Geometric Deep Learning - Project presentation
28/05/2019

Dynamic Graph CNN

Francesco Saverio Zuppichini

Photo by Scott Webb on Unsplash

Model

Model

Dynamic Graph CNN for Learning on Point Clouds

Yue Wang*
MIT
yuewang@csail.mit.edu

Yongbin Sun*
MIT
yb_sun@mit.edu

Ziwei Liu
UC Berkeley
zwliu@icsi.berkeley.edu

Sanjay E. Sarma
MIT
sesarma@mit.edu

Michael M. Bronstein
USI / TAU / Intel
michael.bronstein@usi.ch

Justin M. Solomon
MIT
jsolomon@mit.edu

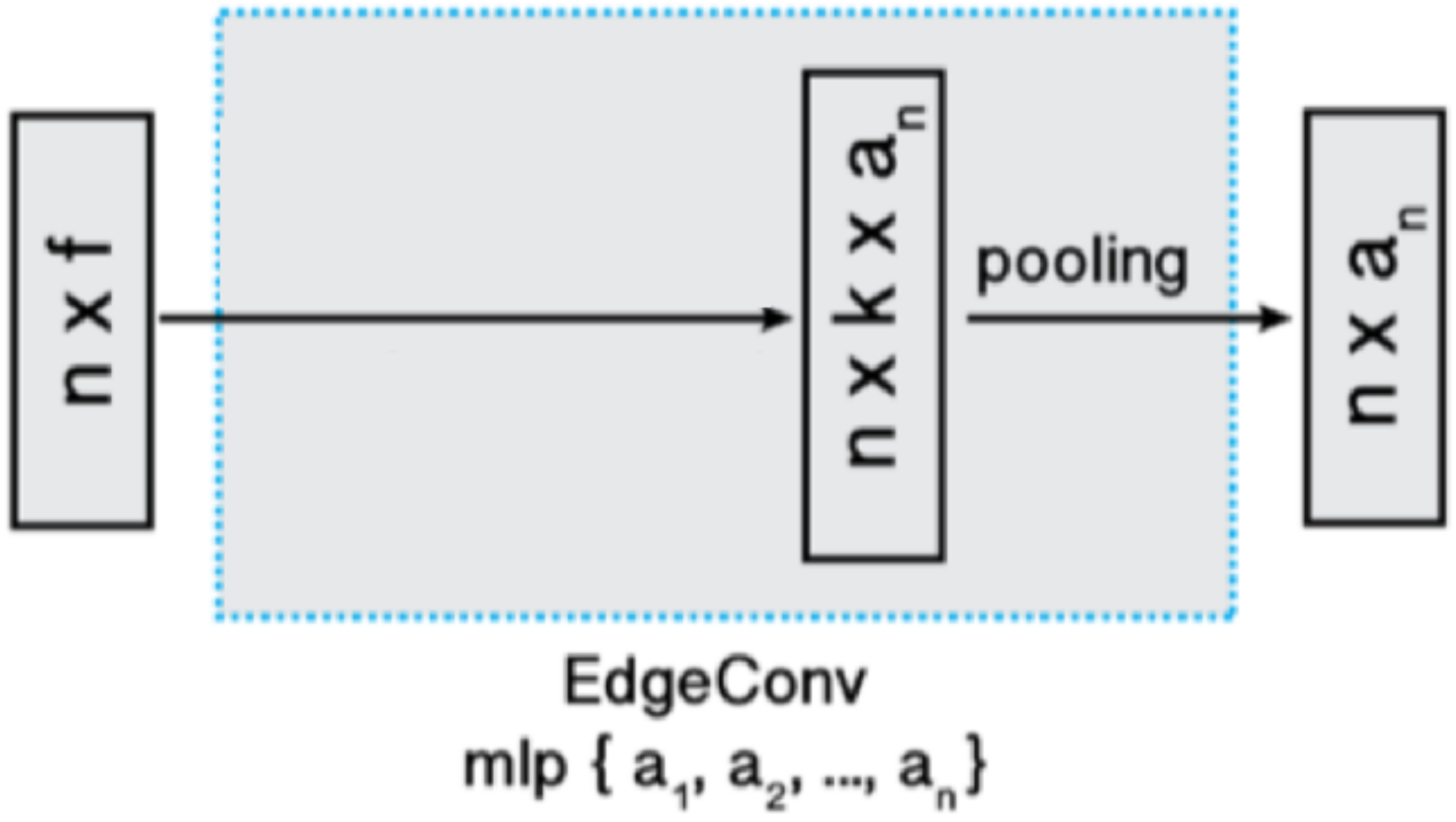
classify/segment **point cloud**

plug and play architecture

use **local** and **global** information

Model

EdgeConv



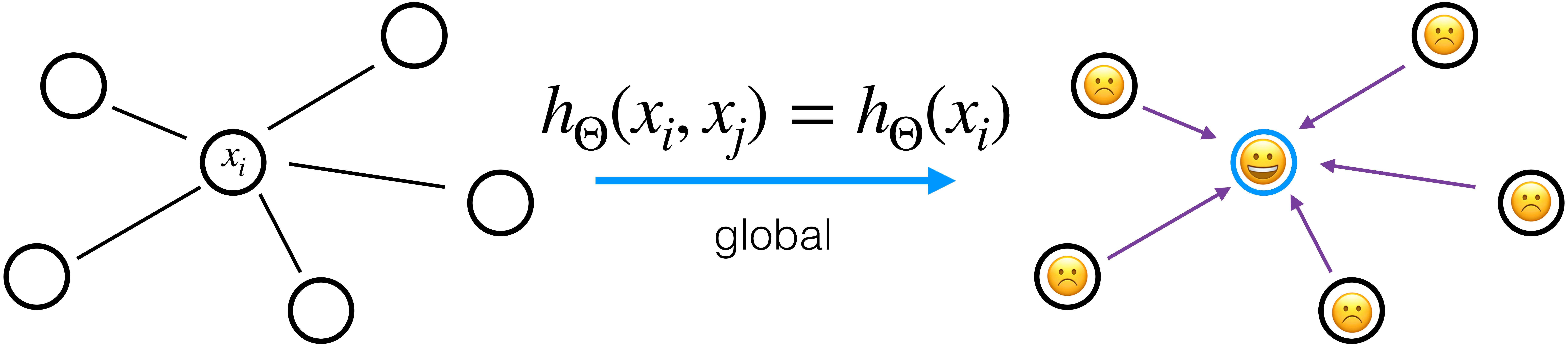
$$x_i' = \square_{j:(i,j)\in\mathscr{E}} h_{\Theta}\left(x_i, x_j\right)$$

