Imperial College London

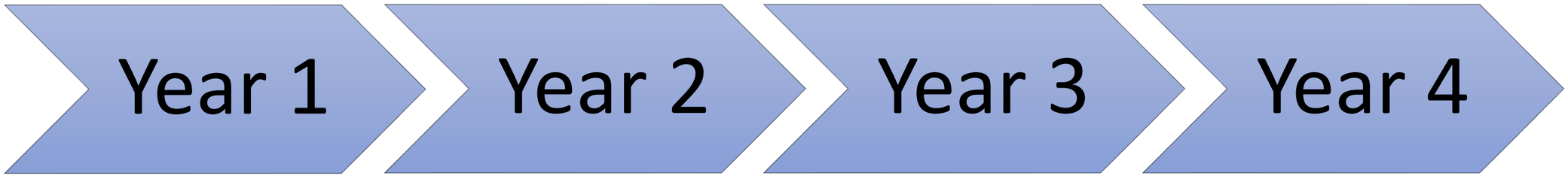
Weekly Meeting - 21062024

Ahmad Agus Widodo
PhD Student in Geotechnics Sections
Department of Civil and Environmental Engineering

Supervisors:
1. Dr Truong Le (Geotechnic Sections)
2. Dr Philippa J Mason (Royal School of Mines)

This PhD is funded by Indonesian Endowment Funds for Education (LPDP)

Research Milestone

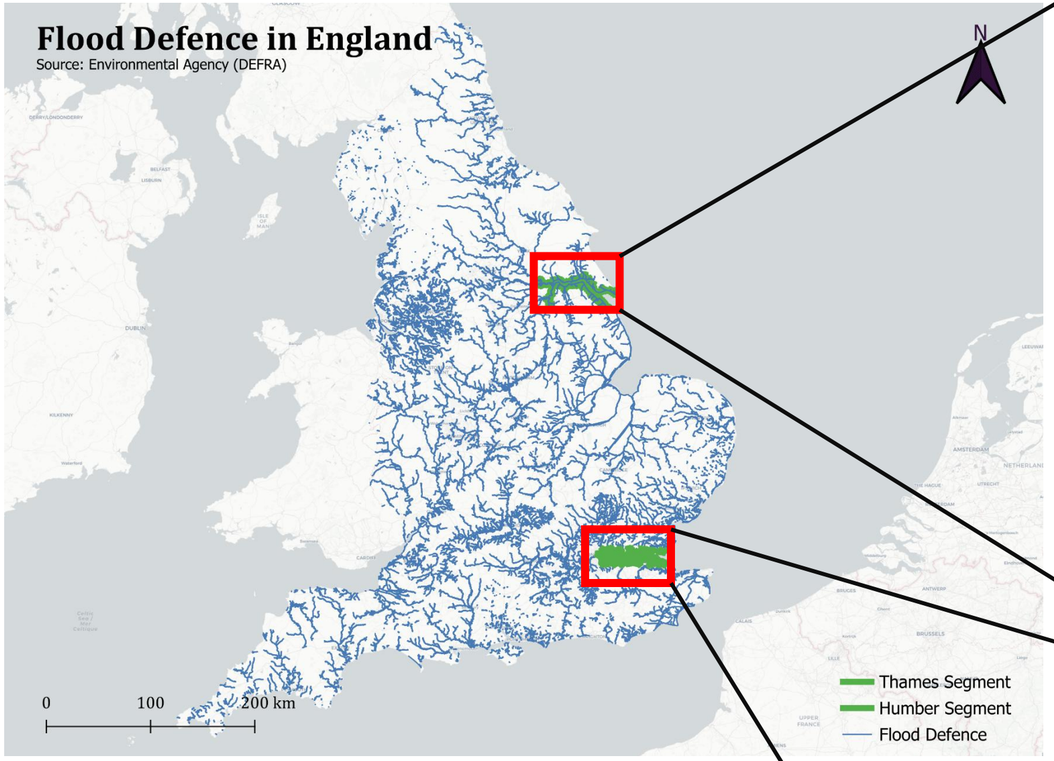


1. Segmentation Processes (focusing on algorithm of KPConv, Bi-LSTM, and Pointnet++)
2. Understanding labelling for crack/landslide

1. Labelling crack/landslide*
2. Analysing landslide data before/after landslide, calculating the rate (considering climate and environmental conditions)
3. Paper 1

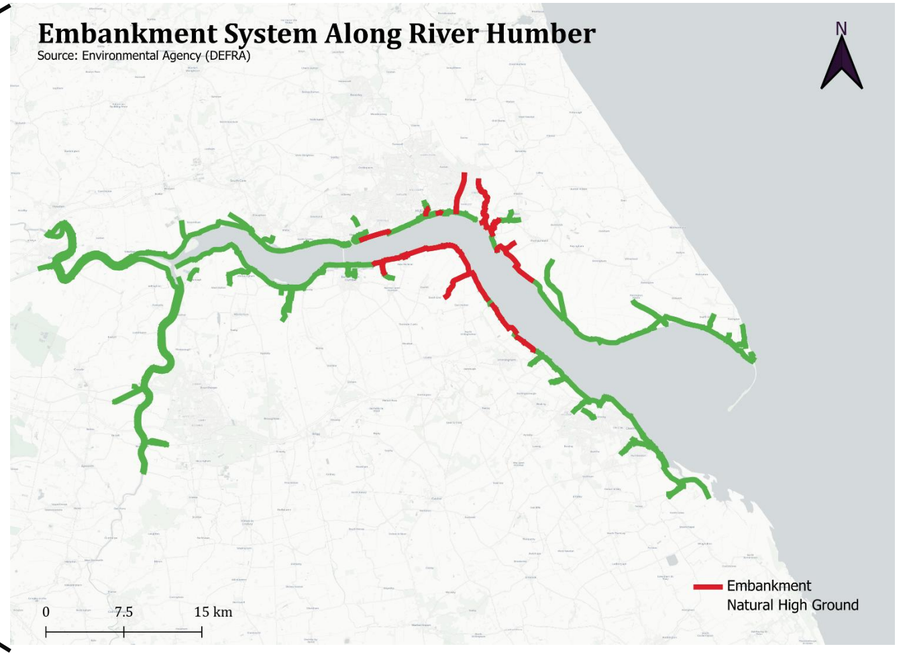
1. Predicting potential landslide rate using climate scenario (global, regional, local data) and surface features and data
2. Paper 2

Study Area 1 : Embakment / Levee

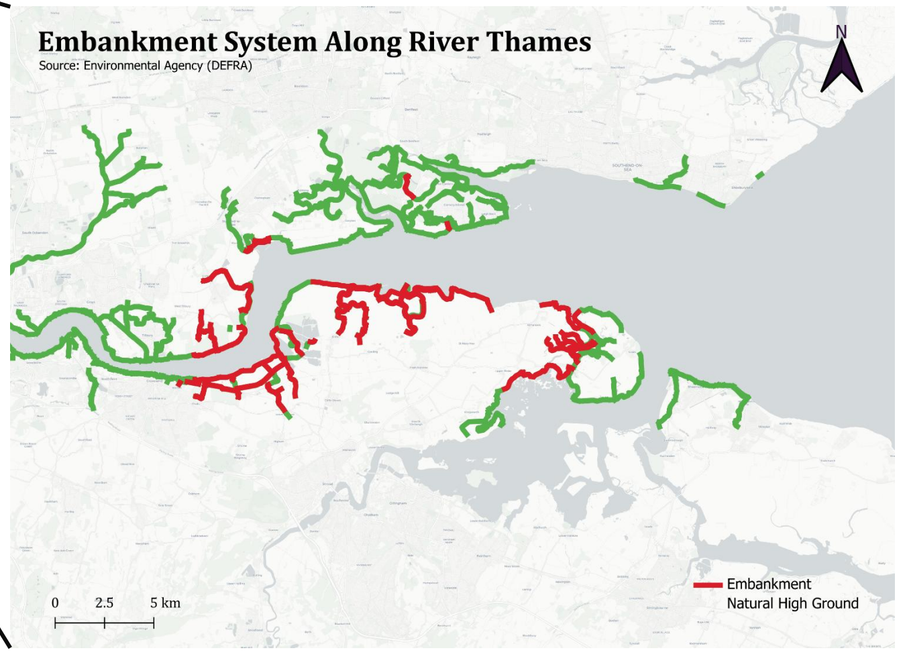


Total length of 19,506 kilometres (km) and it is divided into 8 regions:

- Southeast (2741 km)
- Northwest (2335 km)
- Anglian (2279 km)
- Southwest 1941 km)
- Thames (1558 km)
- Midlands (1395 km)
- Yorkshire (1143 km)
- Northeast (314 km).

