

Master's Thesis in Artificial Intelligence

Change Detection and Semantic Segmentation of Historical Maps to Detect Indicators of Soil Contamination

Author

Quintess Barnhoorn
s4993314

Internal supervisor

Prof. Dr. Marc van Kreveld
Utrecht University

External supervisors

Hester Bijen MSc.
Gerrit van Tilburg MSc.
TAUW bv

Second assessor

Dr. Tim A.E. Ophelders
Utrecht University



Faculty of Science
Utrecht University
The Netherlands
July 1, 2022

Abstract

Historical maps are commonly used as a source of information to find indicators of soil contamination. The Netherlands has an extensive collection of historical maps, which makes it possible to analyse maps of any location from up to 200 years in the past. Because the analysis of the maps is time-consuming, an automated solution is sought. This thesis makes the first step in this regard by semantically segmenting 90 years of historical maps and detecting changes in the landscape per decade. Automatic change detection with historical maps is a novelty. The methods for this are based on work on remote sensing data. To perform both segmentation and change detection, one general convolutional neural network architecture is proposed, of which several variations are tested. The variations serve to distinguish and compare three main influences on the performance: the first is the influence of multi-temporal inputs on semantic segmentation. Adding the map from a decade after the segmented map to the input did not result in significantly higher performances overall; it is primarily useful in situations where a feature is drawn ambiguously while it is drawn clearly in the decade after it. The second influence is the chosen method to combine multiple maps. The two methods that were tried, namely concatenation and Convolutional Gated Recurrent Units, did not differ in performance for any task. Concatenation is therefore concluded to be the superior method, as it is faster and simpler. The final influence is the choice of performing the two tasks simultaneously or individually. Performing both tasks improved the change detection score while decreasing the segmentation score. Overall, the models learned to generalise features on maps from different decades and locations, with a semantic segmentation F1 score of 88% and a change detection F1 score of 72%. These scores are significantly higher than those of the baseline models based on random forests, which shows the added value of using neural networks for tasks that benefit from context information.

Contents

1	Introduction	3
1.1	Background	3
1.2	Soil Sample Planning	3
1.3	Indicators of Contamination	4
1.4	Problem Description	5
2	Literature	7
2.1	Map Analysis	7
2.2	Remote Sensing	10
2.3	Image Processing with Neural Networks	12
2.4	Discussion	18
3	Methods	19
3.1	Data	19
3.2	Models	21
4	Experimental Setup	24
4.1	Parameters	24
4.2	Evaluation	24
5	Results and Evaluation	26
5.1	Speed and Memory Consumption	26
5.2	Segmentation	27
5.3	Change Detection	34
5.4	Segmentation and Change Detection	37
5.5	Overarching Results	37
6	Discussion	39
6.1	Limitations	40
6.2	Practical Recommendations	41
6.3	Future Research	42
7	Conclusion	43
	References	44
A	Tables	50
B	Semantic Segmentation	50
C	Change Detection	54

1 Introduction

1.1 Background

TAUW is a consultancy and engineering bureau that focuses on environmental topics and sustainability. The largest subdivision of TAUW is concerned with the inspection and analysis of the soil. This analysis concerns several domains such as soil quality, groundwater levels and quality, and contaminants like lead or asbestos. These analyses are commissioned by third parties for specified areas of land, and the reasons for them differ considerably. For example, they can be used to determine the possible land uses of a plot, to estimate the value of a plot before selling it, or to estimate the risks of new projects. In cases where dangerous contaminants are involved, legal responsibilities and liability claims come into play, which increase the urgency of determining the exact cause and location of the contamination. In order to get an estimate of the substances that are present in the soil, samples need to be taken at the site. The exact locations where soil samples should be taken have to be planned and marked on a map prior to the collection of the soil, in order to ensure that a representative sample is taken and that no important information is missed. The process of choosing the location for such samples is done manually; the soil advisors check various sources about the site before choosing the recommended sampling locations. These locations are then given to the field workers who collect the samples.

Manually choosing the sampling locations ensures that the analysis of the site is customised to the exact needs of each project, and allows for the use of different sources of information depending on the need. Nevertheless, there are parts of the planning procedure that are repeated for each project. One of these is the inspection of historical maps to determine what the land was used for in the past. Any landmarks or changes in the land use that could be significant are marked on the map to ensure that those locations are sampled thoroughly. The analysis of the maps is done by hand, and involves looking at all maps throughout the years that depict the site. In order to speed up this repetitive task, TAUW is looking into automating it.

The practical motivation for this research is the wish to create a model that can help soil advisors with the analysis of historical maps of different time periods. The academic motivation is that there exists no common approach to solve such a task for this kind of data, and that finding a solution for this problem could be useful for researchers in many fields. Furthermore, there is a large database of historical maps of the Netherlands available, which allows for a wider approach than just a case study with one time period or location as is usually the case. The first step in this work is to find out what kind of information needs to be extracted from the maps, which is explained in the rest of this chapter: Section 1.2 first explains how the planning of soil sampling is done in broad terms. Section 1.3 then explains what planners look for on a historical map. Finally, Section 1.4 presents the exact problem description and research questions of this thesis.

1.2 Soil Sample Planning

This section details what the current workflow is when deciding a sampling plan, and what elements influence the decisions in this process. The sampling locations for a project are decided based on different factors, depending on the situation and the aim of the project. The most common types of soil investigation are the following [64]: geotechnical investigations are concerned with the soil types and the structure of their layers, as well as the groundwater. The second type is biological investigation, which concerns the organisms and organic materials that are present. This information is valuable for determining the fertility of the soil. The third type is environmental health investigation, which concerns the presence of contaminants and harmful substances in the soil. This is important information for the health of the population and the environment, and such investigations are frequently performed when the owner or use of a plot changes. While there are many other types of soil investigation, for this paper we shall focus on the third type: environmental health investigation.

Contamination in the soil is for a large part caused by human activities, from depositing waste to pesticides that enter the soil through rain [47]. The causes of the contamination can often not be seen when looking at the site, either directly or through aerial photographs. This is because the contamination may have occurred a long time ago; many contaminants do not degrade naturally, or only do so very slowly. Depending on the composition of the contaminant, the soil type, the pH levels and the drainage, harmful substances can spread across the land or stay at the same

location for a very long time [48]. Because the properties of a plot are generally not homogeneous and external conditions such as rainfall are not known, the spread of contaminants cannot be predicted entirely even if the source location of the contamination is known. The sampling needs to be planned intelligently in order to detect not only the contamination at the source location, but the spread across the site as well. If it is not certain whether or where there is a contamination, a sampling plan needs to be drawn up that can detect them with a reasonable probability.

The choice of the sampling locations is not just left to the expertise of the soil advisor. The norms of handling soil investigations in the Netherlands are recorded by the Dutch Normalisation Institute (Nederlands Normalisatie Instituut, NEN). The NEN records norms for all kinds of domains, not just the soil investigations. These are guidelines defined by commissions made up of parties involved in the market, and are not legally binding. Nevertheless, they are followed in order to ensure the quality of the investigations. The NEN norms are divided into separate documents for each kind of investigation, such as asbestos investigations in debris or water bed investigations. The document for exploratory investigations is the most relevant one for this research, as it concerns the sampling for possible contaminants in the soil. For regular exploratory investigations this promotes an even spread of sampling points, a given set of drilling depths and the number of sample points depending on the size of the plot [54]. When there is a reason to suspect that there is a contamination, the number of required sample points increases. If the location of the contamination is also known, the sampling points should cover the area of the contamination; otherwise, they should be spread evenly but in a denser pattern. In short, the number of sampling points and their general pattern is dictated by the NEN norms.

The exact locations of the sampling points are chosen by the soil advisors themselves. There are several approaches to making a sampling pattern, and often multiple are used for a single site [19][76]. Firstly there is judgement sampling, where the sampling locations are chosen based on historical and visual information, and the expert judgement of the sampler. The reliability of this approach relies on the completeness of the information and the knowledge of the sampler. This always be used to some extent when sampling for contaminants with suspicion. A general location chosen with judgement sampling can then be combined with systematic approaches. These include grid sampling, where the area is divided into evenly spaced blocks, or transect sampling which samples in a line. If the area is heterogeneous, stratified random sampling can be used to divide the area into sample areas with different sizes and shapes based on their characteristics. These techniques can be combined in order to match the situation at the sampling location. For example, a transect can be laid across an elongated contamination, while the rest of the terrain is sampled with a systematic grid. The indicators that show where possible contaminants are located are discussed in the next section.

1.3 Indicators of Contamination

The choice of the sampling location is based on the norms dictated by NEN, knowledge of appropriate sampling approaches, and knowledge about any suspicious areas with respect to contamination at the site. The suspicious areas are determined by knowledge about how the land was used in the past. The kinds of activities and landmarks that are indicators of contamination have been compiled in the Netherlands by Rijkswaterstaat, a part of the Ministry of Infrastructure and Water Management. This list, called the Uniform Source Classification (Uniforme Bron Indeling, UBI), contains all indicators of contamination along with a UBI class which denotes the expected level of contamination. These range from 1 to 8, with 1 denoting the lowest chance of serious contamination and 8 the highest, along with an urgency indicator meaning that action is required immediately [59]. On the list are currently about 1350 activities and distinguishes between different types of factories, landfills with different contents, agricultural uses, types of infrastructure, and many more categories. In order to find out which kinds of activities have occurred at the site throughout the years, historical sources need to be consulted; specifically historical maps. Not all sub-categories can be determined from maps; they are often too specific to be recorded on a map, which only show the general use of the land.

The general categories of interest when doing a preliminary investigation are listed below, along with the kind of contamination that they might indicate and what they look like on maps.

- **Filled up ditches and waterways:** This category is mentioned first because it is the most commonly searched indicator. When ditches are no longer in use, for example if meadows are

exchanged for housing or industry, the ditches are filled up with dirt or waste. Waterways can be forced to change course, in which case the old riverbed has to be filled as well. The UBI class of this category ranges from 1 in case normal dirt is used, to 8 in case industrial waste is used. In case the filling is not known the class is 2.

- **Artificial elevations and landfills:** While these two categories are separate on the UBI list, they are mentioned together because they often look the same on a map: either with height lines or with small stripes around them. It often cannot be seen whether such an elevation is artificial or natural, which means other sources are required as well. They have relatively high UBI classes; 6 for artificial elevations, and 8 for landfills. This is because it is almost certain that the waste contained in them is harmful for the environment.
- **Orchards:** Orchards are at risk because of their probable use of pesticides and herbicides. Their UBI class is therefore 5. Orchards are often well-documented on maps, and therefore easily retrieved. Other forms of agriculture are also on the UBI list, but they are not always visible on maps.
- **Roads and railways:** These categories are both easily found on maps, as they are some of the most important features of a topographic map. They are also likely at the same location as they are now, especially railways. As for contamination, the foundation of a road can contain debris and possibly asbestos. When there is a suspicion of asbestos the UBI class is 7, with normal debris it is 5. Contamination from railways can come from wear and tear of the wheels, overhead cables, rails and the maintenance of the tracks [58]. The threat of these contaminants is not exceedingly large, and the tracks are regularly monitored, giving it a UBI class of 4.
- **Industrial terrain:** Industry is a large polluter, which is reflected in the fact that the majority of the items on the UBI list are different types of factories. These can range from a UBI class of 1 for fur processing to an 8 for chemical industries. The type of industry can sometimes be derived from the names written on the maps; otherwise it only indicates that it is some kind of industrial terrain or even just that there are buildings. In the latter cases, other sources have to be consulted to find out what was manufactured there.
- **Buildings from 1945-1993:** This period is suspicious because of the widespread use of asbestos in construction projects at the time. Any building that was built in this time-frame could contain asbestos, and any demolished building from this period is marked with a UBI class 6. Because maps are not remade every year, particularly as one goes further back in time, the exact building date may not be visible from the map alone. Still, they can provide an estimate of when new buildings were erected. If more detailed information is needed, the Dutch Base Registration of Addresses and Buildings (Basisregistratie Adressen en Gebouwen, BAG) can be consulted.

It should be noted that if one searches for these indicators on historical maps, they are primarily looking at indicators that are not there in the present; the advisor is likely already aware of those. An exception to that rule are buildings, as the building year needs to be known before an estimate of their contamination can be determined. In fact, any significant change in the land can be important to know, as land changes go hand-in-hand with the moving of soil, either to the site or from it. It is therefore important to monitor the indicators mentioned above as well as any changes in the landscape, whether they involve the indicators or not.

1.4 Problem Description

In the previous subsections, it was discussed how a soil advisor plans the locations of soil samples and how suspicious landmarks are discovered using historical maps. This thesis focuses on automating the task of extracting information from historical maps to assist the soil advisors in the planning process. Specifically, a tool is desired that can identify and report suspicious landmarks and changes in the landscape. The landmarks that are identified are water, orchards, and buildings. These three are chosen because they commonly occur on maps, are useful for the soil advisors, and have distinct characteristics. These characteristics provide different challenges, which give insight into how the models react to certain types of features. On the maps in question, the waterways are very narrow and long, which requires high precision and information from the context in order

to classify them properly. Especially ditches, which are not always coloured, require context information to separate them from other black lines. The orchards are marked with a pattern that does not cover the whole space, meaning that the boundaries of the area need to be found by the model. Finally, the buildings are quite small but filled in. They require high precision, and the model needs to distinguish them from letters on the map which are also black and of a similar size. This category also covers industrial terrains. Aside from the indicators, any significant change in the landscape are reported as well. Figure 2 illustrates the desired segmentation and landscape changes for a given input.

The models have to assign each pixel to one of the segmented classes, and determine whether it has changed or not. To do so they need information about the context of the pixel, and learn to generalise the features of maps of different time periods. To accomplish these tasks, a sub-category of neural networks called convolutional neural networks (CNNs) is used. These machine learning models can learn to process image data and return useful information about them based on training examples. Within this basic framework, there are three aspects which we can vary: the input that goes into the models, the architecture of the models themselves, and the output that it must learn to produce. The first option, namely the input, is at minimum a single map to be segmented semantically or two maps between which the changes must be detected. While multiple maps are not necessary for semantic segmentation, the maps of other decades may provide additional information that is useful to the current decade. The influence of multi-temporal inputs on semantic segmentation has not been explored before, which makes it the first topic that is explored in this thesis. For both change detection and the expanded semantic segmentation, a way to combine multiple maps within a single model is necessary. This brings us to the variation in model architecture: two ways of combining multiple maps are investigated, namely concatenation and a recurrent neural network (RNN). Concatenation simply adds the maps together, resulting in a fully convolutional neural network (FCNN), while a recurrent unit has a more complex way of combining temporal inputs. While both methods are used in the literature for these kinds of tasks, there is as of yet no consensus as to which is preferred. The difference between these two methods is investigated for both tasks. Finally, there is the output that the models have to return. There are two tasks that need to be performed, but the choice remains to perform them simultaneously or individually. The difference between these two conditions is the final point of interest. Concretely, this thesis answers the following research questions:

1. Does input from maps of other time periods make FCNNs and/or RNNs perform better at the semantic segmentation of historical maps than if they would only receive input from the current time period?
2. Does an RNN perform better than a FCNN at the semantic segmentation and/or change detection of historical maps?
3. How does learning both semantic segmentation and change detection simultaneously affect the performance of the models compared to when they learn them separately, and does an RNN perform the combined task better than a FCNN?

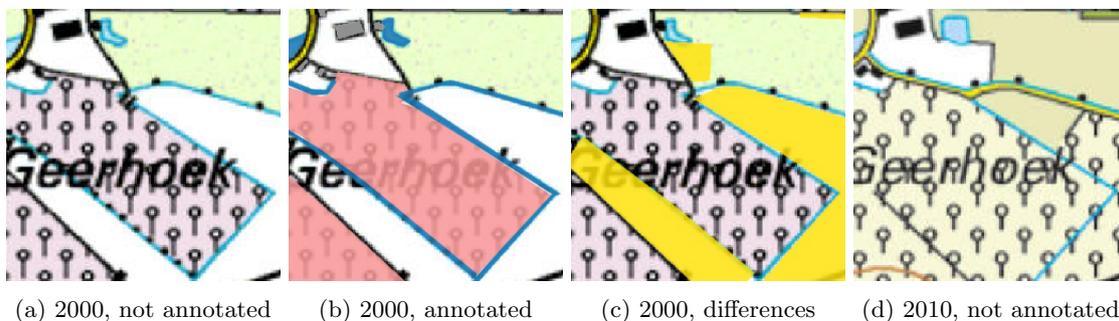


Figure 2: Example of (b) the desired segmentation and (c) the differences between the maps. (a) and (d) show the original images.

