JURSE 2019
VANNES-FRANCE

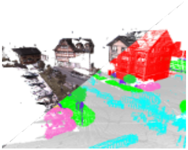
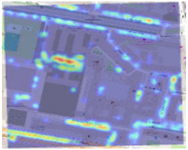
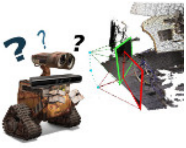
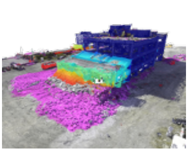
Deep Learning for Remote Sensing Tutorial

Presentation outline

Presentation Layout

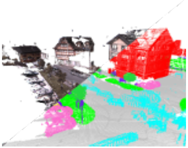
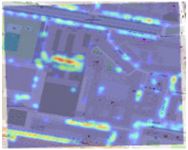
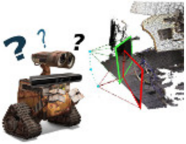
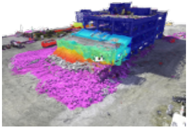
- Researcher at ONERA (French Aerospace Laboratory), Univ. Paris Saclay
- **Focus:** deep learning for remote sensing, robotics and perception

- Researcher at ONERA (French Aerospace Laboratory), Univ. Paris Saclay
- **Focus:** deep learning for remote sensing, robotics and perception
- Chair of IEEE GRSS Tech. Committee on Image Analysis and Data Fusion: http://www.grss-ieee.org/community/technical-committees/data-fusion/
- Co-organizer of the Data Fusion Contests since 2016: http://www.grss-ieee.org/community/technical-committees/data-fusion/2019-ieee-grss-data-fusion-contest/

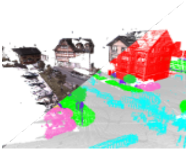
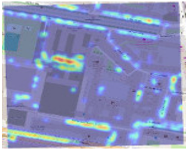
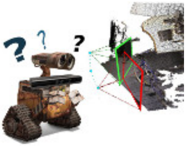
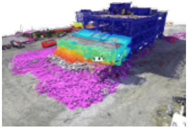
Bertrand Le Saux

- Researcher at ONERA (French Aerospace Laboratory), Univ. Paris Saclay
- **Focus:** deep learning for remote sensing, robotics and perception
- Chair of IEEE GRSS Tech. Committee on Image Analysis and Data Fusion: http://www.grss-ieee.org/community/technical-committees/data-fusion/
- Co-organizer of the Data Fusion Contests since 2016: http://www.grss-ieee.org/community/technical-committees/data-fusion/2019-ieee-grss-data-fusion-contest/
- CVPR/EarthVision 2019 co-organizer: https://www.grss-ieee.org/earthvision2019/

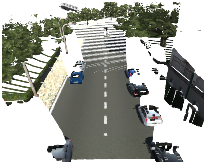
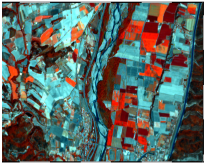
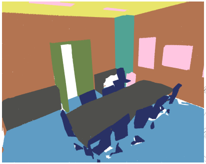
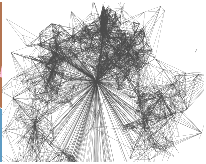
Bertrand Le Saux

- Researcher at ONERA (French Aerospace Laboratory), Univ. Paris Saclay
- **Focus:** deep learning for remote sensing, robotics and perception
- Chair of IEEE GRSS Tech. Committee on Image Analysis and Data Fusion: http://www.grss-ieee.org/community/technical-committees/data-fusion/
- Co-organizer of the Data Fusion Contests since 2016: http://www.grss-ieee.org/community/technical-committees/data-fusion/2019-ieee-grss-data-fusion-contest/
- CVPR/EarthVision 2019 co-organizer: https://www.grss-ieee.org/earthvision2019/
- Web: https://blesaux.github.io/

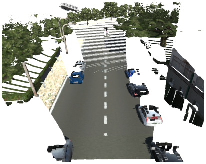
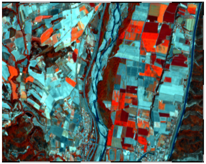
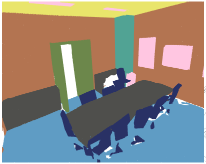
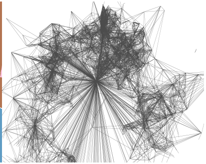
Loic Landrieu

- Researcher at the French Mapping Agency in the machine learning department STRUDEL.

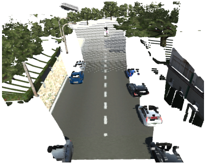
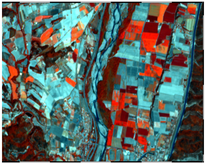
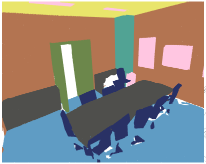
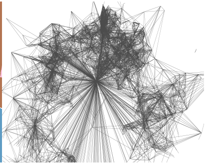





Loic Landrieu

- Researcher at the French Mapping Agency in the machine learning department STRUDEL.
- **Focus:** using the data-structure to increase precision and speed of learning methods.

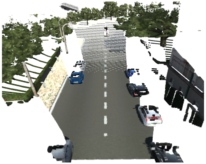
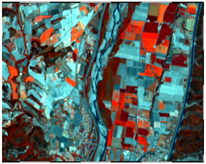
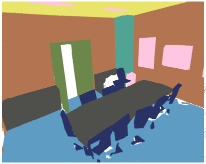
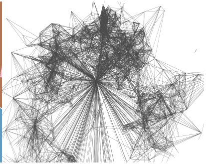





- Researcher at the French Mapping Agency in the machine learning department STRUDEL.
- **Focus:** using the data-structure to increase precision and speed of learning methods.
- **PhD:** learning and optimization on large graphs at INRIA (Bach/Obozinski).

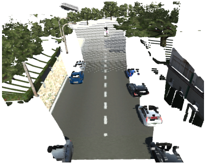
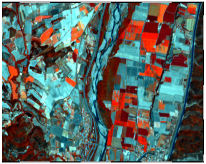
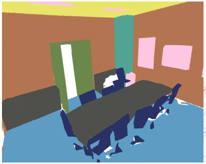
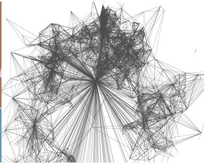
Loic Landrieu

- Researcher at the French Mapping Agency in the machine learning department STRUDEL.
- **Focus:** using the data-structure to increase precision and speed of learning methods.
- **PhD:** learning and optimization on large graphs at INRIA (Bach/Obozinski).
- **Current projects:**
- Analysis of very large point clouds.

Loic Landrieu

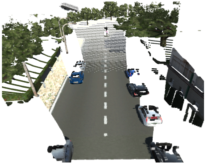
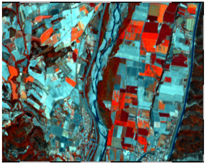
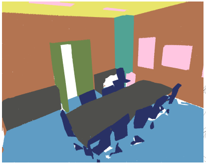
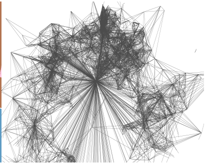
- Researcher at the French Mapping Agency in the machine learning department STRUDEL.
- **Focus:** using the data-structure to increase precision and speed of learning methods.
- **PhD:** learning and optimization on large graphs at INRIA (Bach/Obozinski).
- **Current projects:**
- Analysis of very large point clouds.
- Real-time analysis for autonomous vehicles.

Loic Landrieu

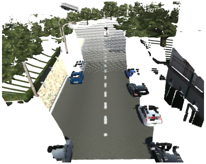
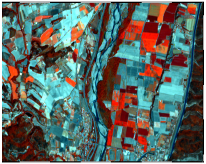
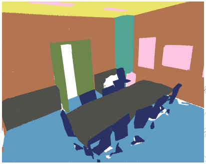
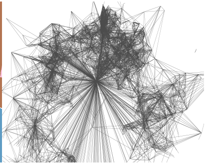
- Researcher at the French Mapping Agency in the machine learning department STRUDEL.
- **Focus:** using the data-structure to increase precision and speed of learning methods.
- **PhD:** learning and optimization on large graphs at INRIA (Bach/Obozinski).
- **Current projects:**

- Analysis of very large point clouds.
- Real-time analysis for autonomous vehicles.
- Superspectral satellite imagery time-sequences analysis.

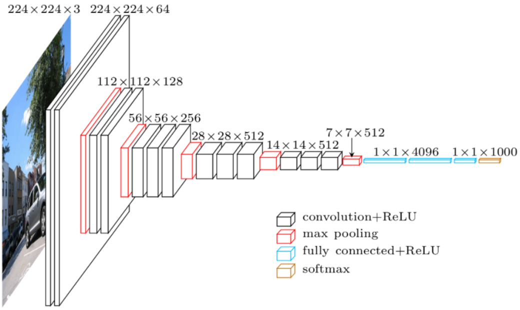
Loic Landrieu

- Researcher at the French Mapping Agency in the machine learning department STRUDEL.
- **Focus:** using the data-structure to increase precision and speed of learning methods.
- **PhD:** learning and optimization on large graphs at INRIA (Bach/Obozinski).
- **Current projects:**
- Analysis of very large point clouds.
- Real-time analysis for autonomous vehicles.
- Superspectral satellite imagery time-sequences analysis.
- Learning on huge graphs.

Presentation Layout

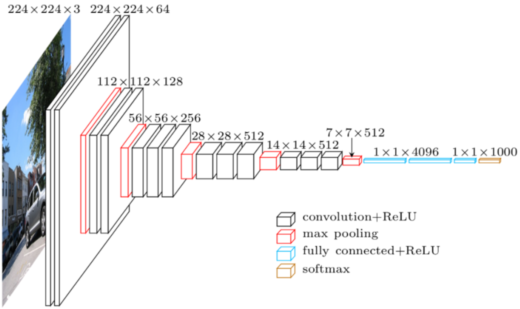
- **Motivation:** sufficient data trumps expertise.



credit : jeremyjordan.me

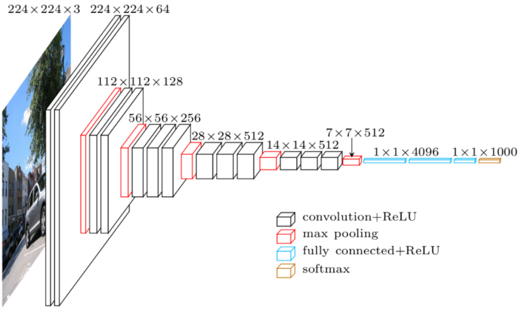
Deep Learning basics

- **Motivation:** sufficient data trumps expertise.
- **Main idea:** replace all handcrafted features (SIFTs, HOGs, etc...) with learned ones.



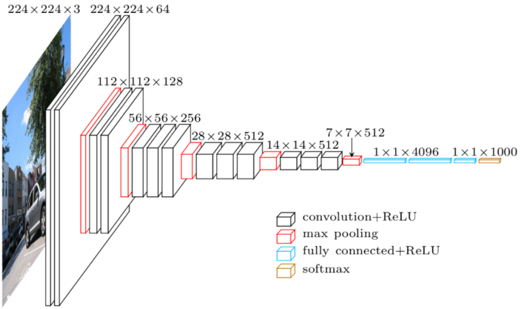
credit : jeremyjordan.me

- **Motivation:** sufficient data trumps expertise.
- **Main idea:** replace all handcrafted features (SIFTs, HOGs, etc...) with learned ones.
- Each unit (or neuron) is simple, the network architecture is complex.



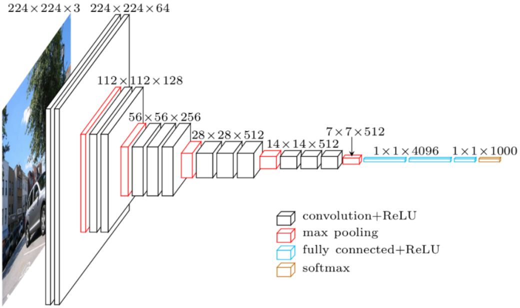
credit : jeremyjordan.me

- **Motivation:** sufficient data trumps expertise.
- **Main idea:** replace all handcrafted features (SIFTs, HOGs, etc...) with learned ones.
- Each unit (or neuron) is simple, the network architecture is complex.
- The network architecture must represent the structure of the data.



credit : jeremyjordan.me

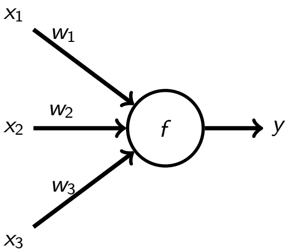
- **Motivation:** sufficient data trumps expertise.
- **Main idea:** replace all handcrafted features (SIFTs, HOGs, etc...) with learned ones.
- Each unit (or neuron) is simple, the network architecture is complex.
- The network architecture must represent the structure of the data.
- **AT NO POINT ARE ARTIFICIAL NEURAL NETWORKS SUPPOSED TO MODEL AN ACTUAL NEURON/BRAIN.**



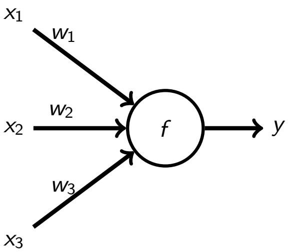
credit : jeremyjordan.me

Artificial Neuron

- $y = f(\sum_i w_i x_i)$
- x_i : inputs
- w_i : weights
- f : non-linearity
- y : outputs



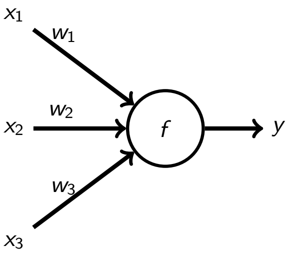
- $y = f(\sum_i w_i x_i)$
- x_i : inputs
- w_i : weights
- f : non-linearity
- y : outputs
- **In short :** a matrix product $X^\intercal W$ and non-linearity.



Artificial Neuron

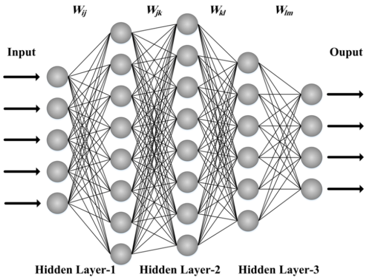
- $y = f(\sum_i w_i x_i)$
- x_i : inputs
- w_i : weights
- f : non-linearity
- y : outputs
- **In short :** a matrix product $X^\intercal W$ and non-linearity.
- Non linearity essential (or else simplifies to a matrix product).

  f = sigmoid, Relu=max$(0, x)$.



Multilayer Perceptron

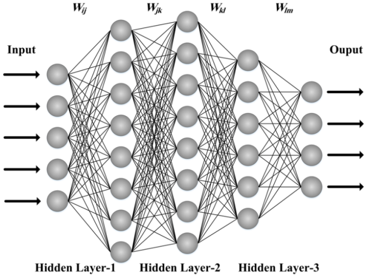
- Organization in layers



credit : pubs.sciepub.com/ajmm

Multilayer Perceptron

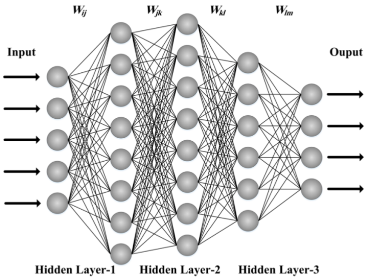
- Organization in layers
- the deeper layers extracts more complicated / abstract features



credit : pubs.sciepub.com/ajmm

Multilayer Perceptron

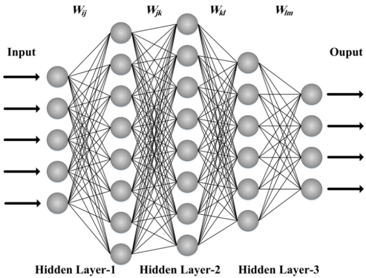
- Organization in layers
- the deeper layers extracts more complicated / abstract features
- Very old model, exists since the 50s



credit : pubs.sciepub.com/ajmm

Multilayer Perceptron

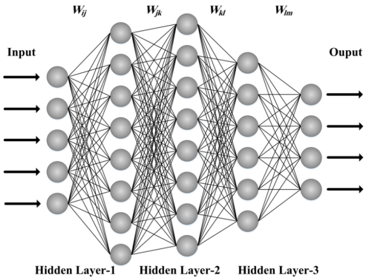
- Organization in layers
- the deeper layers extracts more complicated / abstract features
- Very old model, exists since the 50s
- **Universal Approximation Theorem:** any functions of x can be approximated to arbitrary precision by a MLP with sufficient width.



credit : pubs.sciepub.com/ajmm

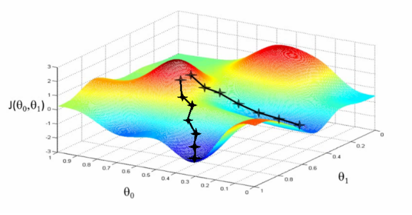
Multilayer Perceptron

- Organization in layers
- the deeper layers extracts more complicated / abstract features
- Very old model, exists since the 50s
- **Universal Approximation Theorem:** any functions of x can be approximated to arbitrary precision by a MLP with sufficient width.
- Simple model, no assumption whatsoever on the data structure.



