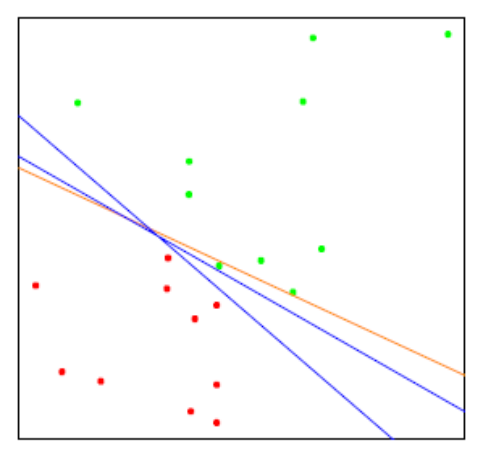


Deep Learning

Introduction to Deep Learning Models

Dr. – Ing. Gabriele Cavallaro
Postdoctoral Researcher
High Productivity Data Processing Group
Juelich Supercomputing Centre, Germany

LECTURE 4

Deep Learning in Remote Sensing: Applications

June 6th, 2018
Juelich Supercomputing Centre, Germany, Germany

UNIVERSITY OF ICELAND
SCHOOL OF ENGINEERING AND NATURAL SCIENCES
FACULTY OF INDUSTRIAL ENGINEERING, MECHANICAL ENGINEERING AND COMPUTER SCIENCE

JÜLICH FORSCHUNGSZENTRUM

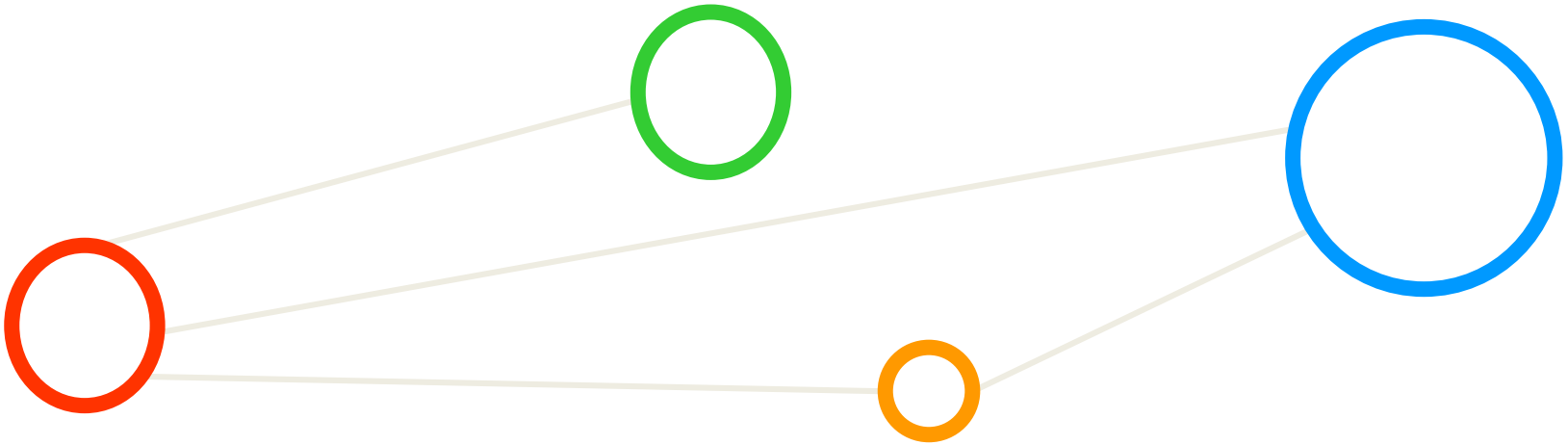
HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

DEEP Projects

Outline of the Course

- 1. Introduction to Deep Learning
- 2. Fundamentals of Convolutional Neural Networks (CNNs)
- 3. Deep Learning in Remote Sensing: Challenges
- 4. Deep Learning in Remote Sensing: Applications
- 5. Model Selection and Regularization
- 6. Fundamentals of Long Short-Term Memory (LSTM)
- 7. LSTM Applications and Challenges
- 8. Deep Reinforcement Learning

Outline

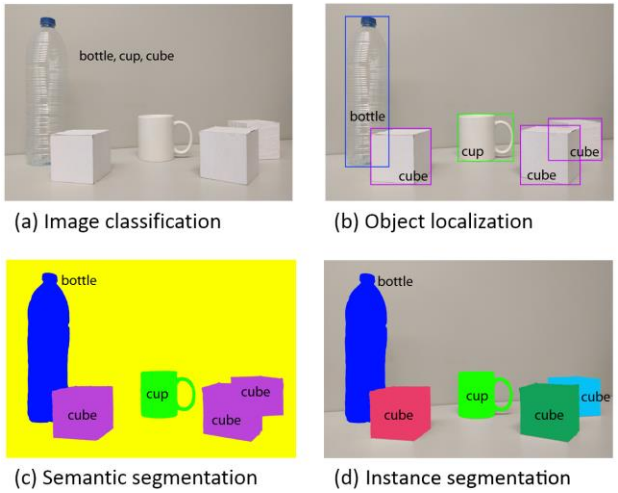


Outline

- DL Architectures for RS Applications
  - Semantic Segmentation
  - Image Classification with CNNs
  - Fully Convolutional Networks (FCNs)
  - Vanishing Gradient Problem
  - Residual Networks (ResNet)
  - Vaihingen Classification Results

Progression from Coarse to Fine Inference

- **Classification**: make a prediction for a whole input
  - What are the classes and ranked list
- **Localization or detection:** towards fine-grained inference
  - Classification and spatial location (e.g., bounding boxes)
- **Semantic segmentation**: fine-grained inference
  - Make dense predictions inferring labels for every pixel
- **Further improvements:** Provide different instances of the same class
  - Decomposition of already segmented classes into their components



(a) Image classification (b) Object localization

(c) Semantic segmentation (d) Instance segmentation

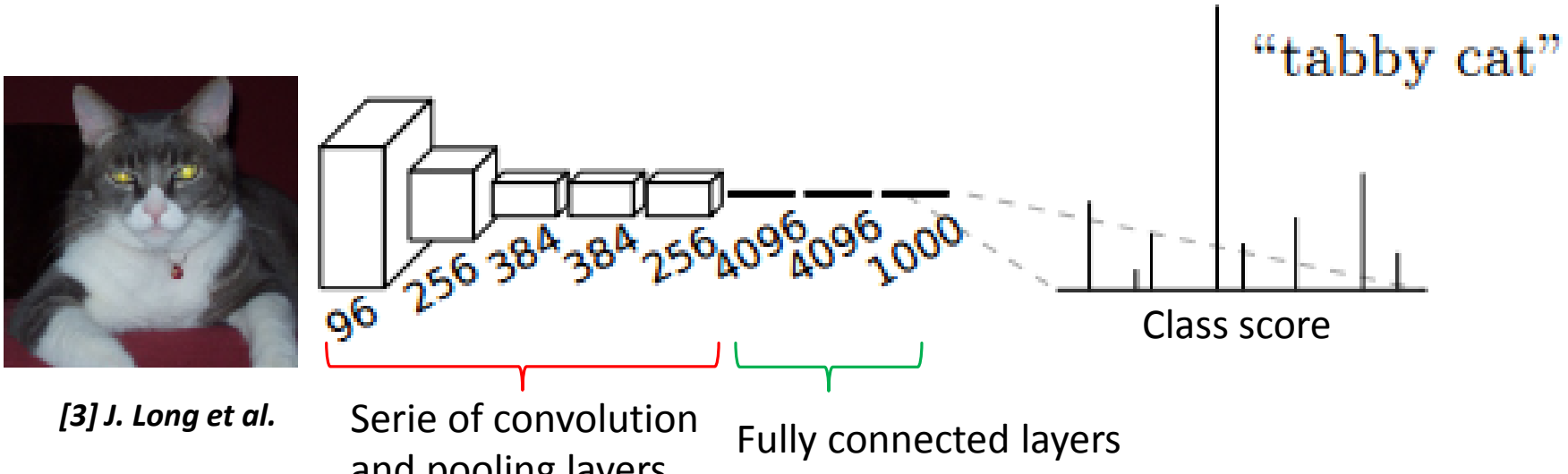
[1] A. Garcia-Garcia

- Many applications nourish from inferring knowledge from imagery
  - Autonomous driving
  - Human-machine interaction
  - Computational photography
  - Image search engines
  - Remote sensing

[2] Image Segmentation

Image Classification CNNs

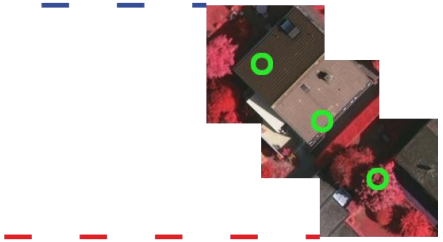
- **Convolutional layers:** convolution operation to the input
  - Emulate the response of an individual neuron to visual stimuli
  - Each convolutional neuron processes data only for its receptive field
- **Polling layers:** progressively reduce the spatial size of the representation
  - Reduce the amount of parameters and computation and control overfitting
- **Fully connected layers** connect every neuron in one layer to every neuron in another layer
  - Same principle as the traditional multi-layer perceptron (MLP) network



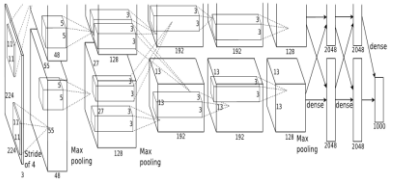
[3] J. Long et al.