2 Literature

This section discusses relevant literature related to the analysis of historical maps, and the image processing techniques that are relevant to the topic. To the knowledge of the author there has been no research yet in the application of automated analysis of historical resources in the field of soil analysis. Section 2.1 starts with a general exposition of the different challenges and techniques that are encountered when digitising historical maps. Section 2.2 discusses the techniques that are commonly used in the field of remote sensing. The segmentation of maps by land use and detecting the changes in the landscape are common topics when working with remote sensing data. Although the data sources differ, many of their techniques are applicable to the current topic. Section 2.3 explains the general techniques that are used to extract meaning from images with neural networks, and relates this back to work in map analysis. Finally, the lessons that can be learned from the literature are discussed in Section 2.4.

2.1 Map Analysis

Historical maps are relevant in many branches of research such as history and social or environmental studies. They are an important source of geographical and political information, and this importance is highlighted even more the further one goes back in time. The large-scale digitisation of these maps started in the 1980s [16], and has led to a large amount of available scanned maps distributed over various repositories and databases. Still, the information stored in these maps is difficult to obtain in practice. They are only scanned pictures, and it is therefore not possible to look up their information through a search engine or the like. The points on the map are often not linked to coordinates, or the given coordinates might not be correct; the scale of the map might not match the distances found in the real world. This not only complicates relating maps to the real world; it also means that maps of the same area cannot always be compared directly. This comparison is hindered further by the fact that different representations are used in different maps, which requires cross-referencing the legends and cartographic practices; assuming that a legend is available. These different challenges of the digitisation of historical maps are explained in more detail below. First, the extraction of interesting landmarks and features on maps is discussed in Section 2.1.1. Secondly, the comparison between different maps is discussed in Section 2.1.2. Finally, the challenges of using neural networks for historical maps are discussed in Section 2.1.3.

2.1.1 Segmentation

Segmentation of historical maps can be done either with features that correspond to real-life objects such as buildings, or with abstract features such as parking symbols. The term ‘feature’ on a map refers to anything that was put on the map with an intended meaning: the land use, cities, roads, points of interest, soil composition, and so on. Extracting these features from a scanned map gives more semantic information about what is displayed on the map. Very old maps are relatively difficult to analyse automatically, so much work is still done by hand [42]. The georeferencing of recent paper maps is often better and more consistent, which makes automated approaches more viable. Finding abstract symbols is relatively easy because they always look almost exactly the same. This task is done both semi-automatically and fully automatically. Semi-automatic approaches often use SURF (Speeded Up Robust Features), which needs an example image and finds all similar symbols on the map by moving a sliding window with twice the size of the example over the map [5]. The candidate list can be further refined with histogram matching. Histogram matching makes a histogram of the occurrences of all pixel intensities in the candidate image, and compares that to the histogram of the example [46]. A more general description of the shape can handle more diverse drawings within the same map, such as the recognition of drawn trees in [36]. While an advantage of the above techniques is that not much training data is needed, a disadvantage is that each map needs its own hand-picked examples. When there are interfering symbols or different scan qualities, it does not hold up. Additionally, they cannot be used for geographic features which differ in shape. There are several segmentation algorithms that split up the image into regions that differ in shape and size. Region-growing segmentation uses the colour of nearby pixels compared to the mean of the whole to segment a picture. An example of this is the work of Shaw et al. [71], who extract the shapes of lakes from old maps in this way. A downside to such methods is that they need an additional step in order to classify the regions. In order to do that, or to do both the segmentation and the classification in one step, recent work

has been done in the application of machine learning to the field. Many GIS programs provide packages for the automatic segmentation and classification of maps, which are generally used on satellite images. Object-based Image Analysis (OBIA) divides the whole map into segments based on the similarity of areas, which are then classified with a range of machine learning techniques. It needs annotated points in the map in order to start classifying, and the segmentation needs to be tuned to the features of the map in question. This has also been used with historical maps, with an additional step of removing any text from the map [92]. Convolutional neural networks (CNNs, see Section 2.3) are popular as well for map segmentation. They are useful for detecting the general location of features that have no clear border, such as road intersections [67]. They are also used for semantic segmentation, as can be seen in [11]. The paper details a recent competition in the segmentation of historical maps; all of the participating methods use some form of CNN.

A specific type of feature that is very useful on a map is the text. This can be used to georeference the map; to connect it to real-world locations. When talking about the recognition of text in complex scenes, the term ‘scene text recognition’ is used. One possibility for scene text recognition is to use a CNN for the whole process of extracting the letter locations and extracting the characters themselves, as is done by Xie et al. [89]. A more popular option is to first extract the letters from the image with a CNN, and then use a recurrent neural network (RNN, Section 2.3.3) to decipher the characters. Examples of this are the network architectures of Cheng et al. [14] and Loginov [45]. Such an approach is also effective for map content, as is shown by Weinman et al. [87]. Once the characters and/or words are extracted from the map, they need to be linked to an actual place name. When the name consists of multiple words the two first need to be added together before they can be matched to their real counterpart; such work can be aided by information about the fonts and capitalisation, as is done by Lin et al. [41]. This is taken further by Chiang et al. [39], who use a combined network where a textual predictor decides which separate words belong together based on fonts and capitalisation, and a visual predictor which is concerned with the neighbourhood of the words. A gazetteer containing historical place names can be used in a Bayesian model together with the CNN output to output correct place names as is done by Weinman [86]. The methods above all involve finding specific features on single maps. The next section explains more about how the comparison of such features over time is typically done.

2.1.2 Change Detection

Detecting changes on historical maps allows one to study changes in the landscape that were made much further back in time than is possible with satellite data. Because maps are relatively difficult to classify automatically, this is often done by hand, after which the comparison is done automatically. Examples of this are the work of Li et al. [38], who combine data from historical maps and satellite data to map the urban growth around Beijing, and Chen et al. [13], who map the general land use of Taiwan with similar data. Picuno et al. [62] also classify the land use manually, but use machine learning software to map the changes between the classified maps. Others use more automated approaches, such as Liu et al. [42] who use random forests for the classification of general land-use in Chancellorsville, USA and analyse the proportions over the years. Another example is the work of Fuchs et al. [21], who use maximum likelihood estimation to classify maps of central Europe based on colour, and then use an automated spatial allocation model to combine different sources. Two general trends can be seen in the literature: firstly, they classify everything on the map into general classes, after which they compare the differences between those classifications. Secondly, the use of machine learning techniques is not very common. Neural networks in particular are rare in this field. Possible reasons for this are that the studies need a high level of precision, or that the maps are too heterogeneous to process with common automated techniques [77]. In the previous section it was seen that machine learning is frequently used for the extraction of features from maps, but it seems that this does not hold for change detection. Bringing the techniques from other fields to the field of map analysis could ease the burden of ecologists and historians considerably. In Section 2.2.2 some of the techniques that are used with remote sensing data for this task are detailed. These papers can be used as an inspiration for work with historical maps. When creating models with historical map data one encounters additional hurdles, which are outlined in the next section.

2.1.3 Using Neural Networks for Historical Maps

Using neural networks for historical map segmentation comes with some difficulties which one needs to take into account. First of all, there is not a lot of training data available. For real-world photographs there are very large data-sets available online, as well as networks that are trained on them. For scanned maps this is not the case, which means that the researcher has to create training data themselves. One can inflate the number of training images by manipulating existing training samples, such that an annotated image can be used multiple times [87]. Current-day data about the locations of landmarks can also help to create training data automatically as in [81]. A common approach to solve the problem of insufficient training data is to use transfer learning: finding a network trained on photographs or another domain, and then using them in a network for scanned maps. One does need to take into account that photographs and scanned maps have different characteristics, such that the features on maps are relatively small compared to objects in a photograph; coarse classifiers might not be able to locate them accurately. Because of this, enough training data for the scanned maps is still needed, and the borrowed model still has to be trained from the shallow layers to perform well [15]. Some therefore choose not to use transfer learning after all, as the differences are too great [13].

Secondly, there are often interfering elements on maps. This includes text elements, paper folds, and edges where different maps have been added together. Some examples of these can be seen in Figure 3. While a neural network can learn to work around distractors, it needs a large enough receptive field to see from the context which lines are from map features and which are not. An additional difficulty arises when maps from different times and sources are used. The cartographic designs, scan qualities and the archival conditions of the maps can differ across the maps. This large variability needs to be taken into account for design choices such as the complexity of the model, multiple models per cartographic style, or explicitly balancing the different styles in the training data [82]. It could be that some features are clearly visible on one map, but not on the other. One needs to balance the need for generalisation and the extent to which a network can do so. For example, in [22], a network based on U-Net (see Section 2.3.2) is used in order to find archaeological sites on historical maps. For each depiction of ruins, such as mounds and the Arabic word for settlement mound, a different classifier was trained. This makes the search for different representations of the same element explicit, rather than letting the network figure out which set of representations belongs to the same class. On the other hand, this does mean that several models have to be trained, which takes more time and computational power.

In short, when using a neural network for finding features on scanned maps one needs to keep the following in mind: the resolution of the output needs to be high enough to find the small features, the training set needs to be balanced to cover all cartographic variations, and pre-processing and/or transfer learning might be needed to compensate for insufficient amounts of data. With this in mind, we can turn to the exact techniques that are used to build such a model.



Figure 3: Examples of distractors. Red: overlapping text. Yellow: distortions at the border of adjoining maps. Green: different representations of the same feature. Blue: grid lines.

2.2 Remote Sensing

Remote sensing (RS) data pertains to data that is collected from a distance, such as from satellites or aircrafts. Measurements such as radar also fall under RS. The most common type of measurement that is taken for landscape monitoring is reflected sunlight. This can vary from photographs with three bands for the visible colour spectrum to hyperspectral data which can have more than 200 narrow colour bands. These are not limited to the visible colour spectrum; short-wave and thermal infra-red are also used [24]. With this kind of data, it is possible to gather information about areas that are difficult to traverse otherwise. Although the study of this kind of data is not directly related to the analysis of historical maps, the usage of the data is: to determine the land use of an area and to monitor the changes in the landscape are both frequent aims of studies with RS data. A summary of techniques used for determining the land use is given in Section 2.2.1, while Section 2.2.2 discusses the techniques used for change detection.

2.2.1 Segmentation

Segmentation of satellite images is mostly done to classify the land use of an area. Machine learning techniques are commonly used for this task. Especially in the past, it was most common to classify the pixels separately. These pixel-based methods previously had difficulty with low-resolution images, as these could contain more than one land cover type. The current-day satellite images are of a much higher resolution and do not have this problem, as most real-life objects are distributed over multiple pixels [26]. By using the large spectral range of RS data it is possible to classify pixels with only minor input from the surrounding landscape. For example, Sesnie et al. [69] use a large amount of spectral and spatial data to classify each pixel individually by using a decision tree. For some of the parameters a block of 3×3 pixels is used, such as for the local texture. A more recent example is the work of Pan et al. [60], who use a recurrent neural network with spectral data as input to classify single pixels.

Another quite popular method is object-based image analysis (OBIA). These methods first segment the image into objects which ideally correspond to real-life objects and then classify those. GIS software often has options for this kind of classification built-in, such as ArcGIS or GRASS. Object-based methods have the advantage that information such as shape and area can also be used for classification, alongside spectral information. The quality of those parameters depends heavily on the quality of the segmentation, because over- or under-segmentation can lead to inaccurate assessments of the shape and size of an object [43]. The dependency of the classification on the segmentation accuracy is increased when there is only limited spectral information available. An example of object-based classification is the work of Smith [75], who first segments multi-spectral images before classifying them into land-use classes with a random forest classifier. The objects can also represent parcels, as is done by Huang et al. [27]; they first segment the area into parcels, and then overlay it with a grid to produce square tiles as input for a neural network. The weighted average class of those tiles is then used as the class of the parcel. Many more examples can be found in the review of object-based segmentation by Hossain et al. [26]. A comparison between pixel-based and object-based methods is made by Liu et al. [42], who use a pixel-based method combined with maximum likelihood classification and an object-based method combined with random forest classification. The object-based method provides a more clean picture, while the pixel-based method shows a ‘salt-and-pepper’ effect; small islands of a class.

A third option is to classify all pixels at once, without splitting the image into blocks. This has the advantage that context information is taken into account as well. Neural networks are useful for such a task, as they can process many parameters at once and can have many output values. An example of such a method is the work of Kampffmeyer et al. [33], who use patches of 256×256 pixels and classify each pixel at once. The classes that are used in their work are objects such as houses and cars, rather than land-use classes. They also compare this method with a patch-based method which only classifies the centre pixel. The full-image version performed better.

2.2.2 Change Detection

Change detection pertains to the detection of changes over time in a region of interest. Such work is done on satellite images and other remote sensing data to document the changes in the Earth's land coverage. This is useful for many applications, including urban development monitoring, land use detection and disaster assessment. Care is often taken to compare images that were taken around the same time of the year to reduce the effect of the seasons and differing sun angles. Radiometric, atmospheric, and topographic corrections are also used to properly align the images and to reduce noise [29]. Change detection can be done in different ways; one option is to first classify both images separately, and then to find the differences in classification. This method is called post-classification change detection. Example applications of the method are tree coverage [79] or general land use [6]. This only works if the classes cover all possible uses of the space; a change in a non-classified part of the image cannot be identified. Another option is to directly compare the images and then output the differences. This can be done either as a binary classification of changed/not changed, or with more specific classes such as changes in waterways or buildings.

Such change detection algorithms can be roughly classified into two steps: First, a quantitative measurement of the difference between the images has to be made. Secondly, a threshold or other method needs to be used to convert the measurement into a binary classification of changed or not changed. The first step can be done by simply subtracting the values of the pixels from one another (image differencing) or dividing one image by the other (image ratioing) [1]. More advanced methods are principal component analysis (PCA) or change vector analysis (CVA) [3]. PCA is normally used to reduce the number of dimensions in high-dimensional data by combining correlated dimensions into single components. In the case of change detection, PCA is performed on the combination of the two images. The idea is that the first components represent unchanged areas, while later components represent changed areas and the final ones are simply noise. Downsides of the method are that it can be difficult to identify which components should be used and that the results are specific to the images that are given as input. CVA computes both the magnitude and the direction of the change between two sets of data. The direction is mostly useful for applications where changes tend to be of the same domain such that they can be readily recognised, for example soil moisture changes [2]. These methods can in principle be used for each pixel pair if the two images are well-aligned. For more robust results, however, it is better to take the surrounding pixels into account as well [63]. For example, each pixel can receive information of the 3x3 pixel block surrounding it, after which the output is applied to the middle pixel [34]. Another option is to first detect the objects in the image with segmentation algorithms, and to compare those [29]. Which method is preferred differs greatly between communities and applications [2][3].

The methods above output some measurement of the difference between the two images. These 'change masks' need to be classified into binary values of changed or not changed. One option when the numeric values are stable is to pick a threshold based on some examples; this is not practical for scene-dependent methods such as PCA, or methods which output multiple values. More specific techniques can be split into unsupervised and supervised methods, which range from statistical to machine learning algorithms. Unsupervised methods are for example auto-encoders or clustering algorithms such as K-means or Fuzzy C-means clustering [10]. Such algorithms are very useful when no annotated data is available. Supervised methods are for example maximum likelihood classification [2], decision trees [30] and neural networks. Examples of neural networks that are used for change detection can be found in Section 2.3.4. It should be noted that both clustering and supervised methods can be used without first merging the images into a change mask; in that case the way in which the inputs of the algorithm relate to each other needs to be learned by the algorithm itself. This can present problems for less complex methods such as clustering algorithms.

RS data has many convenient properties which make these methods viable. Firstly, it contains multi- or hyperspectral data. Such data contains rich information about each pixel, which makes classification possible with very little context data [60]. Secondly, satellite images are likely to be quite similar if no real changes have occurred, aside from changes in the shadows and the weather conditions. Finally, the images have no scale differences as long as they are taken from the same height, meaning that the images can be combined with high spatial accuracy. This means that pixel-wise and object-based methods can be quite successful, as the overlap of the features is very precise. For images that do not have these properties, neural networks can be used as a more robust alternative.

