ENVIRONMENTAL SCIENCES AND ENGINEERING

SIE Master Project

Novelty Detection in Convolutional Neural Networks Using Density Forests

Author:

Cyril WENDL

Supervisors:

Prof. Devis TUIA (WUR)
Diego MARCOS (WUR)
Prof. François GOLAY (EPFL)
Spring semester 2018

Lausanne, August 22, 2018

Novelty Detection in Convolutional Neural Networks Using Density Forests

Cyril Wendl

August 22, 2018

Abstract

Uncertainty in deep learning has recently received a lot of attention in research. While state-of-the-art neural networks have managed to break many benchmarks in terms of accuracy, it has been shown that by applying minor perturbations to the input data, they are susceptible to fooling, yielding unreasonably high confidence scores while being wrong. While some research has gone into the design of new architectures that are probabilistic in nature, such as Bayesian Neural Networks, other researchers have tried to model uncertainty of standard architectures heuristically. This work presents a novel method to assess uncertainty in Convolutional Neural Networks, based on fitting a forests of randomized Decision Trees to the network activations before the final classification layer. Experimental results are provided for patch classification on the MNIST dataset and for semantic segmentation on satellite imagery used for land cover classification. The land cover dataset consists of overhead imagery of the city of Zurich in Switzerland taken in 2002, with corresponding manually annotated ground truth. The Density Forest confidence estimation method is compared to a number of baselines based on softmax activations and pre-softmax activations. All methods are evaluated with respect to novelty detection. The study shows that using pre-softmax activations of the Fully Connected layer provides a better overall confidence estimate than just using the softmax activations. For the MNIST dataset, softmax measures outperform pre-softmax based novelty detection measures, while in the Zurich dataset, pre-softmax based methods not only show better performance in detecting the left-out class, but they also manage to identify particular objects for which no class exists in the ground truth. Among the main explanations for the varying performance of pre-softmax measures, we find the curse of dimensionality when working with high-dimensional activation vectors and class separability issues due to partially trained networks. Future research should go into studying the influence of the activation vector dimensionality on novelty detection methods, applying them to more diverse datasets and evaluating different novelty detection measures in practical applications, such as Active Learning.

Keywords: Uncertainty, Convolutional Neural Networks, Decision Trees, Density Forest, pre-softmax, novelty detection, patch classification, semantic segmentation, remote sensing, land cover classification

Acknowledgments

I would first and foremost like to thank my supervisors Devis Tuia and Diego Marcos Gonzalez who, with their helpful guidance, critical questions and regular feedback greatly helped me make a hopefully meaningful scientific contribution. I would like to thank Devis Tuia in particular for his financial and administrative support in allowing me to spend a semester at the University of Wageningen in the Netherlands. The periodic exchange with PhD students and researchers of the Laboratory of Geo-information Science and Remote Sensing (GRS) was a strongly motivating factor throughout my Master thesis. I would like to thank Diego Marcos Gonzalez in particular for his regular availability, assistance and critical feedback to my numerous questions. The friendly supervision within the lab of Devis Tuia not only allowed me to advance my Master thesis well but also made me feel welcome and enabled me to spend a wonderful semester in the Netherlands.

Secondly, I would like to address special thanks to my secondary supervisor Prof. François Golay at EPFL (École Polytechnique Fédérale de Lausanne), who supported my Master thesis greatly and allowed me to spend the last week of my Master thesis at EPFL.

I would like to express my gratitude to the Swiss Studies Foundation for providing me scholarships of the Binding foundation and of the Bärbel und Paul Geissbühler foundation, allowing me to focus on my studies, while leaving me with enough time to participate in various extracurricular activities to expand my personal and professional network.

Last but not least, I would also like to thank my family for having supported me both morally and financially during my Master and Bachelor studies at EPFL and at the University of Fribourg respectively. I would like to thank my mother Christine Suter in particular for her support and for having always shown great interest in my studies. I would also like to thank my grandmother Paula Geiser-Suter for her financial support as well as her continued interests in my studies, often leading to interesting discussions about engineering, research and global issues.

Contents

List of Figures

List of Tables

Acronyms

AA	Average Accuracy
AI	Artificial Intelligence
AUROC	Area Under the curve of the Receiver Operating Characteristic
BNN	Bayesian Neural Network
CM	Confusion Matrix
CNN	Convolutional Neural Network
DF	Density Forest
EM	Expectation-Maximization
FC	Fully Connected
FN	False Negatives
FP	False Positives
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
GT	Ground Truth
IF	Isolation Forest
IG	Information Gain
IR	Infrared
***k*-NN**	k-Nearest Neighbors
KS	Kolmogorov-Smirnov
LOF	Local Outlier Factor
MC-Dropout	Monte-Carlo Dropout
ML	Machine Learning
MLP	Multi-Layer Perceptron
MNIST	Modified National Institute of Standards and Technology
MSR	Maximum Softmax Response
OA	Overall Accuracy
OC-SVM	One-Class Support Vector Machine
PCA	Principal Component Analysis
PDF	Probability Density Function
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RF	Random Forest
ROC	Receiver Operating Characteristic
ROI	Region of Interest
SVM	Support Vector Machine
t-SNE	t-distributed Stochastic Neighbor Embedding
TN	True Negatives
TP	True Positives

1 Introduction

The recent breakthroughs in Machine Learning (ML), Artificial Intelligence (AI) and in computing power have set new accuracy standards in many computer vision tasks, including, but not limited to, medical applications [25, 41], autonomous driving [26] and remote sensing [53, 20, 59, 45]. In the field of image classification, a particularly successful used type of Machine Learning algorithms are Convolutional Neural Networks (CNNs). While CNNs have superseded many supervised classification approaches in terms of accuracy, CNNs typically fail to provide reliable and consistent uncertainty measures and network outputs have often been misinterpreted as probabilities [35, 16]. This has raised the question of how to find better uncertainty estimates [22, 13].

Model uncertainty accounts for incomplete information, such as our inability to distinguish a mug from a cup on an image if the handle is invisible [42]. Model uncertainty is also crucial for detecting objects belonging to a new class which was never seen in the training set. A typical application of ML methods in medical imaging is disease detection, in which ML methods can perform some of the tasks usually done by human experts [25]. In such applications, it is not only crucial to determine the diseases with high accuracy, but also to provide a measure of certainty. This is where current ML methods typically fall short of human experts, who can accurately assess their certainty and consult more colleagues if needed [25].

In remote sensing, relevant applications of uncertainty measures include tasks related to land cover classification and change detection after natural disasters. In Active Learning, model performance can be improved by asking the user to provide labels for the most uncertain pixels of an image [33, 57, 39, 50, 51]. For change detection, reliable confidence measures could help reduce the number of false alarms.

Uncertainty estimation has become an active field of research in which many different approaches to capture model uncertainty have been put forward [13, 8, 22, 47, 25, 48]. The sources of uncertainty have traditionally been classified in two parts: epistemic uncertainty, capturing model uncertainty which can be reduced by adding more training samples, and aleatoric uncertainty, representing intrinsic noise in the data, which cannot be reduced by additional training samples [22].

Contrary to probabilistic classification models, many current ML models do not provide a direct measure of confidence. Thus, many authors have proposed new network structures that are either probabilistic or that show better robustness to data transformation and fooling attempts. An example of probabilistic network architectures are Bayesian Neural Networks (BNNs), which provide a direct measure of confidence by optimizing over distributions rather than single-valued weights [13, 22]. Rather than changing the network architecture completely, some authors have proposed approaches to measure confidence by applying minor changes to existing networks, such as adding a special meta-loss function to maximize class distances between activations [29] or training a confidence detector on top of a network [2].

A different set of methods model uncertainty by working with the direct outputs of the network, which can either be activations of intermediate network layers or of the final network layer. Such methods have the advantage that they do not require a change in the network architecture and are applicable to many different standard network types. In classification, a common pitfall is to interpret the network's output activation as probabilities. Some shortcomings of using the softmax activations as a confidence measure are explained in section 2.

Most of these heuristic confidence estimation methods are either evaluated with respect to error detection, which can be seen as a binary classification task consisting of identifying each class prediction as correct or incorrect, or to novelty detection, where the confidence score is used to assess whether a point from the test set belongs to a class not yet seen in the training set [29, 34]. Typically, a confidence measure should give a low value to a false prediction in error detection and to points which it supposes to be of an unseen class in novelty detection.

This study presents a density-based clustering approach for uncertainty estimation within the framework of image classification, using the activations of the pre-softmax layer in a CNN. These activations are retrieved during test time and are clustered using an ensemble of Density Trees, called Density Forests (DFs), following the framework proposed by Criminisi et al. [9]. Similar to Random Forests (RFs), which use many Decision Trees to separate labelled data, Density Forests separate unlabelled data using several Density Trees. An Information Gain (IG) function is used to determine the best split at each tree node by maximizing the Gaussianity of the two resulting splits. Unlike Decision Trees, the goal of Density Trees is not to predict a label but to best partition the data into Gaussian-like clusters. To predict the probability of each sample to belong to a seen class, each Density Tree in a forest is descended and the Gaussian Probability Density Function (PDF) that the sample belongs to the leaf node is calculated.

The objectives of this work are the following:

1. To implement Density Forests and evaluate them with respect to novelty detection, using pre-softmax CNN activations;
2. To compare Density Forests to other novelty detection methods using pre-softmax activations as well as baselines using softmax activations;
3. To study the potential of these novelty detection methods both for the case of patch classification and semantic segmentation.

The Density Forest confidence measure is compared to other, standard novelty detection methods as well as to baseline confidence measures for CNNs. The main requirement for all tested methods is that they do not need any changes in the network architecture: they all have to work directly with the activations of either the final or intermediate network layers.

The performance of the Density Forest algorithm is evaluated both with respect to patch classification in the Modified National Institute of Standards and Technology (MNIST) dataset and with respect to semantic segmentation of the "Zurich Summer v1.0" (Zurich) dataset, using networks trained on $N-1$ of the available classes and predicting on all classes [24, 52]. Receiver Operating Characteristic (ROC) curves are used to evaluate each novelty detection measure.

This work is structured as follows: In section 2 a review of confidence estimation methods and novelty detection methods for CNNs is given. In section 3, the framework and methodology of the Density Forest algorithm are explained. Section 4 presents the datasets to which the novelty detection methods are applied. The experimental setup and implementation details of novelty detection methods are explained in section 5. Results are presented in 6 and followed by a discussion structured by types of novelty detection methods in section 7. Section 8 concludes this work with an outlook into open fields of relevant future research.

2 Literature Review

The shortcomings of standard deep learning tools in capturing model uncertainty have been pointed out by many authors, such as Gal and Ghahramani [15], Nguyen et al. [35], and Mandelbaum and Weinshall [29]. As Subramanya et al. [47] show, the notions of confidence and accuracy are related: a model with high accuracy should ideally give high confidence values to its predictions on average, as otherwise the model would suggest high confidence for incorrect predictions.

Confidence scores for ML methods are typically evaluated either in *error detection*, which aims at classifying each class prediction as correct or incorrect, or in *novelty detection*, consisting of identifying labels different from the classes seen during training. Both error detection and novelty detection can be seen as binary classification tasks, with the aim of attributing a label (-1 or 1) to each data point according to whether it is likely to be wrongly predicted, in error detection, or according to whether it belongs to an unseen class in the training set, for novelty detection [2, 29]. Although most confidence methods focus either on error detection or novelty detection, some methods are applied to both [48, 2, 29].

Novelty detection is an important field of research in Machine Learning and has been subject of numerous literature reviews [4, 30, 31, 38, 34]. Novelty detection is described as the task of classifying test data that differ from the data available during training [38]. This can been seen as "one-class classification", in which a model is trained to model the "normal" distribution, hoping that data following an "abnormal" distribution will be recognized as being different. Related topics are *anomaly detection* and *outlier detection*, which usually require the novelties to be few and different from the samples of the seen classes [34, 38].

Typologies of uncertainty heuristics have been made in categories such as "white box" methods, requiring changes in the model architecture to fit the purpose of detecting confidence, "gray box" methods, which require only some degree of re-training, and "black-box" methods, which require no re-training of the network [2]. This general scheme fits well into the framework of the developed methodology, which mainly focuses on methods requiring little changes to the network structure.

The following literature review tackles some of the most prominent confidence measures with respect to both error detection and novelty detection. A selection of these methods will be tested with respect to their performance in novelty detection and compared to Density Forests. Since many of the presented confidence measures as well as Density Forests are applied to novelty detection, some popular novelty detection methods are also briefly addressed.

The following notation is based on [9]. Vectors are denoted in bold ($\mathbf{v}$), matrices in telescope uppercase letters ($\mathtt{M}$) and sets using calligraphic notation ($\mathcal{S}$). Furthermore, following notation is used for classes and membership probabilities:

$$\mathcal{L} = \{c_i\}_{1 \leq i \leq n_c} \qquad \text{Set of classes} \tag{1}$$

$$P^{(c_i)}(\mathbf{x}) = P(\mathbf{x} \in c_i) \qquad \text{Membership probability of a sample } \mathbf{x} \text{ to class } c_i \tag{2}$$

Since there are many detailed overviews on neural networks architectures and building blocks available, they are only covered very briefly in the following literature review [43, 59, 21].

2.1 Methods Based on Network Output

2.1.1 Single-Pass Methods

A first set of methods look at the network output vector $\mathbf{P}(\mathbf{x})$ of a single pass of the input $\mathbf{x}$ through the network, typically a softmax activation vector for patch classification. The simplest possible baseline consists of considering the Maximum Softmax Response (MSR) as a confidence indicator [17, 58]:

$$C_1(\mathbf{x}) = P^{(c_1)}(\mathbf{x}) \tag{3}$$

where $\mathbf{x}$ is a data point and $c_1 = \operatorname{argmax}_{c \in \mathcal{L}} P^{(c)}(\mathbf{x})$.

A similar approach takes into account the two highest network outputs and calculates a confidence score based on the margin between the two [36, 29]:

$$C(\mathbf{x}) = P^{(c_1)}(\mathbf{x}) - P^{(c_2)}(\mathbf{x}) \tag{4}$$

where $c_1 = \operatorname{argmax}_{c \in \mathcal{L}} P^{(c)}(\mathbf{x})$ and $c_2 = \operatorname{argmax}_{c \in \mathcal{L} \setminus c_1} P^{(c)}(\mathbf{x})$.

The negative entropy H of all normalized softmax activations can be used to take into account all output activations [58]:

$$C_2(\mathbf{x}) = -H(\mathbf{P}(\mathbf{x})) = -\sum_{c \in \mathcal{L}} P^{(c)}(\mathbf{x}) \log P^{(c)}(\mathbf{x}) \tag{5}$$

The entropy is zero if all activations but one are zero and it is minimal when all activations are equally probable.

Many further, similar measures have been developed using the final network output scores to predict uncertainty, such as Sun and Lampert [48], proposing the use of network activations to predict out-of-order behavior of a network, by applying a Kolmogorov-Smirnov (KS) test to the distribution of the softmax activation vectors.

2.1.2 Invariance to Image Transformations

In contrast to the baseline methods explained above which rely on using the softmax output scores of a single pass through the network, Bahat and Shakhnarovich [2] use a more sophisticated approach which relies on multiple softmax outputs of transformed input data (fig. 1). In the case of an image classifier, a test image is transformed multiple times before prediction by applying contrast enhancements, blurring the image, applying a gamma filter or by flipping and rotating the image. The assumption is that a classifier will react differently to transformations in the case of correctly and incorrectly classified data points, being more invariant to transformations of correctly classified input. Softmax activations are retrieved for the original image as well as for each transformed image using the same CNN model. All softmax scores are then reordered in the rank of descending softmax response in the original image. The scores can be truncated such as to keep only the first n components in each softmax response, reducing the subsequent calculation time. All reordered and truncated softmax scores are then concatenated and used as a training set for a Multi-Layer Perceptron (MLP), a simple Neural Network structure, which calculates the probability of the network to make a wrong prediction.

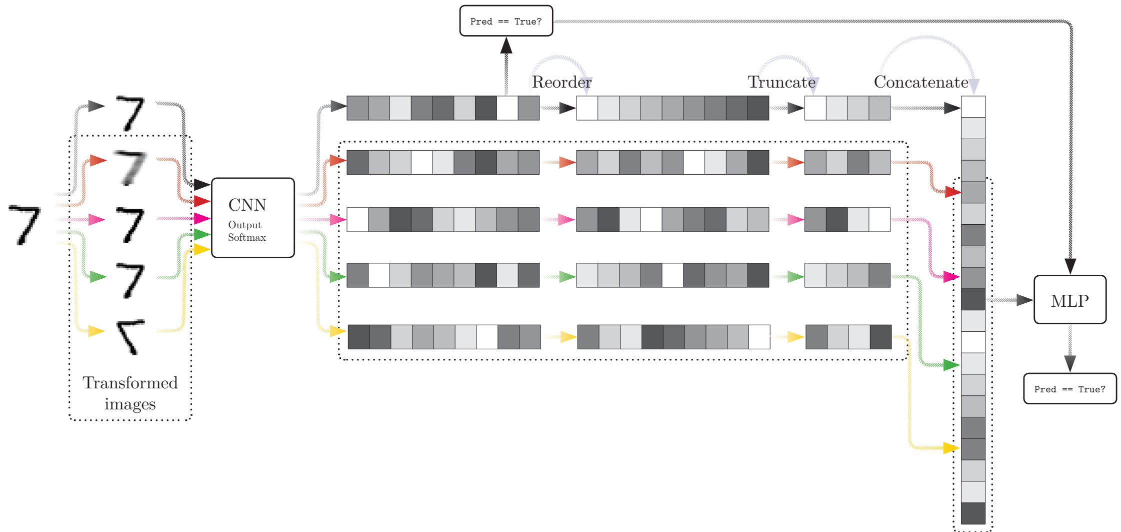


Figure 1: Invariance To Image Transformation: Schema for MNIST digit example [2].

This approach requires training an MLP on top of an existing network to assess uncertainty, which can however be done efficiently and in linear time in the number of image transformations. They also propose a slightly more complex version of their algorithm for novelty detection [2].

2.1.3 MC-Dropout

Gal and Ghahramani [15] propose the use of Monte-Carlo Dropout (MC-Dropout) during prediction to quantify the model uncertainty. In order for MC-Dropout to work the neural network has to be trained with dropout applied after every convolution layer. Then, in order to retrieve the uncertainty $C(\mathbf{x})$ for a data point $\mathbf{x}$, T stochastic forward passes through the network are performed, each time randomly dropping a certain number of weights using test-time dropout. Thus, only a subset of all available weights defined by dropout probability p is used for prediction. The predictions are calculated as the average output of all T runs and the certainty as the mean variance of the softmax outputs. This method has been used as a de-facto uncertainty measure for neural networks in many papers [29, 25, 23, 47, 21].

2.2 Methods Based on Network Activations

As Gal and Ghahramani [15] point out, a model can be very uncertain about its predictions despite yielding a high softmax output. Nguyen et al. [35] demonstrate that neural networks can be easily fooled using an algorithm that generates images unrecognizable to humans but creating almost perfectly certain predictions by the softmax activation output of the network. In addition, Goodfellow et al. [16] have shown that an adversary network can induce minor perturbations into an image which create false predictions with very high confidence. Thus, many authors have decided to model the implicit information contained in the network feature vectors before the softmax activation for error detection and novelty detection tasks [47, 29, 4].

Amongst other authors, Subramanya et al. [47] point out that softmax scores can vary drastically when applying transformations such as Gaussian noise, blurring or JPEG compression to an image. In order to provide a more robust confidence estimator, they propose to model the density over each seen class in the pre-softmax activations: In order to estimate the confidence score of a prediction in a classification problem, they calculate the Gaussian densities belonging to the activations of the N classes. Their confidence estimate yields more robust to image transformations such as noise injection and JPEG compression, but may suffer in case of Gaussian blurring and adversarial training.

Mandelbaum and Weinshall [29] have opted for a similar approach, however approximating local density by the Euclidean distance between a point and its k-Nearest Neighbors (k-NN) in the feature space created by the network in one of its pre-softmax layers. During test time, their method retrieves the activations of each data point and computes the distance to the k nearest neighbors of each sample. In order to provide reasonable uncertainty estimates, this method requires either adversarial training or a special loss function in which the distance between pairs of adjacent points belonging to different classes is maximized. Without this special loss function, their method yields worse or similar results compared to the baselines margin and entropy [29]. Using a modern GPU implementation of nearest neighbor search, they achieve a computational complexity of $O(kN)$, k being the number of nearest neighbors to search and N the number of data points, making their method scale well with big datasets.

2.3 Network Architectures for Confidence Estimation

The current state-of-the-art method to estimate uncertainty in ML is based on BNNs, allowing for direct estimation of the uncertainty from the model probabilities [14, 15, 22]. However, this approach requires a certain number of changes, such as optimizing over weight distributions instead of fixed weights. However, in Gal and Ghahramani [14, 15], the authors demonstrate the equivalence regarding confidence estimation between a BNN and a standard MLP with dropout applied after every weight layer during test time (MC-Dropout).

Goodfellow et al. [16] show that adversarial examples can be used to improve the resistance of the network against *fooling* attempts (sec. 2.2), using *Adversarial Training*. This however comes at a high price in computational cost at up to twice the training time [29].

Further methods have been developed based on *ensembles* of NN models, such as in Lakshminarayanan et al. [23] and Mandelbaum and Weinshall [29]. These methods however suffer from heavily increased computation complexity for training several models.

2.4 Comparison to Novelty Detection Methods

Many more novelty detection methods have been developed to model the support of data belonging to the "normal" class versus new "anormal" data [38, 30, 31]. Novelty detection methods have been mainly characterized as either probabilistic, distance-based, reconstruction-based and domain-based. Since not all methods could be implemented and evaluated in this work, only a subset of the most popular novelty detection are covered here.

Two of the most common novelty detection methods are presented in this paper. The first are Gaussian Mixture Models (GMMs), which assume that the data was generated by an underlying mixture of Gaussian distributions [40]. GMMs work by initializing a pre-defined number of Gaussian distributions with random mean and covariance parameters, then fitting them iteratively to the training data via the Expectation-Maximization (EM) algorithm [10]. Some applications of GMMs to novelty detection are presented in [38]. The second are One-Class Support Vector Machines (OC-SVMs), in which the support boundary is searched for a given set of training points [54, 3]. Support Vector Machines (SVMs) try to find a linear hyperplane using a kernel function which can represent similarities between pairs of data points in an abstract geometric space [49].

A difference between anomaly detection and novelty detection is that anomaly detection usually assumes novelties to be rare [34]. Some popular methods for anomaly detection include Isolation Forests (IFs) and Local Outlier Factor (LOF) [38, 7, 27]. In contrast, GMMs and DFs can be seen as clustering methods, which allow detecting novelties even if they are abundant.

2.5 Summary

A summary of the reviewed literature and methods for confidence estimation in neural networks is shown in table 1. Since the existing body of research on confidence estimation is vast and many related papers exist to each of the presented topics, only a selection of reviewed papers is presented.

Type	Method Name	Focus	Reference
Softmax activations	**MSR, Margin, Entropy**	Softmax activations	[17]
	Tranformations Invariance	Input data Perturbations	[2]
	MC-Dropout	Dropout during prediction	[15]
	KS-test	KS-test on output distribution	[48]
Pre-softmax activations	*Density*	Density Modelling	[47]
	Distance	k-NN Distance	[29]
New Network Architectures	*Adversarial Training*	Training Perturbations	[16]
	BNNs	Probabilistic Modelling	[22]
	Ensembles	Multiple Neural Network	[29]

Table 1: Summary of reviewed confidence measures for neural networks. Implemented baselines are indicated in bold.

3 Methodology

3.1 Overview: Density Forests

Similar to the activations-based methods presented in section 2.2, the confidence measure developed below tries to distinguish activations of the training set from new, unseen activations in the test set. Similar to the way Decision Trees and Random Forests are used in the case of labelled data, Density Trees and Density Forests are used for unlabelled data to distinguish "normal", seen data from unseen, new data points.

The methodology proposed below is based on Criminisi et al. [9], who propose a unified framework for random decision forests, which can be extended to classification, regression, density estimation and other tasks. Therefore, this unifying framework of decision forests is first addressed before covering the extension to Density Forests.

An overview of the methodology for novelty detection using Density Forests is shown in figure 2.

3.2 Decision Trees

A Decision Tree is a binary tree consisting of hierarchical nodes and edges which, at each level, provide the most important features to determine the outcome of a dependent variable, given an arbitrary number of independent variables [9]. Using an entropy function $H(\mathcal{S})$, the goal of a Decision Tree is to find the best possible split to partition the feature space $\mathcal{S}$ at each level starting from top to bottom such as to maximize an Information Gain function:

$$I = H(\mathcal{S}) - \sum_{i \in \{l,r\}} \frac{|\mathcal{S}_i|}{|\mathcal{S}|} H(\mathcal{S}_i) \tag{6}$$

Where I is the Information Gain, H is an optimizer or entropy function, $\mathcal{S}$ is the original dataset at a node and $\mathcal{S}_i$ is left and right data subset resulting from the split. For instance, in the case of labeled data and decision trees, the Shannon entropy is often used (equation 5).

Common stopping criteria for individual trees include maximum tree depth and minimum IG for a given split. Such stopping criteria aim at avoiding overfitting and creating smooth decision boundaries.

3.3 Random Forests

Smoother and more generalized boundaries can be obtained by using Random Forests, which combine a set of individually trained trees on bootstrapped subsamples of the initial data [9, 6]. Bootstrapping is a technique of sampling a smaller subset of data from a larger subset with replacement [60]. The aggregation of multiple predictors based on bootstrapped samples is usually referred to as bagging [5]. A simple visualization of Random Forests on generated synthetic data is provided in appendix B. Aside of the parameters and stopping criteria mentioned for individual trees, important factors for Random Forests are the number of trees, the subsample size of the initial data

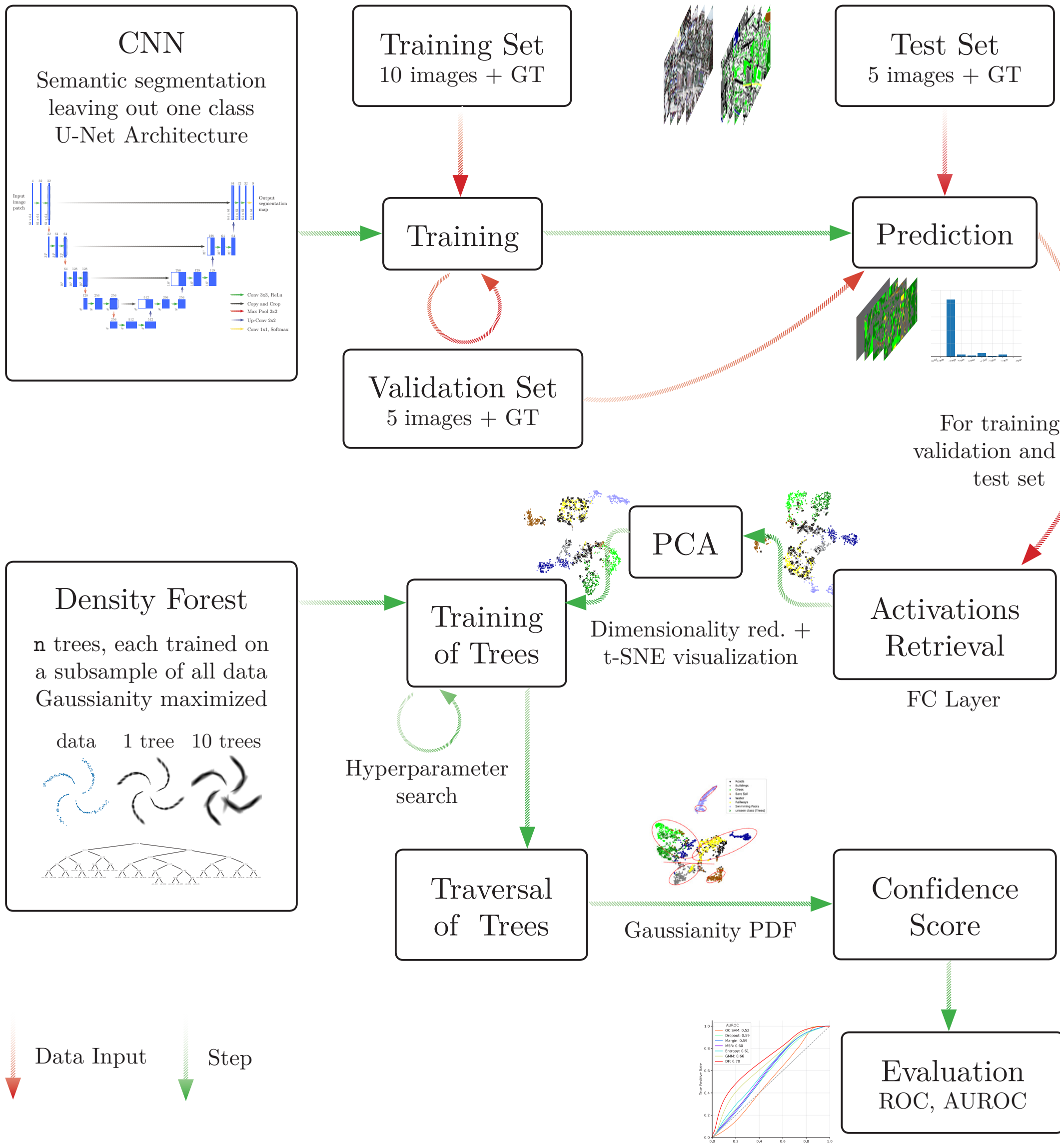


Figure 2: Workflow Diagram

on which each tree is trained and the number of dimensions each node may consider as a candidate dimension for splitting.

