Contents

Preface ix

Author xi

I Backgrounds 1

1 Deep learning background 3

1.1 What is deep learning? 3

1.2 Convolution 5

1.3 Pooling 6

1.4 Activation functions 7

1.5 Challenges ahead for deep learning with remote sensing images 8

2 Software 9

2.1 Orfeo ToolBox 9

2.1.1 Applications 10

2.1.2 Streaming mechanism 10

2.1.3 Remote modules 10

2.1.4 The Python API 10

2.2 TensorFlow 11

2.2.1 APIs 11

2.2.2 Computations 11

2.2.3 Graphs 12

2.3 Orfeo ToolBox + TensorFlow = OTBTF 12

2.3.1 Installation 13

2.3.2 Featured applications 14

2.3.3 Principle 14

2.3.4 Multiple input sources and outputs 16

2.4 QGIS 17

II Patch-based classification 19

3 Introduction 21

4 Data used: the Tokyo dataset **23**
4.1 Description 23
4.2 Remote sensing imagery 23
4.3 Terrain truth 25

5 A simple convolutional neural network **27**
5.1 Normalization 27
5.2 Sampling 29
5.2.1 Selection 29
5.2.2 Extraction 30
5.3 Training 32
5.3.1 Principle 32
5.3.2 Model architecture 33
5.3.2.1 Input 34
5.3.2.2 Layers 34
5.3.2.3 Estimated class 34
5.3.2.4 Loss function 34
5.3.2.5 Optimizer 35
5.4 Generate the model 35
5.5 Train the model from scratch 37
5.6 Comparison with Random Forest 40
5.7 Inference 41

6 Fully Convolutional Neural Network **45**
6.1 Using the existing model as an FCN 45
6.2 Pixel-wise fully convolutional model 46
6.3 Training 49
6.4 Inference 50

7 Classifiers on deep features **51**
7.1 Principle 51
7.2 Overview of composite applications in OTB 52
7.3 Training 53
7.4 Inference 54

8 Dealing with multiple sources **55**
8.1 More sources? 55
8.2 Model with multiple inputs 56
8.3 Normalization 59
8.4 Sampling 60
8.5 Training 61
8.5.1 Inference 62
8.5.1.1 Patch-based mode 62
8.5.1.2 Fully convolutional mode 62

9 Discussion **65**

III Semantic segmentation **67**

10 Semantic segmentation of optical imagery **69**
10.1 Introduction 69
10.2 Overview 69

11 Data used: the Amsterdam dataset **73**
11.1 Description 73
11.2 Spot-7 image 73
11.3 OpenStreetMap data 74
11.3.1 OSM downloader plugin 75
11.3.2 Download OSM data 75
11.3.3 Prepare the vector layer 77

12 Mapping buildings **83**
12.1 Input data pre-processing 83
12.1.1 Satellite image pansharpening 83
12.1.2 Image normalization 84
12.1.3 Sample selection 84
12.1.3.1 Patch position seeding 84
12.1.3.2 Patch position selection 86
12.1.3.3 Patches split 87
12.1.4 Rasterization 88
12.1.5 Patch extraction 89
12.2 Building the model 90
12.2.1 Architecture 91
12.2.2 Implementation 92
12.2.2.1 Exact output 94
12.2.2.2 Expression field 95
12.2.3 Generate the SavedModel 95
12.3 Training the model 96
12.4 Inference 96

13 Discussion **99**

IV Image restoration **101**

14 Gapfilling of optical images: principle **103**
14.1 Introduction 103
14.2 Method 103
14.3 Architecture 105
14.3.1 Encoder 106
14.3.2 Decoder 106
14.3.3 Loss 107

15 The Marmande dataset **109**
15.1 Description 109
15.2 Sentinel-2 images 109
15.3 Sentinel-1 image 112

16 Pre-processing **115**
16.1 Sentinel images 115
16.1.1 Optical images 115
16.1.2 SAR image 117
16.1.2.1 Calibration 117
16.1.2.2 Filtering values 118
16.1.2.3 Linear stretch 118
16.1.2.4 Spatial resampling 118
16.2 Patches 119
16.2.1 Patch position seeding 119
16.2.1.1 Sentinel-2 image masks 119
16.2.1.2 Merge masks 122
16.2.1.3 Grid generation 123
16.2.1.4 Grid filtering 124
16.2.1.5 Patch centroids 126
16.2.1.6 Training and validation datasets 126
16.2.2 Extraction of patches 127
16.3 More: automate steps with the OTB Python API 128
16.3.1 Build the pipeline 129
16.3.2 Run the pipeline 132

17 Model training **133**
17.1 Training from Python 133
17.2 Get the code 134
17.3 Use the code 134
17.3.1 Description 134
17.3.2 Parameters 134
17.4 Export the model 135

18 Inference **137**
18.1 Inputs and outputs 137
18.2 Generating the image 137
18.3 Postprocessing 139

19 Discussion **143**

Bibliography **145**

Index **149**