x_i: point

$e_{ij} = h_{\Theta}(x_i, x_j)$ edge features

Model

choice of edge function

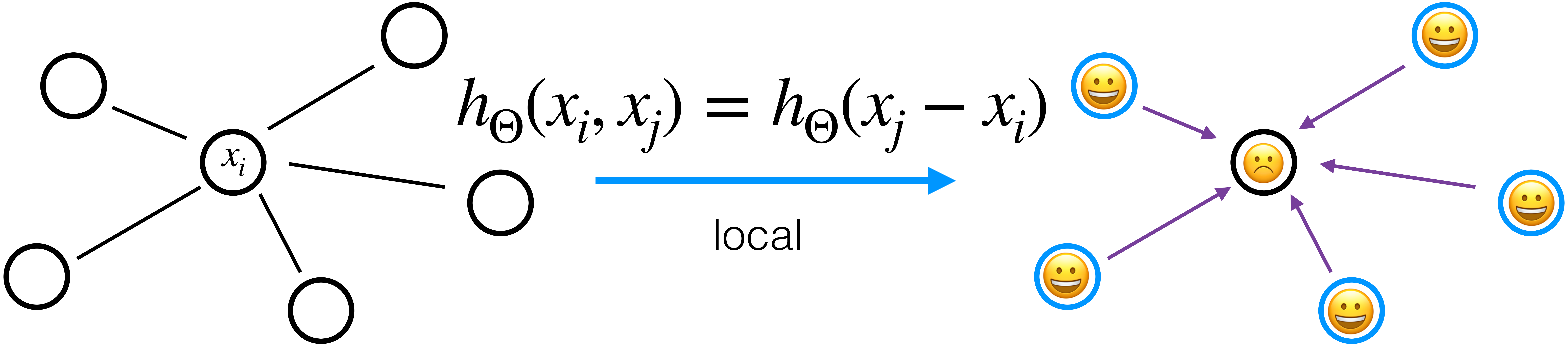


$$h_\Theta(x_i, x_j) = h_\Theta(x_i)$$

global

Model

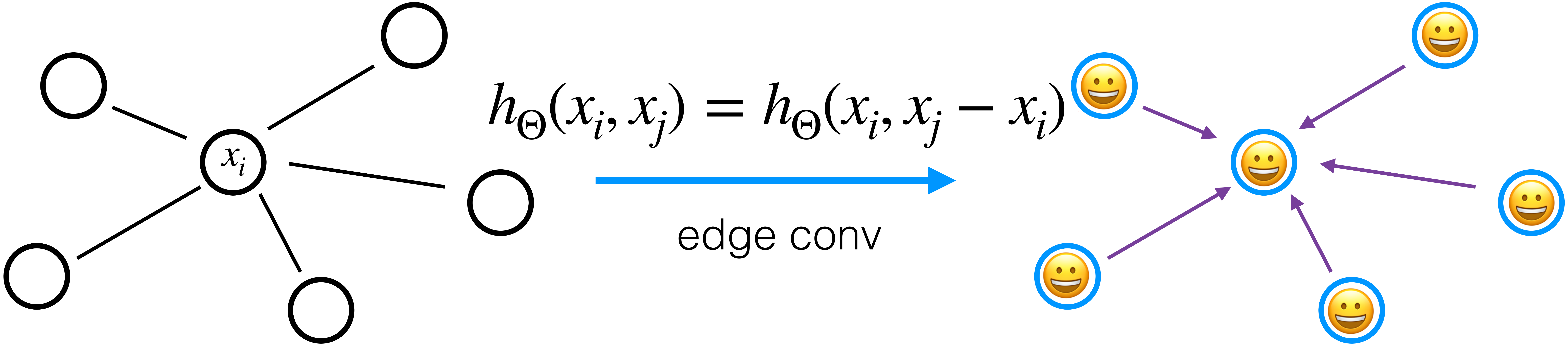
choice of edge function



$$h_\Theta(x_i, x_j) = h_\Theta(x_j - x_i)$$

Model

choice of edge function

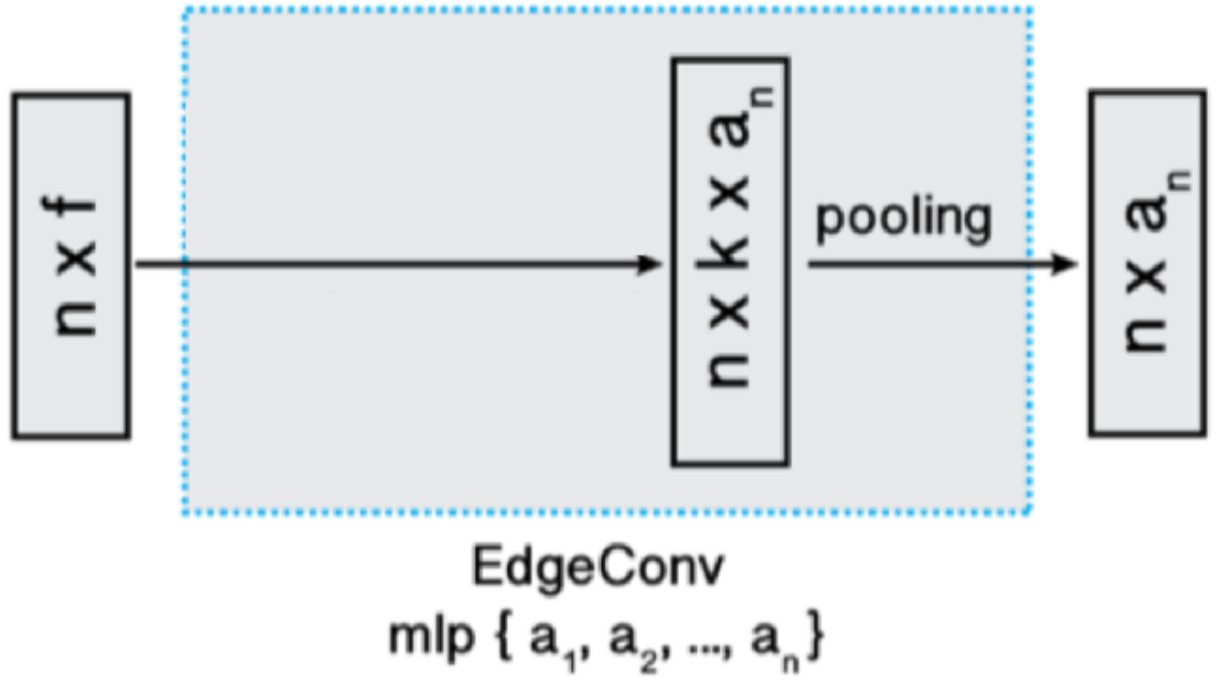


$$h_\Theta(x_i, x_j) = h_\Theta(x_i, x_j - x_i)$$

edge conv

Model

EdgeConv



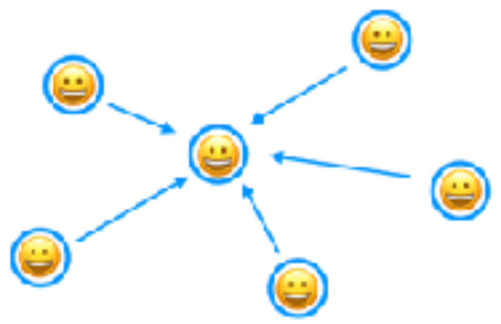
EdgeConv
mlp { a$_1$, a$_2$, ..., a$_n$ }

$$x'_i = \square_{j:(i,j)\in\mathcal{E}} h_\Theta\left(x_i, x_j\right)$$

$$\square = max$$

$$h_\Theta(x_i, x_j) = h_\Theta(x_i, x_j - x_i),\ h_\Theta : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}^F$$

