

LiDAR point classification of power line facilities using the deep neural network PointNet++

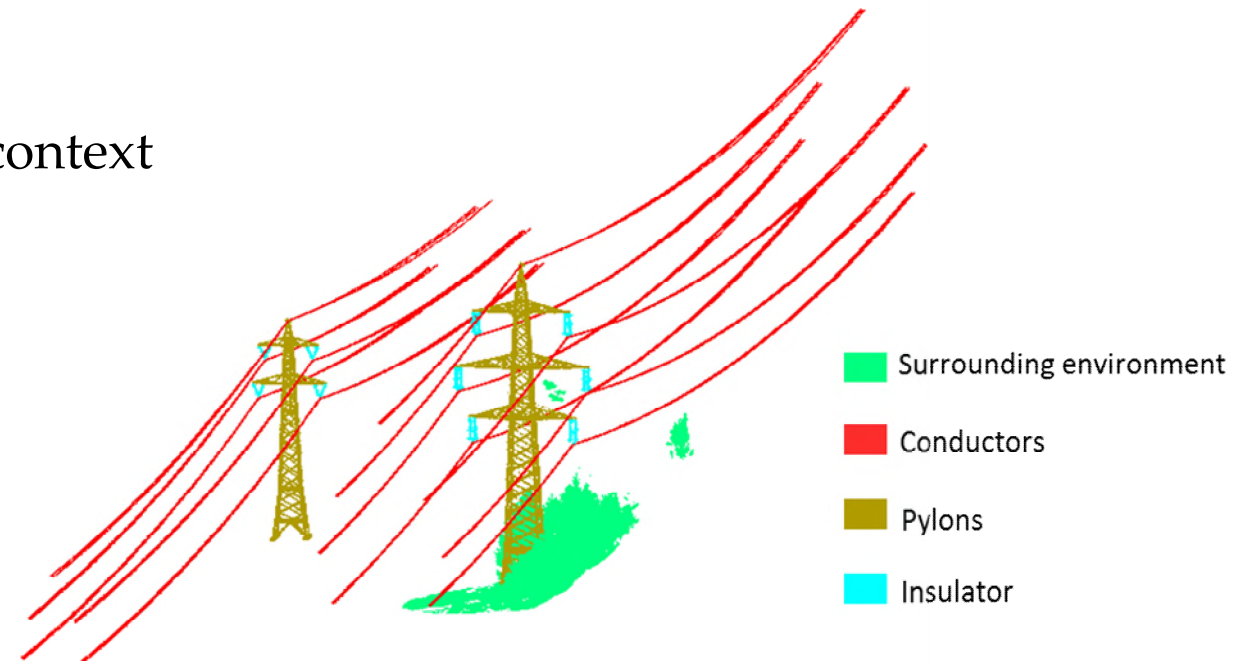
Nan Li Department of Geodesy and Geoinformation, Technische Universität Wien
Olaf Kaehler Research Group Active Vision Technologies, Siemens AG Österreich
Norbert Pfeifer Department of Geodesy and Geoinformation, Technische Universität Wien

Aim

Develop deep neural network learning strategy to achieve good classification of power line facilities without handcrafted features