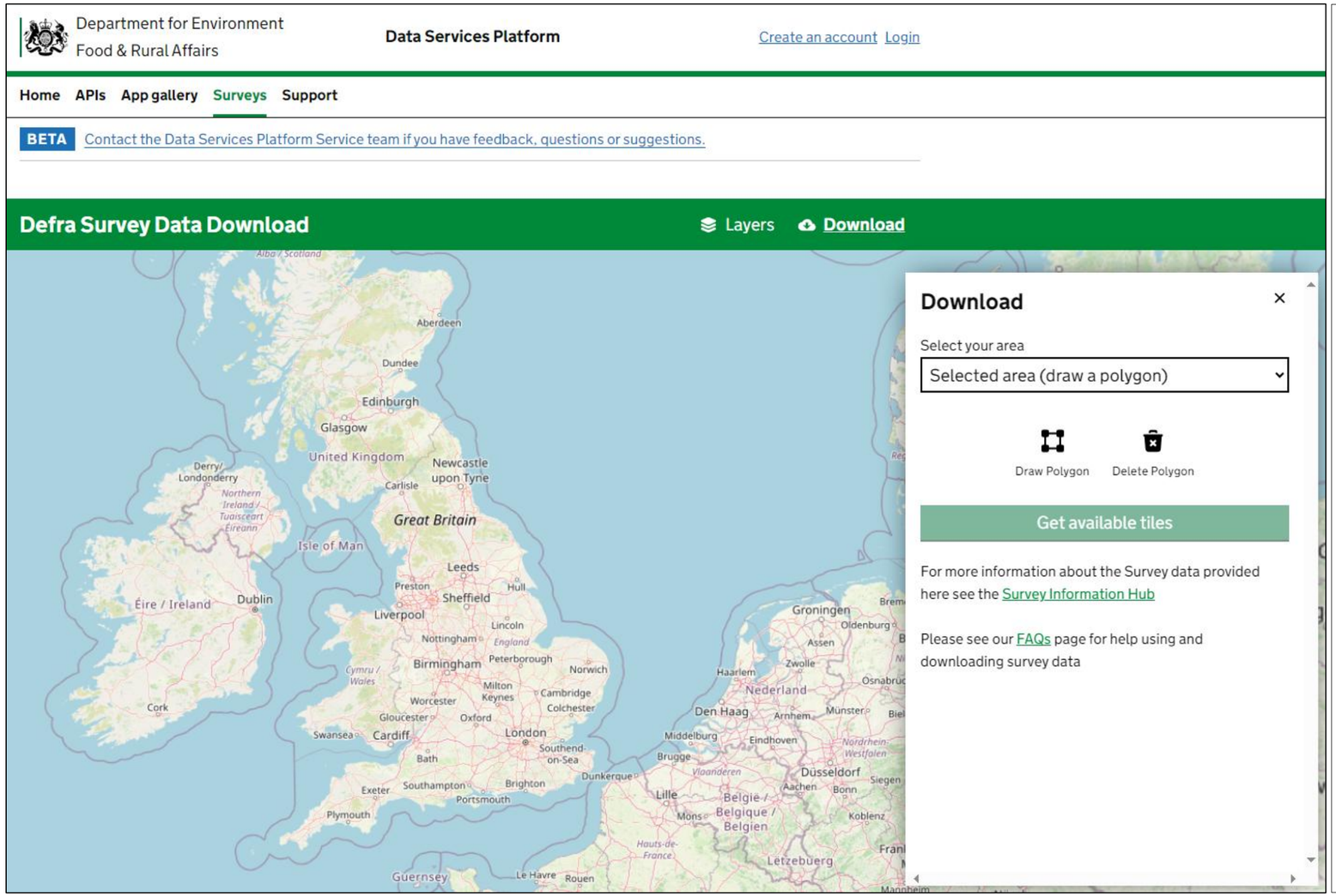


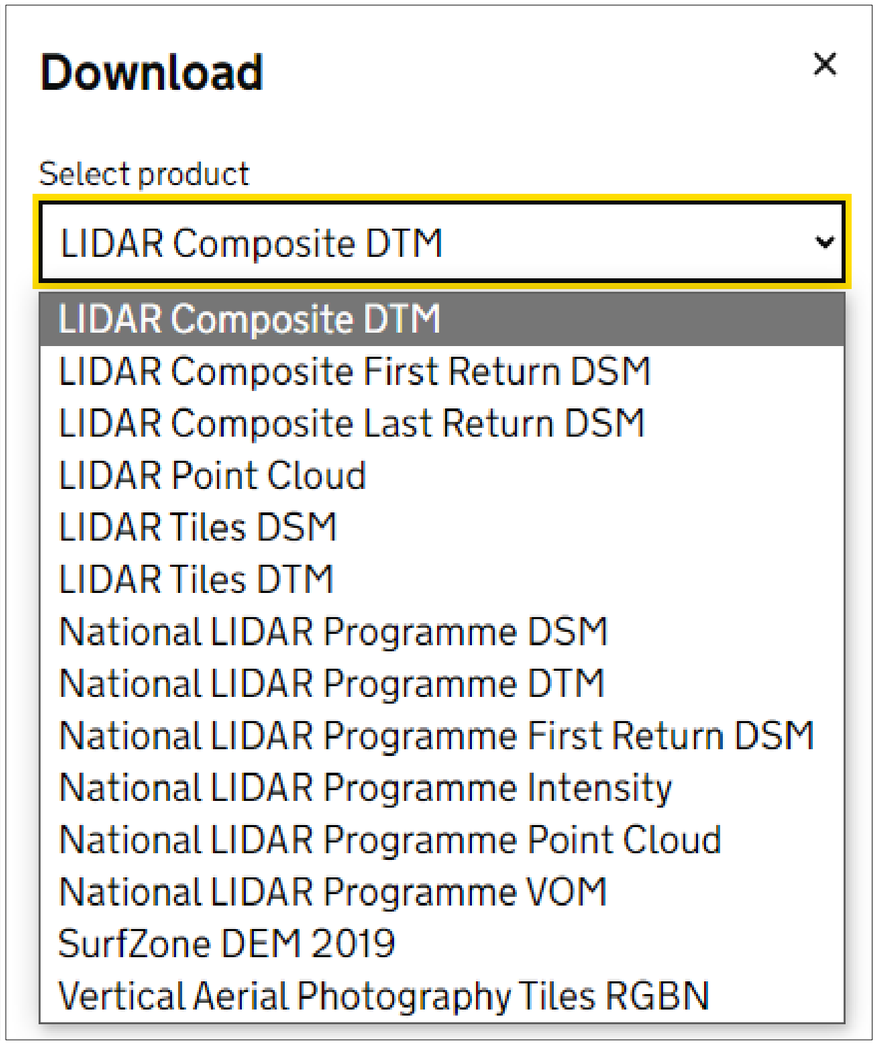
- 240 km of embankments, walls, and other flood defence structures.
- 1953 North Sea significant flooding around the Humber estuary, killing 38 people.
- In 2013, a tidal surge overpowered the defensive structures, flooding over 600 homes and inflicting.



- 370 kilometres of flood walls, embankments, and other structures.
- 1953, North Sea flood triggered by a storm surge flooded the Thames estuary, killing 307 people.
- 2014, the Thames Barrier was closed 50 times to protect London from tidal surges.

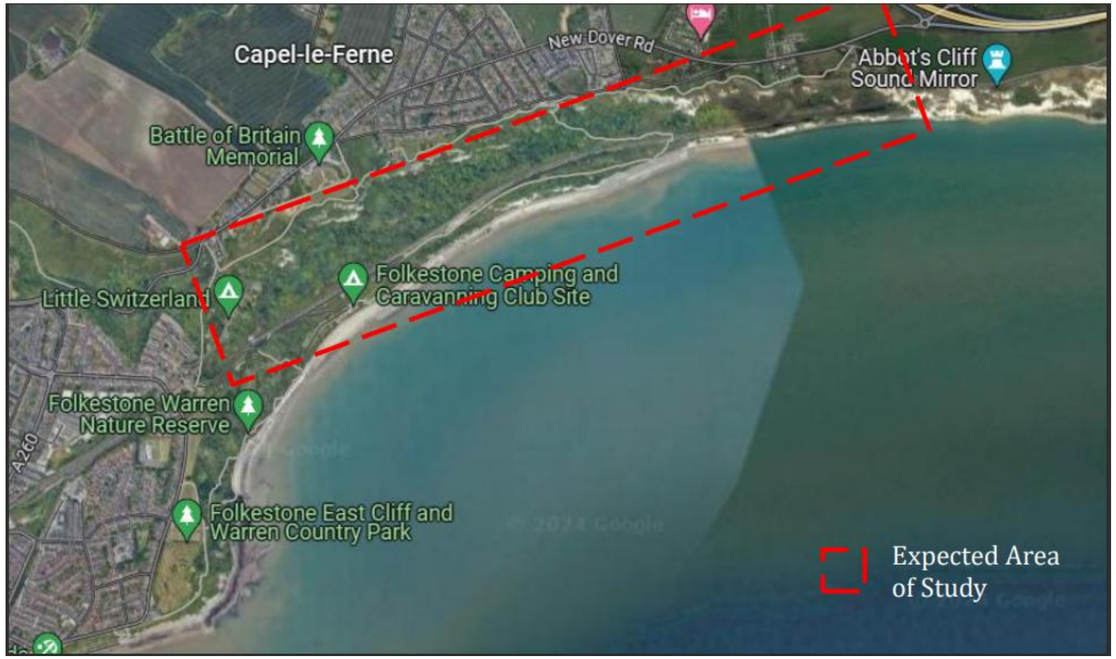
Training Data Source





Link : [Defra Survey Data Download]

Study Area 2 : Railway System



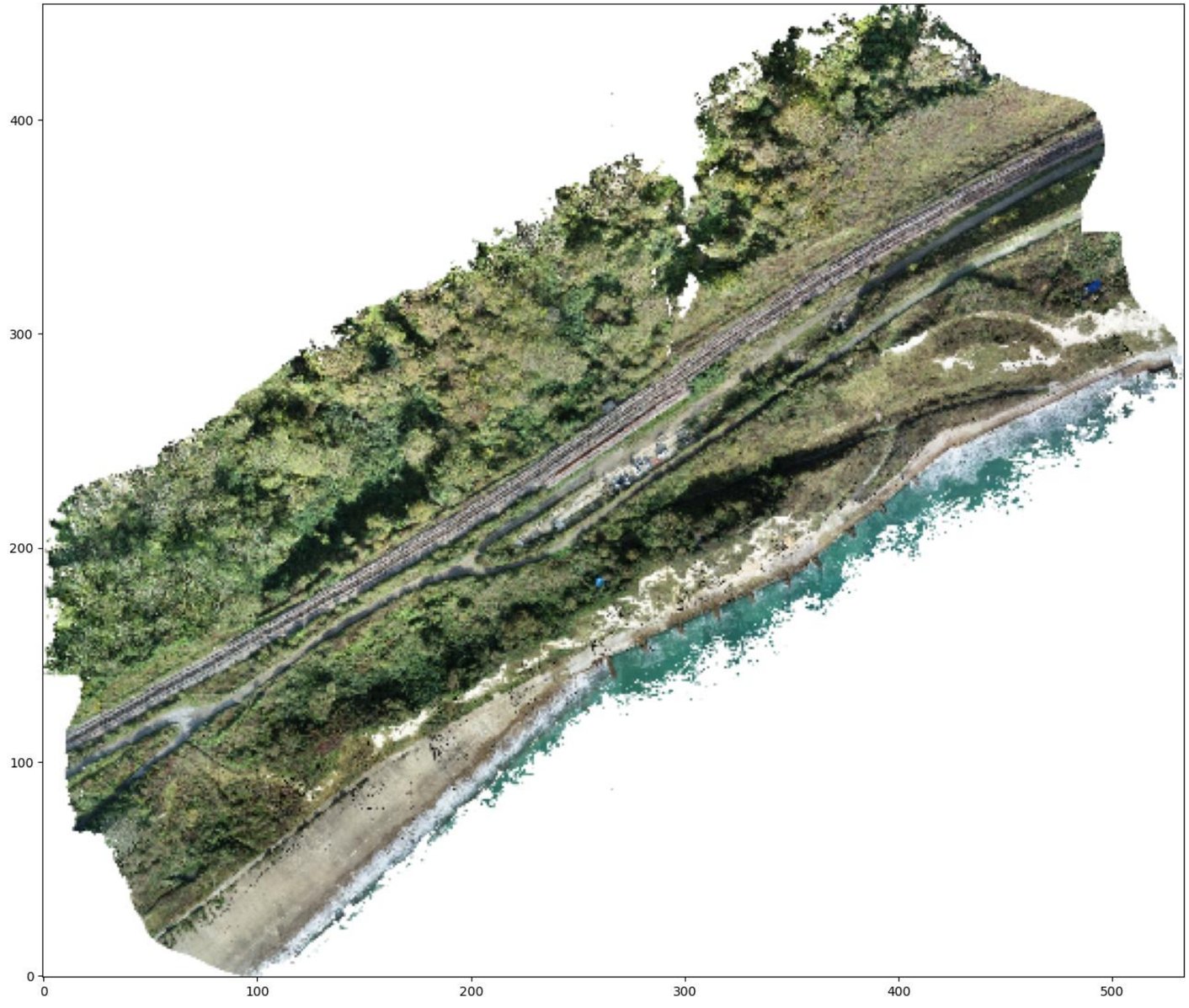
Coordinates :

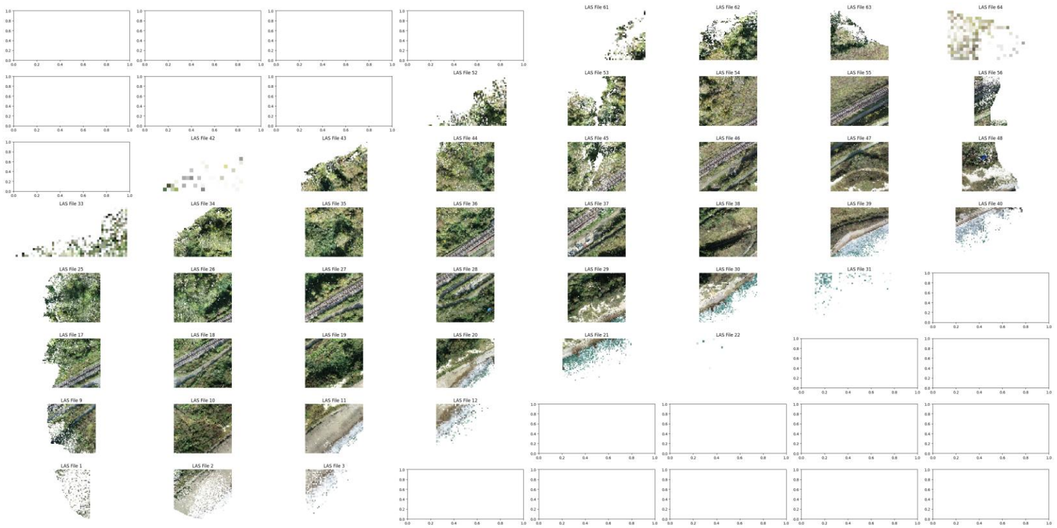
West	: 1.20078W	51.09351N
South	: 1.20421W	51.09141N
East	: 1.23910W	51.10054N
North	: 1.23798W	51.10161N