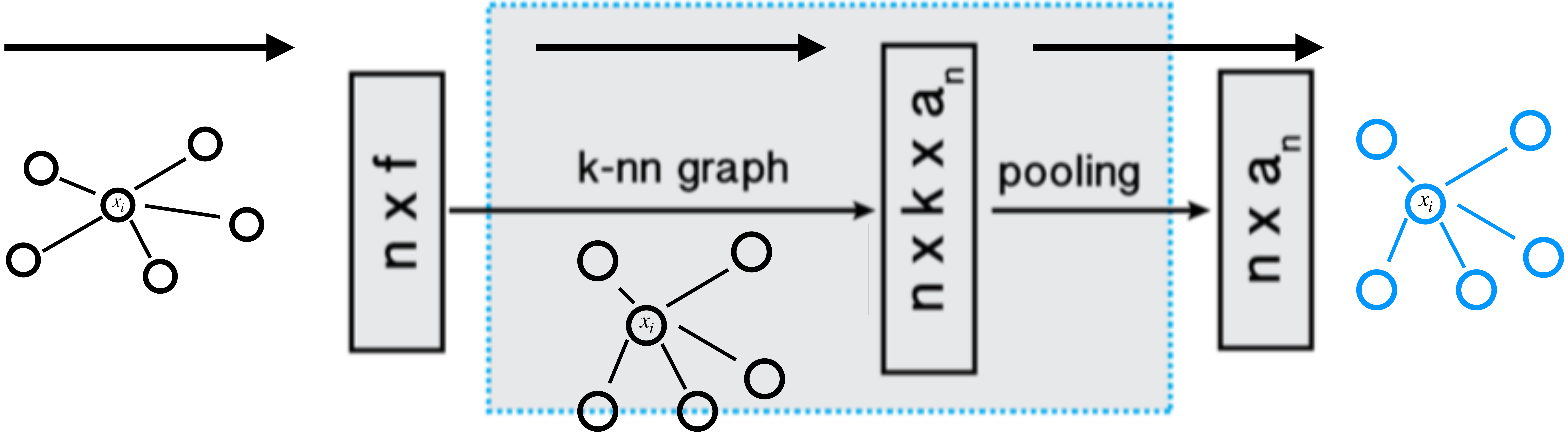
=



Model

Dynamic EdgeConv

$$\max h_{\Theta}^{(l)}(x_i, x_j - x_i)$$

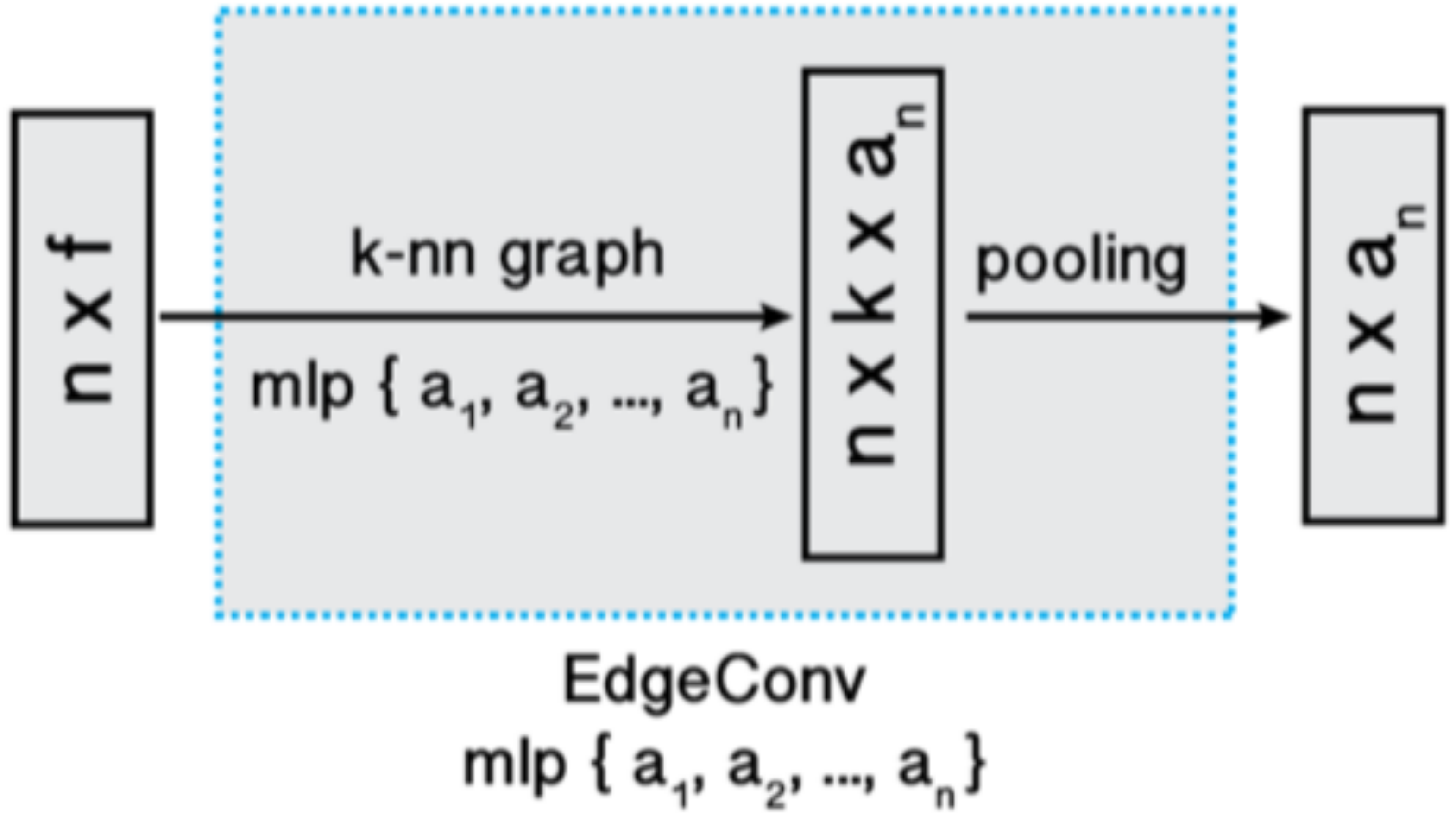


different graph at each layer!

Notation from https://rusty1s.github.io/pytorch_geometric/build/html/notes/create_gnn.html

Model

Dynamic EdgeConv



$$x_i^{(l+1)} = \square_{j:(i,j)\in\mathcal{E}^{(l)}} h_{\Theta}^{(l)}\left(x_i^{(l)}, x_j^{(l)}\right)$$

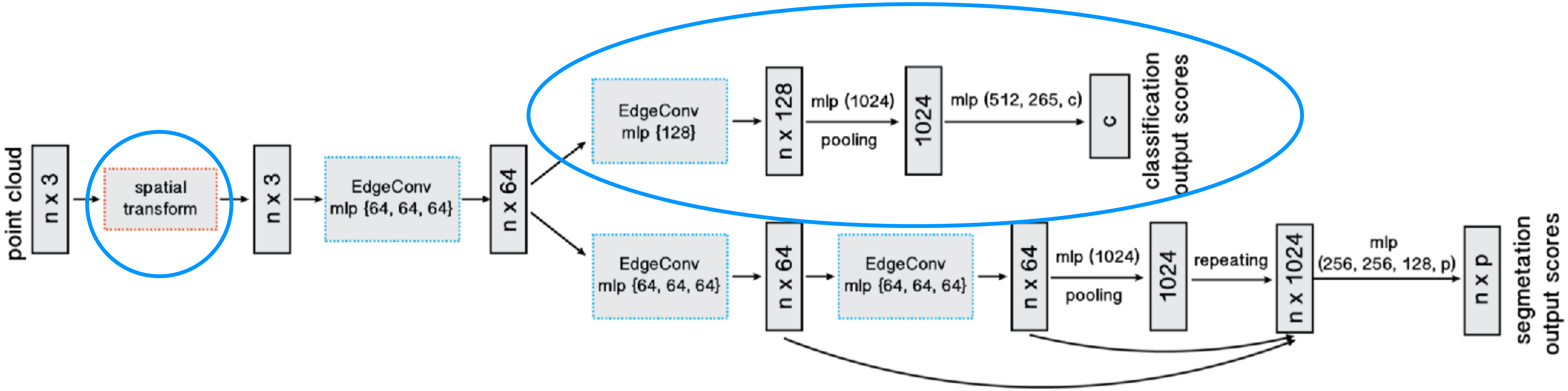
$$\square = max$$

$$h_{\Theta}^{l}(x_i, x_j) = h_{\Theta}^{(l)}(x_i, x_j - x_i),\ h_{\Theta}^{(l)} : \mathbb{R}^{F_l} \times \mathbb{R}^{F_l} \to \mathbb{R}^{F_{l+1}}$$

Notation from https://rusty1s.github.io/pytorch_geometric/build/html/notes/create_gnn.html

Model

Final architecture



Evaluation

Dataset

ModelNet40

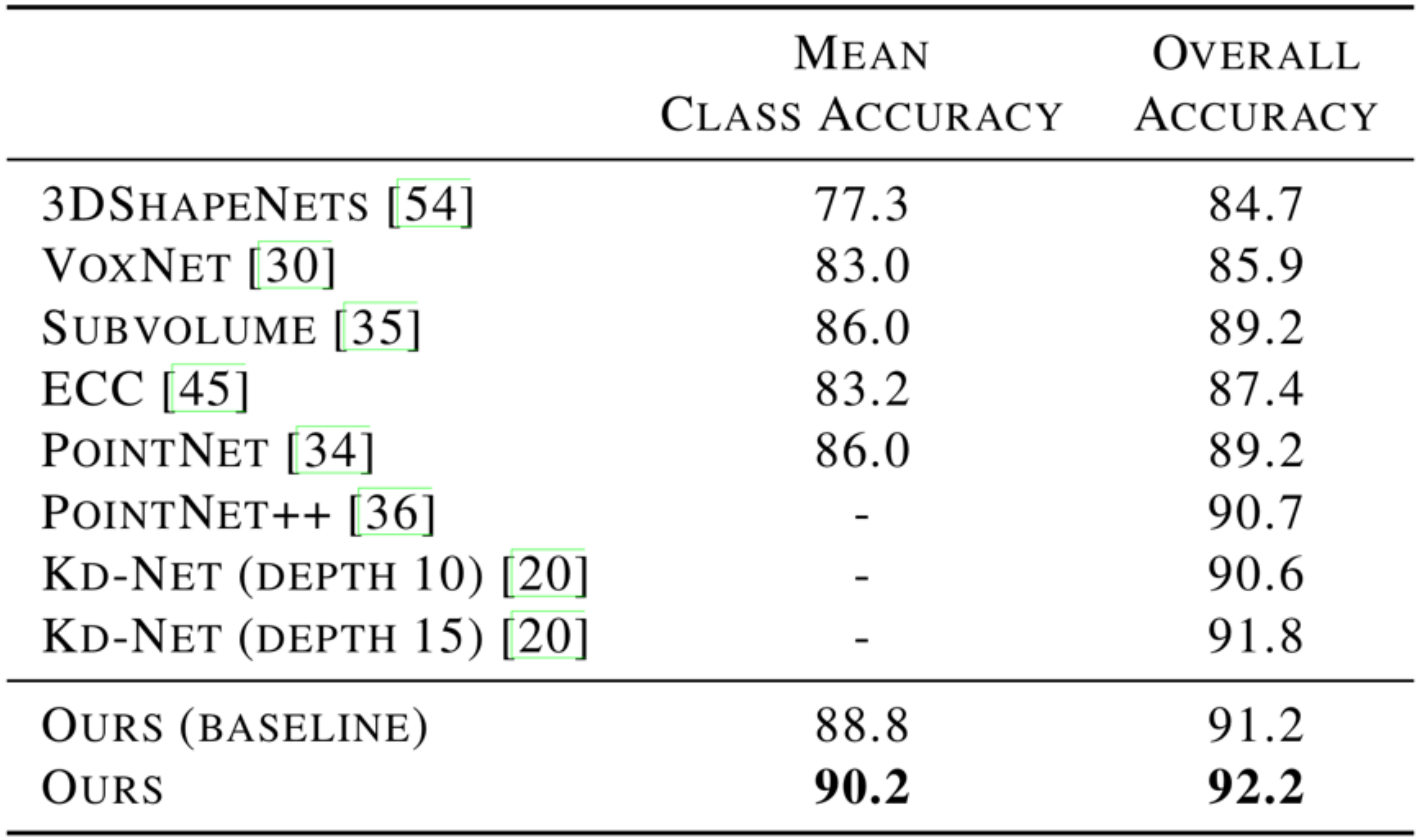
3D ShapeNets: A Deep Representation for Volumetric Shapes (2015)