3.4 Density Forests

A Density Tree can be seen as similar to Decision Trees, but with the aim of clustering *unlabelled* data into regions that maximize the Gaussianity of each cluster [56]. Same as with Decision Trees, a generic Information Gain function is maximized (eq. 6). An unsupervised entropy function is designed based on the assumption of a multivariate Gaussian distribution at each tree node [9]:

$$H(\mathcal{S}) = \frac{1}{2}\log\left((2\pi e)^d|\Lambda(\mathcal{S})|\right) \tag{7}$$

Λ being the associated $d \times d$ covariance matrix. Hence, the Information Gain at the j^{th} split becomes [9]:

$$I_j = \log(|\Lambda(\mathcal{S}_j)|) - \sum_{i\in\{l,r\}} \frac{|\mathcal{S}_j^i|}{|\mathcal{S}_j|}\log(|\Lambda(\mathcal{S}_j)|) \tag{8}$$

For a description of the motivation behind this optimization method, refer to Criminisi et al. [9].

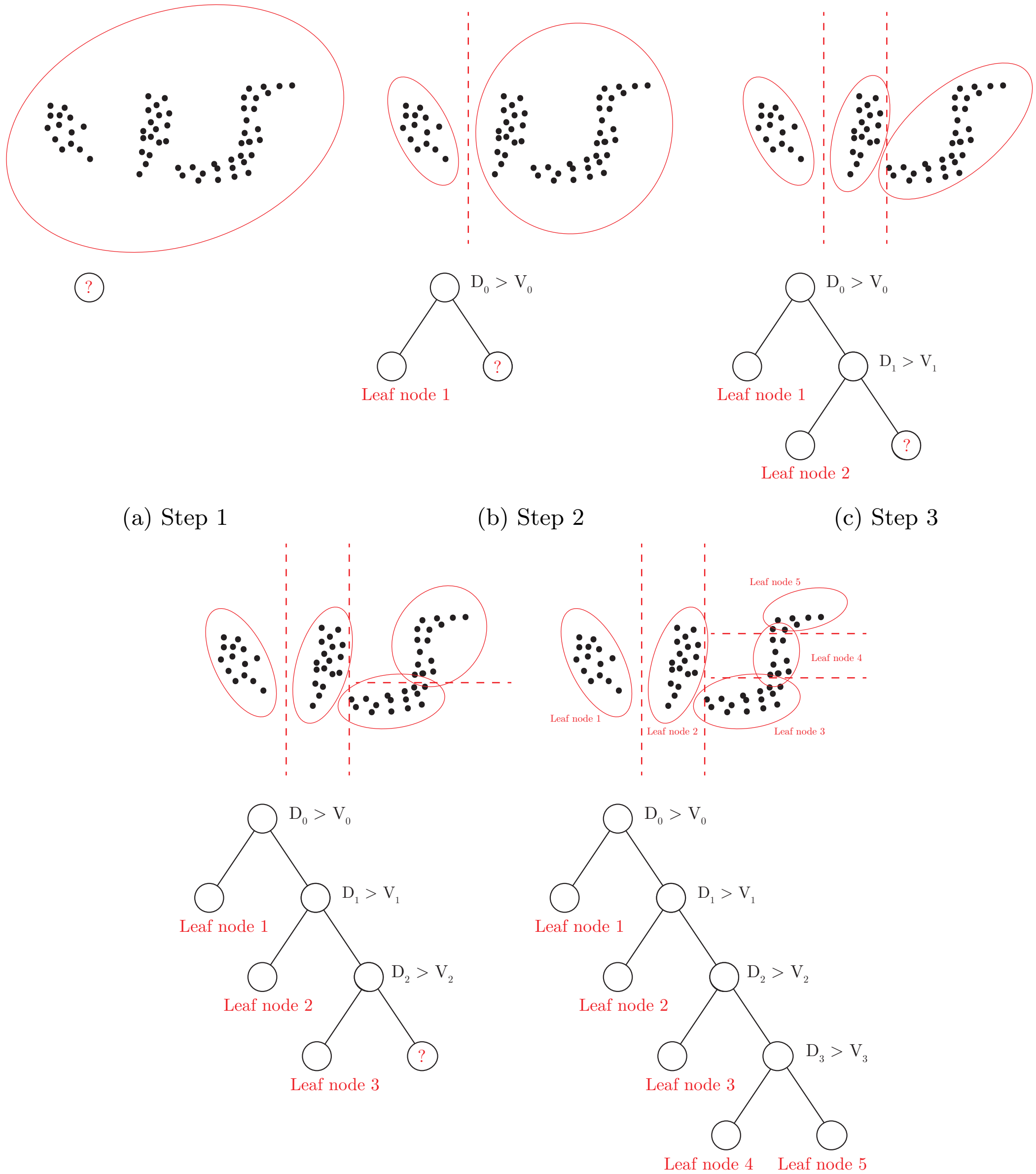
At prediction time, a given data point descends the Density Tree according to the split dimension and value split associated with each tree node until it reaches a leaf node. The probability of a test data point to belong to root node of the tree is calculated as follows [9]:

$$p_t(\mathbf{x}) = \pi_l \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{l(\mathbf{x})}, \Lambda_{l(\mathbf{x})}) \tag{9}$$

With $\boldsymbol{\mu}_{l(\mathbf{x})}$ and $\Lambda_{l(\mathbf{x})}$ denoting the mean and covariance of the leaf node corresponding to the data point $\mathbf{x}$, π_l being the proportion of points falling into the respective leaf node. The multivariate Gaussian PDF is defined as follows:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \Lambda) = \frac{1}{\sqrt{(2\pi)^d \det \Lambda}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \Lambda^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{10}$$

Where $\boldsymbol{\mu}$ is the mean, Λ is the co-variance matrix and d is the number of dimensions of $\mathbf{x}$ [44]. The thus obtained probability is weighted by the percentage π_l of all the data used for training the Density Tree falling into this leaf node. For the purpose of this study, no probability normalization term has been implemented, as we are only interested in relative confidence values between classes. Figure 3 illustrates the splitting steps for a single Density Tree.



(a) Step 1 (b) Step 2 (c) Step 3

(d) Step 4 (e) Step 5

Figure 3: Illustration of tree growth for a fictive dataset. Covariance ellipses are indicated in red lines and splitting lines in red dotted lines, indicating the dimension and value along which a node is split. D_i and V_i denote the split dimension and value of the i-th split. Nodes are considered leaf nodes when no further split is necessary or possible, either because the Information Gain of a new split would be lower than a defined threshold or because the maximum tree depth is reached. In a Density Forest, every tree would see a subset of all points and fit slightly different leaf nodes each time.

4 Datasets

4.1 Synthetic Datasets

First, some synthetic data were produced to test how well individual Density Trees could fit points generated in 2-dimensional space, and visualize the decision boundaries. The following three data generators have been implemented:

1. A generator which produces n clusters according to Gaussian distributions $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \Lambda)$ with parameters $\boldsymbol{\mu}, \Lambda$. The mean μ is chosen between a defined minimum and maximum value in each dimension and the covariance Λ is initialized as an identity matrix multiplied by a given coefficient. For each cluster, the covariance matrices Λ can be randomly multiplied by a coefficient in only one dimension, such as to create an elongated cluster, and non-linear transformations can be applied to make the cluster less Gaussian-like.

2. A data generator producing points along a spiral, according to following parametric equations:

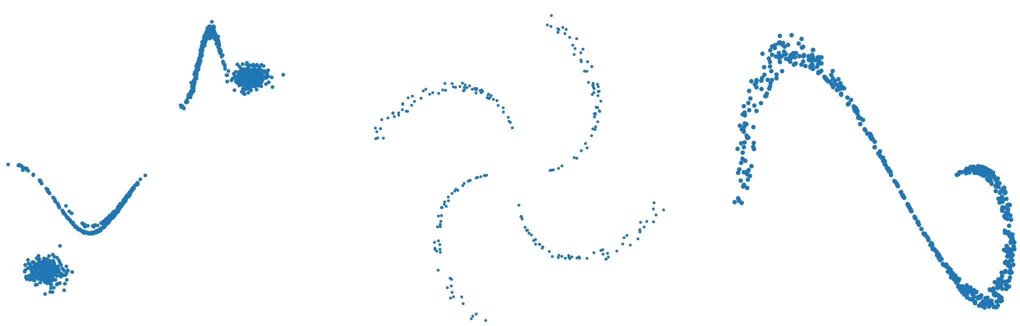
$$\begin{aligned} x &= a\sqrt{\theta}\cos\theta \\ y &= a\sqrt{\theta}\sin\theta \end{aligned} \tag{11}$$

   θ being the angle and a being the distance between successive terms of the spiral.

3. A generator which produces data along an S shape similar to the method above, but with only one arm, generating data points as follows:

$$\begin{aligned} x &= a\sqrt{\theta}\sin\theta \\ y &= xa\sqrt{\theta}\sin\theta \end{aligned} \tag{12}$$

Datasets generated according to these three data generators are shown in figure 4.



(a) 4 clusters with non-linear transformations applied to two clusters

(b) Spiral with 4 arms

(c) S-shaped distribution

Figure 4: Synthetic datasets

4.2 MNIST Dataset

Confidence Measures have been first applied to the MNIST dataset, containing 24×24 gray level images of handwritten digits from 0 to 9 and corresponding labels [24]. The training set and test set consist of 60'000 and 10'000 samples respectively, with roughly the same amount of images for each class. The MNIST dataset is used as a de-facto baseline to test many algorithms related to computer vision and Machine Learning tasks. Some examples of handwritten digits are shown in figure 5.

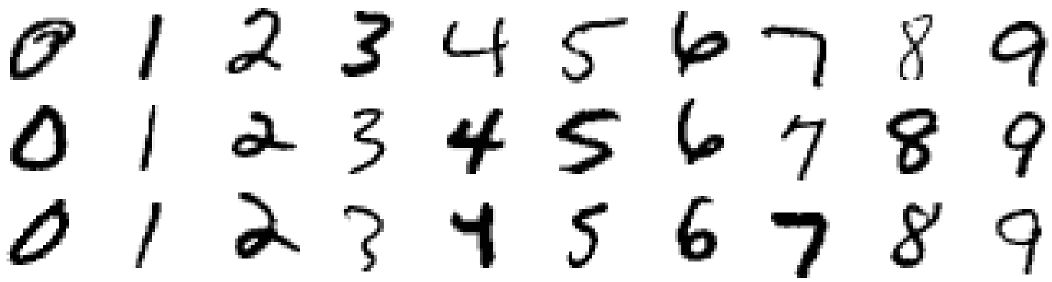


Figure 5: Sample images of the MNIST dataset for true y labels 0 to 9 [24]

4.3 Zurich Dataset

In addition to the MNIST dataset, the confidence measures were applied to semantic segmentation, which consists of attributing a class label to every pixel of an image [53]. For this task, the "Zurich Summer v1.0" (Zurich) dataset was used, consisting of a set of 20 RGB-Infrared (IR) satellite images taken from a QuickBird acquisition of the city of Zurich (Switzerland) in August 2002, together with a corresponding set of annotated ground truth images [52]. 8 different urban and peri-urban classes have been annotated: Roads, Buildings, Trees, Grass, Bare Soil, Water, Railways and Swimming Pools. The Zurich dataset was split into a training set consisting of images 1-10, a validation set consisting of images 11-15 and a test set consisting of images 16-20, corresponding roughly to a 50/25/25 split, although the images are of varying sizes. An example pair of an RGB-IR image and its ground truth is given in fig. 6.

(a) RGB Image (IR not shown) (b) Ground Truth

Background Roads Buildings Trees Grass
Bare Soil Water Railways Swimming Pools

Figure 6: Sample pair of images and ground truth for the Zurich dataset [52]

Classes are imbalanced, with only very few samples for some classes, such as swimming pools or railways (fig. 7).

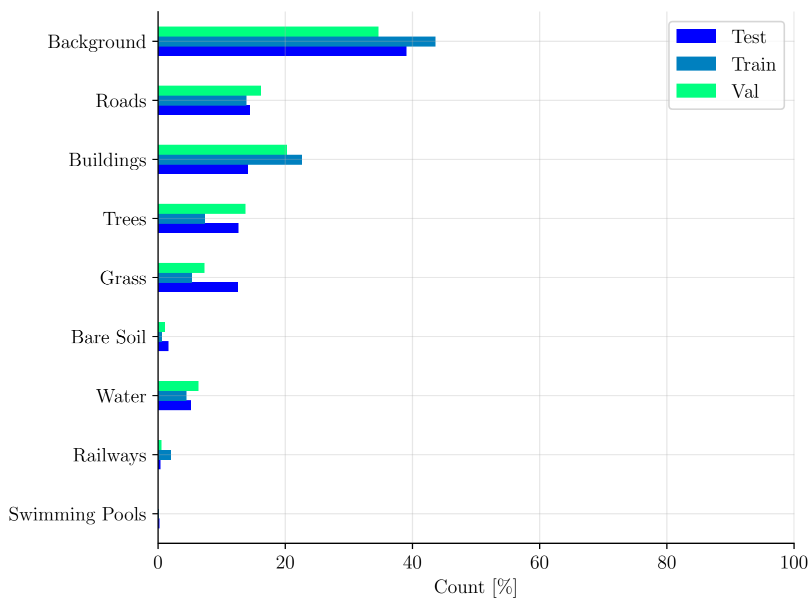


Figure 7: Class distribution in the Zurich dataset

5 Experimental Setup

The following section covers network architectures for training CNNs, the structure of experiments, implementation details for novelty detection methods as well as evaluation and hyperparameter search schemes.

5.1 Network Architectures

5.1.1 MNIST dataset

10 CNN models were each trained on about 54'000 training images of the MNIST dataset without images of the left-out class, and validated on 10'000 test images, including images of the left-out class. CNNs were trained for novelty detection using the network architecture described in table 2. Dropout was added after each MaxPooling layer to avoid overfitting [46]. The Rectified Linear Unit (ReLU) activation function ($f(x) = \max(0, x)$) was used after each convolution layer of the network. The Fully Connected (FC) layer denotes the pre-classification layer in the network, in which every input is connected to every output by a weight and followed by a ReLU activation.

Layer	Parameters
Input	Dimension $\|b\| \times 28 \times 28$
Convolution + ReLU	32 3×3 filters
Convolution + ReLU	64 3×3 filters
MaxPooling	2×2 pool size
Dropout	$p = 0.25$
Flatten	
Dense (FC) + ReLU	128 neurons
Dropout	$p = 0.5$
Output + Softmax	9 neurons

Table 2: Architecture of the CNN used for MNIST digit classification. $|b|$ = batch size, p = dropout probability.

5.1.2 Zurich dataset

Similar as for the MNIST dataset, CNN models were trained for each left-out class, setting each time the labels of the left-out class to "background" in the training set. The U-Net network architecture was applied to perform semantic segmentation on the Zurich dataset (fig 8). It consists of a sequence of down-sampling and up-sampling layers, yielding a "U"-shaped layer architecture [41]. This architecture has shown good performance in various semantic segmentation tasks, and has been especially applied in medical imaging [41, 1, 12]. A full overview of the layers is given in appendix table E.1.

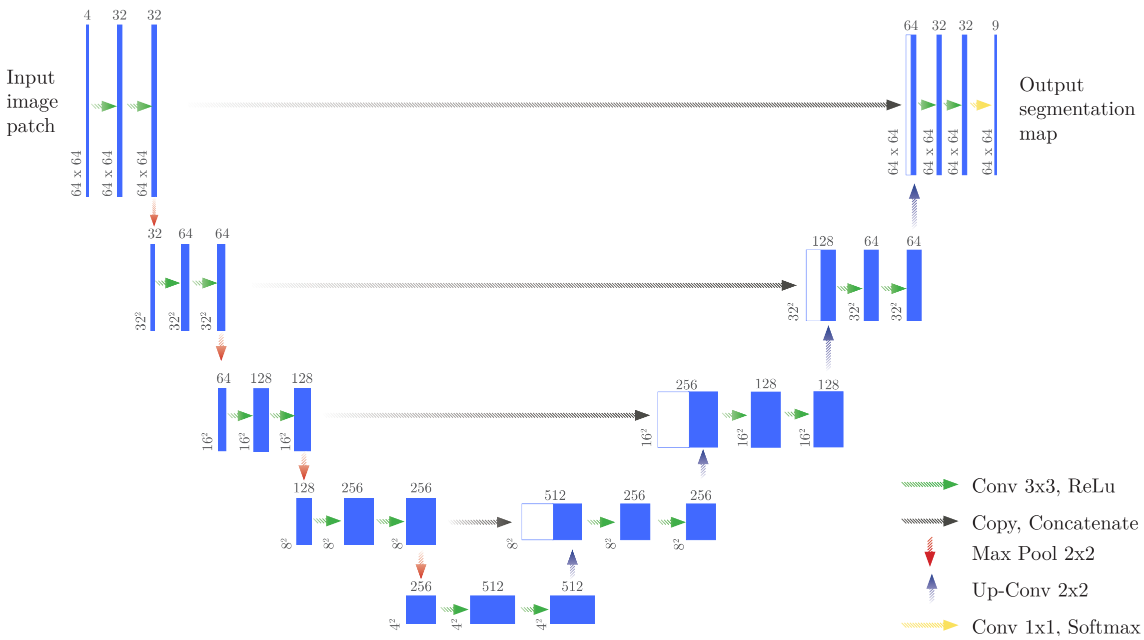


Figure 8: Modified U-Net architecture, according to Ronneberger et al. [41]. Numbers on top of each blue block indicate the number of filters, numbers on the bottom left of the blocks indicate the resolution of each filter. Each convolution layer was padded to avoid cropping.

Image Tiling and Padding For training the U-Net model, the training and validation images were tiled into *non-overlapping* patches of patch size $64 \times 64 \times n_c$ where n_c is the number of channels. For prediction, the test set images were tiled into *overlapping* patches using the same patch size of $64 \times 64 \times n_c$, with shorter strides between patches, keeping only the central overlapping $stride \times stride$ pixels between adjacent patches. Central pixels of a patch are assumed more certain because of the larger available spatial context in the patch compared to pixels near the patch borders which partially lack neighbor information, leading to strong border effects [32]. Although the optimal stride would seem to be 1, keeping only the central pixel would lead to excessive computational costs with little additional accuracy gain. Therefore, intermediate strides have to be chosen empirically. Several strides were chosen and results averaged to further reduce border effects. For instance, for a stride of 32 pixels and a patch size of 64 pixels, patches of $64 \times 64 \times n_c$ are created and only the $32 \times 32 \times n_c$ central pixels are kept for the final prediction. Since only the central part of each patch is kept, the original images first had to be padded in order to avoid cropping. For a given stride and patch size, the image was first padded on each side to make it divisible by the amount of overlapping pixels, and by an additional amount *(patch size - stride)* / 2 on either side of the image to be able to fit patches of size 64×64 with the given stride around the entire padded image. For each patch, only the $stride \times stride$ central pixels were kept, concatenated and the pad was removed from the image to give it the same dimensions as the input image.

Data Augmentation Data augmentation was performed to improve the accuracy of the network. Following transformations have been applied both to the image and ground truth patches, according to Volpi and Tuia [53]:

1. Extraction of patches at random locations of the training set images to avoid seeing the same types of patches in each training minibatch. Class imbalance is not considered during patch extraction.
2. Random horizontal and vertical flipping and random rotations in steps of 90 degrees (0, 90, 180 or 270 degrees). These transformations aim at improving invariance to differing spatial organizations of the image.
3. Noise injection: Add noise to every pixel of the image, sampling the noise from a Normal distribution in $\mathcal{N}(0, 0.01)$ with subsequent rescaling of the data between $[0, 1]$

5.2 Novelty Detection Baselines

Some implementation details are discussed below for for $DensityForests$ and for novelty detection baselines to which Density Forests are compared.

5.2.1 MSR, Margin, Entropy

The confidence measures relying on the softmax output of a network were implemented using the equations of section 2 (eq. 3, 4 and 5).

5.2.2 MC-Dropout

In this study, a slightly simplified version of the MC-Dropout algorithm was implemented, performing dropout during test time only in the pre-softmax layer and calculating the entropy on the mean of the softmax activations (eq. 5). This has the advantage of speeding up calculations by repeating only the forward pass through the last layers of the network instead of having to repeat the entire forward pass.

5.2.3 Pre-Softmax Methods

In addition to the implemented baselines indicated in bold in table 1, novelty detection methods GMM and OC-SVM have been implemented to detect novelties using the network's pre-softmax activations. In this work, the GMM and OC-SVM approaches have been applied to novelty detection using the same activations as Density Forests. Both GMM and OC-SVM model the "normal" data, consisting of activations of seen classes. As confidence values, GMM calculates the log-likelihood for a given a data point and OC-SVM calculates the signed distance from the separating hyperplane, being positive for inliers and negative for outliers. For both GMM and OC-SVM, the scikit-learn implementations were used [37].

5.3 Density Forests

Density Forests have been implemented according to the formulas in section 3. Within each dimension of the dataset, a range of possible split values is checked to find the split which minimizes entropy, i.e., maximizing Information Gain. For each candidate split dimension, a number of candidate split values have been tested randomly between the interval of the *dim*-th smallest and *dim*-th largest data point values, such as to ensure at least *dim* datapoints to either side of the split to ensure invertible matrices.

Stopping criteria for growing individual tree branches have been implemented according to [9], based on maximum maximum tree depth and based on the minimum Information Gain required at each node to continue splitting. The implemented data structure for Density Trees and Decision Trees is represented in appendix A.

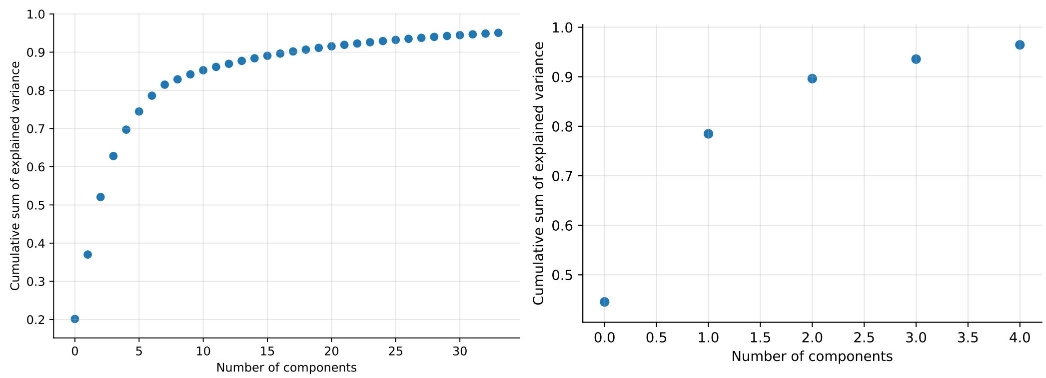
Parameters for Density Forest are listed in table 3.

Parameter	Description	Parameter Range
`n_trees`	Number of trees to create. A higher number of trees linearly increases training and prediction time.	10 - 50
`max_depth`	Maximum depth allowed for each tree. Deeper trees lead to more clusters.	1 - 5
`subsample_pct`	Size of each random data subset sampled from the original dataset used as input for a Density Tree, in percentage of the size of the original dataset	(0, 1)
`min_subset`	Minimum dataset size of each cluster, indicated as percentage of the original dataset size.	0 - `subsample_pct` / 2
`n_max_dim`	Number of dimensions to consider for splitting at each node. If `dim` ≤ 0, all dimensions are considered. If `dim` > 0, a random number of dimensions between 1 and dim is considered for splitting at each node. Further adds randomization to trees.	[0, `dim`)
`ig_improvement`	The Information Gain improvement that has to be made at each node in order to create a new split. Avoids unnecessary splits in already Gaussian-like clusters	(0, 0.9)

Table 3: Density Forest parameters and suggested parameter ranges.

5.4 Dimensionality Reduction and Data Separability

For both the MNIST dataset and the Zurich dataset, activations of the fully connected layer were retrieved, resulting in a large number of input dimensions. For example, the FC layer of the MNIST dataset contains 128 filters, therefore, for a batch of n patches, the activations retrieved during prediction are of dimension $n \times 128$ (table 2). Many of these activations are similar, which causes collinearity and matrix inversion issues during the calculation of the Gaussian entropy. Therefore, a dimensionality reduction was performed using Principal Component Analysis (PCA). For both the MNIST and Zurich datasets, PCA components were chosen such as to explain over 95% of the data variance. For MNIST, the first 30-40 components were kept, while for the Zurich dataset, the first 5-10 dimensions were kept (fig. 9. Despite computational complexity induced by higher dimensions, lower-dimensional data also reduces problems related to the curse of dimensionality, which causes notions of distance between points to become meaningless and which, together with data noise, negatively impacts performance of many clustering algorithms [18, 19]. The number of components kept for each class in the MNIST and Zurich datasets are shown in appendices F and G.



(a) MNIST dataset, left-out class 8

(b) Zurich dataset, left-out class "roads"

Figure 9: Cumulative variance explained as a function of the number of components

Separability between activations corresponding to different classes was ensured before and after PCA, using t-distributed Stochastic Neighbor Embedding (t-SNE) which allows visualization of high-dimensional data [28]. t-SNE tries to find a mapping between high-dimensional data points and their representation in a lower-dimensional space, typically in 2 to 3 dimensions, by preserving the local structure of the original, high-dimensional data (fig. 10). To preserve the original data structure, t-SNE models the pairwise similarity between data points in their high-dimensional and lower-dimensional space. Similarities between pairs of data points x_j and x_i are measured as the conditional probabilities of x_i choosing x_j as its neighbor, if neighbors were picked in proportion to their probability density of a Gaussian centered at x_i [28]. t-SNE minimizes the sum of Kullback-Leibler divergences over all data points, measuring the differences between similarity values in higher and lower dimensions. Since t-SNE directly transforms the training data without finding a parametric mapping and has a computational and memory complexity quadratic in the number of data points, its scalability is very limited [28].

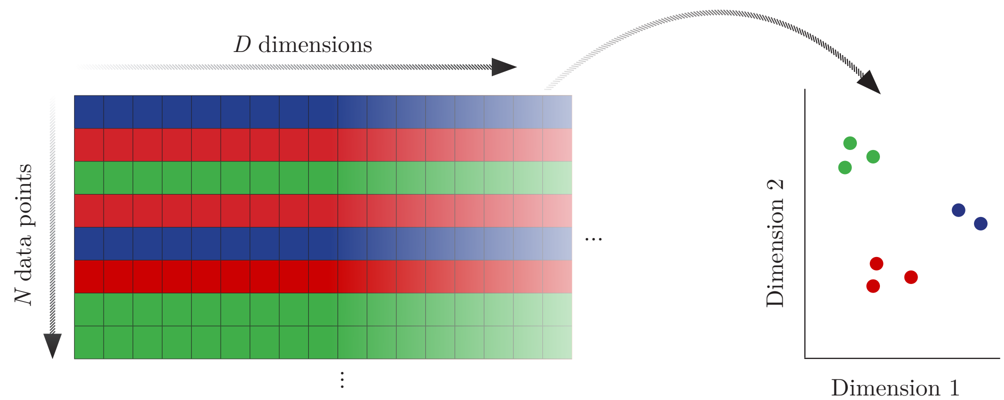


Figure 10: t-SNE schema with toy data. t-SNE finds a mapping between the original, high-dimensional data (left) and the lower-dimensional data representation (right). Classes of data points are indicated in blue, red and green. If data points within the same class are more similar to each other in high-dimensional space, they will be closer to each other in the t-SNE visualization.

Density Forests were trained on the dimensionality-reduced activations belonging to the seen classes of the training set. Based on the observation that wrongly predicted points of seen classes are often outliers and have activations further away from the cluster centers, only activations of correctly predicted training points were used, to improve the performance of pre-softmax novelty detection methods. The baselines GMM and OC-SVM both use the same dimensionality-reduced activations as the Density Forests for the hyperparameter search and for fitting the final model with the best found parameters. For all methods based on pre-softmax activations, hyperparameters are found using the scheme described in section 5.6. The same procedure was also applied to error detection, in which case the training set activations of correctly predicted points were used to fit the GMM, OC-SVM and Density Forest. The performance of the confidence measures for error detection are only presented in the appendix G.

5.5 Evaluation

For CNN models, accuracy metrics are indicated using Overall Accuracy (OA) and Average Accuracy (AA) as well as precision and recall (table 4). While OA reports the percentage of all data points correctly classified, AA calculates the mean of the percentage of correctly classified data points per class. While recall (R) measures how large a fraction of the expected results is actually found, precision (P) measures how many of the results returned are actually relevant. Precision and recall are based on the confusion matrix shown in table 4. True Positives (TP) True Negatives (TN), False Negatives (FN) and False Positives (FP) denote positives and negatives correctly predicted as positives and negatives repectively for TP and TN, and positives and negatives falsely predicted as negatives and positives respectively for FN and FP.

$$
\begin{aligned}
R &= \frac{TP}{TP + FN} \\
P &= \frac{TP}{TP + FP}
\end{aligned}
\tag{13}
$$

he F_1 score is often used as a weighted harmonic mean between precision and recall:

$$
F1 = \frac{2PR}{P + R} \tag{14}
$$

		True labels y				PA
		1	2	…	r	
Predicted labels y	1	n_{11}	n_{12}	…	n_{1r}	$n_{11}/n_{1\bullet}$
	2	n_{21}	n_{22}	…	n_{2r}	$n_{22}/n_{2\bullet}$
	⋮	⋮	⋮	⋱	⋮	⋮
	r	n_{r1}	n_{r2}	…	n_{rr}	$n_{rr}/n_{r\bullet}$
UA		$n_{11}/n_{\bullet 1}$	$n_{22}/n_{\bullet 2}$	…	$n_{rr}/n_{\bullet r}$	OA= $\sum_i n_{ii}/n_{\bullet\bullet}$

Table 4: Confusion Matrix for classification problem with r classes. UA = User's Accuracy = Precision, PA = Producer's Accuracy = Recall, OA = Overall Accuracy, $1, 2, \ldots, r$=classes. n_{ij} counts the number of labels predicted as class i and belonging to the true class j. Bullet indexes signify either the sum of the row (e.g., $n_{O\bullet}$), the sum of the column (e.g., $n_{\bullet O}$) or the sum of all elements of the Confusion Matrix ($n_{\bullet\bullet}$).