Point Cloud Data

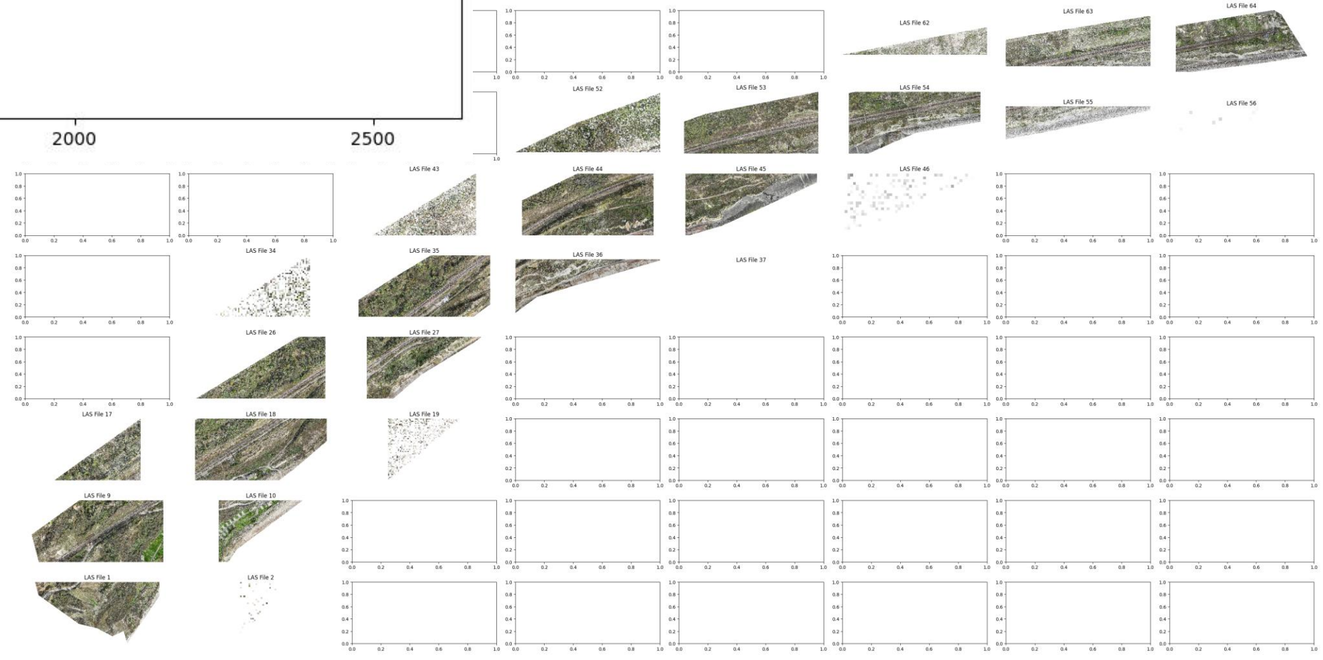
Number of Points	: 267,231,656
Data Density	:

Data LiDAR Point Cloud 2018





Data LiDAR Point Cloud 2024



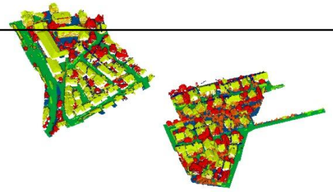
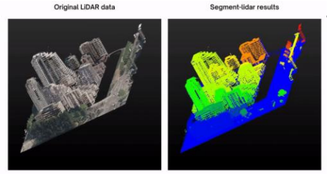
Background Data Folkestone Warren

Data 2018					
Number of Grid	Number of Point	Average Point Density	Area per Point	Average Distance estimation (m)	Average Distance estimation (cm)
44	221751985	1336.699732	0.010446693	0.051922814	5.192281423

Data 2024					
Number of Grid	Number of Point	Average Point Density	Area per Point	Average Distance estimation (m)	Average Distance estimation (cm)
25	267231654	306.1451518	0.003986271	0.061535469	6.15354688

Area per point = 1 / average point density
Average distance estimation = It is assumed that the points are uniformly distributed in square

Segmentation Process

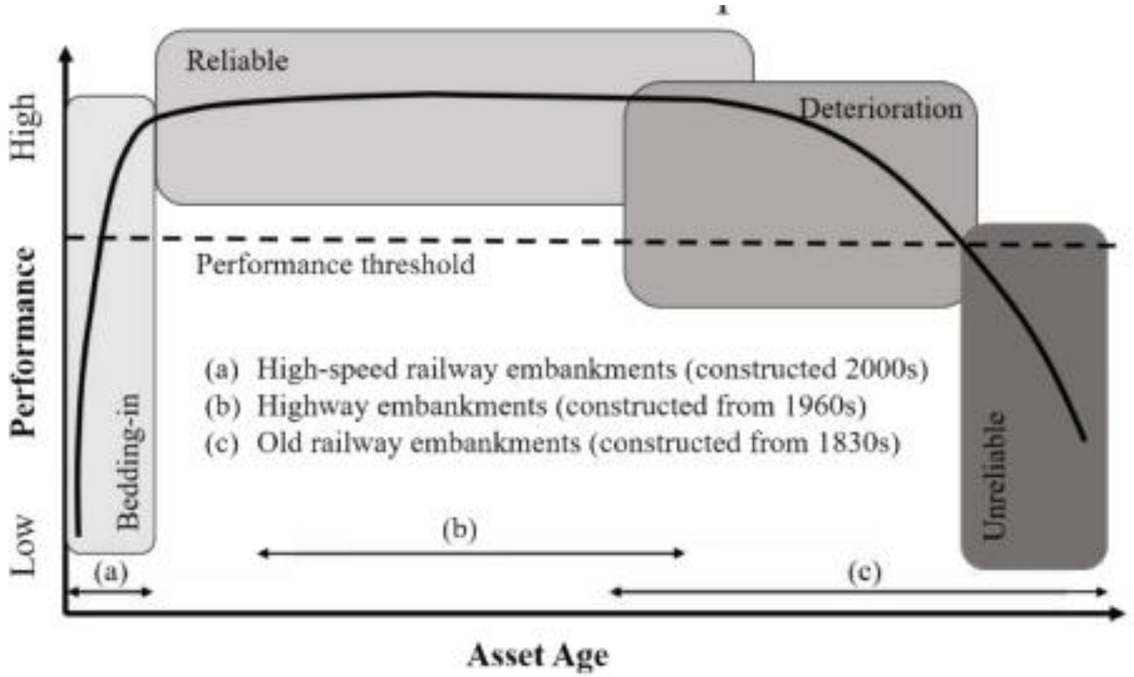
Github Link	Architecture	Data Training	Output
theobdt/aerial_pc_classification: Segmentation of urban aerial point clouds with Deep Learning in Pytorch. (github.com)	Bi-LSTM (Recurrent)	Point Cloud (RGB) Shorter running time Can follow the direction Can input my data	
Yarroudh/segment-lidar: Python package for segmenting LiDAR data using Segment-Anything Model (SAM) from Meta AI. (github.com)	SAM (Attention/Transformer)	Point cloud (RGB) Running time	
HuguesTHOMAS/KPConv: Kernel Point Convolutions (github.com)	KPConv (Convolution)	**S3DIS**: For scene segmentation. The S3DIS (Stanford Large-Scale 3D Indoor Spaces) dataset is a comprehensive dataset used for scene understanding in 3D point clouds. Running time	---very big data to reload---
charlesq34/pointnet2: PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space (github.com)	Pointnet++ (tends to be convolution)		---

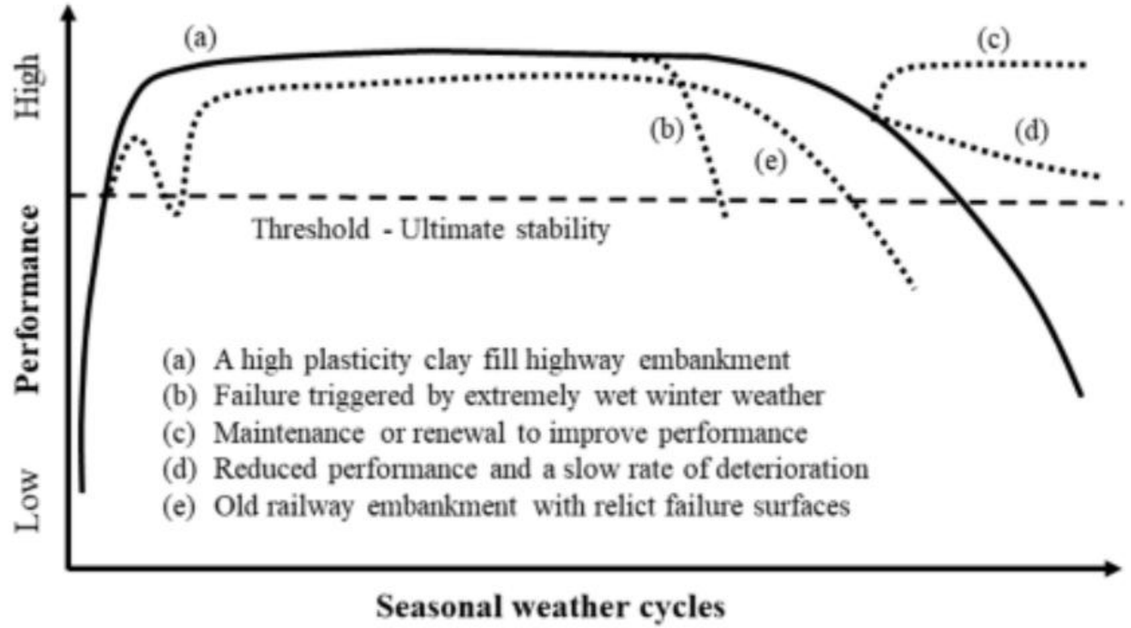
Available Data	Output (so far)	Training Data Available	Method
3D Point Cloud	DEM	DEM (Landslide inventories from Bernard et al. (2021) (zenodo.org))	

Comparison

Architecture	Description	Strength	Weakness
KPConv	KPConv is a convolutional neural network (CNN) designed to process point clouds directly. It uses convolutional kernels to capture local geometric features of point clouds, similar to how standard CNNs operate on images.	**Local Feature Extraction:** KPConv excels in extracting local geometric features due to its convolutional approach, making it highly suitable for tasks requiring fine-grained detail analysis. **Invariance to Point Ordering:** the network is inherently invariant to the order of points, a crucial property for point cloud data. **Scalability:** KPConv can handle large-scale point clouds efficiently	**Complexity:** The convolution operation can be computationally expensive compared to simpler architectures. **Memory Usage:** Higher memory requirements due to convolution operations, especially for large point clouds.
Pointnet++	PointNet++ builds upon the original PointNet architecture, introducing hierarchical feature learning. It applies PointNet recursively on nested partitions of the point set, capturing local and global features.	**Hierarchical Learning:** By learning features at multiple scales, PointNet++ captures both local and global context, making it robust for various point cloud tasks. **Flexibility:** Can be applied to various point cloud tasks including classification, segmentation, and scene understanding. **Efficiency:** generally, more efficient than convolution-based methods like KPConv.	**Sensitivity to Density Variations:** May struggle with varying point densities across the point cloud. **Complexity in Implementation:** More complex to implement due to the hierarchical structure.
Bi-LSTM	Bi-LSTM is a type of recurrent neural network (RNN) that processes data sequences in both forward and backward directions, capturing dependencies in both directions.	**Sequential Data Handling:** Excels at capturing temporal or sequential dependencies, making it suitable for time-series data or sequence-based tasks. **Context Awareness:** Bi-LSTMs can learn long-range dependencies and contextual information. **Simplicity:** Simpler to implement and train compared to some CNN architectures.	**Point Cloud Suitability:** Not inherently designed for point cloud data, which is unordered and non-sequential. **Performance:** generall, underperforms compared to architectures specifically designed for point clouds like KPConv and PointNet++. **Scalability:** May struggle with large-scale point clouds due to the sequential processing nature.

Asset Performance





(a) deterioration and then failure due to strain softening of the clay fill, progressive slope failure and ultimate failure of the embankment

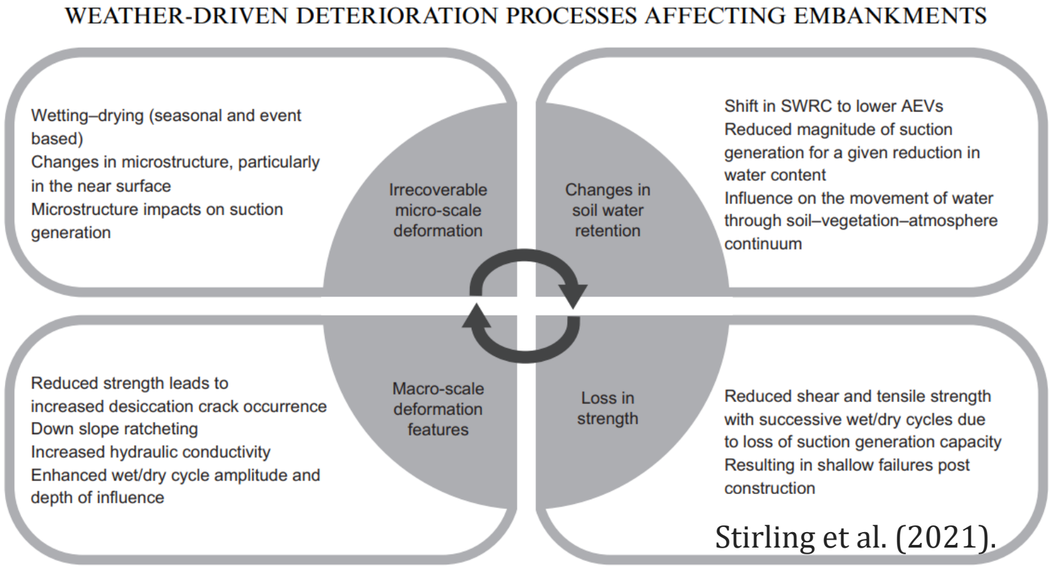
(b) extreme loading due to extremely wet winter weather might trigger a slope failure