2,311 meshed CAD models	9,843 train
40 categories	2,468 test

pipe
range_hood
entrance
cupboard
bridge
stool
vase
toilet
stones
building
dresser
ceiling_lamp
tube
side_table
chair
person
mantel
statue
dishwasher
frame
floor_lamp
computer
billboard
bookshelf
airplane
refrigerator
folding_door
television
brick
guitar
stone
armchair
tv_stand
pool
pan
furniture
cabin
desk_drawer
stairs
laptop
pickup
door_knob
loudspeaker
shrub
iphone
window
ham
curtain
lava
lamp
plant
bowl
sofa
cake
hat
sun
dome
boot
glass_box
helicopter
column
oven
sconce
onion
mast
fish_tank
fish
finger
shower
conference_table
mountain
wardrobe
lego
tire
staircase
cone
tank
deck
trailer
bench
aircraft
ipad
bed
monitor
fan
drawer
tunnel
pencil
chandelier
cup
shelf
clock
speaker
xbox
bottle
swimming_pool
pole
grass
ring
switch
cheese
board
windmill
vault
knobs
fence
machine
chest
ship
cell_phone
mattress
antenna
workbench
track
car
book
pot
railing
coffee_table
table
trash_can
river_water
truck
ipod
door
bow_window
wheel
knife
filing_cabinet
keyboard
rug
mirror
console_table
tree
night_stand
fireplace
plant_pot
lumber
jacuzzi
train
toy
bunk_bed
desk
handrail
flag
faucet
garage_door
door_frame
piano
sign

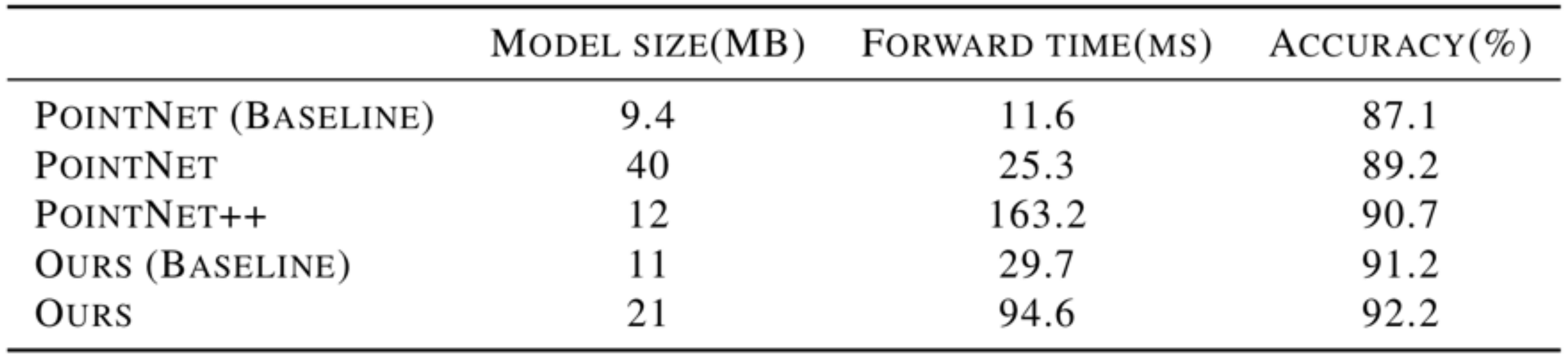
Dataset

Classification Results



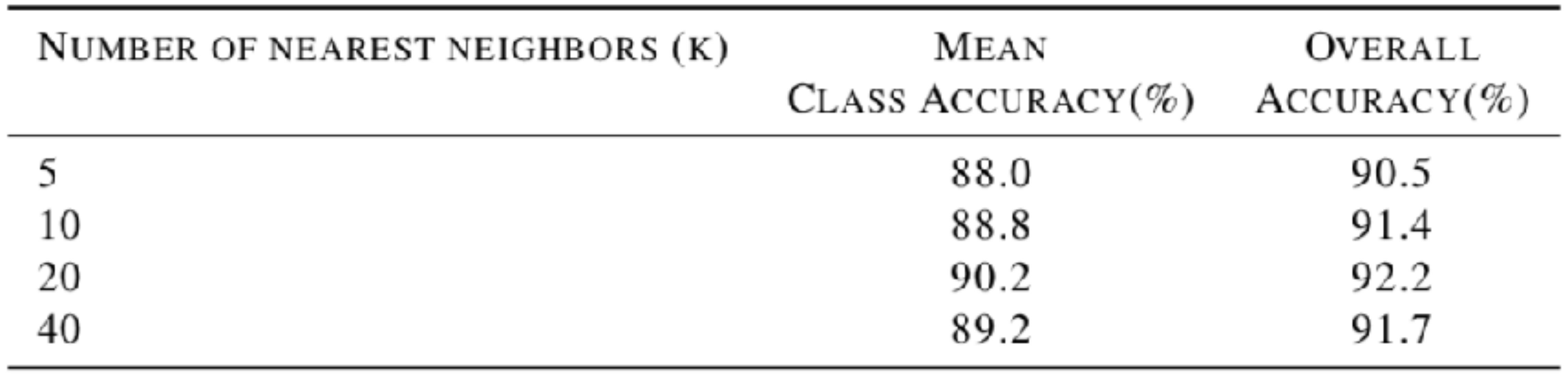
	Mean Class Accuracy	Overall Accuracy
3DShapeNets [54]	77.3	84.7
VoxNet [30]	83.0	85.9
Subvolume [35]	86.0	89.2
ECC [45]	83.2	87.4
PointNet [34]	86.0	89.2
PointNet++ [36]	-	90.7
Kd-Net (depth 10) [20]	-	90.6
Kd-Net (depth 15) [20]	-	91.8
Ours (baseline)	88.8	91.2
Ours	**90.2**	**92.2**

Table 1. Classification results on ModelNet40.



	Model size(MB)	Forward time(ms)	Accuracy(%)
PointNet (Baseline)	9.4	11.6	87.1
PointNet	40	25.3	89.2
PointNet++	12	163.2	90.7
Ours (Baseline)	11	29.7	91.2
Ours	21	94.6	92.2