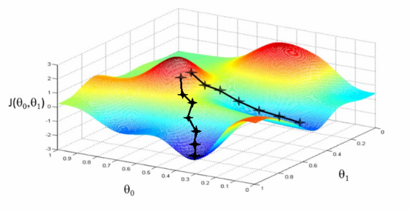
credit : pubs.sciepub.com/ajmm

- **Training a neural network:** finding values for the weights w such that output y is close to a ground truth value $\hat{y}$.



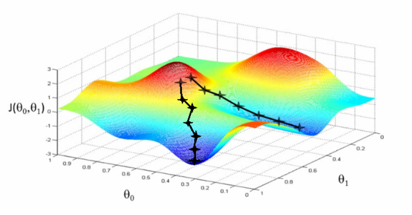
credit : exortech.github.io

- **Training a neural network:** finding values for the weights w such that output y is close to a ground truth value $\hat{y}$.
- We define a loss function $\mathcal{L}(y(w), \hat{y})$ decreasing with the precision (for example: $\|y - \hat{y}\|^2$, cross-entropy).



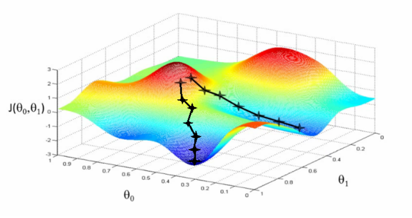
credit : exortech.github.io

- **Training a neural network:** finding values for the weights w such that output y is close to a ground truth value $\hat{y}$.
- We define a loss function $\mathcal{L}(y(w), \hat{y})$ decreasing with the precision (for example: $\|y - \hat{y}\|^2$, cross-entropy).
- $\mathcal{L}$ is usually an (almost) differentiable function but is generally nonconvex.



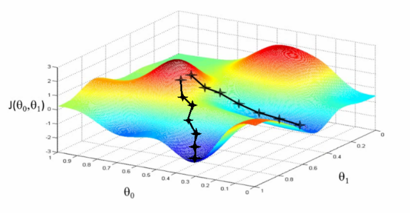
credit : exortech.github.io

- **Training a neural network:** finding values for the weights w such that output y is close to a ground truth value $\hat{y}$.
- We define a loss function $\mathcal{L}(y(w), \hat{y})$ decreasing with the precision (for example: $\|y - \hat{y}\|^2$, cross-entropy).
- $\mathcal{L}$ is usually an (almost) differentiable function but is generally nonconvex.
- However, we can find good weights with Stochastic Gradient Descent.



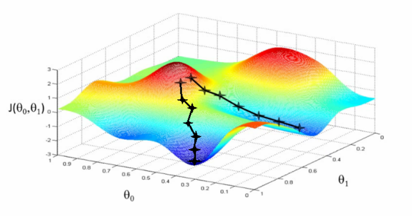
credit : exortech.github.io

- **Training a neural network:** finding values for the weights w such that output y is close to a ground truth value $\hat{y}$.
- We define a loss function $\mathcal{L}(y(w), \hat{y})$ decreasing with the precision (for example: $\|y - \hat{y}\|^2$, cross-entropy).
- $\mathcal{L}$ is usually an (almost) differentiable function but is generally nonconvex.
- However, we can find good weights with Stochastic Gradient Descent.
- Automatic differentiation: gradients are easily computed and propagated ("backprop").



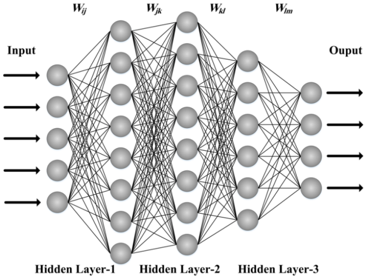
credit : exortech.github.io

- **Training a neural network:** finding values for the weights w such that output y is close to a ground truth value $\hat{y}$.
- We define a loss function $\mathcal{L}(y(w), \hat{y})$ decreasing with the precision (for example: $\|y - \hat{y}\|^2$, cross-entropy).
- $\mathcal{L}$ is usually an (almost) differentiable function but is generally nonconvex.
- However, we can find good weights with Stochastic Gradient Descent.
- Automatic differentiation: gradients are easily computed and propagated ("backprop").
- Supervized learning, require a lot of high quality annotations.



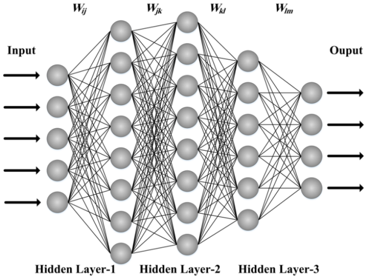
credit : exortech.github.io

- **Problem:** the size of W increase quadratically with the layers' size



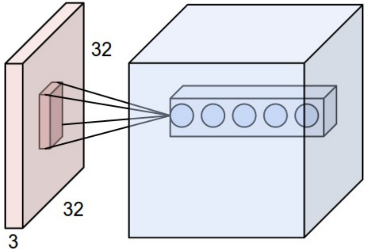
credit : pubs.sciepub.com/ajmm

- **Problem:** the size of W increase quadratically with the layers' size
- 100×100 image : 10^8 parameters per layer! Unmanageable.



credit : pubs.sciepub.com/ajmm

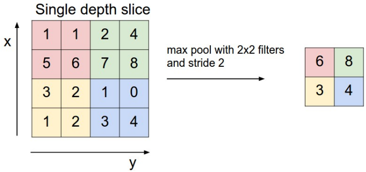
- **Problem:** the size of W increase quadratically with the layers' size
- 100×100 image : 10^8 parameters per layer! Unmanageable.
- **Solution:** local convolutions: values of a layer only depend on a small number of points in the previous layer (the *receptive field*).



credit : cs231n.github.io

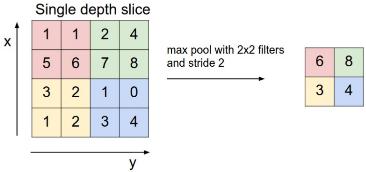
Convolutional Neural Network

- **Problem:** the size of W increase quadratically with the layers' size
- 100×100 image : 10^8 parameters per layer! Unmanageable.
- **Solution:** local convolutions: values of a layer only depend on a small number of points in the previous layer (the *receptive field*).
- Paired with *Pooling layers* which decrease the size of the feature maps.



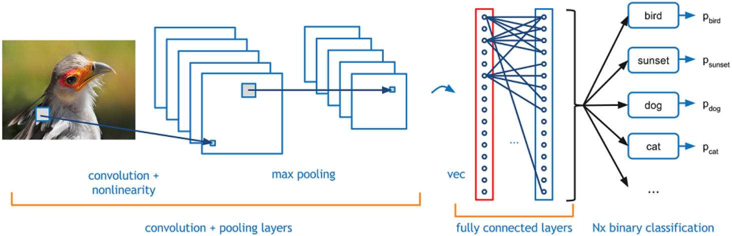
credit : cs231n.github.io

- **Problem:** the size of W increase quadratically with the layers' size
- 100×100 image : 10^8 parameters per layer! Unmanageable.
- **Solution:** local convolutions: values of a layer only depend on a small number of points in the previous layer (the *receptive field*).
- Paired with *Pooling layers* which decrease the size of the feature maps.
- Very successful for images, exploits the spatial structure.



credit : cs231n.github.io

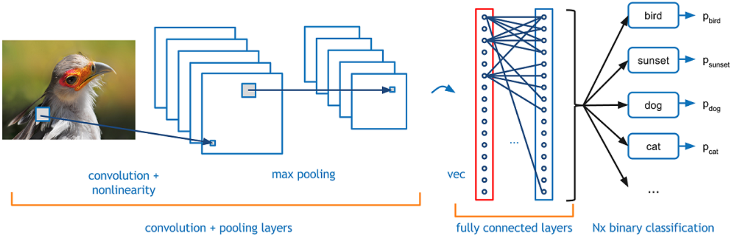
Convolutional Neural Network II



- Traditional structure:
  - Sequence of (Conv + Pool) units to compute local features and decrease embeddings size
  - One (or two) fully connected MLP at the end to analyze the whole image.

credit: adeshpande3.github.io

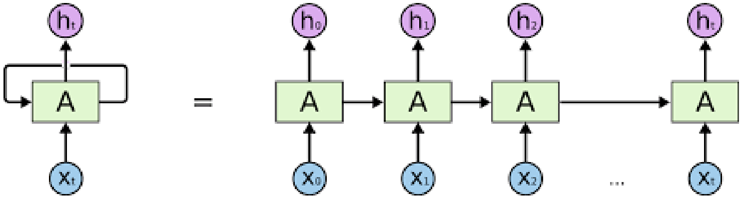
Convolutional Neural Network II



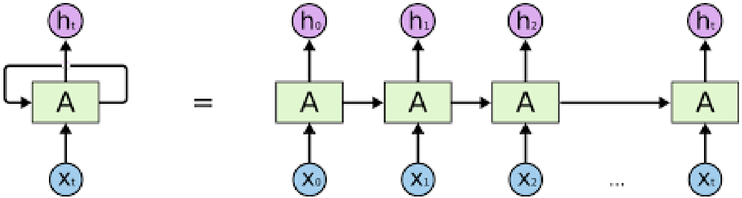
- Traditional structure:
  - Sequence of (Conv + Pool) units to compute local features and decrease embeddings size
  - One (or two) fully connected MLP at the end to analyze the whole image.
  - Deep structures seem to work best.

credit: adeshpande3.github.io

Recurrent Neural Network

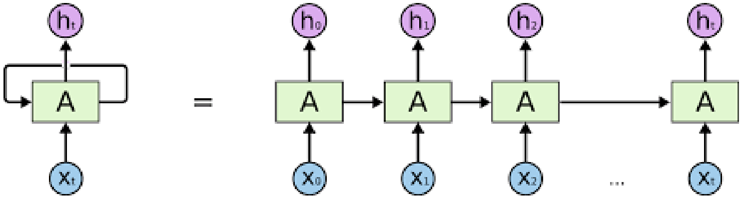


- **Objective:** modeling temporal structure

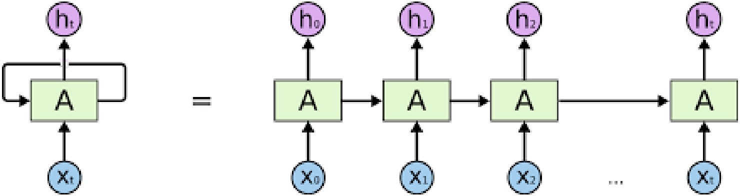


- **Objective:** modeling temporal structure
- **General idea:** the network maintains a hidden state h_t called *memory*

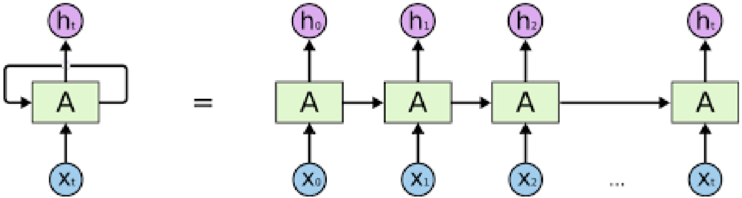
Recurrent Neural Network



- **Objective:** modeling temporal structure
- **General idea:** the network maintains a hidden state h_t called *memory*
- **Update:** $h_{t+1} = f(h_t, x_t)$



- **Objective:** modeling temporal structure
- **General idea:** the network maintains a hidden state h_t called *memory*
- **Update:** $h_{t+1} = f(h_t, x_t)$
- There exists many type of RNNs: LSTMs, GRUs, etc...



- **Objective:** modeling temporal structure
- **General idea:** the network maintains a hidden state h_t called *memory*
- **Update:** $h_{t+1} = f(h_t, x_t)$
- There exists many type of RNNs: LSTMs, GRUs, etc...
- Can also be used to model spatial structure.
- Successful for sequence learning in natural language processing, speech recognition, etc.

Deep Learning In Practice

- Tensorflow, Pytorch: high level python-based language doing all the tedious work.

Deep Learning In Practice

- Tensorflow, Pytorch: high level python-based language doing all the tedious work.
- Example for a 1 hidden later MLP to classify D-dimension data between K classes.

- Tensorflow, Pytorch: high level python-based language doing all the tedious work.
- Example for a 1 hidden later MLP to classify D-dimension data between K classes.

layer1 = Linear(D,L), layer2= Linear(L,K)

Deep Learning In Practice

- Tensorflow, Pytorch: high level python-based language doing all the tedious work.
- Example for a 1 hidden later MLP to classify D-dimension data between K classes.

```
layer1 = Linear(D,L), layer2= Linear(L,K)
optimizer = SGD()
```

Deep Learning In Practice

- Tensorflow, Pytorch: high level python-based language doing all the tedious work.
- Example for a 1 hidden later MLP to classify D-dimension data between K classes.

```
layer1 = Linear(D,L), layer2= Linear(L,K)
optimizer = SGD()
for i_epoch in range(n_epoch):
```

- Tensorflow, Pytorch: high level python-based language doing all the tedious work.
- Example for a 1 hidden later MLP to classify D-dimension data between K classes.

```
layer1 = Linear(D,L), layer2= Linear(L,K)
optimizer = SGD()
for i_epoch in range(n_epoch):
  for batch, gt in enumerate(training_set) % batch :B × D, gt : B
```

- Tensorflow, Pytorch: high level python-based language doing all the tedious work.
- Example for a 1 hidden later MLP to classify D-dimension data between K classes.

```
layer1 = Linear(D,L), layer2= Linear(L,K)
optimizer = SGD()
for i_epoch in range(n_epoch):
  for batch, gt in enumerate(training_set) % batch :B × D, gt : B
    feat = Relu(layer1(batch))                             % feat : B × L
```

- Tensorflow, Pytorch: high level python-based language doing all the tedious work.
- Example for a 1 hidden later MLP to classify D-dimension data between K classes.

```
layer1 = Linear(D,L), layer2= Linear(L,K)
optimizer = SGD()
for i_epoch in range(n_epoch):
  for batch, gt in enumerate(training_set) % batch : B × D, gt : B
    feat = Relu(layer1(batch))                  % feat : B × L
    feat = Relu(layer2(feat))                   % feat : B × K
```

Deep Learning In Practice

- Tensorflow, Pytorch: high level python-based language doing all the tedious work.
- Example for a 1 hidden later MLP to classify D-dimension data between K classes.

```
layer1 = Linear(D,L), layer2= Linear(L,K)
optimizer = SGD()
for i_epoch in range(n_epoch):
  for batch, gt in enumerate(training_set) % batch :B × D, gt : B
    feat = Relu(layer1(batch))                     % feat : B × L
    feat = Relu(layer2(feat))                      % feat : B × K
    output = softmax(feat)                         % output: B × K
```