(c) Once deterioration has been detected, the performance of the embankment can be improved by maintenance or renewal activities within an appropriate timeframe

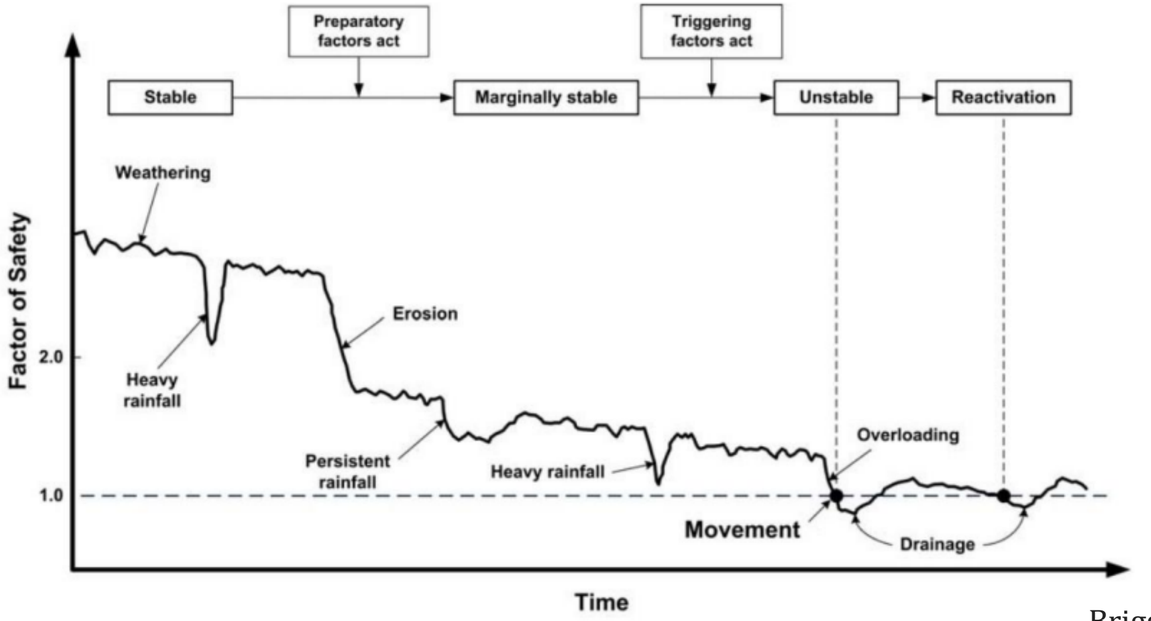
(d) Alternatively, monitoring and inspection might show that an embankment is deteriorating at a slower rate than projected, so that interventions can be delayed or reduced

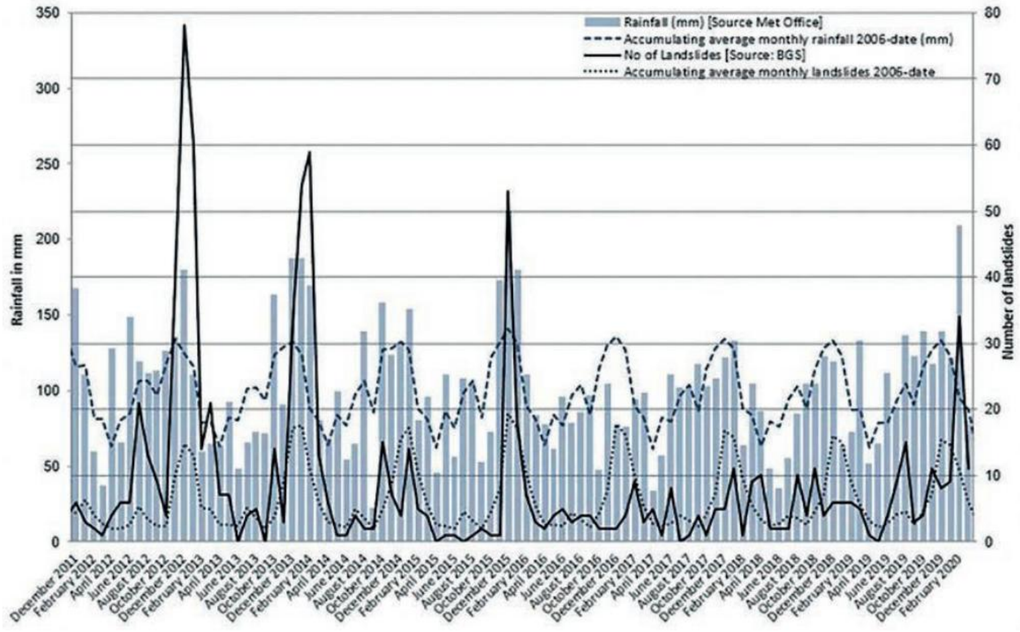
Briggs et al. (2019)

Rainfall Impacting to Slope Condition



Stirling et al. (2021).





Briggs et al. (2019)

Note

1. High temperatures leading to increased rates of evapotranspiration
2. Increased evapotranspiration draws water out of the slope leading to lower saturation at the near surface
3. Lower saturation leads to reduced relative hydraulic conductivity which reduces water movement within the slope and leads to the development of larger suctions but reduces the ability of surface recharge due to rainfall to allow their dissipation
4. Increased rainfall rate and a lower relative hydraulic conductivity leads to increased run-off and reduced infiltration

Rouainia et al. (2020).

Possible Dataset : Regional Climate Model (RCM)

Background Data	Model Data Source	Resolution
Global Climate Model (GCM) Reanalysis Data Surface Data (topography, Land use & Land Cover, Soil Data) Climate data (rainfall, temperature, etc) Oceanographic Data (Sea surface temperature, sea current, etc) Aerosol and GHGs Emission Scenario	CORDEX (Coordinated Regional Climate Downscaling Experiment)	0.11° x 0.11° (approximately 12.5 km x 12.5 km)
	RCMES (Regional Climate Model Evaluation System)	0.1° to 0.5° (approximately 10 km to 50 km)
	Meteoblue	3 km for high-resolution local models. For broader regional models, the resolution might range from 10 km to 25 km
	ECMWF (European Centre for Medium-Range Weather Forecasts)	0.11° (approximately 12.5 km) for Europe

Type of Data	Potential Data Availability
Temperature	Surface temperature
	Extreme heat
Precipitation	Total precipitation
	Extreme precipitation
Wind	Wind speed
	Wind direction
Humidity and Evapotranspiration	Relative humidity
	Evapotranspiration
Radiation	Solar radiation
	Longwave radiation
Soil and Surface Data	Soil moisture (water content)
	Surface fluxes (heat, moisture)
Sea Surface	Sea surface temperature
	Sea Ice Extent
Aerosol and air Quality	Trace gases (ozone, co2), Aerosol Concentration

Challenge

Challenge		
Dataset	Noise	Sequences of (multisensor) satellite observations have diverse noise sources, uncertainty levels, missing data, and (often systematic) gaps (e.g., acquisition, storage, and transmission distortions). Aerial and satellite images have a varied
	Heterogeneity	Climate and ecosystem processes reveal a high level of heterogeneity due to differences in geography, topography, and climatic conditions in diverse areas of the earth.
	Deluge of dataset	airborne LiDAR surveys, SAR satellites, stereophotogrammetry, and mobile mapping systems are increasingly used and produce data volumes that raise computational challenges for spectral, spatial, and temporal dimensionalities
Class Imbalance	Data-level technique	Random oversampling can improve the classification of imbalanced image data. Random undersampling can decrease the amount of class imbalance for pretraining a deep CNN.
	Algorithms-level techniques	Cost-sensitive deep learning methods are emerging, which learn network weight parameters and class misclassification costs during training and thus give higher importance to samples with a higher cost.
Data Interpretation		Well-trained neural networks still have the typical issue of the lack of interpretability. Given their complexity, landslides prevention models in the earth system are often not easily traceable back to their assumptions, limiting their interpretability.
Machine Learning		Regardless of the model, it is necessary to select the proper parameters and thresholds of each feature.

Ma, Z., Mei, G., & Piccialli, F. (2021). Machine learning for landslides prevention: a survey. *Neural Computing and Applications*, *33*(17), 10881-10907.

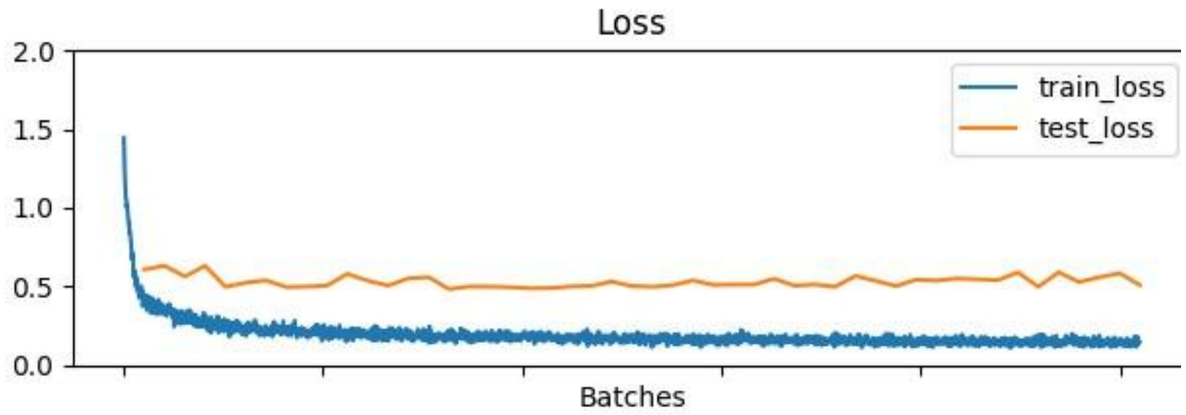
Training Model

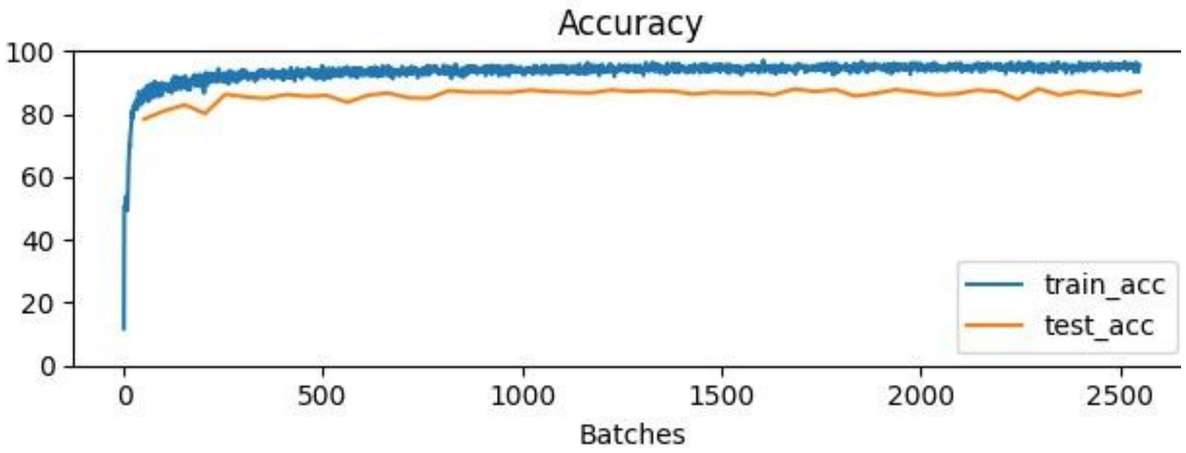
Aerial Classification using BI-LSTM

- Training Dataset : Points are labelled in 9 classes that are composed of : powerline, low vegetation, impervious surfaces, cars, fence/hedge, roof, facade, shrub and tree. The point density of the dataset is ~5 pts/sqm. The training set contains 753 876 points and the test set contains 411 722 points.