Semantic Segmentation Approach: Sliding Window

- Break the images into many small crops and classify the central pixel
- Redundant and computational expensive
  - Store not only every pixel but also the surrounding pixels
  - Increases the data size by a factor determined by the number of neighbouring pixels
- Very inefficient and not reusing shared features between overlapping patches



Extract patch



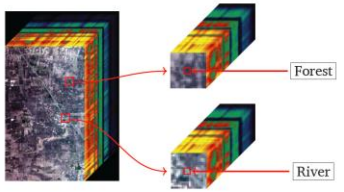
Classify center pixel with CNN

Building

Building

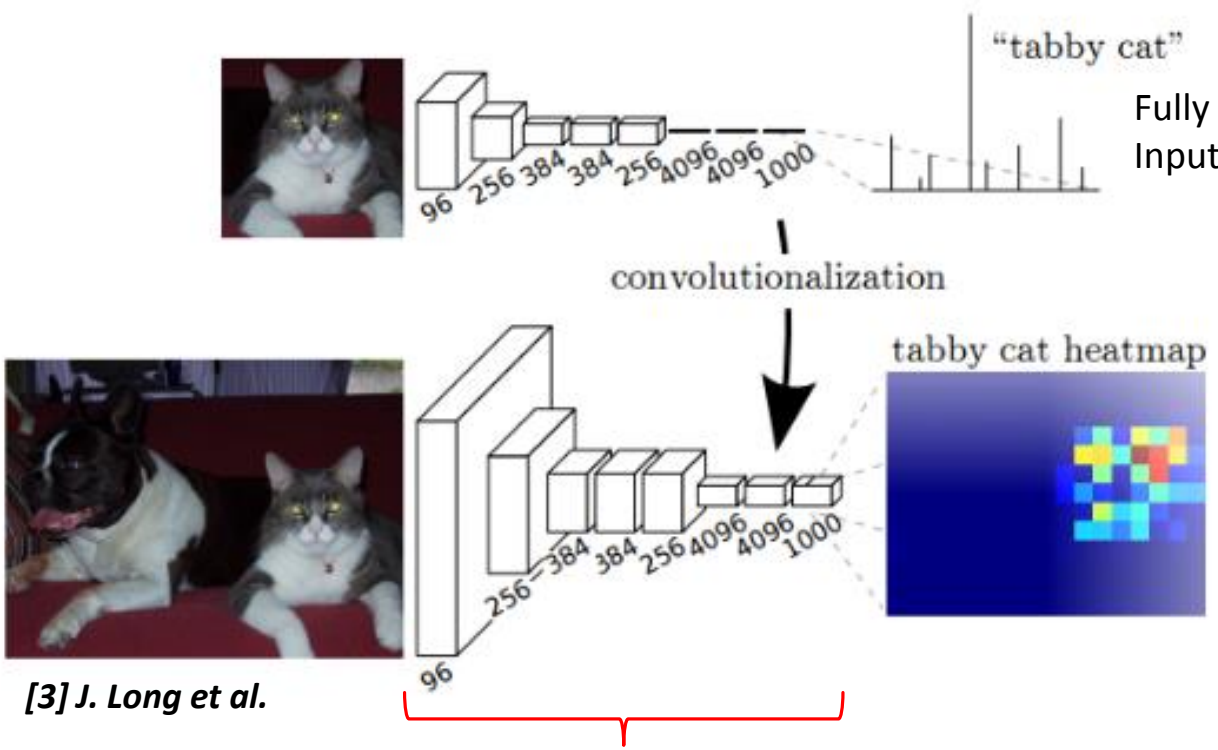
Tree

More visible with Hyperspectral images



Semantic Segmentation Approach: Fully Convolutional

- Semantic segmentation tasks have input images with different sizes
- Fully convolutional networks can take input of any arbitrary size
- Produce correspondingly-sized dense output with efficient inference and learning



Fully connected layers:
Inputs and feature maps of fixed size

Convolution can have inputs of any size

"Transforming a classification-purposed CNN to produce spatial heatmaps by replacing fully connected layers with convolutional ones"

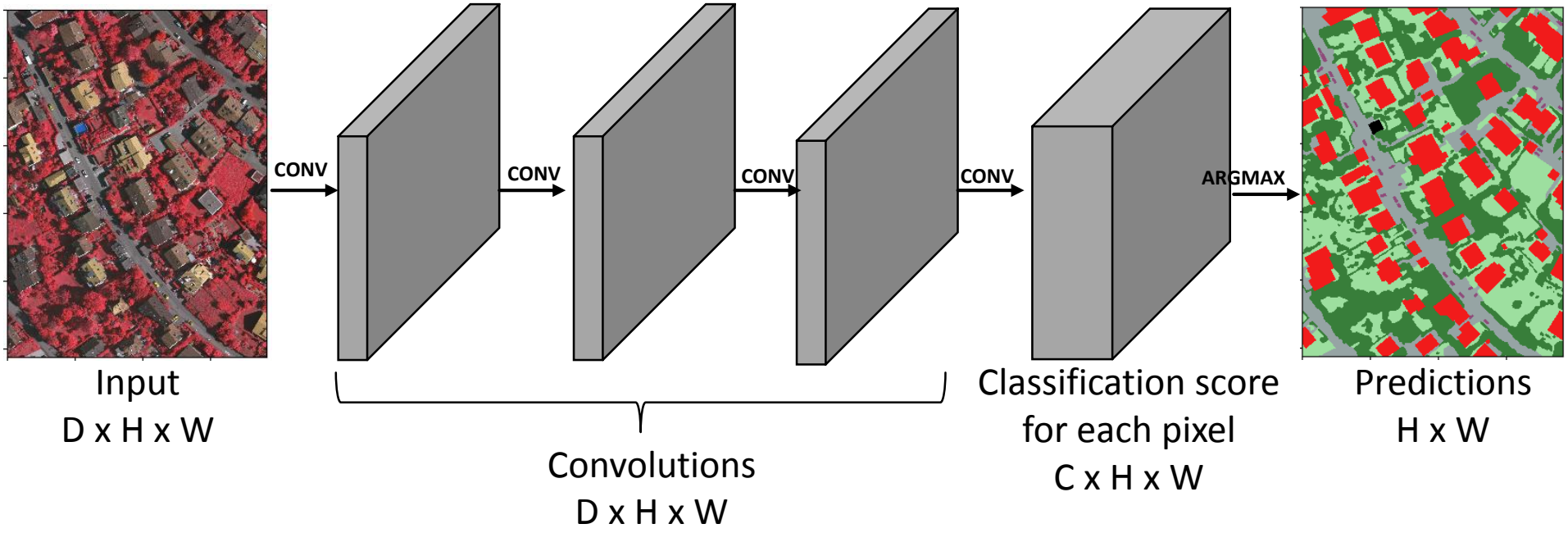
[3] J. Long et al.

Serie of convolution and pooling layers

Semantic Segmentation Approach: Fully Convolutional

- Network as a bunch of convolutional layers to make predictions for pixels all at once
- Each layer preserve the size of the input
- The final convolutional layer output a tensor (C: number of classes)

Create a training set for this network Is very 'expensive'



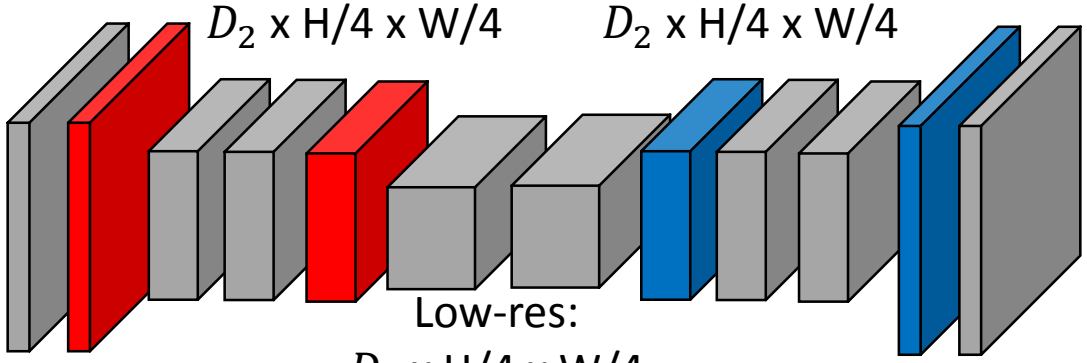
- Problem: convolutions at original image resolution will be very expensive
  - E.g., Hyperspectral input images with D>100 bands

Semantic Segmentation Approach: Fully Convolutional

- Design networks as a bunch of convolutional layers
- With downsampling and upsampling inside the network

Downsampling:
Pooling, strided convolution

Input
D x H x W



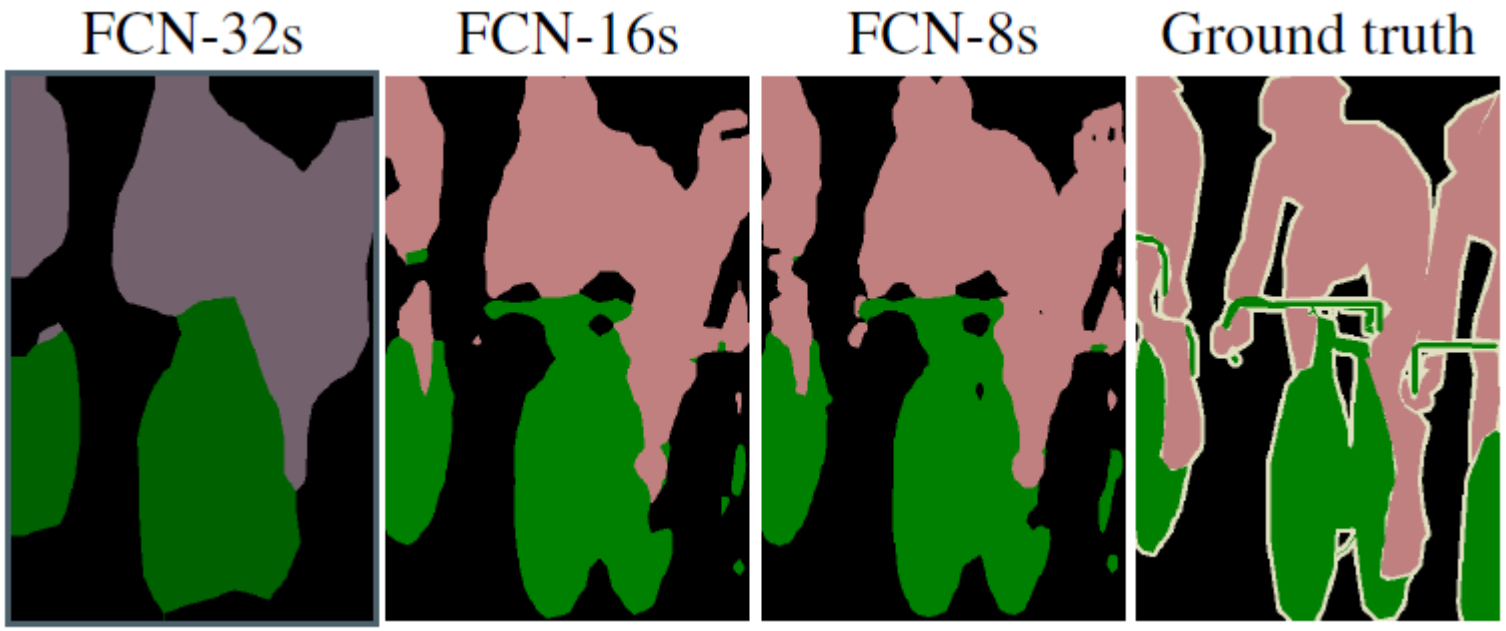
Upsampling:
Unpooling or stride
Transpose convolution

