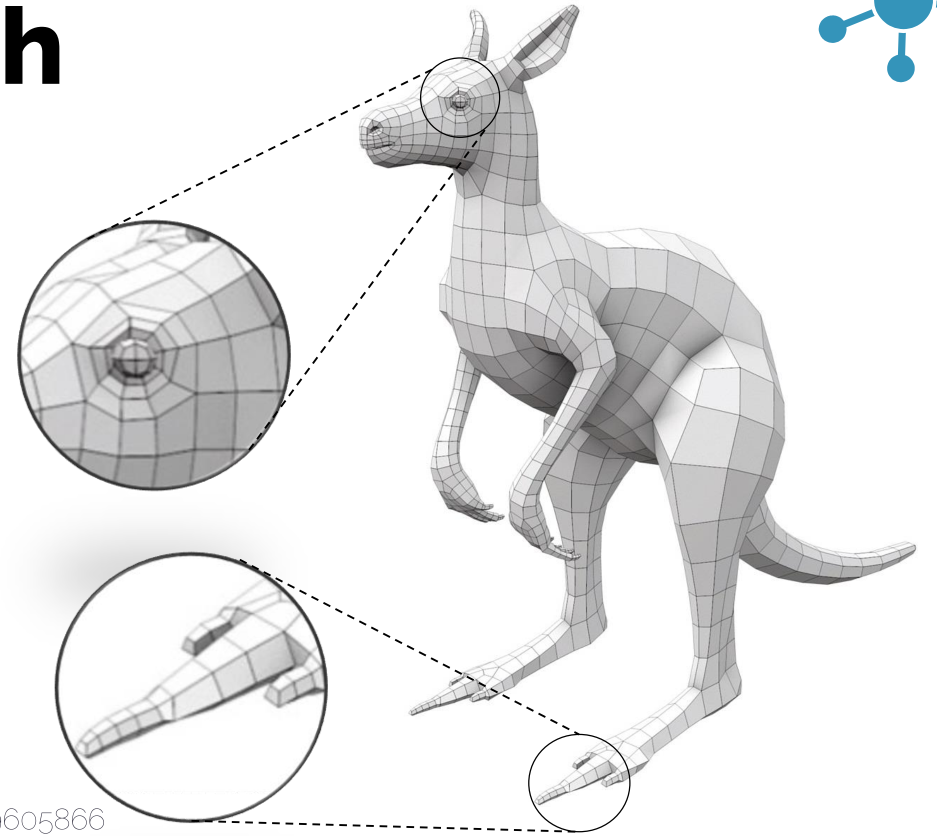
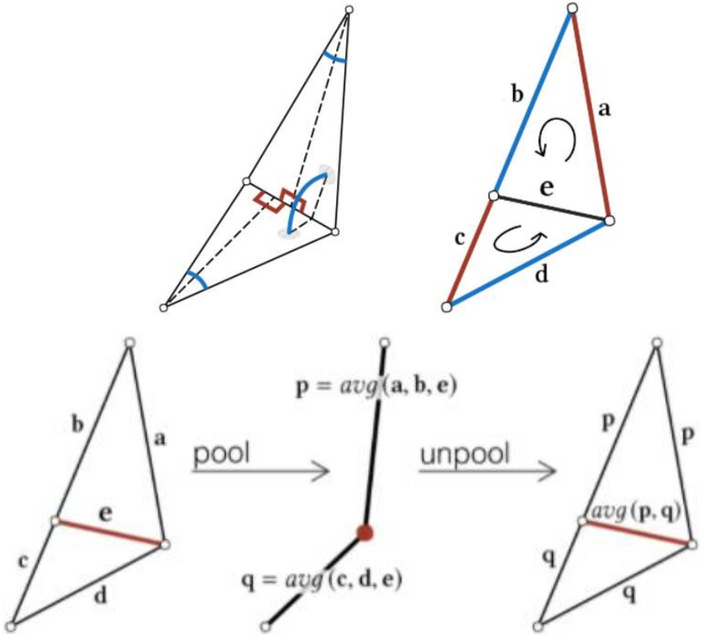


Image credit:
https://3docean.net/item/32-animals-base-meshes/19605866

MeshCNN [Hanocka, 2019]

- A method for 3D shapes processing.
- Works directly on the mesh structure.
- Combines specialized convolution and pooling layers
- Leveraging mesh edges intrinsic geodesic connections
- Showed great performance on classification and segmentation tasks.