Challenges

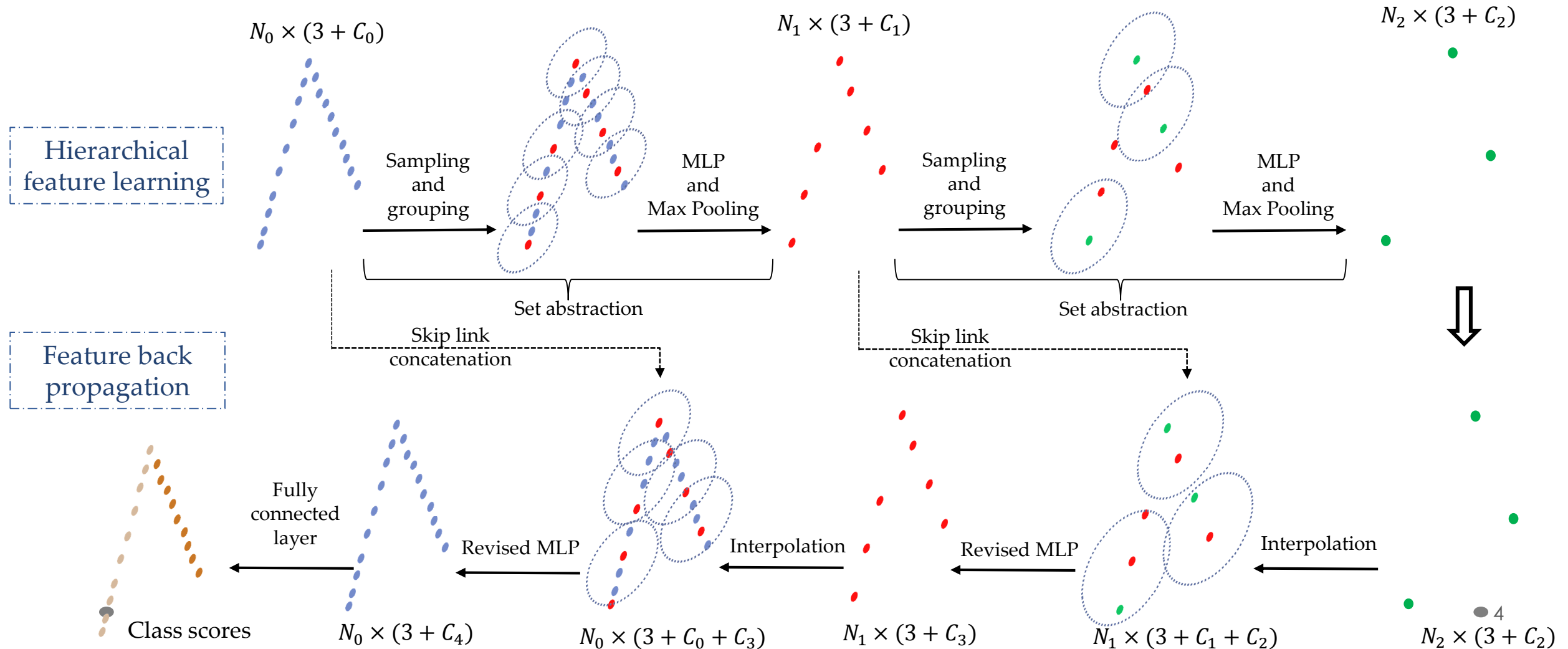
- Classes defined by function in application context (compare LC / LU discussion)
- Rare class / class imbalance
- Density variation within data set / between data sets
- Large data volume