Predictions
H x W

- Rather than transitioning to a fully connected layer Increase the spatial resolution
- Computationally very efficient
- Networks can be deep and work on lower spatial resolution in many of the layers

Semantic Segmentation Approach: Fully Convolutional

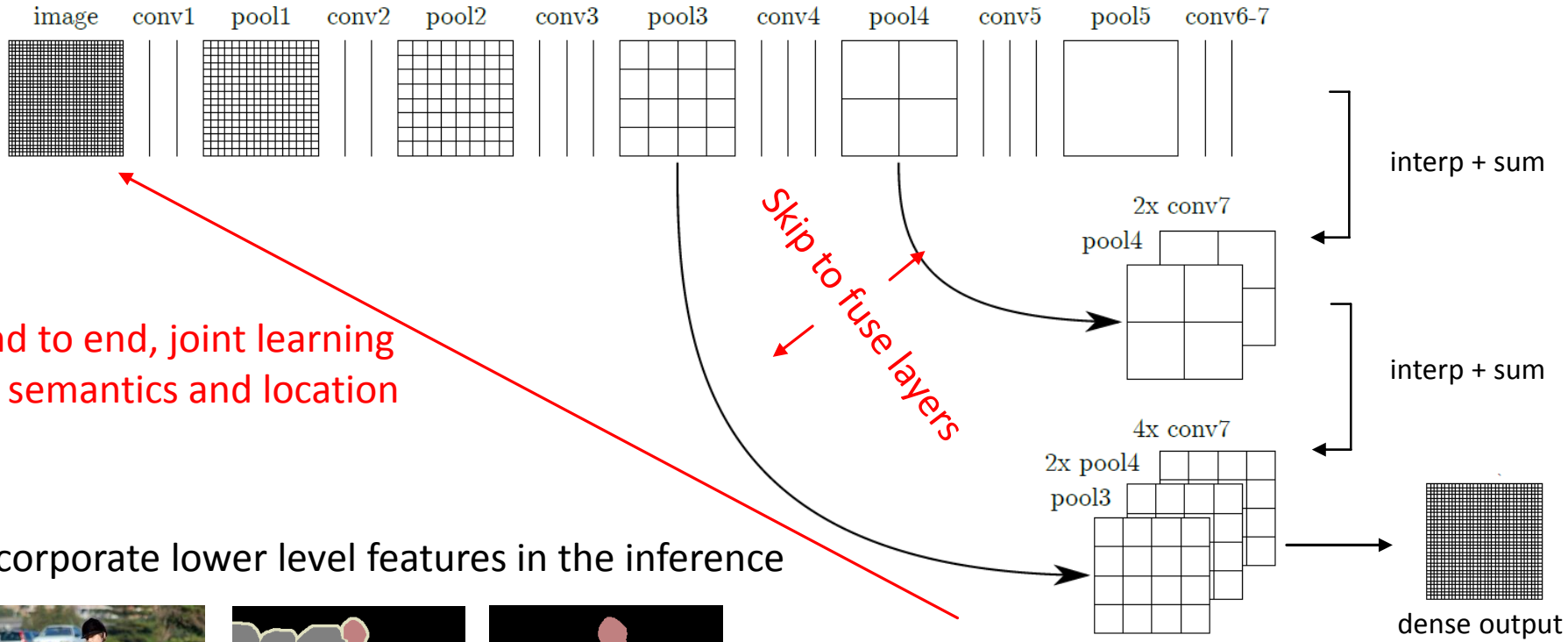
- The predictions of FCNs can be too coarse because of the upsampling steps
- Low level of details in the upsampled output.
- FCN can be improved by making direct use of shallower and more local features



[3] J. Long et al.

Combining Feature Hierarchies: Skip Connections

- Combine layers of the feature hierarchy for refining the spatial precision of the output
- Fuse features across layers to define a nonlinear local-to-global representation



- Incorporate lower level features in the inference

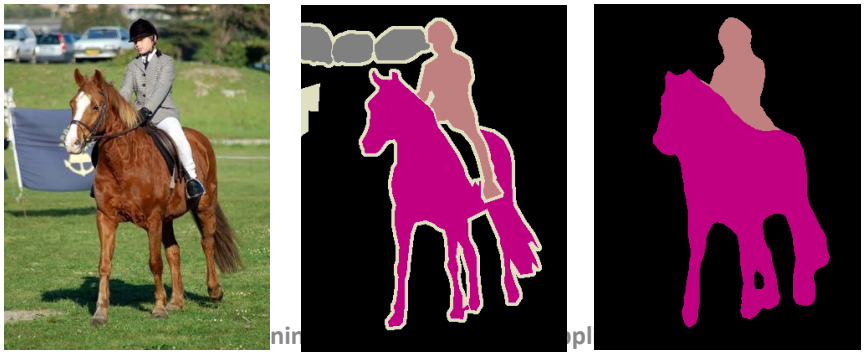
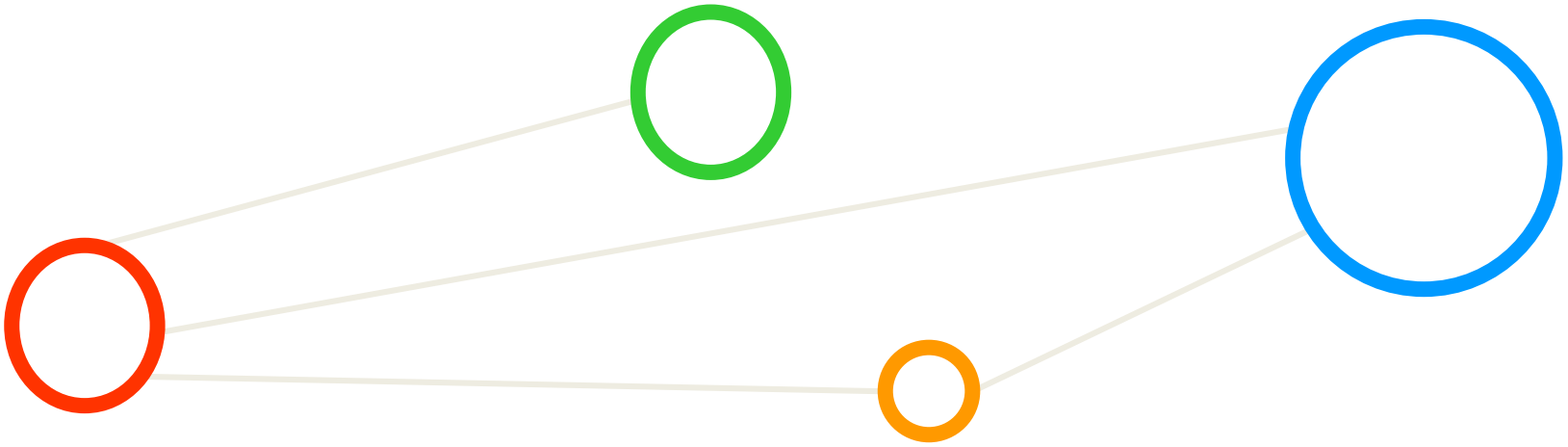


Image GT Output

The network can adjust the coarse prediction to edges and specific part of the image

Residual Networks (ResNet)



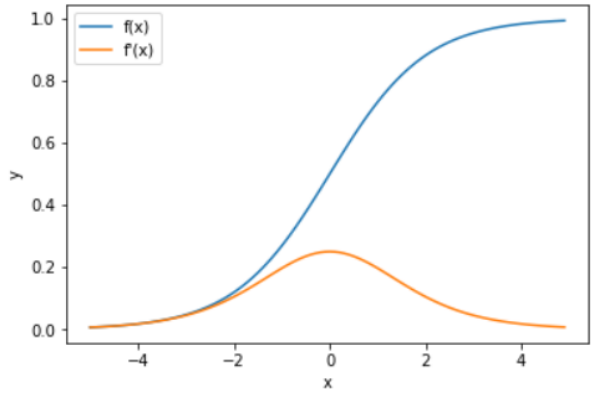
Vanishing Gradient Problem

"No matter how deep a network is, it should not be any worse than the shallower network"

[4] K. He et al.

- With more parameters to learn, the train data should be fit at least as well as before

- **Vanishing gradient** problem: difficulty in learning the parameters of the earlier layers
  - Networks with gradient based methods (e.g., Backpropagation).



