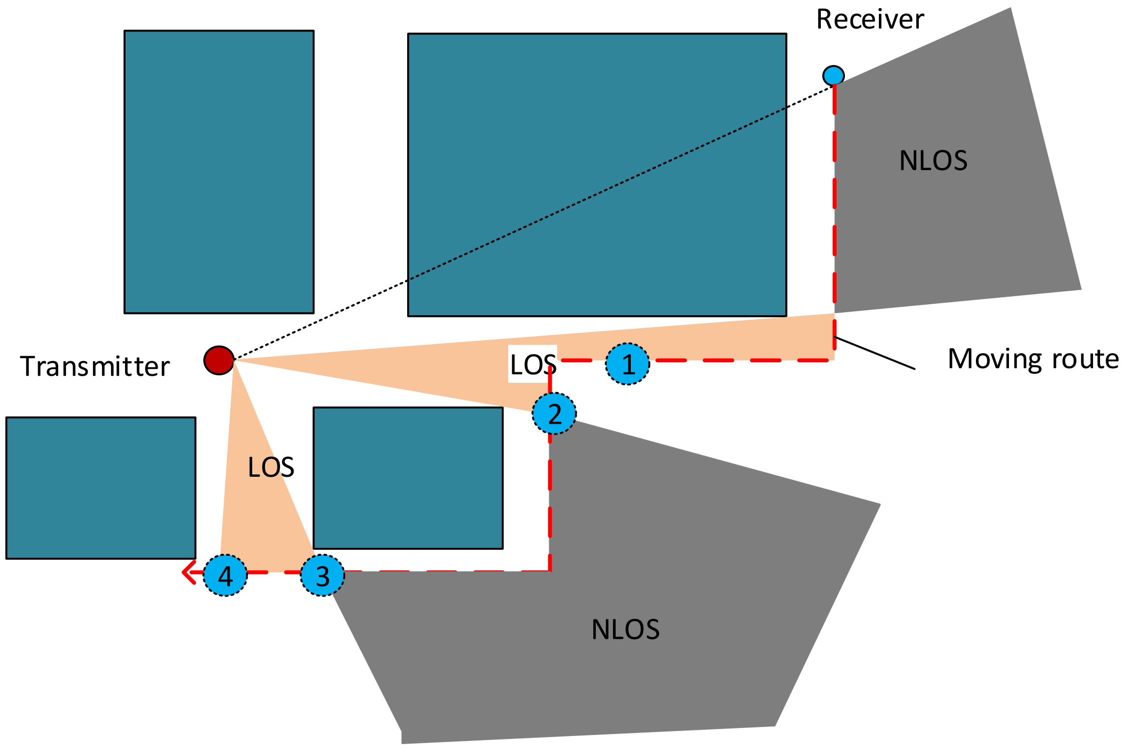
ML and CF for Channel Estimation

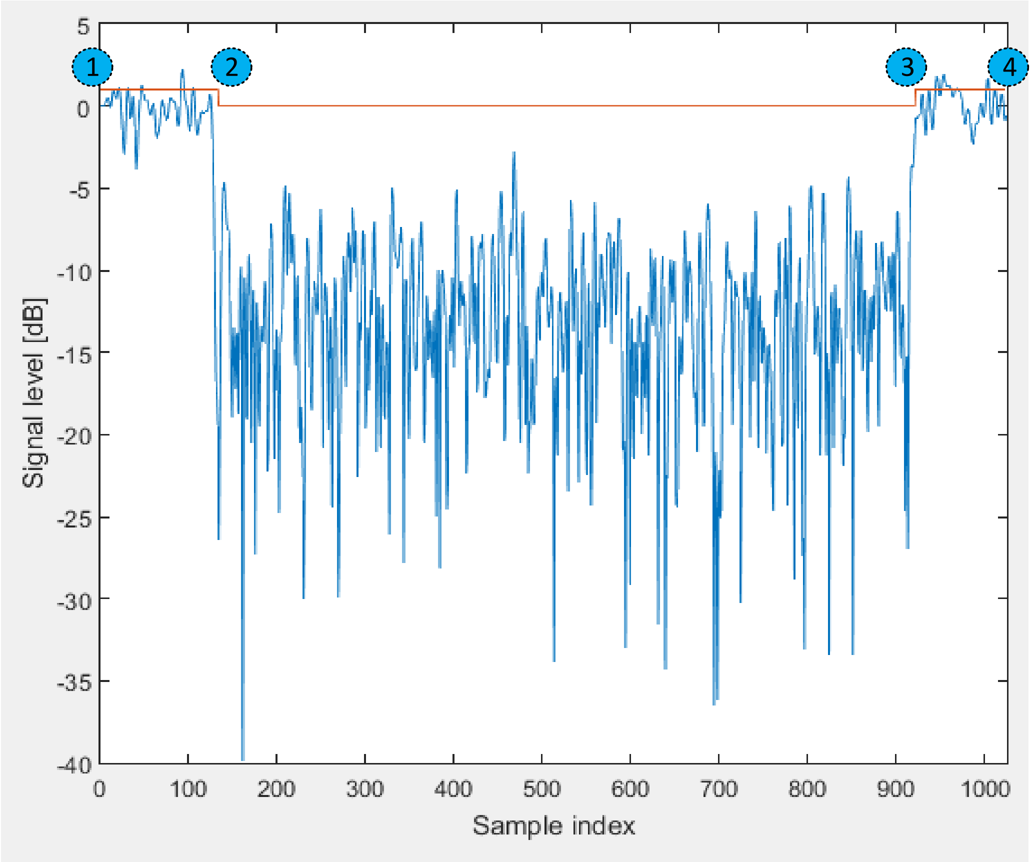
Duc-Tuyen Ta

Outline

- Recall the idea of Geographical channel similarity
- Generate Fading Channel Data
- Channel type segmentation
  - Reference: Semantic Segmentation
  - CNN based channel type segmentation
- Discussion:
  - Framework for the proposed channel estimation approach