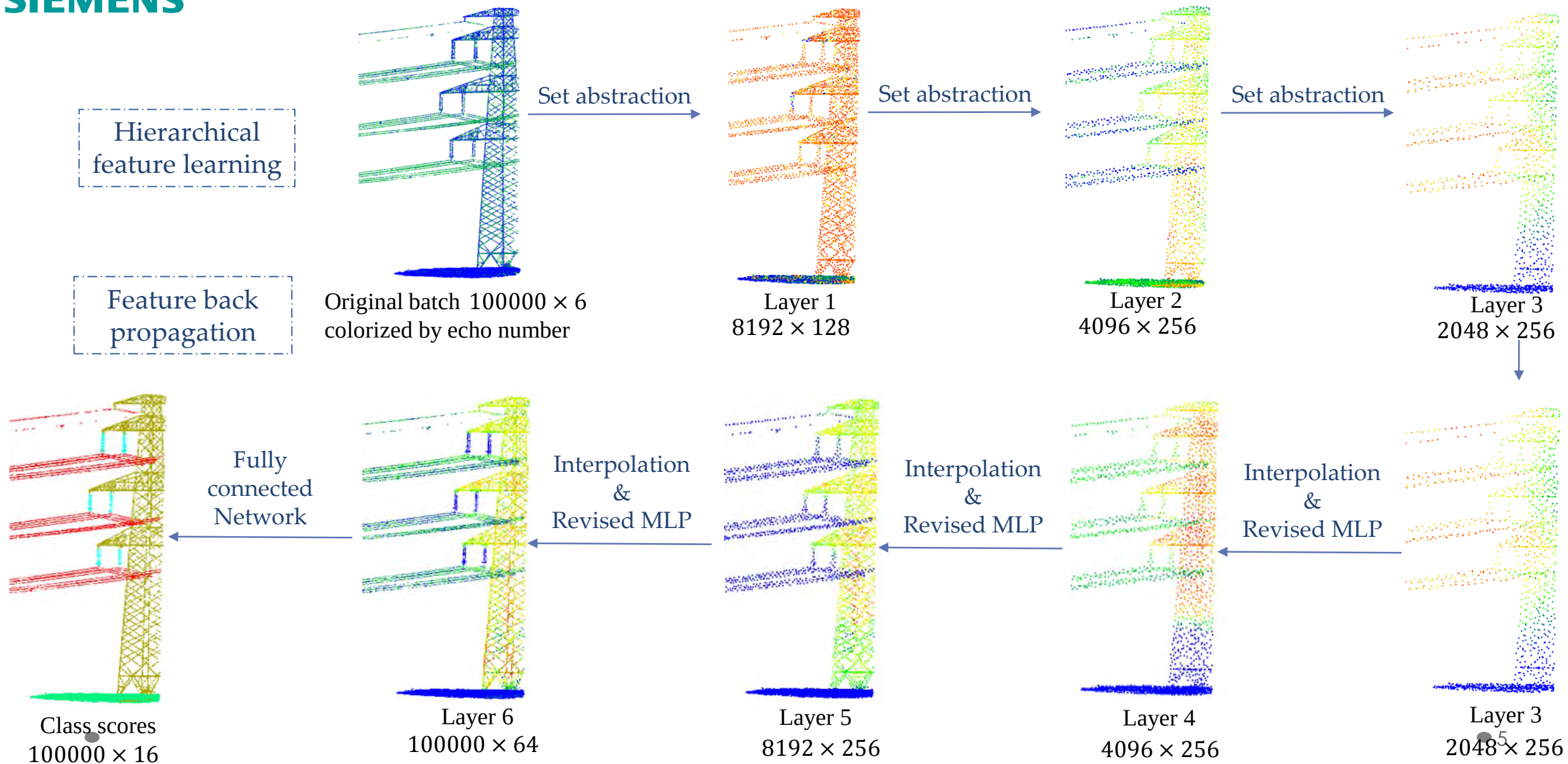
Outline

- The Architecture of PointNet++
- Classification results by PointNet++
- Investigations on parameters and PointNet++
- Comparison with PointCNN
- Conclusions

The Architecture of PointNet++



The Architecture of PointNet++



Network Training Configuration

--- semantic segmentation of power line facilities

Base configuration

- Class balancing
- 391 batches are used for training, 100000 points per batch
- Input attributes: X,Y,Z, number of return, return number, intensity

SA layers setting

SA layers	Number of sampled points	Neighbors searching radius	Number of nearest Neighbors	MLP layer	Feature aggregation
Layer 1	8192	1	16	[64,64,128]	Max pooling
Layer 2	4096	5	64	[128,128,256]	Max pooling
Layer 3	2048	15	64	[128,128,256]	Max pooling

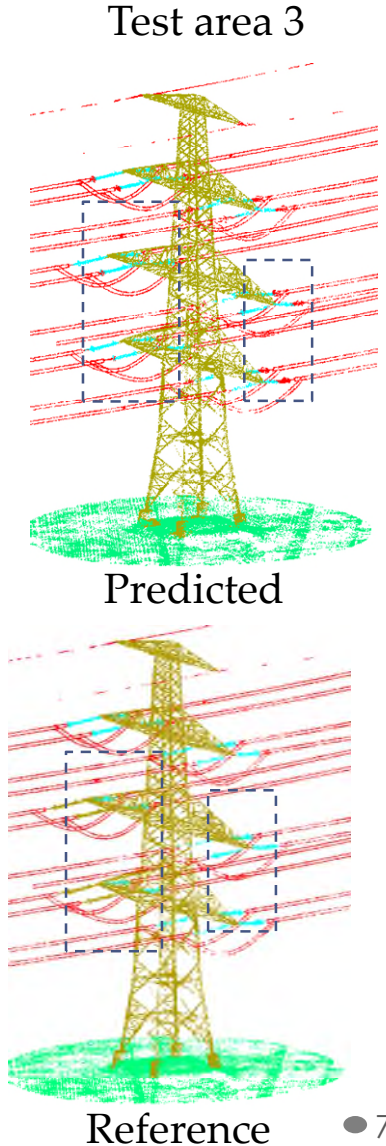
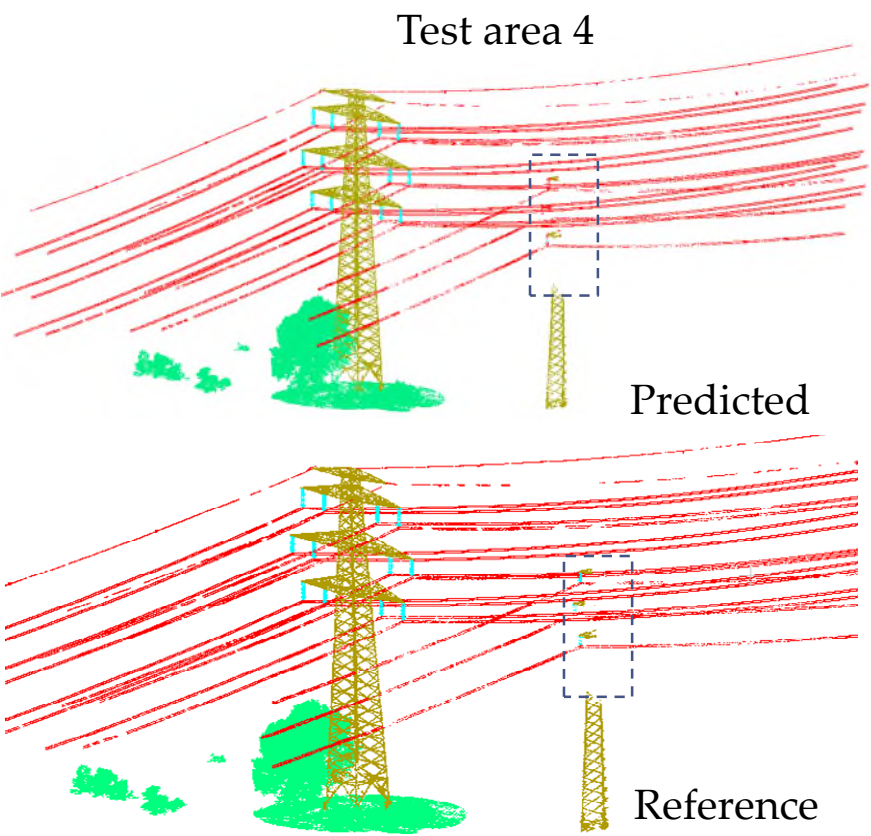
Hyper-parameters setting