MeshCNN on quads

Goal

- Extend MeshCNN method to process quad meshes.

Main contributions

- Develop and implement MeshCNN operations for quad meshes.
- Generate quad mesh classification dataset (QuadZoo5).
- Train a classification model for QuadZoo5.

Motivation: why quads?

Quads* are good for [Bommes et al. 2013]:

- Polygonal modeling
- High-order surface modeling
- Texturing
- Finite element simulation
- Compression

* Semi-regular quad mesh (example on the right) →

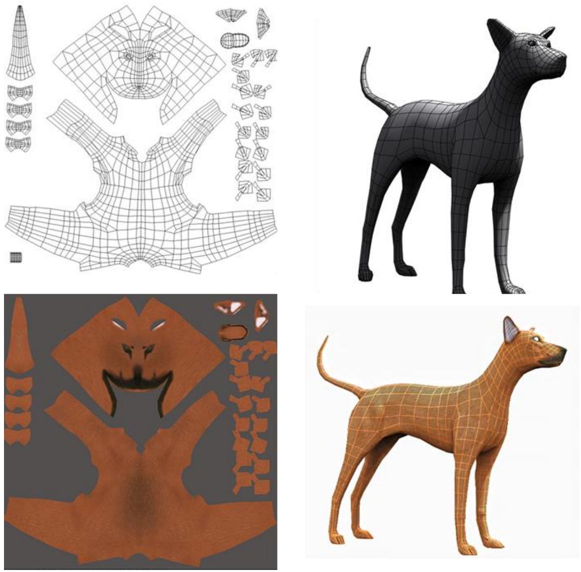


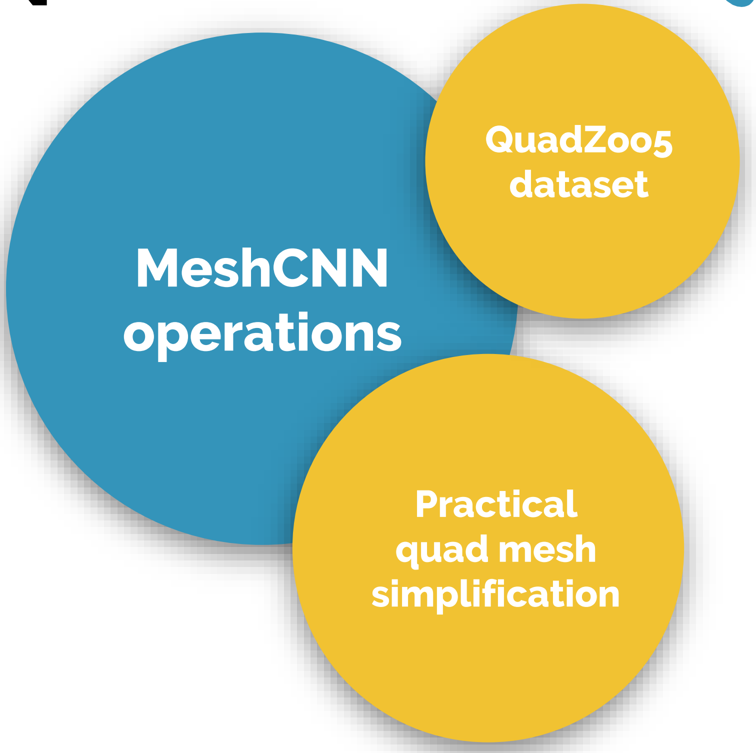
Image from: https://3docean.net/item/low-poly-dog/14334265