Geographical channel similarity





Channel type segmentation for collaborative filtering

Generate Fading Channel Data

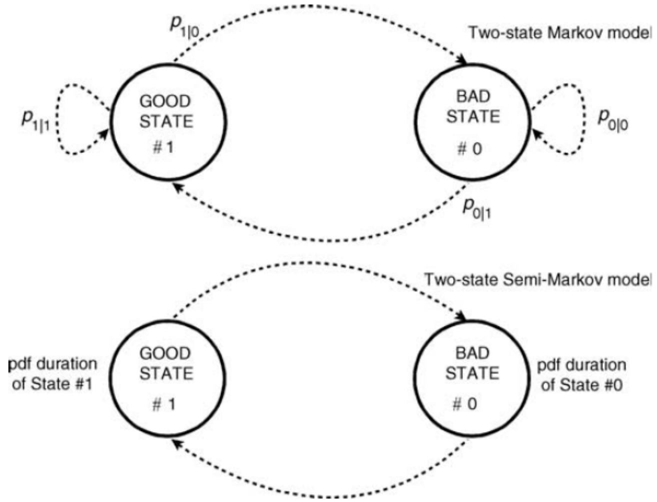


Figure 9.3 Markov vs. semi-Markov models

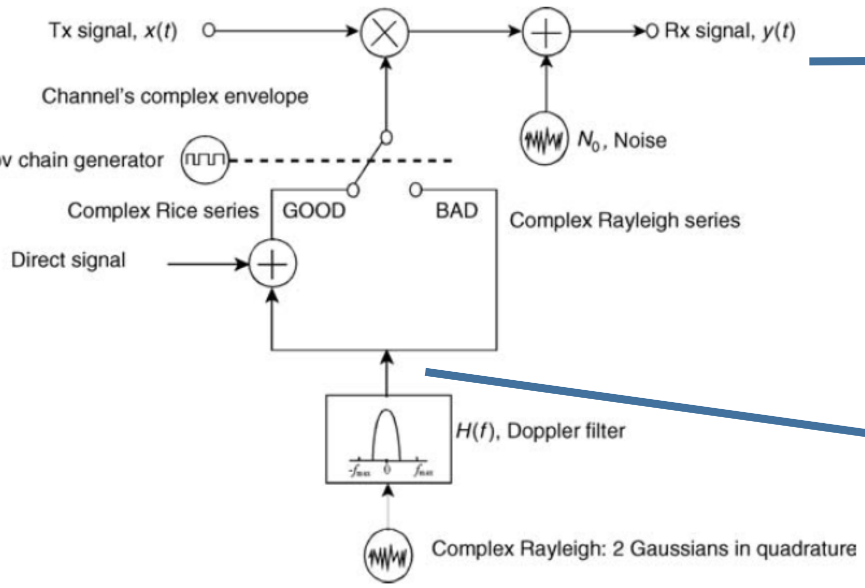
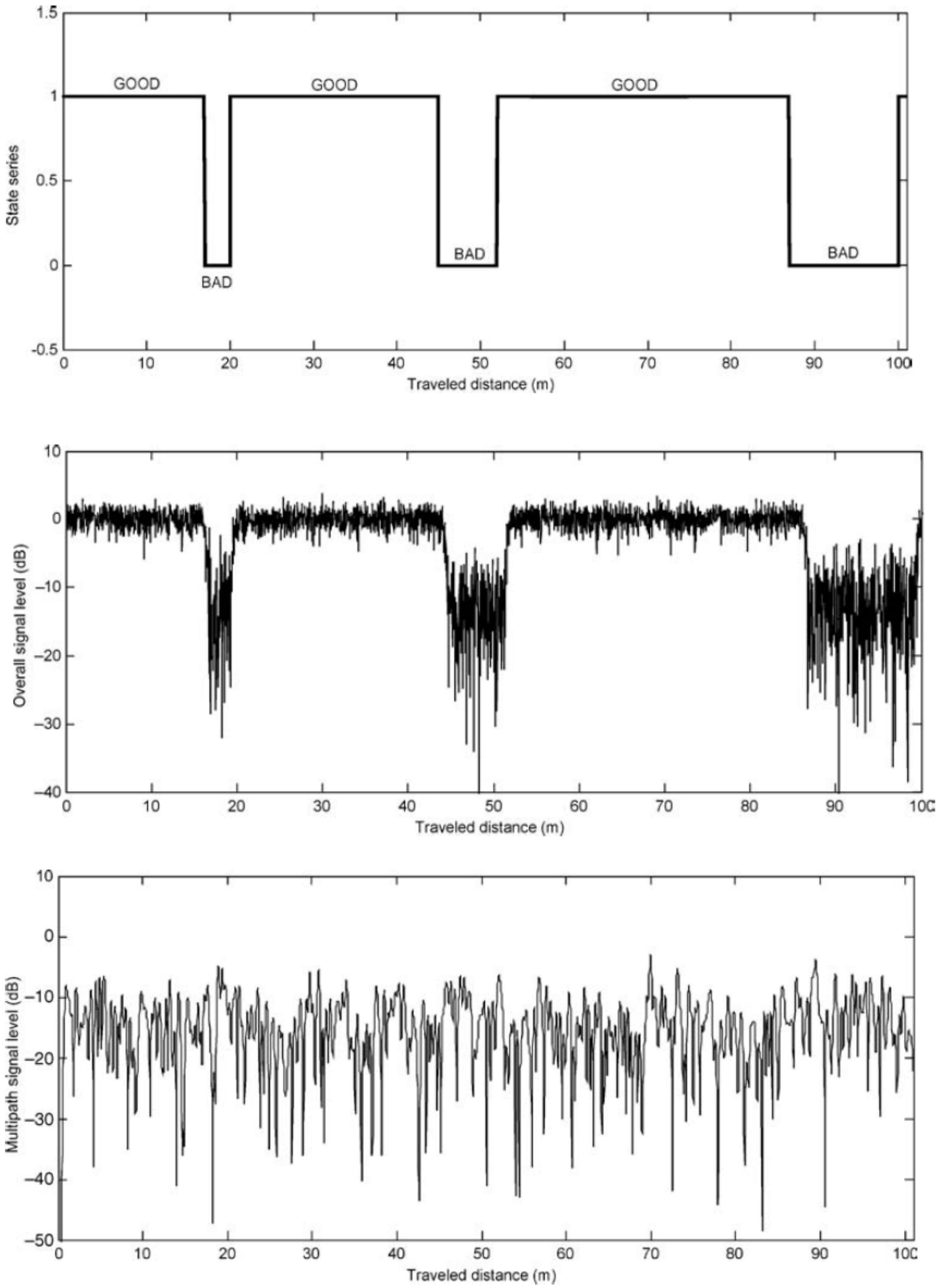


Figure 9.2 Two-state Markov plus Rayleigh/Rice LMS channel simulator [2]