Evaluation of the confidence measures is made according to the target of novelty detection: We are interested in a binary outcome, predicting whether a new point will belong to a class seen during training or to an unseen class. Performance of the uncertainty measures is evaluated using ROC curves, which plot the TP rate against the TN rate for varying confidence thresholds. The Area Under the curve of the Receiver Operating Characteristic (AUROC) is used to summarize the ROC curve.

5.6 Hyperparameter Search

For the synthetic datasets, no hyperparameter search was performed since their only purpose is to illustrate the behavior of Density Trees and Density Forest. Density Trees and Density Forests were trained using the parameters listed in table 5. The same parameters have been used for each synthetic dataset.

Parameter	Value
`max_depth`	5
`n_trees`	20
`subsample_pct`	.01
`min_subset`	.0001
`n_max_dim`	-1
`ig_improvement`	.5

Table 5: Density Forest parameters for each synthetic dataset

For the MNIST dataset and the Zurich dataset, best hyperparameters were searched for the novelty detection methods OC-SVM, GMM as well as Density Forest. The ranges of tried parameter combinations are listed in appendix H. For each candidate parameter combination, models were trained in parallel several times on a subset of the training set activations belonging to points of the seen classes and evaluated on a subset of all validation data[1]. Finally, the parameter set with the highest AUROC was applied to the test set. The final AUROC indicated in the results is calculated on the test set, using the best parameters found as described before. In addition to finding best hyperparameters, kernel spaces have been visualized for OC-SVM (fig. H.1 and H.2). Hyperparameter results and visualizations are shown in appendix H.

6 Results

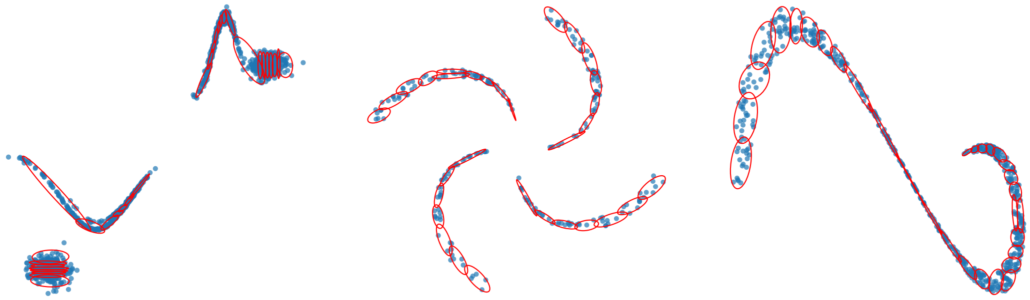
6.1 Experiments

In the following section, the experiments will be presented in the following order: In section 6.2, Density Forests will be illustrated by fitting individual Density Trees as well as Density Forests to the generated synthetic data, visualizing the effects of the Density Tree parameters on the number and shape of generated clusters. In section 6.3, Density Forests as well as the baseline confidence measures explained in section 2 will be applied to the MNIST dataset and their performance will be compared with regard to novelty detection. In section 6.4, Density Forests as well as the baseline confidence measures will be applied to the Zurich dataset, again measuring performance with respect to novelty detection and highlighting in addition some objects with particularly low confidence values.

6.2 Synthetic Dataset

Figure 11 shows points of the synthetic datasets generated according to the data generators described in section 4.1 and covariance ellipses of an individual Density Tree trained on all data using the parameters listed in table 5.

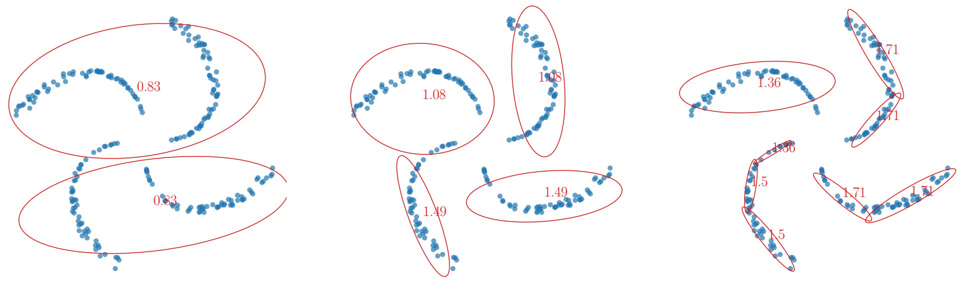
[1]For the class "swimming pools" of the Zurich dataset, hyperparameters were both trained and evaluated on the training set, since the there are no samples of the "swimming pools" class in the validation set.



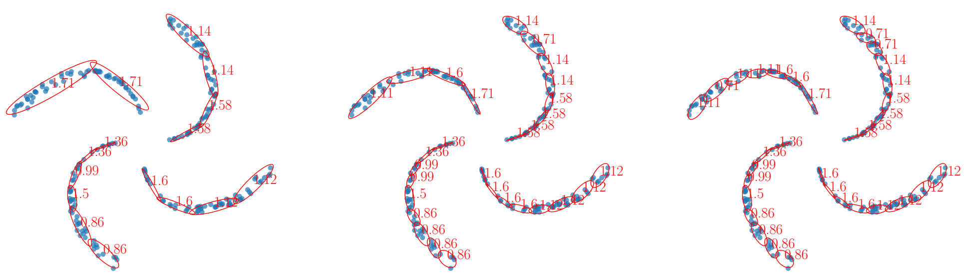
(a) Dataset 1 (b) Dataset 2 (c) Dataset 3

Figure 11: Covariance ellipses of individual Density Tree fitted on synthetic datasets

Figure 12 shows the covariance ellipses of the splits found at each depth of a Density Tree used to fit dataset 2. With deeper levels of the tree, more splits are created, but only in regions of the spiral arms with fewer points. It seems that in these regions, since there are less points, more ellipses are necessary to fit the local distribution.



(a) Depth = 0 (b) Depth = 1 (c) Depth = 2



(d) Depth = 3 (e) Depth = 4 (f) Depth = 5

Figure 12: Splitting steps of a single node, showing the data, covariance ellipses and Information Gain of the parent node for dataset 2

Figure 13 shows the Gaussian PDF distribution of the corresponding leaf nodes on a regular grid according to a single tree and according to a Density Forest of 20 trees.

(a) Density Tree (D1) (b) Density Tree (D2) (c) Density Tree (D3)

(d) Density Forest (D1) (e) Density Forest (D2) (f) Density Forest (D3)

Figure 13: Gaussian PDF distribution according to single tree (top row) and according to a Density Forest of 20 trees (bottom row)

While the single tree in figure 13 still clearly reflects the ellipsoids visible in figure 11, Density Forest show smoother probability distributions. In particularly densely concentrated regions, such as near the central parts of the spiral arms in dataset 2, darker regions represent regions of higher density, due to smaller Gaussians being fitted to theses regions.

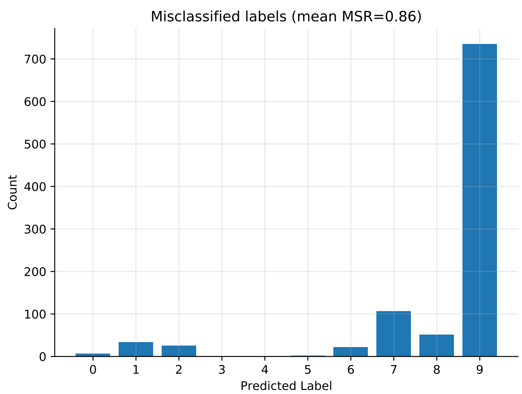
6.3 MNIST Dataset

Using the CNN architecture listed in table 2, 10 models were trained on all training data except for images belonging to the left-out class. The accuracy of each network was evaluated on training and test points belonging to the seen classes (table 6). Since classes are balanced in the MNIST dataset, only the Overall Accuracy, which is almost identical to Average Accuracy in this case, is indicated. Detailed accuracy measures for each left-out class are shown in appendix D.

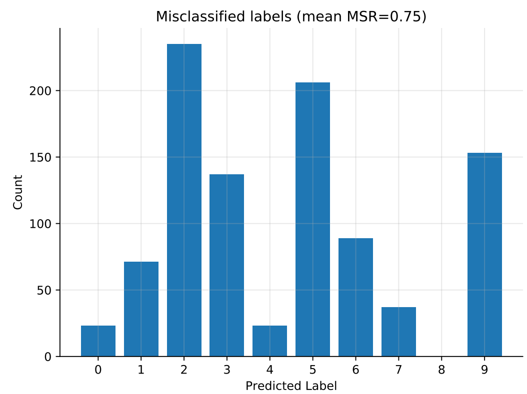
Training set	Test set
99.34	99.03

Table 6: Mean training and test set OA in % for the CNN models trained on $N-1$ classes

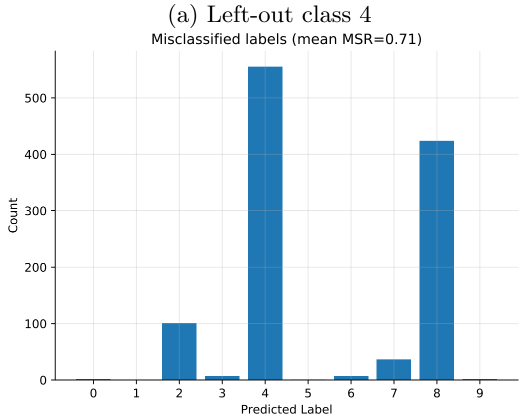
First, predictions were made for the unseen class to inspect the outcome distribution (fig. 14). Some left-out classes are mispredicted more homogeneously than others and have a higher mean MSR. A possible explanation could be the visual similarity between certain digits, such as 4 and 9, which may look similar whereas no digit clearly resembles the digit 8. It is important to notice that higher intrinsic similarity between classes can make the task of novelty detection harder. Prediction counts for all left-out classes are shown in appendix F.1.



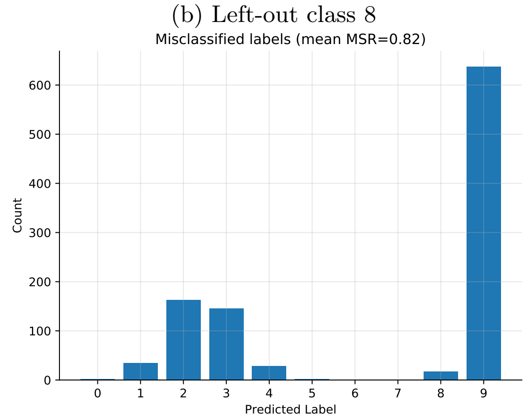
(a) Left-out class 4



(b) Left-out class 8



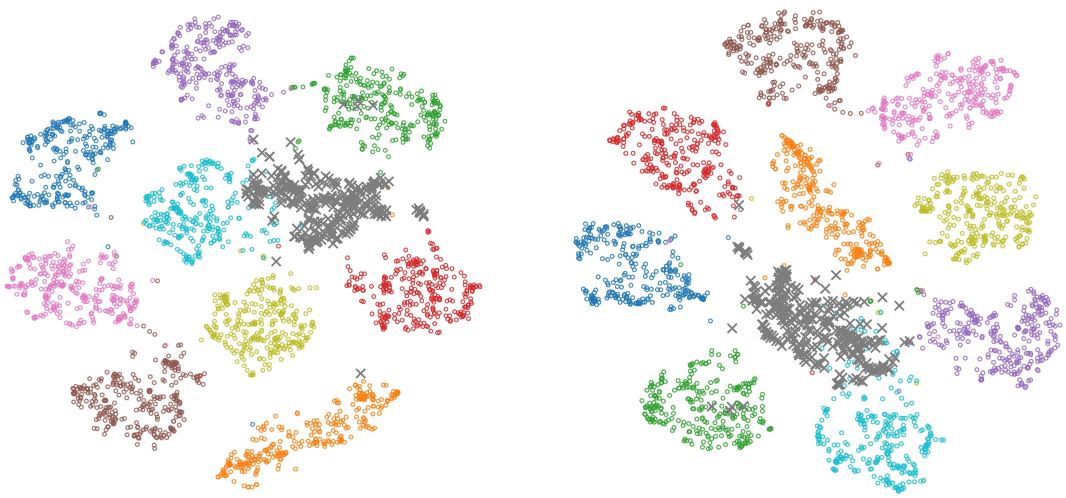
(c) Left-out class 1



(d) Left-out class 7

Figure 14: Predicted labels for networks with left-out classes 4 and 8 (top) and 1 and 7 (bottom). While digits showing a 4 are mostly mislabeled as a 9, the digit 8 is mislabeled less homogeneously. While one could suppose that digits 1 and 7 look similar and might be confused, digits 7 are mostly classified as digit 9, and digit 1 mostly as digits 4 and 8.

Second, ReLu activations of the Fully Connected layer were extracted as described in section 5.4. For each image, a vector of 128 activations was retrieved. PCA was applied to this vector in order to reduce the redundancy of the filters. For the activations before and after PCA, data separability was checked using t-SNE visualizations (fig. 15). The t-SNE visualizations in all cases looked similar before and after PCA, and are therefore only presented for one of the 10 trained models. t-SNE visualizations for activations of each left-out class after PCA are shown in appendix F.



(a) t-SNE before PCA

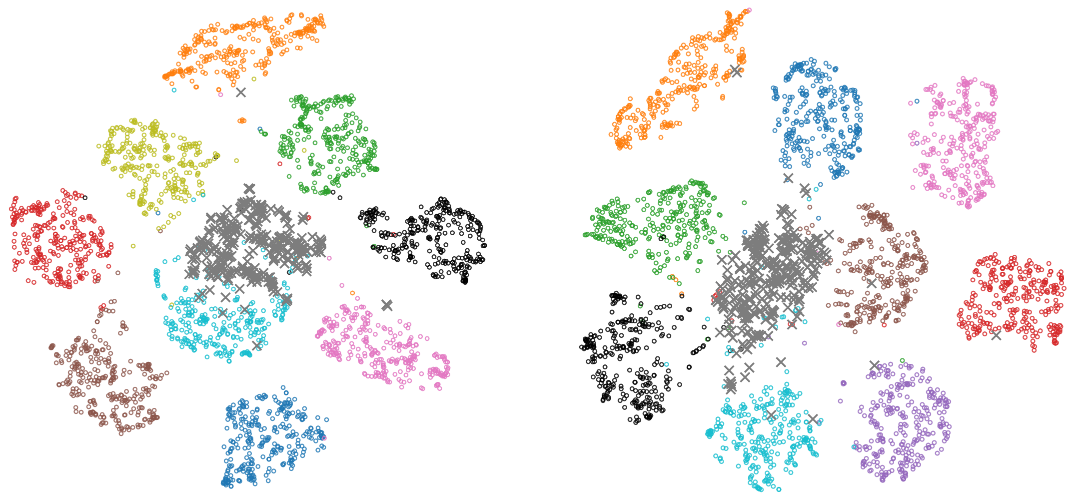
(b) t-SNE after PCA

CLASS: • 0 • 1 • 2 • 3 • 4 • 5 • 6 • 7 • 8 • 9
× Unseen class

Figure 15: t-SNE of MNIST activations before and after PCA for the model with left-out class 7. Both before and after PCA, activations of different classes seem well separable, including the unseen class.

Figure 15 shows that the t-SNE visualizations of the network activations before and after PCA look almost identical. Note that the rotation of the plot before and after applying PCA is irrelevant, since the lower-dimensional data representation yielded by t-SNE only models the pairwise proximities between data points and not their absolute position in space. For novelty detection methods to work well, the unseen class should be well-separated from the other, seen classes. The t-SNE visualization indicates that all classes are well-separated, including the unseen class.

The tendencies with respect to the class confusion shown in figure 14 can also be observed in the t-SNE visualizations (figure 16): while class 4 tends to be confused mainly with class 9, class 8 is confused with several clusters. Unseen class 8 seems to be slightly better separable than unseen class 4, which is very close to the activations of class 9.



(a) Class 4

(b) Class 8

CLASS: • 0 • 1 • 2 • 3 • 4 • 5 • 6 • 7 • 8 • 9
× Unseen class

Figure 16: t-SNE of MNIST activations, after PCA transformations.

After retrieving the activations of all data points and applying PCA dimensionality reduction, parameter search has been performed for GMM, OC-SVM and Density Forests. Best hyperparameters found for each left-out class and novelty detection method are listed in table H.1.

AUROC metrics for the Density Forest method and baseline methods are indicated in table 7. Metrics for each left-out class are indicated in appendix table D.2.

MSR	Margin	Entropy	MC-Dropout	GMM	OC-SVM	DF
0.97	**0.97**	**0.97**	0.96	0.67	0.75	0.75

Table 7: Mean AUROC for each left-out class in the MNIST dataset

In the MNIST dataset, GMM, OC-SVM and DF perform worse compared to the baseline methods MSR, margin, entropy and MC-Dropout. Reasons for the lower performance of the pre-softmax based novelty detection methods are discussed in section 7.

6.4 Zurich Dataset

6.4.1 Overall Results

Since class imbalance is greater than in the MNIST dataset, model performance is indicated for each left-out class separately in table 9. For comparison, accuracy metrics for a CNN trained on all classes is given in table 8. In this section, illustrations are shown for the model trained on all classes but “roads” and the equivalent figures for the other models are shown in appendix G.

Class	Precision [%]	Recall [%]	F1 Score [%]
Roads	83.63	69.77	76.07
Buildings	64.93	88.00	74.72
Trees	79.80	82.69	81.22
Grass	94.06	67.52	78.61
Bare Soil	48.04	71.81	57.57
Water	97.94	91.41	94.56
Railways	0.01	0.01	0.01
Swimming Pools	84.75	89.80	87.20
Average	69.15	70.13	68.75

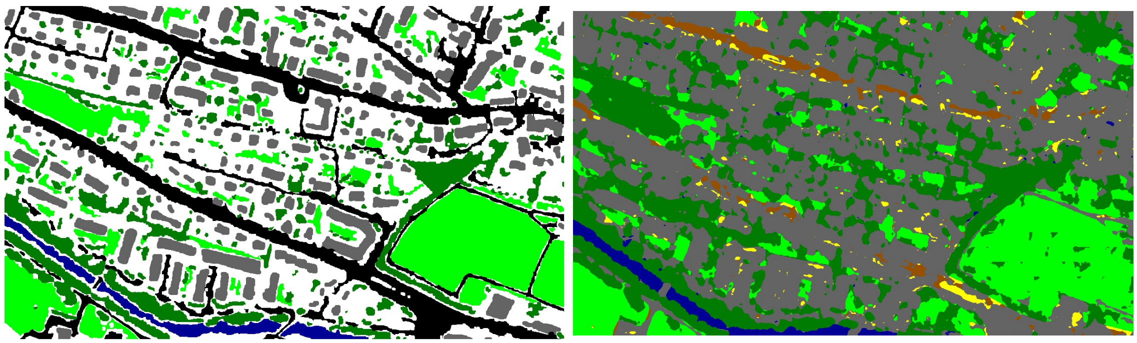
Table 8: Test set accuracy for the U-Net CNN trained on all classes (Overall Accuracy: 77.59 %)

The low accuracy for the class “railways” could be due to the low number of class samples (fig. 7), to the visual similarity with the “roads” class or to suboptimal network parameters and architecture.

	Training set		Validation set		Test set	
Left-out Class	OA [%]	AA [%]	OA [%]	AA [%]	OA [%]	AA [%]
Roads	71.33	65.09	91.79	80.39	87.92	76.39
Buildings	68.12	62.50	89.44	77.93	83.94	69.38
Trees	65.36	58.84	88.79	72.59	88.58	71.40
Grass	57.27	54.49	79.55	68.15	81.49	69.23
Bare Soil	59.41	53.97	80.43	72.28	80.49	71.01
Water	65.37	62.90	85.73	71.83	84.04	68.31
Railways	62.37	57.96	82.55	71.02	82.11	71.09
Swimming Pools	59.56	57.72	80.55	70.50	82.34	71.34
Mean	63.60	59.18	84.85	73.09	83.86	71.02

Table 9: Accuracy measures for the U-Net CNN trained on $N-1$ classes.

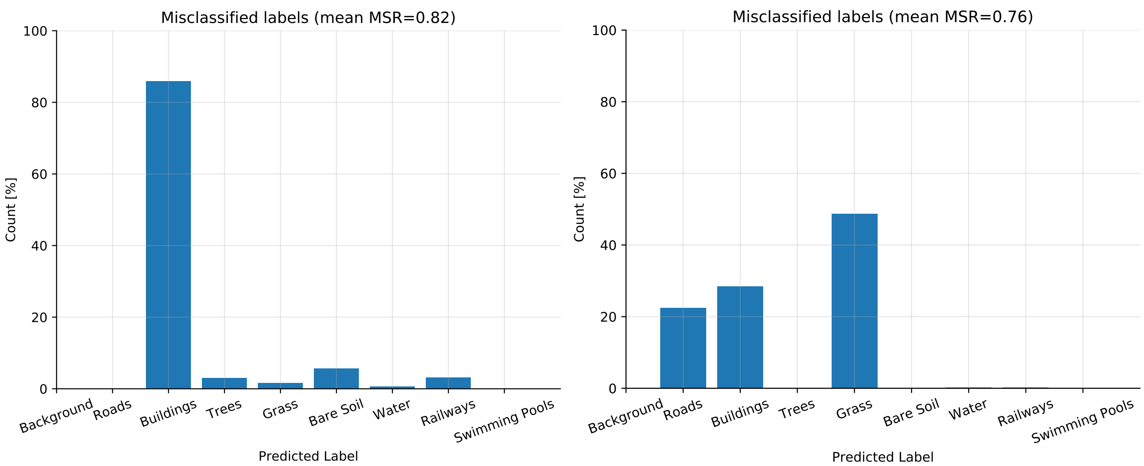
A sample visualization of a prediction from a model with the left-out class “roads” is shown in figure 17. Most pixels annotated as “road” in the ground truth are either classified as “buildings”, “bare soil” or “railways”.



(a) Ground Truth (b) Prediction

Figure 17: Prediction and Ground Truth for the model trained without the roads class

As for the MNIST dataset, misclassification distributions were checked for all models (fig. 18). Again, more homogeneously misclassified class samples are correlated with higher mean MSR. Prediction counts for each left-out class are shown in appendix G.1

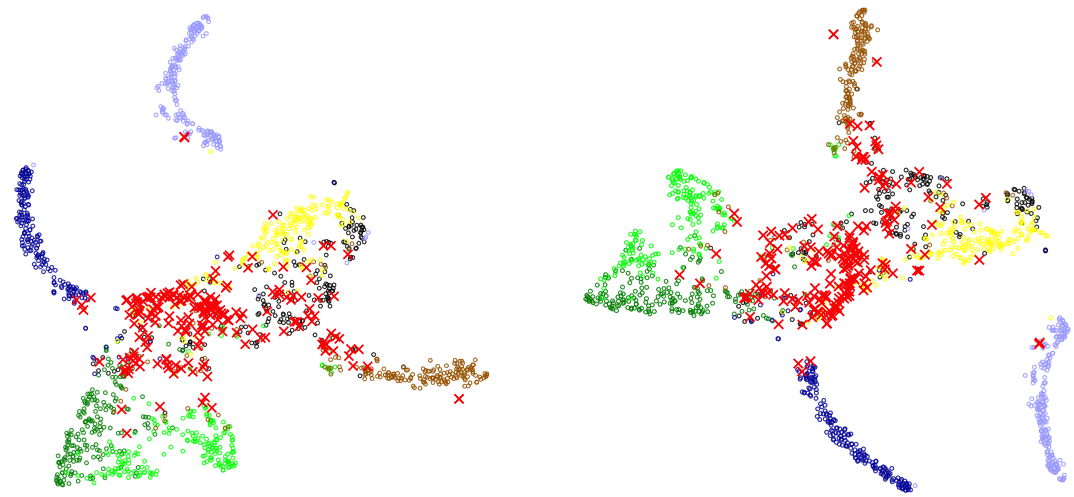


(a) Left-out class “Roads” (b) Left-out class “Trees”

Figure 18: Predictions for network with left-out class “roads” and “trees”. While roads are mostly mislabelled as buildings, trees are mislabelled less homogeneously.

After network training, activations of all images in the training, validation and test sets were retrieved and PCA was applied to reduce their dimensionality. The cumulative variance explained by each additional component is shown for the model with the left-out class “roads” in figure 9b. As figure 9 shows, less components are needed to explain a high percentage of the variance compared to the MNIST dataset.

Similar to the MNIST dataset, it was checked by t-SNE visualization if the pre-softmax activations are separable, both before and after PCA (fig. 19). Ideally, both t-SNE plots should show the same relative arrangement between classes. While clusters may be of different sizes, they should be separated according to the classes for a well-enough trained network.



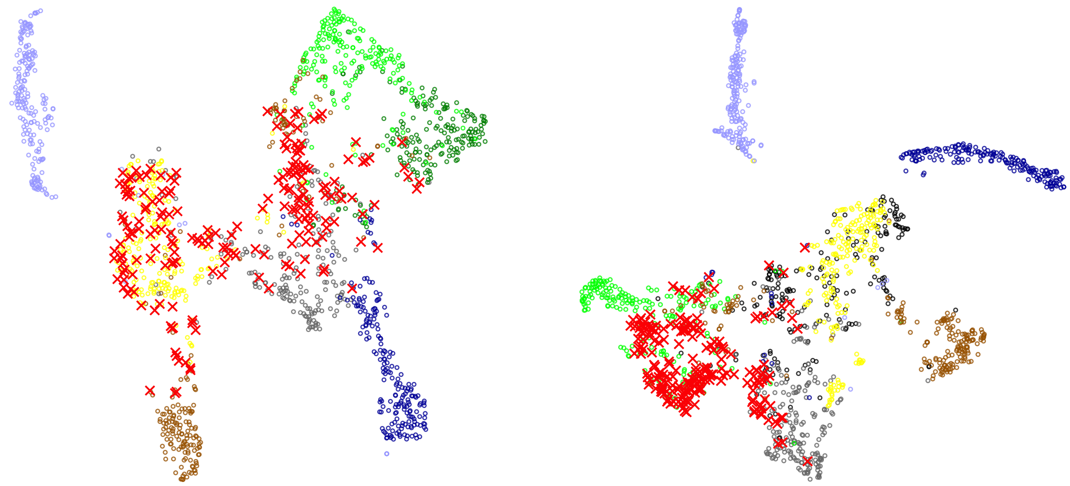
(a) t-SNE before PCA

(b) t-SNE after PCA

Class: ● Roads ● Buildings ● Trees ● Grass
● Bare Soil ● Water ● Railways ● Pools
× Unseen class

Figure 19: t-SNE of Zurich dataset activations, model with left-out class buildings. The same number of points are shown by class, although the real class distribution is imbalanced (figure 7).

Fig 19 shows that t-SNE visualizations before and after PCA look almost the same. The unseen class "buildings" is confused most strongly with the class "roads" and partly with other classes. t-SNE visualizations of the classes "roads" and "trees" are shown in figure 20.



(a) Roads (b) Trees

Class: • Roads • Buildings • Trees • Grass • Bare Soil • Water • Railways • Pools × Unseen class

Figure 20: t-SNE of Zurich dataset activations after PCA for left-out classes "Roads" and "Trees".

Although there seems to be a confusion between the classes "roads" and "railways" which suggests difficulties to separate both classes, the class "railways" is very rare (fig. 7). It is expected that Density Trees mainly fit ellipses in high-density regions containing many points. Therefore, it is likely that a Density Forest with limited depth will fit few leaf nodes directly to the "railways" class. Figure G.3 in appendix shows t-SNE visualizations of the activations after PCA for all left-out classes and for the model trained on all classes. While some classes are separable in most cases, such as bare soil, water and swimming pools, some classes are more mixed together, such as buildings and roads, or grass and trees. It is also noteworthy that some classes are only separable from others when seen during training, such as the class "water", while some also remain separable when unseen, such as the class "swimming pools".

Best parameters for each left-out class model and novelty detection method are shown in appendix table H.2. AUROC metrics for each left-out class and novelty detection method are shown in table 10.

Left-Out Class	MSR	Margin	Entropy	MC-Dropout	GMM	OC-SVM	DF
Roads	0.60	0.59	0.61	0.59	0.66	0.52	**0.70**
Buildings	0.65	**0.66**	0.65	0.65	0.58	0.62	0.61
Trees	0.74	0.74	0.74	0.75	0.66	**0.78**	0.55
Grass	0.38	0.37	0.39	0.39	0.47	0.24	**0.55**
Bare Soil	0.68	**0.70**	0.63	0.59	0.66	0.67	0.65
Water	0.59	0.60	0.58	0.58	0.59	**0.79**	0.66
Railways	0.57	0.59	0.53	0.54	0.55	**0.69**	0.40
Swimming Pools	0.26	0.28	0.23	0.28	**0.99**	0.97	**0.99**
Average	0.56	0.57	0.55	0.54	0.65	**0.66**	0.64

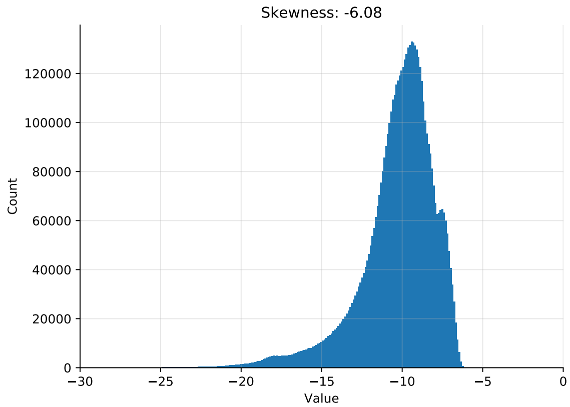
Table 10: AUROC for each left-out class and novelty detection method

Table 10 indicate that the pre-softmax-based methods outperform other softmax-based novelty detection methods in 6 of 8 classes, with OC-SVM and DF showing the highest performance. On average, the increase in AUROC from softmax-based methods to pre-softmax based methods corresponds to about 0.1, or an increase of 18%. DF clearly outperforms other methods on the classes "roads" and "grass". Pre-softmax-based methods most clearly outperform softmax-based methods on the swimming pools class. Visual results of the different methods are shown in appendices G.7 - G.8. ROC curves used to produce the AUROC metrics of table 10 are shown in appendix G.15.