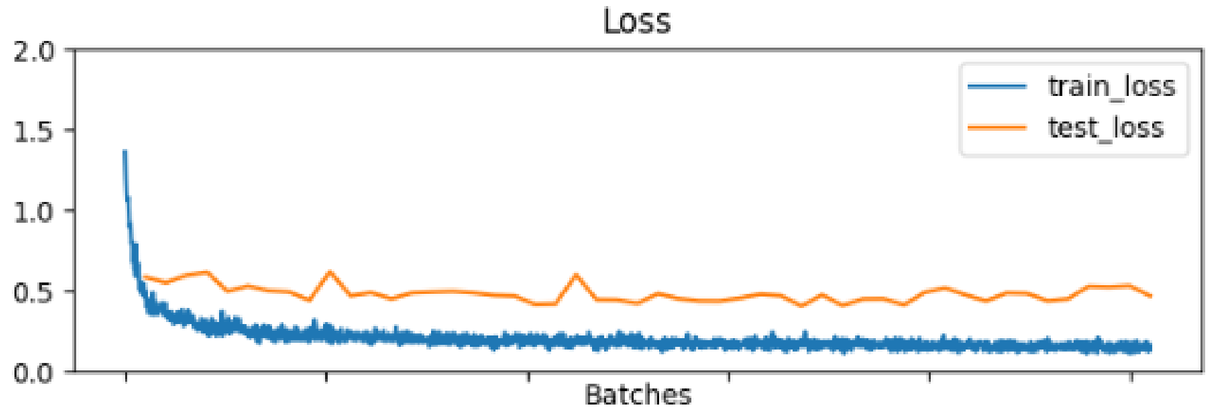
Result of Training and Testing Data 4 Classes

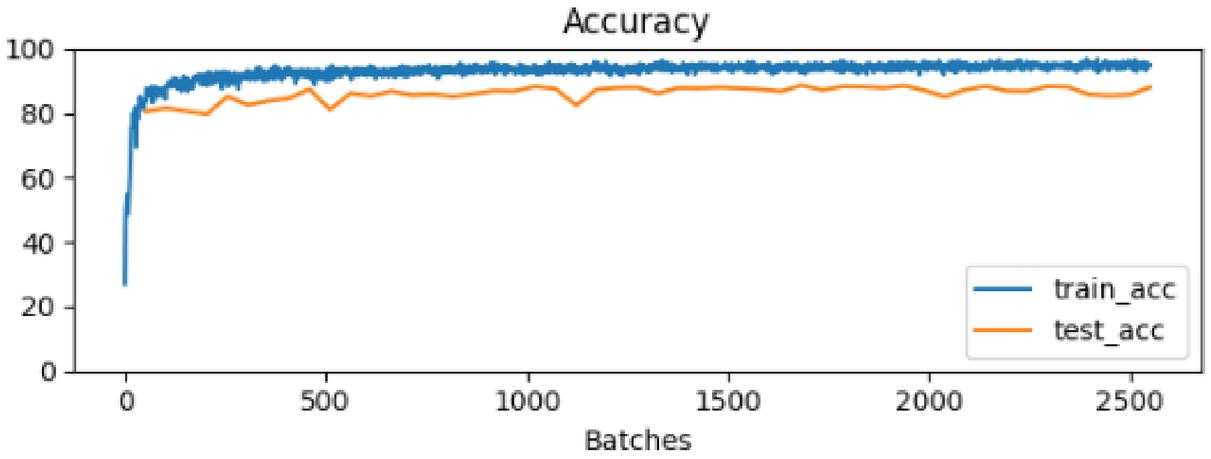
My result:





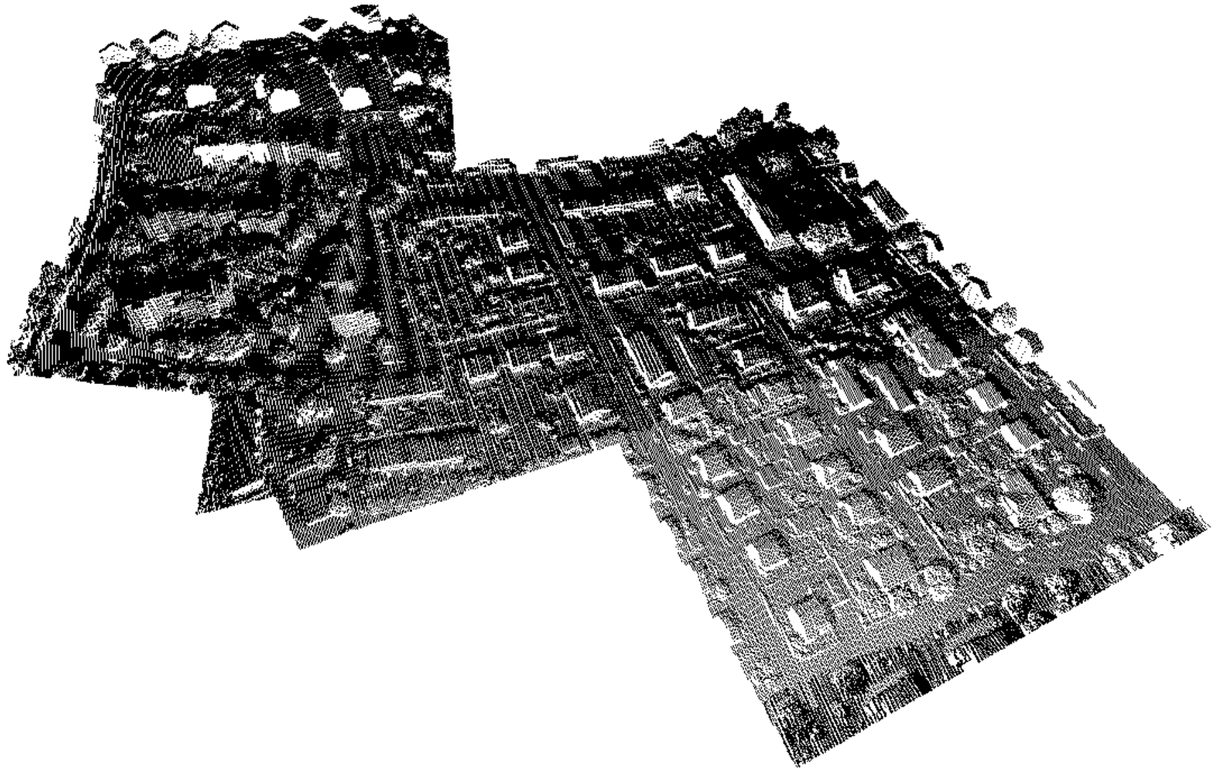
Paper:



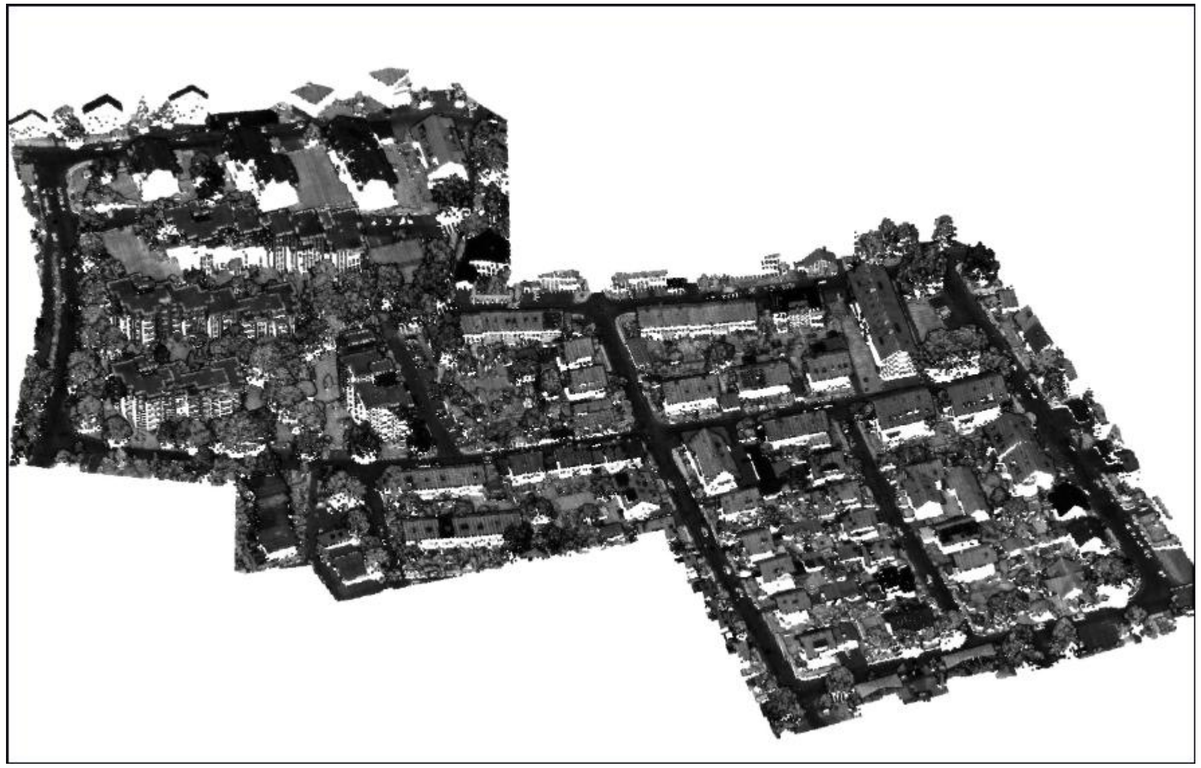


Result of Preprocessing

My result:

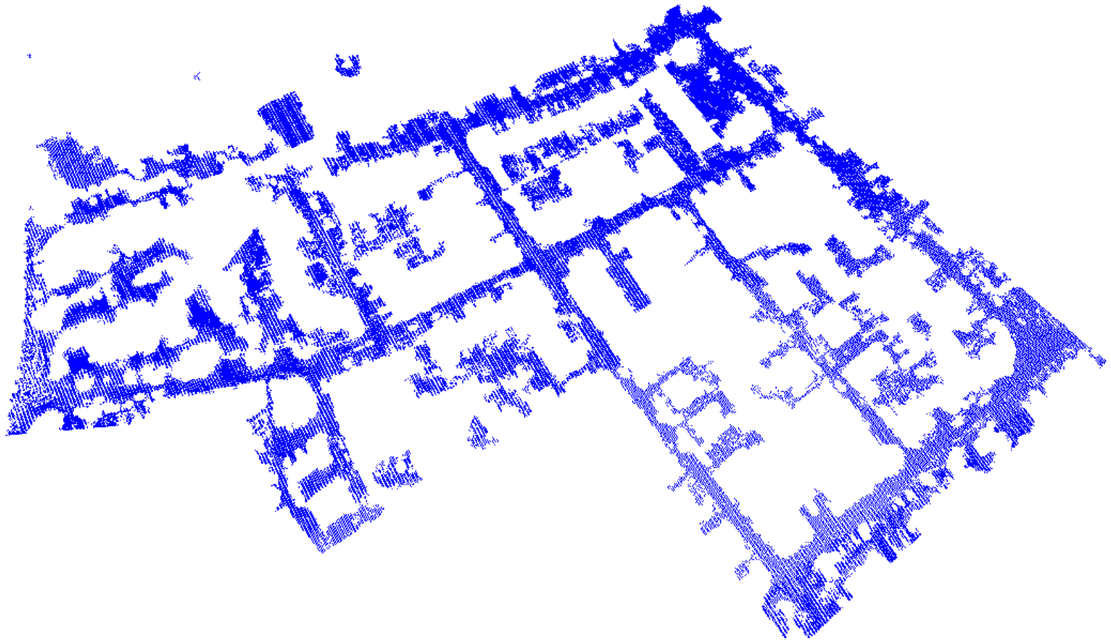


Paper:

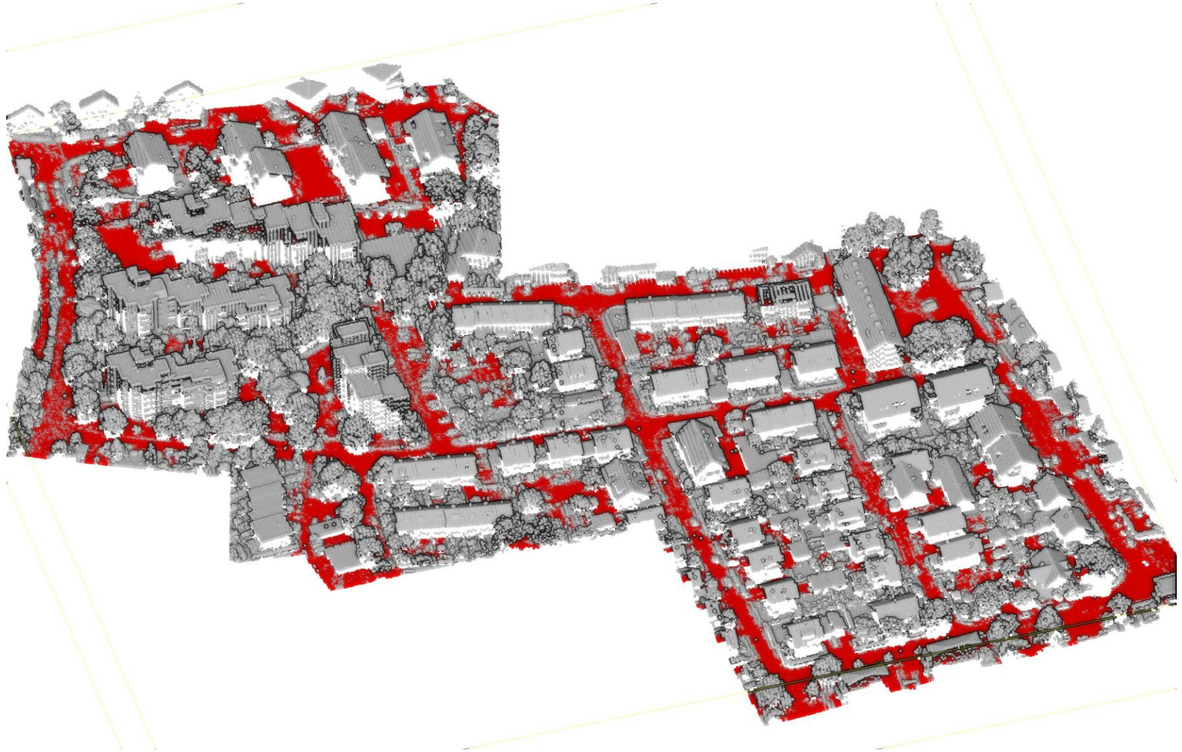


Result of Ground Extraction

My result:

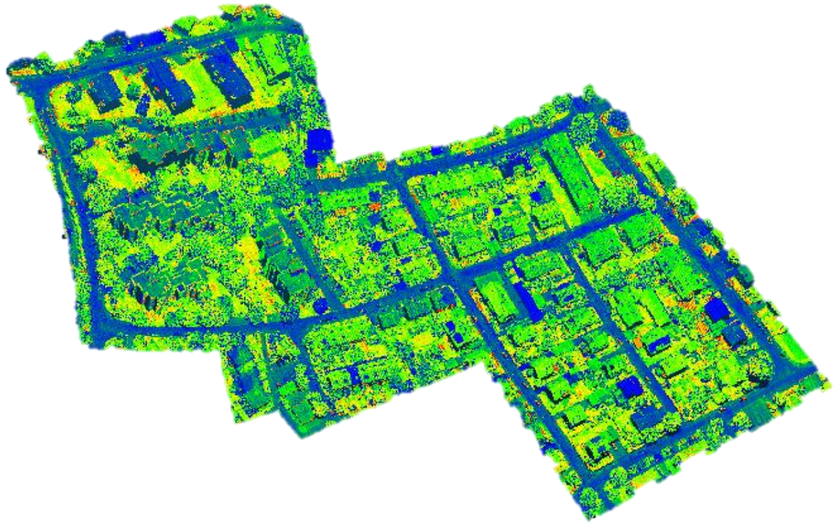


Paper:

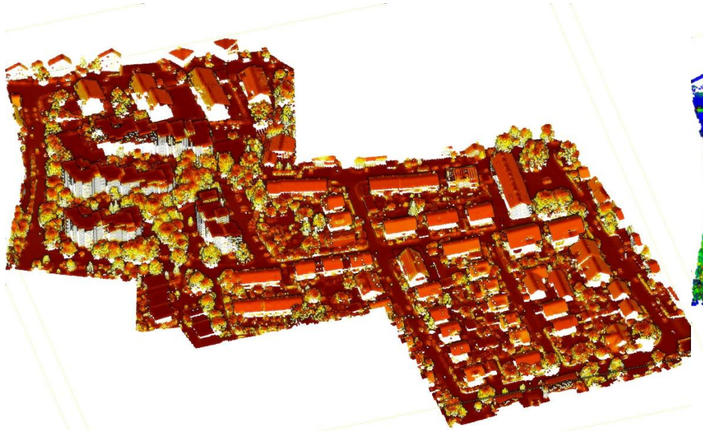


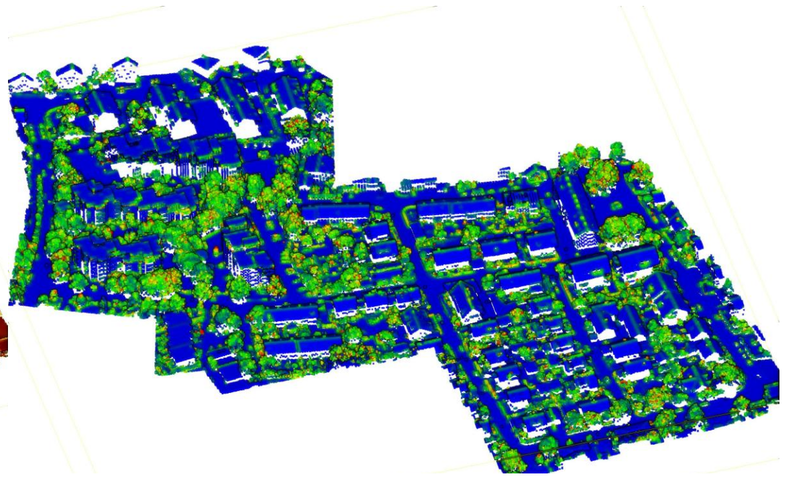
Result of Features

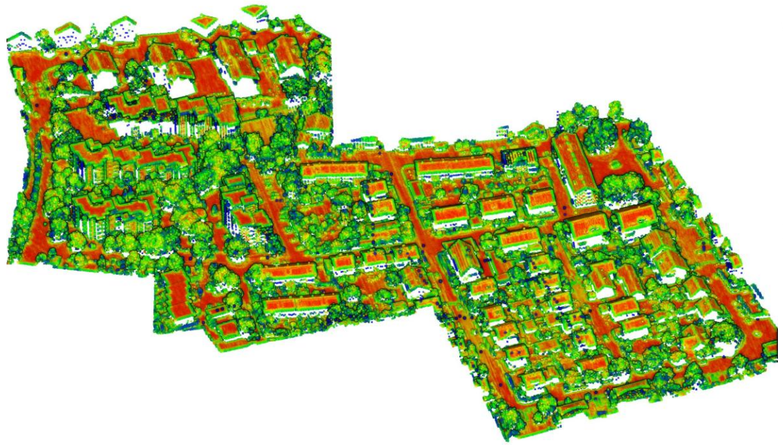
My result:

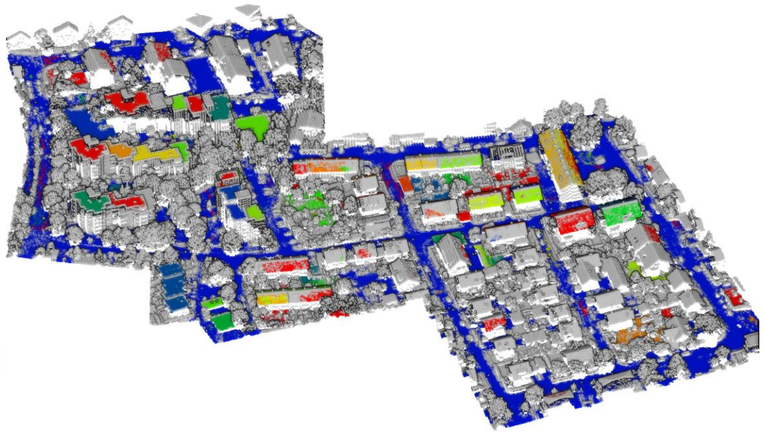


Paper:



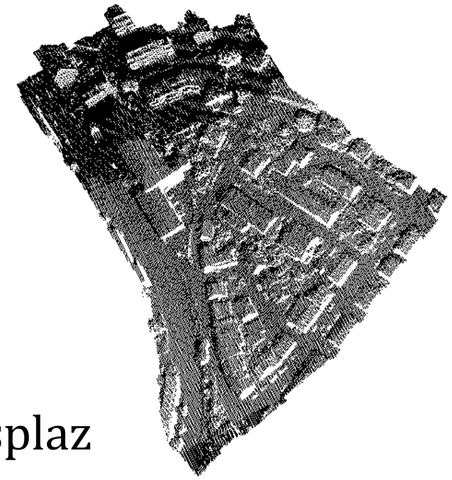


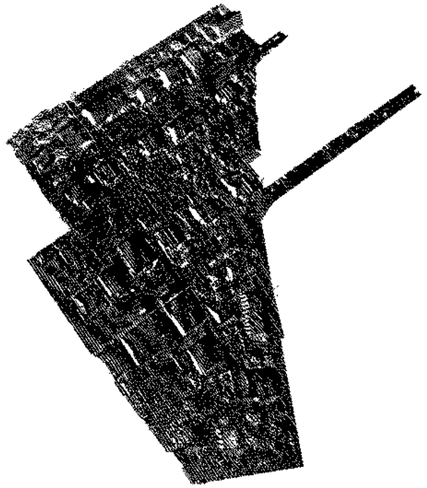




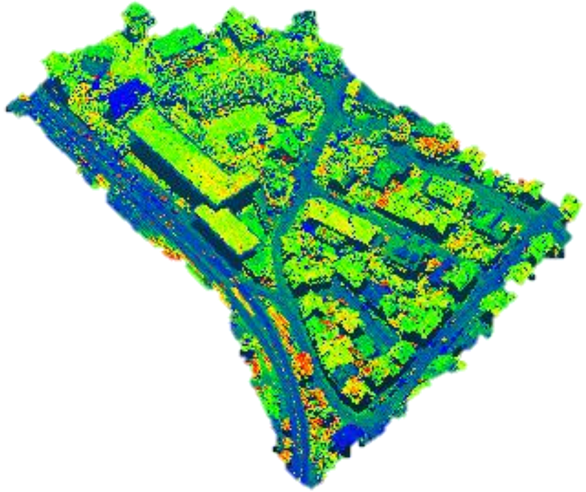
Result of Bi-LSTM Prediction

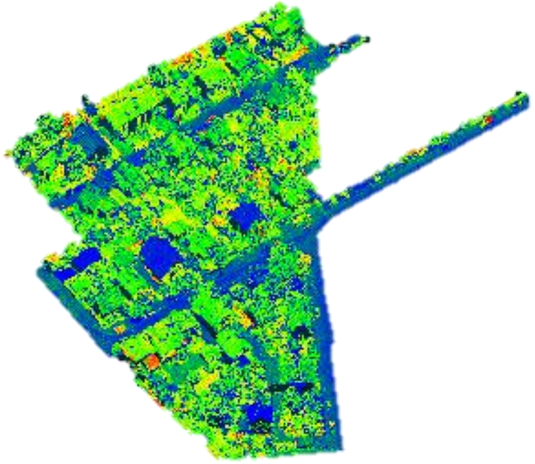
My result:





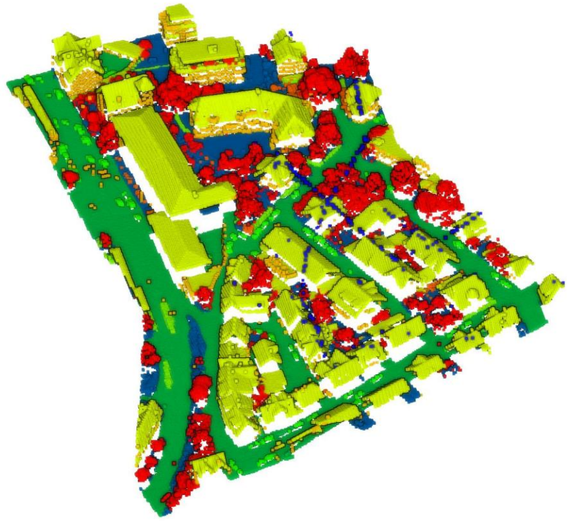
Displaz

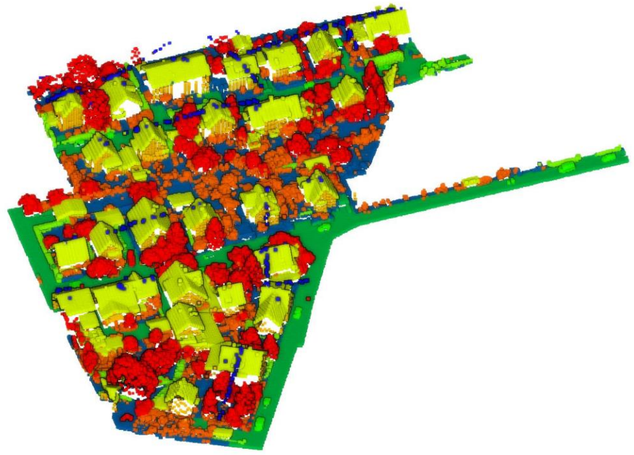




CloudCompare

Paper:





Result of Training and Testing Data 4 Classes

My result:

	GLO	Roof	Facade	Vegetation	Total	Precision	Recall	F1-score
GLO	179535	7439	118	13584	200676	95.922358	89.465108	92.581276
Roof	2032	89342	2588	15086	109048	88.101530	81.929059	84.903258
Facade	680	1058	4045	5441	11224	50.142556	36.038845	41.936654
Vegetation	4920	3569	1316	69239	79044	66.994678	87.595516	75.922454
Total/Avg	187167	101408	8067	103350	399992	75.290281	73.757132	73.835911
Total/Weighted Avg	187167	101408	8067	103350	399992	86.789081	85.541961	85.774921

Paper:

	GLO	Roof	Facade	Vegetation	Total	**Precision**	**Recall**	**F1-score**
GLO	179498	9057	245	11876	200676	96.27	89.45	92.73
Roof	1046	94777	2110	11115	109048	85.43	86.91	86.17
Facade	657	1669	4286	4612	11224	51.84	38.19	43.98
Vegetation	5248	5436	1626	66734	79044	70.74	84.43	76.98
Total/Avg (Weighted)	186449	110939	8267	94337	399992	**76.07 (87.02)**	**74.74 (86.33)**	**74.96 (86.46)**

Aerial Classification.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

Files

cfg
ckpts
2024-06-20_10-39-21
ckpt.pt
config.yaml
figure.png
data
features
ground_only
ground_rasterized
predictions
2024-06-20_10-39-21
metrics
vaihingen3D_test.ply
preprocessed
vaihingen3D_test.pts
vaihingen3D_train.pts

+ Code + Text

RAM Disk

[28]
5h

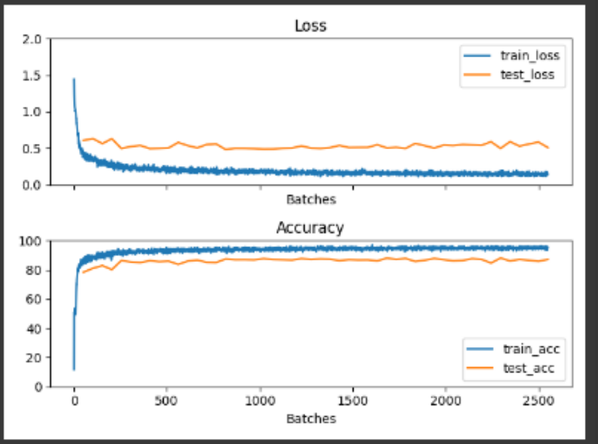
```
Epoch [49/50], Step [10/50], Loss: 0.1198, Acc: 95.02 %
Epoch [49/50], Step [20/50], Loss: 0.1450, Acc: 95.19 %
Epoch [49/50], Step [30/50], Loss: 0.1127, Acc: 95.21 %
Epoch [49/50], Step [40/50], Loss: 0.1472, Acc: 95.06 %
Epoch [49/50], Step [50/50], Loss: 0.1510, Acc: 95.40 %
Test Loss : 0.5839, Test Acc : 85.87 %

Epoch [50/50], Step [10/50], Loss: 0.1550, Acc: 95.20 %
Epoch [50/50], Step [20/50], Loss: 0.1713, Acc: 95.38 %
Epoch [50/50], Step [30/50], Loss: 0.1200, Acc: 95.63 %
Epoch [50/50], Step [40/50], Loss: 0.1225, Acc: 94.80 %
Epoch [50/50], Step [50/50], Loss: 0.1450, Acc: 95.08 %
Test Loss : 0.5087, Test Acc : 87.19 %

Figure saved to ckpts/2024-06-20_10-39-21/figure.png
Figure(640x480)
```

2m

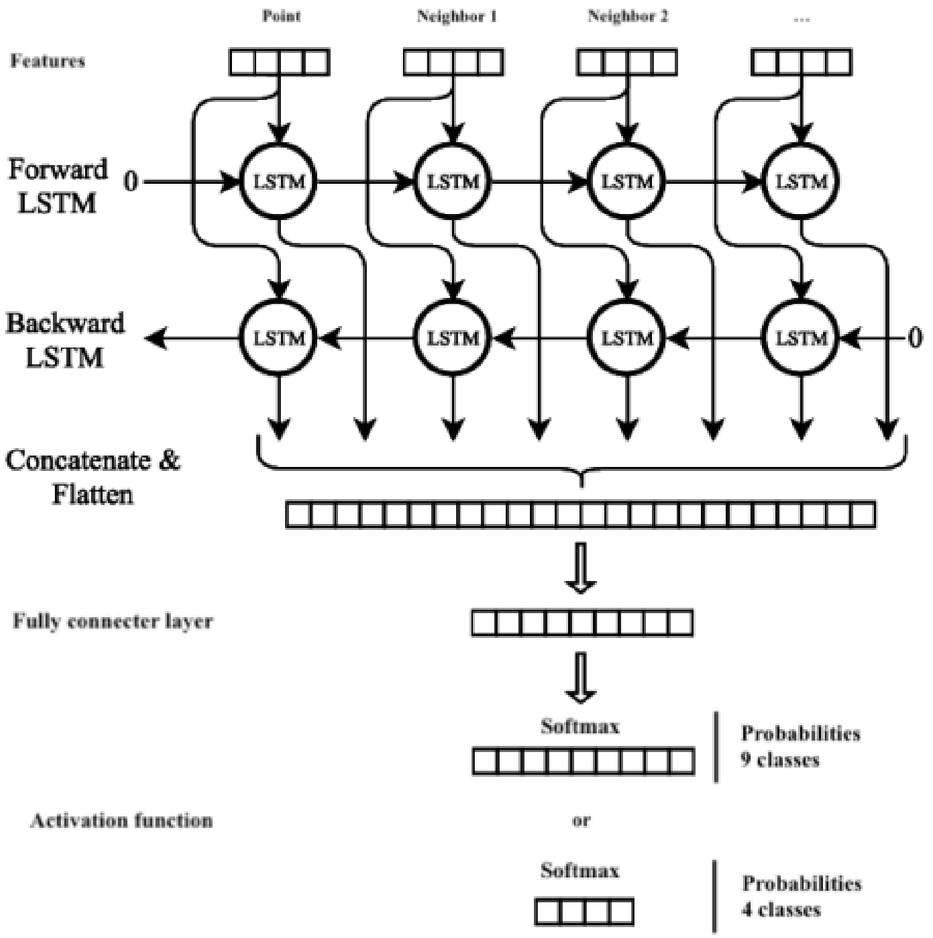
```
python3 test.py -f data/features/vaihingen3D_test.ply --ckpt ckpts/2024
```

```
Device in use : cpu
Loading checkpoint: ckpts/2024-06-20_10-39-21
Num classes: 4

Loading model.. DONE

Processing file: data/features/vaihingen3D_test.ply
* Preparing dataloader.. /usr/local/lib/python3.10/dist-packages/torch/
  warnings.warn(_create_warning_msg(
DONE
* Processing point cloud: 100% 400/400 [14:12<00:00,  2.13s/it]
* Predictions PLY file saved to: data/predictions/2024-06-20_10-39-21/
data/predictions/2024-06-20_10-39-21/metrics/vaihingen3D_test.tex
* Metrics written to: data/predictions/2024-06-20_10-39-21/metrics/vai
```

compute_features.py config_bilstm.yaml fig



Bi-LSTM Architecture



- **200 hidden units**
- **Features**:
  The input consists of a sequence of features for a point and its neighbors (Neighbor 1, Neighbor 2, etc.).
- **Bidirectional LSTM**:
  Forward LSTM: Processes the sequence of features in the forward direction (left to right). **Backward LSTM**: Processes the sequence of features in the backward direction (right to left). Each LSTM unit captures dependencies and patterns in the input sequence. The forward and backward LSTMs allow the network to capture information from both past and future contexts.
- **Fully Connected Layer**:
  The concatenated and flattened vector is passed through a fully connected layer (also known as a dense layer). This layer performs a linear transformation of the input features, followed by an optional non-linear activation function.
- **Concatenate & Flatten**:
  The outputs of the forward and backward LSTMs are concatenated and flattened into a single vector. This vector combines information from both directions for each point in the sequence.
- **Softmax**
  - **Range:** The outputs of the softmax function are in the range (0, 1), making them suitable for representing probabilities.
  - **Sum to One:** The sum of all the outputs of the softmax function is 1, ensuring they form a valid probability distribution.
  - **Differentiability:** The softmax function is differentiable, which is important for gradient-based optimization methods used in training neural networks.

Local Descriptor

- **Eigenvectors** are vectors that, when transformed by a matrix, change only in magnitude, not direction.
- **Eigenvalues** are the scalars that represent the factor by which the eigenvectors are scaled during the transformation.
- Let's have λ1, λ2 and λ3 the eigenvalues resulting from the PCA in the local neighborhood of radius r, such as λ1 ≥ λ2 ≥ λ3. The following local descriptors can be derived :

  - Normals : eigenvector associated with the eigenvalue λ3.

  - Planarity : how planar the surrounding of each point is. Allows separation of non- planar regions (ex : vegetation) from planar regions (ex : roofs).

$$planarity = \frac{\lambda_2 - \lambda_3}{\lambda_1}$$

  - Verticality : with nz being the z-coordinate of the normal at a particular point

$$verticality = \frac{2arcsin(n_z)}{\pi}$$

  - Linearity

$$linearity = \frac{\lambda_1 - \lambda_2}{\lambda_1}$$

  - Sphericity

$$sphericity = \frac{\lambda_3}{\lambda_1}$$

  - Anisotropy

$$anisotropy = \frac{\lambda_1 - \lambda_3}{\lambda_1}$$

  - Surface variation

$$surface\_variation = \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3}$$

Height above Ground

Note:
1. Some features are trickier to extract like height above ground
2. An approximation of ground level is needed.

Region Growing

1. to separate point clouds into segments (points belonging to the same surfaces are grouped together).
2. In order to grow multiple regions, we iterate the procedure several times, removing the grown regions from the original point cloud P.

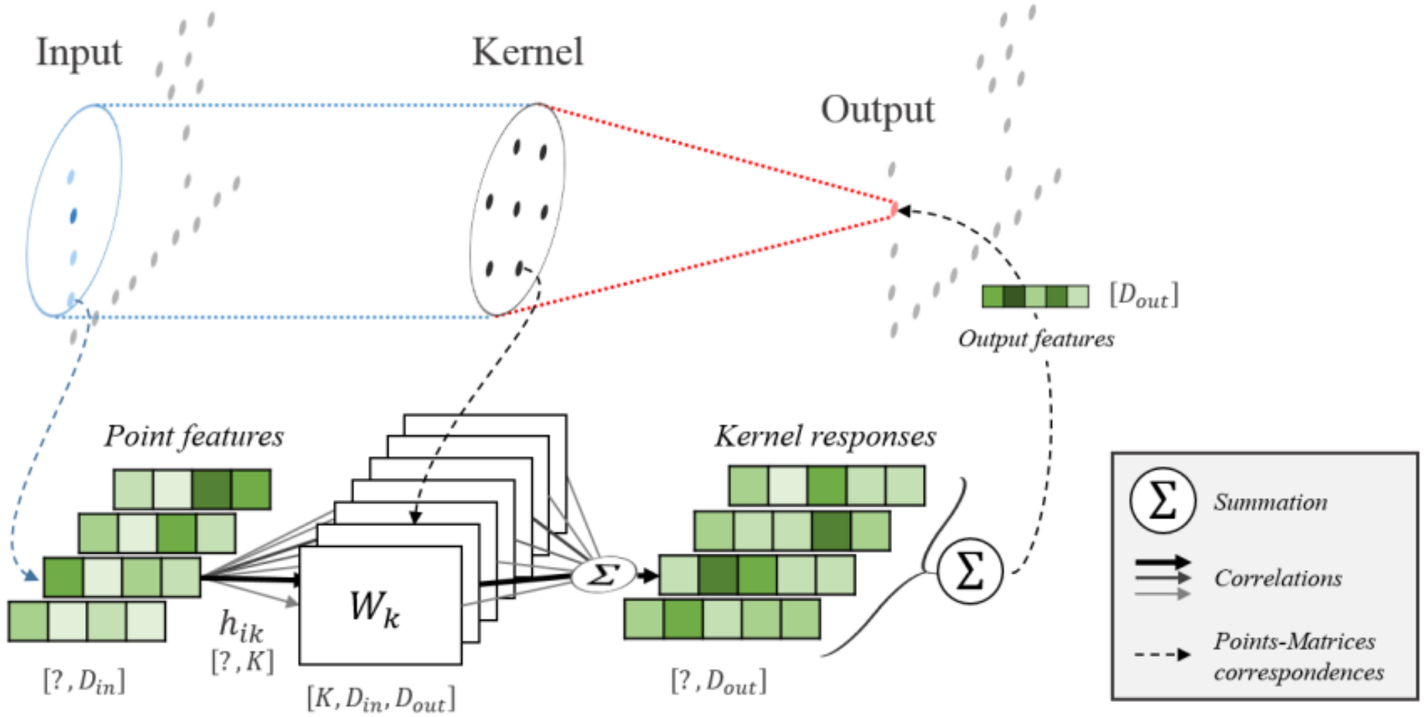
Ground Extraction

1. This procedure allows to filter out facades, fences (slope intra-regional) and roofs (slope inter-regional).The ground itself is not globally at the same level and might have been sliced into several parts by the region growing procedure
2. To reconstruct the ground as a whole segment, we need to stitch some of them together to create a ground mask
3. evaluate the slope intra-regional of each region defined as the ratio of the maximal difference of height with the span of the region
4. compute the slope inter-regional, defined as the slope between the 1% of points in the candidate region closest to the current ground, and the current ground
5. If both of these slopes are smaller than predefined thresholds, we add the region to the current ground