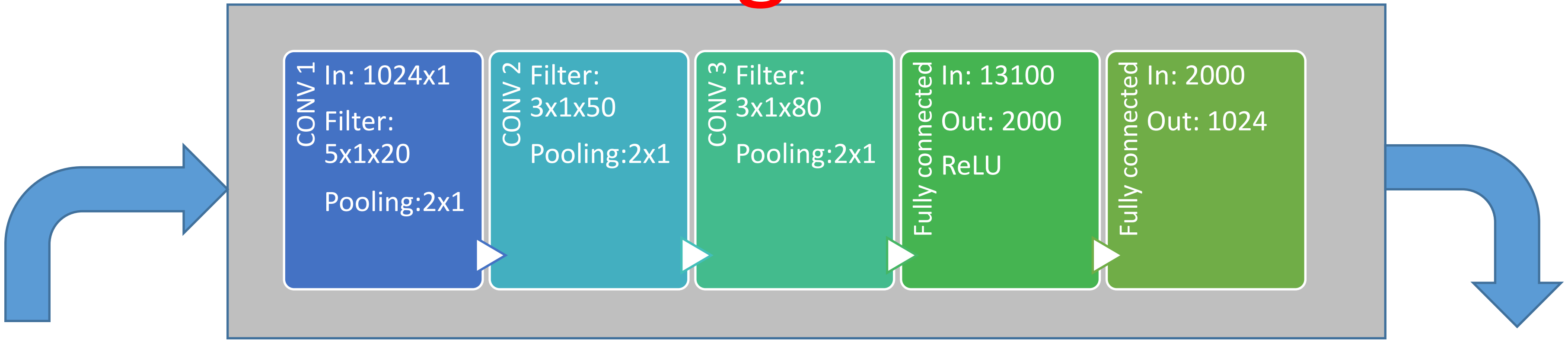
Generate Fading Channel Data

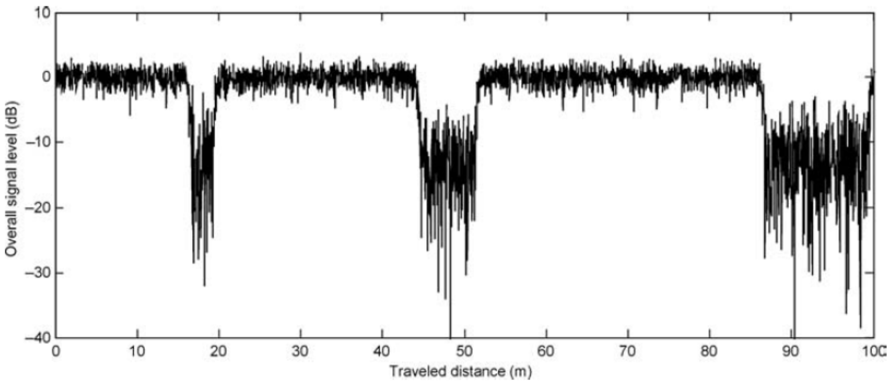
- No sample=1024/2048;
- Distance = 100/40 m;
- f=1500:15:2500 MHz
- Sigma_Rayleigh = 0.1:0.05:0.5
  - Sigma_Rice=Sigma_Rayleigh*(.4+0.6*rand)
- P00 = 0.5:0.05:0.95
- P11 = 0.5:0.05:0.95

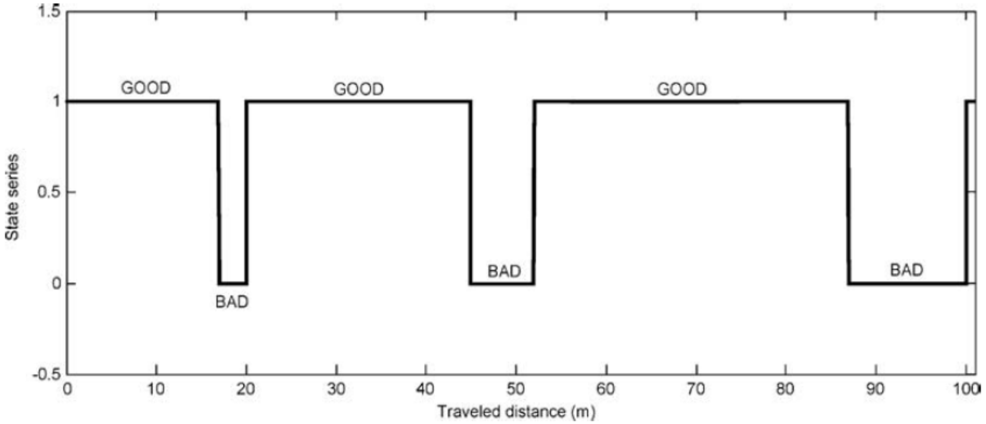
→ 60300 samples = 50k for training +10 k for testing

CNN for channel type segmentation

Not good?