Quad MeshCNN

Method and basics

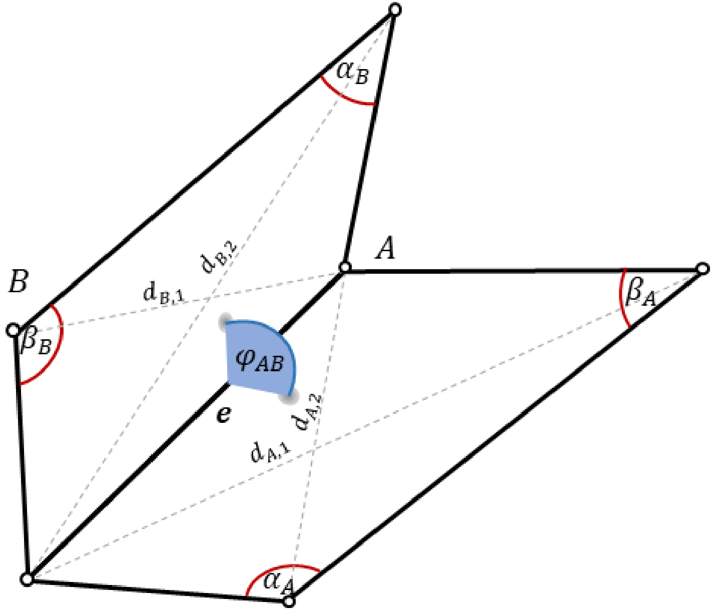
Quad MeshCNN

- MeshCNN extension
    - Input edge feature
    - Quad mesh convolution
    - Quad mesh pooling
    - Quad mesh unpooling
- Quad mesh simplification
- Classification dataset generation



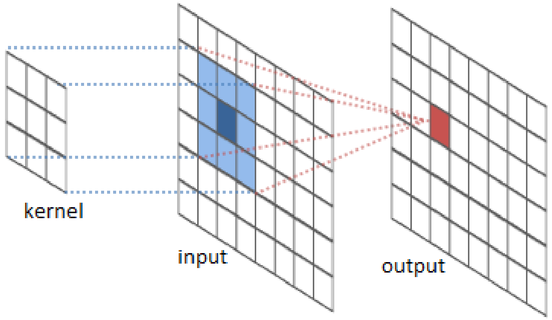
Input edge features

- Relative geometric features
  - Invariant to similarity transformations
- 7-dimentional vector

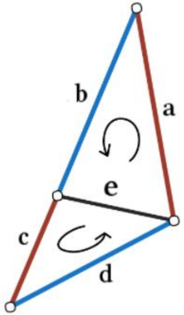


Mesh Convolution

- **Image convolution**
  - A dot product with a kernel
  - Convolution support is consistent



- **MeshCNN convolution**
  - Face normal defines face edges order
  - Convolution done on edges neighbourhood



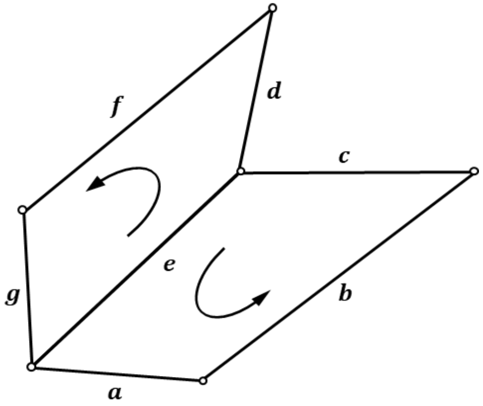
Quad Mesh Convolution

- Convolution on edges neighbourhood
- Edge neighbourhood ordering ambiguity

  (a,b,c,d,f,g) or (d,f,g,a,b,c)?

  → **Solution**: symmetric functions

$$f_e \cdot k_0 + \sum_{j=1}^{6} f_e^j \cdot k_j$$

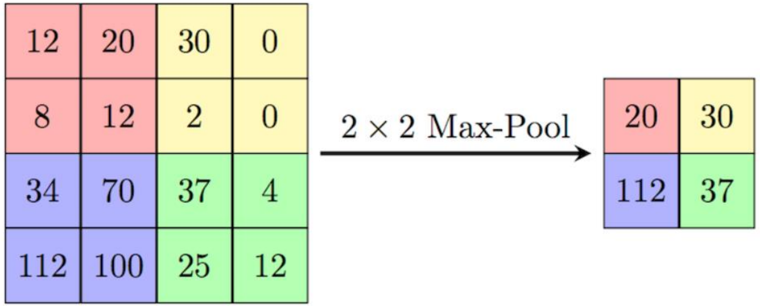


$$(f_e^1, f_e^2, f_e^3, f_e^4, f_e^5, f_e^6) = (|a-d|, a+d, |b-f|, b+f, |c-g|, g+c)$$

