MULTISPECTRAL REMOTE SENSING

Master's Degree in Geospatial Engineering

LABORATORY PROJECT 2

DEEP LEARNING FOR BUILDING DETECTION USING HIGH-RESOLUTION IMAGES	
OBJECTIVE	Development of a complete workflow for building segmentation in Greater Lisbon using U-Net (PyTorch) and high-resolution orthophotos
DURATION	5 May 2026 – 26 May 2026
EVALUATION	Oral presentation – 2 June 2026
BIBLIOGRAPHY	The U-Net : A Complete Guide | Medium Building a U-Net Architecture for Image Segmentation with Python and Keras | by Musti Öztemiz | Medium Cook your First U-Net in PyTorch. A magic recipe to empower your image... | by Mostafa Wael | TDS Archive | Medium UNet for Building Segmentation (PyTorch) https://www.youtube.com/watch?v=IHq1t7NxS8k https://campus.datacamp.com/courses/deep-learning-for-images-with-pytorch/image-segmentation?ex=9

Building detection with U-Net workflow

STEP 1 **Area of interest (AOI) definition and data preparation in QGIS**	**1.1 Define the AOI** Select a sub-region inside the high-resolution orthophoto, mainly covered by buildings **1.2 Download building data from OSM**[1] **1.3 Load the high-resolution orthophoto** **1.4 Check CRS consistency** Both layers must use the same CRS; if different, reproject both to EPSG: 32629 **1.5 Clip data using AOI** Clip both the raster and vector data using AOI ***Output: AOI ortophoto and AOI building shapefile***
STEP 2 **Data preprocessing in PYTHON**	**2.1 Load input data (orthophoto and building shapefile)** **2.2 CRS check** **2.2 Convert the buildings vector file into a raster mask** **2.4 Save ground truth mask** ***Output: binary building mask (0= background, 1= building)***
STEP 3 **Tile generation for U-NET in PYTHON**	3**.1. Generate training patches** Split orthophoto and mask into tiles (256 x 256 pixels) ***Output: image tiles (X) and mask tiles (y)***
STEP 4 **U-Net model training in PYTHON**	**4.1 Define the model architecture** Encoder: ResNet34 (pretrained on ImageNet) Decoder: U-Net with skip connections **4.2 Training configuration** Loss function: BCE + Dice Optimizer: Adam (lr= 1e-3) Epochs: 10 to 30 ***Output: trained U-Net model for building segmentation***

[1] https://download.geofabrik.de/europe/portugal.html

STEP 5 **Prediction and post-processing in PYTHON**	**5.1 Sliding window prediction** Split the image into overlapping patches (e.g., 256 x 256); run prediction for each patch; store outputs in a full-size probability map ***Output: raw probability map (continuous values 0-1)*** **5.2 Seamless reconstruction** Combine overlapping predictions by averaging overlapping regions for a smooth output (removes tile boundary artifacts) ***Output: seamless probability raster (no tile borders)*** **5.3 Thresholding (binary building map)** Convert probabilities to classes by applying a threshold (commonly 0.5); if >= 0.5, building (1), else background (0) ***Output: binary building map*** **5.4 Export the prediction as GeoTIFF** ***Output: Geotiff file fully compatible with QGIS and GIS workflows*** **5.5 Evaluate the model performance** Compare the predicted map versus the ground truth mask • IoU (spatial overlap quality) • Dice (segmentation similarity, especially effective for buildings) • Accuracy (pixel-wise correctness) ***Output: quantitative performance report***

U-NET HYPERPARAMETERS

Hyperparameter	Meaning	Standard Values
Epochs (EPOCHS)	The number of times the model sees the entire training dataset during training	50-200 epochs are common for U-Net training *Too few epochs: underfitting (model does not learn enough patterns)* *Too many: overfitting (model memorizes training data and performs poorly on new data)*
Patch size (PATCH_SIZE)	Dimensions of the image tiles (patches) used as input to the network	128 x 128, 256 x 256, or 512 x 512 *Small patches: less spatial context but lower GPU memory* *Large patches: more contextual information and better object continuity, but require more GPU memory and longer training time*
Prediction stride (stride)	Distance (in pixels) that the sliding window moves when extracting patches; controls patch overlap	128 with a patch size of 256 produces 50% overlap *Smaller stride: more overlap, smoother predictions, better border continuity, but more computation* *Larger stride: faster processing, but may create visible tile boundaries*
Batch size (batch_size)	The number of image patches processed together before updating weights	2-16 are typical values for segmentation (8 is a common compromise between stability and memory usage) *Small batches (2-8): lower memory consumption but noisier gradient updates* *Large batches (32-64): more stable and faster training, but require substantially more GPU memory*
Number of filters	Number of feature maps learned at each convolution level; determines model capacity and feature extraction complexity	64 → 128 → 256 → 512 → 1024 (typical U-Net architecture) More filters: improved feature extraction and potentially higher accuracy; however, this also increases computational cost, training time, and GPU memory requirements
Optimizer (optimizer)	Algorithm responsible for updating model weights during training	Common optimizers: Adam: fast convergence and stable training; most used in U-Net SGD: slower training, but may provide better generalization RMSprop: common in medical image segmentation
Learning rate (lr)	Controls how much the network weights are adjusted after each training step	0.001 (1e-3) or 0.0001 (1e-4) (a suitable learning rate produces a smooth and progressive decrease in training loss) *Too high: unstable training and oscillating loss* *Too low: very slow convergence*
Loss function (BCE+Dice)	Measures the segmentation error by comparing predictions with reference labels	BCE + Dice loss (balances pixel accuracy and object shape quality) *BCE (Binary Cross Entropy): evaluates pixel-wise classification accuracy* *Dice loss: evaluates spatial overlap between prediction and ground truth*
Activation function (ReLU)	Introduces non-linearity into the network, enabling the model to learn complex spatial patterns	ReLU (Rectified Linear Unit): converts negative values to zero; computationally efficient Sigmoid: converts outputs to probabilities between 0 and 1 (commonly used in binary segmentation output layers) Softmax: used for multiclass segmentation problems
Threshold (threshold)	Converts predicted probabilities into final binary segmentation classes; Pixels above the threshold are assigned to the positive class (in our case, buildings)	0.5 (threshold selection strongly influences the final appearance and accuracy of the segmentation map) *Lower threshold (0.3-0.4): detects more objects and reduces false negatives (FN), but increases false positives (FP) and noise* *High threshold (0.7-0.8): cleaner segmentation with fewer false positives (FP), but may miss small or uncertain objects and produce fragmented results*