Table 2. Complexity, forward time and accuracy of different models

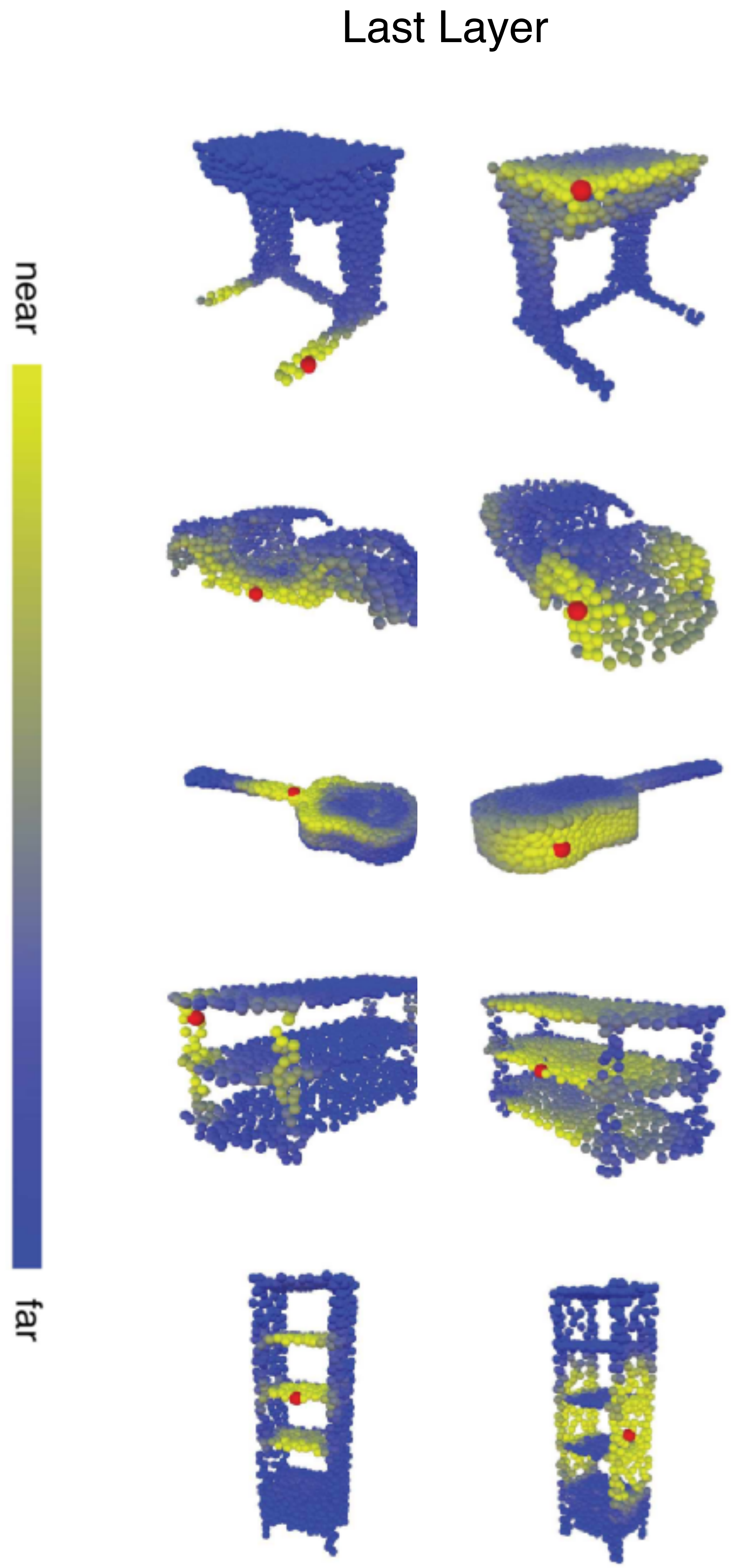


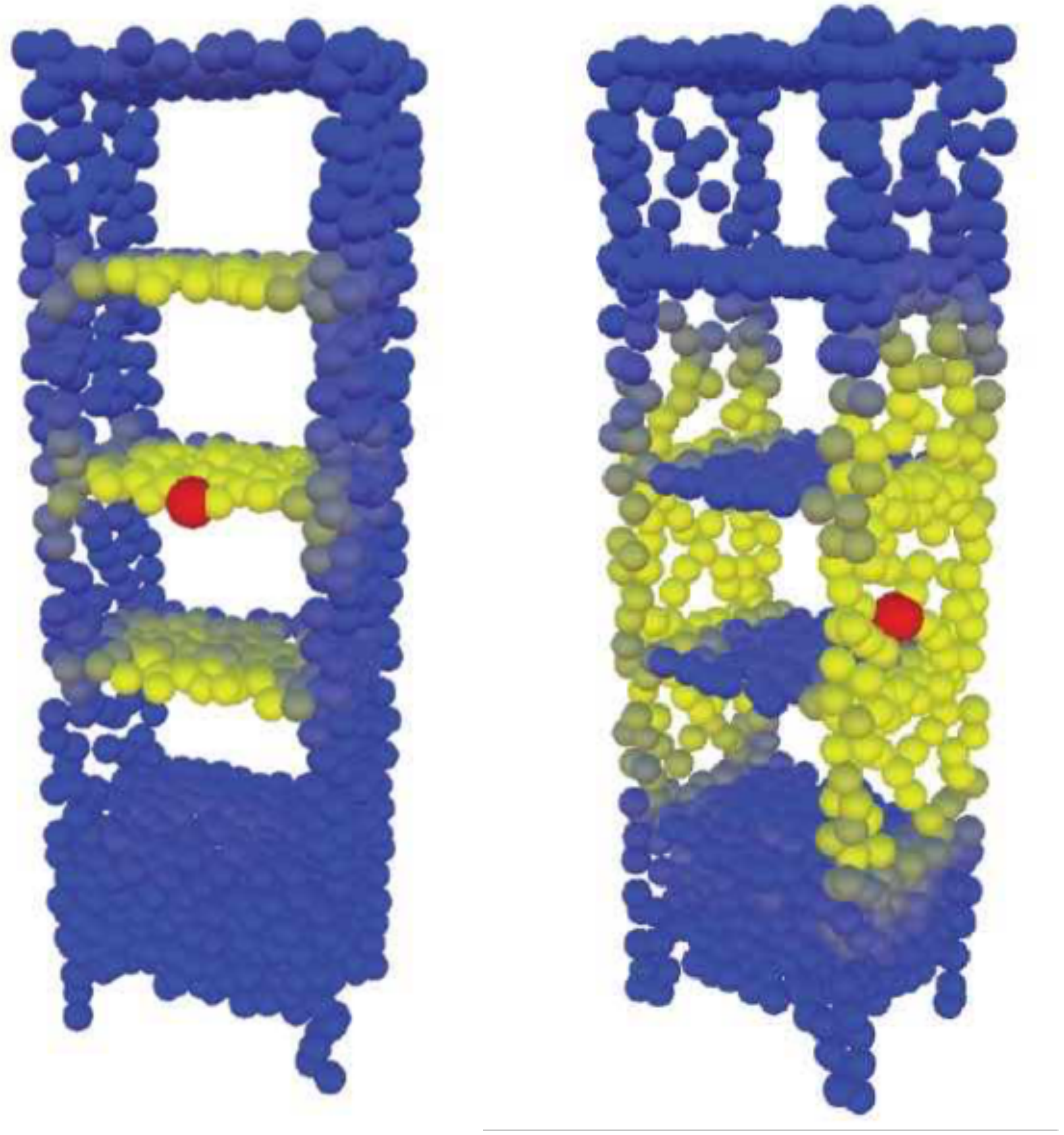
Number of nearest neighbors (k)	Mean Class Accuracy(%)	Overall Accuracy(%)
5	88.0	90.5
10	88.8	91.4
20	90.2	92.2
40	89.2	91.7

Table 4. Results of our model with different numbers of nearest neighbors.