2.3 Image Processing with Neural Networks

In order to detect and locate the important features on the maps and the changes in the landscape over time, convolutional neural networks (CNNs) are employed. The general structure of a CNNs is explained in Section 2.3.1. The task for which they are used in this thesis is segmentation, which requires a specific architecture. This is discussed in Section 2.3.2. Finally, Section 2.3.3 discusses recurrent neural networks (RNNs), which are used to process temporal data, such as images or maps from different times.

2.3.1 Convolutional Neural Networks

CNNs are a type of neural network that deals with three-dimensional data, where information is assumed to be spatially correlated; this mostly comes down to images. The third dimension is generally for colours when images are involved; a ‘depth’ of 3 for the values red, green and blue form the colour of a single pixel. CNNs are loosely based on the structure of the animal visual cortex; the early layers encode patterns and simple shapes with high spatial resolution, while the deeper layers encode more global information which combines the outputs of the earlier layers. In contrast to the visual cortex and conform to most neural network architectures, standard CNNs are completely feed-forward; this means that no information from later layers is relayed back to the earlier ones. A CNN consists of a series of layers, just like a normal neural network. Each layer receives as input an array of numbers called a tensor, applies one or multiple operations to it, and outputs a new tensor. Three main types of layers are used with CNNs: convolutional layers, pooling layers, and fully-connected layers.

Convolutional layers are used to extract features that span multiple elements in the input. This is done by sliding a kernel, which is a weight matrix, over the image. The weights are multiplied with the value at the corresponding element, which are then added together to create a single element in the resulting output. The output of a convolutional layer is called a feature map. An example can be seen in Figure 4. The weights remain constant throughout the convolution, which means that the same pattern is collected at each part of the image. The speed at which the kernel moves over the image is determined by its stride: a stride of one means it moves one element at a time, a larger stride that the weights ‘skip’ some elements. In order to ensure that the feature map remains similar in size to the input tensor, padding is used. Padding refers to adding a layer of zeroes around the input tensor, to ensure that the tensor starts with its middle weight at the top left corner. In this way, each element is at the centre of the kernel once (with a stride of 1). Multiple kernels are typically applied in one layer. Each of those kernels results in a feature map which are combined as channels in the resulting output, serving as the third dimension. In numbers: if the input tensor is of size $n \times n$, and kernels of size $m \times m$ are applied to it with stride s and p padding, then the resulting feature map is a square with length and width $\frac{n-m+2p}{s} + 1$. The third dimension is equal to the number of kernels. When the input has multiple channels, the kernel averages over all channels to produce its feature map; its size therefore becomes $m \times m \times c$. The kernel size, number of kernels, stride and padding are hyperparameters and have to be chosen beforehand. Only the weights in the kernels are learned [90]. The resulting feature maps are put through an activation function before serving as the input of the next layer. Activation functions are important in neural networks to keep the numbers within an acceptable range for computation, and to introduce non-linearity into the architecture. The most commonly used activation function is currently a rectified linear unit (ReLU), which filters out negative numbers by setting them to zero [53].

The second type of layer is the pooling layer. A pooling layer is used to reduce the dimensionality of the feature maps, in order to decrease the number of parameters and to generalise the maps. They have a square filter with a set size, which slides over the input. The two most common types of pooling are max pooling and average pooling. Max pooling takes the maximum value of the elements in its filter and relays that to the feature map. Average pooling takes the average of its elements and returns that. Pooling reduces the size of the image, but it does not change the number of channels that results from the use of multiple kernels in a convolutional layer. An advantage of dimensionality reduction with pooling is that no weights need to be trained for the layer. An alternative for dimensionality reduction which does need trainable weights is using a convolutional layer with a stride higher than 1; an advantage of such a method is that the process is more controlled.

The third type of layer is a fully connected layer. Contrary to convolutional layers, it does not

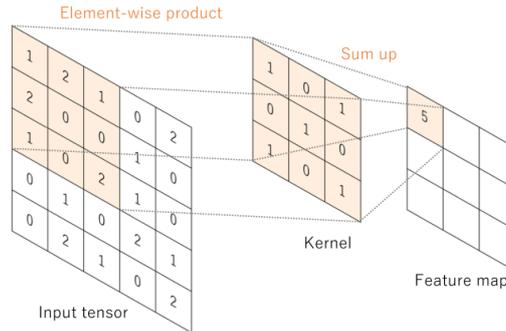


Figure 4: Example of a convolution operation with a kernel size of 3×3 . [90]

take the spatial structure of the data into account; each element has its own corresponding weight. Each neuron in this layer takes a vector as input and multiplies this with its weight vector. It then adds those results together along with a bias term, puts them through an activation function and outputs a single value. This results in an output size equal to the number of neurons. Because this layer requires one-dimensional input, output of the convolutional layers first has to be flattened before they can be used. The flattening comes down to listing all values in the feature maps in a row. A fully connected layer has as hyperparameters the number of neurons and the choice of activation function. The ReLU activation function is typically used here as well. In the case of image classification a fully connected layer is often used as the last layer, with each neuron representing a possible class.

Both the convolutional and the fully connected layers have weights that have to be learned. The purpose of any neural network is to be an advanced function approximator; the function that it has to approximate has to be learned from examples. This involves training the network with many input-output pairs in a supervised manner. First, the difference between the desired output and the output of the network is computed; this is done with a loss function. For simple classification this is commonly done with cross-entropy loss, both for binary and multi-class problems. The goal is to minimise this loss as much as possible, which amounts to minimising the difference between the network output and the desired output. Once the loss is computed, the weights are updated using back-propagation, which uses the gradient of the loss with respect to the weights in order to estimate how much the individual weights have to be changed. The actual change of the weights depends on the optimisation function that is used, which show different behaviours when faced with the same gradients. For more information about back-propagation one can consult [23]. The goal of training a network is to use it on new data. While the network is trained on a training dataset, a test set needs to be kept separately to see whether the network can generalise to other inputs. This is important because the number of weights that is needed for a neural network is hard to estimate from the start. If too high a number is chosen, the network can be prone to overfitting.

A common goal of a CNN is to say something about the input as a whole. This means that the last layer of the network is often a fully connected layer, which results in one or multiple classes. An example of a class domain is the land use that is visible on remote sensing images as done by Huang et al. [27], where a large image is first cut into pieces according to additional data about the street blocks. Those pieces are then classified as a whole, and combined to form a pattern of the large image. Such a technique for finding the locations of classes in an image is only possible if a way to decompose the image is known beforehand. Otherwise, both the locations and the classes in the image need to be discovered by the network. This process is called image segmentation.

2.3.2 Segmentation

Within the context of neural networks, segmentation is commonly combined with classification. When one wants to assign specific classes to the found segments there are some options to consider, which can be seen in figure 5. Firstly, there is semantic segmentation. This assigns a class to each pixel, but it does not recognise to which object that pixel might belong. Secondly there is classification and localisation. This classifies the picture as a whole, but it also shows where the object roughly is with a bounding box. The third option is object detection; here multiple objects

are classified, and also classified as separate objects rather than pixels belonging to a class. The objects are marked with a bounding box. Finally, instance segmentation finds the separate objects, classifies them, and also marks the exact pixels that belong to the object. For the segmentation of historical maps the first option, semantic segmentation, is most appropriate. The reason for this is that the exact locations of the landmarks are important to decide the sampling locations, but the exact object to which the landmark belongs does not matter. The remainder of this section therefore focuses on techniques for semantic segmentation.

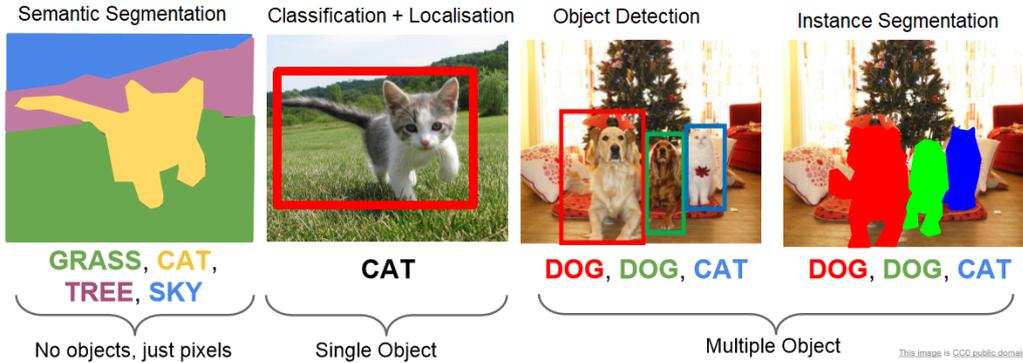


Figure 5: Different options for partitioning a picture into classes. Adapted from [37]

In order to semantically segment an image, the output needs to be as follows: it needs to be the same size as the input image, and the number of channels needs to be equal to the number of classes in order to assign each pixel a class. As we have seen in section 2.3.1, a standard CNN only has layer options to decrease the size of the image. The simplest option is to use a Fully Convolutional Network (FCN), which replaces the usual fully-connected layer at the end of a CNN with a convolutional layer with the size of the input image [50]. In order to get to the size of the input image, the information of previous layers needs to be up-sampled. To prevent the output from being too coarse, *skip connections* are used which transfer the feature maps of earlier layers to the final layer which are then combined to create the up-sampled output. Networks that use a lot of skip connections are also called residual networks. Huang et al. [28] go even further with their network DenseNet and include skip connections from each feature map between convolutional blocks to all future feature maps. Skip connections help to ensure that the resolution remains high after many convolutions, and prevent vanishing gradients in large networks.

Another well-known architecture is the encoder-decoder network, introduced by Noh et al. [56]. Instead of up-sampling at the last moment, this architecture is ‘symmetric’. It first reduces the sizes with regular convolutions, which is the encoder. It then uses *transposed convolution*, also called deconvolution, to gradually up-sample to the intended image size; this is the decoder. A more detailed explanation of transposed convolutional layers can be found in [51]. Such a network still has some trouble with keeping fine-grained information intact, as the encoding process removes spatial information. To remedy this, skip-connections can be added between the encoder and decoder layers with the same size. This was first proposed by Ronneberger et al. who called the architecture U-Net [65].

Rather than increasing the receptive field gradually, one might want to take into account information at both global and local scales at the same time. For this a *multi-scale pyramid* structure can be used [50]. It uses several pooling operations in parallel at different kernel sizes to extract information at different scales. These are then independently convoluted, after which they are all up-sampled and combined again. A representative example of such a network is the Pyramid Scene Parsing Network (PSPNet) by Zhao et al. [93]. It first lets any other CNN create a feature map of the image, after which a pyramid pooling module is inserted. The result of this is then combined with the original feature map and convolved once more before outputting the segmentation.

In order to take a large part of the image into account by increasing the receptive fields, one needs large kernel sizes. It is, however, computationally expensive to use those as they have many weights that need to be trained. A solution for this is to use *dilated convolutions* [91]. These kernels have zero padding between their weights, such that they cover a larger area at a time; the overall size of the kernel can be 5×5 , even though it only uses 3×3 weights; this is illustrated

in figure 6. As long as a stride of 1 is used, no elements are skipped completely. Because these layers can already reduce the dimensions of the image drastically, the use of pooling layers is less essential. This is advantageous, as pooling layers decrease the resolution of the final image. One thing to keep in mind is that the input needs to be padded more in order to properly take the edges of the image into account. Applying convolutions in parallel with different dilation rates can have a similar effect to pyramid pooling. An example of this is the work of Wang et al. [85], who use multiple dilation rates and concatenate their results to improve on the general U-Net architecture.

There are many more advanced techniques in image segmentation, which cannot all be discussed in detail here. Attention-based models first decide which areas of the image to focus on before classifying it. Generative Adversarial Networks use two models, one for creating a realistic segmentation and one for discriminating between the generated and the ground-truth segmentations. Some models are combined with non-neural network approaches, such as active contour models or conditional random fields. A good overview of the possible methods can be found in [50]. An important point to consider is that many techniques that are used in such architectures have been carried over from image classification with standard CNNs. One should be mindful not to assume that techniques that work well for classification, also work well for semantic segmentation [91]. An example is the use of pooling layers; these inevitably decrease the resolution, which can be detrimental to the result. The different techniques that we have seen above that aim to improve the results for segmentation specifically are the use of skip connections, transposed convolution, pyramid pooling, and dilated convolutions. These are mixed and matched with regular convolution and other techniques to form the final architectures of the many segmentation networks available.

In order to train a segmentation network, a suitable loss function has to be chosen. One can use one loss for the entire image, similar to how it will be judged in the end; examples for this are the Jaccard loss and dice loss [31]. These are attractive because one can see during training how well the model will perform in the evaluation. Another option is to compute a loss function separately for each pixel; in this case variants of the cross-entropy loss are often used. The exact variant depends on the distribution of the classes in the data.

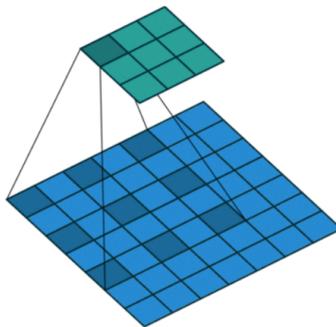


Figure 6: A dilated convolution with kernel size of 3×3 and a dilation rate of 2. The dark blue elements are weighted and added to form the dark green element in the next feature map.[80]

2.3.3 Recurrent Neural Networks

RNNs are neural networks that focus on processing sequential data. They are widely used in areas such as language processing and generation, speech recognition, and providing image descriptions [68]. An RNN has, contrary to regular neural networks, cycles in its structure. It has a hidden state which is updated with information from a timestep and then passed to the next. The originally proposed structure of an RNN is hardly used anymore because it suffers from vanishing gradients and cannot capture long-term dependencies [50]. There are two widely used variants: Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU). Both architectures make use of ‘gates’ to control the flow of information, instead of the original RNN which combined the new input with the output of the previous without any filter. LSTM makes use of three gates: a forget gate which decides what information to keep in memory from the previous output, an input gate which decides what values in the memory will be updated, and an output gate which decides the output. A GRU has only two gates: an update gate which decides what information from the past to keep, and a reset gate which decides what information from the past to forget.

Because the GRU has a simpler structure and fewer parameters, it can be trained faster. The performance of the two models is comparable, although LSTM seems to be more suited for very long sequences [17]. Because the sequences for our application are not particularly long and simple architectures are preferred, only the GRU is discussed in more detail; a comprehensive explanation of LSTMs can be found in [57].

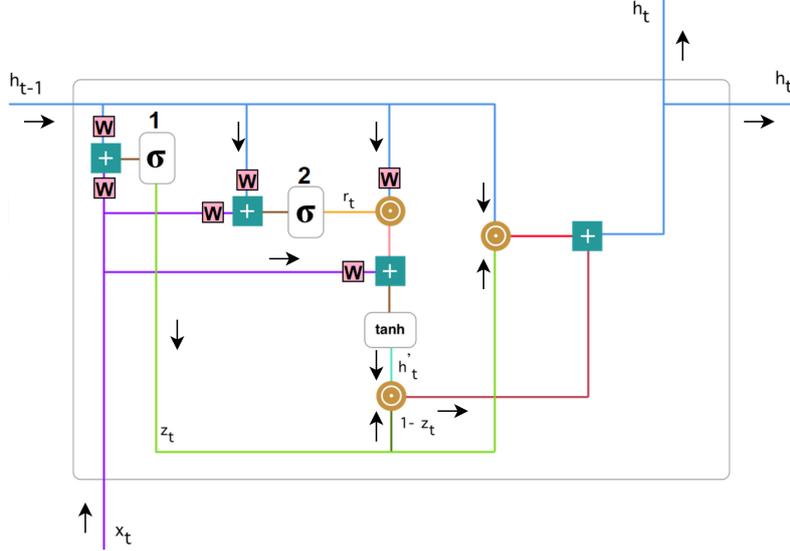


Figure 7: Schematic of a GRU. 1: update gate. 2: reset gate. x_t : input at the current timestep. h_{t-1} : output of the previous timestep. σ : sigmoid function. Yellow circles: element-wise product. Pluses: element-wise additions. Pink W: multiplication with weights. Adapted from [35]

A schematic of a GRU can be seen in figure 7, which is explained here in more detail. The past information h_{t-1} that is passed on in a GRU is the output of the previous timestep; there is no difference between what is kept in the cycle for the future, and what is given as output for the rest of the model at the same timestep. The computation of the new state h_t is as seen in Equation 1. Both parts of the equation use the term z_t . This denotes the output of the update gate, which determines how much information of the previous timestep should be kept. The gate has a weight for both x_t and h_{t-1} , adds them together, and puts it through a sigmoid function to form an output between 0 to 1. These numbers denote how much of the past information is relevant. The exact computation can be seen in Equation 2.