Parameters	Value
Batch size	1
Training epoch	10
Initial learning rate	0.001
Decay rate	0.7
Decay step	500

Classification results by PointNet++

- Trained model is applied on 4 different areas (one campaign)
- Use recall as accuracy metric

- Surrounding environment
- Pylons
- Conductors
- Insulators



Recalls

Test data	Enviroment (%)	Conductors (%)	Pylons (%)	Insulator (%)	Average (%)
Test area 1	98.22	99.86	99.94	91.67	97.42
Test area 2	94.44	99.87	99.98	94.24	97.13
Test area 3	95.79	99.96	95.34	71.53	90.65
Test area 4	99.92	99.89	96.00	96.80	98.15
Average	97.09	99.89	97.82	88.65	95.84

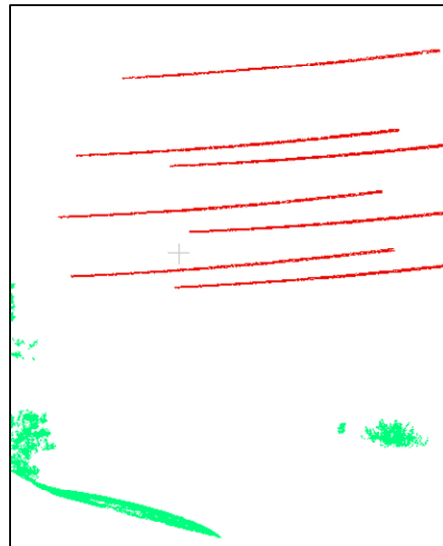
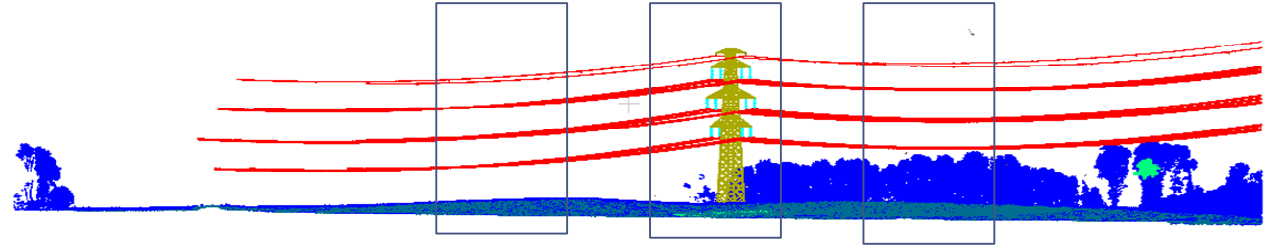
Investigations on Training Configuration + Network

- Training strategy
 - Class balancing strategy
 - Input features
- Hyper-parameters investigation
 - Learning rate
- Ability of generalization
 - Fine-tuning on datasets with different density