Dataset

Features space



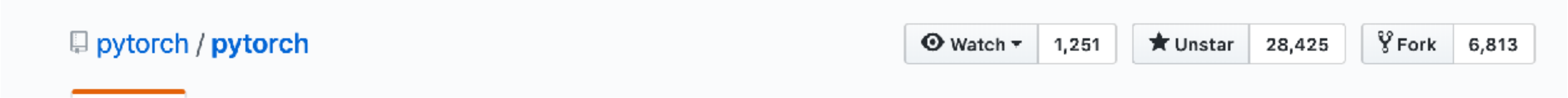


Results

Implementation

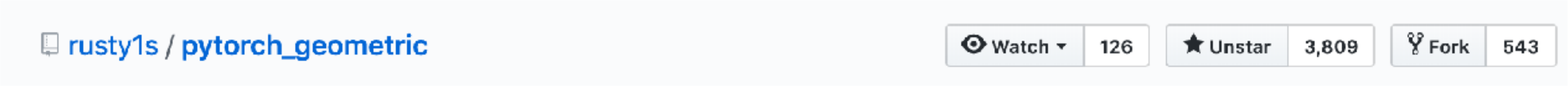
Tools

PyTorch



pytorch / pytorch

PyTorch geometric

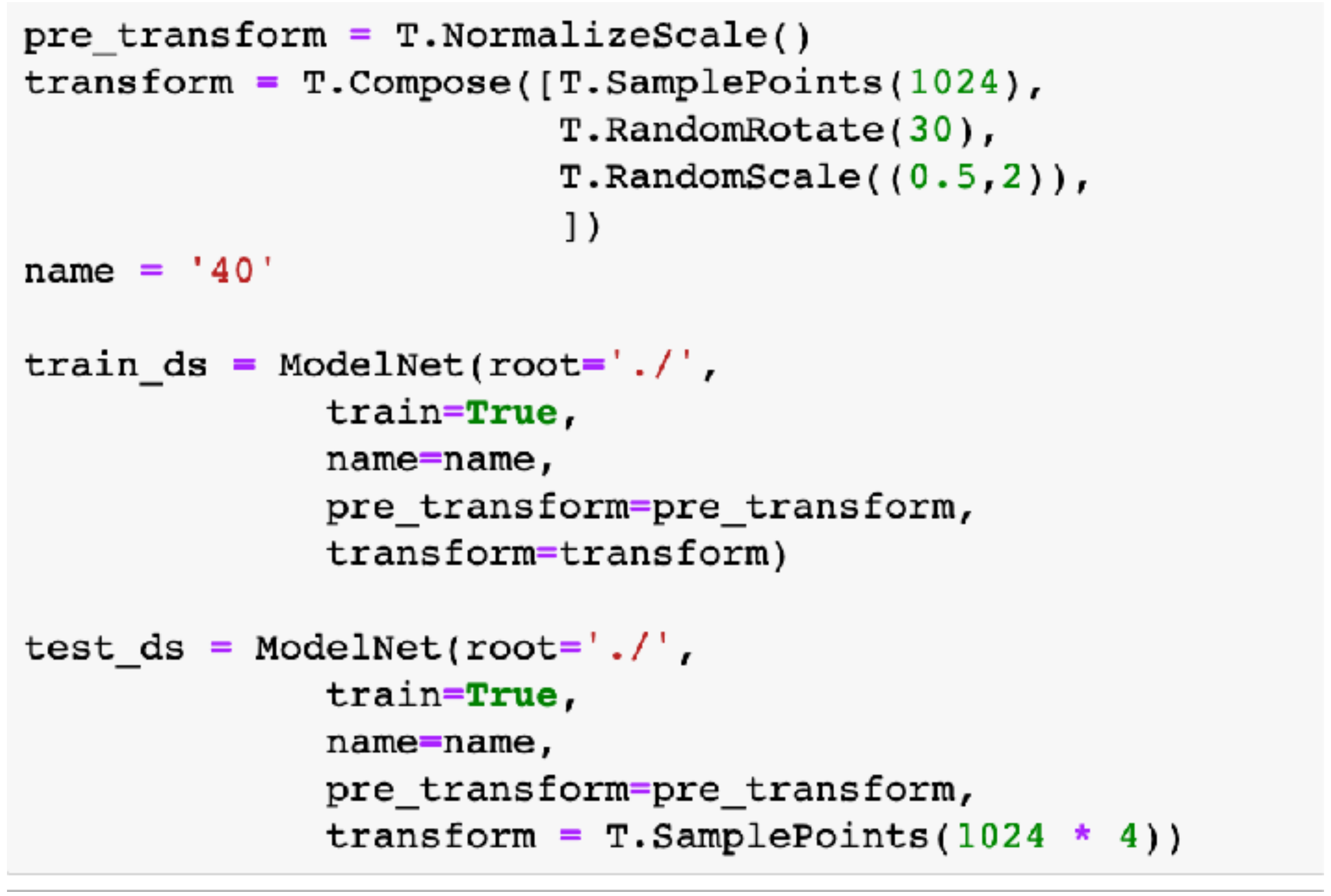


rusty1s / pytorch_geometric

Matthias Fey

Implementation

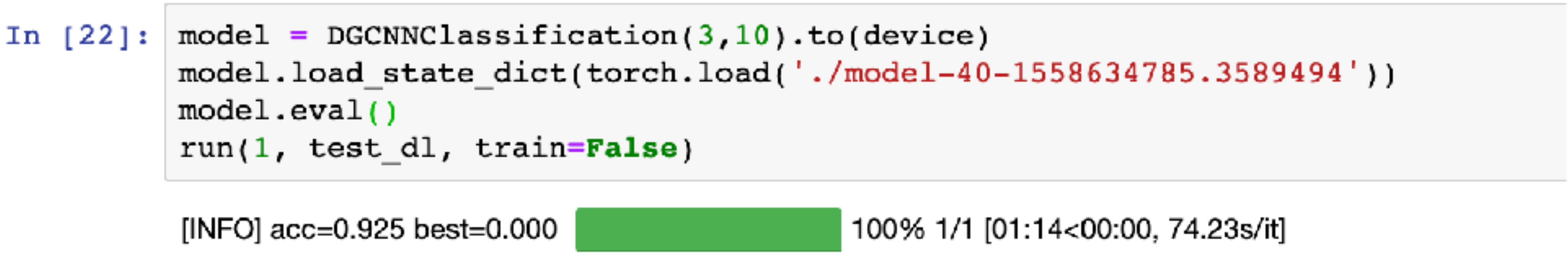
Early results



```
pre_transform = T.NormalizeScale()
transform = T.Compose([T.SamplePoints(1024),
                       T.RandomRotate(30),
                       T.RandomScale((0.5,2)),
                       ])
name = '40'

train_ds = ModelNet(root='./',
             train=True,
             name=name,
             pre_transform=pre_transform,
             transform=transform)

test_ds = ModelNet(root='./',
             train=True,
             name=name,
             pre_transform=pre_transform,
             transform = T.SamplePoints(1024 * 4))
```



```
In [22]: model = DGCNNClassification(3,10).to(device)
         model.load_state_dict(torch.load('./model-40-1558634785.3589494'))
         model.eval()
         run(1, test_dl, train=False)

[INFO] acc=0.925 best=0.000 100% 1/1 [01:14<00:00, 74.23s/it]
```

https://github.com/FrancescoSaverioZuppichini/GDL-project

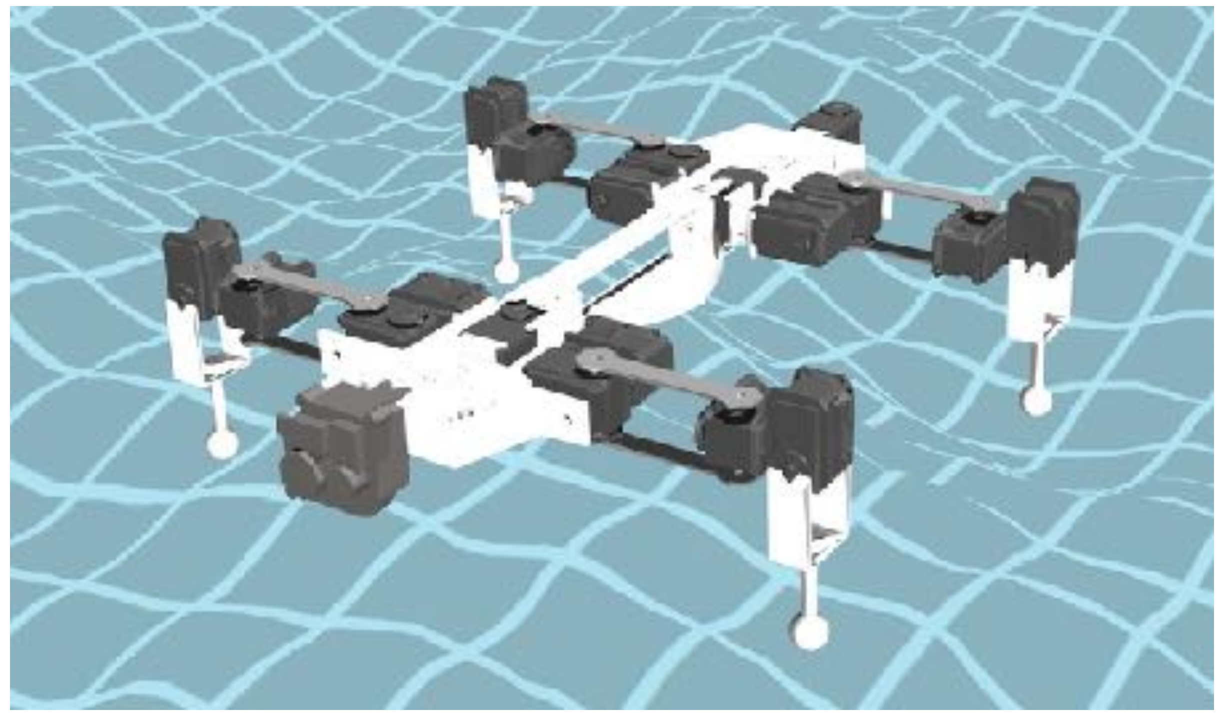
Extension

Extension

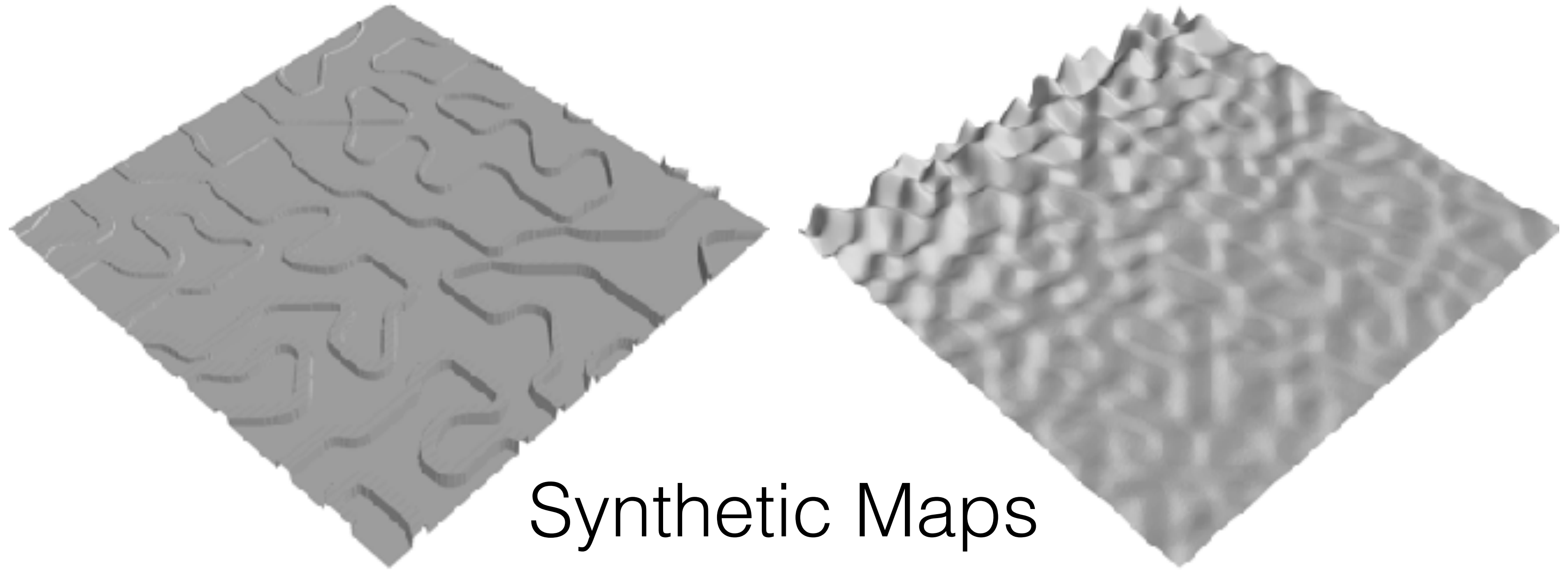
Comparing DGCNN with a CNN on my Master Thesis Dataset composed by 470k images