The next part that needs to be computed is the candidate state h'_t , which stores the information from the present combined with filtered information from the past. For this, the output of the reset gate has to be computed. The reset gate decides how much of the previous information should be forgotten; if the output is close to 0, it will read the new input as if it is the first timestep. The reset gate r_t is computed in the same way as the update gate, which can be seen in Equation 3. Once this is computed, the candidate state can be computed by first combining r_t with a weighted h_{t-1} , and then adding that to a weighted x_t . Equation 4 shows the exact computation.

The information from the past h_{t-1} is weighted by multiplying it element-wise with the output of the update gate z_t . This value is then added to the candidate state h'_t , weighted with $(1 - z_t)$, which holds information about the present. The final computation is seen in Equation 1. The resulting architecture is a recurrent unit that can retain long-term information from sequential data. Each of these units has 6 weight vectors that have to be trained, which have the same dimensions as the input. A layer has multiple GRU units in parallel, which can all focus on a different type of information. One might be reminded of the different kernels in a CNN layer, which are similarly separated.

$$h_t = (1 - z_t) \circ h'_t + z_t \circ h_{t-1} \quad (1)$$

$$z_t = \sigma(W_z^x x_t + W_z^h h_{t-1}) \quad (2)$$

$$r_t = \sigma(W_r^x x_t + W_r^h h_{t-1}) \quad (3)$$

$$h'_t = \tanh(W_c^x x_t + r_t \circ W_c^h h_{t-1}) \quad (4)$$

GRU architecture can also be used in combination with a CNN, for instance to analyse videos [73][74] or magnetic resonance images [25]. These examples use the input from the other images to enhance the segmentation of the current one. One can see that in the video images, the shared movement of object parts give more information about whether they belong together. In some cases, the output of a CNN is first converted into 1D data before running it through the RNN, in which case the structure described above is still used. The pixels can also be run through a GRU independently, using the colour spectrum as the sequential dimension [60][78]. When a 2D image has to be processed by a GRU a large number of weights have to be trained, as the weight vectors of a standard GRU are fully connected. A solution to this was proposed by Ballas et al. [4], who replace the weight vectors in the GRU with convolutional kernels; they call it a GRU-RCN (GRU Recurrent Convolution Network). For each channel in the input, a different set of GRUs is needed. The number of weights therefore becomes $6 \times k_1 \times k_2 \times O_x \times O_h$, where $k_1 \times k_2$ is the kernel size, O_x is the number of channels in the input, and O_h is the number of GRU units per channel. The input of the layer can be either raw image data or output from another CNN; [4] uses the latter. An LSTM version was also proposed by Shi et al. [72], which does not use a separate network as input and only has LSTM-CNN layers.

2.3.4 Change Detection

Neural networks are relatively robust, and can deal with noisy data. This is useful for change detection, where much noise is present due to misalignments of images and irrelevant differences. Additionally, they can infer semantic information from the images to generalise the representations, which is especially useful with images that have different graphical styles. Many of the architectures that have been discussed before could be used for change detection. One question still remains: how does one give multiple images as an input to a network? A first option is to stack them on top of each other as if they were channels of the same image. An example of such a network is that of Liu et al. [44], who stack the images and then use convolutional layers to arrive at a binary output of changed or not changed. A second option is to use image differencing; using the absolute difference of the two images as input. Daudt et al. [9] use a ‘Siamese’ network similar to U-Net, where the two images are first processed in parallel by the encoder network. The decoding part of the network then uses the differences between the two feature maps from the encoding network as input. They also have a version where a concatenation of the two feature maps is used instead of the difference. The difference between concatenating at the start or at the middle of the network is also explored by Lim et al. [40]. Concatenation and differencing can also be used together: Seydi et al. [70] use a network that has three paths, two for the images and one for their differences. The third path gets both the stacked feature maps of the other two and their difference as input after every few convolution steps. A final option for combining the two images is to use RNNs. This can be done by first processing the two images separately with a CNN, and then flattening the feature maps to put them through an RNN [52][12]. They both use a fully connected RNN which results in a fully connected layer to create the output. In both cases there are only two timesteps for the RNN; one for each image. Such a model can also be extended to use multiple images, for example to predict future land use [7]. While the advantage over the CNN methods is not apparent when two images are used, the possibility to generalise to more images makes RNNs a good choice when multiple timesteps are concerned.

There is no clearly favoured method for change detection, although there are indications that the RNN methods generalise better for data with irregular time intervals [8]. All of the methods shown above focus only on either change detection or precise classification, and have no other outputs. There are indications that networks that have both similarity and classification outputs perform better on the similarity task than those that give only a similarity output [94][10]. This could be because the network has more feedback on what features make the pictures different. While both papers have a model that outputs the changes and the classification at the same time, Caye Daudt et al. [10] also have a model which is first trained on the classification, and re-uses the trained classification layers in the change detection task. That version performed even better than the combined task, suggesting that a sequential training order is beneficial.

2.4 Discussion

Several lessons can be learned from the existing literature regarding map analysis and image processing. Based on previous studies, the difficulties of the models come from obscuring elements on the maps such as grid lines and text, as well as the high precision that is needed for the small elements. To mitigate these problems, models need to be built that incorporate information about larger parts of the map to distinguish between distractors and features, and that retain a high resolution of the image. To accomplish this, a CNN can be used for the segmentation, which is a common approach for extracting features from both maps and satellite images. The high resolution of the image will be retained by using dilated convolutions and skip connections within the model architecture. To circumvent the distractors even more, information from other maps of the same location can be used. As long as the landscape has not changed, this could help to fill in missing information where one map has distractors but the other has not. One can go even further with an RNN to include images from more distant timeframes, which is not feasible with fully convolutional methods. Supporting the semantic segmentation with maps from other time-frames is the first of the tasks that will be compared in the evaluation. A final problem concerns the lack of training data for historical maps. One option to solve this is to use transfer learning, but this comes with its own share of problems because of the large differences in the data domains. Because of this, the model will be trained on the map data only. To avoid overfitting on the smaller dataset the chosen network will be kept relatively small, and data augmentation will be used to inflate the number of datapoints.

Aside from segmenting the images correctly, the changes in the landscape also have to be detected. Change detection is usually not automated when working with historical maps. Satellite data, on the other hand, has a longer history of automated change detection. Based on the model architectures that are used there, both our own models and the baseline models can be constructed. One point of contention remains as to how the maps have to be combined; concatenation, subtraction, division and recurrent units are all used. Division is more frequently used in models where the combination is done before putting it in a machine learning model, which makes it well-suited for the baseline models. The neural networks will compare concatenation and recurrent units, as these are the most common within neural network models. The trade-offs between these two methods will be the focus of the second task in the evaluation.

Because this research stands on the intersection of several fields of study, it can contribute to several of them. Firstly, the soil remediation and analysis branches can benefit from a more automated analysis of historical land use. This can contribute to the swift and thorough analysis of the soil, which in turn is good for the environment. Secondly, researchers in ecology and history can benefit from more automation in the detection of changes and features in the landscape. While feature detection is automated relatively frequently, change detection is not. The added value for feature detection is that the existing studies are often centred around a single map or a small set of maps, while this study uses many different maps, from a large range of time. This can give a more general idea of how well the models perform with map data, without focusing on the exact specifics of individual maps. It can also show whether the models are able to generalise the different representations of map features. Finally, the field of AI can benefit from a more thorough comparison of CNNs and RNNs with regards to image data. Especially the processing of image data from different sources with RNNs has not been tested often. This thesis hopes to shed more light on the strengths and weaknesses of these networks with regards to image segmentation and change detection, and how the combination of these tasks influences the performance of the networks.

3 Methods

3.1 Data

The data consist of topographic maps of the Netherlands. The maps have been gathered by the Dutch Topographic Service since 1815, and have been managed by the Kadaster since the Topographic Service merged with it in 2004. The maps are available online on the website Topotijdreis.nl [32]. The maps that are used are dated from 1929 to 2019. This time range was chosen because older maps often have more scale inconsistencies, which hampers the comparison between maps of the same area. It is also the timeframe that is most useful for soil advisors with regards to detecting possible contamination. They are all georeferenced in the EPSG 28992 (Amersfoort / RD New) datum. The georeferencing of the older maps has been done by hand, while the newer maps are automatically georeferenced. Unfortunately the Kadaster has no data in its possession concerning the estimated accuracy of the maps, because a large part of the work was done years ago. The inaccuracy of the maps is therefore not taken into account when evaluating the performance of the models. Not every year has its own map; the maps are only created periodically, and these time intervals are not constant over the country and over the years. If all years are used by default, this will result in large amounts of data where no changes in the landscape can be seen at all. The maps are therefore sampled at a 10 year interval. Because some maps could not be loaded into the software properly, the sampled maps are not exactly 10 years apart; the maps are from 1929, 1940, 1950, 1962, 1970, 1980, 1990 and 2019. Note that these are the maps that were the most recent at that time, not the year that the map was created. Maps of multiple scales are available, but only the largest scale is used, which is roughly 1:6000. This is done because the features that are sought are not very large, and can be missed on smaller-scale maps. Because there is some variability in the landscape between the provinces, 10 different locations across the country are chosen for sampling. The exact sampling locations are chosen by hand, rather than sampling randomly; this is done to ensure that the frequency of the segmentation classes is high enough. From each location an area of 1 to 3 square kilometres is selected and annotated, after which the samples are taken from that area in a grid pattern. A total of 256 tiles is sampled, which results in 2.560 sample images in total; one for each year.

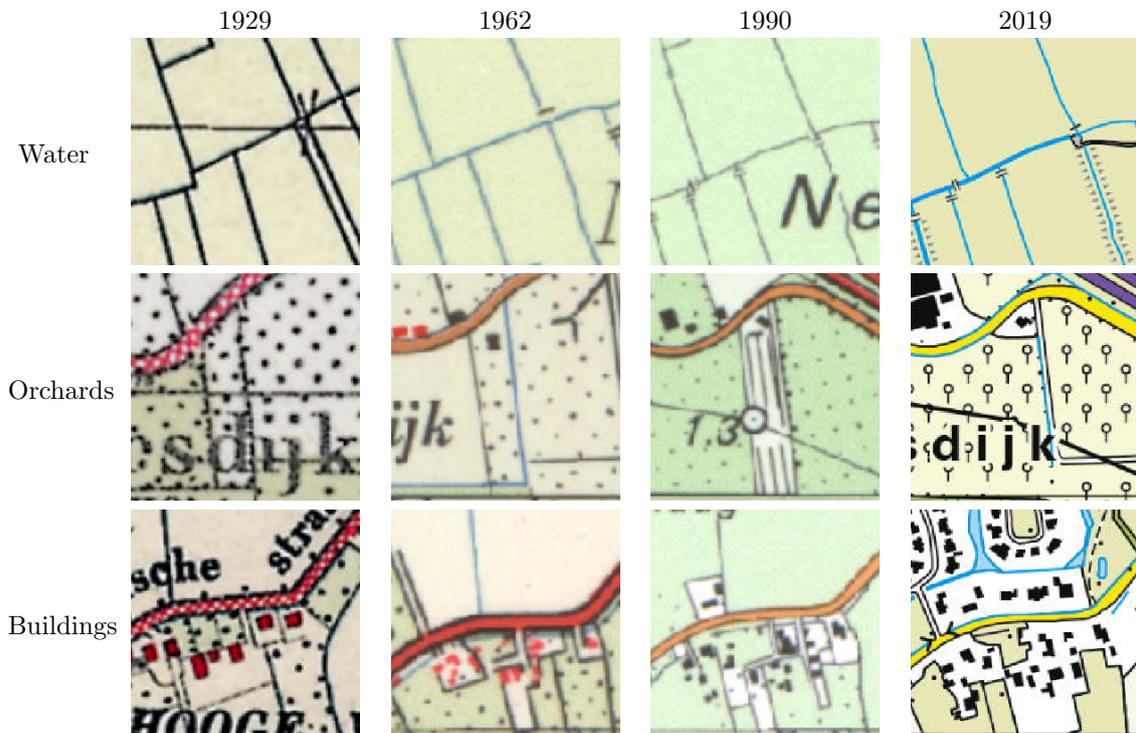


Figure 8: Examples of the segmentation classes over time.

The features that have to be detected are waterways, orchards, and buildings. Examples of these features can be seen in Figure 8. Any changes in the landscape have to be reported as well.

The changes are given as a difference between the more recent map and the current map, meaning that the sequence goes from 2020 to 1930. The marking of these features on the map is done manually by the author using the open-source software QGIS. Because this is done manually, the annotations are flawed; the segmentation could be off by some pixels, and in the case of change detection it is very possible that changes have been missed.

To annotate the maps, some criteria were set to ensure that they were annotated consistently. Regarding the buildings, every building is annotated, even if it is only a shed rather than a full-sized building. Related features such as greenhouses or covered storage areas are annotated differently on the maps and are not included. Regarding the waterways, any water that is in contact with the ground is included. This means that a water purification plant does not count, as it is separate from the groundwater. Of special interest are the ditches. It can be unclear on maps whether a line is a ditch or simply a partition between two plots when no colouring is used. In such cases, it is marked as a ditch if there is a bridge over it, or if the maps of both the past and the future of the area have coloured it as a ditch. Regarding the orchards, the whole plot is marked when an orchard pattern is found on it. Houses on the same plot are only given a buffer around them when this is also apparent in the orchard pattern; otherwise, only the house is spared out. Finally, the changes in the landscape have a set of rules. Something is denoted as a change when the land use type has changed. Examples of land use are croplands, grasslands, orchards, forests, roads and rivers. When it appears that a building has been extended, the extension is only marked if it is a large expansion. Concretely, this means that residential houses generally do not have marked expansions, but factories and other large buildings do. If it appears that a building has been replaced, both the old and the new building are marked. Small changes such as the disappearance of a bridge over a ditch is not marked, as these are not always drawn even if they are there. When the features of the two maps are not correctly aligned because of differences in the georeferencing, the presumed location on the older map is used. If this location cannot be determined because there is no common reference point close to the feature, the change is drawn on the location of the feature on the newer map.

The distribution of the classes on the tiles can be seen in Tables 1 and 2. As can be seen, the distributions are not equal across the years. Because each dataset has the same number of tiles for each year, this is no cause of concern for the train-test split. However, it does mean that years which have a significantly higher percentage of a certain class will have more weight in the total assessment. As for the general distributions, it can be seen that water is present in the largest percentage of tiles, while orchards are more rare. However, when counting the number of pixels, it can be seen that the orchards make up a relatively large number of pixels compared to the other two segmentation classes. This is because each orchard takes up much more pixels than individual buildings. The changes are both present in many tiles, and in many pixels. The distribution across time shows a visible dip in 1940; this is because the maps used in 1940 and 1950 are often the same ones. A probable reason for this is the Second World War. 1950 has many changes compared to the other years, with the same probable cause. Finally, it should be noted that the distribution across the individual tiles is not equal; some locations have more examples of a certain class than others. Because this imbalance is not taken into account when making the train and test splits, it is possible that the distributions between the sets differ as well.

Class	1929	1940	1950	1962	1970	1980	1990	2000	2010	2019	Mean
Building	51.95	55.47	58.98	62.11	63.28	64.06	64.45	65.23	65.23	67.97	61.88
Water	67.97	67.58	65.23	62.89	60.55	69.53	72.27	83.20	85.16	87.11	72.15
Orchard	27.11	43.26	48.44	29.61	51.95	46.48	41.41	39.84	39.06	37.50	43.48
Change	62.11	25.00	93.36	62.11	71.88	87.11	92.97	91.41	82.81	–	74.31

Table 1: Percentages of tiles that contain at least 1 pixel of the classes.