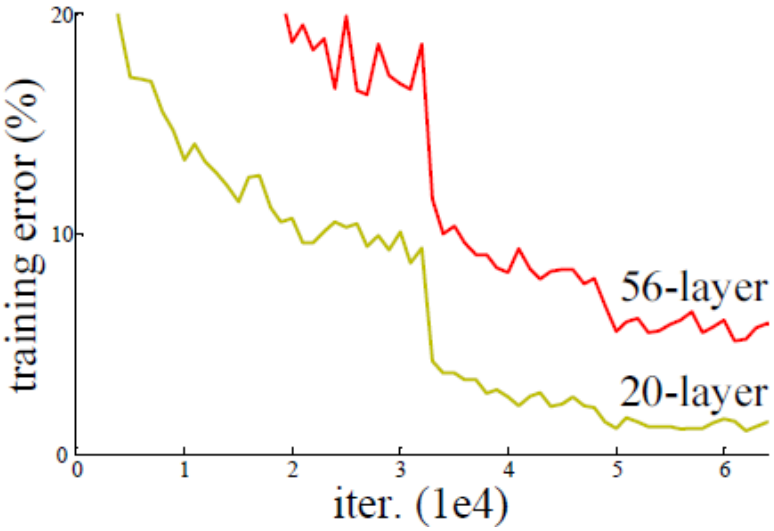
Sigmoid gradient

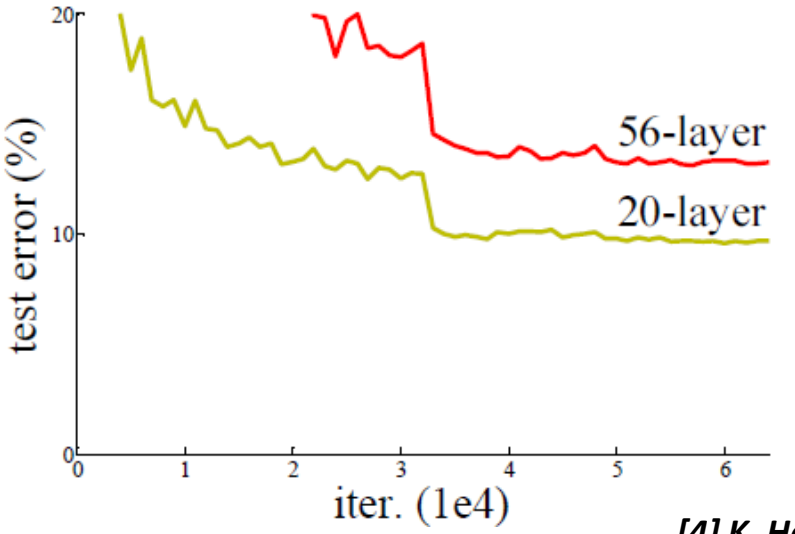
- Possible Solutions:
  - Multi-level hierarchy
  - Long short-term memory
  - Faster hardware
  - Residual networks
  - Other activation functions (e.g., ReLU)

[5] The vanishing gradient problem

Degradation Problem

- Vanishing gradient becomes worse as the number of layers increases
- Accuracy gets saturated and then degrades rapidly
- Unexpectedly, **such degradation is not caused by overfitting**
  - adding more layers to a suitably deep model leads to higher training error
- Example with 2 “plain” networks with different depths on CIFAR-10 datset





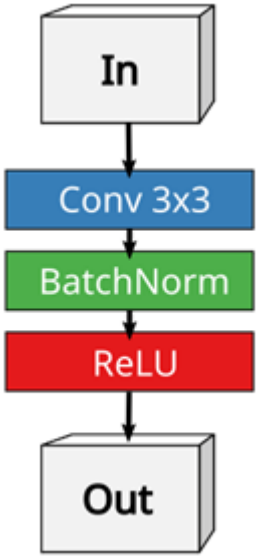
[4] K. He et al.

The deeper network has higher training error and this test error

Residual Networks (ResNet)

- Architecture which solves vanishing gradient problem in a simple way
- Residual networks can be much deeper than their 'plain' counterparts,
  - Yet they require a similar number of parameters (weights)

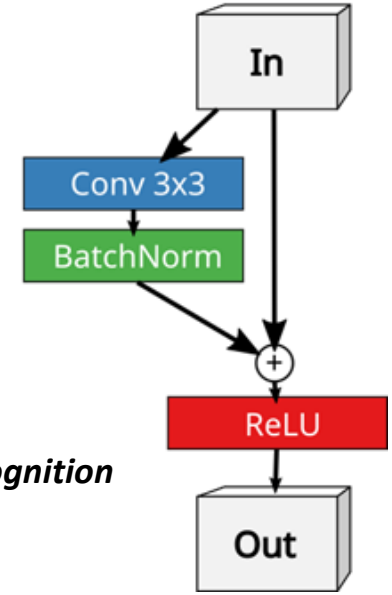
Plain

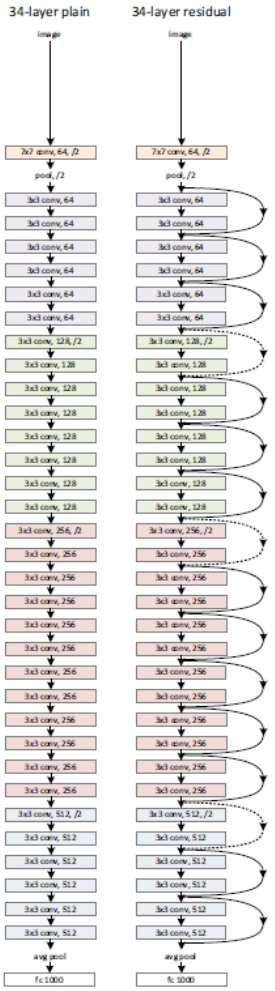


"If there is trouble sending the gradient signal backwards, why not provide the network with a shortcut at each layer to make things happen more smoothly"?

[6] Deep Residual Learning for Image Recognition

Residual

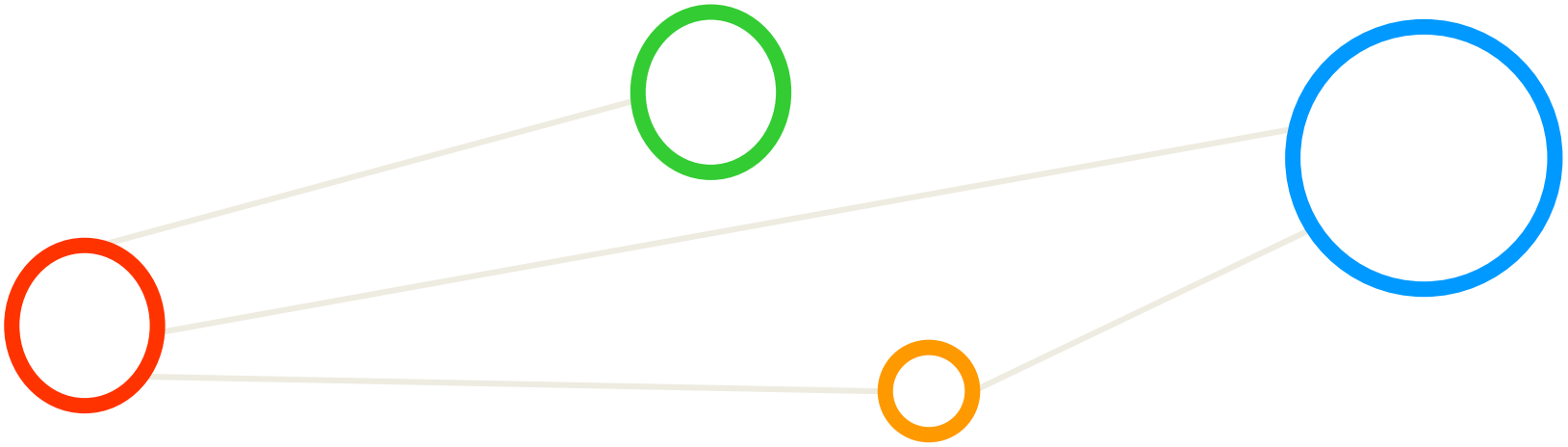




[4] K. He et al.

The gradient could "skip" all the layers and reach the bottom without being diminished