Extension

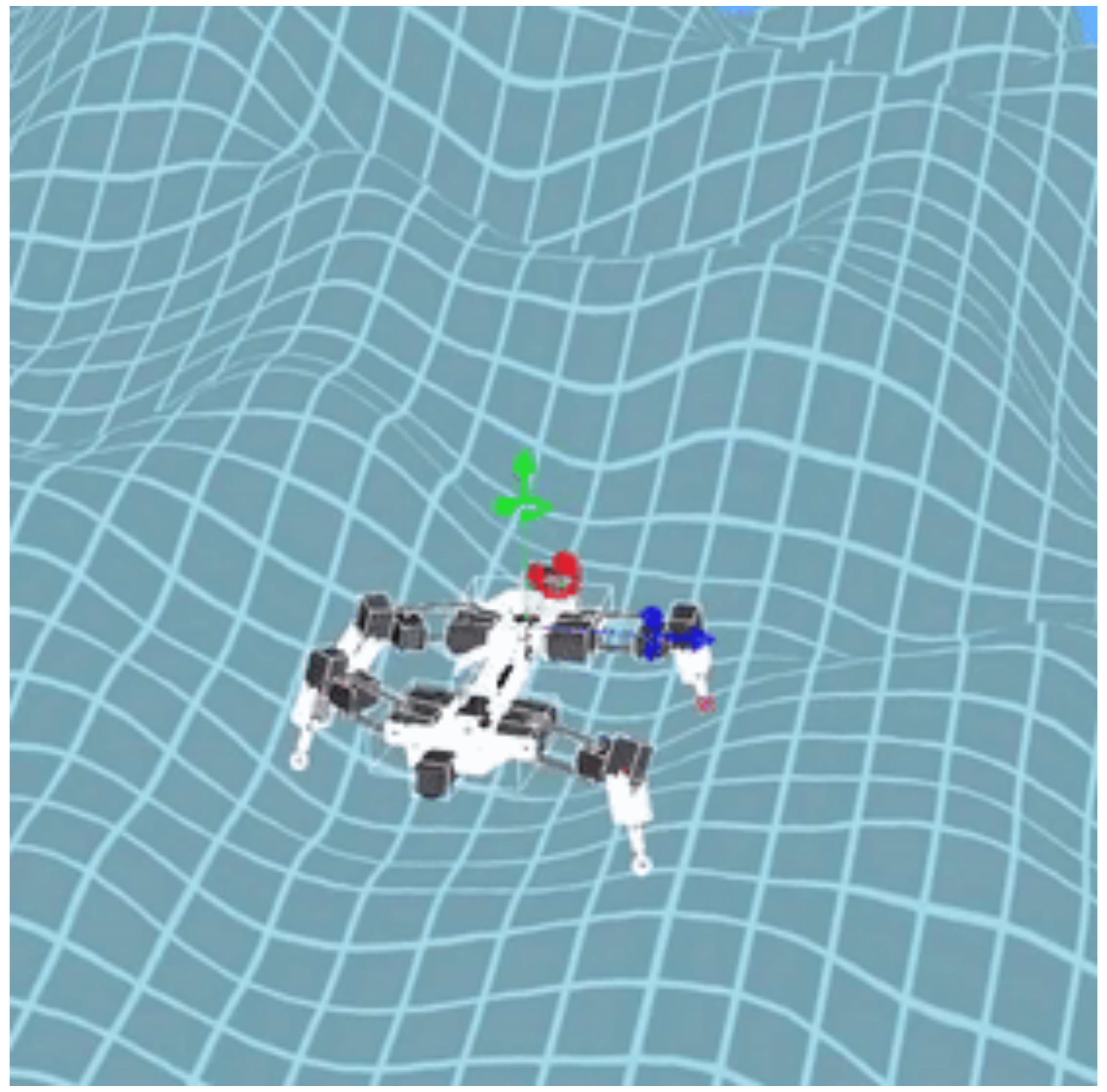
Dataset generation



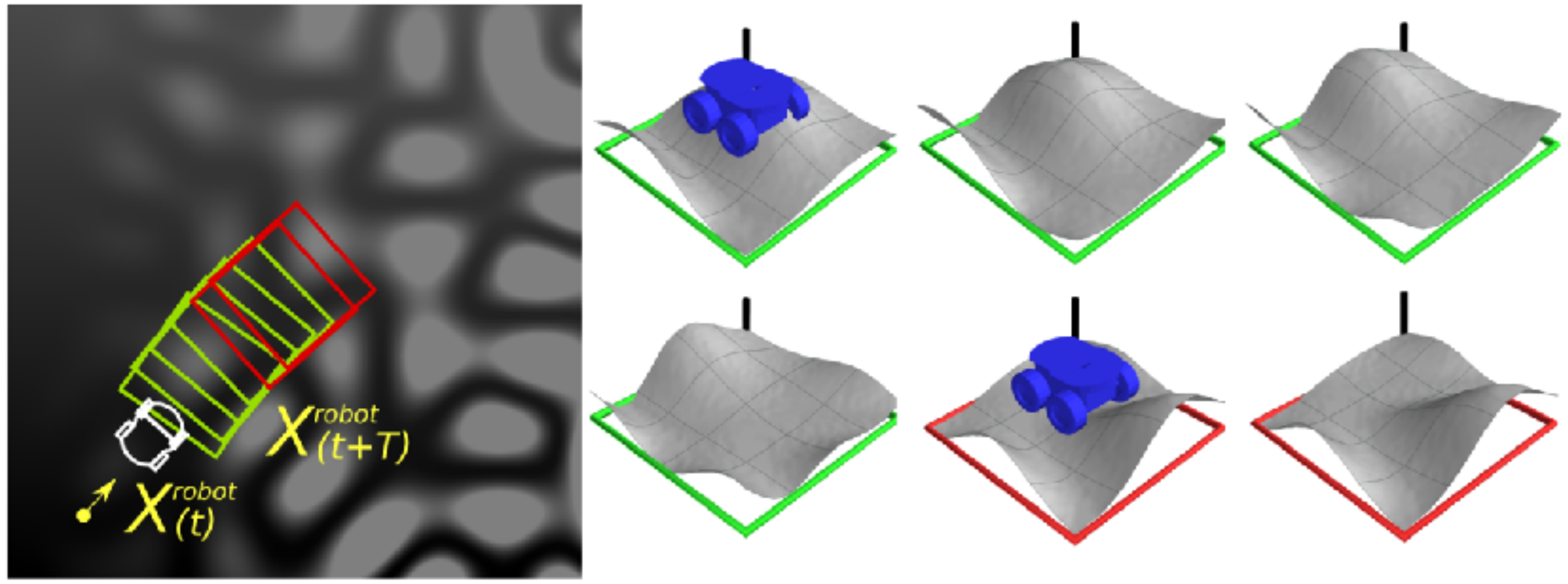
My loved Robot called Krock



Synthetic Maps



run the robot in the simulator

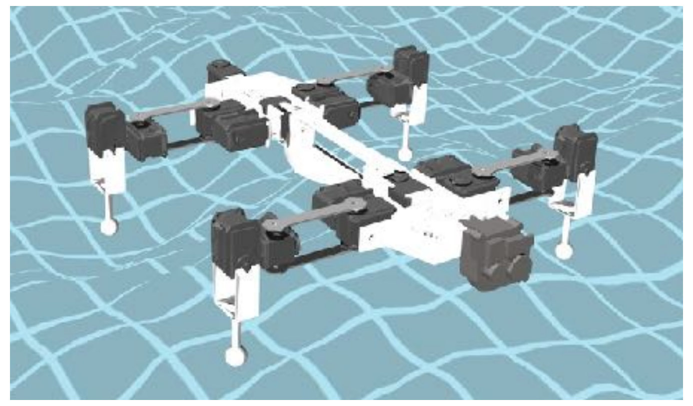


Crop a patch around the robot and label it

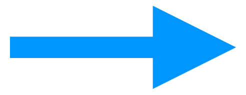
Extension

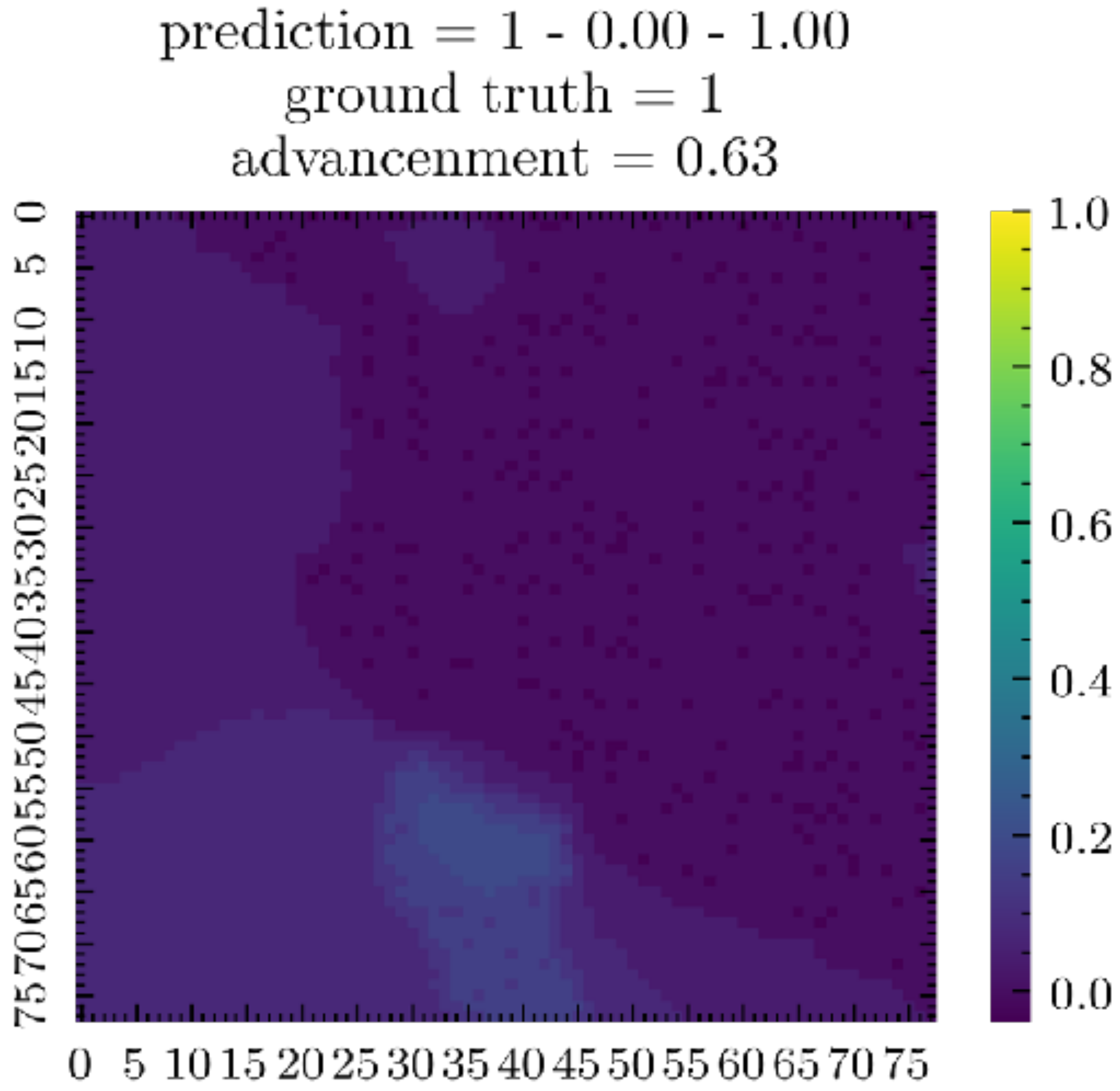
Patch example

Each patch is a 79x79 gray image, where each pixel represent the height in meters.

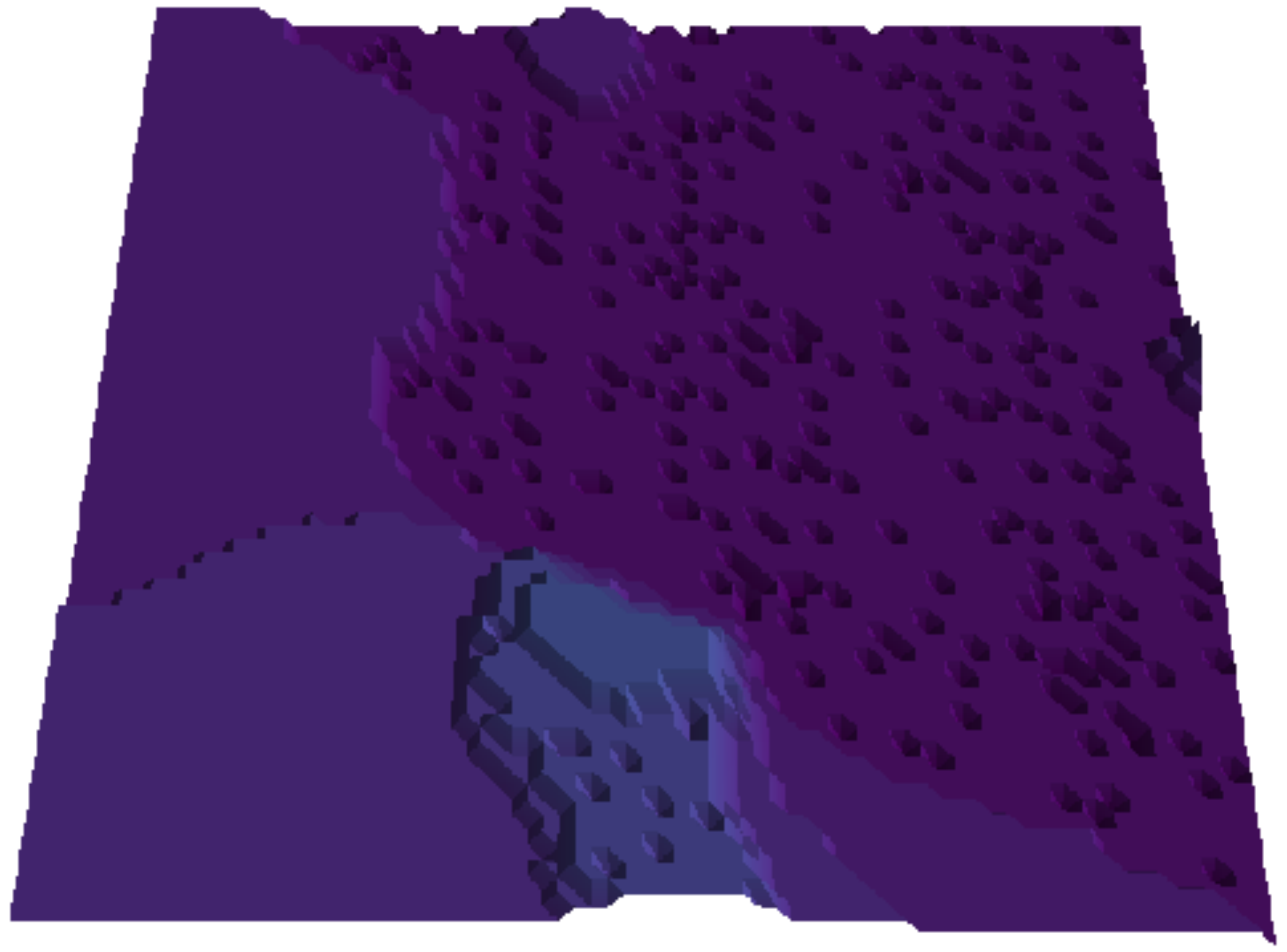