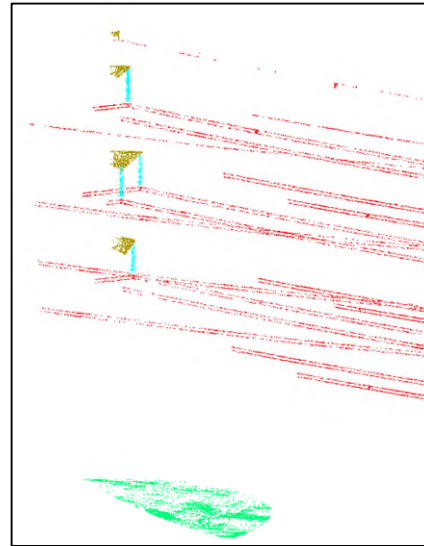
Class balancing strategy

- Data preparation:
cutting into batches
- Batch selection

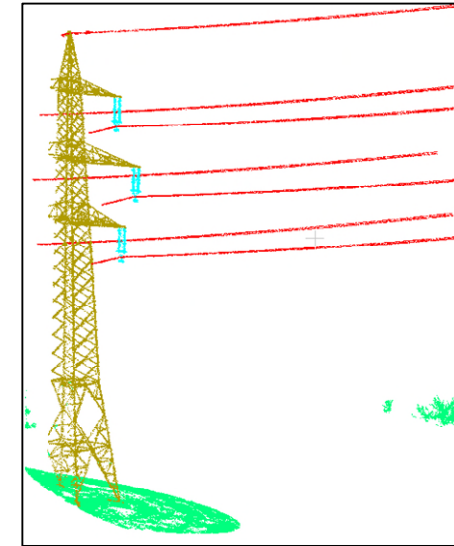
- o Class divergence $> \text{threshold}$: classes evenly distributed in the batch, only includes Ex3.
- o Class divergence > 0 : all classes appear in the batch, includes Ex1 and Ex3.
- o Class divergence > -1 : no selection, includes Ex1, Ex2, Ex3.



Ex1



Ex2



Ex3

Class balancing strategy

- Balance of rare class should be considered
- Ensure diversity of objects

Recalls and Precisions

Batch selection		Environment	Conductors	Pylons	Insulators	Average
Recall	Class divergence $\geq$ threshold	99,06%	99,84%	95,55%	75,13%	92,40%
	Class divergence ≥ 0	98,93%	99,73%	96,64%	88,30%	95,90%
	Class divergence ≥ -1	88,65%	99,60%	98,85%	74,47%	90,39%
Precision	Class divergence $\geq$ threshold	98,32%	99,37%	83,87%	98,39%	95,43%
	Class divergence ≥ 0	98,60%	99,10%	95,92%	90,10%	95,93%
	Class divergence ≥ -1	98,23%	98,37%	94,90%	90,22%	91,70%

Input features

- using X,Y,Z, number of return, return number, intensity
- only using X,Y,Z

Dataset A: used for previous experiments

Dataset B: collected from different campaign

Dataset A – Recalls

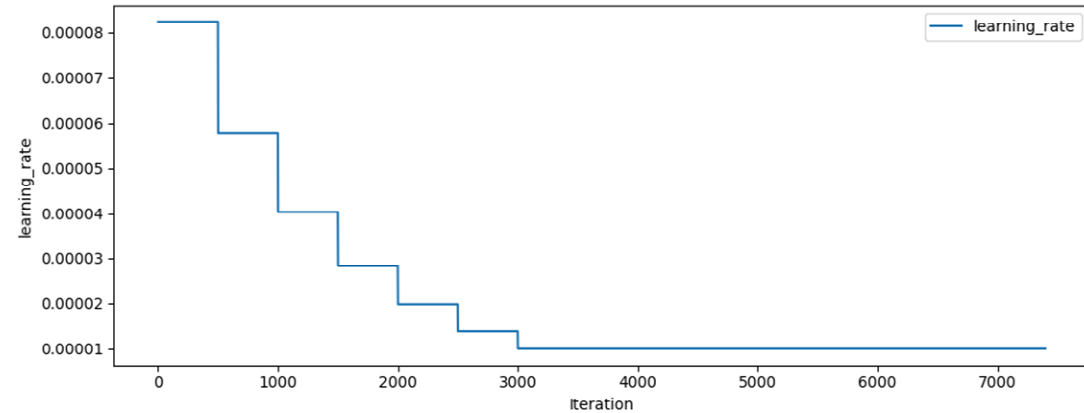
Features	Environment	Conductors	Pylons	Insulators	Average
X,Y,Z, number of return, return number, intensity	97,09%	99,89%	97,82%	88,65%	95,84%
X,Y,Z	99,92%	99,91%	95,95%	96,32%	98,02%

Dataset B -- Recalls

Features	Environment	Conductors	Pylons	Insulators	Average
X,Y,Z, number of return, return number, intensity	96,37%	99,32%	91,72%	71,06%	89,62%
X,Y,Z	92,90%	94,28%	95,06%	24,67%	76,73%

Learning rate investigation

- Constant learning rate
- Decay learning rate



Dataset A: mean **recall** for each class over all test tiles is summarized:

LR	Decay	Environments	Conductors	Pylons	Insulators	Average
0.0001	No decay	98,82%	99,83%	96,62%	85,75%	95,26%
0.001	Rate=0,8	90,17%	99,82%	98,91%	86,92%	93,95%
0.001	Rate=0,7	97,09%	99,89%	97,82%	88,65%	95,84%
0.001	Rate=0,6	96,53%	99,87%	98,71%	85,43%	95,14%
0.001	Rate=0,5	99,97%	99,88%	96,64%	86,93%	95,85%
0.001	Rate=0,2	99,06%	99,79%	96,95%	85,57%	95,34%

Learning rate investigation

- Constant learning rate
- Decay learning rate

Dataset B: mean **recall** for each class over all test tiles:

LR	Decay	Environments	Conductors	Pylons	Insulators	Average
0.0001	No decay	75.40%	79.87%	81.37%	74.95%	77,90%
0.001	Rate=0,8	96,60%	98,10%	93,89%	73,53%	89,65%
0.001	Rate=0,7	96,37%	99,32%	91,72%	71,06%	89,62%
0.001	Rate=0,5	96,38%	98,85%	93,88%	73,14%	88,06%

The ability of generalization

- Input dropout:
Randomly dropping out points that fed into network during training
- Multi-scale grouping (MSG) :
 - Apply grouping layers with different scales (different K neighbors)
 - Features at different scales are concatenated to a multi-scale feature
- Fine-tuning
Fine-tune exiting networks' weights by continue training on target datasets:
 - be adapted to the new dataset, or ultimately to all datasets
 - Speed up training

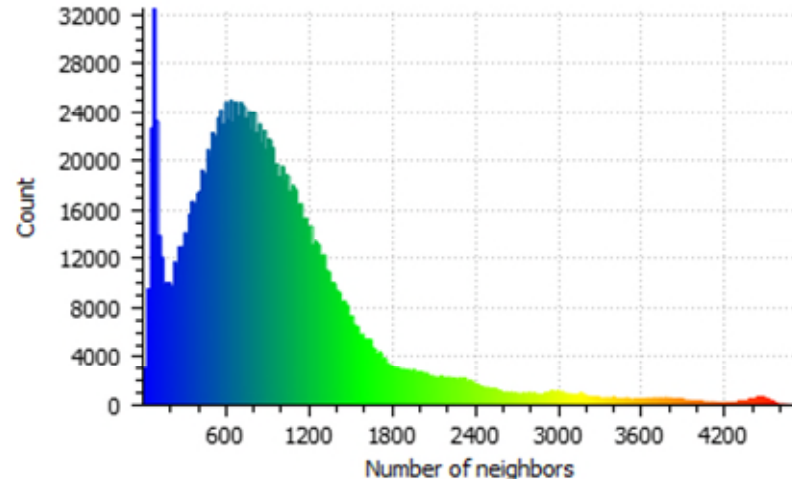
Fine-tuning on different datasets

Can the network be applied to datasets collected from different campaigns?

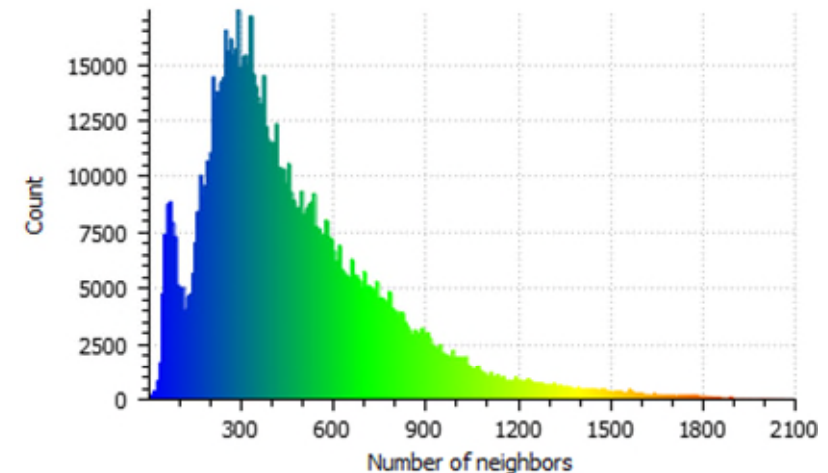
Experiments:

- 1, Directly applied, no further fine-tuning
- 2, Base network pre-trained on Dataset A, fine-tuning on Dataset B
- 3, Base network pre-trained on Dataset A, fine-tuning on Dataset A and Dataset B
- 4, Training on mixed datasets of Dataset A and Dataset B from scratch