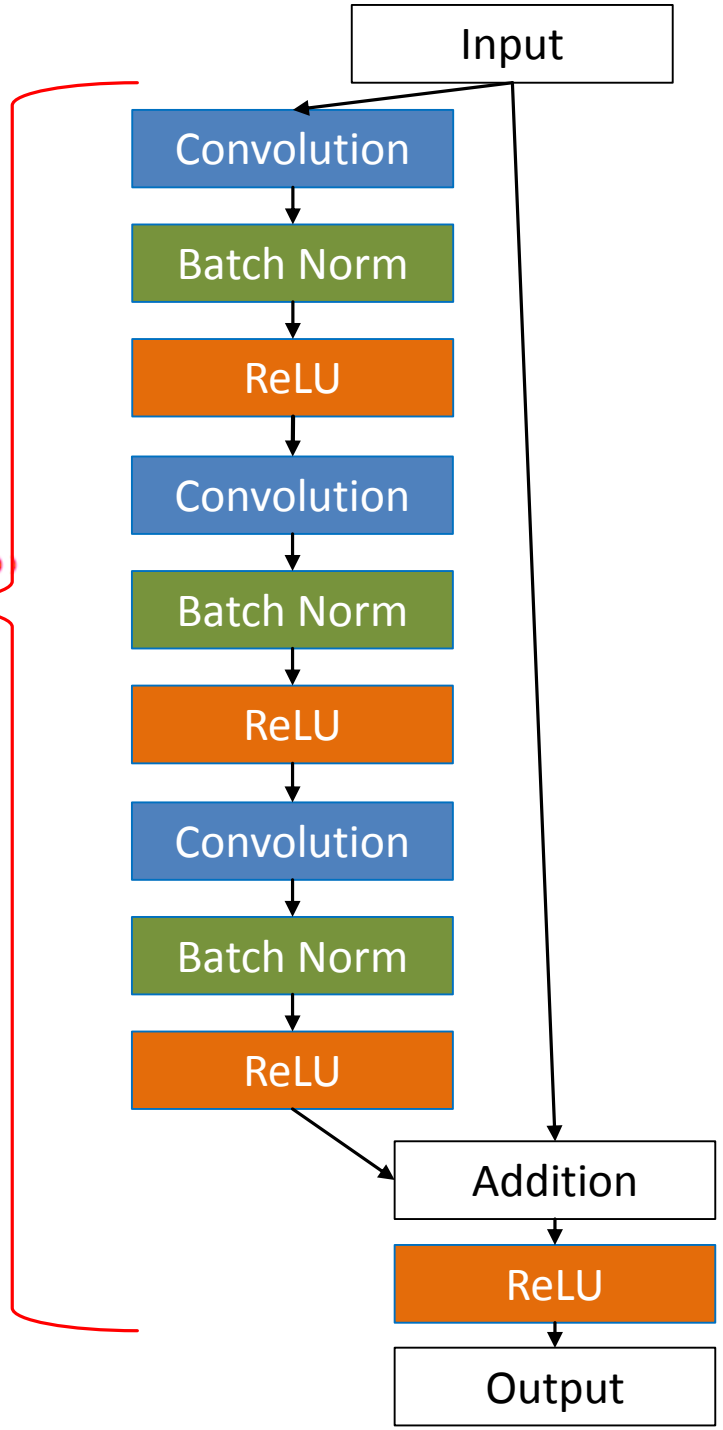
The ResNet50 FCN Model



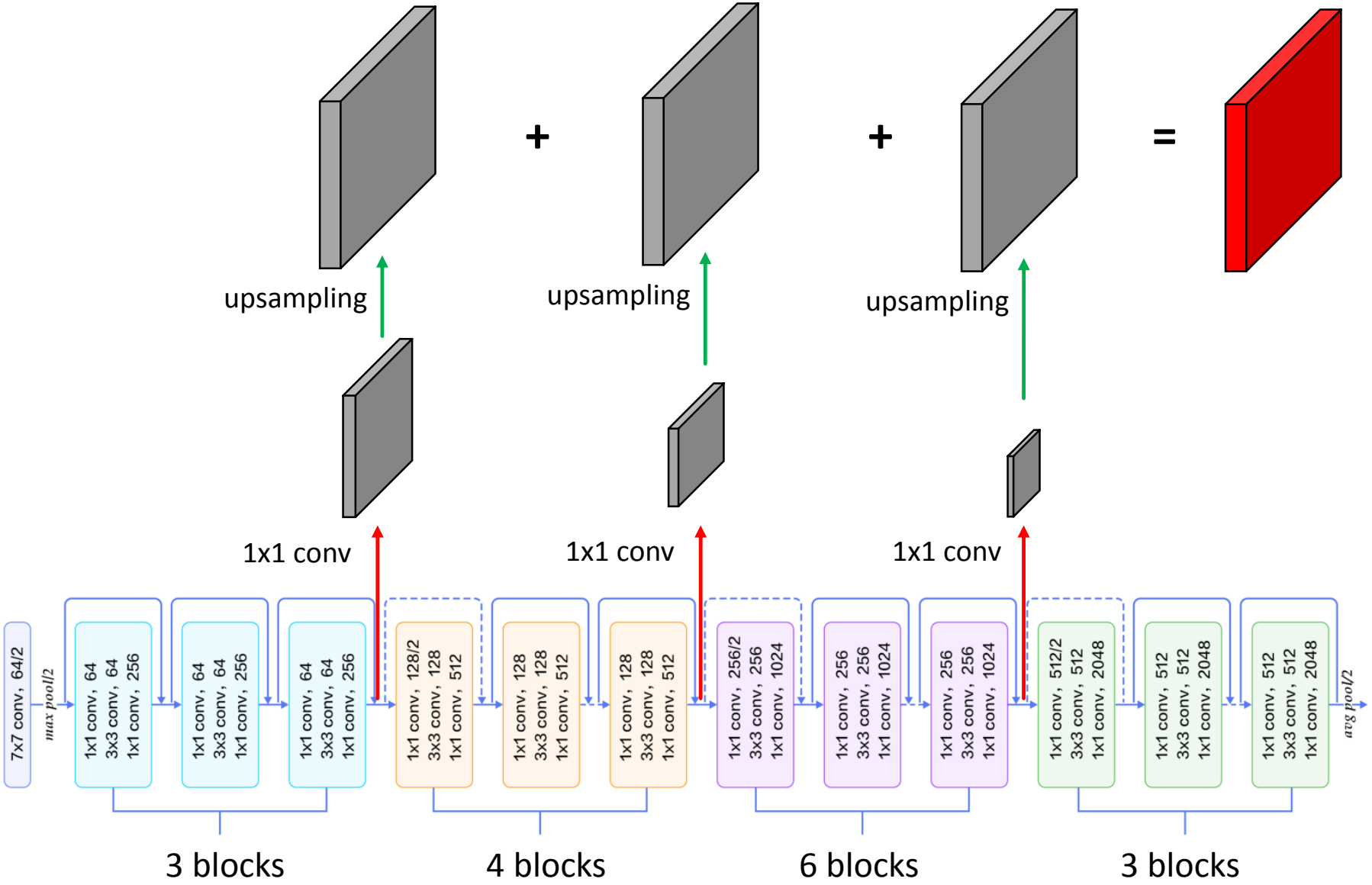
ResNet50 FCN

- *~/semseg/*resnet50-fcn/resnet50_edit.*py*

```
201     x = ZeroPadding2D((3, 3))(img_input)
202     x = Conv2D(64, (7, 7), strides=(2, 2), name='conv1')(x)
203     x = BatchNormalization(axis=bn_axis, name='bn_conv1')(x)
204     x = Activation('relu')(x)
205     x = MaxPooling2D((3, 3), strides=(2, 2))(x)
206
207     x = conv_block(x, 3, [64, 64, 256], stage=2, block='a', strides=(1, 1))
208     x = identity_block(x, 3, [64, 64, 256], stage=2, block='b')
209     x = identity_block(x, 3, [64, 64, 256], stage=2, block='c')
210
211     x = conv_block(x, 3, [128, 128, 512], stage=3, block='a')
212     x = identity_block(x, 3, [128, 128, 512], stage=3, block='b')
213     x = identity_block(x, 3, [128, 128, 512], stage=3, block='c')
214     x = identity_block(x, 3, [128, 128, 512], stage=3, block='d')
215
216     x = conv_block(x, 3, [256, 256, 1024], stage=4, block='a')
217     x = identity_block(x, 3, [256, 256, 1024], stage=4, block='b')
218     x = identity_block(x, 3, [256, 256, 1024], stage=4, block='c')
219     x = identity_block(x, 3, [256, 256, 1024], stage=4, block='d')
220     x = identity_block(x, 3, [256, 256, 1024], stage=4, block='e')
221     x = identity_block(x, 3, [256, 256, 1024], stage=4, block='f')
222
223     x = conv_block(x, 3, [512, 512, 2048], stage=5, block='a')
224     x = identity_block(x, 3, [512, 512, 2048], stage=5, block='b')
225     x = identity_block(x, 3, [512, 512, 2048], stage=5, block='c')
226
227     x = AveragePooling2D((7, 7), name='avg_pool')(x)
```



The ResNet is adapted into an FCN

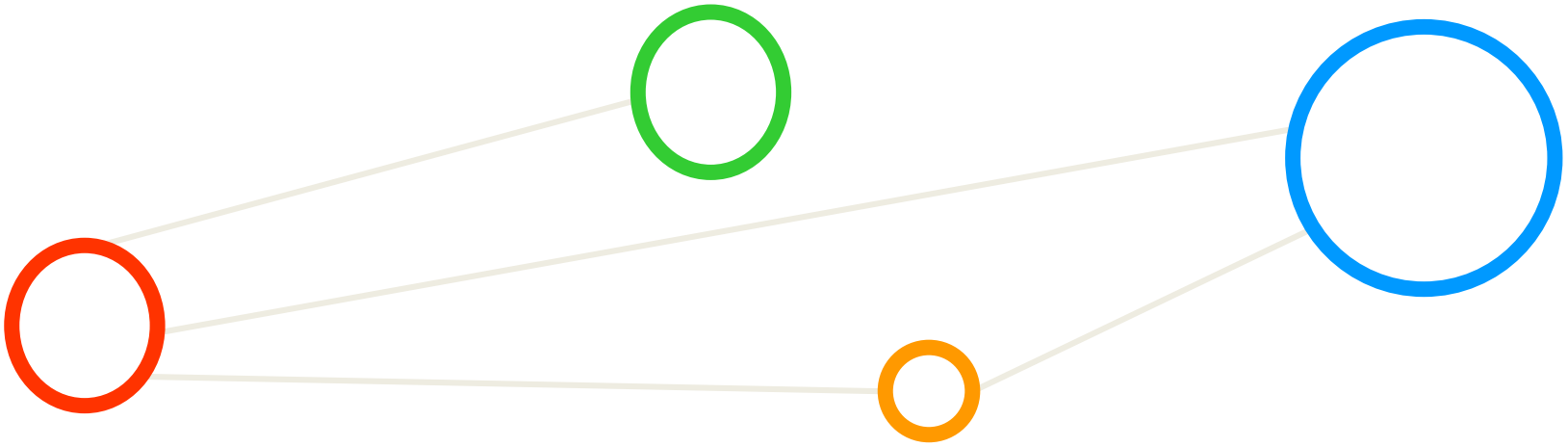


Residual Network 50 FCN

- Python fu*~/semseg/resnet50-fcn/ model_generator.py*

```
15  # the function to generate a FCN version of the ResNet50 model:
16  def generate_resnet50_fcn(use_pretraining):
17      num_labels = 6
18      input_dim_row = 256
19      input_dim_col = 256
20      input_shape = (input_dim_row, input_dim_col, 3)
21      input_tensor = Input(shape=input_shape)
22      weights = 'imagenet' if use_pretraining else None
23      standard_model = ResNet50(include_top=False, weights=weights, input_tensor=input_tensor)
24
25      # get the activations after different network parts by name:
26      x32 = standard_model.get_layer('act3d').output
27      x16 = standard_model.get_layer('act4f').output
28      x8 = standard_model.get_layer('act5c').output
29
30      # apply 1x1 convolution to compress the depth of the output tensors to the number of classes:
31      c32 = Convolution2D(filters=num_labels, kernel_size=(1, 1), name='conv_labels_32')(x32)
32      c16 = Convolution2D(filters=num_labels, kernel_size=(1, 1), name='conv_labels_16')(x16)
33      c8 = Convolution2D(filters=num_labels, kernel_size=(1, 1), name='conv_labels_8')(x8)
34
35      # resize the spatial dimensions to fit the spatial input size:
36      r32 = Lambda(resize_bilinear, name='resize_labels_32')(c32)
37      r16 = Lambda(resize_bilinear, name='resize_labels_16')(c16)
38      r8 = Lambda(resize_bilinear, name='resize_labels_8')(c8)
39
40      # sum up the activations of different stages to get information of different solution
41      m = Add(name='merge_labels')([r32, r16, r8])
42
43      # apply a softmax activation function to get the probability of each class for each pixel
44      x = Reshape((input_dim_row * input_dim_col, num_labels))(m)
45      x = Activation('softmax')(x)
46      x = Reshape((input_dim_row, input_dim_col, num_labels))(x)
47
48      # return the FCN version of the ResNet50 model:
49      return Model(inputs=input_tensor, outputs=x)
```