Class	1929	1940	1950	1962	1970	1980	1990	2000	2010	2019	Mean
Building	1.96	2.65	3.38	4.13	4.36	4.36	4.83	5.01	4.86	4.99	4.05
Water	4.52	5.05	4.48	4.38	4.46	7.41	7.92	8.32	8.60	8.95	6.41
Orchard	6.23	10.23	11.79	15.19	15.23	9.74	7.93	8.15	7.93	8.56	10.10
Change	10.15	4.16	15.96	8.30	16.72	20.58	17.67	11.75	11.32	–	12.96

Table 2: Percentages of pixels that contain the classes.

From the larger marked area, equal-sized images are extracted to serve as input for the models. These images are 256×256 pixels, with each pixel representing approximately one square metre. The images are sampled in a grid with a stride of 249 meters. This means that there is an overlap of 7 meters between adjacent tiles. The overlap is present because the sampling areas were previously fitted to tiles of 200×200 pixels, but were later changed to the current image size to conform to standard neural network inputs sizes. In order to make the model more robust to small changes in the images and to create more training data, augmented versions are added to the dataset as well. This includes adding Gaussian noise with a standard deviation of 0.01, blurring it with a mean filter with kernel size 5, flipping it vertically or horizontally, and rotating it [67]. The rotation is either 90, 180 or 270 degrees. The augmented images are not saved on the disk, but are generated during training in order to save memory. Each augmentation has a set chance of being activated after an image has been chosen for training, which means that multiple augmentations can be applied at the same time. The dataset is repeatedly split into training, validation and test sets for the experiments. A tile is only present in one of those sets, in order to prevent similar images from occurring in both the training and the test set. This also means that all years are perfectly balanced between the sets. The proportions of features that are shown are not explicitly balanced. Only the training data are augmented; the validation and test data are not. The test data therefore consists of tiles with the same dimensions of the training data. Although classifying a larger area is also interesting, a downside to such an approach is that there is less variation in the landscape that is classified, as there is only a limited amount of data available. This possibility is therefore not part of the experiments.

3.2 Models

This Section details the architectures of the models that are used to segment and classify the maps. In order to put the performance of the main models into perspective, two baseline models have been constructed which are detailed in Section 3.2.1. The neural network models are detailed in Section 3.2.2. A short summary of these models can be found in Table 3, along with the shorthand names that are used to refer to them.

Model	Input Format	Output format	No. Input Maps	Combine method
Forest _{block}	7×7 pixel block	single pixel	1 (segment) / 2 (cd)	division
Forest _{object}	features object	all pixels object	1 (segment) / 2 (cd)	division
NN ₁	full map	all pixels map	1	none
NN ₂	full map	all pixels map	2	concatenation
RNN ₂	full map	all pixels map	2	CGRU
RNN _{all}	full map	all pixels map	all newer maps	CGRU

Table 3: The models that are tested. CD stands for change detection.

3.2.1 Baseline models

Two baseline models are constructed to put the performance of the neural networks into context. They do not represent the state-of-the-art with respect to image segmentation; rather, they are used to give an idea of how the neural networks compare to non-neural network models. The two models differ in the way in which they segment the image before processing the segments, and represent two ways of doing so that are popular with remote sensing data: pixel-based and object-based classification [29][43]. Predicting the whole image at once is not feasible for conventional classification algorithms. The first model is pixel-based: it takes a block of 7×7 pixels, and uses this to predict the class of the middle pixel. The values that are used here are the colour values of all the pixels. The second model is object-based: it first segments the image into objects, and then classifies the whole object based on its characteristics. The segmentation algorithm that is used is Quickshift [83]. Felzenszwalb [20] and a variation of Watershed [55] were also tried, but those resulted in segmentations that did not capture the objects in the image well. Characteristics that are used are mean colour (three values, RGB), standard deviations of the colour, area in number of pixels, perimeter and hu moments (7 values), resulting in 15 values for each object. The separate colours of the pixels in the object cannot be used, as this would result in different numbers of values between objects. There is no information about the neighbouring objects for the same reason. The

values of both methods are processed by a random forest classifier, which outputs a single class for either the middle pixel or the object. Because pixel-wise methods are known to produce a ‘salt-and-pepper’ pattern, its result is smoothed with a 3×3 averaging operation to filter out small islands of classes. The output of the object-based model is taken as is. These models are used both for semantic segmentation and for change detection. In all cases, the input map is smoothed with a 5×5 averaging operation to remove some noise from the images. For change detection, a change mask is created by dividing the older map by divided the values of the newer map. Subtraction was also tried for the change mask, but this gave worse results in test runs. This change mask is then given to the models as input; the object-based methods finds objects in the change mask itself, not in the individual maps.

3.2.2 Neural Networks

The neural networks are based on the general architecture of U-Net [65], with an encoding branch that downsamples the images with an increasing number of channels, and a decoding branch that upsamples the images again. Skip connections are used between parts of the encoding branch and the decoding branch that have the same dimensions. The general architecture can be seen in Figure 9. The number of downsampling steps is reduced to 3 compared to the original U-Net to keep the network small and to prevent overfitting. To compensate for the resulting smaller receptive field, each depth has one convolutional layer with a dilation rate of 2. The result of this layer is concatenated with the result of the previous layer without dilation and passed to the next depth together. This structure is inspired by the work of Wang et al. [85]. The downsampling is performed with a strided convolution instead of with max-pooling, as this resulted in a higher output resolution. This strided convolution is done with a kernel size of 2×2 instead of the regular 3×3 to reduce checkerboard patterns caused by overlapping pixels. The exact layout of the layers, which is repeated at each depth, can be seen in Figure 10.

The difference between the models lies in how many timeframes they use as input, and how they combine inputs from multiple timeframes. The first model NN_1 is the simple semantic segmentation model, which does not receive input from other timeframes. It therefore passes the feature maps from its own encoding path directly to its decoding path. The second model NN_2 combines the inputs of the two timeframes by concatenating the two before passing it the the decoding branch. This structure is inspired by the work of Caye Daudt et al. [9], who use a similar set-up. Subtracting and dividing the two maps was also tried in test runs, but this resulted in erratic training curves and lower performances. The third and fourth models, RNN_2 and RNN_{all} , combine the maps by using a convolutional GRU; a GRU with convolutional layers. This GRU has the same number of feature maps as output as the individual input maps. The difference between the third and fourth model lies in how many timeframes they get as input; the third model gets two, while the fourth gets all maps that are more recent than the current one.

All models use the same loss function, namely the pixel-wise cross-entropy loss. The outputs of the model are put through a sigmoid function to create outputs between 0 and 1. A sigmoid is used rather than softmax because the probabilities of the different classes are not related to each other; a pixel can be both changed and a building. For evaluation, the threshold for classifying the pixel as the class is set to 0.5. The standard cross-entropy function can be seen in equation 5, where \hat{y} denotes the predicted class and y the true class. This is computed separately for each class. The loss is therefore not a single value, but separate values for each pixel and each output class.

$$L_i = -(y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})) \quad (5)$$

The optimisation function that is used is Nesterov Adam. This is a variant of the commonly used Adam optimiser that uses Nesterov momentum, and has been shown to perform better than Adam in some cases, for example for change detection [52]. Adam in general is an adaptive optimiser, which means that the learning rate is not equal for each parameter; rather, each parameter has its own learning rate that is changed over time. The learning rate that is given to the algorithm is therefore a maximum learning rate, not an absolute one. Still, it can sometimes be useful to clip the learning rate manually to prevent it from getting too high in the latest stages of learning. Both clipping the learning rate manually with a scheduler or by clipping it based on the validation loss was tried in test runs, but it did not increase the performance or training time. The maximum

learning rate is therefore used during the whole training session. A summary of optimisation techniques can be found in [66].

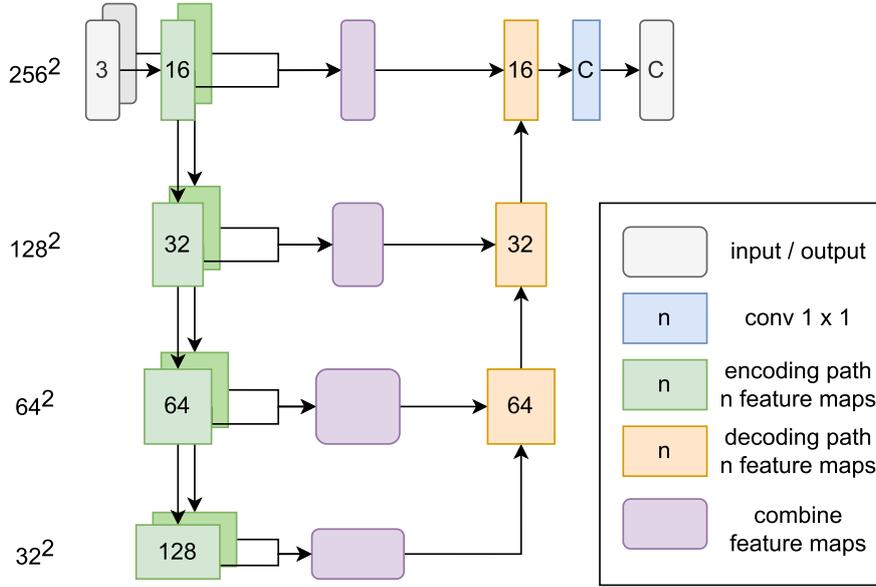


Figure 9: The general model structure. The numbers on the left are the sizes of the feature maps, the numbers in the blocks the number of feature maps (the channels). C stands for the number of classes that are predicted. The exact layers in the encoding and decoding blocks can be found in Figure 10.

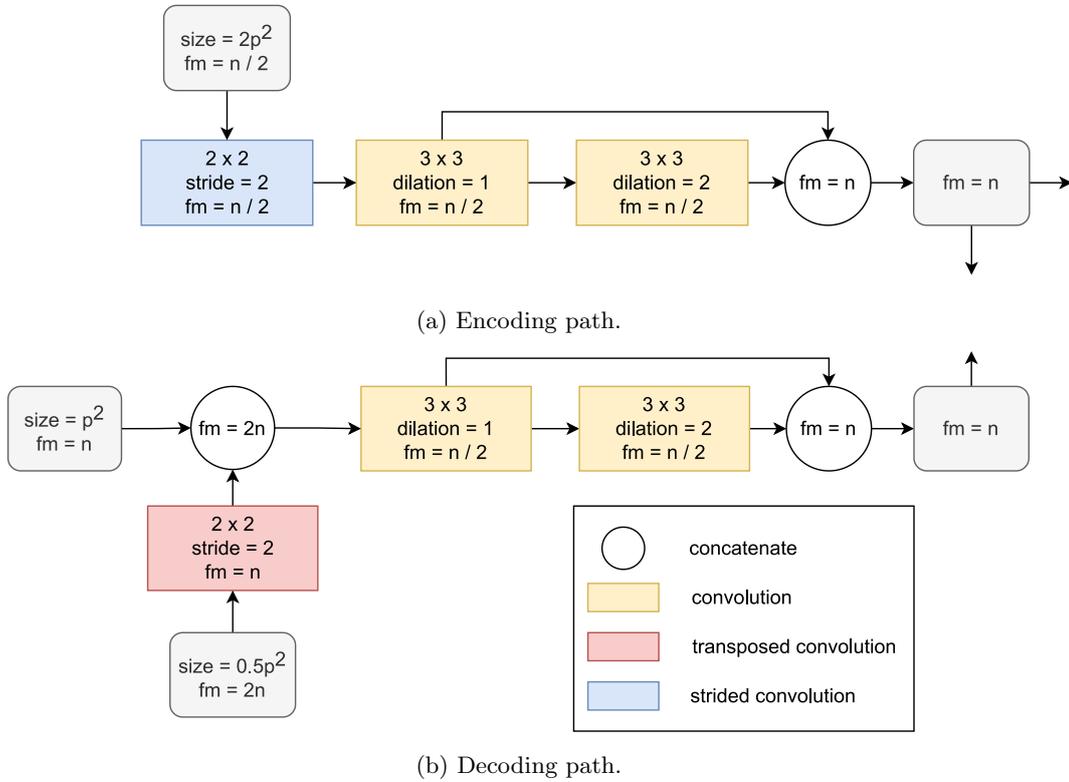


Figure 10: Exact layers for the encoding and decoding paths. These are used at each depth in Figure 9. FM stands for the number of feature maps. The size is the height and width of the image; p stands for the size that the output of the block has, which can be seen on the left side in Figure 9.

4 Experimental Setup

4.1 Parameters

This section lists the exact parameters that are used during the experiments. The experiments are run on an external computer through a virtual machine. The computer has an Intel Xeon E5-2690 v4 processor running at 2.60GHz with 6 CPU cores, 112 GiB of RAM, a NVIDIA Tesla 1X P100 GPU, and it runs Windows 10. It should be noted that the baseline models run on the CPU, while the neural networks are trained on the GPU. This is done because there is no Random Forest implementation available that supports GPU computation. The parameters for the baseline models are the following: the maximum depth of the decision trees is 40, the minimal number of samples for a split is 30, and the minimal number of samples for a leaf node is 5. Six decision trees are produced for each model. These trees are built in parallel using the six CPU cores. Each decision tree uses one third of the training data. The Random Forest implementation of the Python package Scikit-learn [61]. The Quickshift segmentation algorithm of model Forest_{object} converts the colours to Lab colour-space first, and uses a sigma of 0.3 for additional smoothing. The implementation is that of the Python package Scikit-image [84]. As for the neural networks, the Nadam optimiser uses a maximum learning rate of 0.001 with a clip normalisation of 0.001. The batch size is 16 for all models except for the RNN_{all} model, which uses a batch size of 8. These batch sizes were chosen because of memory limitations. Additionally, the RNN_{all} model is limited to batches which contain the same year, as opposed to the other models where the batches are mixed. This is done because the software cannot handle batches that contain data of different sizes. While the learning rate is relatively high for the used batch size, it gave better results in test runs compared to lower values. The number of training epochs depends on the performance of the models on the validation set; if this performance does not increase for 20 epochs, the training session is stopped. An upper bound of 250 training epochs is set in case this does not happen. The implementation of the neural networks is done with the Python package Tensorflow [49]. The data is also preprocessed for the neural networks. Each augmentation of the training data a 30% chance to be activated, meaning that the chance that the original image is shown is 24%. Multiple augmentations can be activated at the same time.

4.2 Evaluation

The models are compared with one another in different aspects. The training time, number of parameters and loss over time are shown as a measure of efficiency. The performance of the models is shown separately for the necessary tasks. First of all, the performance of the models when only segmenting the images without change detection is tested. Secondly, the performance of the models when only detecting changes in the landscape is tested. Finally, the performance of the models when performing both the segmentation and the change detection task is tested. This is measured both separately for the two tasks, and as a combined score. The segmentation results are presented separately as a mean of the performance of all classes. Which model configurations perform which task can be seen in Table 4.

Model	Segmentation	Change detection	Both
Forest _{block}	yes	yes	no
Forest _{object}	yes	yes	no
NN ₁	yes	no	no
NN ₂	yes	yes	yes
RNN ₂	yes	yes	yes
RNN _{all}	yes	yes	yes

Table 4: Overview of which models participate in which experiments.

The performance is measured in several ways based on the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). A positive means that the model predicts that the class is there, a negative that it is not; it is true if the ground truth agrees with the classification, otherwise it is false. The measures are precision, recall, F1 score, and intersection over union (IoU, also called Jaccard similarity). The calculations can be seen in equations 6 to 10. The precision shows how many of the classified pixels were actually of that

class; a low score means that that class is assigned too liberally. The recall shows how many of the pixels of the class were successfully detected. The F1 score provides a balanced combination of precision and recall. IoU focuses on the overlap between the classification and the ground truth; the true negatives are not important as those are ignored by both. While the F1 score and the IoU are similar in their calculations, the interpretation of their results is slightly different; the IoU punishes misclassifications more harshly, which results in a ‘worst-case’ performance measure. The F1 score, on the other hand, averages the scores more evenly. The accuracy denotes how many of the classifications were correct, taking into account both the positive and the negative cases. These measurements are commonly used in image segmentation, and provide a balanced estimation of the performances of the models [79][44][91].