Ground Rasterization

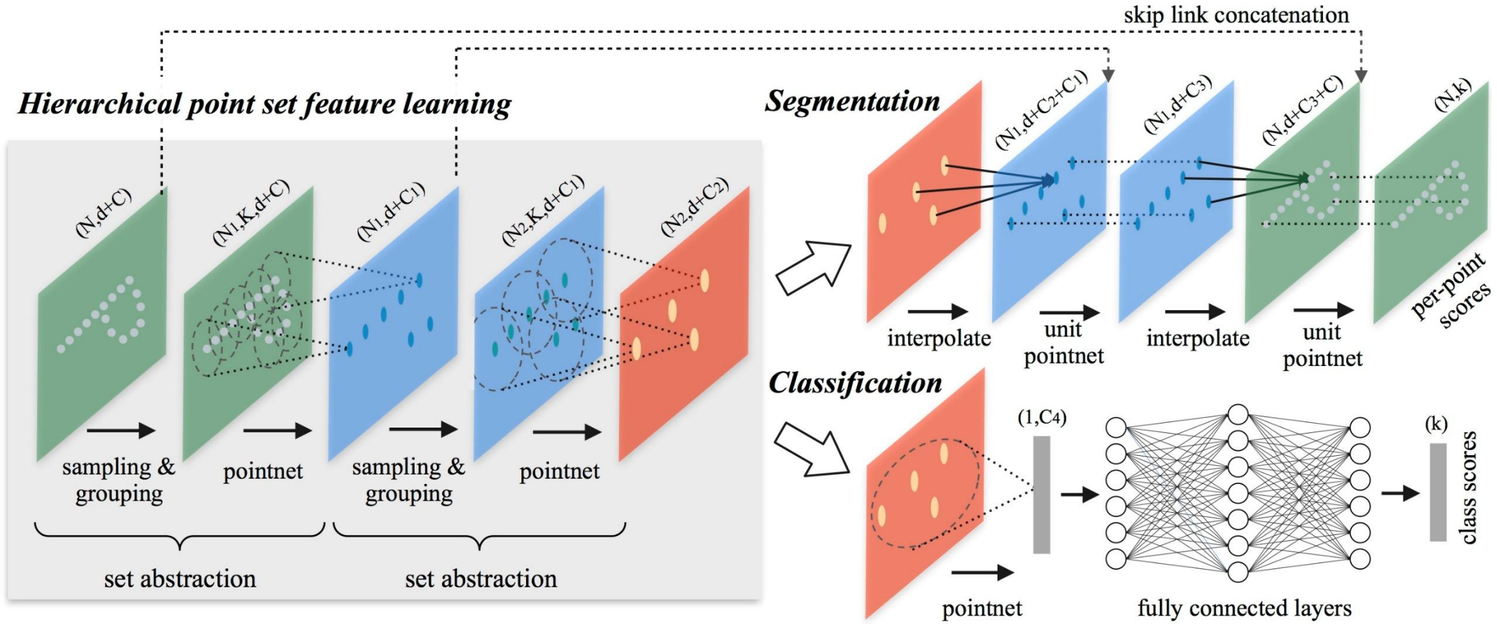
1. to interpolate the altitude of the ground on a grid, from the ground set previously computed
2. Closest neighbor : each point of the grid has the same altitude of its closest neighbor in the ground set.
3. Delaunay triangulation : the space is divided into triangles, which vertices are points of the ground set. The altitude of grid points is then interpolated linearly from the vertices of the triangles they fall into.

KPConv



1. **Point Features Extraction**: Each point in the input point cloud is associated with a feature vector of dimension.
2. **Kernel Function Application**: The kernel functions are applied to the point features, considering the relative positions of points within the defined kernel range. These functions compute the influence of neighbouring points, effectively capturing local geometric relationships.
3. **Weighting and Summation**: The kernel responses are multiplied by the weight matrices to produce intermediate features. These intermediate features are then summed up to generate the final output features for each point.
4. **Output Features Generation**: The weighted and summed responses form the output feature vector for each point in the point cloud, with a new dimensionality

Pointnet++



Hierarchical Point Set Feature Learning

1.Set Abstraction Layers:

1. **Sampling & Grouping**: The point cloud data is sampled and grouped into local regions.
2. **PointNet**: Local features are extracted using PointNet applied to these groups.
3. This process is repeated to build hierarchical features, moving from fine to coarse resolutions.

Segmentation Pathway

2.Feature Interpolation: The hierarchical features are interpolated back to the original resolution.

3.Unit PointNet: Further feature extraction is done using PointNet units.

4.Skip Link Concatenation: Features from different levels are concatenated to preserve both local and global information.

5.Per-Point Scores: The final output provides per-point segmentation scores.

Classification Pathway

6.Fully Connected Layers: The extracted features are passed through fully connected layers.

7.Class Scores: The final output provides class scores for classification.

Key Points

•**Set Abstraction**: Layers progressively abstract the point set into hierarchical features.

•**Segmentation and Classification**: The architecture supports both segmentation (per-point classification) and global classification tasks.

•**Skip Connections**: These help in retaining detailed information from different abstraction levels, improving the model's accuracy in segmentation tasks.

Bi-LSTM Architecture

Task	Function
Preprocessing	•**Reading Point Cloud Data**: The read_pts function is used to read the point cloud data from the provided file paths. •**Centering and Scaling**: If centering is enabled, the point cloud coordinates are centered around the origin. The coordinates are then scaled by a specified factor. The main function is to normalise and standarise the data •**Extracting Features and Labels**: The features and labels from the point cloud data are extracted and converted to appropriate data types.
Compute Feature	**Argument Parsing**: The script starts by parsing command-line arguments to specify the input files, configuration file, and the steps to be performed. **Configuration Loading**: It reads a configuration file (YAML format) which contains parameters for different feature extraction steps. **Feature Extraction Pipeline**: The main function compute_features processes each input PLY file through a series of specified steps, which can include: **Descriptors Computation**: Calculates local descriptors for the point cloud data. **Region Growing**: Segments the point cloud into regions based on certain descriptors and parameters. **Ground Extraction**: Identifies and extracts ground points from the point cloud. **Ground Rasterization**: Converts the extracted ground points into a raster grid. **Height Above Ground Calculation**: Computes the height of points above the ground level. **Output Saving**: The processed data, including computed features and intermediate results, is saved in PLY format to specified directories. **Pipeline Control**: The script can run all steps in sequence (--full_pipeline), start from a specific step (--from_step), or execute a custom list of steps (--steps).

Bi-LSTM Architecture

config_features_extraction.yaml	**Descriptors**: **Local Descriptors**: A set of descriptors (normals, verticality, linearity, planarity, sphericity, curvature) are computed for each point in the point cloud. **Radius**: The neighborhood radius used for computing these descriptors. **Preferred Orientation**: The preferred orientation for normals calculation. **Epsilon**: A small value added to the denominator to avoid division by zero in some descriptor calculations. **Region Growing**: **Regions**: The point cloud is segmented into a specified number of regions based on certain criteria. **Radius**: Radius used to select candidate points for region growth. **Descriptor**: The descriptor (e.g., planarity) used to guide the region growing process. **Thresholds**: Limits for height difference, normal angle difference, and descriptor value difference between points. **Ground Extraction**: **Slope Constraints**: Parameters defining the maximum allowable slopes for ground regions, both internally within a region and between regions. **Percentile Closest**: The percentile of closest points in a candidate region used to compute inter-region slope. **Ground Rasterization**: **Method**: The method used for rasterizing ground points into a grid (either "closest_neighbor" or "delaunay"). **Step Size**: The size of the grid steps used in the rasterization process. **Height Above Ground**: This section likely contains parameters (not shown here) for calculating the height of points above the rasterized ground surface.

Bi-LSTM Architecture

config_bilstm.yaml	**Network**: **Hidden Size**: Specifies the number of units in the hidden layers of the neural network. **Number of Layers**: Defines the number of hidden layers in the network. **Relation Type**: Indicates the type of relations considered in the model (details depend on specific implementation). **Training**: **Batch Size**: Number of samples per batch during training. **Max Batches**: Maximum number of batches to process per epoch. **Epoch End**: Number of epochs for training. **Num Workers**: Number of worker threads used for data loading. **Shuffle**: Whether to shuffle the data at the beginning of each epoch. **Test**: **Batch Size**: Number of samples per batch during testing. **Max Batches**: Maximum number of batches to process per epoch during testing. **Num Workers**: Number of worker threads used for data loading during testing. **Shuffle**: Whether to shuffle the data before testing. **Optimizer**: **Learning Rate**: Learning rate for the optimizer used in training the model. **Data**: **Features**: List of features used as input to the model. **Number of Neighbors**: Number of neighbors to consider for each point in the dataset. **All Labels**: Whether to use all labels available in the dataset (boolean).

Bi-LSTM Architecture

train	**Argument Parsing**: It parses command-line arguments for configurations, debug mode, logging interval, checkpoint paths, and data paths. **Device Configuration**: Determines whether to use a GPU or CPU for training. **Checkpoint Initialization**: Initializes or loads a checkpoint to save and resume training progress. **Configuration Loading**: Loads training configurations from a specified YAML file. **Data Preparation**: Loads the training and test datasets using a custom dataset class (AerialPointDataset) and creates data loaders for batching and shuffling. **Model Setup**: Defines the BiLSTM model, loss function (CrossEntropyLoss), and optimizer (Adam). **Training and Evaluation Functions**: **train**: Trains the model for one epoch, logging loss and accuracy at specified intervals. **evaluate**: Evaluates the model on the test dataset, calculating average loss and accuracy. **Training Loop**: Runs the training loop for the specified number of epochs, calling the train and evaluate functions, and updating the checkpoint with the best model state. **Metrics Plotting**: After training, it plots and saves the training and test loss/accuracy over batches.

Bi-LSTM Architecture

test	**Argument Parsing**: Parses command-line arguments to specify input files, checkpoint path, batch size, number of workers for data loading, and prediction folder path. **Device Configuration**: Determines whether to use a GPU or CPU for predictions. **Checkpoint Loading**: Loads a pre-trained model checkpoint and the corresponding configuration file. **Model Setup**: Initializes the BiLSTM model using the loaded configuration and loads the model's state from the checkpoint. **Prediction Function**: Defines a function to perform predictions on the dataset: Iterates over the data loader to compute predictions. Stores the predicted classes for the entire dataset. **Evaluation Function**: Defines a function to evaluate the predictions: Computes a confusion matrix and various metrics (precision, recall, F1-score). Aggregates metrics for overall evaluation. **Metrics Writing Function**: Saves the evaluation metrics to both LaTeX and text files for reporting purposes. **Processing Each Input File**: Loads the point cloud data and prepares a data loader. Performs predictions using the model. Converts and evaluates the true labels and predictions. Saves the predictions and errors back into a PLY file. Writes the evaluation metrics to files.

Reference

- Rouainia, M., Helm, P., Davies, O., & Glendinning, S. (2020). Deterioration of an infrastructure cutting subjected to climate change. *Acta Geotechnica*, *15*(10), 2997-3016.
- Briggs, K. M., Dijkstra, T. A., & Glendinning, S. (2019). Evaluating the deterioration of geotechnical infrastructure assets using performance curves. In *International Conference on Smart Infrastructure and Construction 2019 (ICSIC) Driving data-informed decision-making* (pp. 429-435). ICE Publishing.
- Stirling, R. A., Toll, D. G., Glendinning, S., Helm, P. R., Yildiz, A., Hughes, P. N., & Asquith, J. D. (2021). Weather-driven deterioration processes affecting the performance of embankment slopes. *Géotechnique*, *71*(11), 957-969.
- Thomas, H., Qi, C. R., Deschaud, J. E., Marcotegui, B., Goulette, F., & Guibas, L. J. (2019). Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 6411-6420).
- Özdemir, E., Remondino, F., & Golkar, A. (2019). Aerial point cloud classification with deep learning and machine learning algorithms. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *42*, 843-849.