The ResNet50 FCN Training Function



Python Function for ResNet50 FCN with Augmentation

- Location ***~/semseg/*resnet50-fcn/train_resnet50_fcn.py**

```
45  def train_with_augmentation(data_path,output_model,transfer_learning_flag):
46      num_labels = 6
47
48      print('Load data ... ')
49      x_train, y_train = dio.load_data(data_path + 'vaihingen_train.hdf5')
50      print('Training samples: {}'.format(x_train.shape))
51      x_val, y_val = dio.load_data(data_path + 'vaihingen_val.hdf5')
52      print('Validation samples: {}'.format(x_val.shape))
53
54      print('generate augmented images ...')
55      x_train_aug, y_train_aug = augmentation.every_element_randomly_once(x_train, y_train)
56      x_val_aug, y_val_aug = augmentation.every_element_randomly_once(x_val, y_val)
57
58      print(x_train.dtype)
59      print(x_train_aug.dtype)
60
61      # put each array together with its augmented version:
62      print('a')
63      x_train = np.concatenate([x_train, x_train_aug])
64      y_train = np.concatenate([y_train, y_train_aug])
65      x_val = np.concatenate([x_val, x_val_aug])
66      y_val = np.concatenate([y_val, y_val_aug])
67
68      # shuffle the samples:
69      print('b')
70      x_train, y_train = augmentation.shuffle_4d_sample_wise(x_train, y_train)
71      x_val, y_val = augmentation.shuffle_4d_sample_wise(x_val, y_val)
72
73      print('preprocess the input data (normalization, centering) ... ')
74      x_train = preprocess_input(x_train, mode='tf')
75      x_val = preprocess_input(x_val, mode='tf')
77      print('preprocess the labels ... ')
78      y_train -= 1
79      y_val -= 1
80      y_train = to_categorical(y_train, num_labels)
81      y_val = to_categorical(y_val, num_labels)
82
83      print('load model ... ')
84      resnet50_fcn_model = model_generator.generate_resnet50_fcn(use_pretraining=transfer_learning_flag)
85      resnet50_fcn_model.summary()
86
87      print('compile model ... ')
88      resnet50_fcn_model.compile(optimizer=keras.optimizers.Adam(),
89                                 loss=keras.losses.categorical_crossentropy,
90                                 metrics=['accuracy'])
91      resnet50_fcn_model.fit(x_train, y_train, batch_size=8, epochs=20, validation_data=(x_val, y_val))
92
93      resnet50_fcn_model.save_weights(output_model)
```

Load:
vaihingen_train.hdf5
vaihingen_validation.hdf5

Apply one random augmentation to each 256x256 patch
Rotate(90,180,270) or Flip (up,down)

augmentation.py

Concatenate original data with the augmented data

Preprocessing

Generate the model

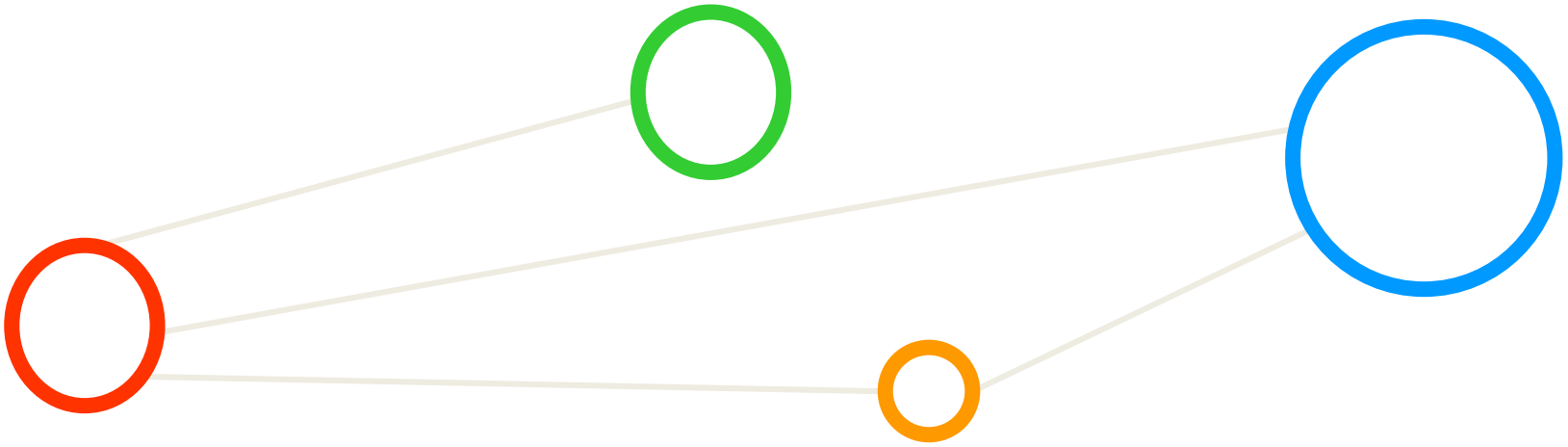
Train the model

Python Function for ResNet50 FCN with Augmentation

- ***~/semseg/*resnet50-fcn/train_resnet50_fcn.py**
- ***train_resnet50_fcn.py*** requires 4 parameters

```
96  def main(arguments):
97
98      data_path = arguments[1]
99      output_model = arguments[2]
100     augmentation_flag = arguments[3]
101     transfer_learning_flag = arguments[4]
102
103     if augmentation_flag=='True':
104         train_with_augmentation(data_path,output_model,transfer_learning_flag)
105     elif augmentation_flag=='False':
106         train(data_path,output_model,transfer_learning_flag)
107     else:
108         sys.exit()
109     end
110
111 if __name__ == '__main__':
112
113     if len(sys.argv)<4:
114       print ''
115       print '***********************************'
116       print 'Four paremeters need to be specified:'
117       print '1. Location of vaihingen_train.hdf5 and vaihingen_val.hdf5 (e.g., /homea/hpclab/train002/semseg/data/ )'
118       print '2. Location + name of the output model (e.g., /homea/hpclab/train002/semseg/models/resnet50_fcn_weights.hdf5)'
119       print '3. Augmentation: True or False'
120       print '4. Transfer learning (load weights trained on ImageNet): True or False'
121       print '***********************************'
122
123       sys.exit()
124
125     main(argv)
```

Practical 4:
Check the Outcomes of the Job Previously Submitted



Batch Scripts Previously Submitted

```
#!/bin/bash -x
#SBATCH--nodes=1
#SBATCH--ntasks=1
#SBATCH--output=train_resnet50_fcn_out.%j
#SBATCH--error=train_resnet50_fcn_err.%j
#SBATCH--time=01:00:00
#SBATCH--mail-user=g.cavallaro@fz-juelich.de
#SBATCH--mail-type=ALL
#SBATCH--job-name=train_resnet50_fcn

#SBATCH--partition=gpus
#SBATCH --gres=gpu:1

#SBATCH--reservation=deep_learning


### location executable
RESNET50_FCN=/homea/hpclab/train002/semseg/resnet50-fcn/train_resnet50_fcn.py

module restore dl_tutorial

### submit
python $RESNET50_FCN /homea/hpclab/train002/semseg/data/
/homea/hpclab/train002/semseg/models/resnet50_fcn_weights.hdf5 True False
```

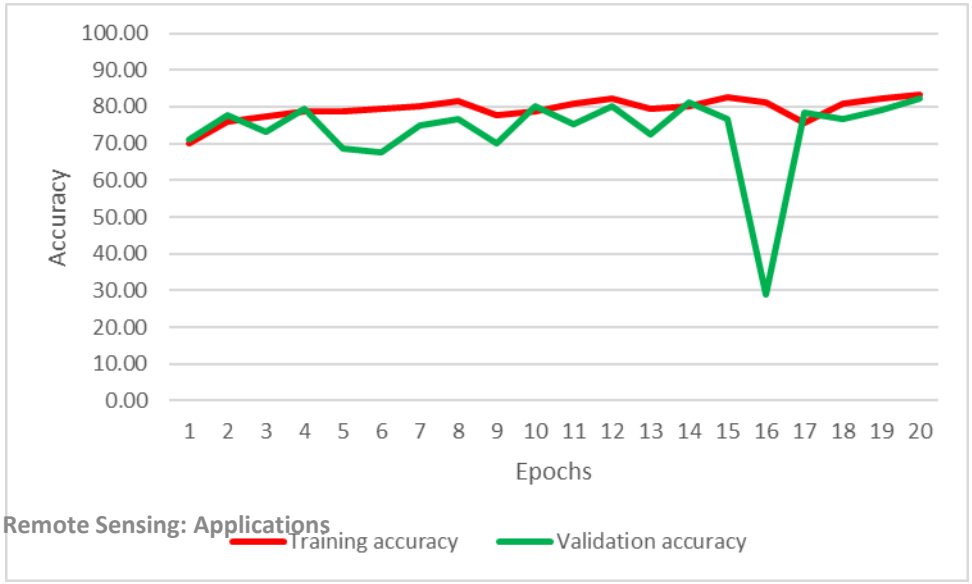
Augmentation

Transfer Learning

ResNet50 FCN Trained Model (20 Epochs)