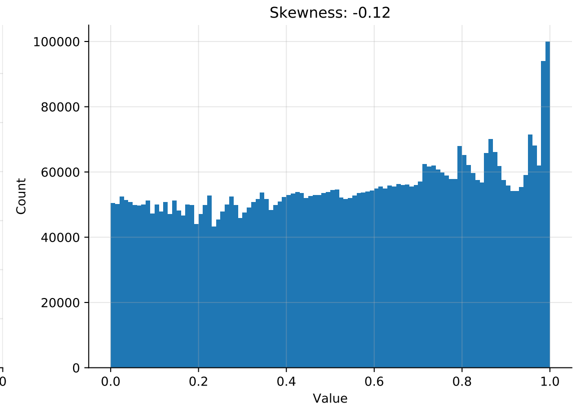
6.4.2 Visual Interpretation

To illustrate the performance of the novelty detection methods, confidence images for individual left-out classes can be visualized and analyzed. In this section, results are compared between MSR and Density Forests according to the classes on which the Density Forest performs better or worse than other novelty detection methods. Figures for all methods and left-out classes are shown in figures G.7 - G.14.

A property of novelty detection methods is that they attribute extreme values to outliers, leading to skewed distributions of the confidence values. This can be observed in the results of the methods GMM, OC-SVM and DF. In the case of novelty detection, some particular objects of the image may be recognized as very uncertain, leading to lesser visibility of confidence differences in the rest of the image. Therefore, histogram equalization was applied to the confidence images of GMM, OC-SVM and DF to enhance the overall image contrast and show more local differences between classes [11] (fig. 21). Individual objects with very high uncertainty visible in the original images are discussed in section 6.4.3.

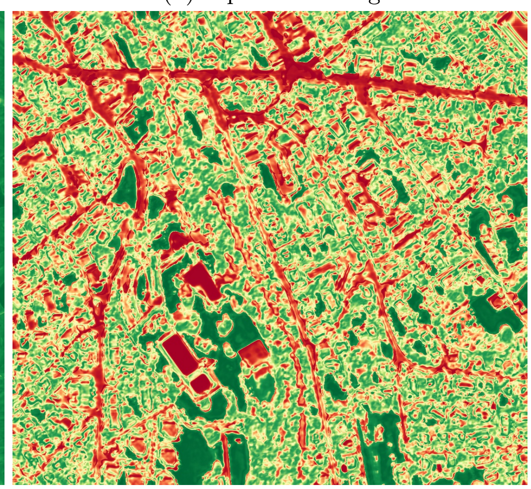


(a) Original histogram



(b) Equalized histogram

(c) Original confidence image



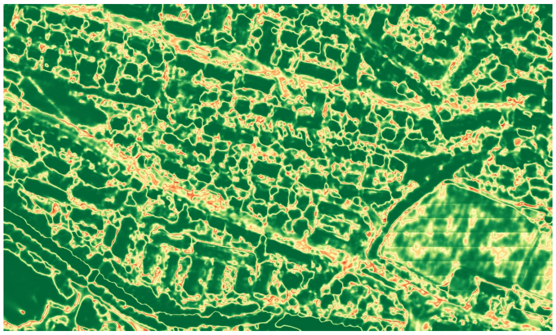
(d) Equalized confidence image

Certainty

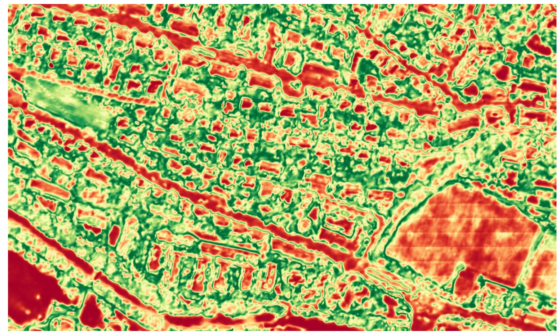
Low  High

Figure 21: Original and equalized confidence distributions for DF, using the left-out class "Roads". While outliers are visible in the original figure, smaller differences in confidence values within classes are better visible after histogram equalization.

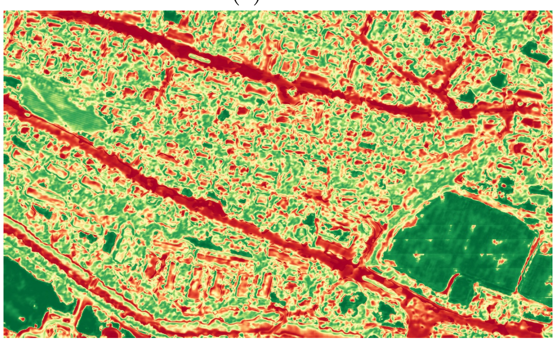
Figure 22 shows visual results for the class "roads". Density Forest here outperforms the other methods, confirming visually that it has the highest AUROC values of all confidence estimation methods (table 10). The unseen class "Roads" clearly appears to be the least certain, while for MSR, the roads seem as uncertain as other classes. MSR just like margin and entropy, the other baseline methods based on the softmax output, mainly show low certainty along class changes (figure G.7). While OC-SVM highlights the roads as uncertain, it also recognizes many other objects as uncertain, belonging to the seen classes buildings, grass and water.



(a) MSR

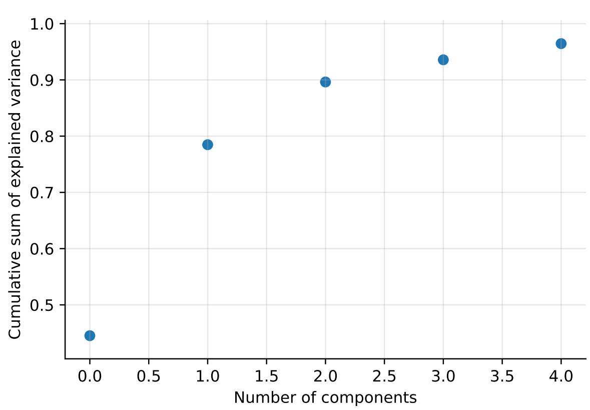


(b) OC-SVM

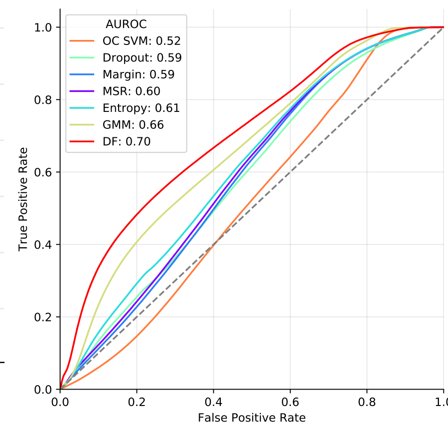


(c) DF

(d) Ground Truth



(e) Cumulative variance explained by PCA components



(f) ROC curve

CONFIDENCE

Low High

GROUND TRUTH

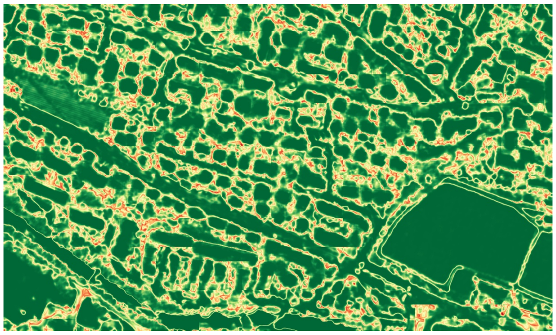
Background Roads Buildings Trees Grass Bare Soil Water Railways Swimming Pools

Figure 22: Visual uncertainty results for selected methods on left-out class "roads" and corresponding ground truth. Contrast stretching and histogram equalization have been applied to OC-SVM and DF images for better visibility. Cumulative variance explained per PCA components and ROC curves are shown below the confidence images.

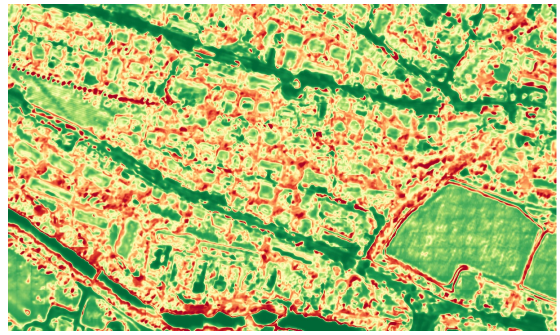
Figure 23 shows a class for which the Density Forest underperforms. In this case, OC-SVM shows the best performance in detecting the unseen class “trees” (along the river and on the top-left side of the big grass field). While the winning method OC-SVM and DF both show low confidence in the regions containing trees, the Density Forest in addition shows very low confidence along the river and along some parts of the roads.

For both left-out classes roads and trees, a low number of PCA components are needed to explain the initial variance of activations. Visual results, explained variance by PCA components and ROC curves are provided for all left-out classes in appendix G.

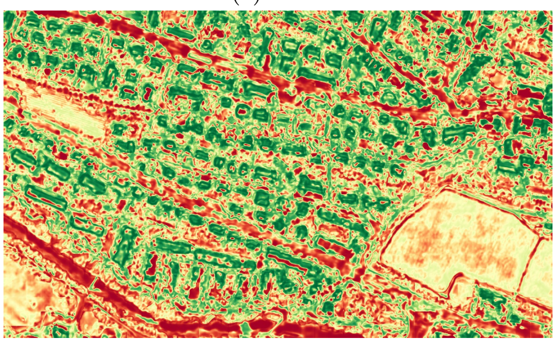
Figure 24 shows the t-SNE points colored by confidence values of different novelty detection methods for the left-out class “trees”. All points with a solid border belong to the unseen class. Ideally, those points should be colored red (uncertain) while points of other classes should be yellow or green. Figure 24 also shows different behavior in the seen classes between novelty detection methods. Softmax-based methods MSR, margin, entropy and MC-Dropout all behave similarly and identify the regions of lowest confidence at the borders of clusters and within the unseen class. The pre-softmax based confidence measures GMM, OC-SVM and DF behave similarly and generally attribute a higher confidence to the cluster centers and unseen class, while GMM and DF also attribute lower certainty to points of the class “bare soil’ and identify the class “swimming pool” as particularly uncertain, which could indicate their ability better identify heterogeneous classes, such as “bare soil”, and classes with few samples, such as “swimming pools”.



(a) MSR

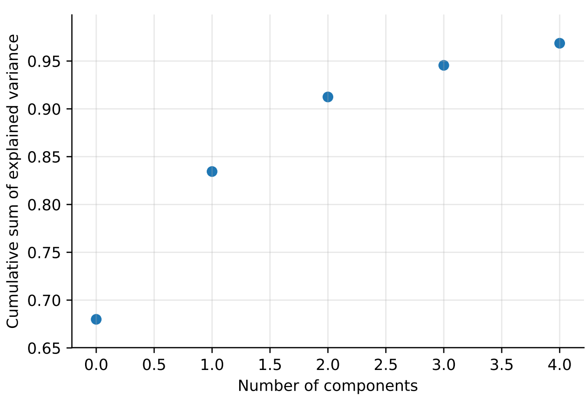


(b) OC-SVM

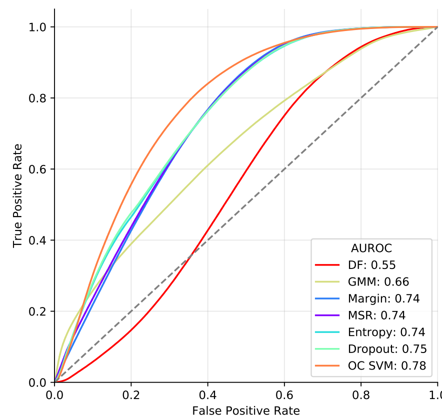


(c) DF

(d) Ground Truth



(e) Cumulative variance explained by PCA components



(f) ROC curve

CONFIDENCE

Low High

GROUND TRUTH

Background Roads Buildings Trees Grass Bare Soil Water Railways Swimming Pools

Figure 23: Visual uncertainty results for selected methods on left-out class "trees" and corresponding ground truth. Contrast stretching and histogram equalization have been applied to OC-SVM and DF images for better visibility. Cumulative variance explained per PCA components and ROC curves are shown below the confidence images.

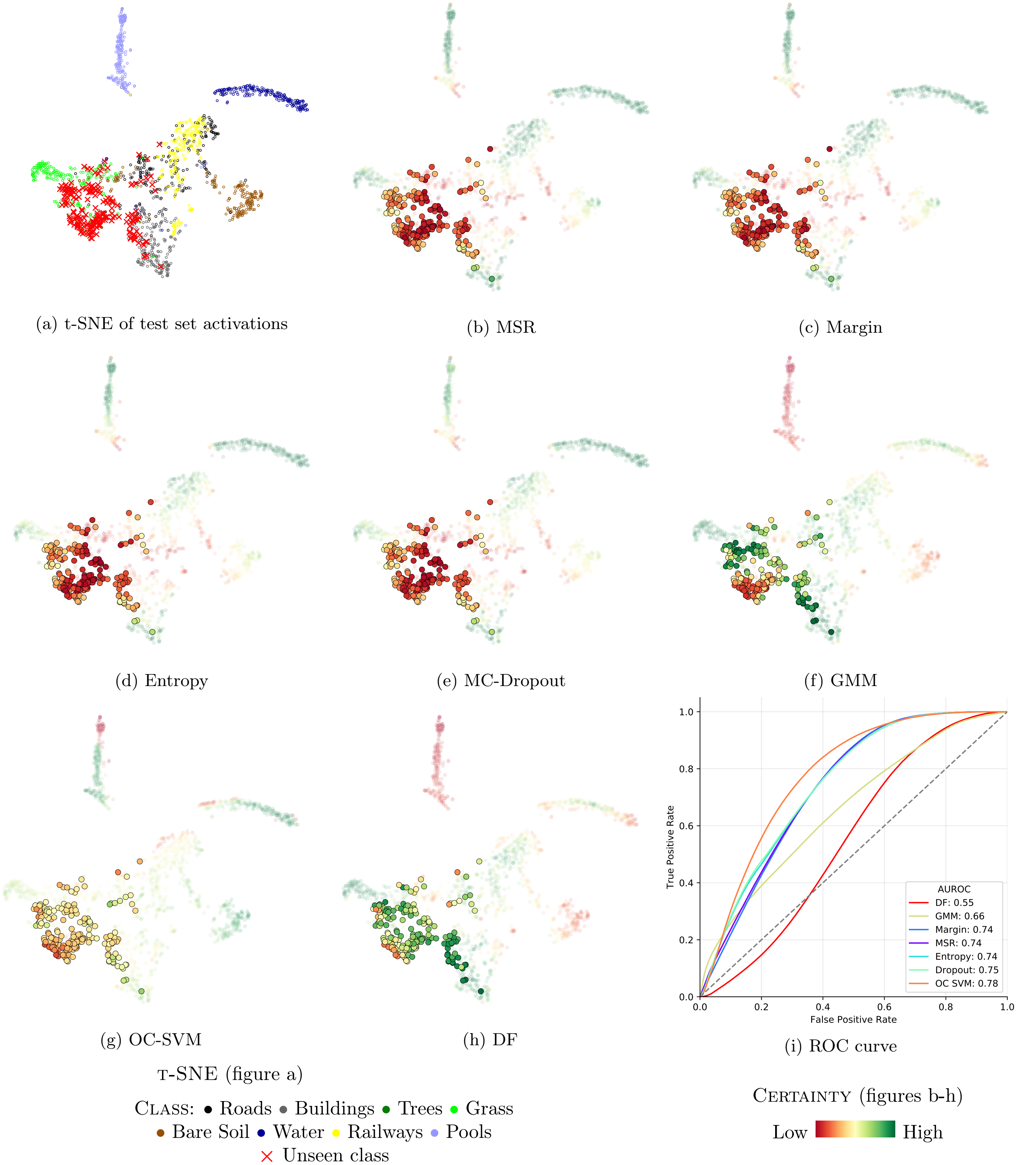


Figure 24: t-SNE visualizations colored according to scores of various novelty detection methods for the left-out class "Trees". Points belonging to the unseen classes are indicated with a solid-edge circle. Ideally, all solid-edge circles should be red, and all other points green to yellow. The original t-SNE plot of the test set activations after PCA and the ROC curves for each method are shown for comparison.

Although figure 24 suggests that softmax-based methods MSR, margin, entropy and MC-Dropout work best for the unseen class “trees”, this is not confirmed by the ROC curve in subfigure 24i, showing that the best-performing method is OC-SVM. The reason for this visual mismatch could be that OC-SVM attributes the lowest confidence values partly to samples of the seen classes swimming pools and water. While on the t-SNE figures, the same number of points is shown per class, in reality there are very few samples for the swimming pool class (fig. 7). If the confidence values were shown on a t-SNE plot with a points proportional to the real class distribution, confidence values of OC-SVM after histogram stretching would indicate lower confidence values compared to softmax-based methods. However, In that case the swimming pool class as well as the classes “railways” and “bare soil” would barely appear on the t-SNE visualization.

6.4.3 Particular Objects

In addition to analyzing the performance of Density Forest and baseline methods in novelty detection, a few particular objects can be distinguished on the test images for which the pre-softmax activations-based methods GMM, OC-SVM and Density Forest all show particularly low confidence. In the following, confidence maps of MSR and Density Forests are shown for particular objects within which Density Forests indicate low confidence. GMM or OC-SVM mostly yield confidence values similar to those of Density Forests for these objects and are shown in appendix G.

Figure 25 shows a particular Region of Interest (ROI) annotated as a swimming pool, containing visible swimming lanes. While MSR attributes a fairly high confidence to this region, again just highlighting the class borders, DF indicates very low confidence values within this object.

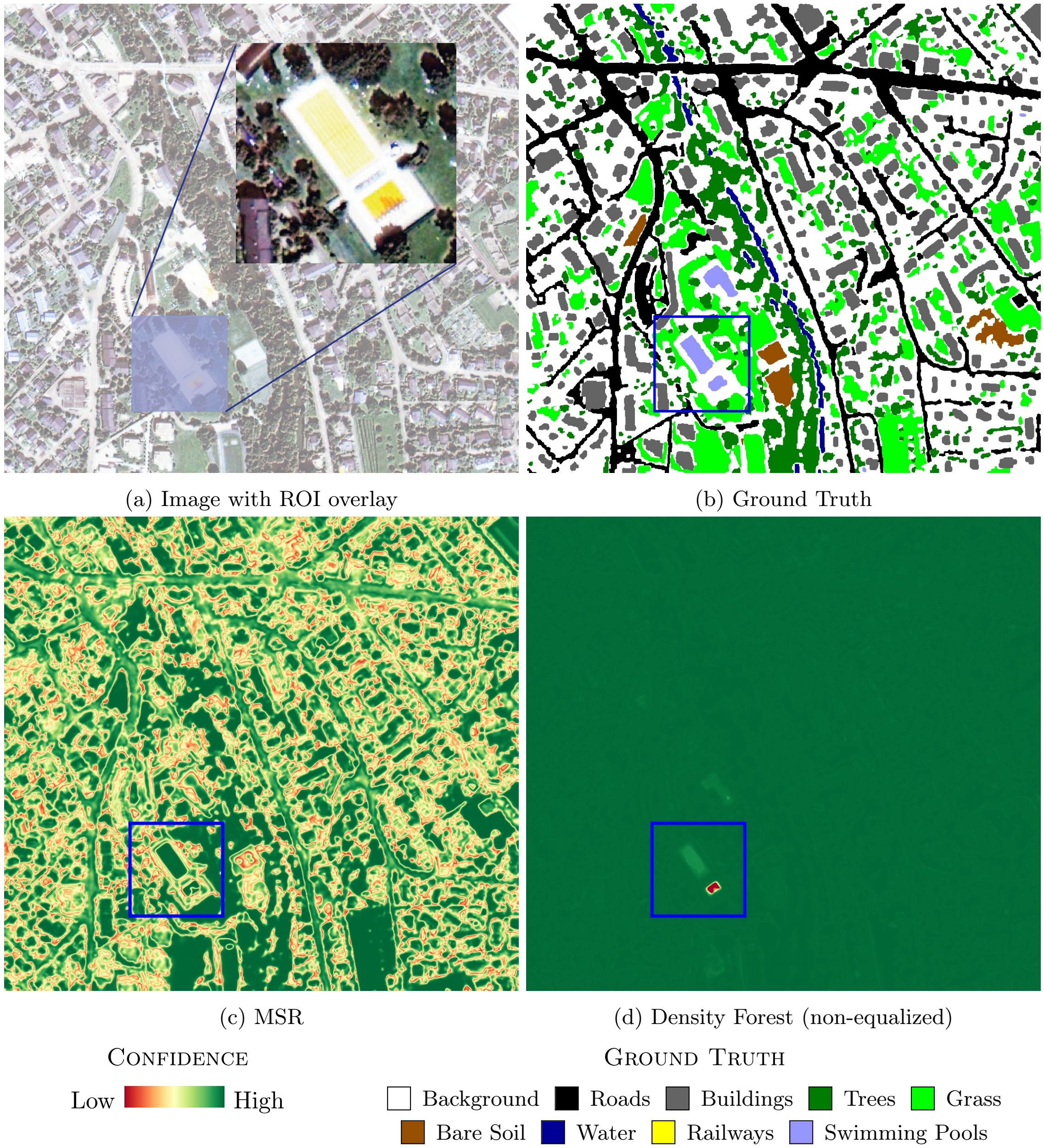
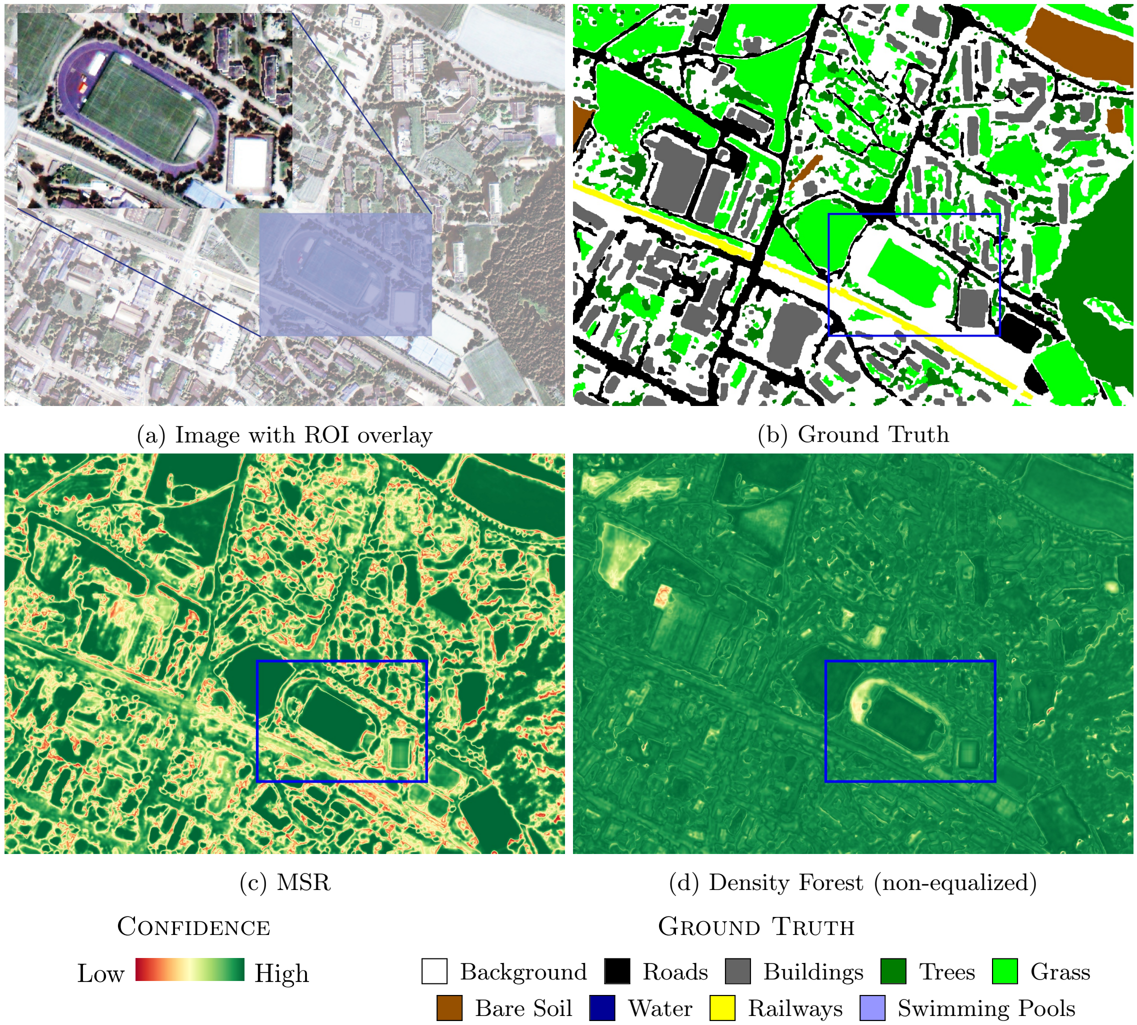


Figure 25: Swimming pool object with MSR and DF confidence images

Figure 26 shows another region around a soccer pitch for which DF yields low confidence, seemingly a race track.



(a) Image with ROI overlay (b) Ground Truth (c) MSR (d) Density Forest (non-equalized)

Figure 26: Soccer pitch object with MSR and DF confidence scores

For the swimming pool in figure 25, the reason for the varying confidence between MSR and Density Forest could be the number of training samples for the swimming pool class (fig. 7). For the race track in figure 26, there is no corresponding land cover class, explaining the lower confidence of the Density Forest.

Generally, GMM, OC-SVM and DF attribute very low values to particular objects whose activations seem very different from those of seen classes or for objects belonging to very infrequent classes. For the class “swimming pools”, all three methods strongly outperform softmax-based confidence estimators (table 10). Since the class “swimming pool” is very rare, this might indicate that these methods work best for outlier detection, where the novelties are very few and different from the “normal” distribution.

7 Discussion

The following discussion is structured by types of novelty detection methods which are either based on softmax activations or on pre-softmax activations.

7.1 Softmax-Based Methods

Overall Performance Softmax-based methods show a consistently high performance in the MNIST dataset with an average AUROC of above 90% (table 7). For the Zurich dataset, the performance of the softmax-based measures is less consistent with an average AUROC of 54-57 %, being just slightly better than a predictor making random guesses based on the novelty frequency, which would correspond to an AUROC of around 50% (table 10). The deficiencies of softmax-based measures are also visible on the confidence images of the Zurich dataset, showing low confidence values mainly along class changes (appendix G).

Similarities between Softmax-Based Methods In most cases, the softmax-based confidence measures MSR, margin, entropy and MC-Dropout all show very similar results. In the MNIST dataset, the performance of these classifiers is almost identical (table 7) while for the Zurich dataset, performance differences between softmax-based methods are on the order of 1-2 percents (table 10). This indicates that taking into account the full softmax activation vector does not necessarily improve the novelty detection performance compared to simple MSR. With respect to MC-Dropout, part of the similarity to other softmax-based methods could be due to the simplified version which was used in this study, using dropout in test time only after the FC layer and with a dropout probability of 0.1. A higher dropout probability should be defined and compared to standard MC-Dropout in future studies.

7.2 Pre-Softmax-Based Methods

Overall Performance Regarding the MNIST dataset, pre-softmax based methods are consistently outperformed by softmax-based methods, while in the Zurich dataset, Density Forests and other pre-softmax-based confidence estimation methods show on average a significant improvements over softmax-based methods in terms of novelty detection performance. While the performance of the classifiers GMM, OC-SVM and DF is partly inconsistent within left-out classes, this indicates that taking into account the pre-softmax network activations should yield more information than just estimating the confidence based on the softmax output. Yet, the performance of various novelty detection methods differ strongly between classes. Some of the reasons for this varying performance are discussed below, including network architectures, different lengths of feature vectors yielded by the pre-softmax activation functions, network accuracy, input data types, class imbalance and class confusion, as well as differences between kernel-based and forest-based classifiers.

FC Features and the Curse of Dimensionality First, the MNIST and Zurich datasets both use different network architectures which yield a different number of features in the FC layer. While the MNIST CNN (table 2) has 128 filters in the FC layer, the U-Net used for the Zurich dataset has only 32 filters in the corresponding layer before the classification layer. The number of filters in the FC layer strongly correlates with the number of PCA components needed to explain the variance of the full feature vector (figure 9). To explain 95% of the variance, only 5-10 PCA components are needed for the Zurich dataset while for the MNIST dataset, 30-40 components have to be kept (tables 2 and E.1). Linked to the number of components kept after dimensionality reduction, part of the inconsistent performance of Density Forests as well as GMM could be due to their underlying assumption of a Gaussian distribution in high dimensions. Both methods rely on finding the support of the activations belonging to seen classes through fitting a certain number of Gaussian distributions to the data. In high-dimensional, noisy data, this might be especially difficult due to the curse of dimensionality, making the notion of distance less meaningful in clustering problems [18]. Different dimensionality reduction methods than PCA would be desirable in order to reduce the data to a lower number of dimensions while preserving the structure of the high-dimensional data. The influence of other dimensionality reduction method on the performance of Density Forests should be studied more deeply.