Mesh Pooling

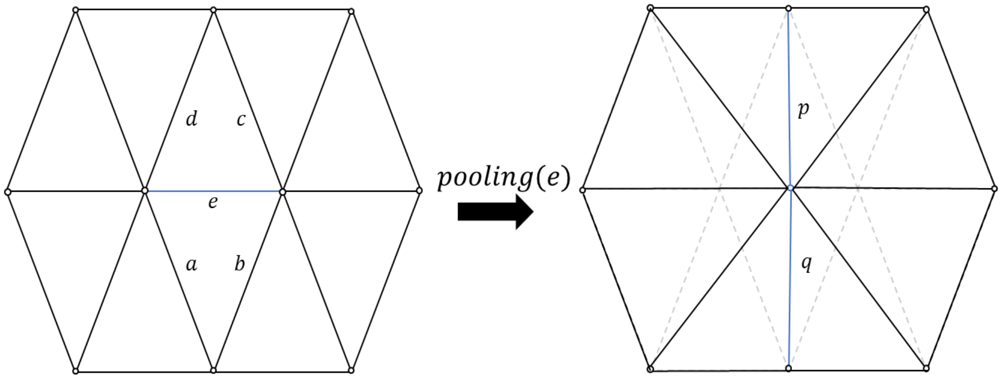
Reminder: pooling layer

- Reduces the dimension of feature activation maps
- Summarizes the features present in a region of activation map
- Provides basic translation invariance



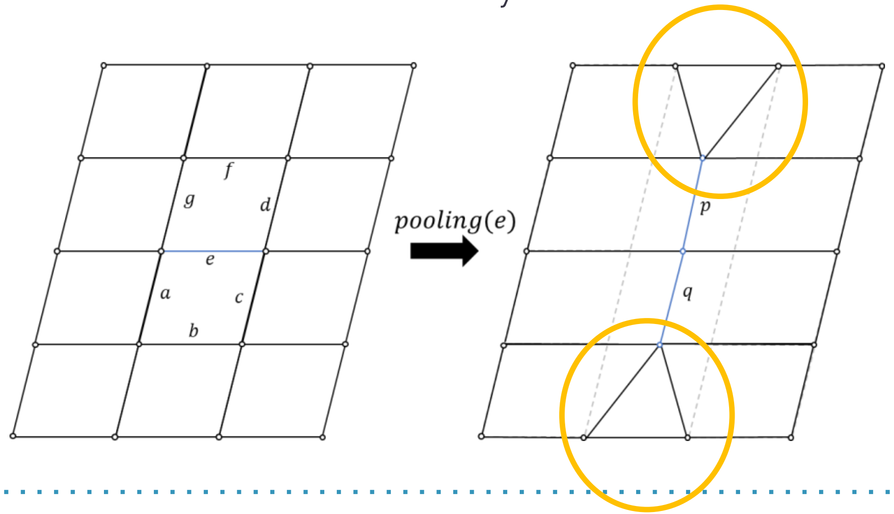
Mesh Pooling

- **Reminder: Triangle mesh pooling**
  - Based on edge collapse [Hoppe 1997]
  - Deleting edges with the smallest feature activations



Quad Mesh Pooling

- **Quads: Problem with same edge collapse method**
  - Quad structures are more delicate
  - Should be handled differently



Quad Mesh Pooling

- **Use local operations from quad mesh simplification method [Tarini et al. 2010]:**
  - Optimizing operations
  - Coarsening operations
  - Cleaning operations