Robot traverse from left to right





example of an data sample

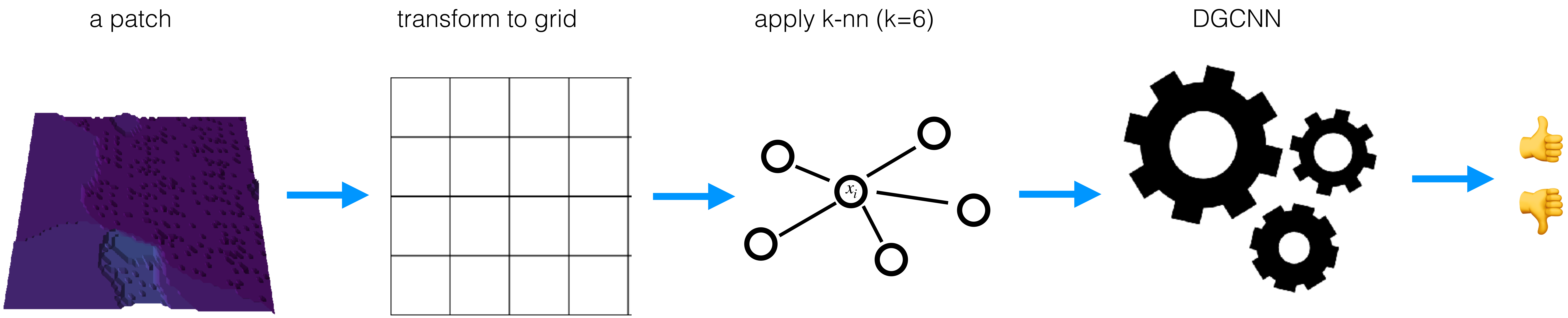


3d render

Extension

Pipeline

To feed my images into DGCNN we have to convert them to **graphs**



a patch

transform to grid

apply k-nn (k=6)

DGCNN

Thank you for your attention

Francesco Saverio Zuppichini

Thanks to
Roberto Falcone (template)
Luca Morreale (hardware)
Eynard Davide (hardware)
Dario Mantegazza & Alessia Ruggeri (moral support)

Photo by Scott Webb on Unsplash

Title

Subtitle

item

item

item