Data Complexity Differences in the network accuracy as well as performance of novelty detection methods between the MNIST and Zurich datasets could be linked to the amount of training data, their complexity and their within-class heterogeneity. Despite the much higher number of parameters trained in the U-Net used for the Zurich dataset, the MNIST networks reach a test accuracy of around 99%, while the U-Net reaches only a lower test set accuracy of about 84% on average (tables 6 and 9). Most importantly, it might be more difficult to separate complex classes such as "roads" and "buildings", which are described by a rich set of spectral and geometric attributes within the objects themselves and in their surroundings, compared to the more simple case of hand-written digits. In addition, land use classes defined by the ground truth might be noisier and less homogeneous, such as samples of various land cover types grouped in the class "bare soil". A further difference to the MNIST dataset may be that some unseen classes in the Zurich dataset systematically combine features of other classes: The swimming pool class might be such an example, where both concrete around the object and water within the object define the class, combining features from both the classes roads and water.The fact that activations of the swimming pool class systematically combine only certain parts of seen classes may give it its distinct position in the feature space.

Despite the lower network accuracy, the increased complexity of classes might provide the network with more features to accurately describe and thus separate each class. t-SNE may not show the distances between classes accurately since it focuses on preserving distances between nearby data points rather than distances between points far away from each other, therefore the distances between well-separated data clusters may be meaningless [28, 55]. Due to these reasons, the complexity and heterogeneity of land use classes makes it seem plausible that activations of different classes in the Zurich dataset are further away from each other compared to activations of different classes of hand-written digits.

Class Imbalance and Class Separability Part of the inconsistency of pre-softmax methods might be due to the distribution and frequencies of the pre-softmax activations. The t-SNE visualizations show different clustering behavior for the activations of the Zurich dataset than for activations of the MNIST dataset, with some classes in the Zurich dataset being better separated from other classes and some classes being separated only if seen during training (fig. 19). Pre-softmax-based methods seem to work better on class activations which are well-separated from other, frequent classes. Regarding the Zurich dataset, the network accuracy is lower, however some classes are only confused with infrequent other classes, which should be less of a problem, as Density Forests mainly fit clusters to frequent classes. For instance, all pre-softmax-based confidence methods clearly outperform other baselines in the class "swimming pools" (fig. G.14). This might be in part due to the fact that these activations are very well separable from others, even for the model trained without the class "swimming pools" (fig. G.3h). While t-SNE visualizations also indicate good separation of the class water for models trained with the class, the class appears to be less well-separable when not seen during training (fig. G.3f). It is thus to very likely that part of the varying performance between the unseen classes of the Zurich dataset is due to how well the unseen class is separated even if it has not been seen during training.

Kernel Methods and Forest-Based Methods Similarly, although the task of a neural network is to make data linearly separable, a suboptimal network will still have non-linearly separable activations in the FC layer. A possible advantage of algorithms such as OC-SVM could be their ability to increase data separability by explicitly reorganizing non-linearities from non-optimally trained networks, using the kernel trick. Although Density Forests do not explicitly apply any kind of feature augmentation, forest-based methods can model non-linear data distributions through bagging of many weak learners. Optimal hyperparameters for OC-SVM included non-linear kernels in some cases (table H.2). This could be due to an inability of the network to separate the classes related to the high data complexity, too few training samples or to the network structure and parameters. Both kernel-based and forest-based methods require a certain number of hyperparameters to be tuned to correctly model such non-linearities. Which of both methods is better able to model non-linearities can not be seen from the presented results, which show roughly equivalent accuracies between OC-SVM and DF. The relation between the data complexity, network architecture and best OC-SVM kernel for novelty detection should be studied further.

8 Conclusion

Density Forests try to model the density of a complex, potentially non-linear distribution through bagging of weak learners. In the case of CNNs, Density Forests have been used to model the support of activations belonging to seen classes, to detect new, unseen data in the test set. This study has shown that the developed method works in many cases, within certain limitations, and that softmax-based confidence measures are unsuitable for novelty detection in the Zurich dataset. This conclusion lists the main contributions made in this work, its limitations, future research to be done and ends with an outlook on the bigger picture of relevant novelty detection applications in ML.

8.1 Main Contributions

Following contributions have been made in this work:

- Density Forests have been implemented and applied to a synthetic dataset as well as a more complex dataset consisting of high-dimensional neural network activations. Their potential for novelty detection in the MNIST dataset and in the more complex Zurich dataset has been assessed and discussed.
- Importantly, a ready-to-use library for Density Forests was implemented, which can be easily installed using the command `pip install density_forest`. The code, detailed usage instructions, code documentation and illustrations are hosted on GitHub (`https://github.com/CyrilWendl/SIE-Master`), including a variety of methods training Density Trees or Density Forests on a training set and predicting confidence values for a test set. In addition, the library includes a number of generic functions that include:

    1. Perform hyperparameter search to find the best parameter set for training a Density Forest as described in section 5.6.
    2. Generate synthetic datasets as described in section 4.1.
    3. Generic plotting functions to produce all the visualizations of this report.

    The syntax for fitting Density Forests, predicting confidence values, performing hyperparameter search and using auxiliary functions is described in the `README.md` of the GitHub repository.
- This study has aimed to show the potential of using the pre-softmax activation vector to improve novelty detection in trained CNNs. In a more complex, real-world example, this idea has proven successful, however only within the limited scope of the restrictions described below.

8.2 Limitations

While pre-softmax-based methods and Density Forests in particular have shown improvements over softmax-based measures in the Zurich datasets, they only work under a set of limitations.

First, the implemented pre-softmax-based confidence measures require a certain number of hyperparameters to be tuned in order to work. This task can be time-consuming and the correct range of parameters may be difficult to identify. In some cases, this may even be impossible due to a too small dataset, making it difficult to split the dataset into a training, validation and test set of sufficient sizes to perform hyperparameter search. In addition, if the unseen class is very rare, it may not be present in a validation set used to find optimal parameters.

Second, the network structure may play an important role for the performance of both softmax-based and pre-softmax-based confidence measures. A network with many filters in the FC layer may produce redundant and noisy information spread over many filters, resulting in a long activation vector even after PCA. This not only deteriorates the training and prediction time of each model but may also lead to greater difficulties in separating high-dimensional activations due to the curse of dimensionality. In addition, the influence of the network accuracy on the performance of each confidence measure is to be discussed as well as the influence of the data complexity on the novelty detection performance.

Third, Density Forests, GMM and OC-SVM work better for activations that are very different and rare, such as those of the class "swimming pools". Thus, in order to effectively identify unseen points, a special meta-loss would need to be introduced in the network to maximize the distance between activations of different classes.

8.3 Future Research

Further research should go into understanding the reasons for the inconsistent behavior of Density Forests.

Datasets and Architectures To further study the performance of Density Forests, the method should be applied to further datasets and to different network architectures.

Dimensionality Reduction Regarding Density Forests, the influence of the dimensionality on the performance of Density Forests should be evaluated more thoroughly. For instance, different network architectures with a varying number of pre-softmax activations could be trained and compared with respect to the Density Forest performance. Such a comparison has not been implemented because of the explicit assumption that Density Forests should be applicable to many different standard network architectures without modifying them. Yet, there are good reasons to assume that Density Forests will deal better with lower-dimensional activations, related to the curse of dimensionality.

Meta-Loss to Increase Class Separability For similar reasons, contrary to the method proposed by Mandelbaum and Weinshall [29], no special meta-loss has been implemented to maximize the distance between activations belonging to different classes, again because of the assumption that the proposed confidence measure should not require any changes in the network architecture. It would however be interesting to see if there are significant performance gains for any of the pre-softmax-based methods if such a meta-loss was implemented.

Parameter Sensitivity In addition to finding the best parameter set, the sensitivity of each parameter should be assessed. This could potentially reduce the number of hyperparameters and accelerate parameter search.

Further Research Ideas To avoid modeling the distribution of high-dimensional activations, a different idea for a confidence measure would be to work directly with the network activations and weights: In addition to retrieving the activations $\mathbf{x}_i^{(L)}, i \in 1, ..., n$, one could also retrieve the weights $\mathbf{w}_{i,j}^{(L)}$ for a class j just before they are passed to the final softmax activation (fig. 27). By looking at their sign (negative / positive contribution to a class), the degree of agreement between activations could be used as a confidence indicator.

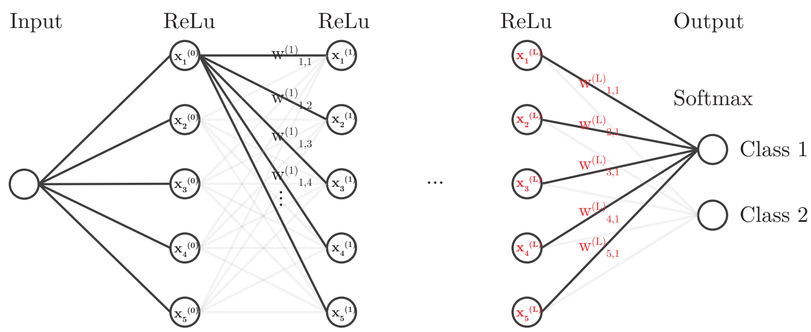


Figure 27: Alternative confidence measure scheme: red parts are to be retrieved and their entropy calculated to measure the degree of agreement of the input to the softmax activation function.

Further, similar ideas to estimate the confidence using activations include maximizing differences between the separating planes produced by each softmax input via a special meta-loss. This way, the network would learn to find as many distinct separations between classes as possible, and thus map class distributions implicitly. However, this again contradicts the basic idea of the proposed methodology, which is to build on an existing network architecture.

Evaluation Any of the analyzed confidence methods are heuristic in nature and are evaluated according to an application-specific goal. The notion of confidence in many cases remains related to particular applications such as error detection or novelty detection. A formal proof of the equivalence between some optimal, pre-softmax activations-based confidence measure and probabilistic uncertainty measures yet has to be provided. Due to their heuristic nature, confidence measures should also be evaluated in applied contexts, such as active learning.

8.4 Outlook

Ultimately, this study has shown a strong potential of Density Forests and pre-softmax confidence measures to improve novelty detection performance without changing the structure of the network. Although less successful in error detection, the methods GMM, OC-SVM and DF in most cases show better performance than the baselines relying on the softmax output. Further research possibilities in this direction should show whether the information contained in the network activations can be used to determine the uncertainty of a network without changing its architecture.

In terms of practical applications, it would be of great help if standard CNNs could provide more reliable confidence measures. Active Learning is an example where novelty detection using confidence values would be very helpful to save time spent by users manually annotating ground truth. By asking users to provide labels for the least certain parts of an image, the accuracy of a CNN could be improved maximally at each iteration and thus time spent annotating images is reduced to a minimum. Such applications could ultimately lead to better land cover maps, helping to recognize spatial developments of global importance such as urbanization, illegal logging, littoralization or other socially and environmentally relevant developments, while spending less time efforts for ground truth annotation and obtaining more accurate results. Further applications of better confidence measures are countless, including medical diagnosis, autonomous driving and weather forecasting. Ultimately, a good model will fail in many situation if it cannot provide a reliable measure of confidence. While this study has aimed at making a contribution to understanding model uncertainty in ML, there are still major questions to be answered in future research.

References

[1] M. Z. Alom, M. Hasan, C. Yakopcic, T. M. Taha, and V. K. Asari. "Recurrent Residual Convolutional Neural Network based on U-Net (R2U-Net) for Medical Image Segmentation". In: *CoRR* abs/1802.06955 (2018).

[2] Y. Bahat and G. Shakhnarovich. "Confidence from Invariance to Image Transformations". In: (Apr. 2018).

[3] A. Beghi, L. Cecchinato, and M. A. Rampazzo. "A One-Class SVM Based Tool for Machine Learning Novelty Detection in HVAC Chiller Systems". In: 2014.

[4] C. M. Bishop. "Novelty Detection and Neural Network Validation". In: 1994.

[5] L. Breiman. "Bagging Predictors". In: *Machine Learning* 24 (1996), pp. 123–140.

[6] L. Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32.

[7] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. "LOF: Identifying Density-Based Local Outliers". In: *SIGMOD Conference*. 2000.

[8] S. Choi, K. Lee, S. Lim, and S. Oh. "Uncertainty-Aware Learning from Demonstration using Mixture Density Networks with Sampling-Free Variance Modeling". In: *CoRR* abs/1709.02249 (2017).

[9] A. Criminisi, E. Konukoglu, and J. Shotton. *Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning.* Tech. rep. 2011. URL: `https://www.microsoft.com/en-us/research/publication/decision-forests-for-classification-regression-density-estimation-manifold-learning-and-semi-supervised-learning/`.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum Likelihood from Incomplete Data via the EM Algorithm". In: 1977.

[11] "Digital Image Processing Third Editionà-p E. Woods". In: 2012.

[12] H. Dong, G. Yang, F. Liu, Y. Mo, and Y. Guo. "Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully Convolutional Networks". In: *MIUA*. 2017.

[13] Y. Gal. "Uncertainty in Deep Learning". PhD thesis. University of Cambridge, 2016.

[14] Y. Gal and Z. Ghahramani. "Dropout as a Bayesian approximation: Insights and Applications". In: *Deep Learning Workshop, ICML*. 2015.

[15] Y. Gal and Z. Ghahramani. "Dropout As a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 1050–1059.

[16] I. J. Goodfellow, J. Shlens, and C. Szegedy. "Explaining and Harnessing Adversarial Examples". In: *CoRR* abs/1412.6572 (2014).

[17] D. Hendrycks and K. Gimpel. "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks". In: *CoRR* abs/1610.02136 (2016). arXiv: `1610.02136`. URL: `http://arxiv.org/abs/1610.02136`.

[18] A. Hinneburg, C. C. Aggarwal, and D. A. Keim. "What Is the Nearest Neighbor in High Dimensional Spaces?" In: *VLDB*. 2000.

[19] A. Hinneburg and D. A. Keim. "Optimal Grid-Clustering: Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering". In: *VLDB*. 1999.

[20] M. Kampffmeyer, A. B. Salberg, and R. Jenssen. "Semantic Segmentation of Small Objects and Modeling of Uncertainty in Urban Remote Sensing Images Using Deep Convolutional

Neural Networks". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2016, pp. 680–688.

[21] M. Kampffmeyer, A.-B. Salberg, and R. Jenssen. "Semantic Segmentation of Small Objects and Modeling of Uncertainty in Urban Remote Sensing Images Using Deep Convolutional Neural Networks". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2016), pp. 680–688.

[22] A. Kendall and Y. Gal. "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" In: *CoRR* abs/1703.04977 (2017). arXiv: `1703.04977`. URL: `http://arxiv.org/abs/1703.04977`.

[23] B. Lakshminarayanan, A. Pritzel, and C. Blundell. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles". In: (Dec. 2016).

[24] Y. LeCun, C. Cortes, and C. Burge. *The MNIST Database of Handwritten Digits*. URL: `http://yann.lecun.com/exdb/mnist/`.

[25] C. Leibig, V. Allken, M. S. Ayhan, P. Berens, and S. Wahl. "Leveraging uncertainty information from deep neural networks for disease detection". In: 7 (Dec. 2017).

[26] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. R. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun. "Towards fully autonomous driving: Systems and algorithms". In: *2011 IEEE Intelligent Vehicles Symposium (IV)* (2011), pp. 163–168.

[27] F. T. Liu, K. M. Ting, and Z.-H. Zhou. "Isolation Forest". In: *2008 Eighth IEEE International Conference on Data Mining* (2008), pp. 413–422.

[28] L. van der Maaten, G. E. Hinton, and Y. Bengio. "Visualizing Data using t-SNE". In: 2008.

[29] A. Mandelbaum and D. Weinshall. "Distance-based Confidence Score for Neural Network Classifiers". In: (Sept. 2017).

[30] M. Markou and S. Singh. "Novelty detection: a review - part 1: statistical approaches". In: *Signal Processing* 83 (2003), pp. 2481–2497.

[31] M. Markou and S. Singh. "Novelty detection: a review - part 2: : neural network based approaches". In: *Signal Processing* 83 (2003), pp. 2499–2521.

[32] D. Marmanis, J. D. Wegner, S. Galliani, K. Schindler, M. Datcu, and U. Stilla. "Semantic Segmentation of Aerial Images with an Ensemble of Cnns". In: 2016.

[33] A. Menderes, A. Erener, and G. Sarp. "Automatic Detection of Damaged Buildings after Earthquake Hazard by Using Remote Sensing and Information Technologies". In: *Procedia Earth and Planetary Science*. World Multidisciplinary Earth Sciences Symposium, WMESS 2015 15 (2015), pp. 257–262. ISSN: 1878-5220. DOI: `10.1016/j.proeps.2015.08.063`. URL: `http://www.sciencedirect.com/science/article/pii/S1878522015003264` (visited on 12/12/2016).

[34] F. de Morsier. "Semi-supervised and unsupervised kernel-based novelty detection with application to remote sensing images". In: (2014).

[35] A. M. Nguyen, J. Yosinski, and J. Clune. "Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images". In: *CoRR* abs/1412.1897 (2014). arXiv: `1412.1897`. URL: `http://arxiv.org/abs/1412.1897`.

[36] W. Ouerghemmi, A. Le-Bris, N. Chehata, and C. Mallet. "A Two-Step Decision Fusion Strategy: Application to Hyperspectral and Multispectral Images for Urban Classification". In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XLII-1-W1. Copernicus GmbH, 2017, pp. 167–174.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[38] M. A. F. Pimentel, D. A. Clifton, L. A. Clifton, and L. Tarassenko. "A review of novelty detection". In: *Signal Processing* 99 (2014), pp. 215–249.

[39] T. Postadjian, A. L. Bris, H. Sahbi, and C. Mallet. "Investigating the Potential of Deep Neural Networks for Large-Scale Classification of Very High Resolution Satellite Images". In: vol. IV-1-W1. Copernicus GmbH, 2017, pp. 183–190. (Visited on 08/17/2017).

[40] D. A. Reynolds. "Gaussian Mixture Models". In: *Encyclopedia of Biometrics*. 2009.

[41] O. Ronneberger, P. Fischer, and T. Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2015, pp. 234–241.

[42] C. Rupprecht, I. Laina, R. DiPietro, and M. Baust. "Learning in an Uncertain World: Representing Ambiguity Through Multiple Hypotheses". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 3611–3620.

[43] J. Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks : the official journal of the International Neural Network Society* 61 (2015), pp. 85–117.

[44] SciPy.org Documentation. *Scipy.org Reference Guide, Statistical functions (scipy.stats)*. 2014. URL: `https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.multivariate_normal.html` (visited on 01/05/2018).

[45] E. Shelhamer, J. Long, and T. Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 3431–3440.

[46] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958.

[47] A. Subramanya, S. Srinivas, and R. V. Babu. "Confidence estimation in Deep Neural networks via density modelling". In: *CoRR* abs/1707.07013 (2017).

[48] R. Sun and C. H. Lampert. "KS(conf): A Light-Weight Test if a ConvNet Operates Outside of Its Specifications". In: *CoRR* abs/1804.04171 (2018).

[49] L. Szymanski and B. McCane. "Visualising kernel spaces". In: 2011.

[50] D. Tuia, E. Pasolli, and W. Emery. "Using active learning to adapt remote sensing image classifiers". In: *Remote Sensing of Environment* 115.9 (2011), pp. 2232–2242. ISSN: 0034-4257. DOI: `https://doi.org/10.1016/j.rse.2011.04.022`. URL: `http://www.sciencedirect.com/science/article/pii/S0034425711001507`.

[51] D. Tuia, M. Volpi, L. Copa, M. F. Kanevski, and J. Muñoz-Marí. "A Survey of Active Learning Algorithms for Supervised Remote Sensing Image Classification". In: *IEEE Journal of Selected Topics in Signal Processing* 5 (2011), pp. 606–617.

[52] M. Volpi and V. Ferrari. "Semantic segmentation of urban scenes by learning local class interactions". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2015), pp. 1–9.

[53] M. Volpi and D. Tuia. "Dense semantic labeling of sub-decimeter resolution images with convolutional neural networks". In: *IEEE Trans. Geoscience and Remote Sensing* 55 (2017), pp. 881–893.

[54] Y. Wang, J. Wong, and A. Miner. “Anomaly intrusion detection using one class SVM”. In: *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.* (2004), pp. 358–364.

[55] M. Wattenberg, F. Viégas, and I. Johnson. “How to Use t-SNE Effectively”. In: *Distill* (2016). DOI: 10.23915/distill.00002. URL: http://distill.pub/2016/misread-tsne.

[56] C. Wendl. *Confidence Estimation for Convolutional Neural Networks using Density Trees*. 2017. URL: https://github.com/CyrilWendl/SIE-Project (visited on 05/15/2018).

[57] J. A. Womble, B. J. Adams, and K. C. Mehta. “Automated Building Damage Assessment Using Remote-Sensing Imagery”. In: *Proceedings of the ASCE/SEI Structures Congress*. 2007. URL: http://ascelibrary.org/doi/abs/10.1061/40943(250)8 (visited on 01/01/2017).

[58] H. Zaragoza and F. d’Alché-Buc. “Confidence Measures for Neural Network Classifiers”. In: *Proceedings of the Seventh Int. Conf. Information Processing and Management of Uncertainty in Knowlegdege-Based Systems (IPMU)*. Vol. 1. Editions EDK (Paris), Jan. 1998, pp. 886–893. URL: https://www.microsoft.com/en-us/research/publication/confidence-measures-for-neural-network-classifiers/.

[59] X. xiang Zhu, D. Tuia, L. Mou, G.-S. Xia, L. Zhang, F. Xu, and F. Fraundorfer. “Deep learning in remote sensing: a review”. In: *CoRR* abs/1710.03959 (2017).

[60] A. M. Zoubir and D. R. Iskander. “Bootstrap Methods and Applications”. In: *IEEE Signal Processing Magazine* 24 (2007), pp. 10–19.

9 Appendices

A Code Documentation

The implementation of the Density Forest code can be found on GitHub: `https://github.com/CyrilWendl/SIE-Master`. Jupyter notebooks are available to illustrate results of the novelty detection methods applied to all datasets:

1. Synthetic dataset: `/Code/Density Forest.ipynb`
2. MNIST dataset: `/MNIST/MNIST Novelty Detection.ipynb`
3. Zurich dataset: `/Zurich/Zurich Novelty Detection.ipynb`

B Random Forest

The result of a Decision Tree with no stopping criterion applied to synthetic labelled data is shown in figure B.1. The ruggedness of the decision boundaries is in part due to visualizing the decision boundaries on a discrete coordinate mesh.

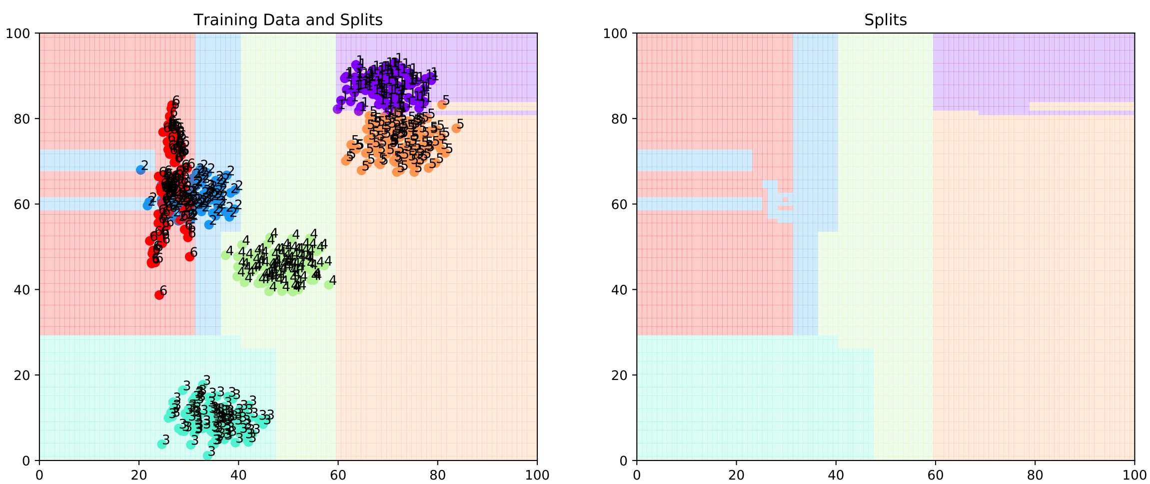


Figure B.1: Decision boundaries of a single Decision Tree on 2-dimensional synthetic data with labels 1 to 6, splitting the data until every leaf node only contains data of one cluster. Left: decision boundaries with Data, right: decision boundaries only. The Decision Tree clearly overfits the data.

The Decision Tree clearly overfits the data and tends to produce edgy boundaries. The associated Decision Tree is shown in figure B.2.

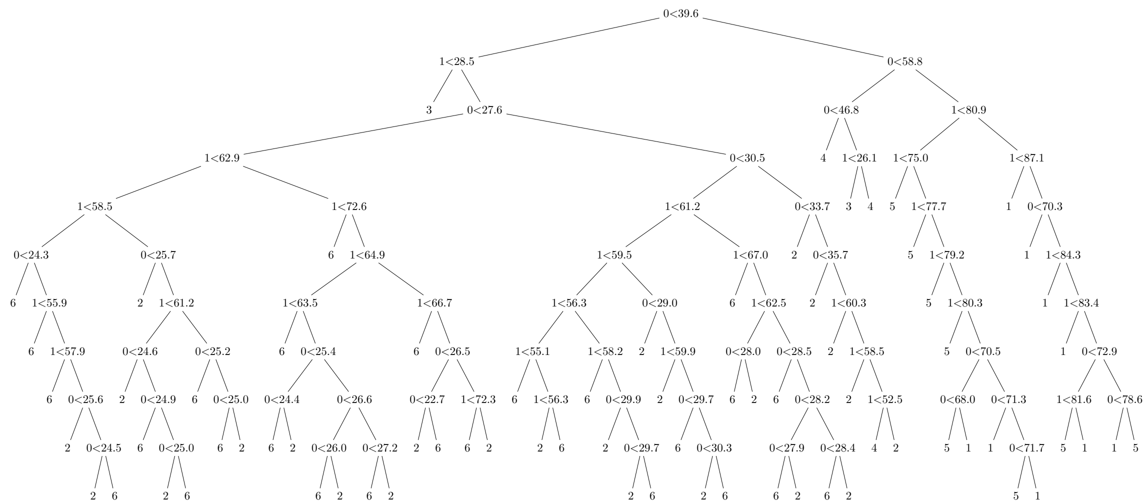


Figure B.2: Decision tree with unlimited depth on the training data for shown for illustrative purposes, with the split dimension and split value at every non-leaf node and the class label at every leaf node. The tree clearly overfits the data and produces edgy decision boundaries

The classification using Random Forest is shown in figure B.3.

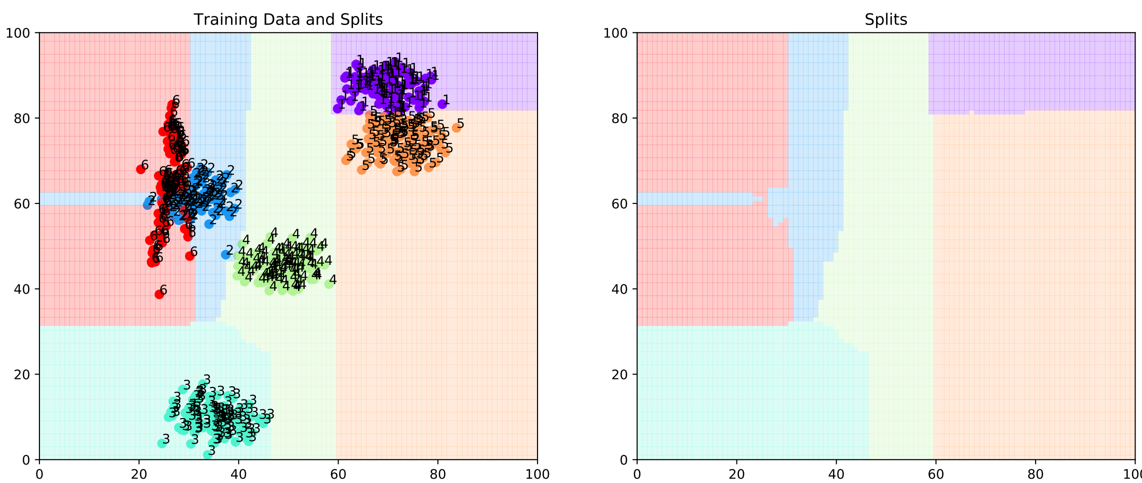


Figure B.3: Decision boundaries of a Random Forest on 2-dimensional synthetic data. 1000 Decision Trees have been trained on a 30% bootstrap sample of the original data. Left: decision boundaries with Data, right: decision boundaries only. The Random Forest manages to smooth out the class decision boundaries.

C Data Structure

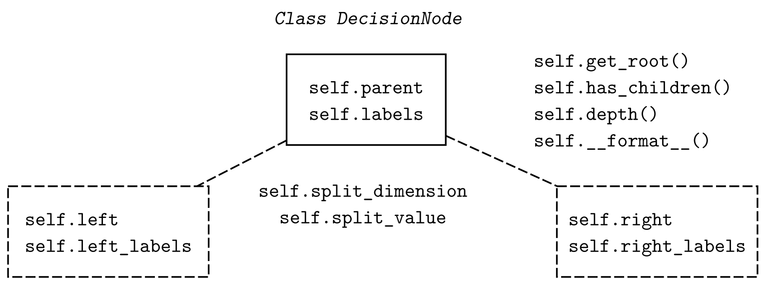


Figure C.1: Implemented data structure for Decision Tree nodes. Every node saves a pointer to its parent, the unique labels contained at its split level, the split dimension and value, methods for tree descending and formatting as well information about its child nodes.