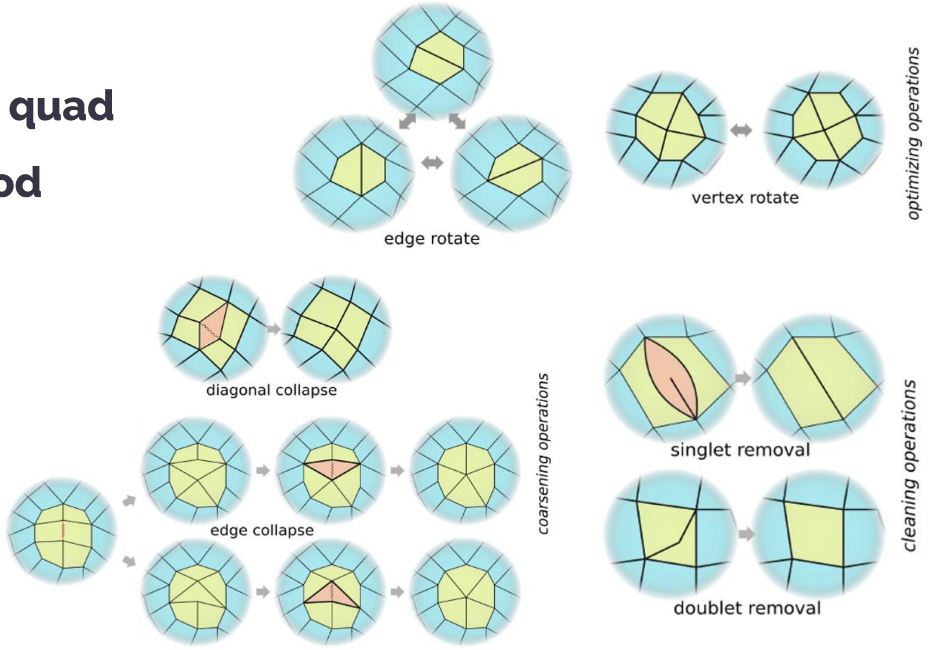
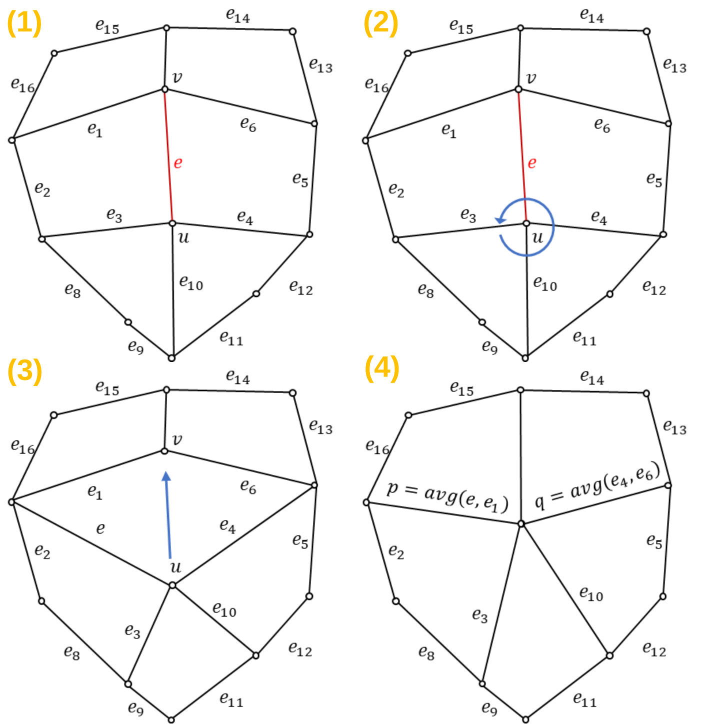


Image credit: Tarini et al. 2010

Quad Mesh Pooling

Algorithm 1 – quad edge-collapse

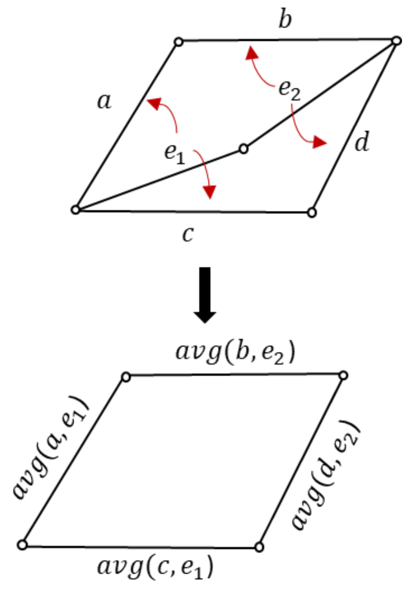
1. Extract edge_id **neighborhood information**.
2. Let (u, v) be the vertices of the edge_id.
   Select vertex u for step (3).
3. Perform **vertex_rotation** around vertex u.
4. Perform **diagonal_collapse** from vertex v.