Fully Convolutional Networks for Semantic Segmentation

Ref: https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

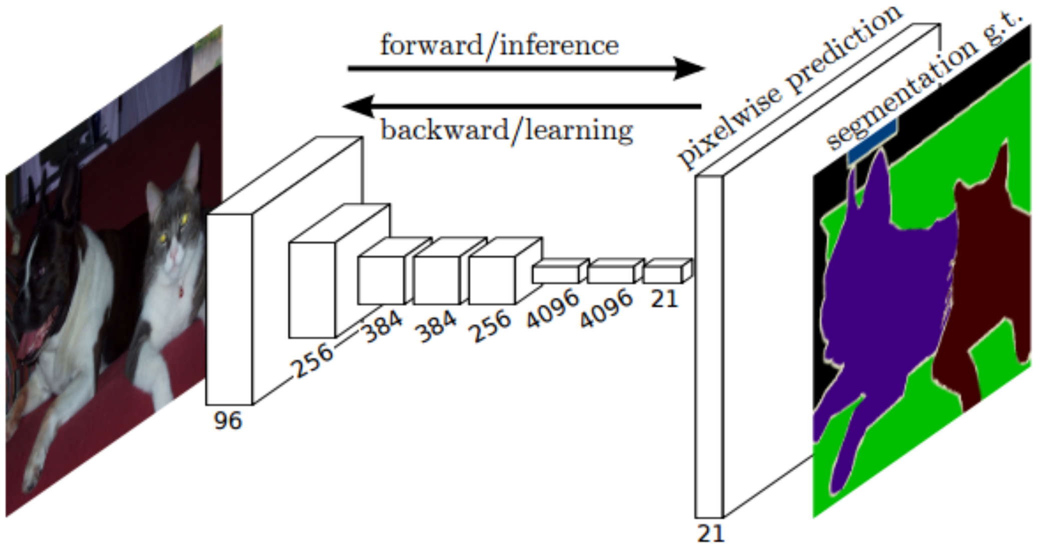


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

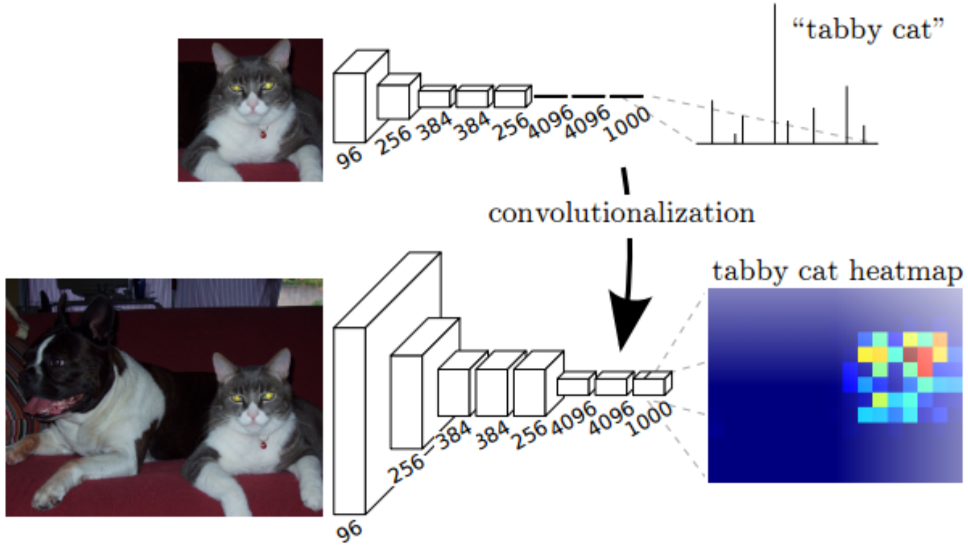
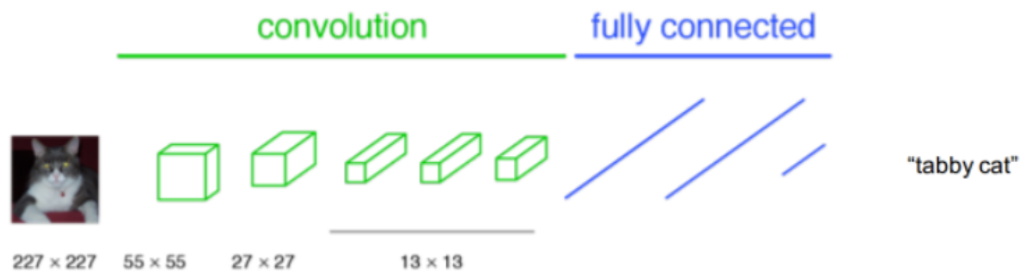
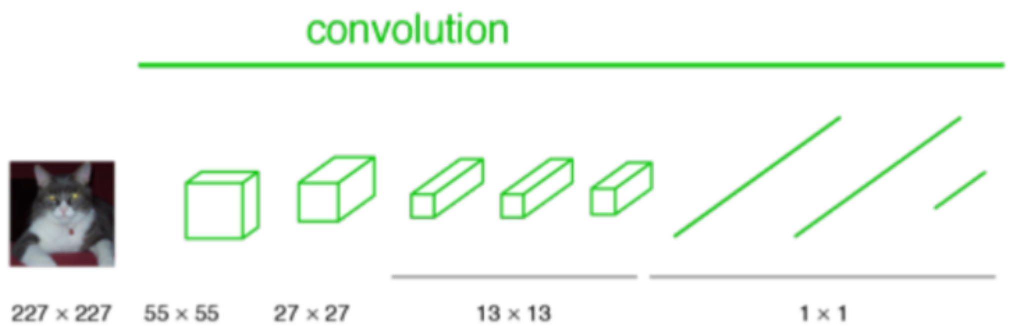


Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end dense learning.

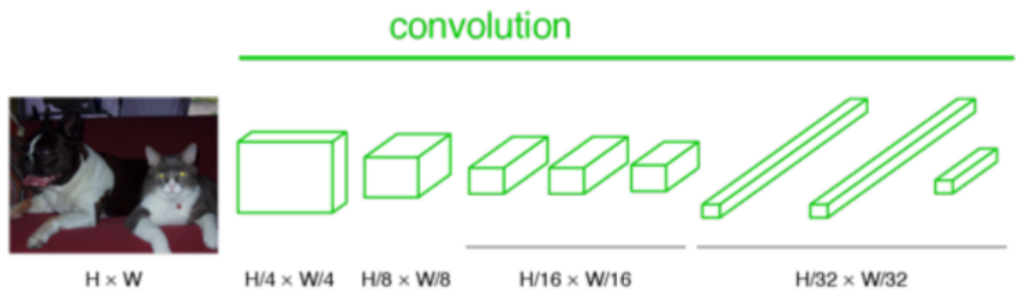
1) We start with a normal CNN for classification with



2) The second step is to convert all the FC layers to convolution layers 1x1 we don't even need to change the weights at this point. (This is already a fully convolutional neural network). The nice property of FCN networks is that we can now use any image size.

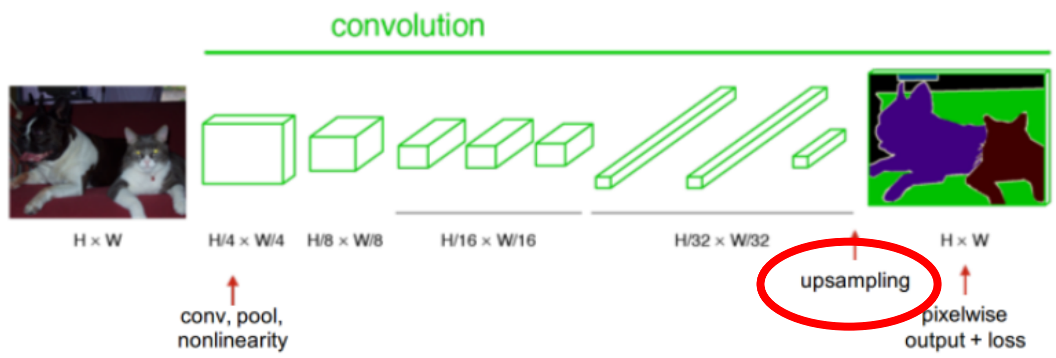


Observe here that with a FCN we can use a different size H x N. The diagram bellow show a how a different size would appear



3) The last step is to use a "deconv or transposed convolution" layer to recover the activation positions to something meaningful related to the image size. Imagine that we're just scaling up the activation size to the same image size.