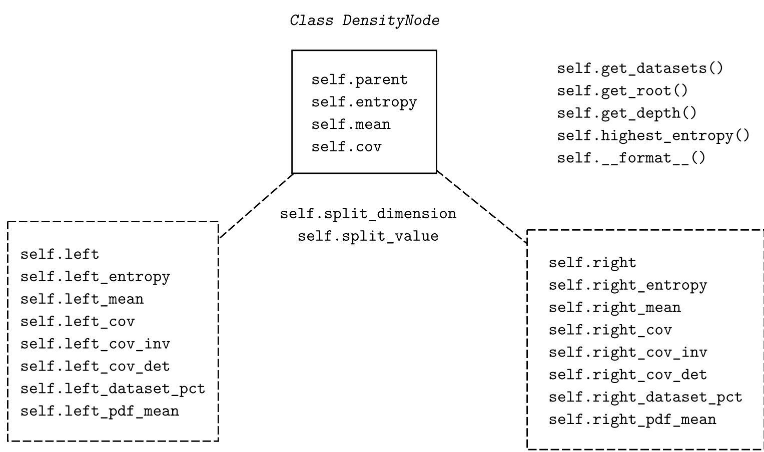


Figure C.2: Implemented data structure for Density Tree nodes. Every node saves a pointer to its parent, the split dimension and value as well as its distribution parameters including mean and covariance. In addition, each density tree node has a set of methods for tree descending and formatting as well information about its child nodes. For each child node, pre-calculated inverse and determinant of the covariance are saved to speed up calculation of Gaussian PDF during prediction

D MNIST Evaluation Metrics

Left-out class	Training set		Test set	
	OA	AA	OA	AA
0	99.29	99.30	99.01	99.02
1	99.21	99.21	98.97	98.98
2	99.34	99.35	98.80	98.80
3	99.27	99.27	99.12	99.12
4	99.33	99.32	98.95	98.93
5	99.46	99.46	99.11	99.11
6	99.36	99.36	98.94	98.94
7	99.39	99.38	99.20	99.19
8	99.38	99.38	99.03	99.04
9	99.39	99.39	99.19	99.18

Table D.1: Accuracy metrics in % for the CNN trained on $N-1$ classes for the MNIST dataset

Left-Out class	MSR	Margin	Entropy	MC-Dropout	GMM	OC-SVM	DF
0	0.98	0.98	**0.98**	0.98	0.83	0.93	0.43
1	0.99	0.99	**0.99**	0.99	0.96	0.91	0.47
2	0.97	0.97	**0.97**	0.96	0.77	0.82	0.55
3	0.96	0.96	**0.96**	0.96	0.89	0.87	0.69
4	0.92	0.92	0.93	0.91	0.79	**0.95**	0.45
5	0.96	0.96	**0.96**	0.96	0.46	0.87	0.41
6	0.97	0.97	**0.97**	0.97	0.85	0.92	0.63
7	0.97	0.97	**0.97**	0.96	0.94	0.88	0.40
8	0.98	0.98	**0.98**	0.98	0.82	0.90	0.61
9	0.98	0.98	**0.98**	0.98	0.92	0.92	0.70
Mean	0.97	0.97	**0.97**	0.97	0.82	0.90	0.53

Table D.2: AUROC for each left-out class in the MNIST dataset

E Zurich Network Architecture

Name	Layer	Parameters
	Input	Dim: $(\|b\|, w, h, n_c)$
	Conv + ReLu	32 5×5 filt
	Dropout	$p = .1$
`conv1`	Conv + ReLu	32 5×5 filt
	MaxPooling	2×2 pool size
	Conv + ReLu	64 3×3 filt
	Dropout	$p = .1$
`conv2`	Conv + ReLu	64 3×3 filt
	MaxPooling	2×2 pool size
	Conv + ReLu	128 3×3 filt
	Dropout	$p = .1$
`conv3`	Conv + ReLu	128 3×3 filt
	MaxPooling	2×2 pool size
	Conv + ReLu	256 3×3 filt
	Dropout	$p = .1$
`conv4`	Conv + ReLu	256 3×3 filt
	Conv + ReLu	512 3×3 filt
	MaxPooling	2×2 pool size
	Dropout	$p = .1$
`conv5`	Conv + ReLu	512 3×3 filt

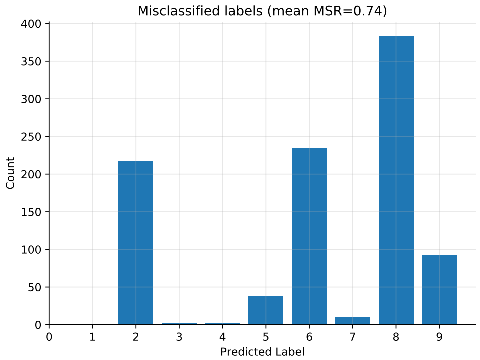
(a) Downsampling Layers

Type	Parameters
Concat(Conv T, `conv4`)	256 2×2 filt, 2 str
Conv + ReLu	256 3×3 filt
Dropout	$p = .1$
Conv + ReLu	256 3×3 filt
Concat(Conv T, `conv3`)	128 2×2 filt, 2 str
Conv + ReLu	128 3×3 filt
Dropout	$p = .1$
Conv + ReLu	128 3×3 filt
Concat(Conv T, `conv2`)	64 2×2 filt, 2 str
Conv + ReLu	64 3×3 filt
Dropout	$p = .1$
Conv + ReLu	64 3×3 filt
Concat(Conv T, `conv1`)	32 2×2 filt, 2 str
Conv + ReLu	32 5×5 filt
Dropout	$p = .1$
Conv + ReLu	32 5×5 filt
Conv + Softmax	$N - 1$ 1×1 filt

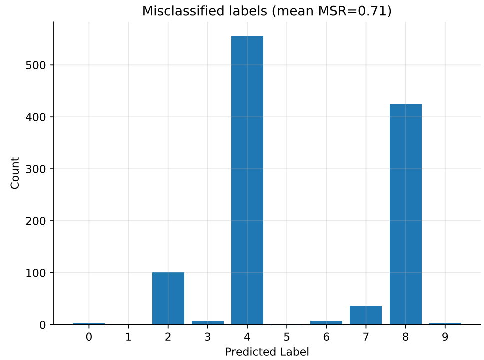
(b) Upsampling Layers

Table E.1: U-Net Architecture of the CNN used for Zurich Dataset, according to Ronneberger et al. [41]. Conv = convolution, filt = filters, str = stride, p = dropout probability, dim = dimensions, Input dimensions $|b|, w, h, n_c$ = batch size, width, height, number of channels, N = number of classes. A convolution or transpose convolution always takes the previous layer in the network as input.

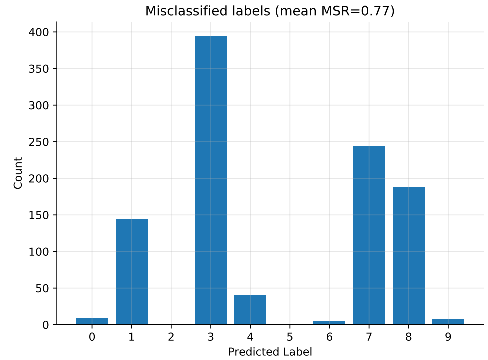
F MNIST Dataset Figures



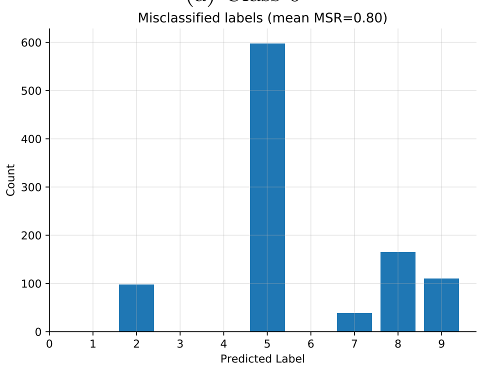
(a) Class 0



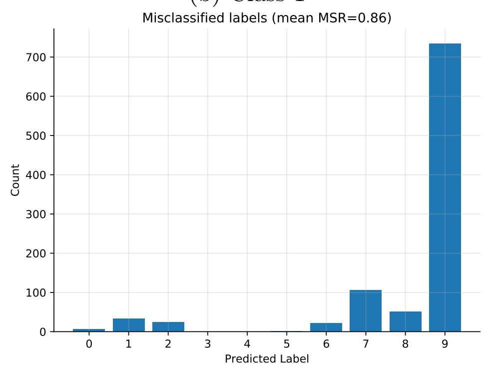
(b) Class 1



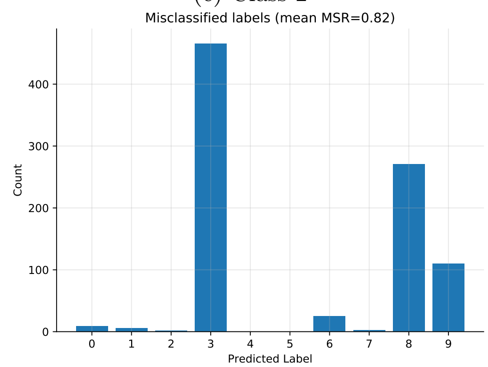
(c) Class 2



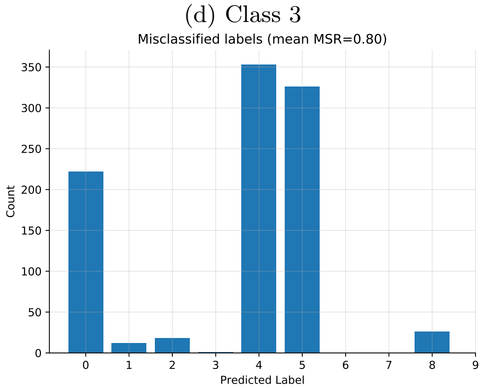
(d) Class 3



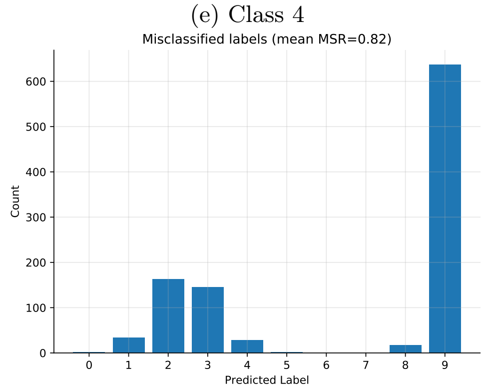
(e) Class 4



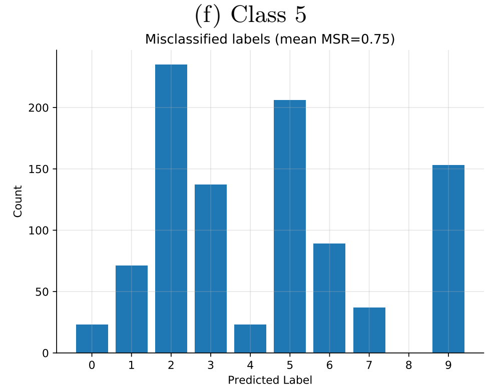
(f) Class 5



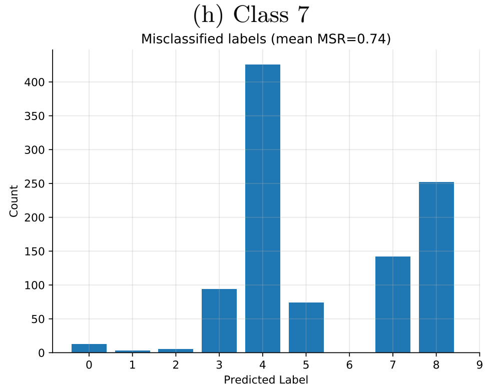
(g) Class 6



(h) Class 7

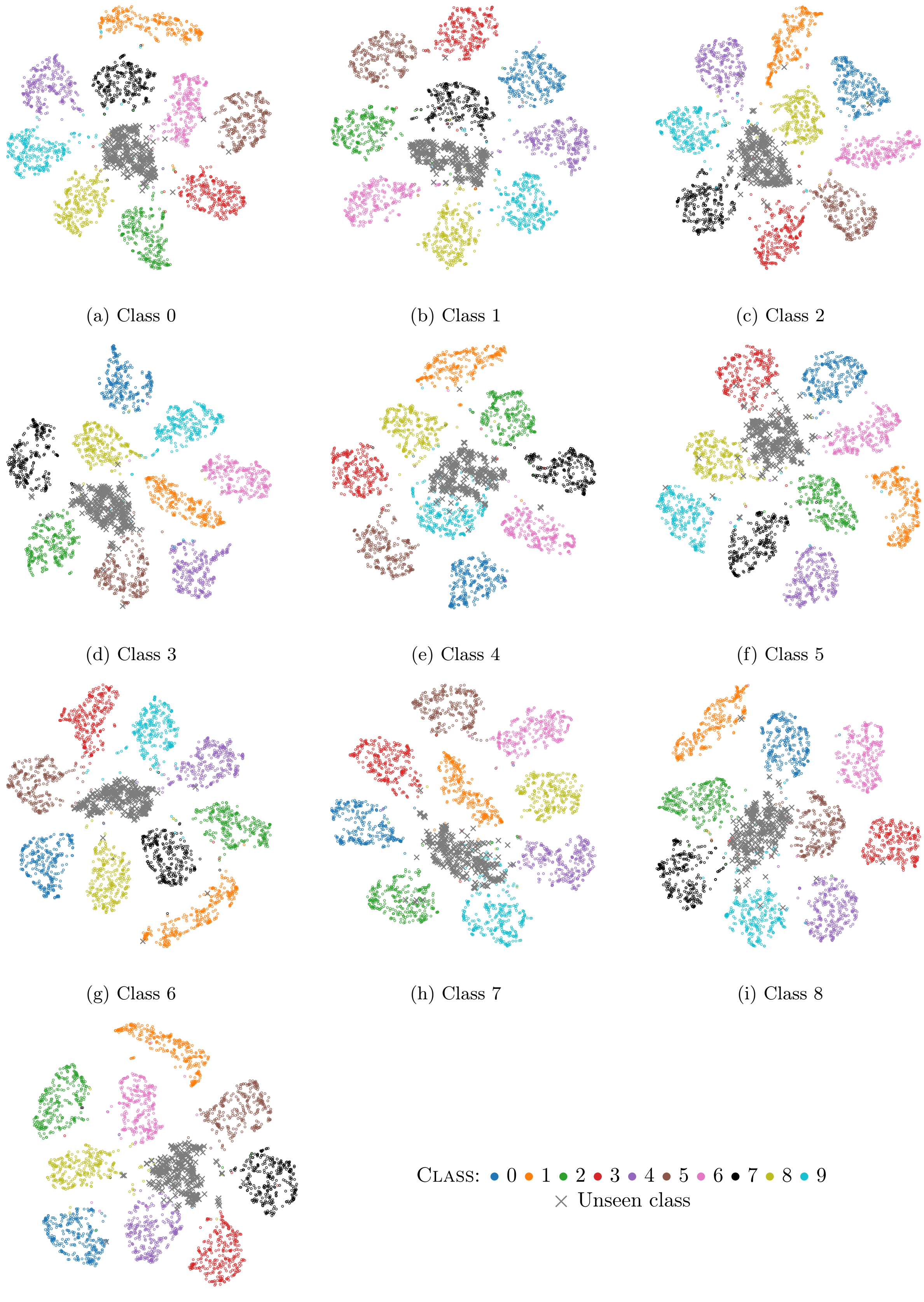


(i) Class 8



(j) Class 9

Figure F.1: Count of predictions for each left-out class



(a) Class 0 (b) Class 1 (c) Class 2

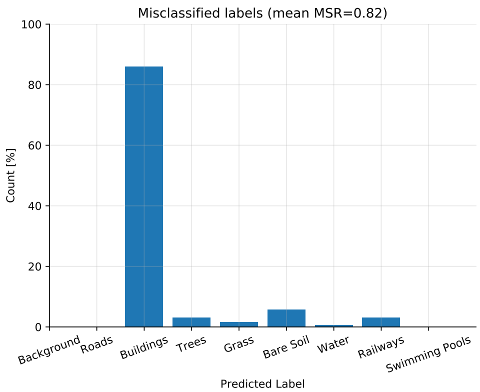
(d) Class 3 (e) Class 4 (f) Class 5

(g) Class 6 (h) Class 7 (i) Class 8

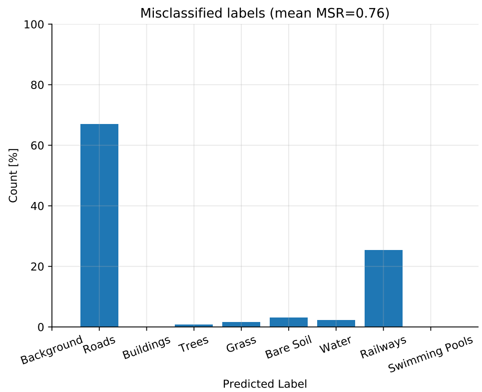
(j) Class 9

Figure F.2: t-SNE of MNIST dataset activations after PCA transformations for each left-out class

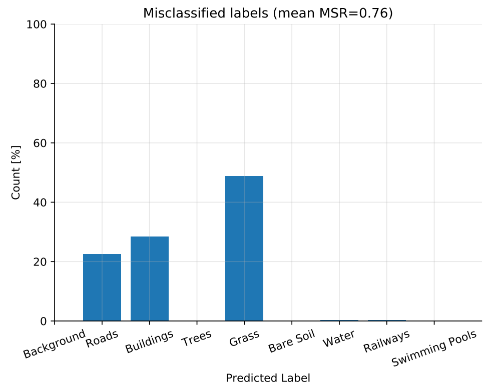
G Zurich Dataset Figures



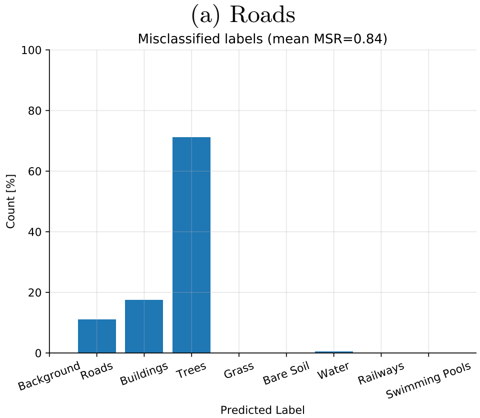
(a) Roads



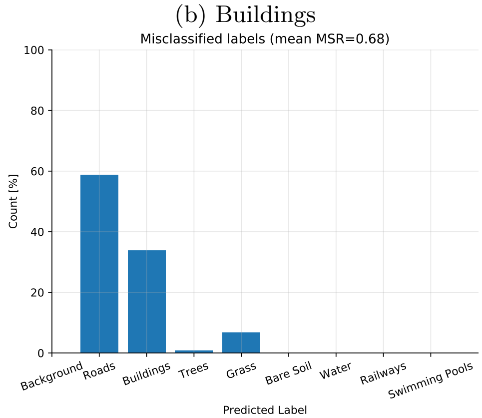
(b) Buildings



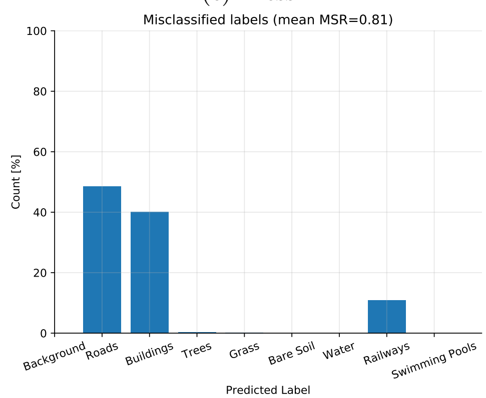
(c) Tress



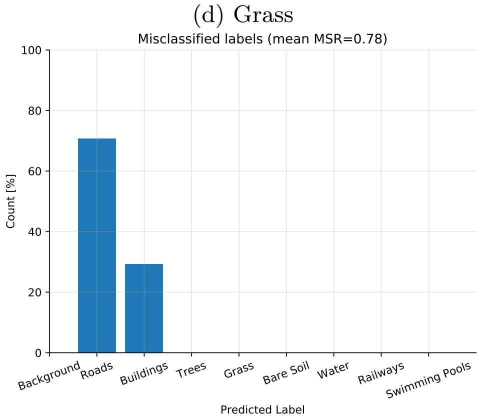
(d) Grass



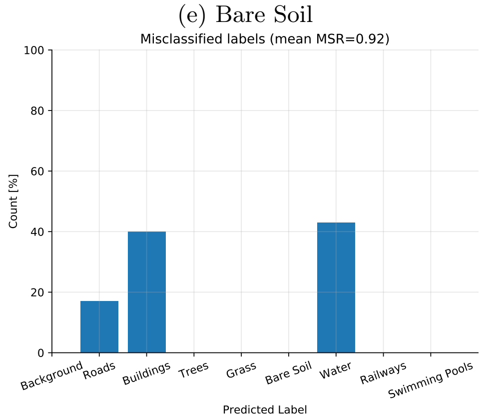
(e) Bare Soil



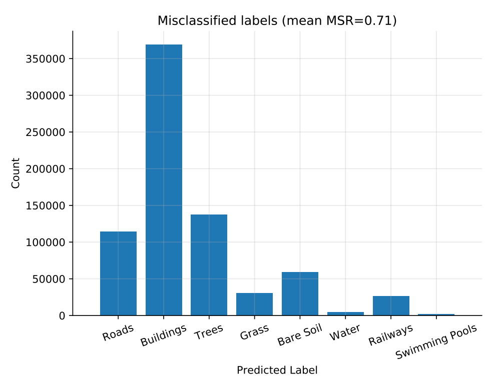
(f) Water



(g) Railways

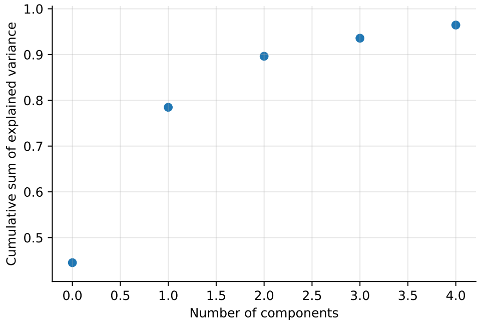


(h) Swimming Pools

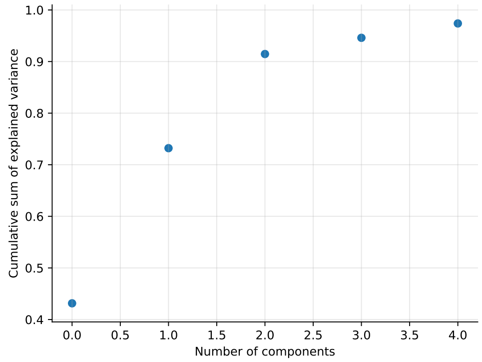


(i) All classes

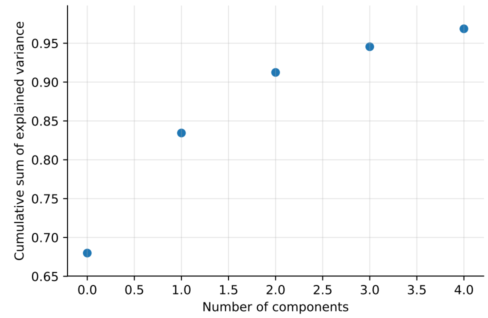
Figure G.1: Count of label predictions for each left-out class and for the CNN trained on all classes



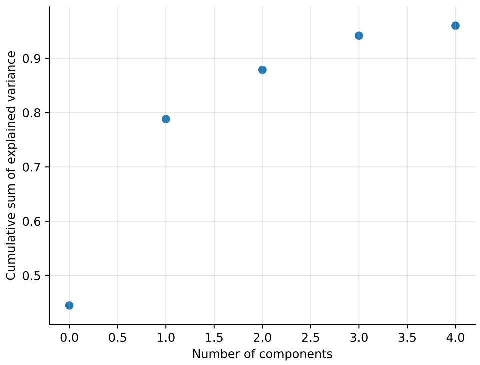
(a) Roads



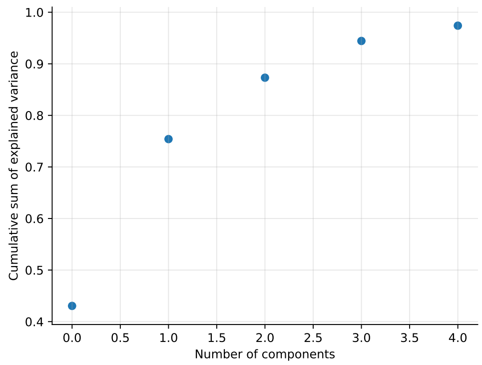
(b) Buildings



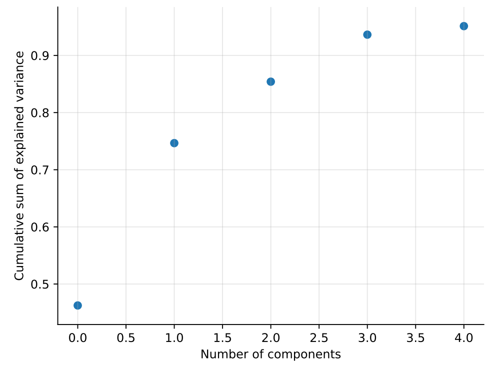
(c) Tress



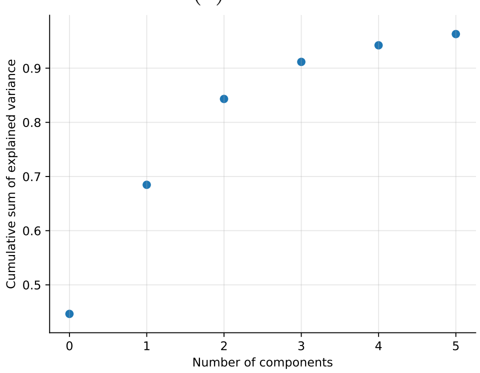
(d) Grass



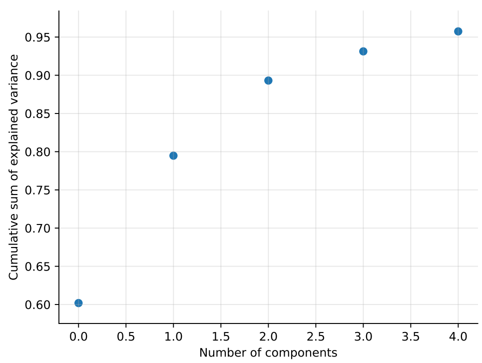
(e) Bare Soil



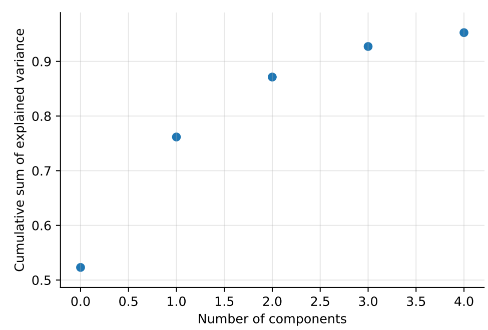
(f) Water



(g) Railways

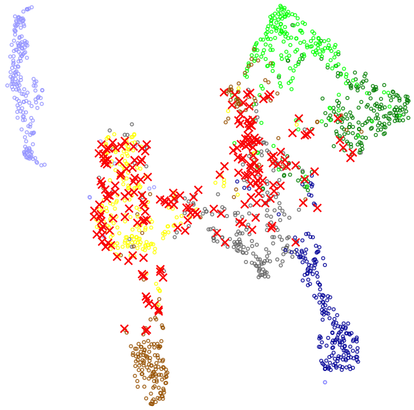


(h) Swimming Pools

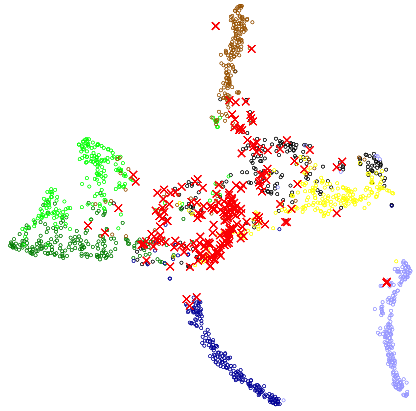


(i) All classes

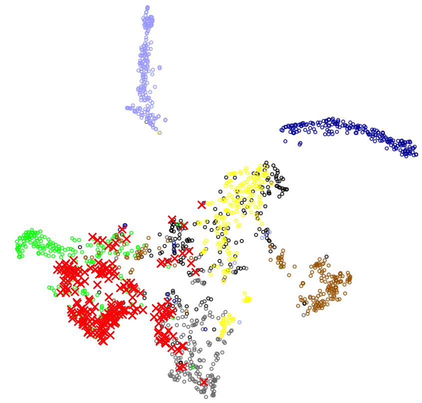
Figure G.2: Cumulative variance explained by PCA components for activations of each left-out class and for activations of the model trained on all classes. The number of PCA components was chosen such as to explain more than 95% of the variance.