- Tensorflow, Pytorch: high level python-based language doing all the tedious work.
- Example for a 1 hidden later MLP to classify D-dimension data between K classes.

```
layer1 = Linear(D,L), layer2= Linear(L,K)
optimizer = SGD()
for i_epoch in range(n_epoch):
  for batch, gt in enumerate(training_set) % batch :B × D, gt : B
    feat = Relu(layer1(batch))                         % feat : B × L
    feat = Relu(layer2(feat))                          % feat : B × K
    output = softmax(feat)                             % output: B × K
    loss = cross_entropy(output, gt)                   % compute the loss
```

Deep Learning In Practice

- Tensorflow, Pytorch: high level python-based language doing all the tedious work.
- Example for a 1 hidden later MLP to classify D-dimension data between K classes.

```
layer1 = Linear(D,L), layer2= Linear(L,K)
optimizer = SGD()
for i_epoch in range(n_epoch):
  for batch, gt in enumerate(training_set) % batch :B × D, gt : B
    feat = Relu(layer1(batch))                 % feat : B × L
    feat = Relu(layer2(feat))                  % feat : B × K
    output = softmax(feat)                     % output: B × K
    loss = cross_entropy(output, gt)           % compute the loss
    loss.backward()                            % compute the gradient
```

- Tensorflow, Pytorch: high level python-based language doing all the tedious work.
- Example for a 1 hidden later MLP to classify D-dimension data between K classes.

```
layer1 = Linear(D,L), layer2= Linear(L,K)
optimizer = SGD()
for i_epoch in range(n_epoch):
  for batch, gt in enumerate(training_set) % batch : B × D, gt : B
    feat = Relu(layer1(batch))              % feat : B × L
    feat = Relu(layer2(feat))               % feat : B × K
    output = softmax(feat)                  % output: B × K
    loss = cross_entropy(output, gt)        % compute the loss
    loss.backward()                         % compute the gradient
    optimizer.step(i_epoch)                 % gradient step
```

- Tensorflow, Pytorch: high level python-based language doing all the tedious work.
- Example for a 1 hidden later MLP to classify D-dimension data between K classes.

```
layer1 = Linear(D,L), layer2= Linear(L,K)
optimizer = SGD()
for i_epoch in range(n_epoch):
  for batch, gt in enumerate(training_set) % batch :B × D, gt : B
    feat = Relu(layer1(batch))                          % feat : B × L
    feat = Relu(layer2(feat))                           % feat : B × K
     output = softmax(feat)                             % output: B × K
    loss = cross_entropy(output, gt)                    % compute the loss
    loss.backward()                                     % compute the gradient
    optimizer.step(i_epoch)                             % gradient step
  print("epoch = epoch, loss = avg_loss)                % monitor loss decrease
```

Deep Learning In Practice

- Tensorflow, Pytorch: high level python-based language doing all the tedious work.
- Example for a 1 hidden later MLP to classify D-dimension data between K classes.

```
layer1 = Linear(D,L), layer2= Linear(L,K)
optimizer = SGD()
for i_epoch in range(n_epoch):
  for batch, gt in enumerate(training_set) % batch :B × D, gt : B
    feat = Relu(layer1(batch))                    % feat : B × L
    feat = Relu(layer2(feat))                     % feat : B × K
    output = softmax(feat)                        % output: B × K
    loss = cross_entropy(output, gt)              % compute the loss
    loss.backward()                               % compute the gradient
    optimizer.step(i_epoch)                       % gradient step
  print("epoch = epoch, loss = avg_loss)          % monitor loss decrease
return softmax(Relu(layer2(Relu(layer1(test_set)))))
```

Presentation Layout

In the next hour, we will get insight on:

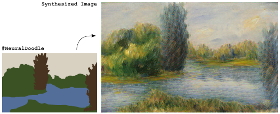
- Motivation for Deep Learning on Raster Imagery
- Semantic Segmentation Fully-conv. Networks
- Multi-modal Classification
- Object Detection
- Change Detection and Multi-temporal Analysis
- Classif. of Hyperspectral Data
- Deep Learning on SAR Data
- Resources
- Practical Session: Semantic Segmentation
- Bibliography 2D

Lots of common application in everyone's life:

Self-driving cars

Facebook's facial recognition

Google Translate... images



Painting like Monet!

Figure: Applications of Deep Learning

So... what can we do in remote sensing?

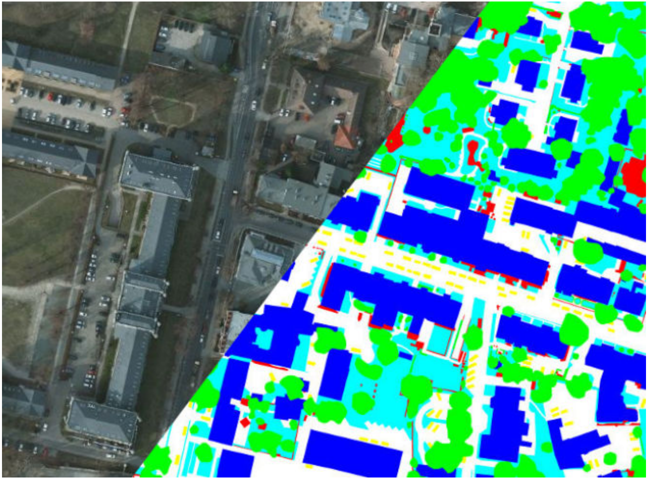


Figure: Can we understand and translate images to cartography?

Motivation for Deep Learning on Raster Imagery

Data Evolution	*Feature Extraction*	*Classification*
resol. −	Pixel-based filtering	Manual modeling, thresholding
resol. +	Textures, local features	Distribution estimates (GMMs, etc.)
data +		Learning-based classifiers (SVMs, ensemble methods: random forests, boosting): high-dimensional, non-linear, complex
resol. ++	Complex features (object modeling)	active learning, latentSVM
data ++	Patchs of pixels, filter banks	Deep neural networks (RBM, RCNN)

resolution, quantity

Figure: An history of classification in remote sensing

Motivation for Deep Learning on Raster Imagery

- In some specific cases, standard machine learning approaches or sensor-based heuristics are well enough...
- but Convolutional Neural Networks make good **generic classifiers**

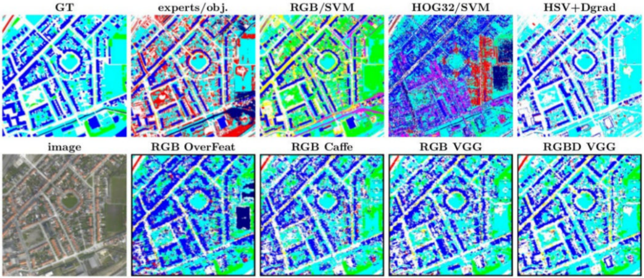


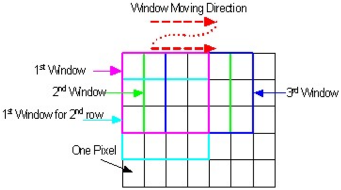
Figure: Benchmarking old-school vs. deep learning methods (Campos-Taberner et al., 2016)

Simple Deep Learning Baseline

3D	*Algorithm*	Imp. surf.	Build.	Low veg.	Tree	Car	Clutter	Boat	Water	Overall acc. %	Cohen κ
★	Expert	58.97	63.87	74.55					92.39	∅	∅
	RGB/SVM	53.89	53.53	50.32	32.97	24.02	13.75	12.12	98.52	60.77	0.52
★	RGB*d*/SVM	14.51	67.79	38.03	27.43	7.15	1.12	14.58	98.45	50.76	0.41
★	RGB*d*I/SVM	60.86	69.01	57.12	38.12	11.59	20.49	15.04	94.42	63.83	0.56
	HOG32/SVM	28.94	43.17	48.77	27.32	30.24	17.39	12.61	88.02	52.45	0.41
	HOG16/SVM	39.52	38.45	35.65	29.99	21.93	16.13	13.52	80.02	49.4	0.36
	HSV/SVM	71.60	46.97	68.38	0.12	0.00	13.71	0.00	92.14	70.16	0.60
★	HSVDGr/SVM	73.30	70.85	68.75	0.17	0.00	17.11	0.00	92.37	73.60	0.65
	SOM							51.45		∅	∅
	DtMM					48.46				∅	∅
	RGB OverFeat/SVM	55.86	63.34	59.48	64.44	36.03	28.31	41.51	92.07	67.97	0.59
	RGB Caffe/SVM	62.32	62.66	63.23	60.84	31.34	32.49	46.57	95.61	71.06	0.63
	RGB VGG/SVM	63.18	64.66	63.60	66.98	31.46	43.68	51.92	95.93	72.36	0.64
★	RGB*d* VGG/SVM	66.02	74.26	65.04	66.94	32.04	44.96	50.61	96.31	74.77	0.67
★	RGBd^+ VGG/SVM	67.66	72.70	68.38	78.77	33.92	45.6	56.10	96.50	76.56	0.70
★	RGBd^+ trained AlexNet	79.10	75.60	78.00	79.50	50.80	63.40	44.80	98.20	83.32	0.78

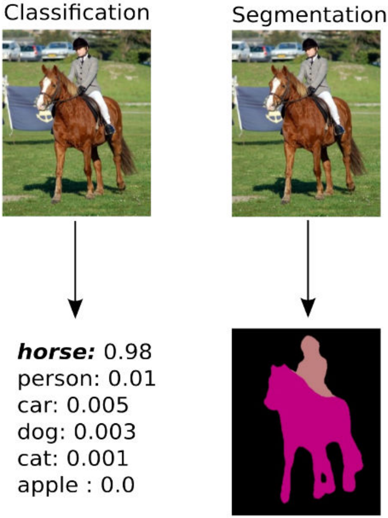
Figure: Benchmarking old-school vs. deep learning methods (Campos-Taberner et al., 2016)

Remote Sensing processing:

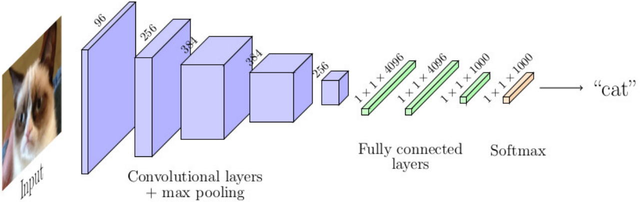


- Extract patches from the image using a sliding window with overlap
- Train on images with ground-truth
- Apply on test patches (average several predictions on overlapping pixels)

- **Classification:** 1 image (or local patch in remote sensing) $\longrightarrow$ 1 label
- **Segmentation:** 1 pixel $\longrightarrow$ 1 label
- Image = structured pixel ensemble



AlexNet CNN for classification (Krizhevsky et al., 2012)



Fully-convolutional AlexNet for semantic segmentation (Long et al., 2015)

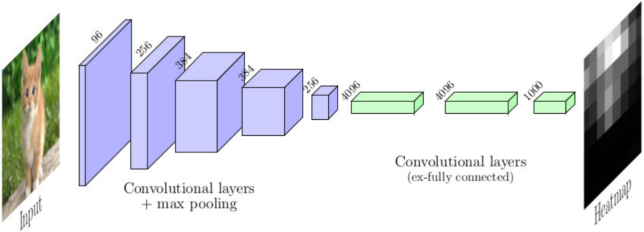
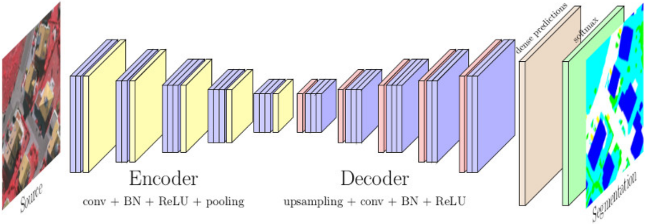
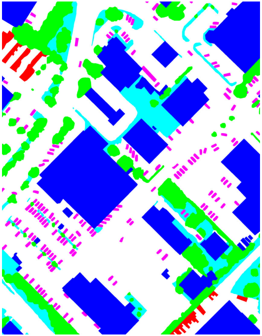


Figure: Fully-convolutional networks (FCNs)

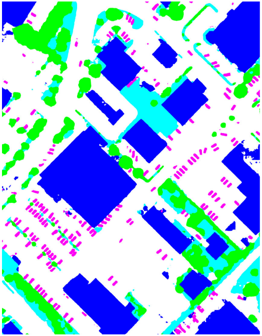


- SegNet: A deep convolutional Encoder-Decoder architecture for Image Segmentation (Bandrinarayanan et al., 2016)
- And today: U-Net (Ronneberger et al., 2015), Hourglass (Newell et al. 2016))

IR/R/G 10cm/pixel



Ground-truth



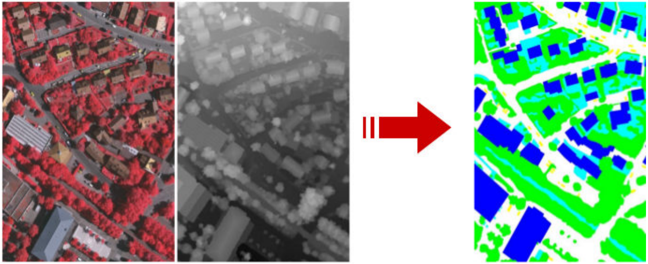
SegNet

Figure: SegNet semantic segmentation (Audebert et al., 2017a)

Multi-modal Semantic Segmentation

How to automatically generate maps from aerial imagery?

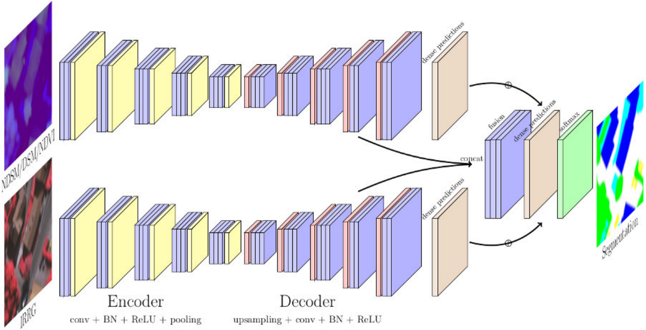
- Non-standard (yet complimentary) imagery: multispectral, LiDAR...
- Auxiliary data: existing open-source maps (OpenStreetMap, etc.)

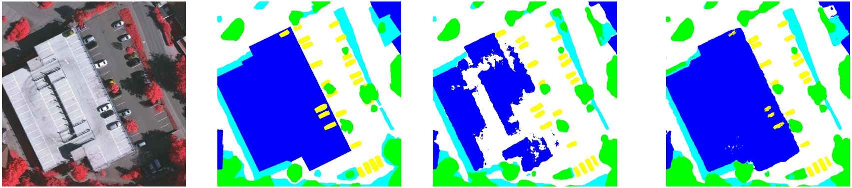


Multi-modal Semantic Segmentation

How to automatically generate maps from aerial imagery?

- Dual-stream networks which translate data to representations (coding)...
- Fusion and decoding from representations to maps





IR/R/G | Ground-truth | IR/R/G pred. | IR/R/G +DSM pred.

Figure: Multimodal semantic segmentation results (Audebert et al., 2017a)

- Sensor-based information helps!

Multi-modal Semantic Segmentation

How to automatically generate maps from aerial imagery using auxiliary maps?

- Incorporate information to guide the process, FuseNet: (Hazirbas et al., 2016)
- Faster training and better results!