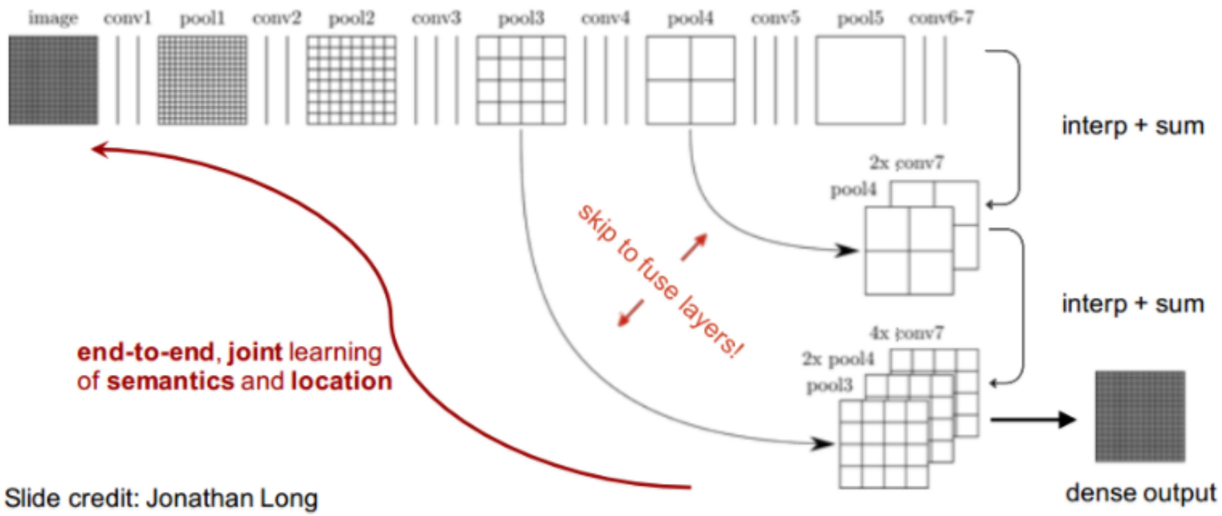
This last "upsampling" layer also has some lernable parameters.



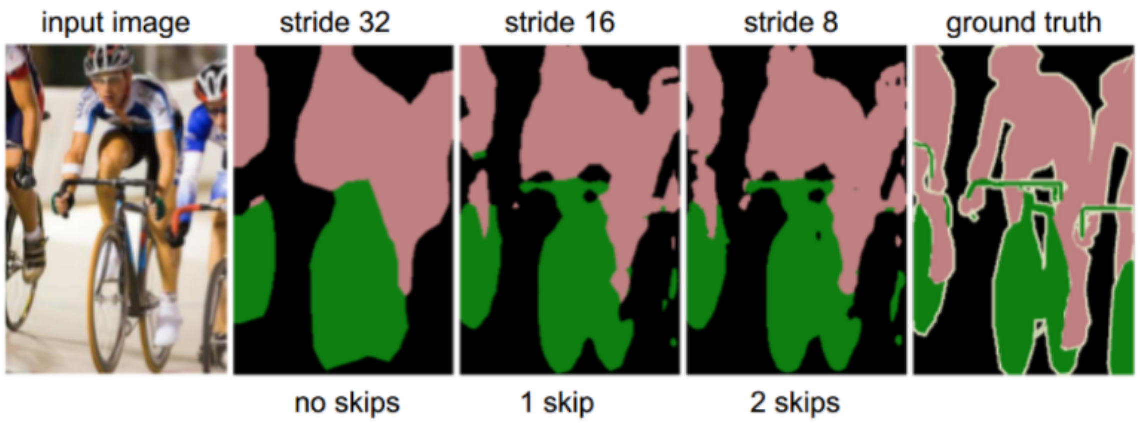
Now with this structure we just need to find some "ground truth" and to end to end learning, starting from a pre-trainned network ie: Imagenet.

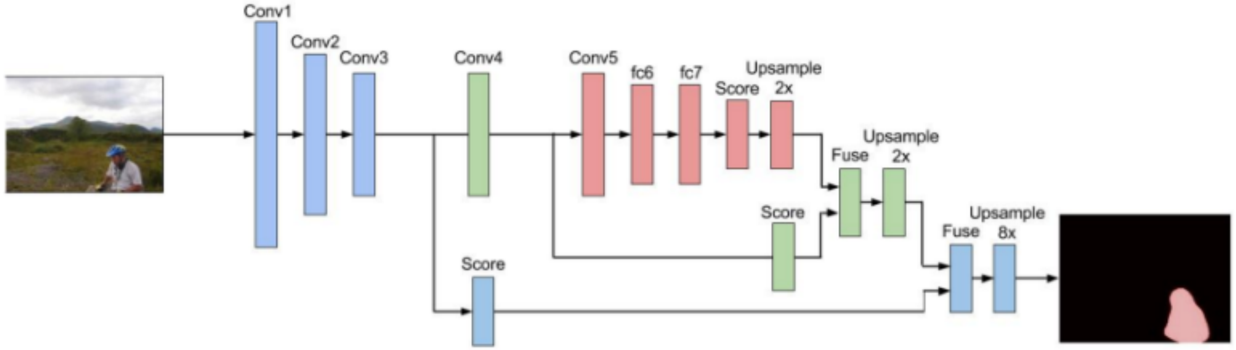
The problem with this approach is that we lose some resolution by just doing this because the activations were downscaled on a lot of steps.

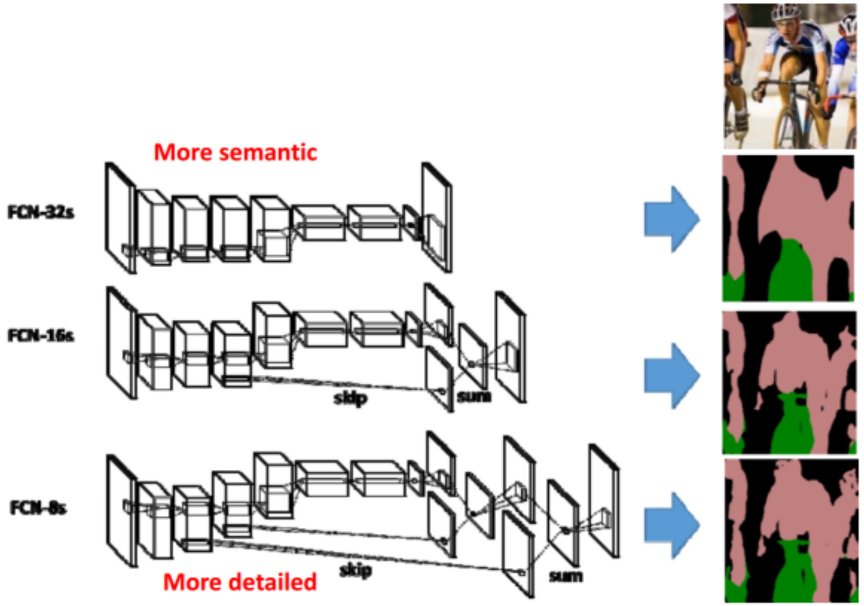
Those up-sampling operations used on skip are also learn-able.