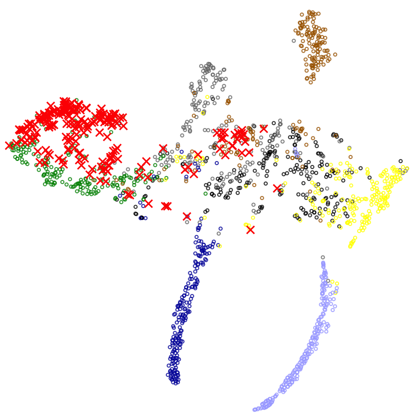
(a) Roads



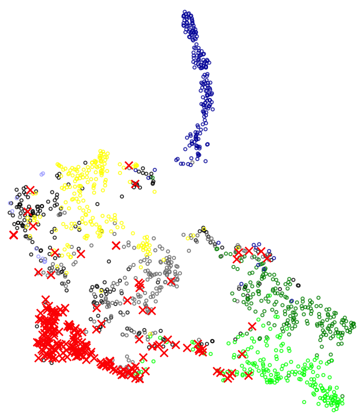
(b) Buildings



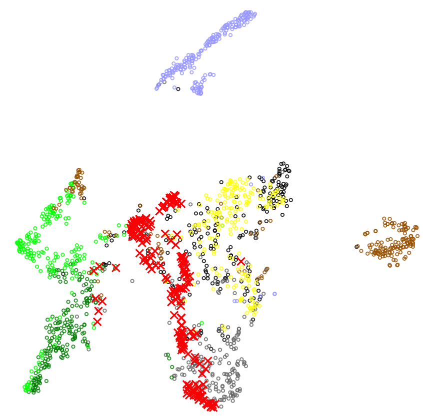
(c) Tress



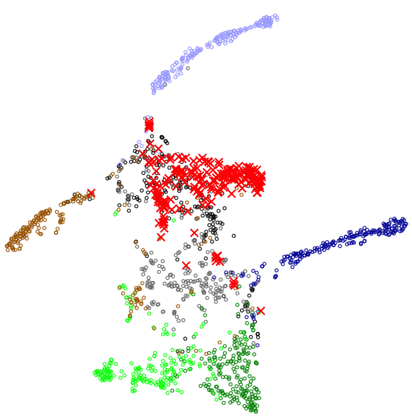
(d) Grass



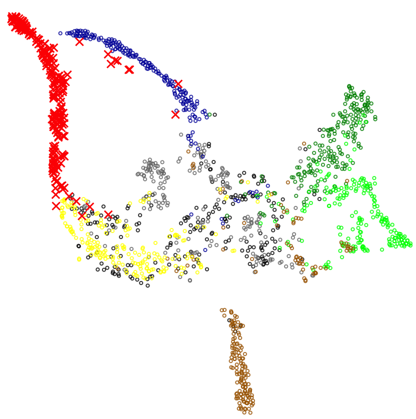
(e) Bare Soil



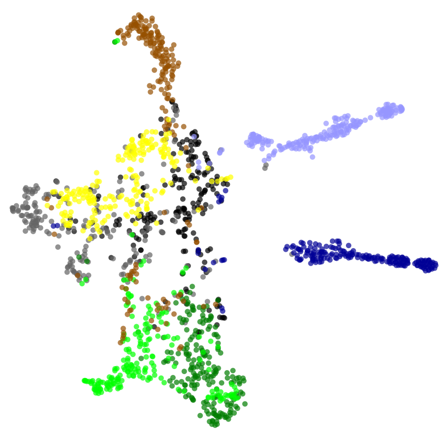
(f) Water



(g) Railways



(h) Swimming Pools



(i) All classes

CLASS: ● Roads ● Buildings ● Trees ● Grass
● Bare Soil ● Water ● Railways ● Pools
× Unseen class

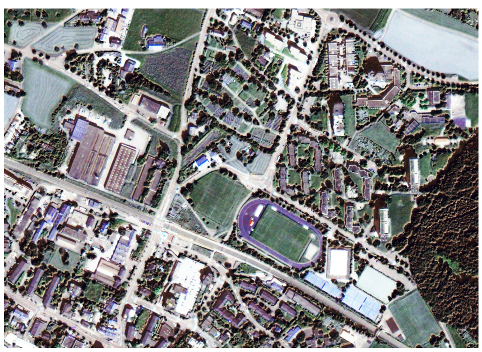
Figure G.3: t-SNE of Zurich dataset activations after PCA transformations for each left-out class and for the activations of the network trained on all classes. The same number of points are shown by class to show class separability, although the real class distribution is imbalanced (table 8).

On the following pages, confidence scores are shown for all left-out classes and for one confidence estimation method at a time. For each left-out class, the confidence values are shown for one image of the training set containing some object with the corresponding ground truth. More images can be found on `https://github.com/CyrilWendl/SIE-Master/tree/master/Figures/Zurich/Im_cert`. The following images are shown below:

(a) Image

(b) Ground Truth

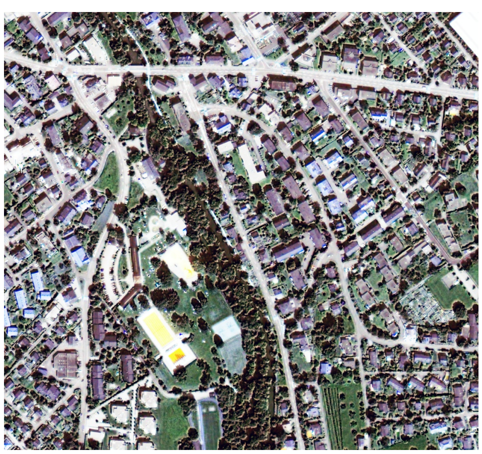
Figure G.4: Image and ground truth pair for confidence images shown for left-out classes roads, buildings, trees, grass and water



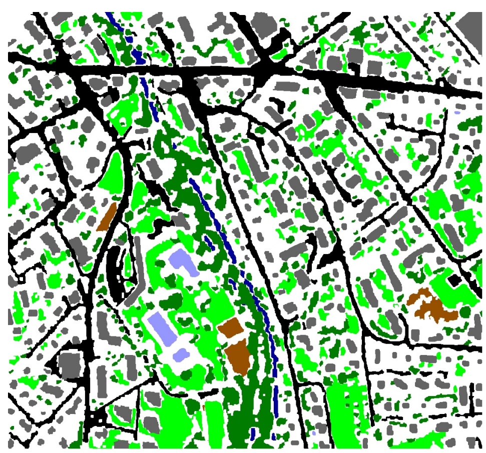
(a) Image

(b) Ground Truth

Figure G.5: Image and ground truth pair for confidence images shown for left-out class bare soil



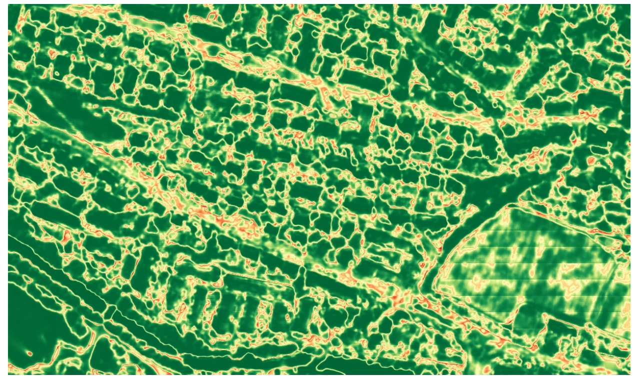
(a) Image



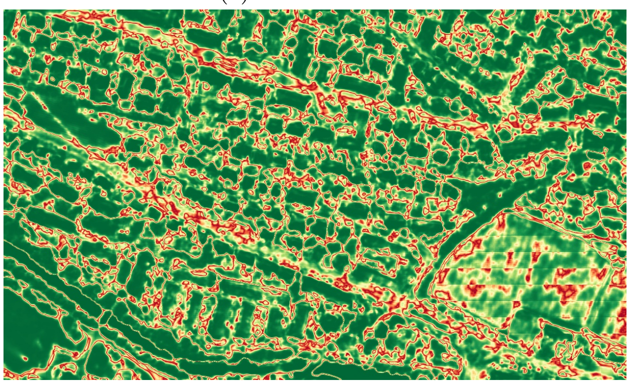
(b) Ground Truth

Figure G.6: Image and ground truth pair for confidence images shown for left-out class swimming pools

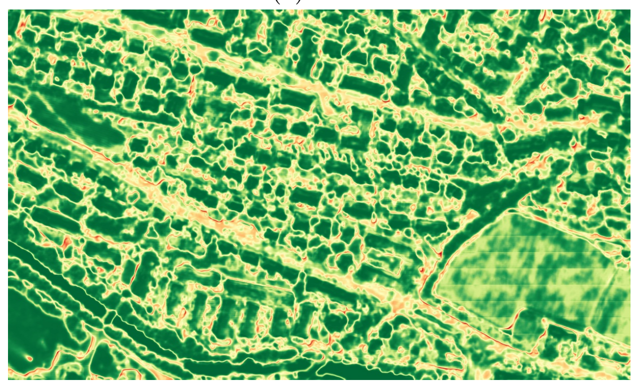
(a) Ground Truth



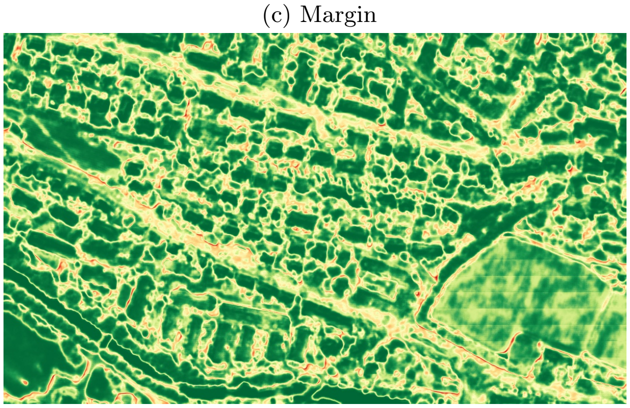
(b) MSR



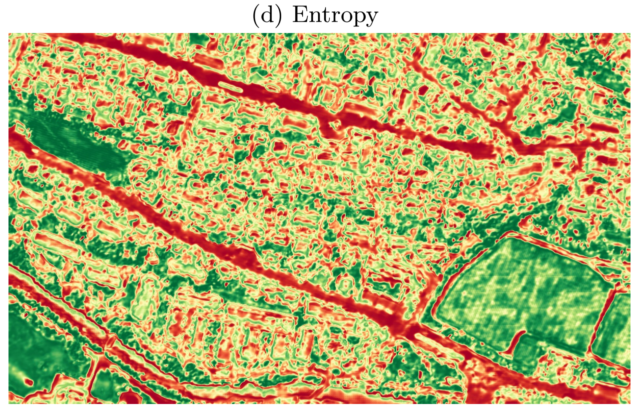
(c) Margin



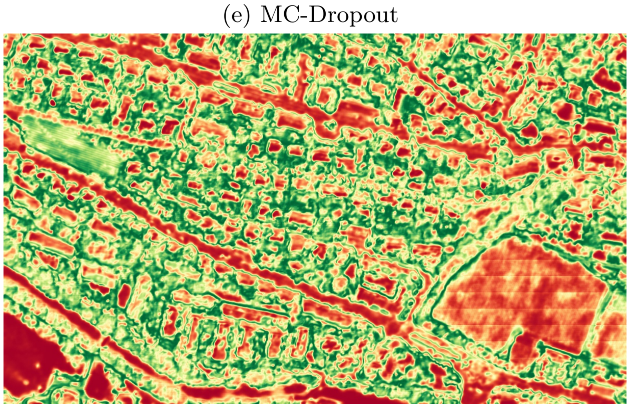
(d) Entropy



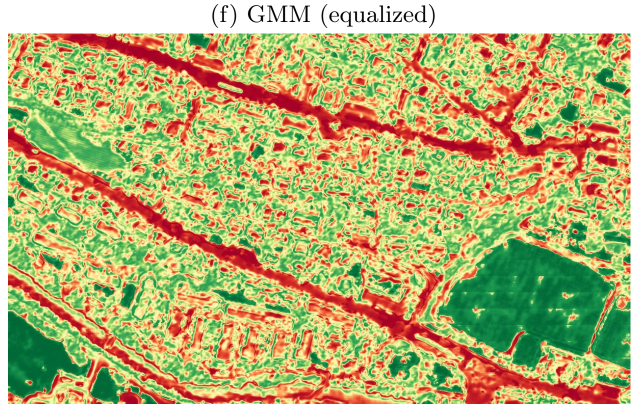
(e) MC-Dropout



(f) GMM (equalized)



(g) OC-SVM (equalized)



(h) DF (equalized)

Ground Truth

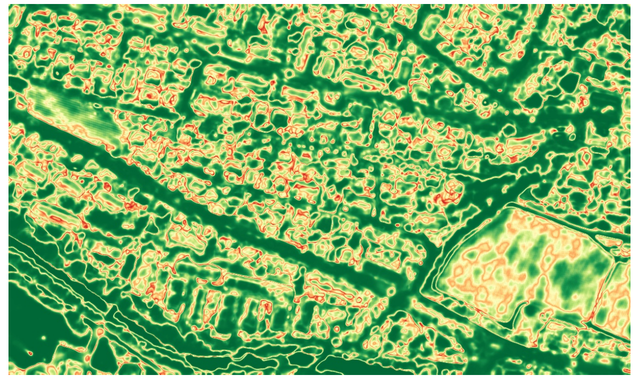
Background Roads Buildings Trees Grass
Bare Soil Water Railways Swimming Pools

Confidence

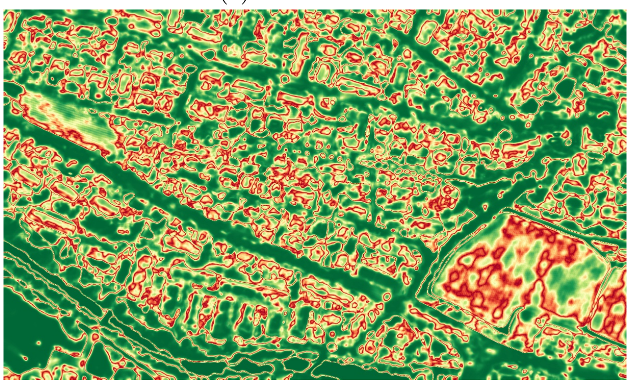
Low High

Figure G.7: Ground Truth and visual results for left-out class "roads".

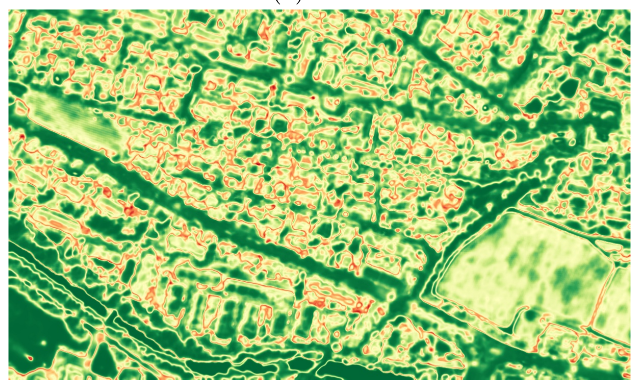
(a) Ground Truth



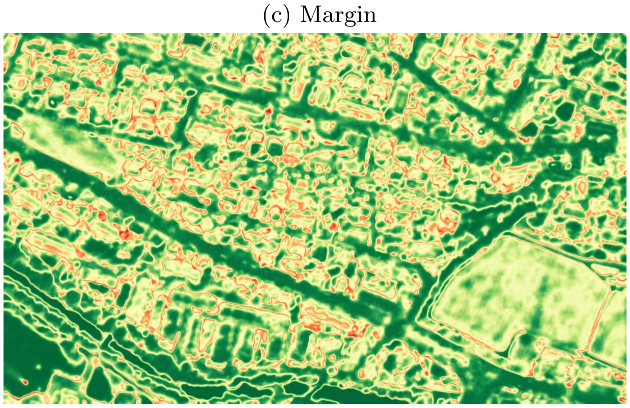
(b) MSR



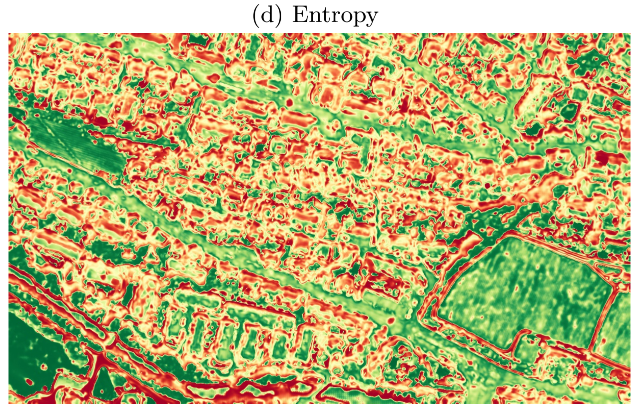
(c) Margin



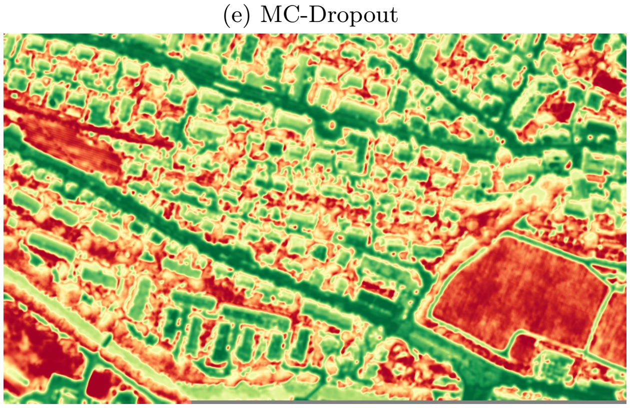
(d) Entropy



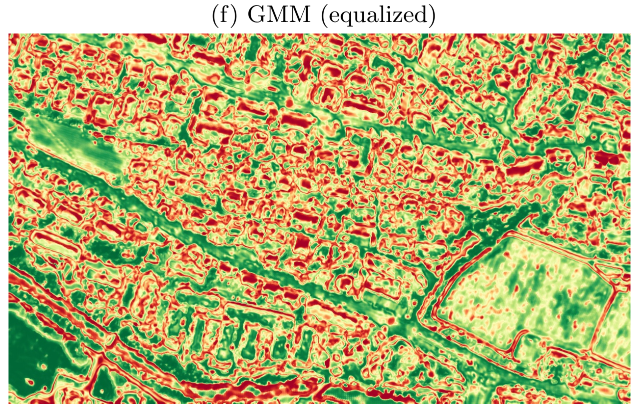
(e) MC-Dropout



(f) GMM (equalized)



(g) OC-SVM (equalized)



(h) DF (equalized)

Ground Truth

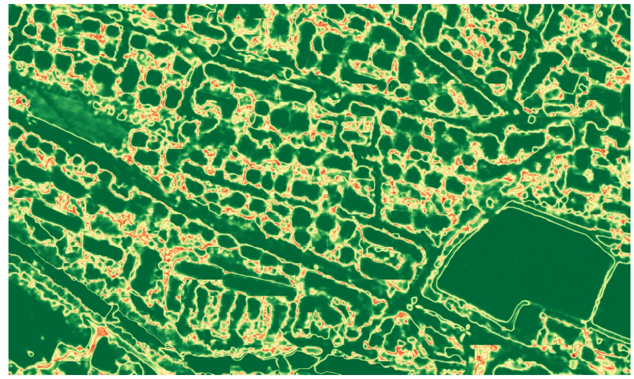
Background Roads Buildings Trees Grass
Bare Soil Water Railways Swimming Pools

Confidence

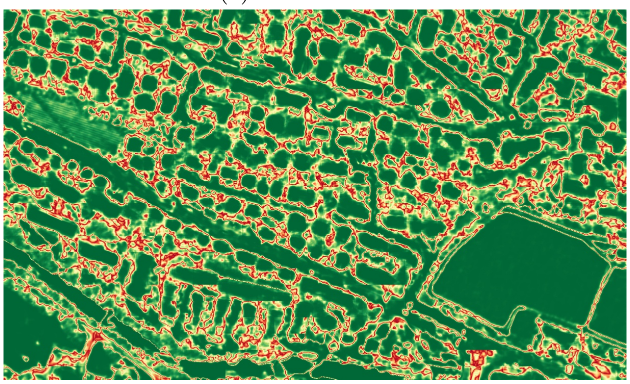
Low High

Figure G.8: Ground Truth and visual results for left-out class "buildings".

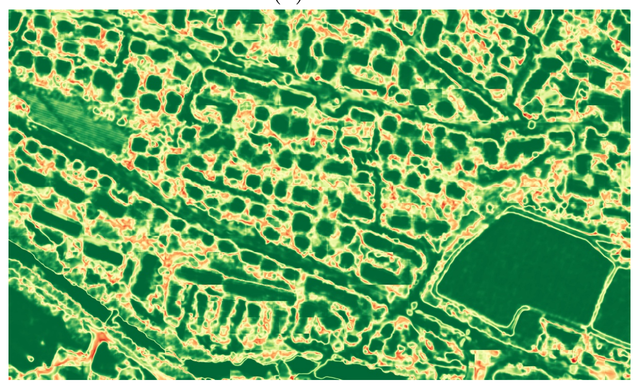
(a) Ground Truth



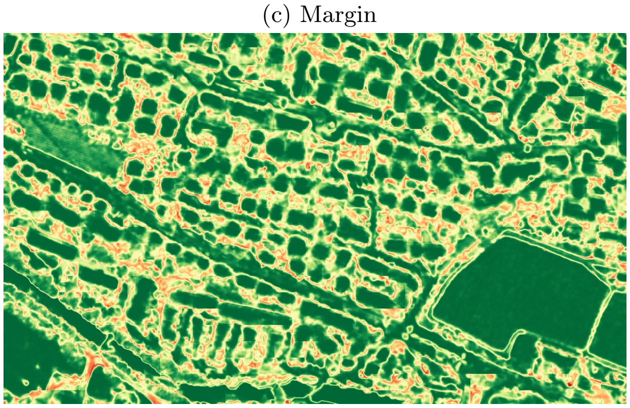
(b) MSR



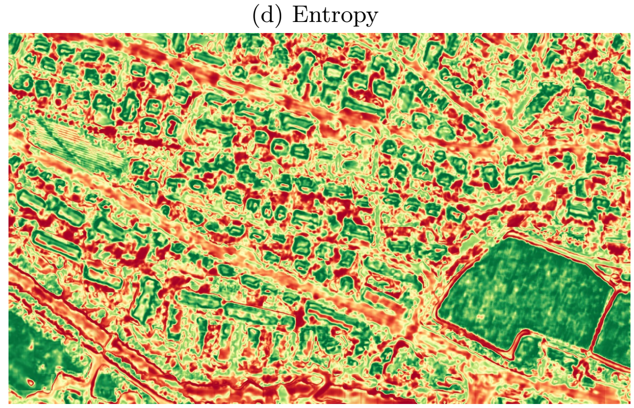
(c) Margin



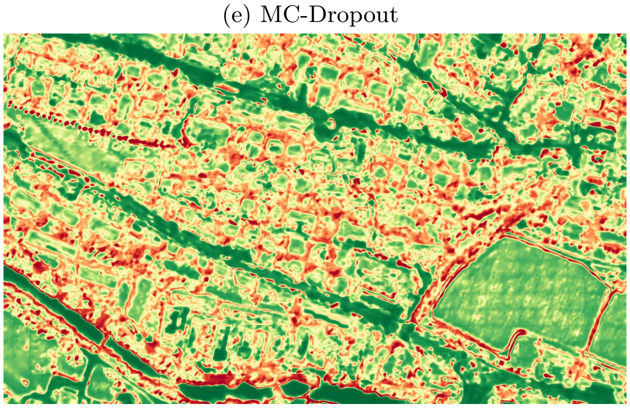
(d) Entropy



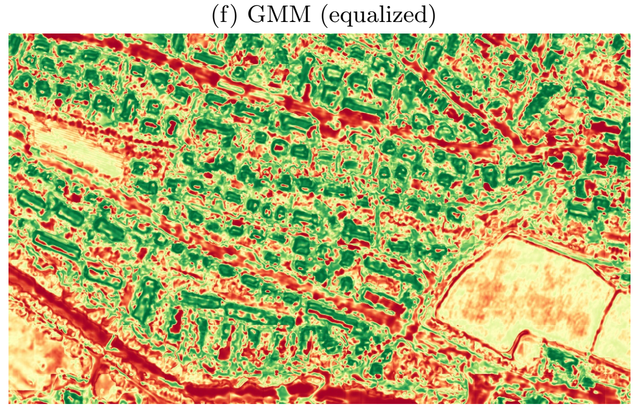
(e) MC-Dropout



(f) GMM (equalized)



(g) OC-SVM (equalized)



(h) DF (equalized)

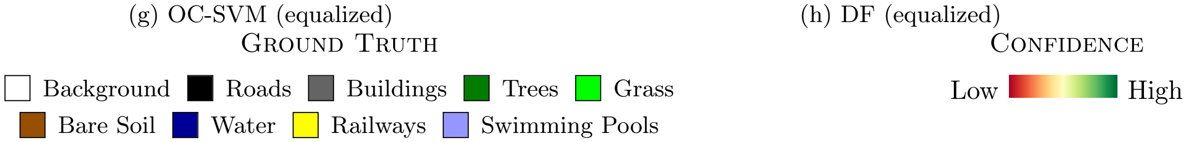
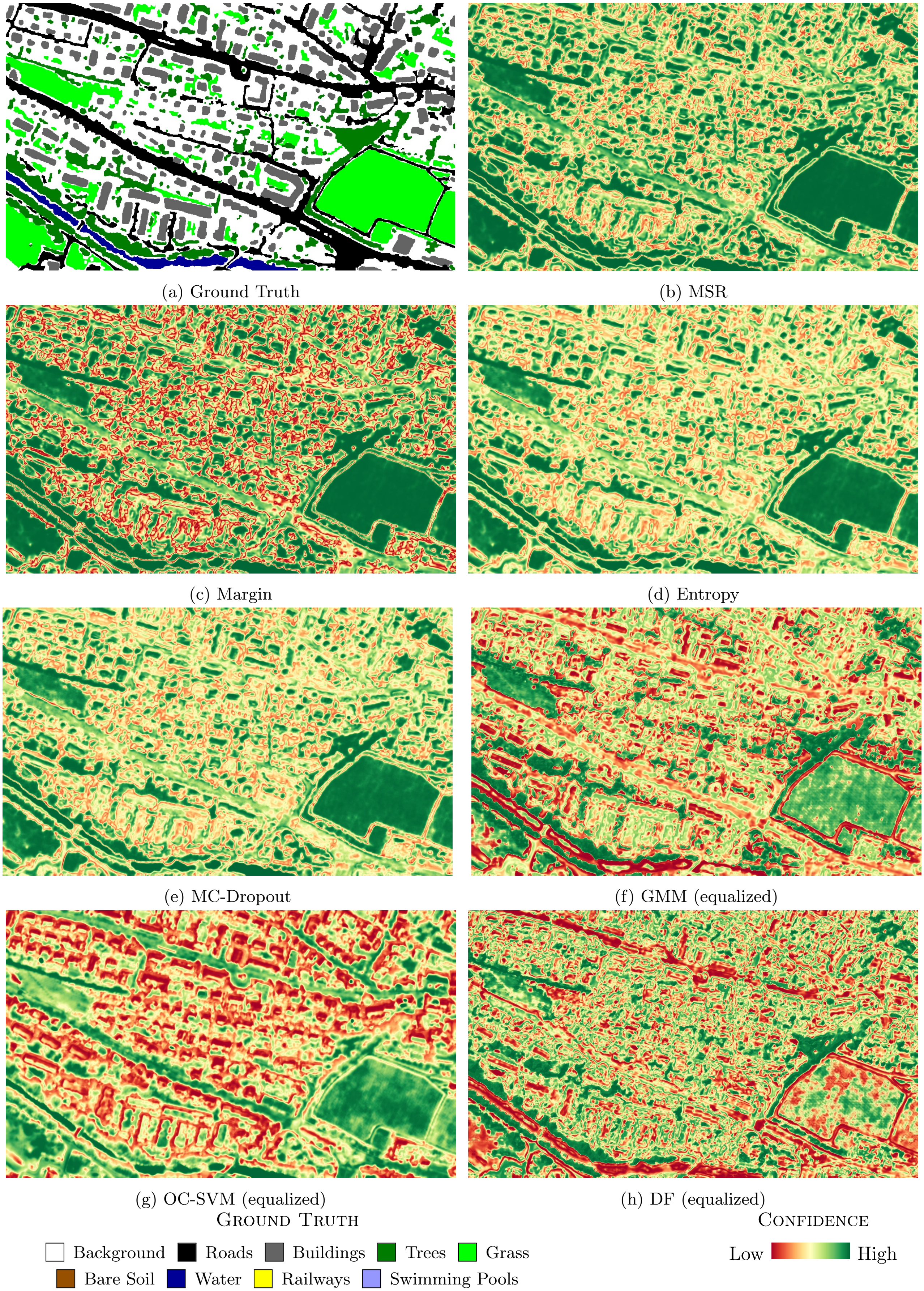


Figure G.9: Ground Truth and visual results for left-out class "trees".



(a) Ground Truth (b) MSR (c) Margin (d) Entropy (e) MC-Dropout (f) GMM (equalized) (g) OC-SVM (equalized) (h) DF (equalized)

Figure G.10: Ground Truth and visual results for left-out class “grass”.