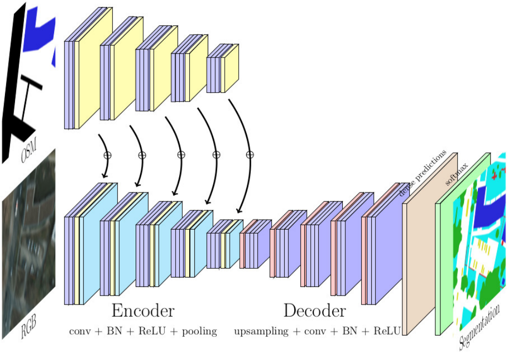


Figure: Joint learning of RGB imagery and OSM (Audebert et al., 2017b)

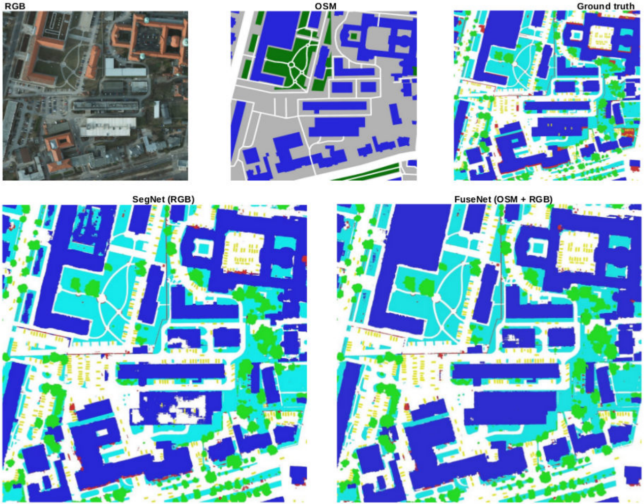
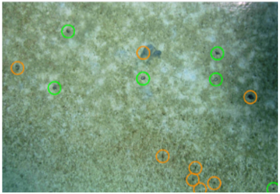


Figure: RGB+OSM semantic segmentation results (Audebert et al., 2017b)

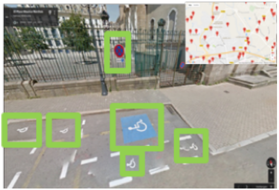
Object Detection

Any generic detection network (e.g. Faster-RCNN) can be fine-tuned to fit user's needs, e.g.:

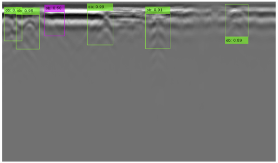
- mapping and monitoring marine turtles from UAVs in environmental surveys
- detecting accessibility signs to map related parking lots
- detect and map buried networks from geophysical data (ground-penetrating radar)



Turtle detection
WIPSEA



Accessibility mapping
(Nassar & Lefèvre, 2019)



Buried network detection
(Pham & Lefèvre, 2018)

How to extend semantic analysis to multi-temporal data ?

- detect changes
- monitor activity in high-revisit rate acquisitions
- focus on specific changes (urban, agriculture, vehicles, industrial activity...)

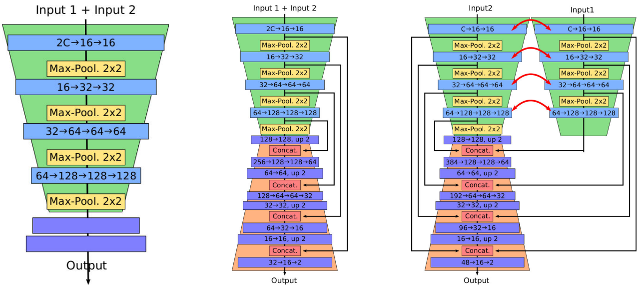
Date 1

Date 2

Change map

How to extend semantic analysis to multitemporal data ?

- As before, simply concatenating images...
- Or with siamese networks, i.e. dual stream nets which share weights.



Standard CNN U-Net Siamese network

Figure: CNN / FCN architectures for change detection. (Daudt et al., 2018)

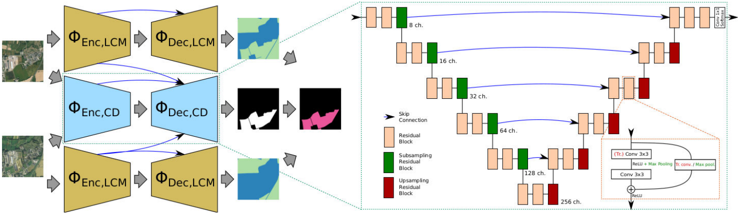
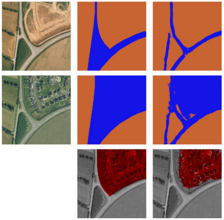


Figure: Semantic change detection. (Daudt et al., 2019)

Semantic change detection:

- Joint multi-task learning of semantics and differences with FCNs
- Prediction of land cover and change maps



Siamese networks can work with very different inputs! (e.g. ground vs aerial imagery)

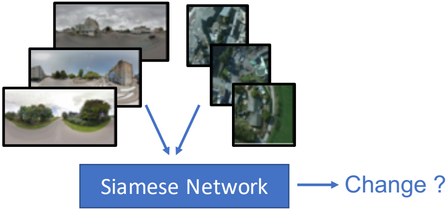


Figure: Multiview change detection (Lefèvre et al. 2017)

Recurrent Neural Networks

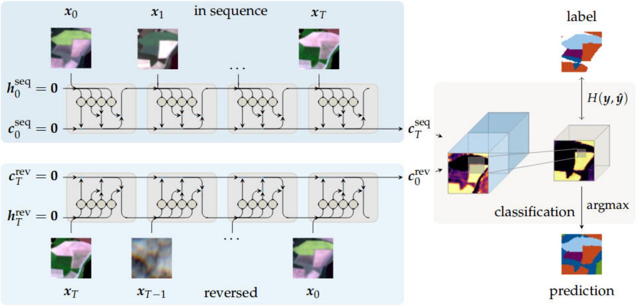


Figure: Multi-Temporal Land Cover Classification with Recurrent Auto-encoders (Russwurm & Köner, 2018)

Classification of Hyperspectral Data

How to extend semantic analysis to hyperspectral imaging (HSI) data ?

- RGB to 100+ bands, image to data cube;
- finer spectral description, out-of-visible;
- lower resolution but finer class discrimination (materials, stressed or healthy vegetation...)

Figure: Hyperspectral data cube: Houston (Texas, USA) – IEEE GRSS IADF TC's Data Fusion Contest 2018

CNN architectures adapted to HSI classification:

- 1D CNN: spectrum classification
- 1D RNN: spectral sequence classification

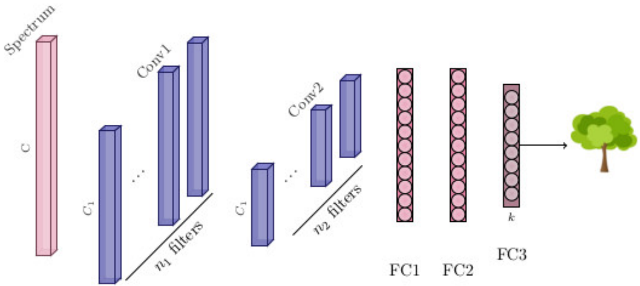


Figure: CNN 1D

CNN architectures adapted to HSI classification:

- Spatial-spectral, 2D+1D approaches
- Reduce to RGB-like data + 2D CNN
- PCA or supervised reduction : alternate 2D and 1D convolutions

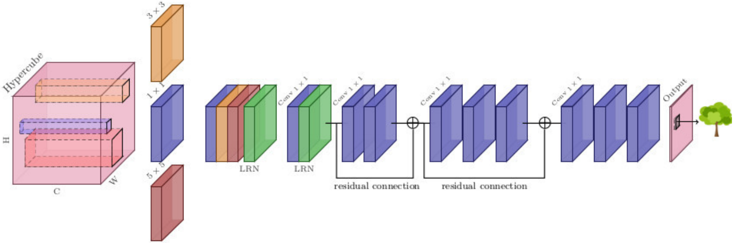


Figure: CNN 2D+1D (Lee et al., 2016)

CNN architectures adapted to HSI classification:

- End-to-end 3D pattern recognition: apply learnable (w, h, B) filters on the hypercube

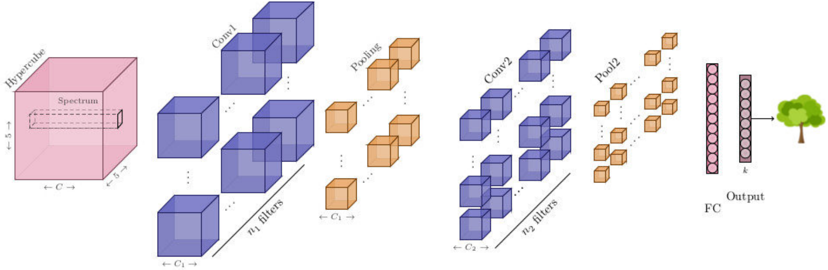


Figure: CNN 3D (Audebert et al, 2019)

HSI classification with CNNs:

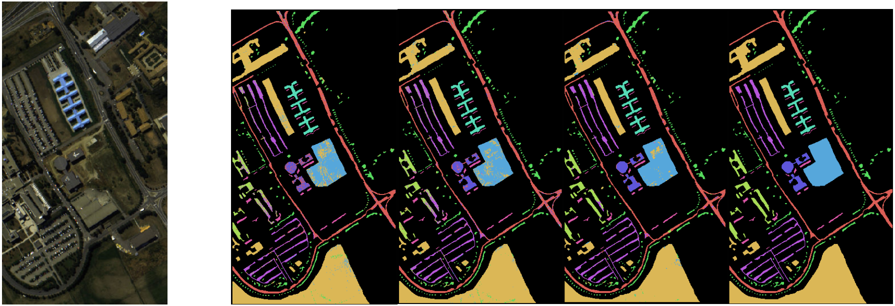


Figure: Comparison of HSI classifications with various CNNs (Audebert et al, 2019)

Deep learning on SAR Data

How to extend deep learning processing to SAR data ?

- Specific physics, different from optical images: intensity + phase;
- High resolution, "cloud-free" images;
- Presence of "speckle" and changing appearance depending on the angle of view.

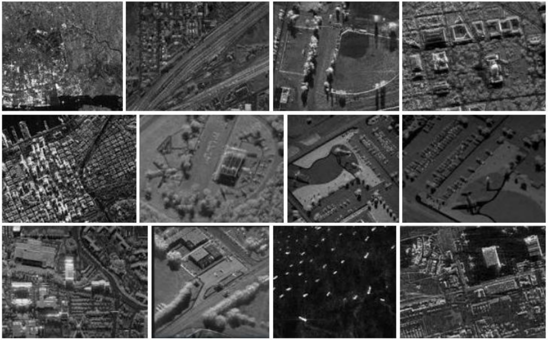


Figure: SAR image examples

Despeckling:

- Inspired by denoising Denoising Auto-Encoders (Vincent et al., 2010);
- Auto-encoder learn to reconstruct the image from itself
- Denoising / despeckling autoencoders learn to reconstruct the image from the image with added speckle

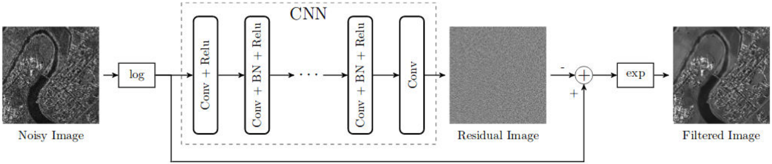
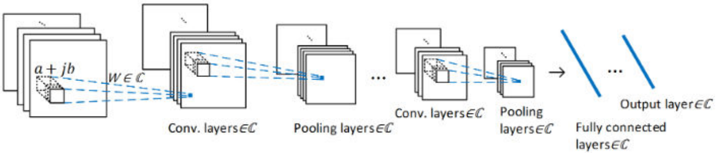


Figure: Despeckling auto-encoder (Chierchia et al., 2017)

Classification of SAR Data

- Usually straightforward (but do they miss something?);
- Complex valued CNN for processing intensity + phase images (Haensch & Hellwich, 2010) (Zhang et al., 2010)



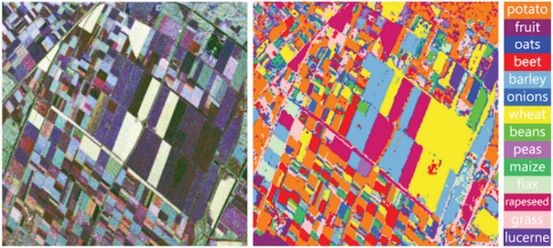


Figure: SAR classification (Zhang et al., 2017) and (Zhu et al., 2017)

Similarly to optical imagery, SAR imagery can be exploited in many ways, e.g.

- ship analysis from Sentinel-1: detection, length estimation, classification

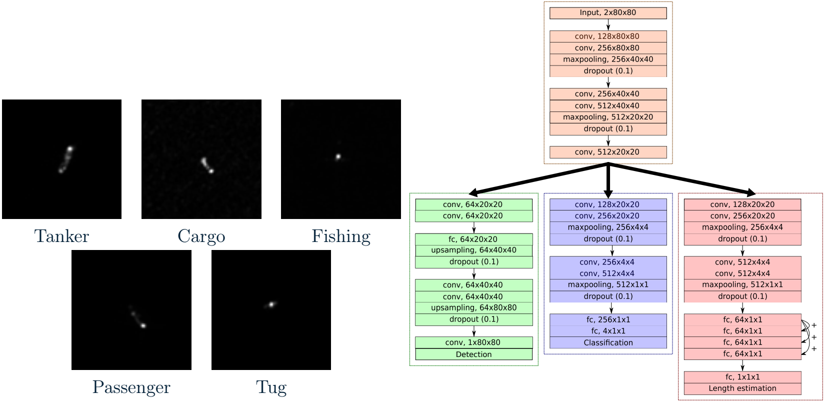


Figure: Dechesne et al., 2019

Public resources

Good reads:

- **Deep Learning for Remote Sensing Data: A Technical Tutorial on the State of the Art**, *Zhang, Zhang and Du*, IEEE Geosci. and Rem. Sens. Mag, 6 (2) June 2016
- **Deep learning in remote sensing: A comprehensive review and list of resources**, *Zhu, Tuia, Mou, Xia, Zhang, Xu, and Fraundorfer*, IEEE Geosci. and Rem. Sens. Mag, 5 (4) Dec. 2017
- **Deep Learning for Classification of Hyperspectral Data: A Comparative Review**, *Audebert, Le Saux and Lefèvre*, IEEE Geosci. and Rem. Sens. Mag., 7 (2) June 2019

Public resources

Toolbox:

- DeepNetsForEO (`https://github.com/nshaud/DeepNetsForEO`): python code for semantic segmentation of aerial / satellite imagery
- DeepHyperX (`https://github.com/nshaud/DeepHyperX`): python toolbox for classification of hyperspectral imagery (spectral, spatial-spectral and 3D convolutions)

Public resources

Public datasets:

- ISPRS datasets: semantic labeling, reconstruction $\rightsquigarrow$ https://www.isprs.org/data/
- IEEE GRSS Data Fusion Contests: http://www.grss-ieee.org/community/technical-committees/data-fusion/data-fusion-contest/
- IEEE GRSS: hyperspectral datasets with standard train/test splits (DFC2018, Pavia, Indian Pines) $\rightsquigarrow$ http://dase.grss-ieee.org/
- INRIA Aerial Semantic labeling dataset: buildings $\rightsquigarrow$ https://project.inria.fr/aerialimagelabeling/
- XView: objects in aerial images $\rightsquigarrow$ http://xviewdataset.org/
- DOTA: Detecting Objects in Aerial images $\rightsquigarrow$ https://captain-whu.github.io/DOTA/dataset.html

Practical session: objectives

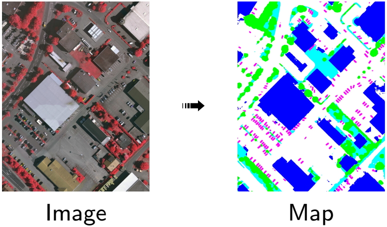
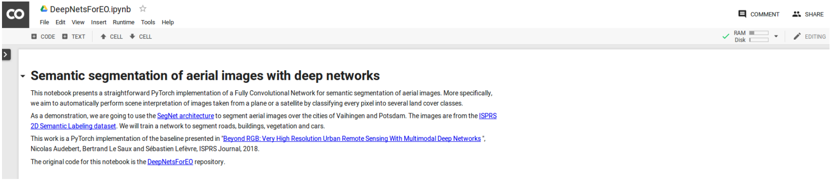


Figure: Let's practice: semantic segmentation of Earth-observation data!