Slide credit: Jonathan Long





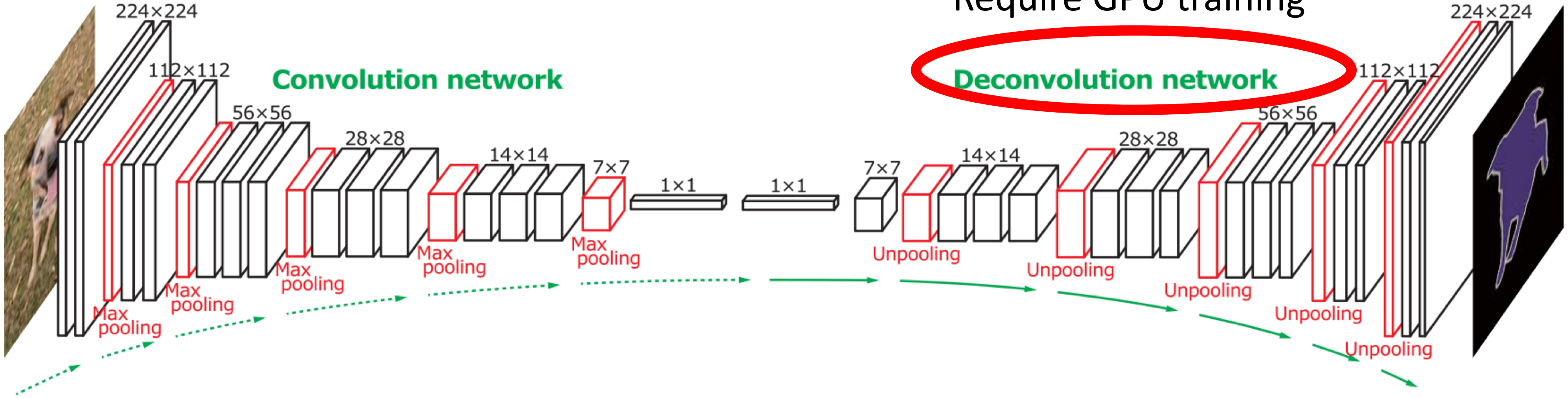


Below we show the effects of this "skip" process, notice how the resolution of the segmentation improves after some "skips"

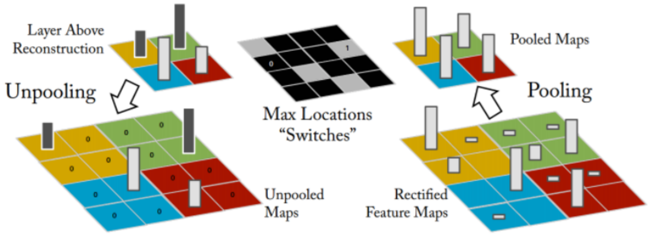
Convolutional Networks for Segmentation

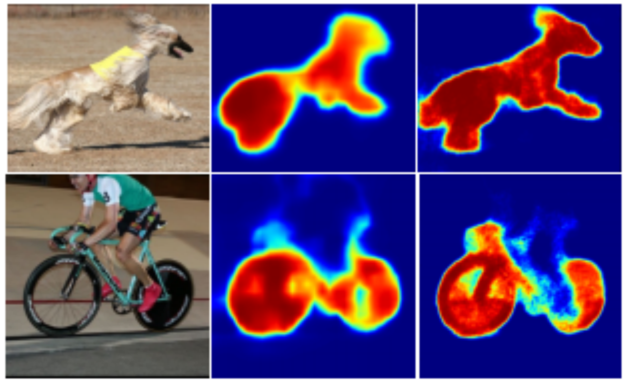
Ref: https://arxiv.org/pdf/1505.04366.pdf