ACCURACY METRICS (BINARY SEGMENTATION)

```
================================
Final full-image metrics
================================
Accuracy : 0.8691
Precision: 0.8855
Recall   : 0.8713
F1-score : 0.8783
IoU      : 0.7831

Confusion matrix
Rows = reference mask
Columns = prediction

            Pred 0    Pred 1
True 0     3443740   530487
True 1      605496  4100723
```

	Pred 0 (Background)	Pred 1 (Buildings)
True 0 (Background)	**True Negative (TN)**	**False Positive (FP)**
True1 (Buildings)	**False Negative (FN)**	**True Positive (TP)**

TP = correctly detected building pixels

FP = background pixels wrongly classified as buildings

FN = building pixels missed by the model

TN = correctly detected background pixels

	Pred 0 (Background)	Pred 1 (Buildings)	Total	RECALL
True 0 (Background)	**3443740**	530487	3974227	
True1 (Buildings)	605496	**4100723**	4706219	0.8713
Total	4049236	4631210	**8680446**	
PRECISION		0.8855		

$$\text{Recall} = \left(\frac{\text{TP}}{\text{TP} + \text{FN}}\right) = \left(\frac{4100723}{4706219}\right) = 0.8713$$

$$\text{Precision} = \left(\frac{\text{TP}}{\text{TP} + \text{FP}}\right) = \left(\frac{4100723}{4631210}\right) = 0.8855$$

$$\text{Accuracy} = \left(\frac{\text{TP} + \text{TN}}{\text{TN} + \text{FN} + \text{FP} + \text{TP}}\right) = \left(\frac{7544463}{8680446}\right) = 0.8691$$

$$\text{F1} - \text{score} = \left(\frac{2 * \text{TP}}{2 * \text{TP} + \text{FP} + \text{FN}}\right) = \left(\frac{8201446}{9337429}\right) = 0.8783$$

For binary segmentation $\text{F1} - \text{score} = \text{Dice coefficient}$

$$\text{IoU} = \left(\frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}\right) = \left(\frac{4100723}{5236706}\right) = 0.7831$$

IoU (Intersection over Union) also called Jaccard Index

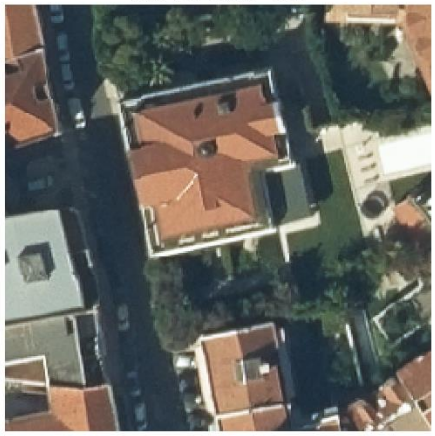
$$= \left(\frac{intersection\ of\ prediction\ and\ ground\ truth}{union\ of\ prediction\ and\ ground\ truth}\right)$$

Reference Building Mask

Building Detection - Final Prediction

Training pair: Orthophoto patch and corresponding mask

Orthophoto patch 49
row=1024, col=256

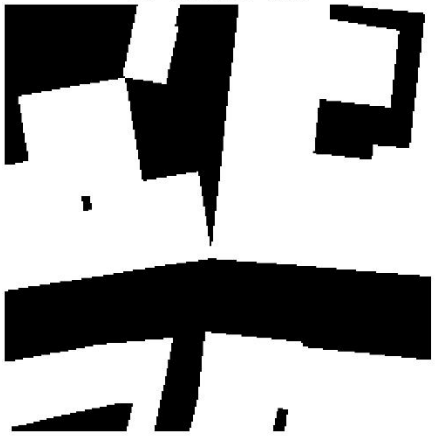


Mask patch 49
row=1024, col=256

Training pair: Orthophoto patch and corresponding mask

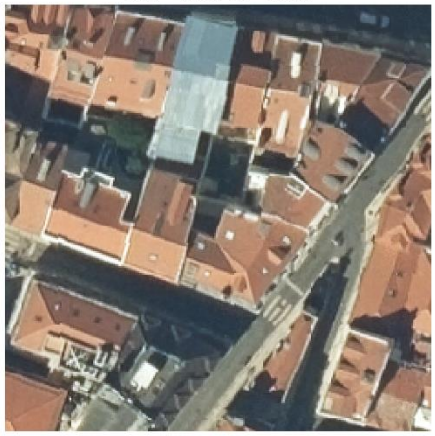
Orthophoto patch 100
row=2048, col=1024

Mask patch 100
row=2048, col=1024



Training pair: Orthophoto patch and corresponding mask

Orthophoto patch 115
row=2304, col=1792



Mask patch 115
row=2304, col=1792