We will:

- load data and perform data augmentation;
- define a network model
- train the network on colab's GPU
- test the net on some test images and evaluate results



Link (in 2 clicks): https://blesaux.github.io/teaching/DL4RS

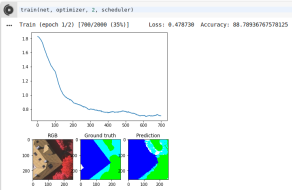


Figure: That's a good start!

Bibliography 2D I

Lefèvre et al, 2017 Lefèvre, S., Tuia, D., Wegner, J.D., Produit, T., Nasser, A.S., Towards seamless multi-view scene analysis from satellite to street-level. Proceedings of the IEEE, 2017
Nassar & Lefèvre 2019 Nassar, A.S., Lefèvre, S., Automated mapping of accessibility signs with deep learning from ground-level imagery and open data. JURSE, 2019.
Pham & Lefèvre 2018 Pham, M.T., Lefèvre, S., Buried object detection from b-scan ground penetrating radar data using faster-rcnn, IGARSS, 2018.
Dechesne et al, 2019 Dechesne, C., Lefèvre, S., Vadaine, R., Hajduch, G., Fablet, R., Multi-task deep learning from sentinel-1 sar : ship detection, classification and length estimation, BiDS, 2019.

Bibliography 2D II

Campos-Taberner et al, 2017 Campos-Taberner et al., Processing of Extremely High-Resolution LiDAR and RGB Data: Outcome of the 2015 IEEE GRSS Data Fusion Contest-Part A: 2D Contest. JSTARS, 2016

Krizhevsky et al., 2012 Krizhevsky, Sutskever and Hinton, ImageNet Classification with Deep ConvolutionalNeural Networks. NIPS, 2012.

Long et al., 2015 Long, Shelhamer and Darell, Fully-convolutional networks for semantic segmentation. CVPR, 2015.

Bandrinarayanan et al., 2016 Bandrinarayanan, Kendall and Cipolla, SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, TPAMI, 2017.

Ronneberger et al., 2015 Ronneberger, Fischer and Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI, 2015.

Newell et al., 2016 Newell, Yang and Deng, Stacked Hourglass Networks for Human Pose Estimation, ECCV, 2016.

Bibliography 2D III

Audebert et al, 2017a Audebert, Le Saux and Lefèvre, Beyond RGB: Very high resolution urban remote sensing with multimodal deep networks. ISPRS PHOTO, 2017

Audebert et al, 2017b Audebert, Le Saux and Lefèvre, Joint Learning from Earth Observation and OpenStreetMap Data to Get Faster Better Semantic Maps. CVPR/EarthVision, 2017

Daudt et al., 2018 Daudt, Le Saux and Boulch, Fully Convolutional Siamese Networks for Change Detection, ICIP, 2018

Daudt et al., 2019 Daudt, Le Saux, Boulch and Gousseau, High Resolution Semantic Change Detection, to appear, 2019

Russwurm and Koener, 2018 Russwurm and Köner, Multi-Temporal Land Cover Classification with Sequential Recurrent Encoders, ISPRS J. Geo-Information, 2018

Lee et al., 2016 Lee and Kwon, Going Deeper with Contextual CNN for Hyperspectral Image Classification, IGARSS 2016

Audebert et al, 2019 Audebert, Le Saux and Lefèvre, Deep Learning for Classification of Hyperspectral Data: A Comparative Review. to appear, 2019

Bibliography IV

Vincent et al, 2010 Vincent, Larochelle, Lajoie, Bengio and Manzagol, Stacked Denoising Autoencoders. JMLR, 2010
Chierchia et al, 2017 Chierchia, Cozzolino, Poggi, and Verdoliva, SAR image despeckling through convolutional neural networks, IGARSS 2017
Haensch & Hellwich, 2010 Complex-Valued Convolutional Neural Networks for Object Detection in PolSAR data, EUSAR, 2010
Zhang et al., 2017 Zhang, Wang, Xu, and Jin, Complex-valued convolutional neural networksand its applications to PolSAR image classification, IEEE TGRS, 2017
Zhu et al. Zhu, Tuia, Mou, Xia, Zhang, Xu, and Fraundorfer. Deep Learning in Remote Sensing: A Review, GRSM, 2017

Presentation Layout

Presentation Layout

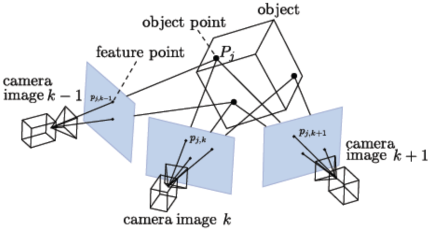
Capturing a 3D world

- 3D data crucial for robotics, autonomous vehicle, 3D scale models, virtual reality etc...

credit: medium, VisionSystemDesign, microsoft

Capturing a 3D world

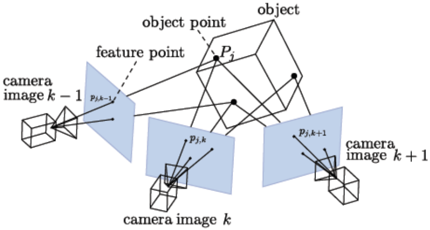
- 3D data crucial for robotics, autonomous vehicle, 3D scale models, virtual reality etc...
- Can be computed from images: stereo, SfM, SLAM (cheap, not precise).



credit: computervisionblog, velodynelidar

Capturing a 3D world

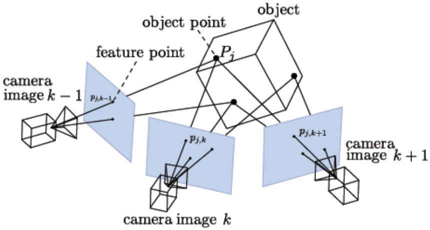
- 3D data crucial for robotics, autonomous vehicle, 3D scale models, virtual reality etc...
- Can be computed from images: stereo, SfM, SLAM (cheap, not precise).
- LiDAR (expensive, precise).



credit: computervisionblog, velodynelidar

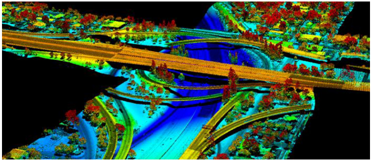
Capturing a 3D world

- 3D data crucial for robotics, autonomous vehicle, 3D scale models, virtual reality etc...
- Can be computed from images: stereo, SfM, SLAM (cheap, not precise).
- LiDAR (expensive, precise).
- Can be fixed, mobile, aerial, drone-embarked.



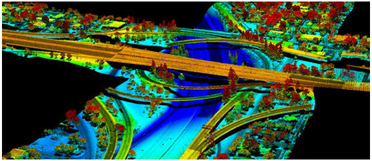
credit: computervisionblog, velodynelidar

- 3D data crucial for robotics, autonomous vehicle, 3D scale models, virtual reality etc...
- Can be computed from images: stereo, SfM, SLAM (cheap, not precise).
- LiDAR (expensive, precise).
- Can be fixed, mobile, aerial, drone-embarked.
- Produces a 3D point cloud: $P \in \mathbb{R}^{n \times 3}$.



credit: clearpath robotics, tuck mapping solutions

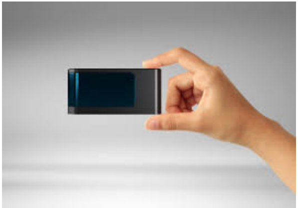
- 3D data crucial for robotics, autonomous vehicle, 3D scale models, virtual reality etc...
- Can be computed from images: stereo, SfM, SLAM (cheap, not precise).
- LiDAR (expensive, precise).
- Can be fixed, mobile, aerial, drone-embarked.
- Produces a 3D point cloud: $P \in \mathbb{R}^{n \times 3}$.
- Large acquisition: n typically in the 10^8s.



credit: clearpath robotics, tuck mapping solutions

Future trends

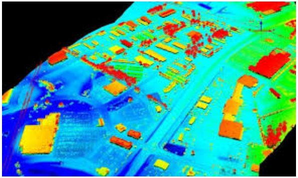
- LiDAR are getting cheaper :$100k\$ \rightarrow 2k\$$ in a few years.



credit: velodynelidar, green car congress

Future trends

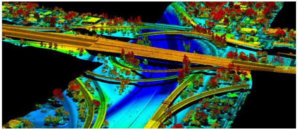
- LiDAR are getting cheaper :100k\$ $\rightarrow$ 2k\$ in a few years.
- Also coming: solid state LiDAR (cheap, fast and resilient), single photon LiDAR (unmatched acquisition density).



credit: velodynelidar, spar3d

Future trends

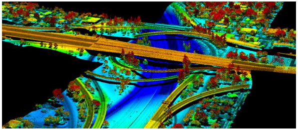
- LiDAR are getting cheaper :100k\$ $\rightarrow$ 2k\$ in a few years.
- Also coming: solid state LiDAR (cheap, fast and resilient), single photon LiDAR (unmatched acquisition density).
- **Major industrial application:** autonomous driving, virtual models, land survey...



credit: tuck mapping solutions, clearpath robotics

Future trends

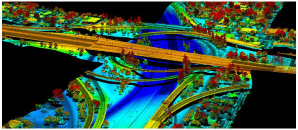
- LiDAR are getting cheaper :$100k\$ \rightarrow 2k\$$ in a few years.
- Also coming: solid state LiDAR (cheap, fast and resilient), single photon LiDAR (unmatched acquisition density).
- **Major industrial application:** autonomous driving, virtual models, land survey...
- **Also to come:** major advances in automatic analysis of 3D data.



credit: tuck mapping solutions, clearpath robotics

Future trends

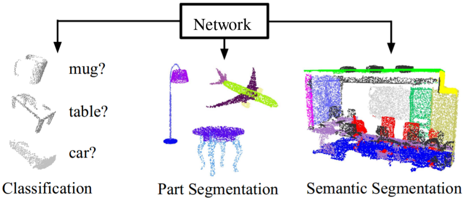
- LiDAR are getting cheaper :100k\$ $\rightarrow$ 2k\$ in a few years.
- Also coming: solid state LiDAR (cheap, fast and resilient), single photon LiDAR (unmatched acquisition density).
- **Major industrial application:** autonomous driving, virtual models, land survey...
- **Also to come:** major advances in automatic analysis of 3D data.
- Rapid progress in harware and methodology + major applications = **a booming field.**



credit: tuck mapping solutions, clearpath robotics

- **Classification**: classify the point cloud among class set $\mathcal{K}$:

$$P \mapsto \mathcal{K}$$



credit: Qi et. al. 2017a

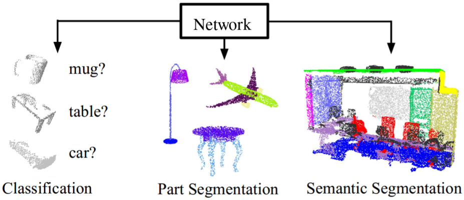
Analysis of 3D point clouds

- **Classification**: classify the point cloud among class set $\mathcal{K}$:

$$P \mapsto \mathcal{K}$$

- **Partition**: cluster the point cloud in C parts/object:

$$P_i \mapsto [1, \cdots, C]$$



credit: Qi et. al. 2017a

- **Classification**: classify the point cloud among class set $\mathcal{K}$:

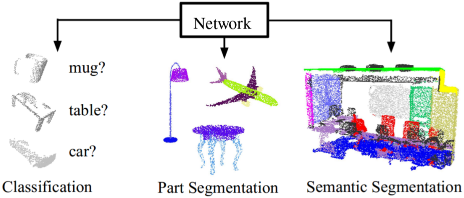
$$P \mapsto \mathcal{K}$$

- **Partition**: cluster the point cloud in C parts/object:

$$P_i \mapsto [1, \cdots, C]$$

- **Semantic Segmentation**: classify each point of a point cloud between K classes:

$$P_i \mapsto [1, \cdots, K]$$



credit: Qi et. al. 2017a

Analysis of 3D point clouds

- **Classification**: classify the point cloud among class set $\mathcal{K}$:

$$P \mapsto \mathcal{K}$$

- **Partition**: cluster the point cloud in C parts/object:

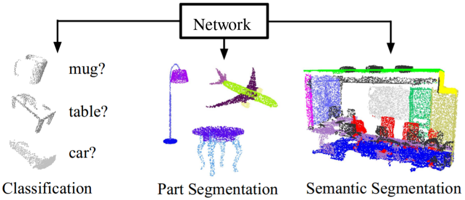
$$P_i \mapsto [1, \cdots, C]$$

- **Semantic Segmentation**: classify each point of a point cloud between K classes:

$$P_i \mapsto [1, \cdots, K]$$

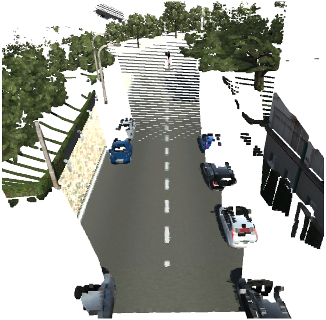
- **Instance Segmentation**: cluster the point cloud into semantically characterized objects:

$$P_i \mapsto [1, \cdots, C]$$
$$[1, \cdots, C] \mapsto [1, \cdots, K]$$



credit: Qi et. al. 2017a

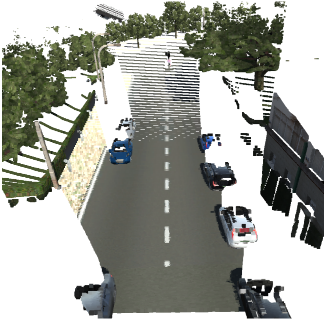
What makes 3D analysis so hard



- Data volume considerable.

credit: Gaidon2016, Engelmann2017, Hackel2017

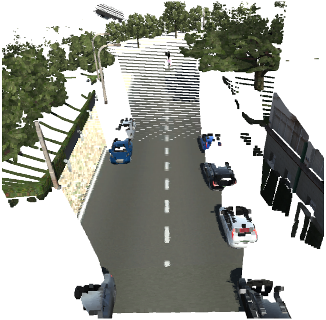
What makes 3D analysis so hard



- Data volume considerable.
- Lack of grid-structure.