The first round of tests does not distinguish between mistakes that were made because the model mistook a feature for another, and those that were made because of imprecision. This distinction is important, as imprecision is less problematic because an estimate of the location of a feature can still be useful. This imprecision can be caused by different factors. Firstly, the ground truth may not be accurate, as it is hand-drawn by the author. The error margin here is assumed to be no more than two pixels for semantic segmentation. Another cause of imprecision is the shift of features between maps of different time periods due to imprecise georeferencing. Because this displacement can differ significantly even on a single map, no definite buffer size can be given for it. This influences not only the performance of the models, but also the quality of the ground truth. Finally, imprecision can be caused by the model itself. To get an estimate of how many of the mistakes were caused by imprecision, the scoring of the classifications is also tested with buffers. This means that a pixel that is misclassified is still marked as correct if the model did classify a nearby pixel as the true class of the pixel. As it is difficult to choose a buffer size based on the displacements of the maps, the acceptable level of imprecision for the models is estimated to be equal to that of the author, which means that a buffer of two pixels is used. These buffered results are then compared with one another in the same way as the non-buffered experiments.

To see whether any found changes in performance are significant, 5-fold cross-validation is performed on all models, with each model using the same folds. This number of folds is used because running the model more times than that would be too expensive and time-consuming. They are then compared with one another using paired ANOVA analyses, to see whether there is any difference between the models. To find out which models differ, paired t-tests with Benjamin-Hochberg False Discovery Rate correction (FDR-BH) are performed. This is only done with the macro F1, because performing statistical tests on all the metrics would result in too many tests, which in turn increases the chance of a type 1 error. The F1 scores provide a balanced score of the performances of the models for comparison. The other metrics are used for analysing how the models perform. It is known that the samples used in the tests are not independent because the training samples are used in multiple folds. This violation of the independency assumption can lead to a higher probability of a type 1 error [18]. Although a 5×2 cross-validation is sometimes recommended in such cases, this is not feasible for this study as it would make the training set too small. The probability of a type 1 error is not excessively high, and some are even of the opinion that it is negligible [88]. Still, it is kept in mind during the data analysis. In addition to these risks, a sample size of 5 is relatively small for an ANOVA test. There is therefore a good chance that the statistical power of the outcomes is relatively low, which in turn increases the chances of a type 2 error. This, too, must be kept in mind. As with any statistical analysis, the significance should not be accepted blindly, but combined with careful examination of the data. Finally, the difference in performance is weighted with the difference in computational efficiency to make an estimation of which method is preferred for which task.

$$\text{Precision (P)} = \frac{TP}{TP + FP} \quad (6)$$

$$\text{Recall (R)} = \frac{TP}{TP + FN} \quad (7)$$

$$\text{F1} = \frac{2 \cdot P \cdot R}{P + R} \quad (8)$$

$$\text{Intersection over Union (IoU)} = \frac{TP}{TP + FP + FN} \quad (9)$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (10)$$

5 Results and Evaluation

For each of the experimental conditions the models were run 5 times, each time with a different subset of the training data. The results that are shown here are averages over those 5 runs. All data on which statistical tests are performed are tested on sphericity and normality, to make sure that they adhere to the assumptions of the tests. The sphericity is checked with Mauchly’s test of sphericity. This checks whether the variances of the differences between the independent variables are equal. The normality of the data is tested with a Shapiro-Wilk test for each model. These assumptions hold for all data that are used. The effect size that is reported in all comparisons is Hedges’ g , rather than the commonly used Cohen’s d . This is done because Hedges’ g gives a more reliable result with small sample sizes.

After the models have been compared on their general performance, their behaviour is examined in more detail. This is done by showing the performances separately for each decade and class, and by showing the change in performance if the evaluation is slightly more lenient. This is done for both the segmentation task and the change detection task. Where possible, the results are related back to the data to explain the patterns in the models’ behaviour.

5.1 Speed and Memory Consumption

The training time and the number of trainable parameters give an indication as to how efficient a model is. These numbers can be found in Table 5. It should be noted that the random forest models are trained on the CPU, while the other models are trained on the GPU. If the neural networks were trained on the CPU as well, they would be much slower than the random forest models. The prediction times are timed with all models running on the CPU to show how fast they would be when used in practical applications. The times of predicting one image are averages over 100 images. The time it takes to give predictions for 100 images is less than 100 times the time of predicting one image for all models; both random forests and neural networks have efficient implementations for predicting many samples at once. The number of parameters refers for the random forest models to the number of nodes in their decision trees, and for the neural networks to the number of trainable weights. These are roughly the same for the neural networks across the different tasks, as the only difference between them is the number of weights in the final layer. The RNN models have a larger number of weights because the GRU needs several weight kernels for each unit. The number of nodes in the $\text{Forest}_{\text{block}}$ model is much larger than that of the $\text{Forest}_{\text{object}}$ model. This is likely because it has more input variables.

Task	Metric	$\text{Forest}_{\text{block}}$	$\text{Forest}_{\text{object}}$	NN_1	NN_2	RNN_2	RNN_{all}
Segment	train time	1:49	0:20	0:15	0:21	0:54	2:31
	predict 1	0:21	1:03	0:09	0:11	0:20	0:49
	predict 100	19:28	34:88	6:15	8:62	19:54	43:64
	no. param	10.135.412	71.463	225.963	280.603	682.315	682.315
CD	train time	0:51	1:45	-	0:31	0:53	2:34
	predict 1	0:18	1:76	-	0:11	0:21	0:51
	predict 100	17:76	135:83	-	8:76	19:26	47:02
	no. param	9.661.337	505.115	-	280.569	682.281	682.281
Both	train time	-	-	-	0:40	0:56	3:25
	predict 1	-	-	-	0:11	0:20	0:50
	predict 100	-	-	-	7:87	17:66	43:05
	no. param	-	-	-	280.620	682.332	682.332

Table 5: Training times in h:m format, prediction times in s:ms format, and the number of parameters in the models.

Interestingly, the $\text{Forest}_{\text{block}}$ model is much slower to train in the segmentation task than the $\text{Forest}_{\text{object}}$ model, while these roles are reversed for the change detection task. A probable cause is that the objects that are found for the $\text{Forest}_{\text{object}}$ model are much smaller and less informative than those found during segmentation, thereby requiring more splits in the decision tree. During prediction the $\text{Forest}_{\text{block}}$ model is faster than the $\text{Forest}_{\text{object}}$ model in both conditions, likely because the preprocessing of the images takes up a larger percentage of the time during prediction than during training. As for the neural networks, they get progressively slower the more complex

the model is. The simplest model NN_1 takes 15 minutes to train, while the RNN_{all} model which processes all previous years is the slowest at two and a half hours of training for the individual tasks. This trend can be seen in the prediction times as well. Of note is that the NN_2 model takes more time to learn the change detection task than the segmentation task, while there is no noteworthy difference for the RNN models between the tasks. Still, both RNN models remain slower than the NN_2 model in all conditions.

5.2 Segmentation

The scores for the segmentation task can be seen in Table 6. The macro scores reflect the average of the scores of the individual segmentation classes, while the micro score takes class imbalance into account. This results in more weight being given to the orchard class, which has as many pixels as the other two classes combined. The results are the averages over all training data, with no distinction between the different years. A repeated measures ANOVA test is performed on the F1 macro scores to see whether the performances of the models differ. It shows that the models perform significantly different from one another [$F(8, 32)=544.25, p<0.001, \eta^2=0.99$]. In order to find out which models differ from which, several pairwise T-tests with FDR-BH correction are performed.

First of all, let us look at the baseline models. These perform significantly worse than the neural network models in each pairwise one-tailed T-test, with all corrected p-values below 0.001. The effect size of each comparison was also quite large, ranging from -6.5 between $Forest_{block}$ and RNN_{all} (both), to -21.2 between $Forest_{object}$ and NN_1 . All comparisons can be found in Appendix A in Table 11. Looking at the recall and the precision of the models, it can be seen that the difference in precision between the baselines and the neural networks is larger than that of the recall. This is especially the case for the $Forest_{object}$ model, with a macro recall of 75% and a macro precision of 54%. Looking at the individual classes, it can be seen that both models perform poorly on the orchard class in particular, with F1 scores of 51% and 58% compared to 87% to 91% for the other models. As the orchard class is the one with the largest number of pixels, this makes the difference between the baselines and the neural networks larger in the micro average scores than in the macro average scores. The two baselines also perform significantly different from one another in a two-tailed T-test [$t(4)=50.3, p\text{-corr}<0.001, g=7.8$]. From Table 6 it can be seen that the $Forest_{block}$ model tends to perform better than the $Forest_{object}$ model. The only metric where the $Forest_{object}$ model performs better is the recall for orchards, at 70% against 67%. Because its precision is relatively low (41%) for that class, however, it does not lead to a better F1 score.

Secondly, the NN_1 model is tested against the other neural networks with the assumption that the others would do better, as they have access to additional information. This assumption does not hold in any of the T-tests, meaning that the other models do not perform significantly better than the NN_1 model. In fact, the NN_1 model performs better than the models that are trained on both tasks. The comparisons between the segmentation-only model are: NN_2 [$t(4)=-0.95, p\text{-corr}=0.34, g=-0.40$], RNN_2 [$t(4)=-0.75, p\text{-corr}=0.39, g=-0.29$], and RNN_{all} [$t(4)=-0.87, p\text{-corr}=0.35, g=-0.43$]. When looking at the performances of the individual classes, it can be seen that the NN_1 model performs similarly to the other segmentation-only models when classifying buildings and orchards. For the water class it has a lower recall (82% against 85% to 86%), while having the highest precision (91% against 89% to 90%).

Thirdly, we test whether the RNN_2 model performs better than the NN_2 model, as it has a more controlled way of integrating the two maps. This is not the case, neither for the models trained only on the segmentation task [$t(4)=0.23, p\text{-corr}=0.58, g=0.10$], nor for those trained on both tasks [$t(4)=-0.5, p\text{-corr}=0.48, g=0.18$]. The NN_2 model consistently performs slightly better overall when performing only the segmentation task. It also has a higher recall for all classes, while the RNN_2 model has a higher precision for all classes. As for the models trained on both tasks, the RNN_2 model performs slightly better than the NN_2 model. This is the case for both the water and the orchard class, while the two have the same score for the buildings class. No particular pattern between the recall and the precision scores is seen in this case.

Fourthly, the RNN_{all} model is tested against the RNN_2 model, to see whether having input of more than two maps is helpful. This is not the case, neither for the models trained on the segmentation task [$t(4)=0.22, p\text{-corr}=0.42, g=-0.1$], nor for those trained on both tasks [$t(4)=1.79, p\text{-corr}=1, g=1.0$]. In fact, when trained on both tasks the RNN_{all} model tends to perform worse than the RNN_2 model (85.2% against 86.8%).

Finally, it is of interest to see whether the models perform differently when they only perform the segmentation task as opposed to when they perform both tasks. As there was no clear prior for this difference, this is tested with a two-tailed test. The NN₂ model performed significantly different in the two conditions [$t(4)=-5.1$, $p\text{-corr}=0.008$, $g=-1.75$]. The RNN₂ model did not [$t(4)=-3.8$, $p\text{-corr}=0.03$, $g=1.16$], nor did the RNN_{all} model [$t(4)=-3.4$, $p\text{-corr}=0.04$, $g=-2.1$]. From Table 6 it can be seen that the models that only performed the segmentation task tend to perform better than those that perform both tasks.

The best performing models on this task are the NN₂ model and the RNN_{all} model trained on the segmentation task, with a shared macro F1 score of 88.3%.

Class	Task	Model	Accuracy	Recall	Precision	IoU	F1
Building	Seg	Forest _{block}	98.3 (0.3)	84.7 (2.1)	75.0 (4.6)	66.1 (4.5)	79.5 (3.3)
		Forest _{object}	96.0 (0.5)	79.6 (1.8)	49.4 (7.6)	43.8 (6.2)	60.7 (6.0)
		NN ₁	98.9 (0.2)	87.0 (3.7)	85.7 (2.1)	76.0 (4.2)	86.3 (2.7)
		NN ₂	98.9 (0.3)	87.3 (1.5)	86.2 (2.4)	76.6 (2.8)	86.7 (1.8)
		RNN ₂	99.0 (0.1)	84.6 (5.7)	87.9 (1.9)	75.8 (5.4)	86.1 (3.6)
		RNN _{all}	99.0 (0.3)	86.9 (3.4)	87.8 (2.1)	77.4 (2.2)	87.3 (1.4)
	Both	NN ₂	98.9 (0.2)	84.8 (3.0)	85.6 (3.1)	74.2 (4.1)	85.2 (2.7)
		RNN ₂	98.9 (0.2)	85.3 (4.2)	85.3 (3.1)	74.3 (3.9)	85.2 (2.6)
		RNN _{all}	98.9 (0.2)	83.8 (2.8)	86.0 (3.3)	73.7 (3.4)	84.8 (2.3)
Water	Seg	Forest _{block}	98.1 (0.2)	85.3 (3.9)	83.4 (4.3)	73.0 (5.9)	84.3 (4.0)
		Forest _{object}	96.7 (0.3)	75.4 (4.1)	71.2 (7.5)	58.0 (7.2)	73.2 (5.9)
		NN ₁	98.4 (0.2)	82.3 (5.2)	91.0 (2.9)	76.2 (5.7)	86.4 (3.7)
		NN ₂	98.5 (0.2)	85.9 (3.6)	88.6 (3.2)	77.4 (5.2)	87.2 (3.3)
		RNN ₂	98.5 (0.1)	84.6 (4.2)	90.0 (2.8)	77.4 (5.4)	87.2 (3.5)
		RNN _{all}	98.4 (0.2)	84.9 (4.4)	88.5 (2.9)	76.5 (4.8)	86.6 (3.1)
	Both	NN ₂	98.3 (0.2)	82.2 (4.7)	87.9 (3.1)	73.9 (5.5)	84.9 (3.7)
		RNN ₂	98.3 (0.2)	82.5 (7.0)	87.9 (3.1)	74.1 (6.4)	85.0 (4.3)
		RNN _{all}	98.2 (0.3)	79.3 (6.1)	88.0 (3.5)	71.7 (6.5)	83.4 (4.5)
Orchard	Seg	Forest _{block}	90.5 (0.7)	66.6 (1.7)	52.2 (6.3)	41.2 (3.8)	58.3 (3.8)
		Forest _{object}	86.7 (0.3)	70.2 (2.0)	40.6 (6.5)	34.5 (4.5)	51.2 (5.0)
		NN ₁	98.2 (0.3)	90.1 (0.3)	92.0 (1.7)	83.5 (1.3)	91.0 (0.8)
		NN ₂	98.2 (0.2)	90.8 (0.4)	91.4 (2.1)	83.7 (1.7)	91.1 (1.0)
		RNN ₂	98.3 (0.3)	90.8 (1.9)	91.9 (1.9)	84.0 (1.4)	91.3 (0.8)
		RNN _{all}	98.3 (0.3)	89.4 (1.9)	93.0 (1.5)	83.8 (1.3)	91.2 (0.8)
	Both	NN ₂	97.9 (0.2)	90.0 (1.1)	89.5 (2.5)	81.4 (2.0)	89.7 (1.2)
		RNN ₂	98.0 (0.3)	90.4 (1.1)	90.1 (2.0)	82.2 (1.8)	90.2 (1.1)
		RNN _{all}	97.5 (1.1)	85.3 (7.9)	89.8 (2.2)	77.8 (7.2)	87.4 (4.8)
Macro	Seg	Forest _{block}	95.6 (0.1)	78.8 (1.8)	70.2 (1.7)	60.1 (1.8)	74.0 (1.4)
		Forest _{object}	93.1 (0.2)	75.1 (1.4)	53.7 (1.9)	45.4 (1.8)	61.7 (1.4)
		NN ₁	98.5 (0.1)	86.5 (1.6)	89.6 (0.6)	78.6 (1.1)	87.9 (0.7)
		NN ₂	98.6 (0.1)	88.0 (1.3)	88.7 (1.0)	79.2 (1.7)	88.3 (1.1)
		RNN ₂	98.6 (0.1)	86.7 (2.3)	89.9 (0.7)	79.1 (1.7)	88.2 (1.1)
		RNN _{all}	98.6 (0.1)	87.1 (1.8)	89.8 (1.3)	79.2 (1.6)	88.3 (1.0)
	Both	NN ₂	98.4 (0.1)	85.7 (1.2)	87.7 (0.8)	76.5 (0.8)	86.6 (0.6)
		RNN ₂	98.4 (0.1)	86.1 (3.0)	87.8 (1.5)	76.9 (1.8)	86.8 (1.3)
		RNN _{all}	98.2 (0.4)	82.8 (2.4)	87.9 (0.8)	74.4 (2.4)	85.2 (1.6)
Micro	Seg	Forest _{block}	95.6 (0.1)	76.2 (2.7)	65.5 (4.0)	54.4 (3.4)	70.4 (2.9)
		Forest _{object}	93.1 (0.2)	73.9 (1.6)	49.7 (3.9)	42.3 (2.8)	59.4 (2.8)
		NN ₁	98.5 (0.1)	87.5 (1.3)	90.7 (0.9)	80.3 (0.9)	89.1 (0.6)
		NN ₂	98.6 (0.1)	88.7 (1.1)	89.9 (1.1)	80.7 (1.7)	89.3 (1.0)
		RNN ₂	98.6 (0.1)	88.1 (1.2)	90.9 (1.2)	80.9 (0.4)	89.4 (0.3)
		RNN _{all}	98.6 (0.1)	87.8 (0.7)	90.7 (1.8)	80.5 (1.4)	89.2 (0.9)
	Both	NN ₂	98.4 (0.1)	87.0 (1.0)	88.7 (1.3)	78.3 (1.2)	87.8 (0.8)
		RNN ₂	98.4 (0.1)	87.6 (1.6)	88.7 (2.1)	78.8 (1.3)	88.1 (0.8)
		RNN _{all}	98.2 (0.4)	83.6 (3.8)	88.9 (0.8)	75.7 (3.3)	86.1 (2.2)

Table 6: Mean segmentation scores as percentages, with the standard deviation in brackets.