Require GPU training



Deconvnet

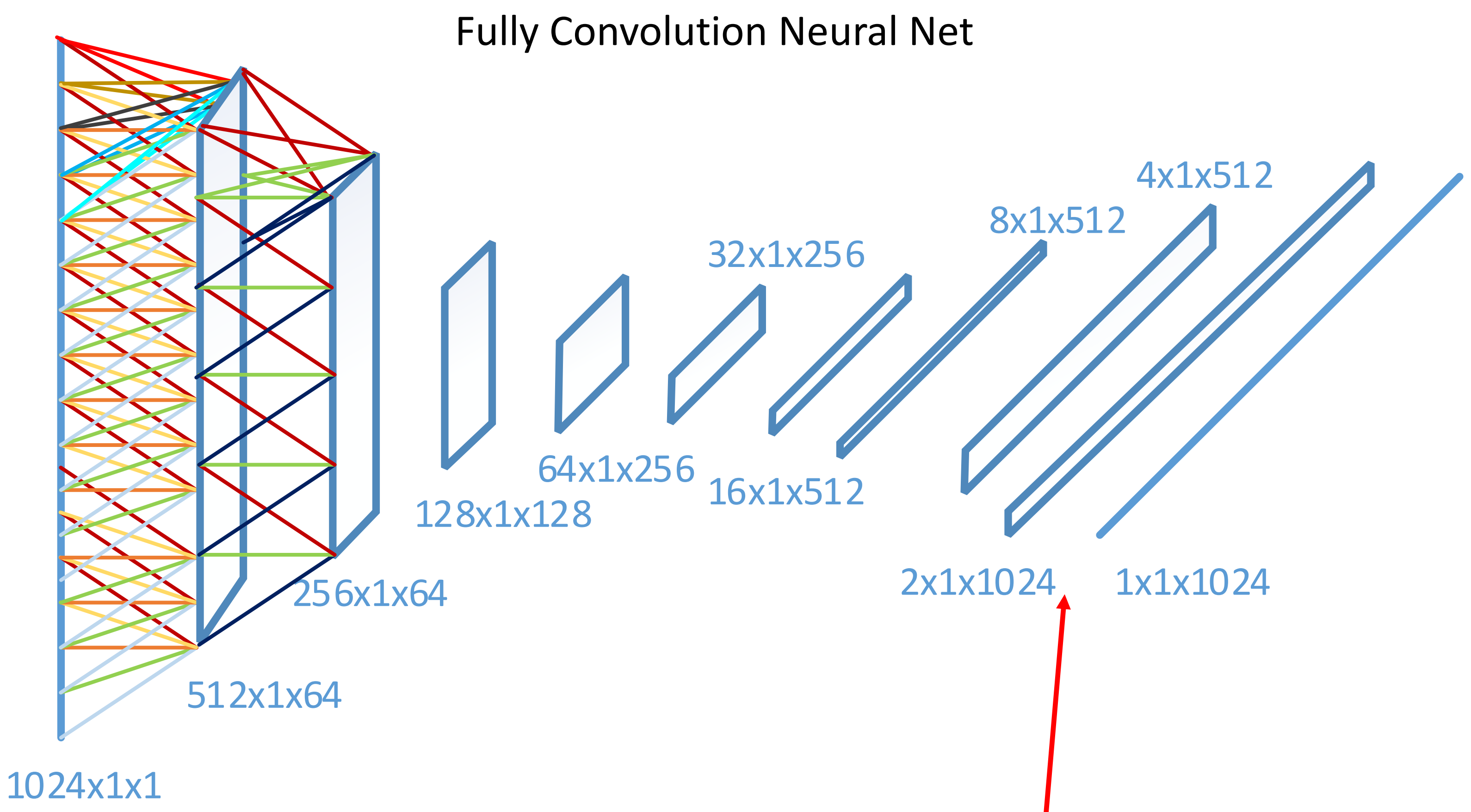




(a) Input image (b) FCN-8s (c) Ours

Figure 5. Comparison of class conditional probability maps from FCN and our network (top: dog, bottom: bicycle).

CNN based channel type segmentation



Using 9 layers of (conv and pooling) and 1 softmax layer
to convert 1024x1x1 to 1x1x1024

Results

```
self.W1, self.b1 = self.weightInital([5, 1, 1, 64], 1)
self.W2, self.b2 = self.weightInital([3, 1, 64, 64], 1)
self.W3, self.b3 = self.weightInital([3, 1, 64, 128], 1)
self.W4, self.b4 = self.weightInital([3, 1, 128, 128], 1)
self.W5, self.b5  = self.weightInital([3,1,128,256],1)
self.W6, self.b6  = self.weightInital([3,1,256,256],1)
self.W7, self.b7  = self.weightInital([3,1,256,256],1)
self.W8, self.b8  = self.weightInital([3,1,256,512],1)
self.W9, self.b9  = self.weightInital([3,1,512,1024],1)
```

At epoch=0, batch=0, cost value / error: 7343513.312 / 0.497

At epoch=0, batch=10, cost value / error: 9606639.438 / 0.488

At epoch=0, batch=20, cost value / error: 6994126.188 / 0.471

At epoch=0, batch=30, cost value / error: 6633453.875 / 0.386

At epoch=0, batch=40, cost value / error: 6435569.094 / 0.365

At epoch=0, batch=50, cost value / error: 6155082.031 / 0.333

At epoch=0, batch=60, cost value / error: 5956943.562 / 0.317

…

At epoch=49, batch=40, cost value / error: 2795950.852 / 0.110

At epoch=49, batch=50, cost value / error: 2718224.102 / 0.109

At epoch=49, batch=60, cost value / error: 2679340.062 / 0.107

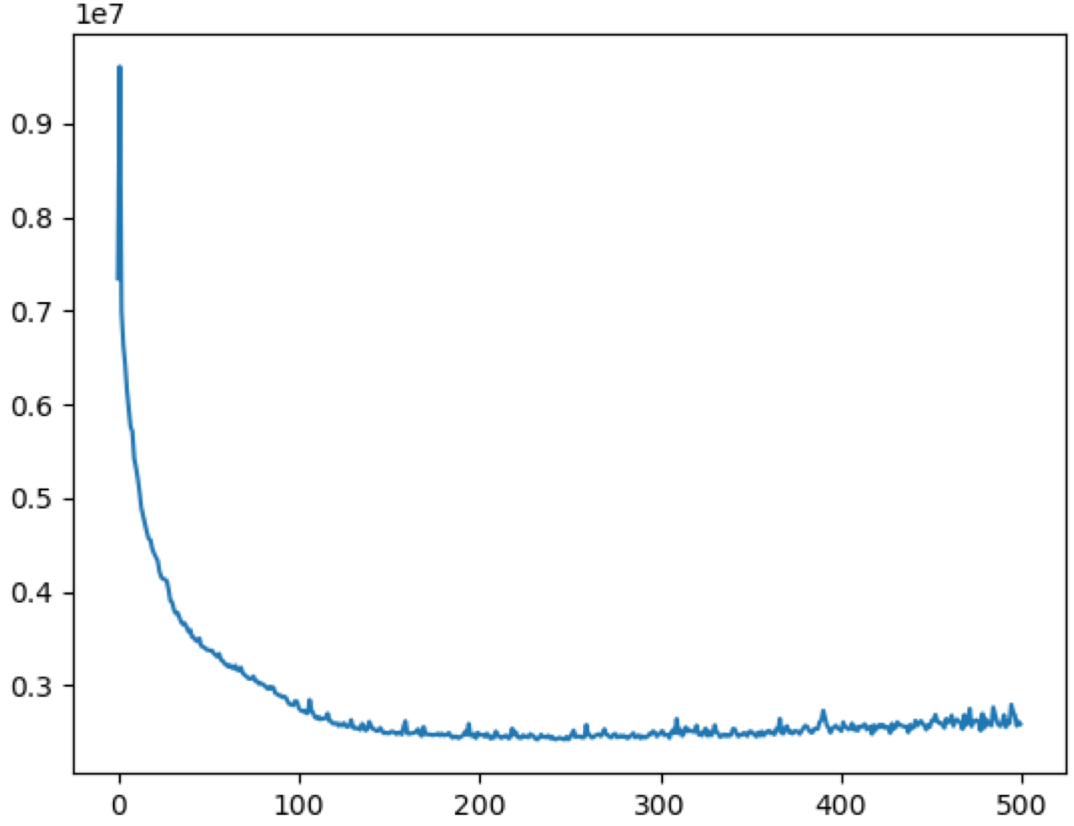
At epoch=49, batch=70, cost value / error: 2567827.773 / 0.105