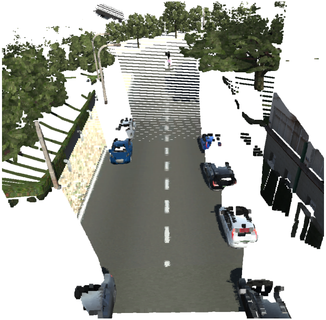
credit: Gaidon2016, Engelmann2017, Hackel2017

What makes 3D analysis so hard



- Data volume considerable.
- Lack of grid-structure.
- Permutation-invariance.

credit: Gaidon2016, Engelmann2017, Hackel2017

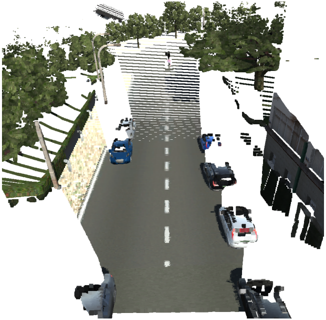


- Data volume considerable.
- Lack of grid-structure.
- Permutation-invariance.
- Sparsity.

credit: Gaidon2016, Engelmann2017, Hackel2017

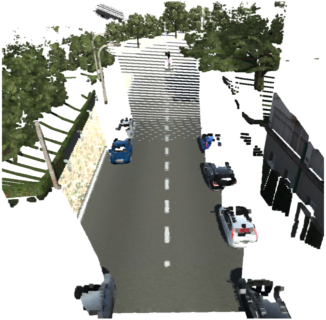
What makes 3D analysis so hard

- Data volume considerable.
- Lack of grid-structure.
- Permutation-invariance.
- Sparsity.
- Highly variable density.



credit: Gaidon2016, Engelmann2017, Hackel2017

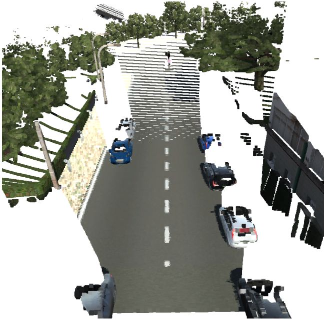
What makes 3D analysis so hard



- Data volume considerable.
- Lack of grid-structure.
- Permutation-invariance.
- Sparsity.
- Highly variable density.
- Acquisition artifacts.

credit: Gaidon2016, Engelmann2017, Hackel2017

What makes 3D analysis so hard



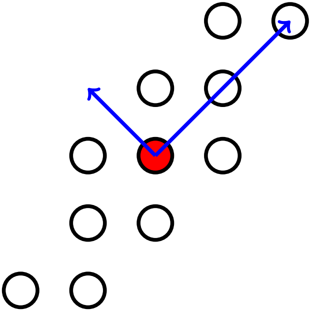
- Data volume considerable.
- Lack of grid-structure.
- Permutation-invariance.
- Sparsity.
- Highly variable density.
- Acquisition artifacts.
- Occlusions.

credit: Gaidon2016, Engelmann2017, Hackel2017

Presentation Layout

- **Step 1:** compute point features based on neighborhood

Demantke2011

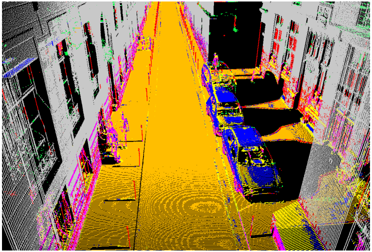


$$\text{Lin} = \frac{\sqrt{\lambda_1} - \sqrt{\lambda_2}}{\sqrt{\lambda_1}}$$

$$\text{Pla} = \frac{\sqrt{\lambda_2} - \sqrt{\lambda_3}}{\sqrt{\lambda_1}}$$

$$\text{Sca} = \frac{\sqrt{\lambda_3}}{\sqrt{\lambda_1}}$$

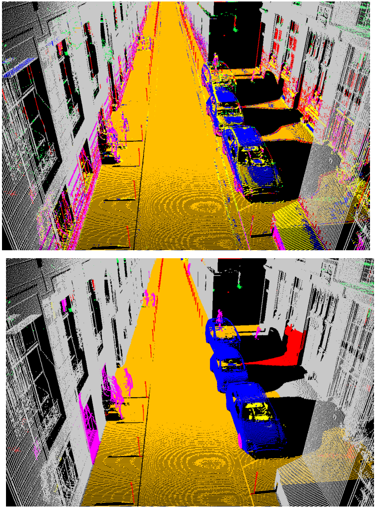
Pointwise classification



- **Step 1:** compute point features based on neighborhood
- **Step 2:** classification (RF, SVM, etc...)

Demantke2011
Weimann2015

credit: landrieu et. al. 2017a



- **Step 1:** compute point features based on neighborhood
- **Step 2:** classification (RF, SVM, etc...)
- **Step 3:** smoothing to increase spatial regularity (with CRFs, MRFs, graph-structured optimization, etc...)

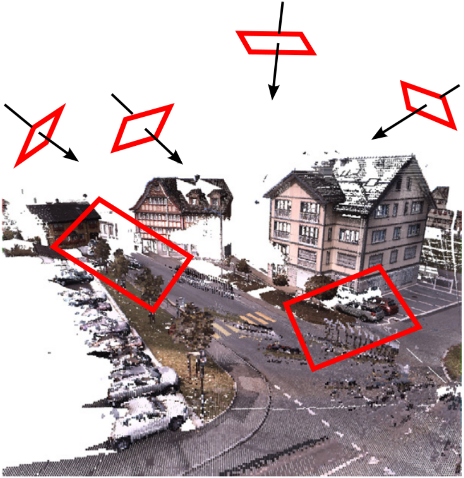
Demantke2011
Weimann2015
Landrieu et. al. 2017a

credit: landrieu et. al. 2017a

Presentation Layout

- **A simple observation:** CNNs works great for images. Can we use images for 3D?

- **A simple observation:** CNNs works great for images. Can we use images for 3D?
- **SnapNet:**



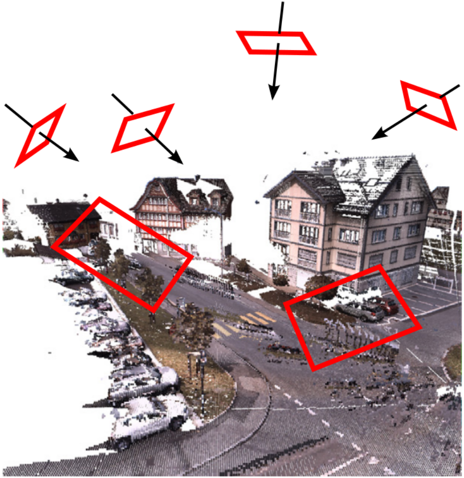
Boulch et. al. 2017

credit: Boulch et. al. 2017

Image-Based Methods

- **A simple observation:** CNNs works great for images. Can we use images for 3D?
- **SnapNet:**

\- surface reconstruction

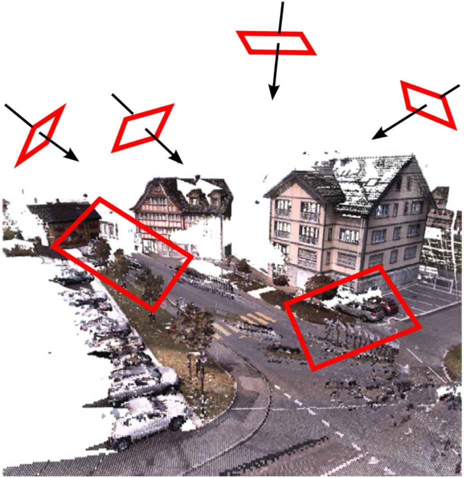


Boulch et. al. 2017

credit: Boulch et. al. 2017

- **A simple observation:** CNNs works great for images. Can we use images for 3D?
- **SnapNet:**
  - surface reconstruction
  - *virtual snapshots*

Boulch et. al. 2017

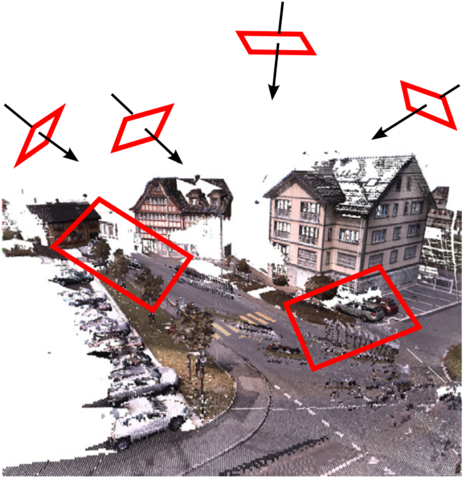


credit: Boulch et. al. 2017

Image-Based Methods

- **A simple observation:** CNNs works great for images. Can we use images for 3D?
- **SnapNet:**
  - surface reconstruction
  - *virtual snapshots*
  - semantic segmentation of resulting images with CNNs

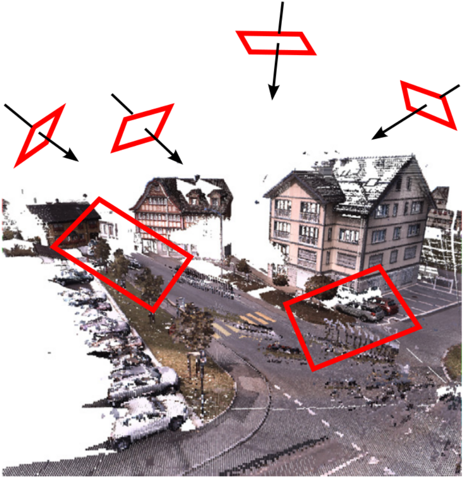
Boulch et. al. 2017



credit: Boulch et. al. 2017

- **A simple observation:** CNNs works great for images. Can we use images for 3D?
- **SnapNet:**
  - surface reconstruction
  - *virtual snapshots*
  - semantic segmentation of resulting images with CNNs
  - project prediction back to p.c.

Boulch et. al. 2017

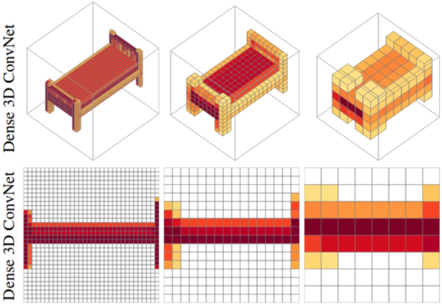


credit: Boulch et. al. 2017

- **Idea:** generalize 2D convolutions to regular 3D grids

Voxel-Based Methods

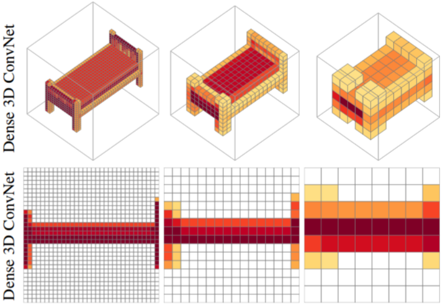
- **Idea:** generalize 2D convolutions to regular 3D grids
- Voxelization + 3D convNets



Wu2015

credit: Riegler2017, Tchapmi2017, Jampani2017

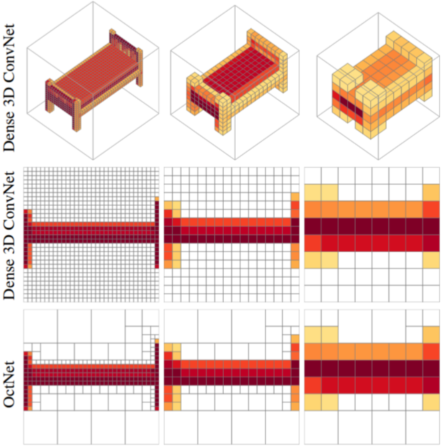
- **Idea:** generalize 2D convolutions to regular 3D grids
- Voxelization + 3D convNets
- **Problem:** inefficient representation, loss of invariance, costly (cubic)



Wu2015

credit: Riegler2017, Tchapmi2017, Jampani2017

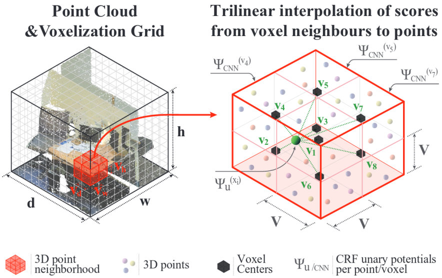
- **Idea:** generalize 2D convolutions to regular 3D grids
- Voxelization + 3D convNets
- **Problem:** inefficient representation, loss of invariance, costly (cubic)
- **Idea 1:** OctNet, OctTree based approach



Wu2015 , Riegler2017

credit: Riegler2017, Tchapmi2017, Jampani2017

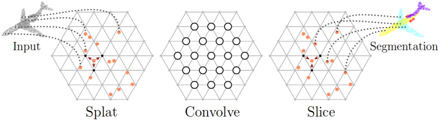
- **Idea:** generalize 2D convolutions to regular 3D grids
- Voxelization + 3D convNets
- **Problem:** inefficient representation, loss of invariance, costly (cubic)
- **Idea 1:** OctNet, OctTree based approach
- **Idea 2:** SegCloud, large voxels, subvoxel predictions with CRFs.



Wu2015 , Riegler2017 , Tchapmi2017, Jampani2018.

credit: Riegler2017, Tchapmi2017, Jampani2017

- **Idea:** generalize 2D convolutions to regular 3D grids
- Voxelization + 3D convNets
- **Problem:** inefficient representation, loss of invariance, costly (cubic)
- **Idea 1:** OctNet, OctTree based approach
- **Idea 2:** SegCloud, large voxels, subvoxel predictions with CRFs.
- **Idea 3:** SplatNet, sparse convolutions with hashmaps.



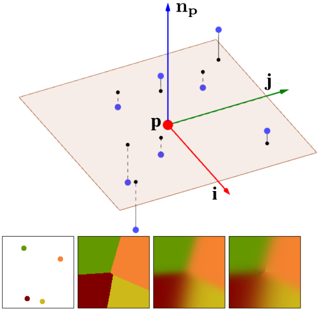
Wu2015 , Riegler2017 , Tchapmi2017, Jampani2018.

credit: Riegler2017, Tchapmi2017, Jampani2017

3D Convolution-Based Methods

- **Idea:** generalize 2D convolutions to 3D point clouds as unordered data.

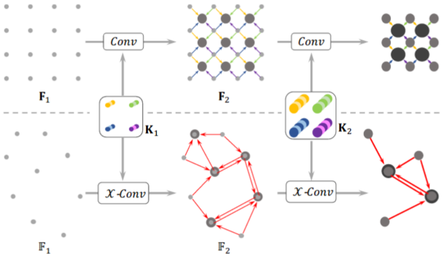
- **Idea:** generalize 2D convolutions to 3D point clouds as unordered data.
- **Tangent Convolution:** 2D convolution in the tangent space of each point.



Tatarchenko2018

credit: Tatarchenko2018, Li2018

- **Idea:** generalize 2D convolutions to 3D point clouds as unordered data.
- **Tangent Convolution:** 2D convolution in the tangent space of each point.
- **PointCNN :** χ-convolutions: generalized convolutions for unordered inputs.

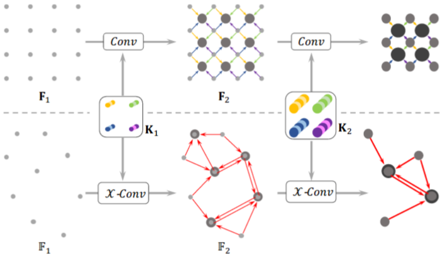


Tatarchenko2018 , Li2018.

credit: Tatarchenko2018, Li2018

3D Convolution-Based Methods

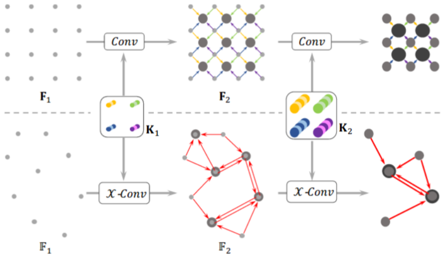
- **Idea:** generalize 2D convolutions to 3D point clouds as unordered data.
- **Tangent Convolution:** 2D convolution in the tangent space of each point.
- **PointCNN :** χ-convolutions: generalized convolutions for unordered inputs.
- **Principle:** the network learns how to permute *ordered* inputs



Tatarchenko2018 , Li2018.

credit: Tatarchenko2018, Li2018

- **Idea:** generalize 2D convolutions to 3D point clouds as unordered data.
- **Tangent Convolution:** 2D convolution in the tangent space of each point.
- **PointCNN :** χ-convolutions: generalized convolutions for unordered inputs.
- **Principle:** the network learns how to permute *ordered* inputs
- The invariance is learnt!

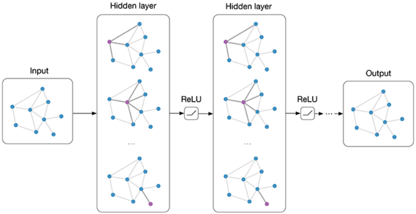


Tatarchenko2018 , Li2018.

credit: Tatarchenko2018, Li2018

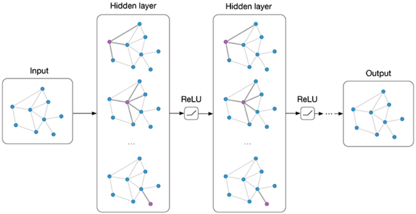
Graph-Neural Network

- Generalize convolutions to the general graph setting.



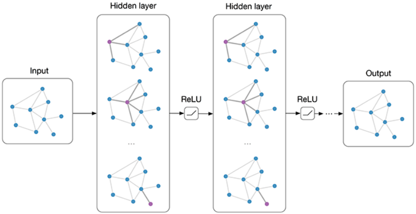
Graph-Neural Network

- Generalize convolutions to the general graph setting.
- For example: k-nearest neighbors graph of 3D points.