Point number within
the radius of 1m



Dataset A : average 600 pts



Dataset B: average 400 pts

Fine-tuning results

1, No fine-tuning

Base model	Fine_tune	Test data	Environments	Conductors	Pylons	Insulators	Average
Dataset A	None	Dataset A	98,93%	99,73%	96,64%	88,30%	95,90%
		Dataset B	79,65%	94,29%	92,96%	34,76%	75,39%

2, Only fine-tuning on Dataset B

Base model	Fine_tune	Test data	Environments	Conductors	Pylons	Insulators	Average
Dataset A	Dataset B	Dataset A	99,39%	99,51%	87,61%	63,28%	87,45%
		Dataset B	97,47%	99,66%	89,16%	54,96%	85,31%

3, Fine-tuning on both Dataset A and Dataset B

Base model	Fine_tune	Test data	Environments	Conductors	Pylons	Insulators	Average
Dataset A	Dataset A + Dataset B	Dataset A	98,48%	99,84%	98,53%	79,96%	94,20%
		Dataset B	92,25%	98,79%	92,44%	67,29%	87,69%

4, Training on mixed datasets of Dataset A and Dataset B from scratch

Base model	Fine_tune	Test data	Environments	Conductors	Pylons	Insulators	Average
Dataset A + Dataset B	None	Dataset A	96.47%	98.34%	93.29%	36.88%	81.25%
		Dataset B	96.77%	93.85%	88.27%	27.46%	76.64%

Recalls

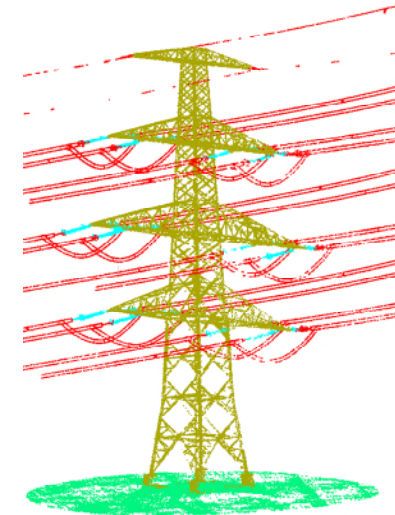
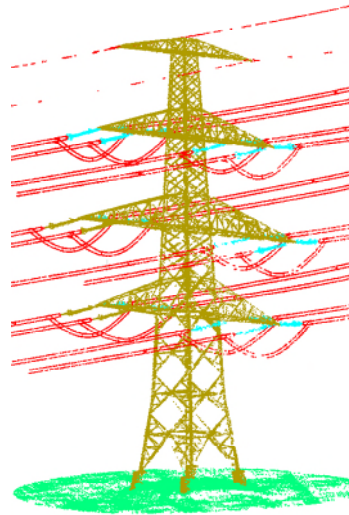
Fine-tuning results

Fine-tuning on both Dataset A and Dataset B

Predicted

Reference

Test area 3



Dataset A:

Test area 4

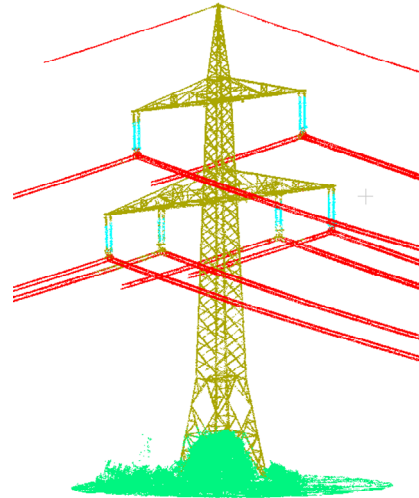


Fine-tuning results

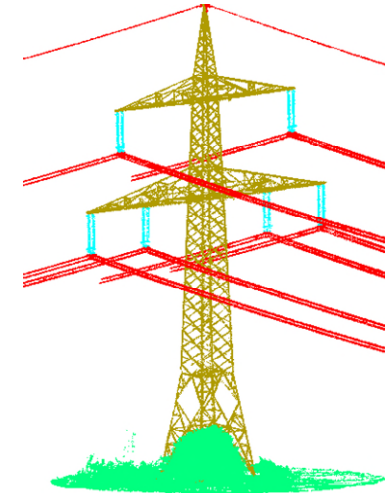
Fine-tuning on both Dataset A and Dataset B