Quad Mesh Pooling

Algorithm 2 – mesh pooling

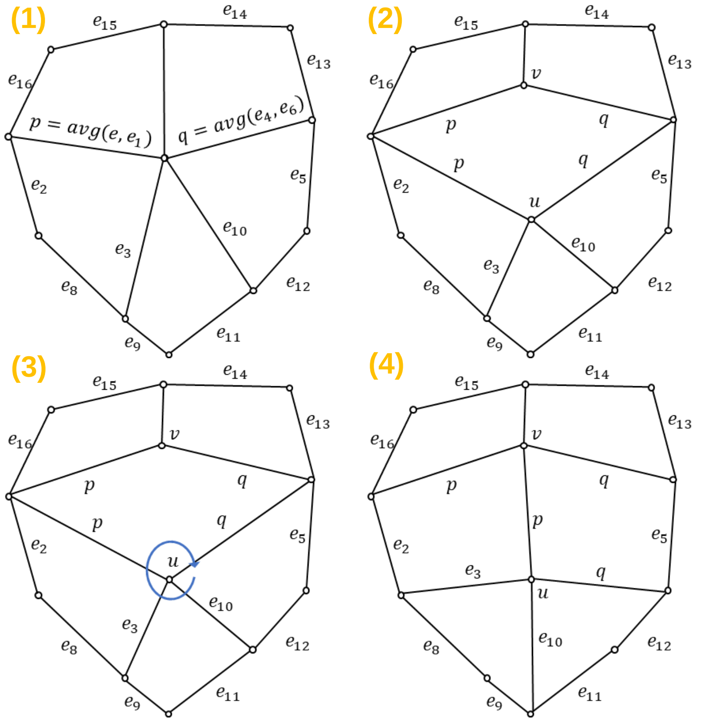
1. Build edges queue
2. While pooling_count < POOL_TARGET do:
   a. edge_id = Queue.pop()
   b. If edge_id is not removed continue to (c), else return to (a)
   c. Run clear_doublets(mesh) and clear_singlets(mesh)
   d. If edge_id is valid continue to (e), else return to (a).
   e. Perform edge_collapse(edge_ id).



Clear doublet example

Quad Mesh Unpooling

- Pooling layer collects the **connectivity prior** the pooling operation
- Mesh unpooling is a **partial inverse** of pooling operator
  - **Increases resolution** by restoring the connectivity
  - Unpooled features are a **weighted sum** of pooled features
- Quad mesh unpooling is similar to the triangle mesh unpooling



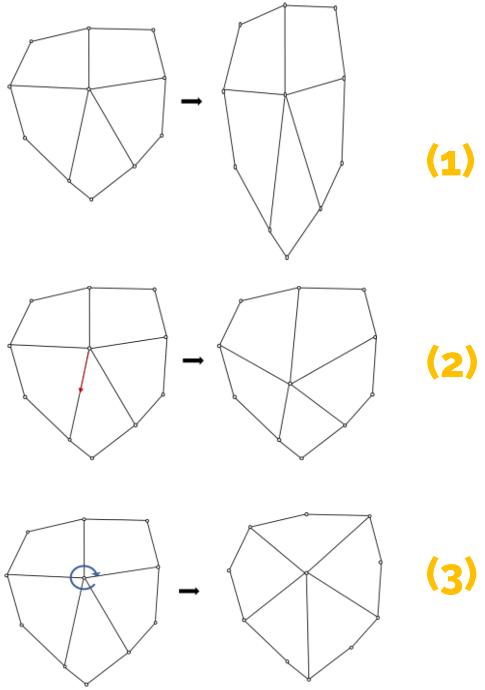
Quad Mesh augmentations

Why?

- Adding more training data
- Regularization - reduces overfitting

How?

- Directly on mesh structure
  (i.e., on vertices and edges).
- Using similarity variant augmentations:
  1. Anisotropic vertices position scaling
  2. Vertex location shift on mesh surface
  3. Tessellation change using vertex rotations



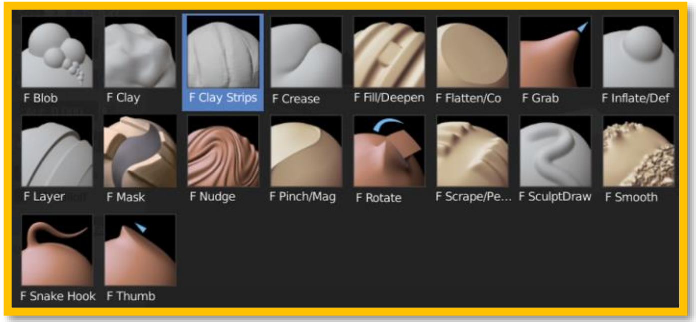
Quad MeshCNN

Application

Classification dataset and classification model

Dataset generation

- **Triangle to quad mesh conversion**
  - Instant field aligned meshes
  - Quadriflow
  - Quad-Remesher
- **Object deformations**
  - Blender sculpting operations
  - Anisotropic scaling