Qi2017, Simonovski2017

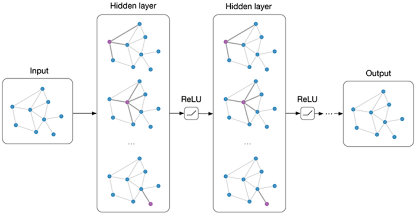
- Generalize convolutions to the general graph setting.
- For example: k-nearest neighbors graph of 3D points.
- **Idea:** Each point maintain a hidden state h_i influenced by its neighbors.



Qi2017, Simonovski2017

- Generalize convolutions to the general graph setting.
- For example: k-nearest neighbors graph of 3D points.
- **Idea:** Each point maintain a hidden state h_i influenced by its neighbors.
- GNN **Qi2017**: an iterative message-passing algorithm using a mapping f and a RNN g:

$$h_i^{(t+1)} = g(\sum_{j \to i} f(h_i^t), h_i^t)$$



Qi2017, Simonovski2017

credit: Qi2017

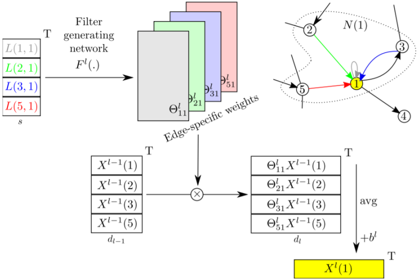
Graph-Neural Network

- Generalize convolutions to the general graph setting.
- For example: k-nearest neighbors graph of 3D points.
- **Idea:** Each point maintain a hidden state h_i influenced by its neighbors.
- GNN **Qi2017**: an iterative message-passing algorithm using a mapping f and a RNN g:

$$h_i^{(t+1)} = g(\sum_{j \to i} f(h_i^t), h_i^t)$$

- ECC **Simonovski2017** messages are conditioned by edge features:

$$h_i^{(t+1)} = g(\sum_{j \to i} \Theta_{i,j} \odot h_i^t, h_i^t)$$



Qi2017, Simonovski2017

credit: Simonovski2017

Presentation Layout

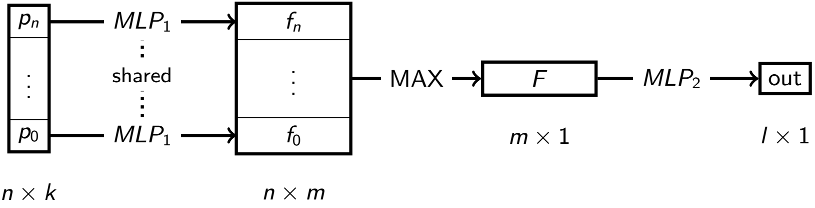
- A cornerstone of modern 3D analysis

Qi et. al.2017a

- A cornerstone of modern 3D analysis
- **A fondamental constraint:** inputs are invariant by permutation

Qi et. al.2017a

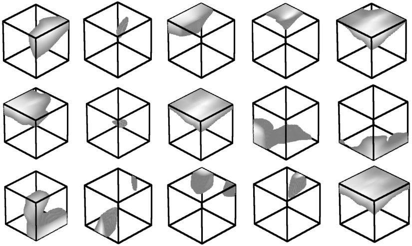
- A cornerstone of modern 3D analysis
- **A fondamental constraint:** inputs are invariant by permutation
- **Solution:** process points independently, apply permutation-invariant pooling, process this feature with a MLP.
- Computes a global shape descriptor.
- n: number of points, k size of observations, $e^{(i)}$ size of intermediary embeddings, $e^{(f)}$ size of output



Qi et. al.2017a

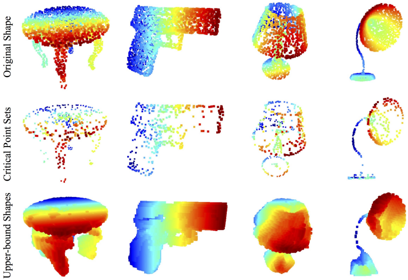
PointNet II

- **Point function**: activate at different parts of the unit cube, learned spatial features.



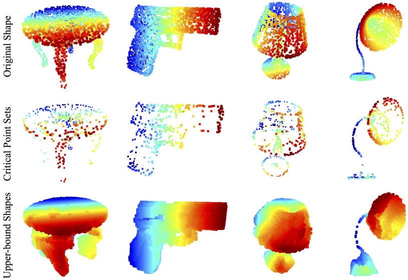
PointNet II

- **Point function**: activate at different parts of the unit cube, learned spatial features.
- **Critical Set**: points selected in the maxpool step. makes up a *skeletton*



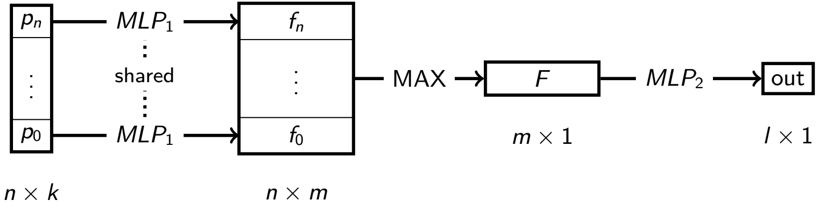
PointNet II

- **Point function**: activate at different parts of the unit cube, learned spatial features.
- **Critical Set**: points selected in the maxpool step. makes up a *skeletton*
- **Upper Bound Shape**: maximal point cloud with exactly the same global embedding



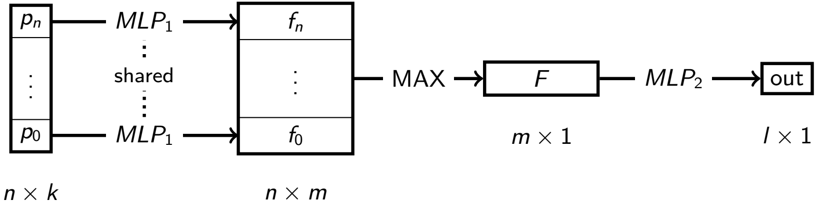
PointNet for Cloud embedding

- `point_input`$_i$: point features (coordinate, color, etc...)



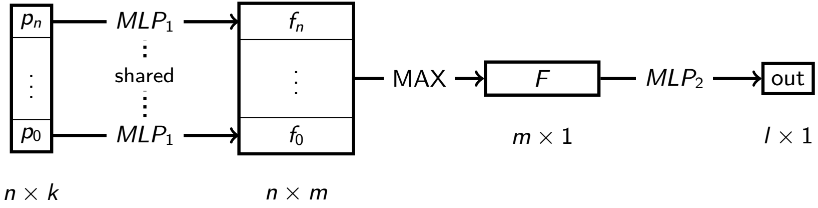
Qi et. al.2017a

- point_input$_i$: point features (coordinate, color, etc...)
- point_embedding$_i$ = MLP_1(point_input$_i$)



Qi et. al.2017a

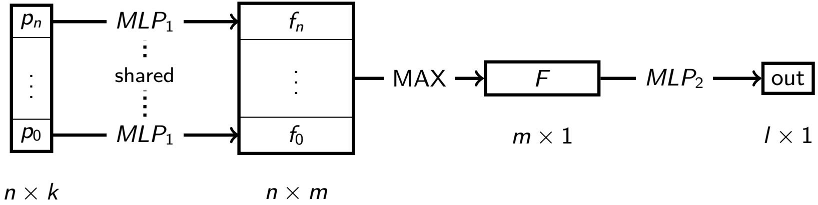
- `point_input`$_i$: point features (coordinate, color, etc...)
- `point_embedding`$_i$ $= MLP_1($`point_input`$_i)$
- `global_embeding` $= max_i($`point_embedding`$_i)$



Qi et. al.2017a

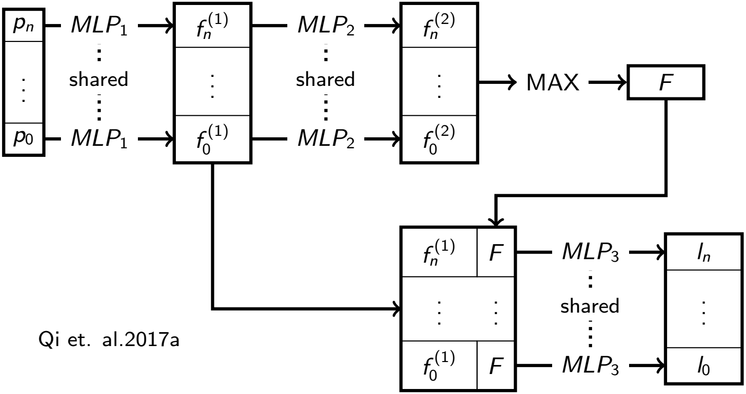
PointNet for Cloud embedding

- $\texttt{point\_input}_i$: point features (coordinate, color, etc...)
- $\texttt{point\_embedding}_i = MLP_1(\texttt{point\_input}_i)$
- $\texttt{global\_embeding} = max_i(\texttt{point\_embedding}_i)$
- $\texttt{cloud\_embeding} = MLP_2(\texttt{global\_embedding})$



Qi et. al.2017a

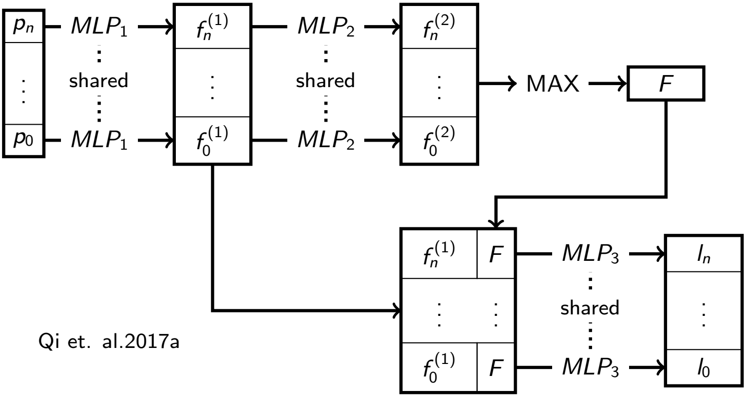
- $\texttt{point\_embedding}_i^{(1)} = MLP_1(\texttt{point\_input}_i)$



Qi et. al.2017a

PointNet for Semantic Segmentation

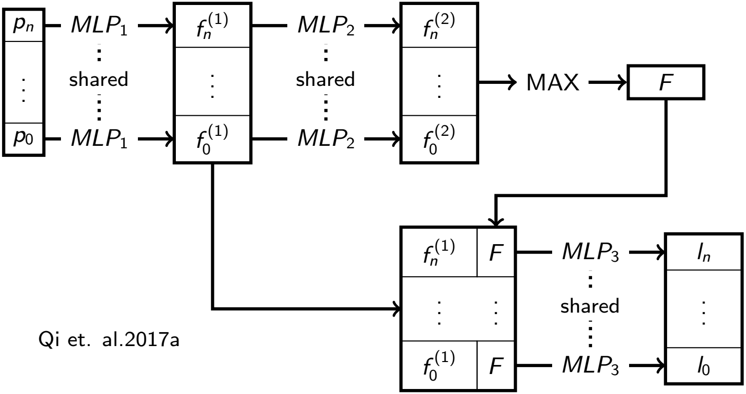
- $\texttt{point\_embedding}_i^{(1)} = MLP_1(\texttt{point\_input}_i)$
- $\texttt{point\_embedding}_i^{(2)} = MLP_2(\texttt{point\_embedding}_i^{(1)})$



Qi et. al.2017a

PointNet for Semantic Segmentation

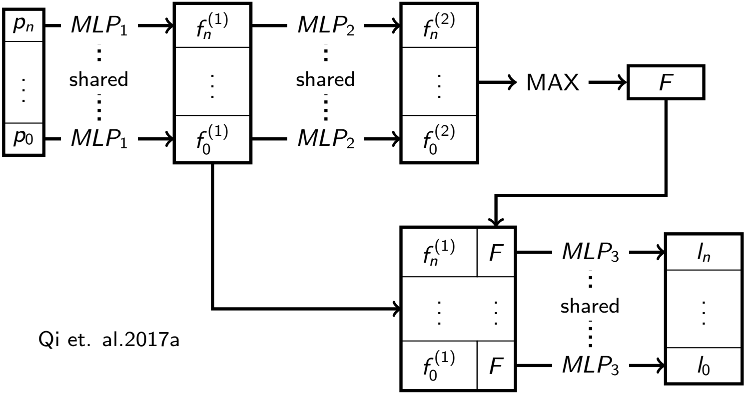
- $\texttt{point\_embedding}_i^{(1)} = MLP_1(\texttt{point\_input}_i)$
- $\texttt{point\_embedding}_i^{(2)} = MLP_2(\texttt{point\_embedding}_i^{(1)})$
- $\texttt{global\_embeding} = max_i(\texttt{point\_embedding}_i^{(2)})$



Qi et. al.2017a

PointNet for Semantic Segmentation

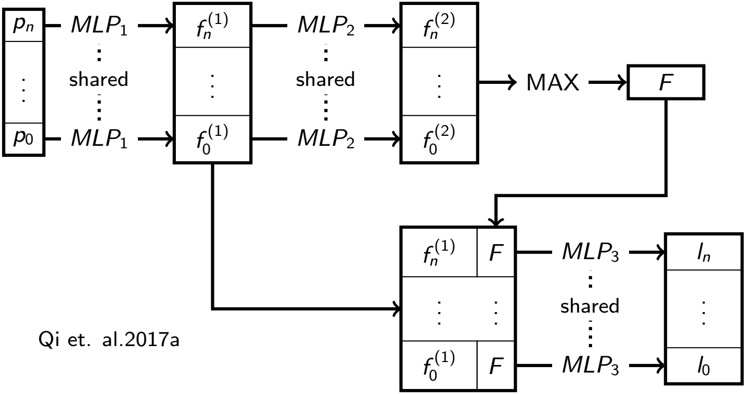
- $\texttt{point\_embedding}_i^{(1)} = MLP_1(\texttt{point\_input}_i)$
- $\texttt{point\_embedding}_i^{(2)} = MLP_2(\texttt{point\_embedding}_i^{(1)})$
- $\texttt{global\_embeding} = max_i(\texttt{point\_embedding}_i^{(2)})$
- $\texttt{point\_logit}_i = MLP_3([\texttt{global\_embedding}, \texttt{point\_embedding}_i^{(1)}])$



Qi et. al.2017a

PointNet for Semantic Segmentation

- $\texttt{point\_embedding}_i^{(1)} = MLP_1(\texttt{point\_input}_i)$
- $\texttt{point\_embedding}_i^{(2)} = MLP_2(\texttt{point\_embedding}_i^{(1)})$
- $\texttt{global\_embeding} = max_i(\texttt{point\_embedding}_i^{(2)})$
- $\texttt{point\_logit}_i = MLP_3([\texttt{global\_embedding}, \texttt{point\_embedding}_i^{(1)}])$
- $\texttt{point\_classif}_i = \texttt{softmax}(\texttt{point\_logit}_i)$

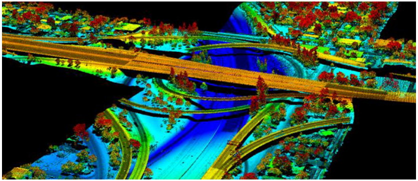


Qi et. al.2017a

Presentation Layout

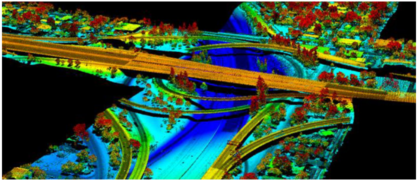
Why we need to scale

- **Problem:** best approaches are very memory-hungry and the data volumes are huge.