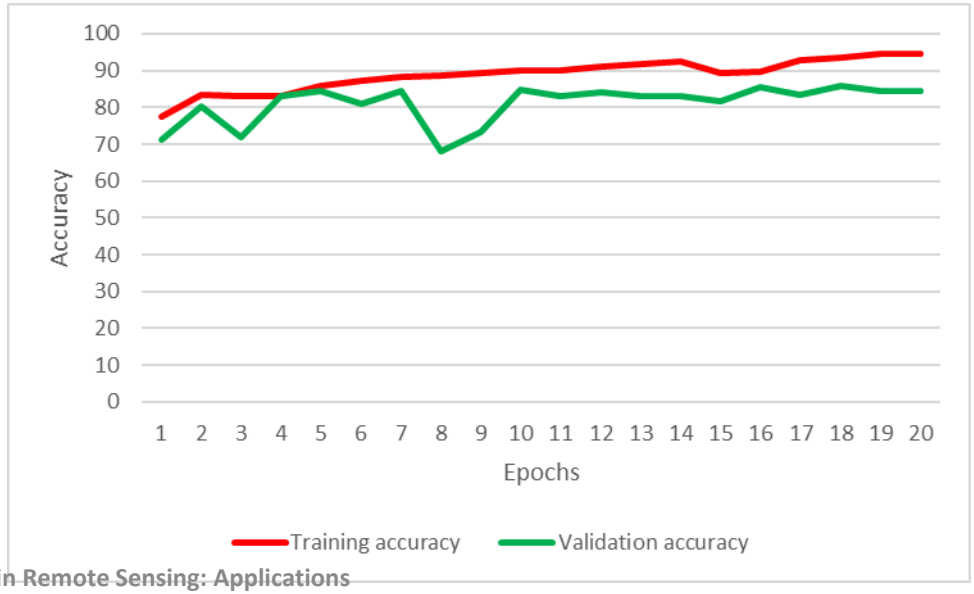
Check the output files: train_resnet50_fcn_out
train_resnet50_fcn_err

```
==================================================================
================================
Total params: 23,609,234
Trainable params: 23,556,114
Non-trainable params: 53,120

__________________________________________________________________
_______________________________
compile model ...
Train on 4166 samples, validate on 736 samples
Epoch 1/20
4166/4166 [==============================] - 102s 24ms/step -
loss: 0.9315 - acc: 0.6992 - val_loss: 0.7566 - val_acc: 0.7103
                              ..............
Epoch 20/20
4166/4166 [==============================] - 67s 16ms/step -
loss: 0.4269 - acc: 0.8338 - val_loss: 0.4783 - val_acc: 0.8239
Training time: 1448.95877409 seconds
```

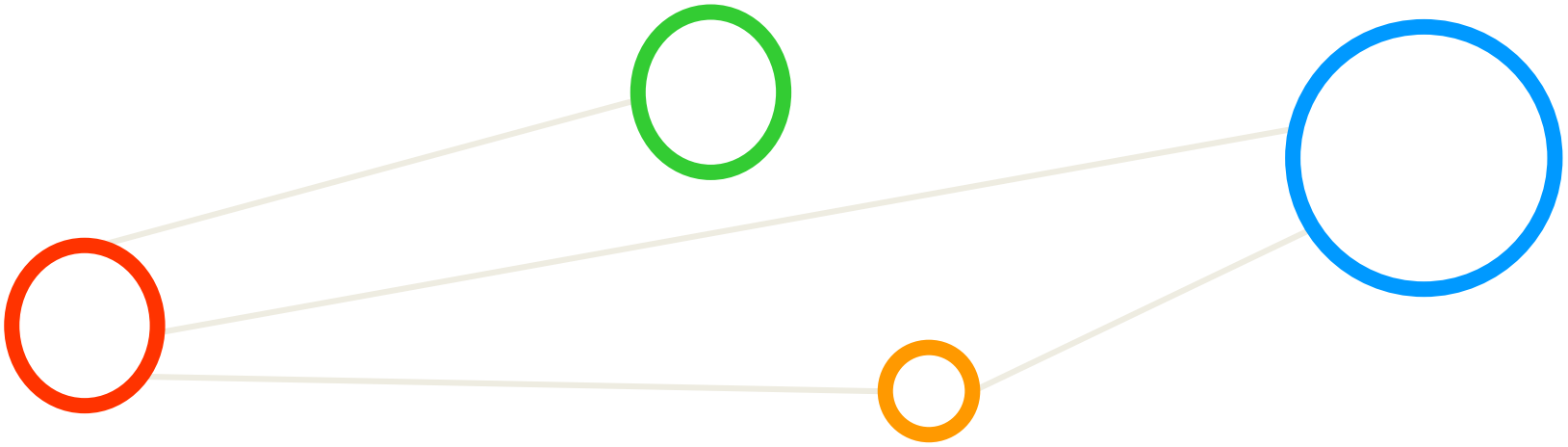


Pre-Trained ResNet50 Trained Model (20 Epochs)

```
==================================================================
================================
Total params: 23,609,234
Trainable params: 23,556,114
Non-trainable params: 53,120

__________________________________________________________________
_________________________________
compile model ...
Train on 4166 samples, validate on 736 samples
Epoch 1/20
4166/4166 [==============================] - 92s 22ms/step -
loss: 0.6261 - acc: 0.7748 - val_loss: 1.2206 - val_acc: 0.7112
                          ..............
Epoch 20/20
4166/4166 [==============================] - 67s 16ms/step -
loss: 0.1395 - acc: 0.9450 - val_loss: 0.5231 - val_acc:  0.8440
Training time: 1439.80376601 seconds
```



Practical 6:
Test the Models



Test Set

Vaihingen_15

Vaihingen_23

Thematic classes:

- ***Impervious surfaces***
- ***Building***
- ***Low vegetation***
- ***Tree***
- ***Car***
- ***Clutter/background***

Test ResNet50 FCN

- Use the function ***~/semseg/*resnet50-fcn/evaluate_network.py**
- **Run the test on the login node (i.e., no batch script submission)**
- **Setup the Python environment: $ module restore dl_tutorial**
- ***evaluate_network.py*** requires 2 parameters

```
************************************
Two parameters need to be specified:
1. The number of the patch to test (e.g., 15 ,23)
2. The network model (e.g., /homea/hpclab/train002/semseg/models/resnet50_fcn_we
ights.hdf5)
************************************
```

- **Run the test on the Vaihingen 15:**
  $ python evaluate_network.py 15 ~/semseg/models/resnet50_fcn_weights.hdf5

Or

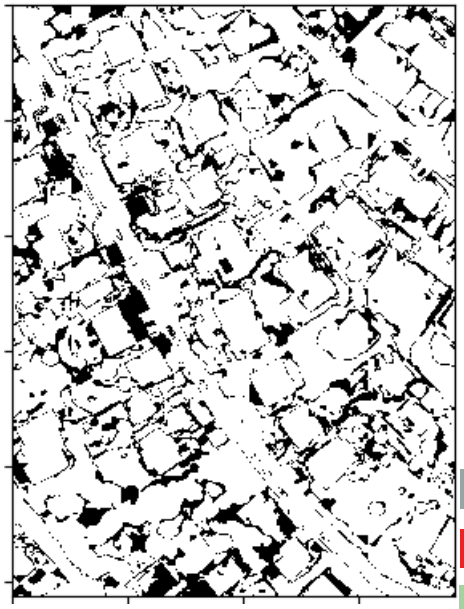
- **Run the test on the Vaihingen 23:**
  $ python evaluate_network.py 23 ~/semseg/models/resnet50_fcn_weights.hdf5

Experiment 1: Test ResNet50 FCN

Vaihingen_15

Groundtruth

Classification map



Classification errors

OA: 80.68%

Vaihingen_23

Groundtruth

Classification map

Classification errors

OA: 78.78%

Impervious surfaces

Building

Low vegetation

Tree

Car

Clutter/background

Experiment 1: Test ResNet50 FCN

Vaihingen_15

Class	Pixels	Accuracy
Impervious surfaces	855112	79.3%
Building	1075170	84.8%
Low vegetation	1309897	80.1%
Tree	1643189	80.3%
Car	31684	39.9%
Clutter/background	7183	0.0%

Confusion matrix

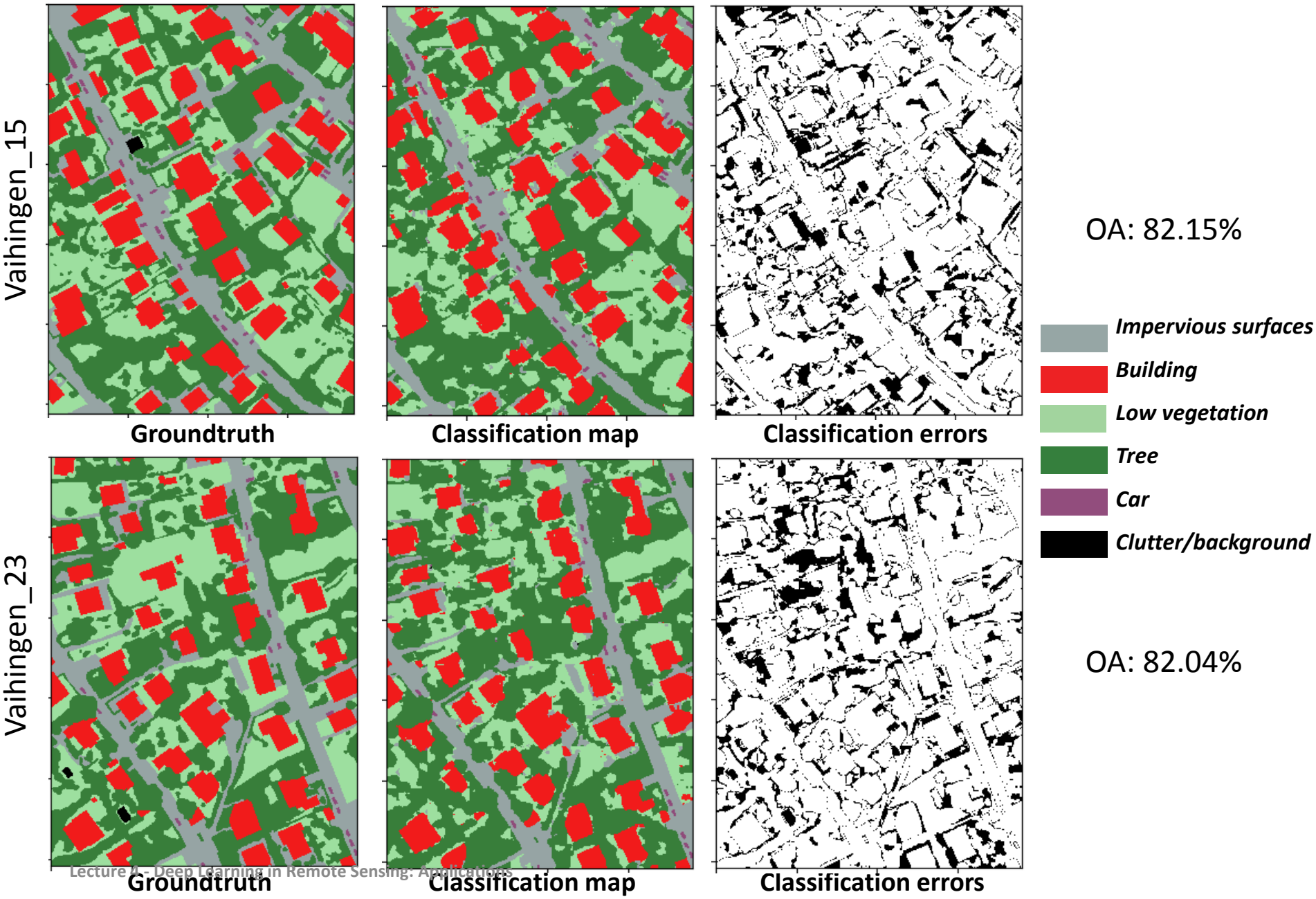
ACTUAL \ PREDICTED						
	678470	27787	111927	35596	1332	0
	77693	911674	73864	11848	91	0
	50176	35116	1049665	174888	44	8
	15034	4887	304333	1318935	0	0
	18528	13	181	305	12657	0
	226	581	6103	13	260	0