5.2.1 Yearly Comparisons

The models are all trained on maps from different years. While the average score is useful for determining which model performs best, the performances over the years can provide additional insights. The macro F1 scores over time are shown in Figure 11. Please note that this and subsequent figures start their y-axis at a score of 0.3, as the scores do not get lower than that. The models that are trained on both tasks are not shown in these graphs, as the trend of their behaviour is similar to that of their counterparts which were only trained on the segmentation task. It can be seen that all models follow a similar trend: they score the worst on the 1930 maps, and gradually get better until 2000, where the scores stabilise. At 1990 there is a dip in performance for all models. As for the baselines, it can be seen that the difference between the `Forestobject` model and that of the neural networks gets smaller as the maps get more recent. In 1990 the dip of the `Forestobject` model is more pronounced than that of the other models as well. This suggests that the change in difficulty between the years has a stronger impact on this model than on the others. The `Forestblock` model has a high jump in performance in 2000, after which its score is quite close to that of the neural networks. Before that, the changes in performances over the years are slightly more pronounced than those in the neural networks, but less so than those of the `Forestobject` model. Next, we can look at the differences between the neural networks. In 2019 the performances of the neural networks are almost exactly equal. The `NN1` network performs better than the other models in 1930 and 1940, after which it becomes slightly worse than the other networks. The `NN2` and the `RNN2` model are never far apart in terms of performance, with the largest difference between them in 1990. The `RNNall` model has the worst performance compared to the other neural networks in 1930, and becomes similar in score to the `NN2` and `RNN2` model in 1950. Its performance is better than the others from 1980 on. This means that it benefits from having less maps as input, rather than more. To summarise, the `NN1` model shows a downward trend in the ranking over time, while the `RNNall` model shows an upward trend. The other two models remain either in the middle of the rankings, or perform slightly better than the other two. These rankings are based on the averages over the different classes, which might even out differences that are especially pronounced for specific classes.

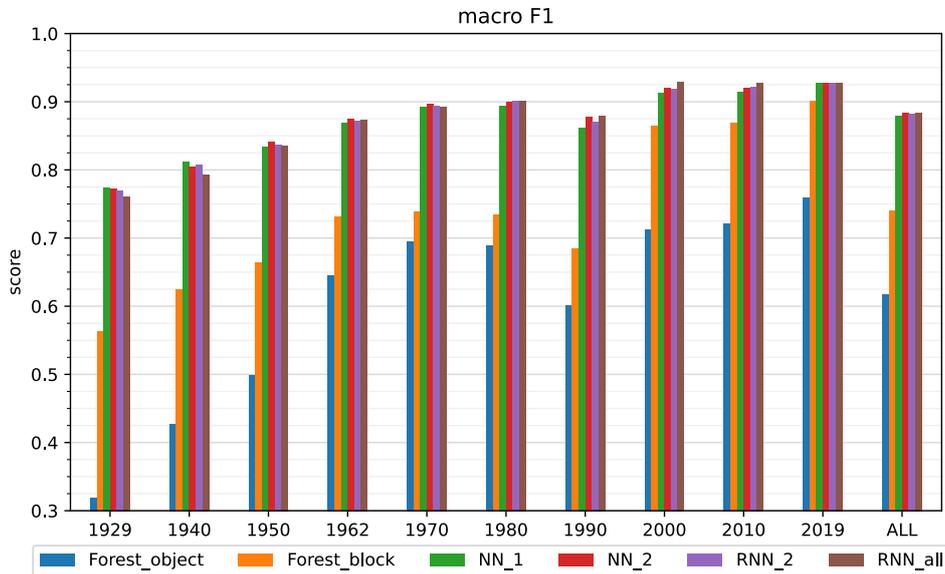


Figure 11: The macro F1 scores for the segmentation task over time.

To get more insight into the differences between the classes they are examined individually, starting with the water class. The F1 scores over time for the water class can be seen in Figure 12. The general trend that can be seen in the macro F1 scores is also present here, with lower scores for 1930 which rise until 2000, where it stabilises. While the scores in the recent years are quite similar to the macro score, the lower scores from 1930 to 1950 are much lower than the macro scores. The performance of the `Forestblock` model is quite close to that of the neural networks, especially from 1950 on. The `Forestobject` model performs better than its macro score, but it remains behind the other models. There are two years where the `NN1` model performs worse than the other neural

networks which stand out, namely 1950 and 1990. The RNN_{all} model performs at roughly the same level as the other neural networks with multiple inputs, except for 1940; there it performs much worse, at about the same level of the $Forest_{block}$ model.

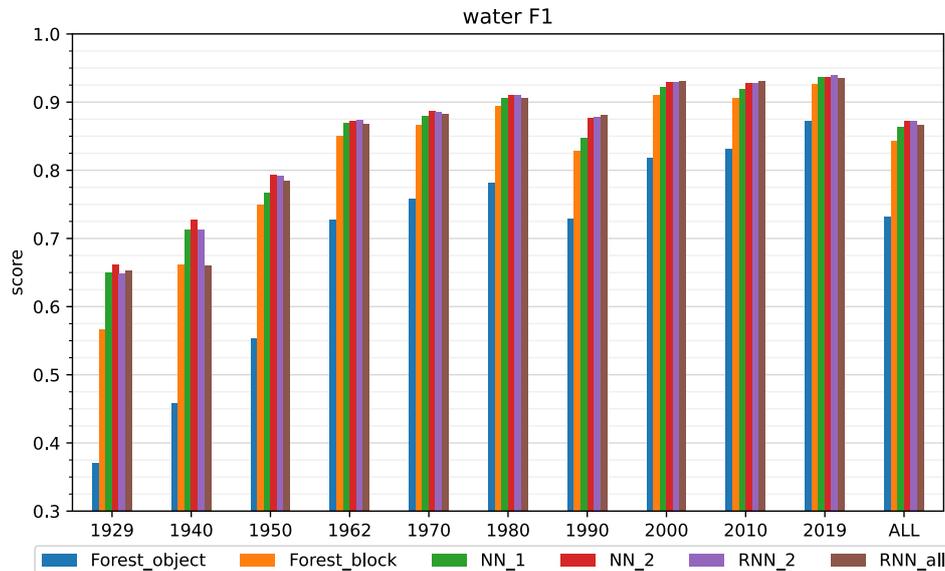


Figure 12: Macro F1 scores over time for the water class.

Secondly, the orchard class is examined. The scores over time for the orchard class can be seen in Figure 13. Two points stand out: First, that the scores of the neural networks do not fluctuate much over time. There are two blocks which have similar scores, namely from 1930 to 1990 and from 2000 to 2019. Only 1930 and 1990 have slightly lower scores. The second point is that the baseline models perform much worse from 1930 to 1990 compared to 2000 to 2019. Especially the $Forest_{block}$ model shows much improvement from 2000 on. There are no meaningful differences to be seen between the neural networks.

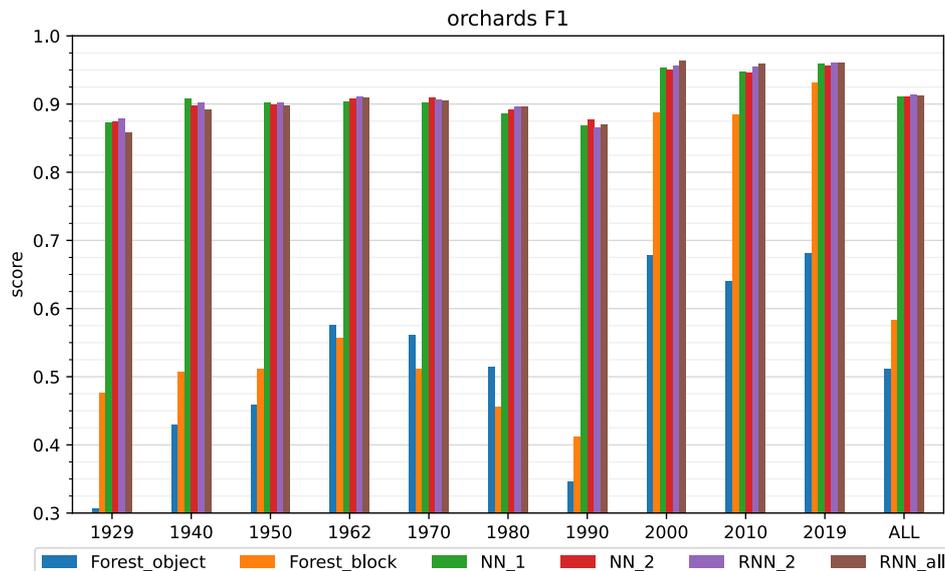


Figure 13: Macro F1 scores over time for the orchard class.

Finally, the building class is examined. The scores over time for the building class can be seen in Figure 14. This is the only class where the scores are not at their highest in the period between 2000 and 2019; instead, the years 1970 and 1980 provide the highest scores. The period from 1930 to 1960 has lower scores, with an upward trend towards the more recent years. The slope of this

trend is not very steep, however. The NN_1 model performs best in 1930 and 1940, and comparable to the other neural networks otherwise. The RNN_{all} model performs the best from 1980 to 2010, and in 1940 as well.

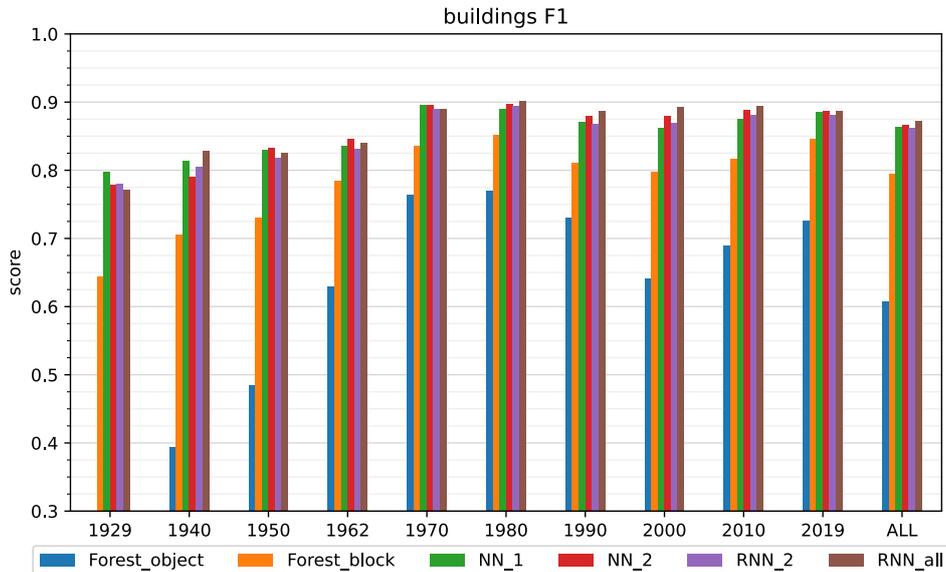


Figure 14: Macro F1 scores over time for the building class.

From the graphs of the individual classes, the trends in the macro scores can be explained in more detail. Firstly, the poor performances of the baselines can for quite a large part be attributed to their low scores on orchards between 1930 and 1990. This is especially the case for the $Forest_{block}$ model, which performs much closer to the neural network models otherwise. The $Forest_{object}$ model consistently reflects shared changes in the performance of the other models with larger intensity. This includes the decrease in performance from 1930 to 1960 for water and buildings and the decreases in performance for water and orchards in 1990. As such we can conclude that the observed behaviour in the macro score did not happen by chance because of a product of different trends, but rather that it is a continuation of the same behaviour for all classes. Compared to the other models, the RNN_{all} model consistently performs better in later years and worse in earlier years. This is in line with the trend seen in the macro scores.

The trends that are seen above can be traced back to the characteristics of the data. A general trend that is seen is that the early years from 1930 to 1950 are often classified worse. A probable cause for this is that the maps from these years are less consistent; while later years use the same style all over the country, the early ones can differ per location. It is also more likely that subsequent years use the same map, which results in less training data being available. Let us now look at the findings per class. First of all, there are trends in the water class. This class is classified worse in the early years from 1930 to 1950 and shows a drop in performance in 1990 as well. A probable cause for this is that ditches are often not coloured in on those maps, or only in a subtle way. The way in which ditches are coloured also changes between the different maps of 1930 to 1950, while the later years have a consistent style. Furthermore, there are fewer pixels in those years that show water, which results in even fewer examples per graphical style. Another noticeable point is that the drop in 1990 is less pronounced for the neural networks that receive input from multiple maps, compared to the NN_1 model. This is likely because the locations of the watered ditches in 2000 aid the recognition of those in 1990. Secondly, there are the orchards. This class is classified markedly worse from 1930 to 1990 by the baseline classes. A probable cause for this is that these years only use a dot pattern to denote orchards, while from 2000 to 2019 they are also marked with a different colour. This pattern might be more difficult to detect for the baselines, as they do not have a large receptive field. The improvement from the easier pattern can also be seen in the performance of the neural networks. The slightly lower performance in 1990 may be caused by the relatively small dots that are used in those maps. Thirdly, there are the buildings. The best performing years here are 1970 and 1980, after which the performance drops a bit. This drop

is likely caused by the fact that from 1990 on the buildings are marked in black, while previously they were marked in red. There are more markers on the map which are also black, which can make it more difficult to distinguish them. The lower performance in the years before 1970 can be either due to the varying graphical styles, or because there are simply fewer buildings on those maps. Both result in smaller amounts of training data. The changing graphical style may also account for the superior performance of NN_1 in 1930 and 1940; the conflicting information about the locations of buildings may have had a detrimental effect on the models with multiple maps as input.