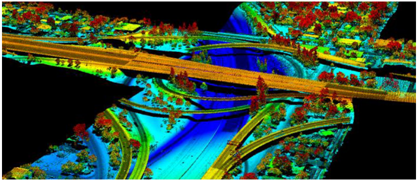
Why we need to scale

- **Problem:** best approaches are very memory-hungry and the data volumes are huge.
- Previous methods only works with a few thousands points.



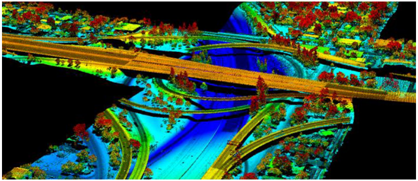
Why we need to scale

- **Problem:** best approaches are very memory-hungry and the data volumes are huge.
- Previous methods only works with a few thousands points.
- **Naive strategies:**
  - **Aggressive subsampling**: loses a lot of information.



Why we need to scale

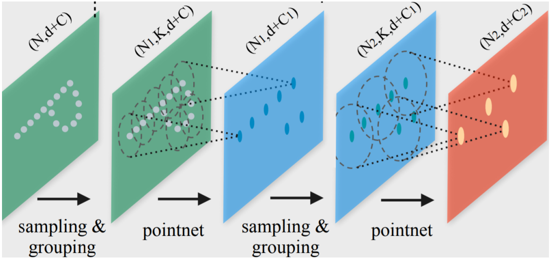
- **Problem:** best approaches are very memory-hungry and the data volumes are huge.
- Previous methods only works with a few thousands points.
- **Naive strategies:**
  - **Aggressive subsampling**: loses a lot of information.
  - **Sliding windows**: loses the global structure.



credit: tuck mapping solution

PointNet++

- Pyramid structure for multi-scale feature extraction.

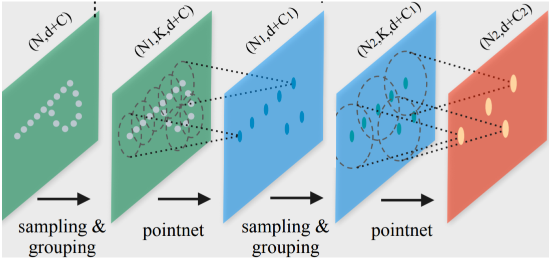


Qi et. al.2017b

credit: Qi et. al.2017b

PointNet++

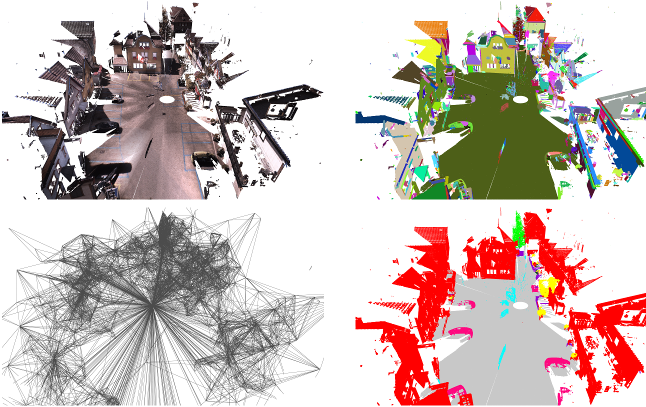
- Pyramid structure for multi-scale feature extraction.
- From local to global with with increasingly abstract features.



Qi et. al.2017b

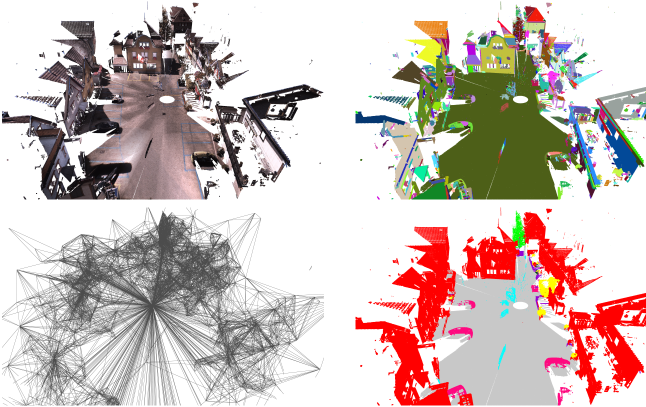
credit: Qi et. al.2017b

- **Observation:**
  $n_{\text{points}} \gg n_{\text{objects}}$.



Landrieu&Simonovski2018

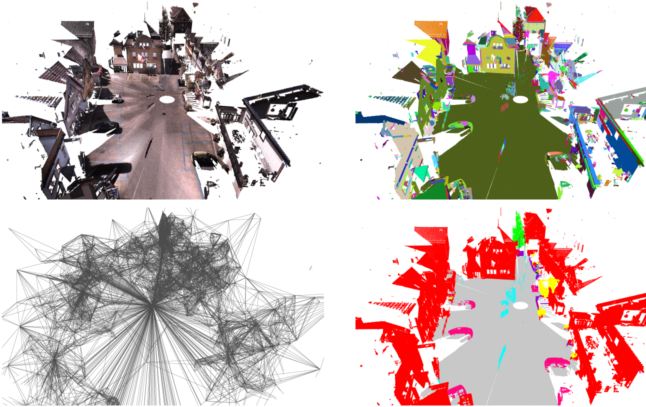
- **Observation:** $n_{\text{points}} \gg n_{\text{objects}}$.
- Partition scene into superpoints with simple shapes.



Landrieu&Simonovski2018

SuperPoint-Graph

- **Observation:**
  $n_{\text{points}} \gg n_{\text{objects}}$.
- Partition scene into superpoints with simple shapes.
- Only a few superpoints, context leveraging with powerful graph methods.

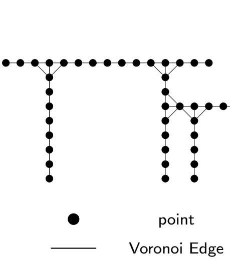


Landrieu&Simonovski2018

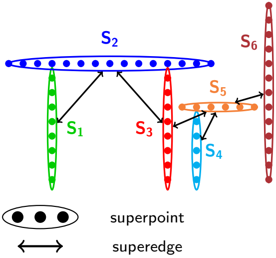
Pipeline

Step	Complexity	Algorithm
Geometric Partition into simple shapes	very high 10^8 points	ℓ_0-cut pursuit
Superpoint embedding learning shape descriptors	low subsampling to 128 points	PointNet
Contextual Segmentation leveraging the global structure	very low $\sim$ 1000 vertices	ECC with GRUs

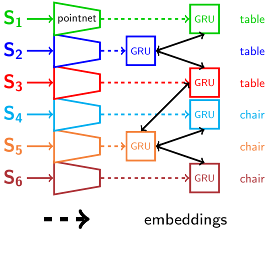
Pipeline



(a) Point cloud



(b) Superpoint graph



(c) Convolution Network

Presentation Layout

Main Annotated Academic Datasets

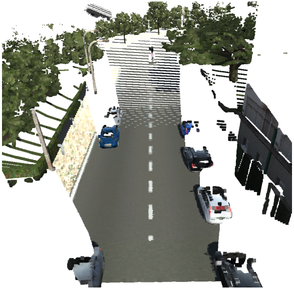
- **Semantic3D**: outdoor, fixed LiDAR, 4×10^9 points

credit: Armeni2016, Gaidon2016, Engelmann2017, Hackel2017

- **Semantic3D**: outdoor, fixed LiDAR, 4×10^9 points
- **S3DIS**: indoor, fixed LiDAR, 6×10^8 points

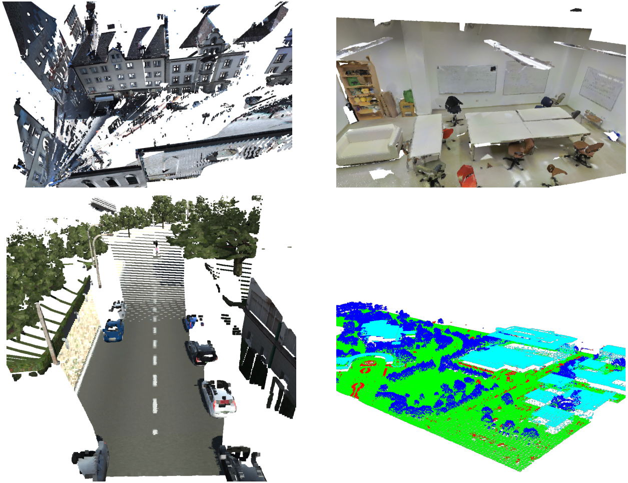
credit: Armeni2016, Gaidon2016, Engelmann2017, Hackel2017

- **Semantic3D**: outdoor, fixed LiDAR, 4×10^9 points
- **S3DIS**: indoor, fixed LiDAR, 6×10^8 points
- **vKITTI**: embarked, virtual LiDAR, 2×10^8 points.



credit: Armeni2016, Gaidon2016, Engelmann2017, Hackel2017

- **Semantic3D**: outdoor, fixed LiDAR, 4×10^9 points
- **S3DIS**: indoor, fixed LiDAR, 6×10^8 points
- **vKITTI**: embarked, virtual LiDAR, 2×10^8 points.
- **GRSS**: aerial LiDAR, $\sim 10^7$ points.



credit: Armeni2016, Gaidon2016, Engelmann2017, Hackel2017

In practice

- **Which one is the best?** Depends on your problem.

In practice

- **Which one is the best?** Depends on your problem.
- PointCNN works very well for smaller clouds
- SPG for large problems with a global structure

In practice

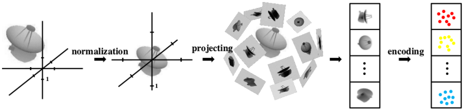
- **Which one is the best?** Depends on your problem.
- PointCNN works very well for smaller clouds
- SPG for large problems with a global structure
- New contenders weekly: PointSIFT, etc...
- **Which algorithm to chose in practice:**
- Start with a simple PointNet + sliding window baseline
- Move on to PointCNN/PointSIFT if precision is an issue
- Move on to SPG for scaling/global structure.

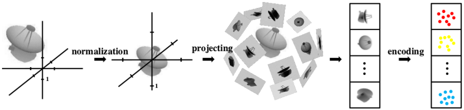
In practice

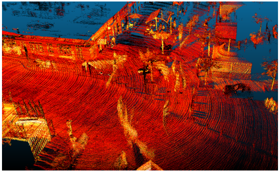
- **Which one is the best?** Depends on your problem.
  - PointCNN works very well for smaller clouds
  - SPG for large problems with a global structure
  - New contenders weekly: PointSIFT, etc...
- **Which algorithm to chose in practice:**
  - Start with a simple PointNet + sliding window baseline
  - Move on to PointCNN/PointSIFT if precision is an issue
  - Move on to SPG for scaling/global structure.
- Pay attention near big conferences (CVPR, ICCV), new models released constantly.

Future of the Field

- Efficient auto-encoders for semi-supervized learning **Zhu2016**

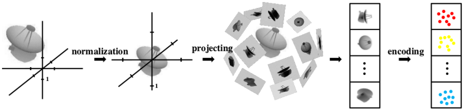


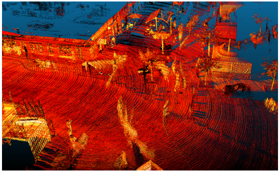


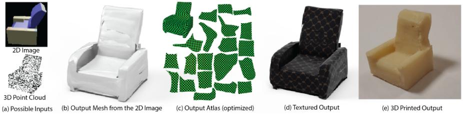


- Efficient auto-encoders for semi-supervized learning **Zhu2016**
- Real-time analysis for dynamic 3D data for autonomous driving

- Efficient auto-encoders for semi-supervized learning **Zhu2016**
- Real-time analysis for dynamic 3D data for autonomous driving
- Deep learning for other remote sensing tasks: segmentation, object detection, surface reconstruction. **Groueix2018**







Presentation Layout

1. Presentation of Speakers
2. Deep Learning Basics
3. Deep Learning on Raster Imagery
4. Deep Learning on 3D Point Clouds
   - Presentation of the Problem
   - Traditional Approaches
   - First Deep-Learning Approaches
   - PointNet - set-based approach
   - Scaling Segmentation
   - In Practice
   - Bibliography

Qi et. al.2017a Qi, C. R., Su, H., Mo, K., & Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. CVPR, 2017

Gaidon2016 Gaidon, A., Wang, Q., Cabon, Y., & Vig, E. Virtual worlds as proxy for multi-object tracking analysis. ,CVPR2016.

Engelmann2017 Engelmann, F., Kontogianni, T., Hermans, A. & Leibe, B. Exploring spatial context for 3d semantic segmentation of point clouds. CVPR, DRMS Workshop, 2017.

Hackel2017i Timo Hackel and N. Savinov and L. Ladicky and Jan D. Wegner and K. Schindler and M. Pollefeys, SEMANTIC3D.NET: A new large-scale point cloud classification benchmark,ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences,2017

Armeni2016 Iro Armeni and Ozan Sener and Amir R. Zamir and Helen Jiang and Ioannis Brilakis and Martin Fischer and Silvio Savarese, 3D Semantic Parsing of Large-Scale Indoor Spaces, CVPR, 2016

Demantke2011 Demantke, J., Mallet, C., David, N. & Vallet, B. Dimensionality based scale selection in 3D lidar point clouds. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2011.

Bibliography II

Weinmann2015 Weinmann, M., Jutzi, B., Hinz, S. & Mallet, C., Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. ISPRS Journal of Photogrammetry and Remote Sensing, 2015.
Landrieu2017a Landrieu, L., Raguet, H., Vallet, B., Mallet, C., & Weinmann, M. A structured regularization framework for spatially smoothing semantic labelings of 3D point clouds. ISPRS Journal of Photogrammetry and Remote Sensing, 2017.
Boulch2017 Boulch, Alexandre, Le Saux, Bertrand, and Audebert, Nicolas, Unstructured Point Cloud Semantic Labeling Using Deep Segmentation Networks, 3DOR, 2017.
Wu2015 Wu, Z., Song, S. Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. 3D shapenets: A deep representation for volumetric shapes. CVPR, 2015.
Riegler2017 Riegler, G., Osman Ulusoy, A., & Geiger, A. Octnet: Learning deep 3d representations at high resolutions, CVPR, 2017.
Tchapmi2017 Tchapmi, L., Choy, C., Armeni, I., Gwak, J. & Savarese, S., Segcloud: Semantic segmentation of 3d point clouds. 3DV, 2017.
Jampani2018, Jampani Su, H., , V. Sun, D., Maji, S., Kalogerakis, E., Yang, M. H., & Kautz, J., Splatnet: Sparse lattice networks for point cloud processing. CVPR2018.
Tatarchenko2018 Tatarchenko, M., Park, J., Koltun, V., & Zhou, Q. Y. Tangent Convolutions for Dense Prediction in 3D. CVPR, 2018
Li2018 Li, Y., Bu, R., Sun, M., Wu, W., Di, X., & Chen, B. PointCNN: Convolution On χ-Transformed Points. NIPS, 2018.

Bibliography III

Qi2017 Qi, X., Liao, R., Jia, J., Fidler, S., & Urtasun, R. 3D Graph Neural Networks for RGBD Semantic Segmentation. In PCVPR, 2017.
Simonovsky2017 Simonovsky, M., & Komodakis, N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. CVPR, 2017.
Landrieu&Simonovski2018 Landrieu, L., & Simonovsky, M. Large-scale point cloud semantic segmentation with superpoint graphs. CVPR, 2018
Qi et. al.2017b Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Advances in Neural Information Processing Systems (pp. 5099-5108).
Zhu2016 Zhu, Z., Wang, X., Bai, S., Yao, C., & Bai, X.. Deep learning representation using autoencoder for 3D shape retrieval. Neurocomputing, 2016.
Groueix2018 Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., & Aubry, M. (2018). A papier-mâché approach to learning 3d surface generation. CVPR, 2018.

Public resources

Today:

- Neural Net Basics
- Deep Learning in 2D Remote Sensing
- Deep Learning in 3D Remote Sensing

Thanks for you attention!

Comments and questions after the course:
`Loic.Landrieu@ign.fr` and `bertrand.le_saux@onera.fr`