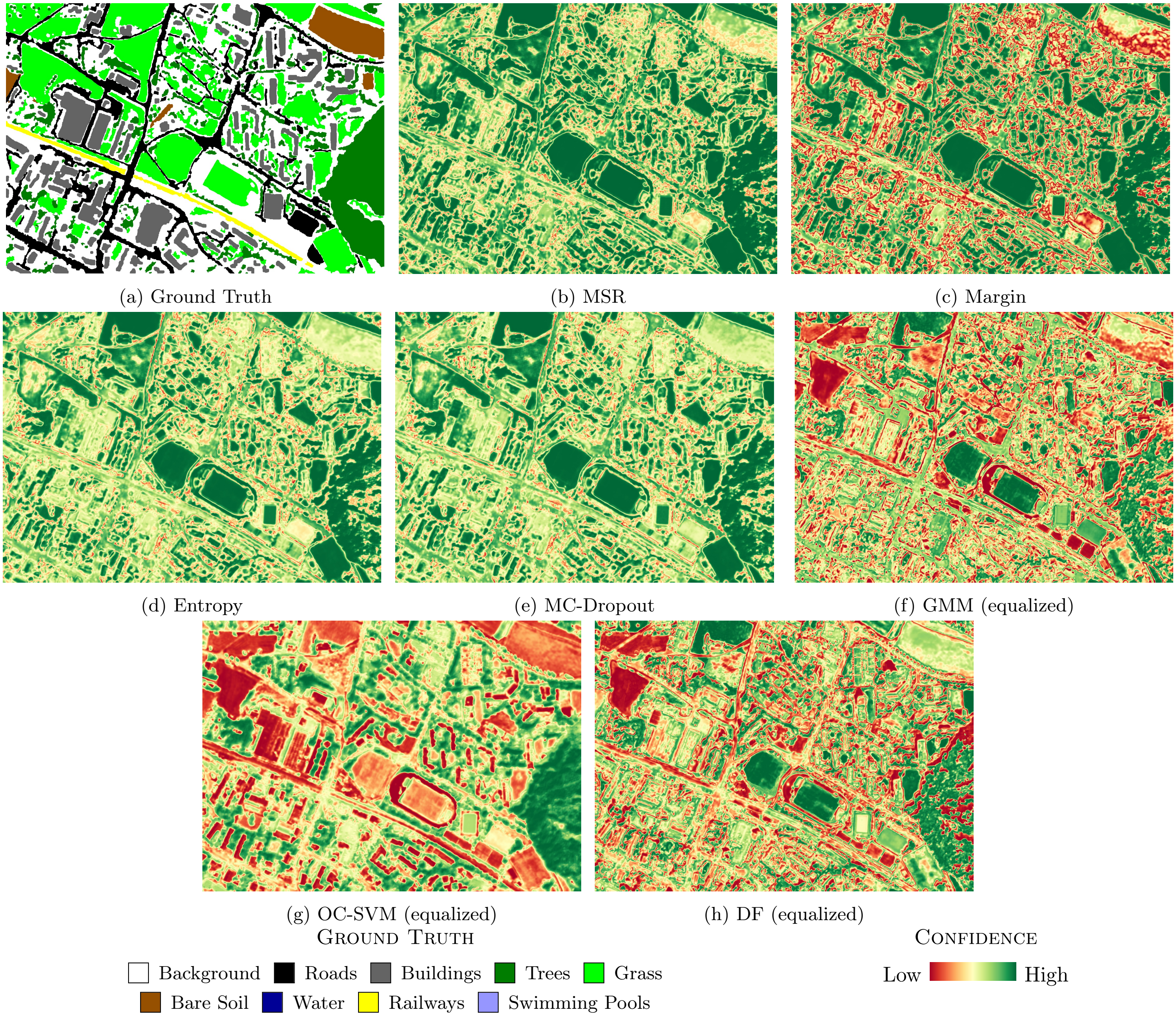


Figure G.11: Ground Truth and visual results for left-out class “bare soil”.

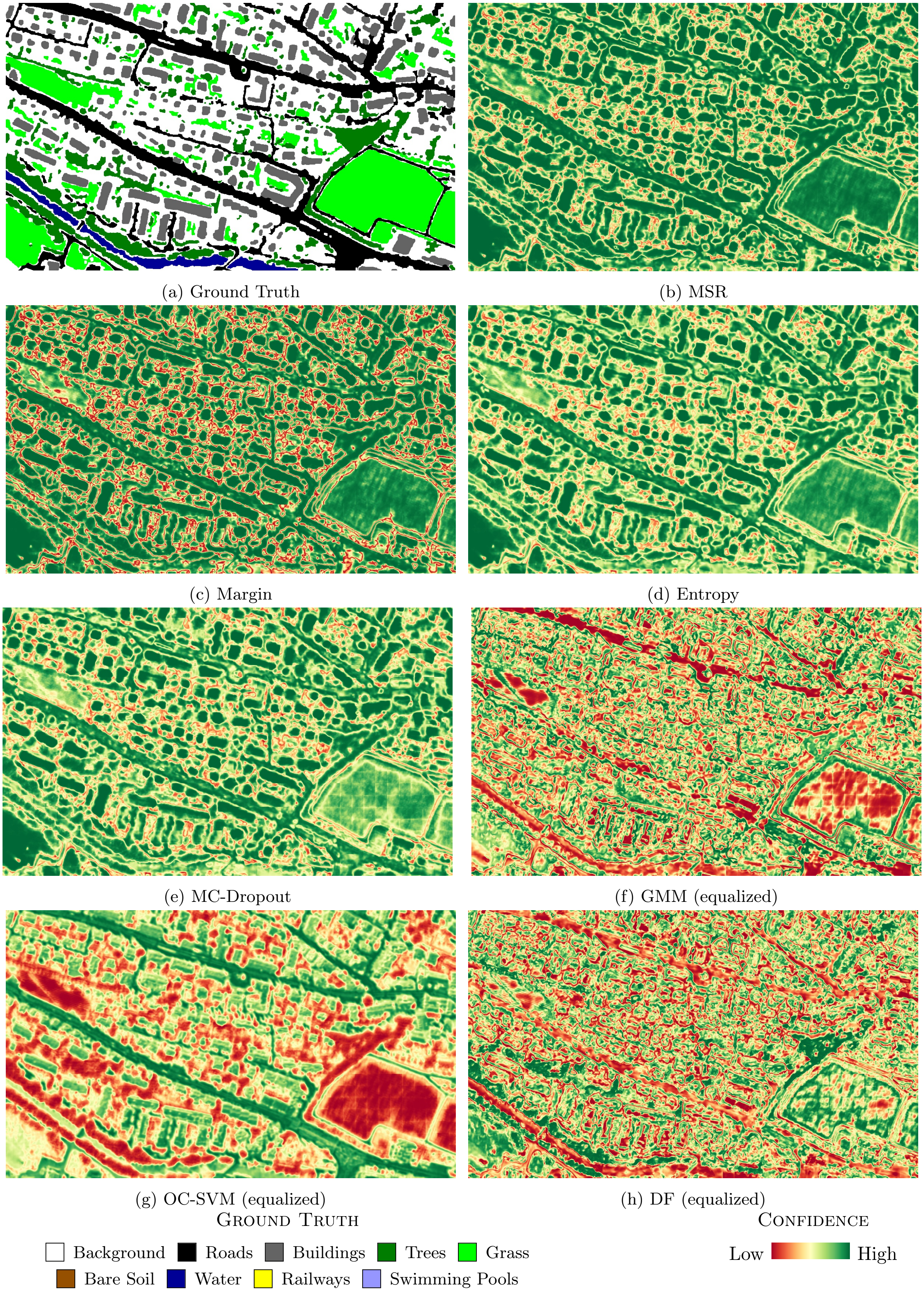


Figure G.12: Ground Truth and visual results for left-out class “water”.

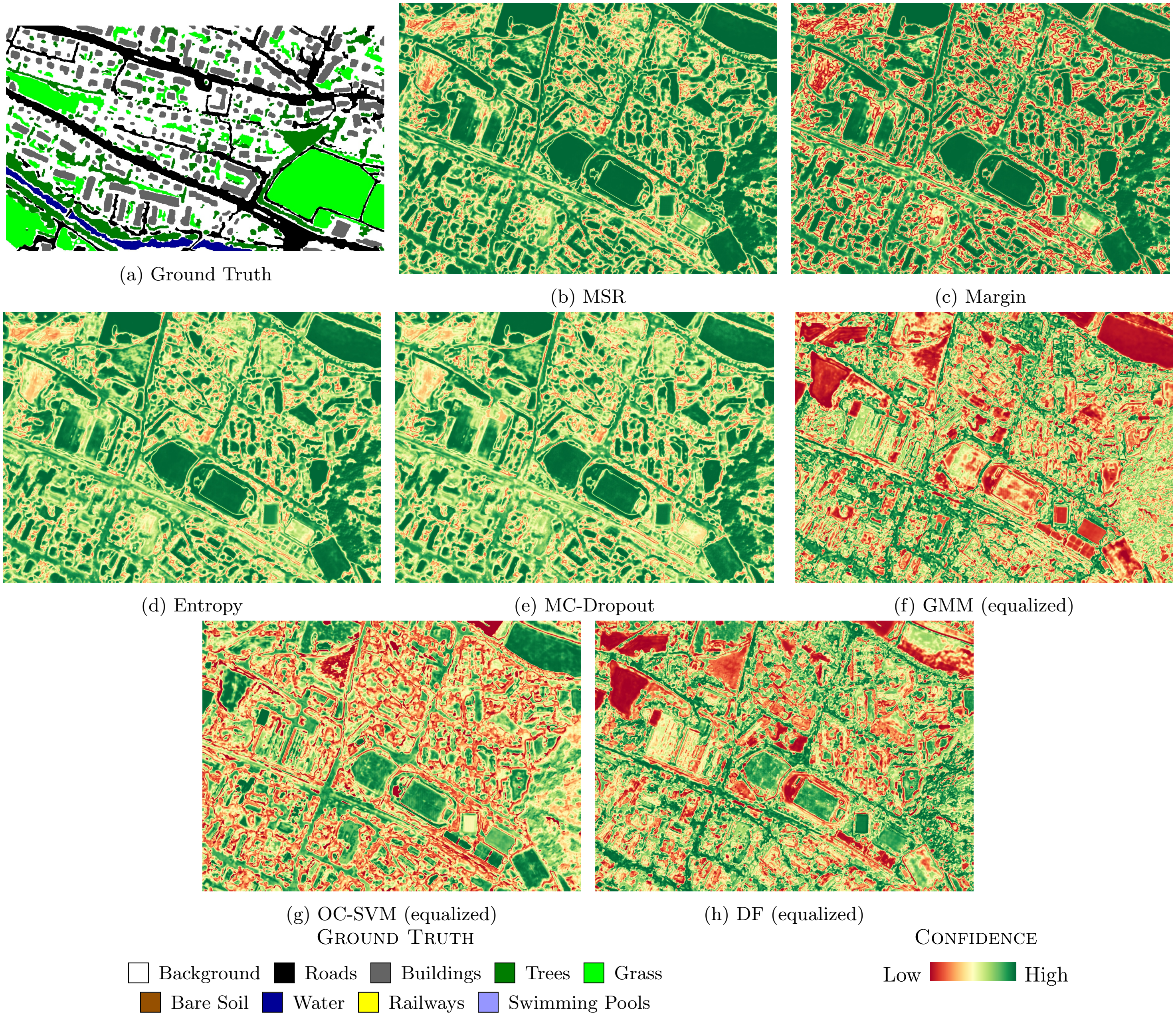
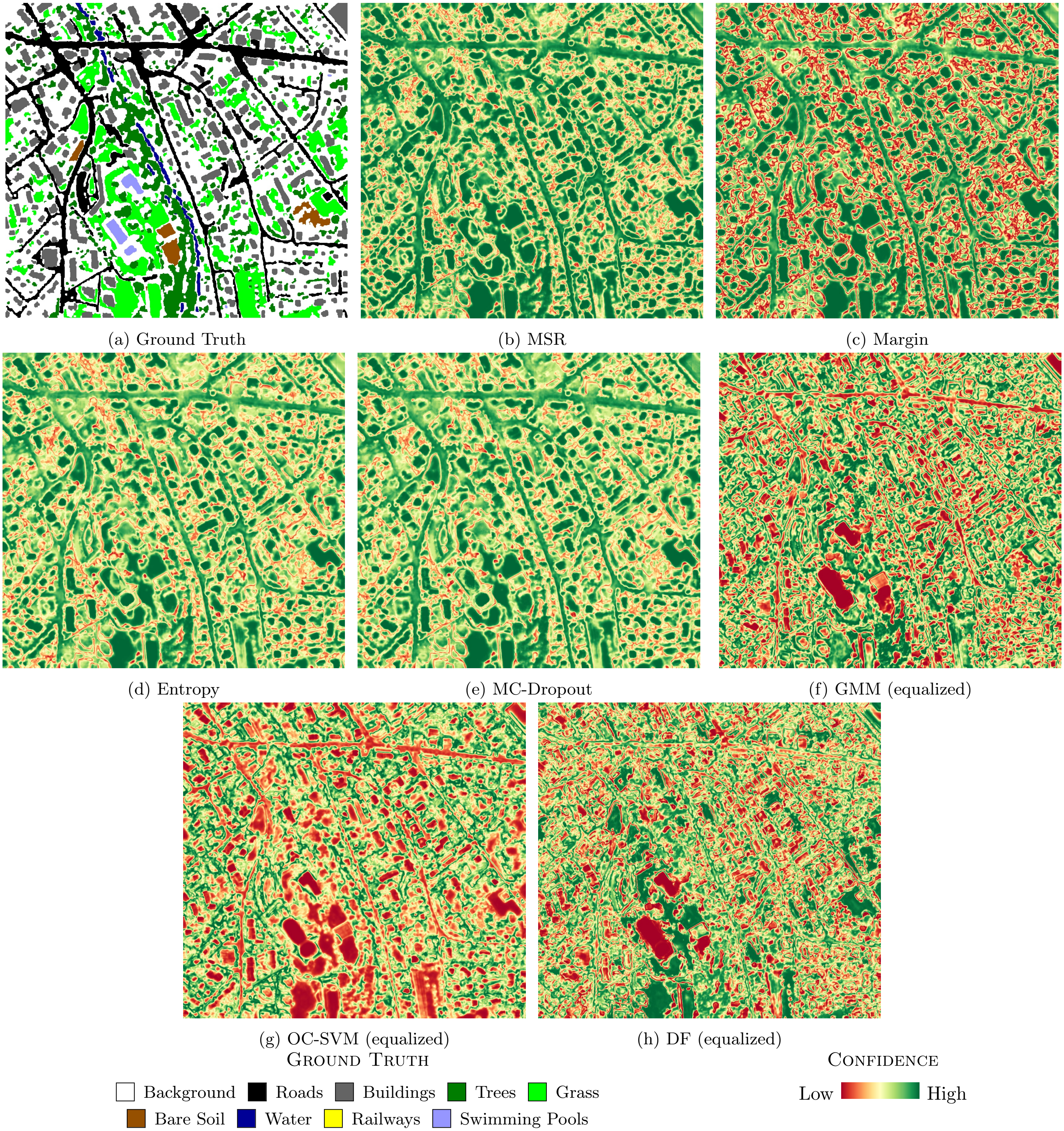


Figure G.13: Ground Truth and visual results for left-out class “railways”.

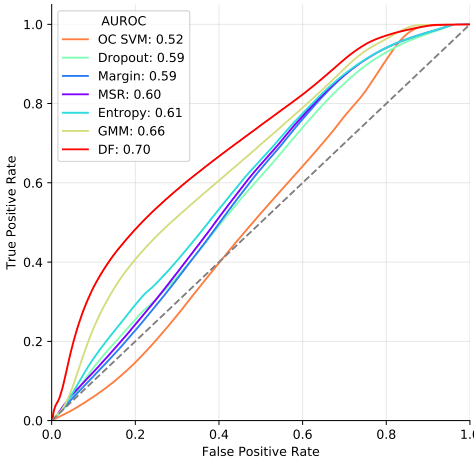


(a) Ground Truth (b) MSR (c) Margin

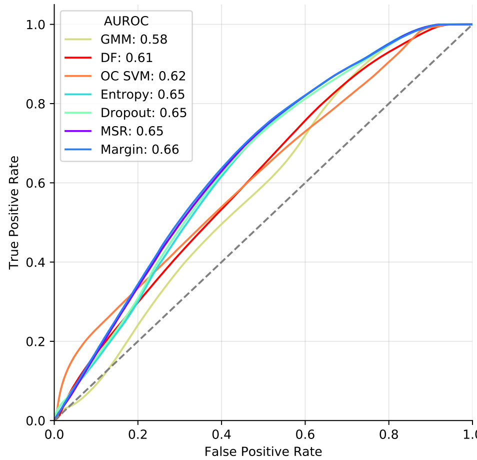
(d) Entropy (e) MC-Dropout (f) GMM (equalized)

(g) OC-SVM (equalized) (h) DF (equalized)

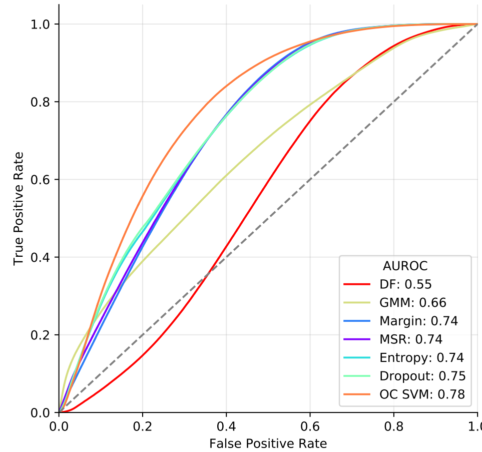
Figure G.14: Ground Truth and visual results for left-out class “swimming pools”.



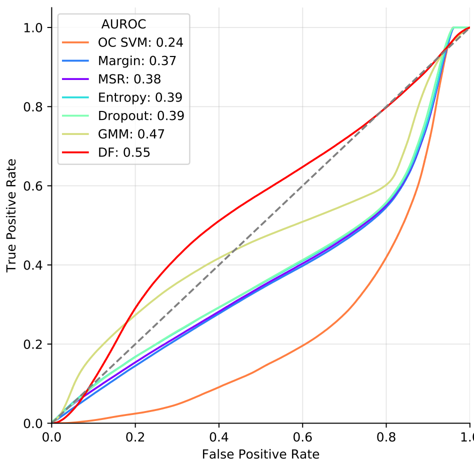
(a) Roads



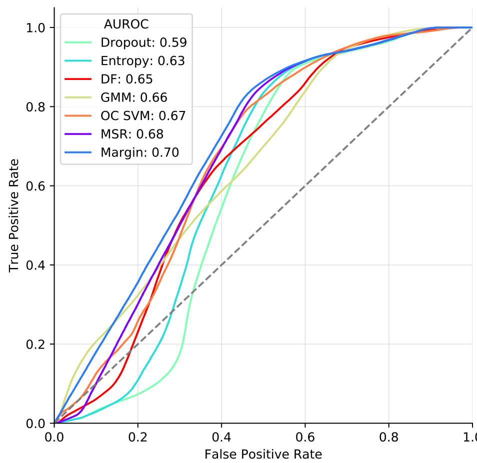
(b) Buildings



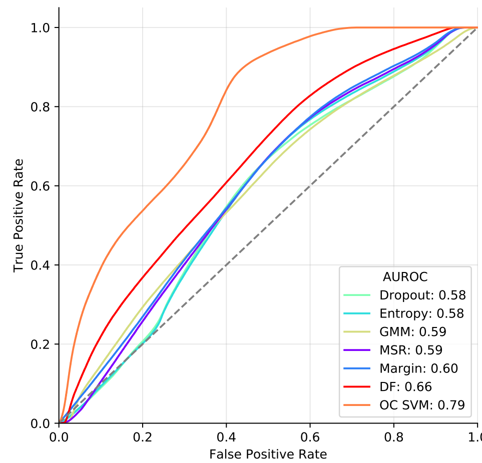
(c) Trees



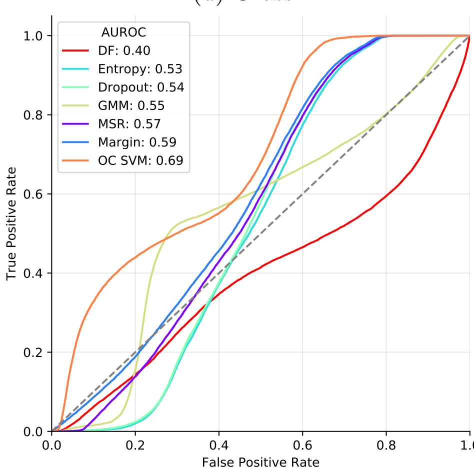
(d) Grass



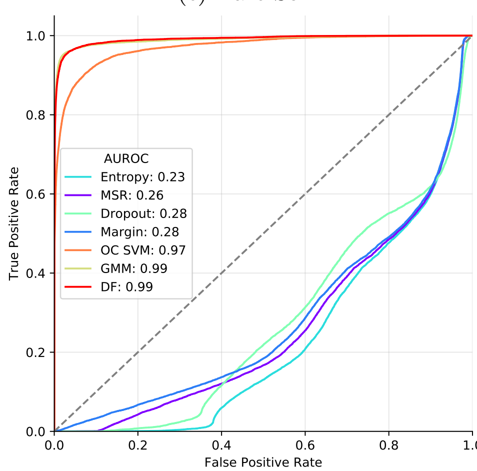
(e) Bare Soil



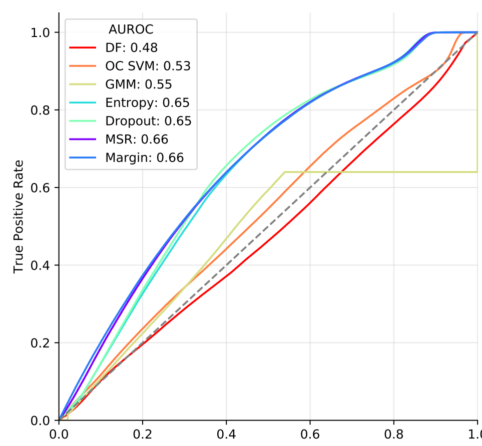
(f) Water



(g) Railways



(h) Swimming Pools



(i) Error Detection

Figure G.15: ROC curves of confidence measures for novelty detection and for error detection

H Hyperparameter Search Results

Hyperparameters were searched in the following ranges:

1. GMM
   - Components: `1 - 10`
2. OC-SVM
   - Kernel: `Radial Basis Function (RBF)`, `poly`
   - Degree: `1 - 3` (only for `poly`)
   - Nu: `1e-4, 1e-3, 1e-2, .1, .3, .5`
3. Density Forests:
   - Maximum depth: `2, 3, 4, 5`
   - Minimum IG: `0, .3, .7`

Hyperparameters for both GMM and OC-SVM are described in the documentation of the scikit-learn implementation [37].

The minimum Information Gain parameter is rather difficult to tune, since the IG usually becomes smaller for higher-dimensional data. Tuning it correctly can however still have an important effect by avoiding unnecessary splits (table 3).

Hyperparameter search results for the MNIST dataset and for the Zurich dataset using the hyperparameter search scheme discussed in section 5.6 and the hyperparameter ranges listed above are represented in tables H.1 and H.2.

Method	GMM	OC-SVM			Density Forest	
Class	Components	Kernel	Degree	Nu	Depth	Min. IG
0	8	poly	15	.1	2	.7
1	8	poly	15	.01	2	.7
2	9	poly	5	.5	4	.3
3	9	poly	13	.01	4	.7
4	4	poly	3	.3	4	.7
5	5	poly	3	.5	5	0
6	9	poly	15	.01	5	0
7	6	poly	15	.1	2	.5
8	8	poly	15	.1	2	0
9	7	poly	9	.3	4	.3

Table H.1: Best hyperparameters for the MNIST Dataset

Method	GMM	OC-SVM			Density Forest	
Class	Components	Kernel	Degree	Nu	Depth	Min. IG
Roads	3	poly	2	0.001	1	0
Buildings	5	poly	1	0.001	3	0
Trees	4	poly	1	0.010	1	.7
Grass	3	poly	1	0.001	3	0
Bare Soil	9	poly	1	0.500	1	0
Water	3	poly	1	0.001	1	0
Railways	9	poly	3	0.500	3	.7
Swimming Pools	3	RBF	-	0.500	3	.7

Table H.2: Best hyperparameters for the Zurich Dataset

Regarding OC-SVM, in most cases, a polynomial kernel of degree 1 was found optimal. This makes sense, since a neural network performs the task of making data linearly separable.

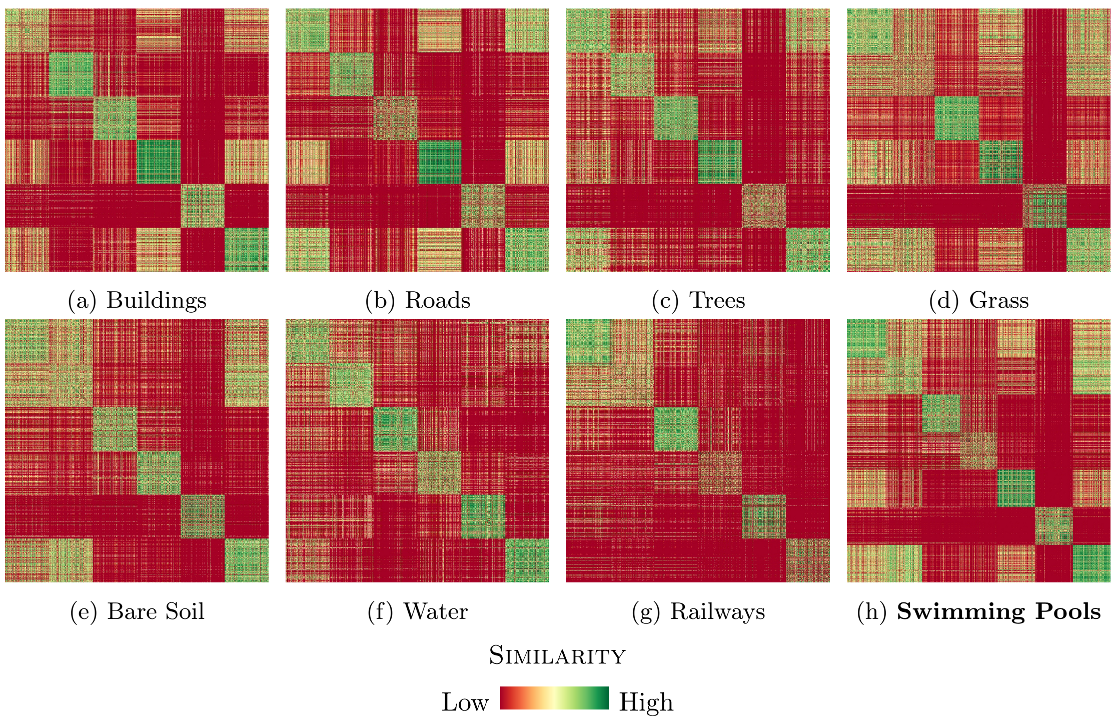


Figure H.1: RBF Kernel visualizations for One-Class Support Vector Machine in Zurich dataset. Kernels were applied to a class-balanced subsample of training activations belonging to the seen classes. Contrast stretching has been applied to the images of the polynomial kernels to highlight more local variation. Best kernels found for each class using hyperparameter search are labelled in bold.

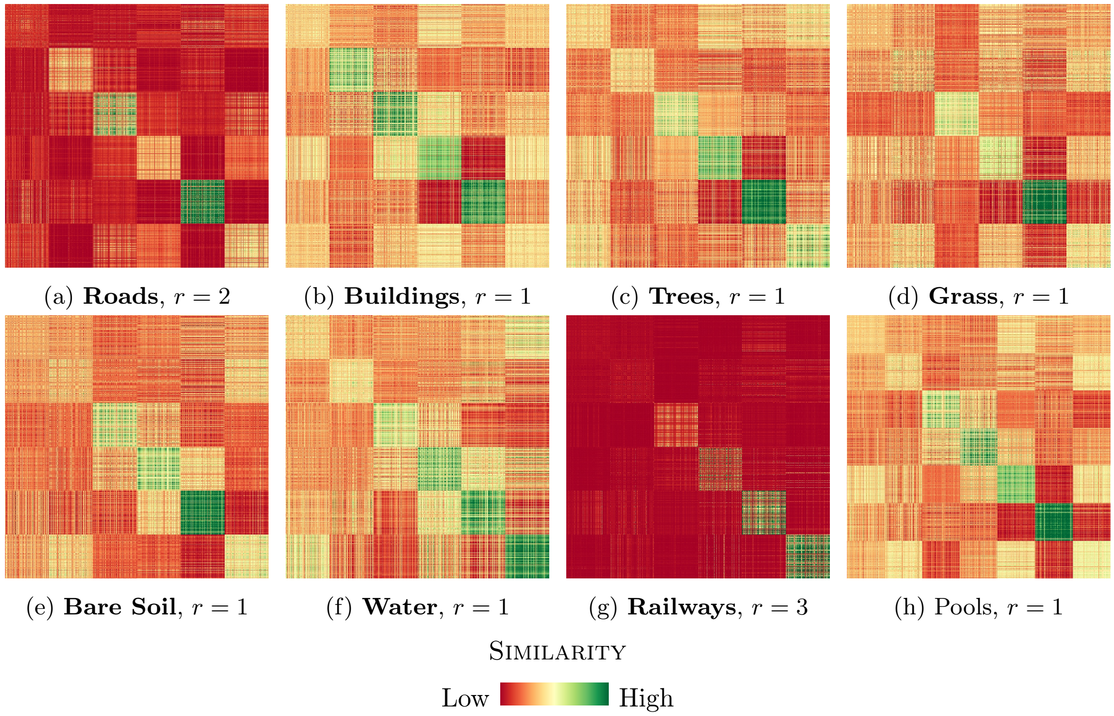


Figure H.2: Polynomial kernel visualizations for One-Class Support Vector Machine in Zurich dataset with best degree r. Kernels were applied to a class-balanced subsample of training activations belonging to the seen classes. Contrast stretching has been applied to the images of the polynomial kernels to highlight more local variation. Best kernels found for each class using hyperparameter search are labelled in bold.