Test area 3

Predicted

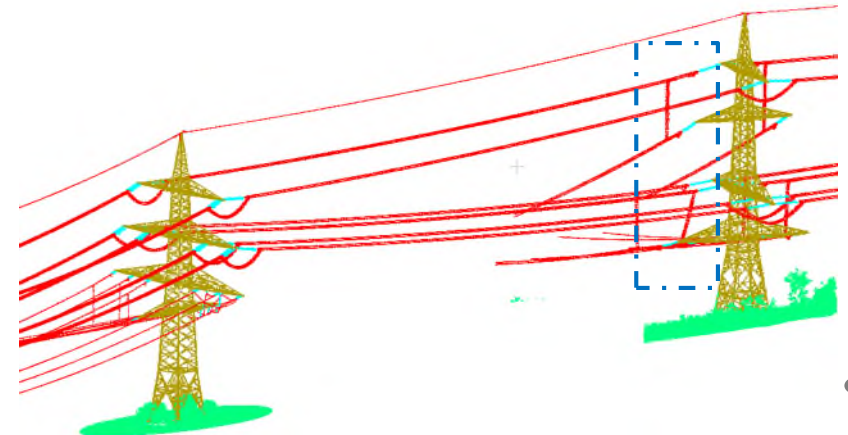
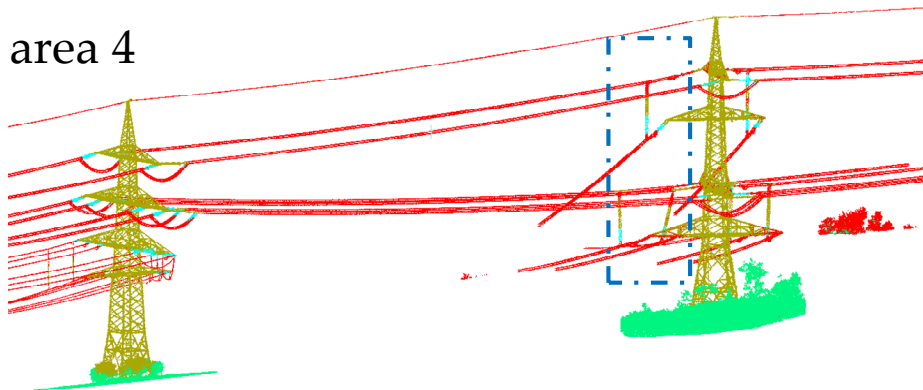


Reference



Dataset B:

Test area 4

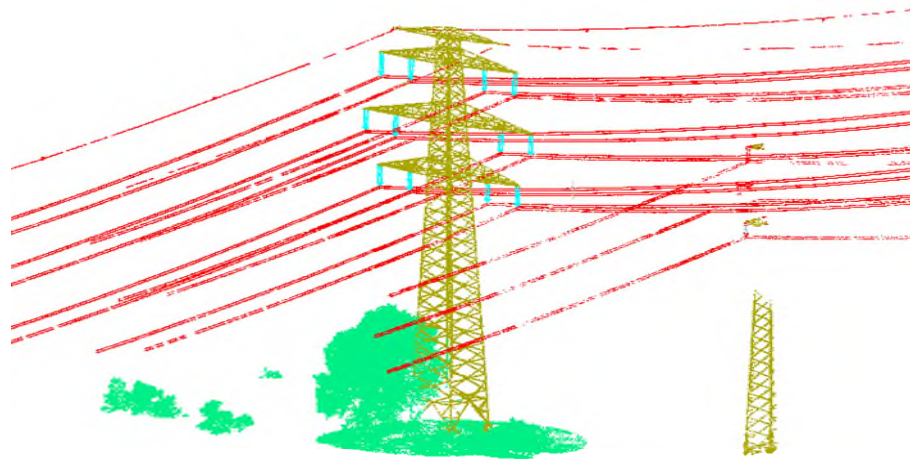


Comparison with PointCNN

- PointNet++ : Learn features via MLP
- PointCNN : Learn a transformation via MLP
 - Simultaneously weight and permute the input features

Recalls

Test Network	Enviroment (%)	Conductors (%)	Pylons (%)	Insulator (%)	Average (%)
PointNet++	97,09	99,89	97,82	88,65	95,84
PointCNN	92,32	99,13	95,88	83,91	92,81



PointNet++



PointCNN

Conclusions

- PointNet++ :
 - appropriate method for semantic segmentation of power line facilities
- Class imbalance
 - Consideration of class divergence improves results
 - Ensuring all classes appear in each training batch lead to best results
- Using LiDAR features (echo ID, nb. of echoes, Intensity)
 - LiDAR features increase robustness of results
- Learning rate
 - Decay of learning rate improves results, but no optimal decay rate for all examples
- Fine-tuning for generalization
 - Best results when fine tuning on old + new data
- PointCNN vs. PointNet++
 - PointNet++ seems better suited, but effort/time spent on PointNet++ was much bigger than for PointCNN

Welcome your comments!