Vaihingen_23

Class	Pixels	Accuracy
Impervious surfaces	801652	72.3%
Building	885284	82.8%
Low vegetation	1345728	79.3%
Tree	1789171	79.9%
Car	15447	42.3%
Clutter/background	7756	0.0%

ACTUAL \ PREDICTED						
	579374	27187	153102	40069	581	1339
	62403	733366	78575	9724	22	1194
	43527	23396	1067768	210626	411	0
	14370	5981	338902	1429851	0	67
	8464	92	189	164	6538	0
	0	1614	5929	213	0	0

Experiment 2: Test Pre-Trained ResNet50 with ImageNet



Experiment 2: Test Pre-Trained ResNet50 with ImageNet

Vaihingen_15

Class	Pixels	Accuracy
Impervious surfaces	855112	75.6%
Building	1075170	93.6%
Low vegetation	1309897	72.2%
Tree	1643189	86.4%
Car	31684	81.9%
Clutter/background	7183	0.0%

Confusion matrix

ACTUAL \ PREDICTED						
	646590	40574	102219	60719	5003	7
	12621	1006203	39785	16494	67	0
	50571	49736	945549	263760	281	0
	17049	7515	199078	1419534	8	5
	5685	24	6	6	25963	0
	634	1	6337	0	211	0

Vaihingen_23

Class	Pixels	Accuracy
Impervious surfaces	801652	76.2%
Building	885284	94.6%
Low vegetation	1345728	66.6%
Tree	1789171	90.4%
Car	15447	84.9%
Clutter/background	7756	0.0%

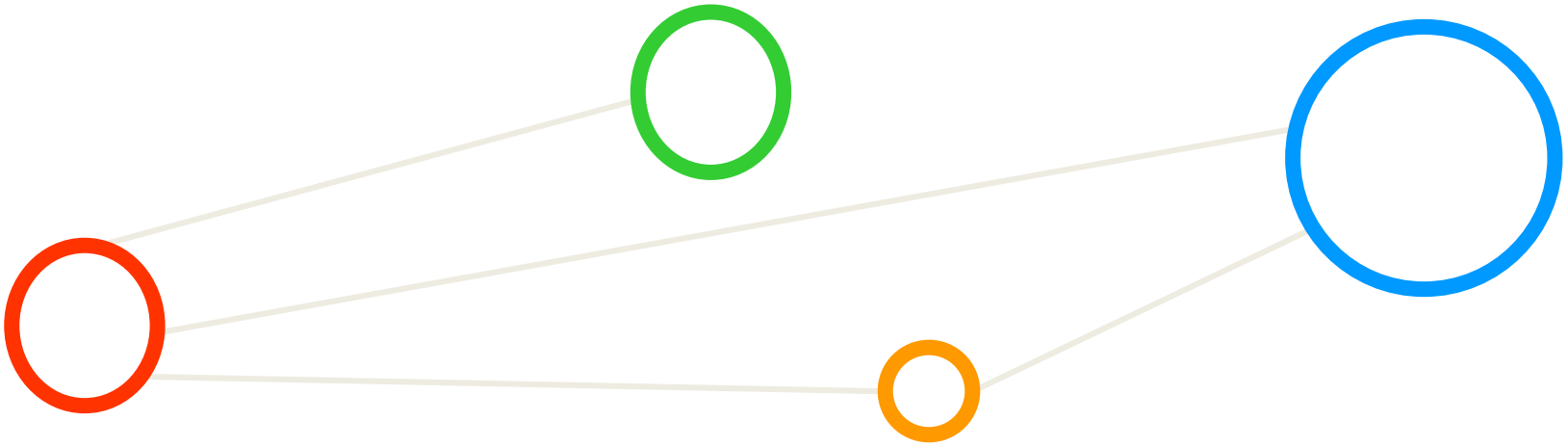
ACTUAL \ PREDICTED						
	611119	34778	93020	60029	2482	224
	8924	837288	21510	17213	279	70
	54358	39947	896551	354659	197	16
	17677	8520	145840	1616651	361	122
	2270	7	0	55	13115	0
	112	239	7290	115	0	0

Results Summary

Vaihingen 15	OA
ResNet50 FCN	80.68%
Pre-trained ResNet50	82.15%

Vaihingen 23	OA
ResNet50 FCN	78.78%
Pre-trained ResNet50	82.04

Tutorial Competition



Our Competition

Vaihingen_15

WG	ACCURACY	TRAINING TIME
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

WG	ACCURACY	TRAINING TIME
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		

Optimizers

SGD 10 1 0.1 0.001 0.0001

RMSprop

Adagrad

Adadelta

Adam

Adamax

What can you change?

- ***/homea/hpclab/train001/……./data_io.py***

```
59  def generate_dataset(image_numbers, overlap_factor):
60      # input / output size of the FCN:
61      size = 256
```

```
113     # generate and save the training and validation set:
114     overlap = 0.6
```

LIST TO BE DONE
-Number of epochs
-More augmented data
……..

Multiple GPUS

Lecture Bibliography (1)

- [1] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, J. Garcia-Rodriguez, “A Review on Deep Learning Techniques Applied to Semantic Segmentation”, in CoRR, 2017.
  Online: http://arxiv.org/abs/1704.06857
- [2] Image Segmentation Using DIGITS 5
  Online: https://devblogs.nvidia.com/image-segmentation-using-digits-5/
- [3] J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, 2015, pp. 3431-3440.
- [4] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770-778.
- [5] The vanishing gradient problem
  Online: http://adventuresinmachinelearning.com/vanishing-gradient-problem-tensorflow/
- [6] Deep Residual Learning for Image Recognition
  Online: http://neural.vision/blog/article-reviews/deep-learning/he-resnet-2015/

Lecture Bibliography (2)

- [15] Introduction to Remote Sensing: Radiometric Corrections

  Online: http://gsp.humboldt.edu/olm_2015/Courses/GSP_216_Online/lesson4-1/radiometric.html
- [16] G. Hughes, "On the mean accuracy of statistical pattern recognizers," *in IEEE Transactions on Information Theory*, vol. 14, no. 1, pp. 55-63, 1968

  Online: http://ieeexplore.ieee.org/document/1054102/?reload=true&tp=&arnumber=1054102
- [17] Normalized Difference Vegetation Index (NDVI)

  Online: http://www.agasyst.com/portals/NDVI.html
- [18] NDVI & Classification

  Online: https://lholmesmaps.wordpress.com/my-work-2/environmental-studies-421-gis-iv-advanced-gis-applications/2-2/
- [19] H. Hotelling, “Analysis of a complex of statistical variables into principal components", in Journal of Educational Psychology, 24, 417–441, and 498–520, 1933

  Online: https://www.scribd.com/document/59617538/Analysis-of-a-Complex-of-Statistical-Variables-Into-Principal-Components

Lecture Bibliography (3)

- [20] A. Plaza, G. Martín, J. Plaza, M. Zortea and S. Sánchez, “Recent Developments in Endmember Extraction and Spectral Unmixing“ , in Optical Remote Sensing, vol 3. Springer, Berlin, Heidelberg, 2011
  Online: https://link.springer.com/chapter/10.1007/978-3-642-14212-3_12
- Online: http://ieeexplore.ieee.org/document/7486184/
- Online: http://ieeexplore.ieee.org/document/7842555/
- [23] Rome Image dataset
  Online: https://b2share.eudat.eu/records/daf6c389e54340b4b1416cf874251e77
- [24] G. Cavallaro, M. Riedel, et al., "Smart data analytics methods for remote sensing applications," in the Proceedings of the IEEE Geoscience and Remote Sensing Symposium, Quebec City, QC, 2014, pp. 1405-1408.
- [25] Airborne Visible/Infrared Imaging Spectrometer (AVIRIS)
  Online: https://www.jpl.nasa.gov/missions/airborne-visible-infrared-imaging-spectrometer-aviris/
- [26] Indian Pines dataset: 220 Band AVIRIS Hyperspectral Image
  Online: https://purr.purdue.edu/publications/1947/1
- [27] G. Cavallaro, M. Riedel, M. Richerzhagen, J. A. Benediktsson and A. Plaza, "On Understanding Big Data Impacts in Remotely Sensed Image Classification Using Support Vector Machine Methods," in the IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 8, no. 10, pp. 4634-4646, Oct. 2015.
- [28] Indian Pines Raw and Processed
  Online: http://hdl.handle.net/11304/9ec5eac8-61b4-4617-ae1c-1f8c8cd3cd74
- [29] YouTube Video, “What is Remote Sensing?”
  Online: https://www.youtube.com/watch?v=nU-CjAKry5c