Dataset generation

- QuadZoo5 dataset vs. SHREC11 dataset

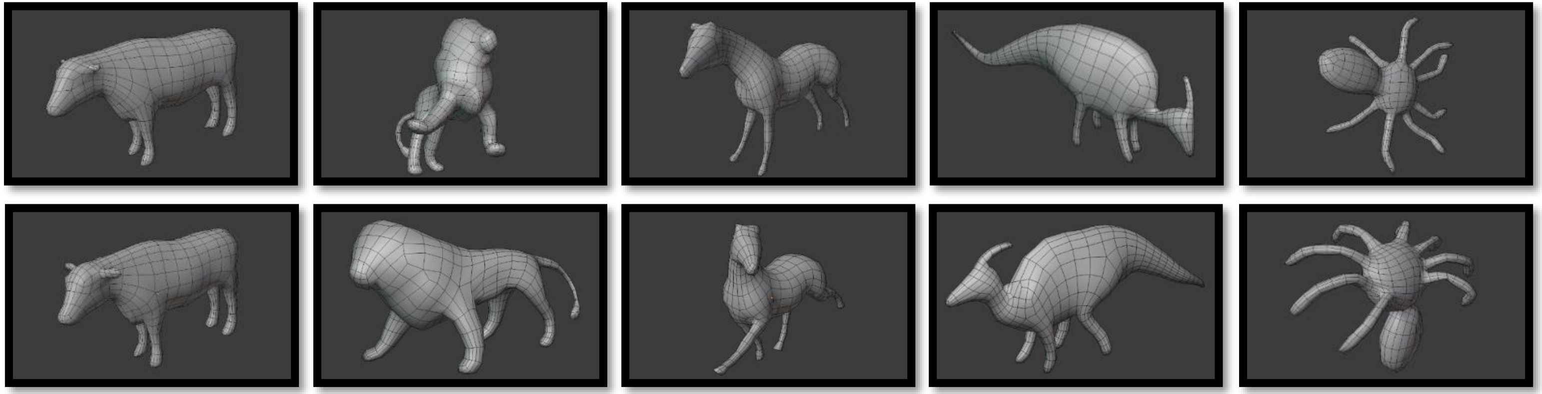
Dataset	# Classes	# Train examples	# Test examples	# Edges input	Mesh type
SHREC11	30	16	4	750	Triangle
QuadZoo5	5	11	4	[1396, 1954]	Quad

QuadZoo5 dataset

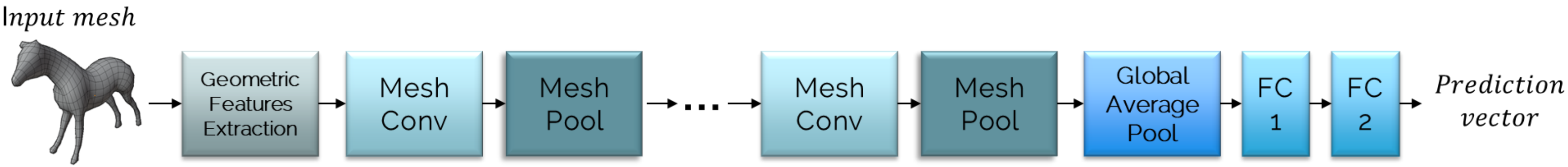
- Dataset examples (train and test)
- Download link