At epoch=49, batch=80, cost value / error: 2612749.945 / 0.106

At epoch=49, batch=90, cost value / error: 2587396.586 / 0.106

Elapsed time: 8:36:07.203214

1024samples / 100m



Accuracy 89%

Results

```
self.W1, self.b1 = self.weightInital([5, 1, 1, 64], 1)
self.W2, self.b2 = self.weightInital([3, 1, 64, 64], 1)
self.W3, self.b3 = self.weightInital([3, 1, 64, 128], 1)
self.W4, self.b4 = self.weightInital([3, 1, 128, 128], 1)
self.W5, self.b5  = self.weightInital([3,1,128,256],1)
self.W6, self.b6  = self.weightInital([3,1,256,256],1)
self.W7, self.b7  = self.weightInital([3,1,256,256],1)
self.W8, self.b8  = self.weightInital([3,1,256,512],1)
self.W9, self.b9  = self.weightInital([3,1,512,1024],1)
self.W10, self.b10  = self.weightInital([3,1,1024,2048],1)
```

At epoch=0, batch=0, cost value / error: 14277907.438 / 0.488

At epoch=0, batch=10, cost value / error: 14461604.625 / 0.490

At epoch=0, batch=20, cost value / error: 12914682.438 / 0.366

At epoch=0, batch=30, cost value / error: 11400067.250 / 0.292

At epoch=0, batch=40, cost value / error: 10514962.219 / 0.258

…

At epoch=19, batch=60, cost value / error: 2640149.461 / 0.040

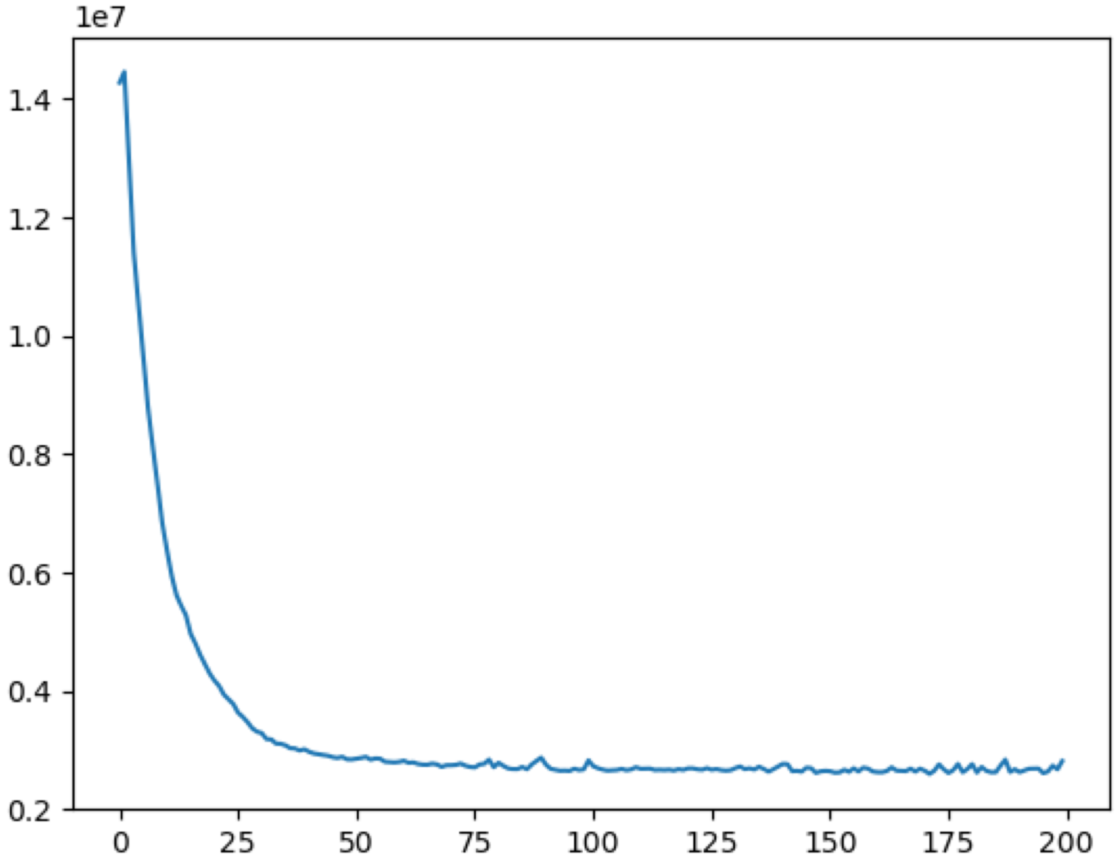
At epoch=19, batch=70, cost value / error: 2738783.000 / 0.041

At epoch=19, batch=80, cost value / error: 2675895.141 / 0.042

At epoch=19, batch=90, cost value / error: 2816133.703 / 0.043

Elapsed time: 8:57:54.880448

2048 samples / 40m



Accuracy 95%

Discussion

Framework for the proposed channel estimation approach

Stage 0: Satellite/aerial images segmentation for channel and channel similarity rough estimation

- Step 0.1: Image segmentation (can utilize available works based on deep learning)
  - Input: Satellite/aerial images of the interest area
  - Output: Segmented image with several classes of: street, building, trees, mountain, river, etc.
- Step 0.2: Channel rough estimation (similar to RF propagation SW)
  - Input:
    - Segmented image at output of step 0.1/ soil, terrain data (if available)
    - Antennas types, locations,
    - Frequency, BW
  - Output: Rough channel information: path loss, fading, shadowing, etc.

Framework for the proposed channel estimation approach

Stage 1: Similarity segmentation/ Similarity acquisition

- Step 1.1: Geometry segmentation
  - Input: Segmented image at output of step 0.1
  - Output: Group similarity (ex: same street, same direction, same building, ...)
- Step 1.2: Channel type segmentation for similarity area segmentation
  - Input: Measured signal power at consecutive points
  - Output:Channel type segmentation
- Step 1.3: Combining Geometry and measure segmentation at step 1.1 and step 1.2 for obtaining a joint similarity between 2 terminals and formulating a unified similarity map

Framework for the proposed channel estimation approach

Stage 2: Collaborative filtering for channel recommendation

- Step 2.1: Interference estimation
  - Input:
    - interest location,
    - Unified similarity map at step 1.3
    - sparse measure of interference levels at some other locations
  - Output: interference level at desired location
- Step 2.2: channel information estimation/fusion
  - Input:
    - Desired locations
    - Similarity map
    - Locations, Antenna info., Freq., BW, Channel rough estimation at step 0.2 (if available)
    - Historical channel info at some other locations
  - Output:
    - Fused estimated channel information
- Step 2.3: Channel recommendation: Combining estimated Interference level at 2.1 and Fused estimated channel information at 2.2 to select appropriate transmission parameters (power, freq., BW, modulation type, etc.)

Thank you!