Train

Test



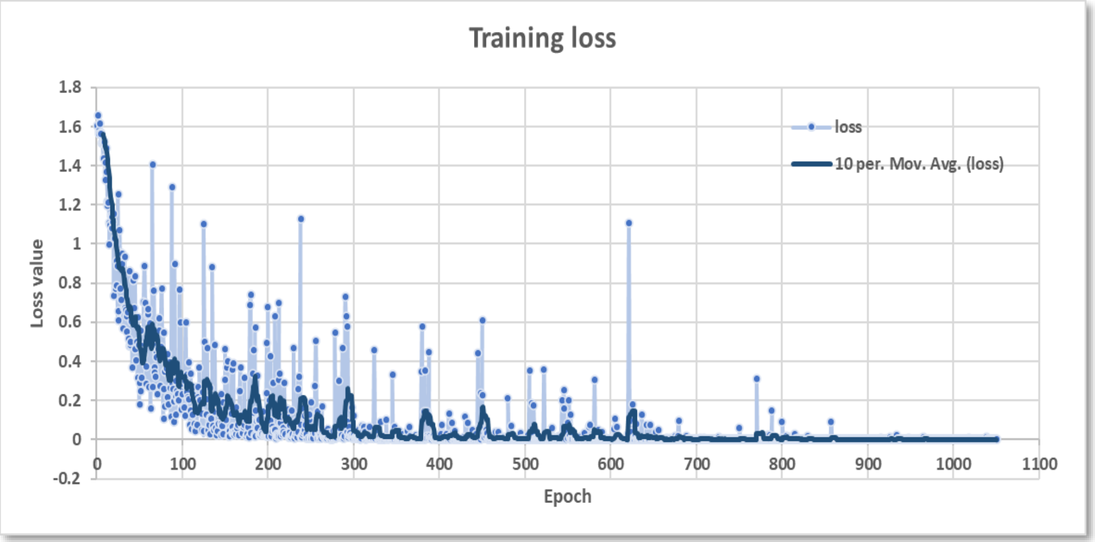
Classification model



#	Layer	Output	#	Layer	Output	#	Layer	Output
1	Input mesh	-	6	MeshPool2 (out Res. 900)	900x128	11	Global Avg. Pool	1x256
2	Geometric features extraction	$n_e x7$ $n_e \in [1396, 1954]$	7	MeshConv3 (+ReLU)	900x256	12	FC1 + ReLU	1x100
3	MeshConv1 (+ReLU)	$n_e x64$	8	MeshPool3 (out Res 600)	600x256	13	FC2 + ReLU	1x5
4	MeshPool1 (out Res. 1100)	1100x64	9	MeshConv4 (+ReLU)	600x256	14	Prediction Vector	1x5
5	MeshConv2 (+ReLU)	1100x128	10	MeshPool4 (out Res. 400)	400x256	15	Classification (arg max)	Class number

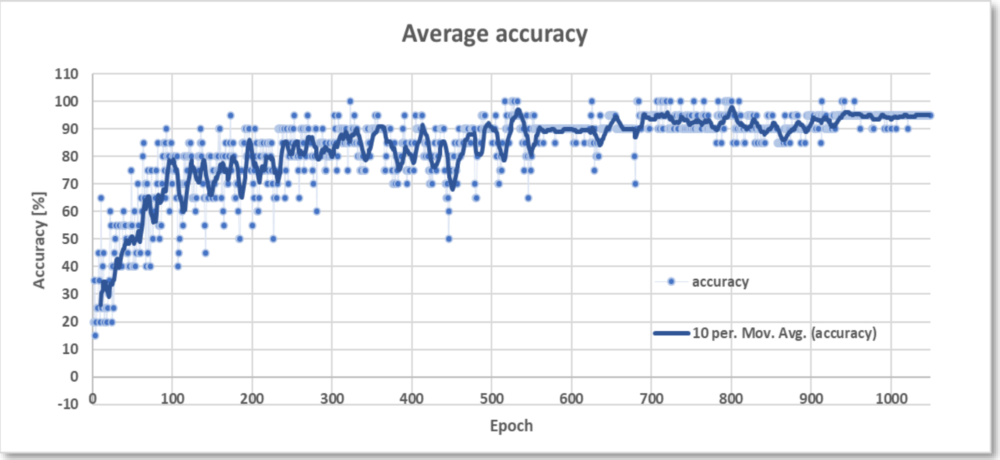
Experiments & Results

- Pytorch framework
- Configuration
  - 1050 epochs
  - Adam optimizer
  - LR scheduler
  - BCE loss
- Data
  - 55 train examples
  - 30 variants each
  - 20 test examples



Experiments & Results

- **Classification accuracy** on test set during training – converged to 95%
- **Confusion matrix** – summarizes correct and wrong predictions



		Predicted class				
	Class	Cow	Dilo	Horse	Lion	Spider
True class	Cow	75%	0%	0%	25%	0%
	Dilo	0%	100%	0%	0%	0%
	Horse	0%	0%	100%	0%	0%
	Lion	0%	0%	0%	100%	0%
	Spider	0%	0%	0%	0%	100%

Pooling examples

Input Mesh

Last pooling layer output

Conclusions and Future work

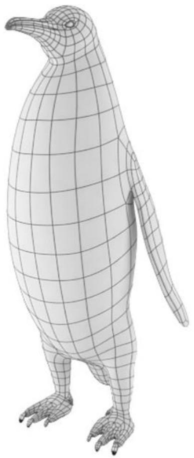
- **Conclusions**
  - QuadMeshCNN is an extension to MeshCNN
  - Quad MeshPool layer based on practical quad mesh simplification method.
  - QuadZoo5 – new classification dataset
  - Classification accuracy of 95%.

- **Future work**
  - Triangle and quad polygons in a mesh
  - Optimizing mesh pooling layer
  - Mesh segmentation task

The Iby and Aladar Fleischman
Faculty of Engineering
Tel Aviv University

Thanks!

Any questions?



GitHub

Special thanks to Rana Hanocka