Class	Task	Model	Accuracy	Recall	Precision	IoU	F1
Building	Seg	Forest _{block}	98.7 (0.4)	94.3 (9.6)	76.9 (2.0)	73.5 (7.5)	84.7 (5.2)
		Forest _{object}	96.3 (0.3)	88.1 (8.5)	51.9 (2.5)	48.4 (4.6)	65.0 (4.4)
		NN_1	99.2 (0.3)	94.2 (7.2)	86.7 (1.0)	82.3 (6.3)	90.3 (4.0)
		NN_2	99.2 (0.3)	94.7 (7.5)	87.1 (1.0)	83.1 (6.6)	90.8 (4.1)
		RNN ₂	99.3 (0.3)	92.9 (8.4)	88.9 (1.0)	83.3 (7.5)	90.9 (4.7)
		RNN _{all}	99.3 (0.3)	94.7 (7.8)	88.6 (0.9)	84.4 (7.0)	91.5 (4.3)
	Both	NN_2	99.2 (0.3)	93.4 (8.6)	86.7 (1.2)	81.8 (7.5)	89.9 (4.8)
		RNN ₂	99.2 (0.3)	93.4 (8.1)	86.4 (1.1)	81.4 (7.1)	89.7 (4.5)
		RNN _{all}	99.2 (0.3)	92.7 (8.9)	87.2 (1.2)	81.5 (7.8)	89.8 (5.0)
Water	Seg	Forest _{block}	98.5 (0.4)	91.4 (6.2)	84.4 (1.0)	78.3 (5.2)	87.7 (3.4)
		Forest _{object}	97.0 (0.3)	80.5 (5.1)	72.5 (1.3)	61.9 (3.9)	76.2 (3.1)
		NN_1	98.9 (0.5)	90.2 (7.9)	91.8 (0.8)	83.5 (7.3)	91.0 (4.6)
		NN_2	98.9 (0.4)	92.8 (7.0)	89.4 (0.8)	83.7 (6.3)	91.1 (3.9)
		RNN ₂	99.0 (0.5)	92.3 (7.7)	90.8 (0.8)	84.5 (7.0)	91.5 (4.3)
		RNN _{all}	98.9 (0.5)	92.7 (7.8)	89.4 (0.9)	83.5 (7.0)	91.0 (4.4)
	Both	NN_2	98.8 (0.5)	90.1 (7.9)	88.9 (1.0)	81.0 (7.1)	89.5 (4.6)
		RNN ₂	98.8 (0.4)	90.4 (7.9)	88.9 (1.0)	81.2 (7.1)	89.5 (4.6)
		RNN _{all}	98.7 (0.5)	88.7 (9.4)	89.2 (1.2)	80.1 (8.4)	88.9 (5.5)
Orchard	Seg	Forest _{block}	92.0 (1.4)	80.8 (14.2)	56.9 (4.7)	50.0 (8.8)	66.6 (8.3)
		Forest _{object}	87.4 (0.7)	77.2 (7.0)	42.9 (2.3)	37.9 (3.5)	54.9 (3.7)
		NN_1	98.5 (0.2)	92.5 (2.4)	92.2 (0.2)	85.8 (2.3)	92.3 (1.3)
		NN_2	98.5 (0.2)	93.2 (2.4)	91.6 (0.2)	85.9 (2.3)	92.4 (1.3)
		RNN ₂	98.5 (0.2)	93.1 (2.3)	92.1 (0.2)	86.2 (2.1)	92.6 (1.3)
		RNN _{all}	98.5 (0.3)	92.2 (2.8)	93.2 (0.2)	86.4 (2.6)	92.7 (1.5)
	Both	NN_2	98.2 (0.3)	92.5 (2.5)	89.7 (0.2)	83.6 (2.2)	91.1 (1.3)
		RNN ₂	98.3 (0.3)	92.9 (2.5)	90.3 (0.2)	84.5 (2.3)	91.6 (1.4)
		RNN _{all}	97.9 (0.4)	89.1 (3.8)	90.3 (0.4)	81.3 (3.4)	89.6 (2.2)
Macro	Seg	Forest _{block}	96.4 (0.7)	88.8 (10.0)	72.7 (2.6)	67.3 (7.2)	79.7 (5.6)
		Forest _{object}	93.6 (0.5)	81.9 (6.9)	55.8 (2.0)	49.4 (4.0)	65.4 (3.7)
		NN_1	98.9 (0.3)	92.3 (5.9)	90.2 (0.6)	83.9 (5.3)	91.2 (3.3)
		NN_2	98.9 (0.3)	93.6 (5.6)	89.4 (0.7)	84.2 (5.0)	91.4 (3.1)
		RNN ₂	98.9 (0.3)	92.8 (6.1)	90.6 (0.7)	84.6 (5.6)	91.6 (3.4)
		RNN _{all}	98.9 (0.4)	93.2 (6.1)	90.4 (0.7)	84.7 (5.5)	91.7 (3.4)
	Both	NN_2	98.7 (0.3)	92.0 (6.3)	88.5 (0.8)	82.1 (5.6)	90.2 (3.6)
		RNN ₂	98.7 (0.3)	92.2 (6.2)	88.5 (0.8)	82.4 (5.5)	90.3 (3.5)
		RNN _{all}	98.6 (0.4)	90.1 (7.3)	88.9 (0.9)	81.0 (6.6)	89.4 (4.2)
Micro	Seg	Forest _{block}	96.4 (0.7)	86.9 (10.7)	68.4 (2.9)	62.0 (7.6)	76.5 (6.1)
		Forest _{object}	93.6 (0.5)	80.6 (6.6)	51.9 (2.1)	46.1 (3.8)	63.0 (3.6)
		NN_1	98.9 (0.3)	92.4 (4.9)	91.2 (0.4)	84.8 (4.5)	91.8 (2.7)
		NN_2	98.9 (0.3)	93.5 (4.8)	90.4 (0.5)	85.0 (4.3)	91.9 (2.6)
		RNN ₂	98.9 (0.3)	93.1 (5.0)	91.3 (0.4)	85.5 (4.6)	92.2 (2.7)
		RNN _{all}	98.9 (0.4)	93.0 (5.2)	91.2 (0.5)	85.3 (4.8)	92.0 (2.8)
	Both	NN_2	98.7 (0.3)	92.2 (5.2)	89.3 (0.6)	82.9 (4.6)	90.7 (2.8)
		RNN ₂	98.7 (0.3)	92.6 (5.0)	89.3 (0.5)	83.3 (4.5)	90.9 (2.8)
		RNN _{all}	98.6 (0.4)	89.9 (6.3)	89.6 (0.7)	81.4 (5.7)	89.7 (3.6)

Table 7: Segmentation scores with a buffer of 2 pixels, as percentages. The improvement with respect to the non-buffered performance is given in brackets.

5.2.2 Buffered Performance

It is of value to know whether lower scores are caused by actual mistakes by the model, and which are caused by imprecision by either the models or the ground truth. To get more insight into this, the performance of the models when given a buffer of two pixels around their predictions is shown in Table 7. This buffer is only counted when it is a correct prediction, to prevent a decrease in the precision score. Because of this, all scores can only improve; this improvement can be seen in brackets behind the mean score. As this buffer only gives an increase in the true positives and a decrease in the false negatives, the changes can be seen most strongly in the recall scores. Let us first look at the macro scores. The two baseline models have the largest improvement in their scores, closely followed by models that are trained on both tasks, and the neural networks trained on only the segmentation task have the least improvement. The general trend seems to be that the models that performed the worst show the most improvement. The IoU scores show a sharper increase in performance than the F1 scores. This metric punishes mistakes more harshly, but ends at a score of 1 for a perfect segmentation just like the F1. This means that at higher scores it improves faster than F1. The accuracy does not change as much as the other scores because only a small percentage of the pixels belongs to one of the segmentation classes. Looking at the individual classes, one can see that the trend of improvement is not the same over the classes and the models. All models benefit with 3% to 5% in their F1 scores in the buildings and water classes. In the water class the neural networks benefit more from the buffer than the baselines, with about 1% more improvement on average. The orchard class shows a marked difference between the improvements of the baselines and those of the neural networks. The baselines have a sharp increase in score, particularly the $\text{Forest}_{\text{block}}$ model; it has an increase in its recall of more than 14%. The neural networks, on the other hand, show less improvement in this class compared to the other classes.

An explanation for the differences in the improvement scores can be found in both the nature of the output of the models, and in the way the ground truth was created. Firstly, there are the large improvements for the baselines in the orchard class. This class consists of large areas, meaning that changes around the edge of an orchard do not influence the overall score as much. A trend that is seen in particular with the $\text{Forest}_{\text{block}}$ model is that it often has holes in its predictions for this class. An example of this behaviour can be seen in Figure 15. This is likely because it has a small receptive field, and therefore cannot link the gaps between the dots to the dots themselves. The $\text{Forest}_{\text{object}}$ model can have similar issues if the area between dots is marked as a different object than the dots themselves. These gaps can be filled up with the buffer, causing a large increase in performance.

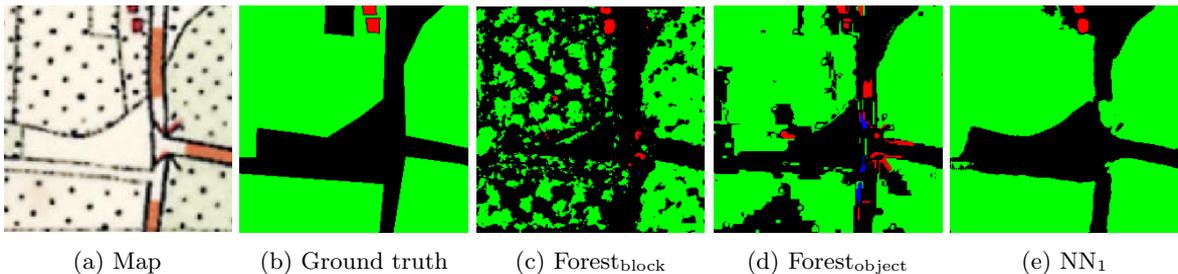


Figure 15: Example outputs of several models showing how they segment orchards. These outputs are not buffered.

Next, there are the substantial increases in performance in both the building and the water classes. These can be partially explained by the way in which the ground truth is constructed; the shapes from which the objects are made always have straight lines, as they are made by connecting corner points. This means that objects that have soft edges are often drawn slightly too large in the ground truth. The effect is especially noticeable in objects which have small surface areas, such as buildings and waterways. An example of this can be seen in Figure 16. Because the borders of the ditches and the buildings are blurry due to the low graphical quality, the size of these objects becomes even more subjective. The models tend to output softer borders than the ground truth, meaning that a buffer around their predictions helps to classify objects that they have identified more accurately.

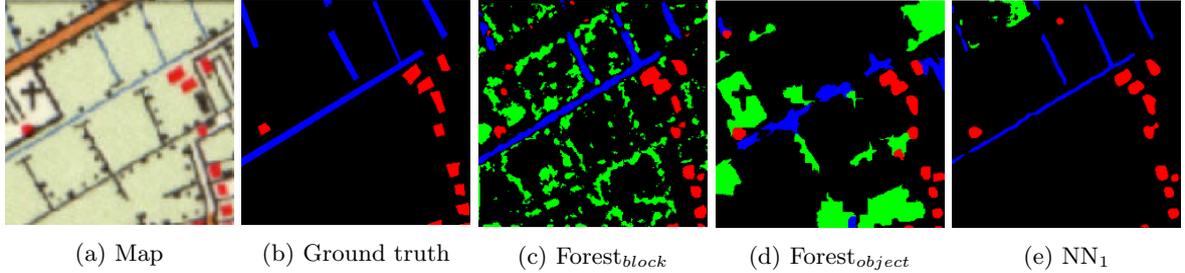


Figure 16: Example outputs of several models showing how they segment buildings and water. These outputs are not buffered.

5.3 Change Detection

The scores for the change detection task can be seen in Table 8. A repeated measures ANOVA test is performed on the F1 scores to see whether the performances of the models differ. It shows that there is a significant difference in performance [$F(7, 28)=143.7$, $p<0.001$, $\eta^2=0.97$]. In order to find out which models differ from which, several pairwise T-tests with FDR-BH correction are performed. To start with, the baseline models perform significantly worse than almost all neural network models. Only the RNN_{all} model that was trained on the change detection task does not perform significantly better than the $Forest_{block}$ model [$t(4)=2.3$, $p\text{-corr}=0.19$, $g=-0.98$]. The comparisons between all other models yield corrected p-values lower than 0.001. The effect sizes are also quite large, ranging from -5.1 to -14.4 . The largest effect sizes can be found between the baselines and the models that were trained on both tasks. These are between -7.1 and -9.3 for the $Forest_{block}$ model and -12.0 to -14.4 for the $Forest_{object}$ model, compared to -5.1 to -5.6 and -5.1 to -10.3 when tested against the single-task models. This means that there is a larger difference between the behaviour of the baselines and that of the models trained on two tasks, compared to the models trained on one task. The full results of the T-tests can be found in Appendix A in Table 12. The two baseline models also perform significantly different from one another in a two-tailed T-test [$t(4)=30.2$, $p\text{-corr}<0.001$, $g=6.3$]. From Table 8 it can be seen that the $Forest_{block}$ model performs better on average.

Task	Model	Accuracy	Recall	Precision	IoU	F1
CD	$Forest_{block}$	89.3 (0.6)	53.3 (1.2)	59.9 (3.8)	39.3 (1.7)	56.4 (1.8)
	$Forest_{object}$	80.4 (0.7)	52.9 (0.8)	33.8 (3.4)	25.9 (2.0)	41.2 (2.5)
	NN_2	93.0 (0.2)	64.4 (4.0)	77.3 (2.5)	54.2 (3.5)	70.2 (2.9)
	RNN_2	93.1 (0.3)	63.6 (3.5)	79.1 (2.9)	54.4 (3.2)	70.4 (2.7)
	RNN_{all}	91.2 (0.5)	50.8 (4.7)	72.9 (6.1)	42.6 (3.9)	59.7 (3.9)
Both	NN_2	93.3 (0.3)	67.9 (0.7)	77.7 (3.4)	56.8 (1.5)	72.4 (1.3)
	RNN_2	93.4 (0.5)	65.0 (3.0)	80.3 (2.3)	56.0 (2.6)	71.8 (2.1)
	RNN_{all}	92.9 (0.7)	63.9 (3.2)	77.1 (3.1)	53.6 (2.1)	69.8 (1.8)

Table 8: Change detection scores, given as percentages.

Aside from the baselines, several comparisons between the neural networks are also made. First of all, the difference between the NN_2 and RNN_2 models is tested. The assumption was made that the RNN_2 model would perform better. This is not the case, as the results of the one-tailed T-test are not significant for both the change detection-only models [$t(4)=-0.43$, $p\text{-corr}=1$, $g=-0.06$] and for the models trained on both tasks [$t(4)=0.76$, $p\text{-corr}=1$, $g=0.33$]. In fact, the NN_2 model performed better when trained on both tasks.

Secondly, the difference between the RNN_2 and the RNN_{all} model is tested. No particular hypothesis was made whether having more maps as input would assist change detection, as changes from another year are not closely connected to changes in the current year. As such, the two are compared with a two-sided T-test. For the change detection task there was a significant difference [$t(4)=4.7$, $p\text{-corr}=0.01$, $g=2.88$], while for the models trained on both tasks there was not [$t(4)=1.21$, $p\text{-corr}=0.37$, $g=0.92$]. The RNN_{all} model performed much worse in the change detection-only task (70% against 60%), and slightly worse when both tasks were trained (72% against 70%).

Finally, the difference between the models that were trained on the change detection task and those trained on both tasks is tested. The assumption is that training on both tasks improves the performance. There was a significant improvement only for the RNN_{all} model [$t(4)=5.5$, $p\text{-corr}=0.007$, $g=2.98$], from 59.7% to 69.8%. For the NN_2 model there was not [$t(4)=2.0$, $p\text{-corr}=0.11$, $g=0.88$], with an improvement from 70.2% to 72.4%. There was no significant improvement either for the RNN_2 model [$t(4)=1.1$, $p\text{-corr}=0.26$, $g=0.51$], with an improvement from 70.4% to 71.8%. This improvement mostly comes from the higher recall scores. Another point of note is that the standard deviations of the models trained on both tasks are lower than their counterparts: the NN_2 model goes from 2.9 to 1.3, the RNN_2 model goes from 2.7 to 2.1, and the RNN_{all} model goes from 3.9 to 1.8. This means that their improved performance is also more consistent.

An example of the model outputs can be seen in Figure 17. Of note is the difference in which the models segment the orchards in the upper middle and right; the models trained on both tasks identify both of them properly, while the other models do not. More examples can be found in Appendix C. In general, the best performing model on this task is the NN_2 model trained on both tasks, with a mean F1 score of 72.4%. It is followed by the RNN_2 model, also trained on both tasks, with a mean F1 score of 71.7%.

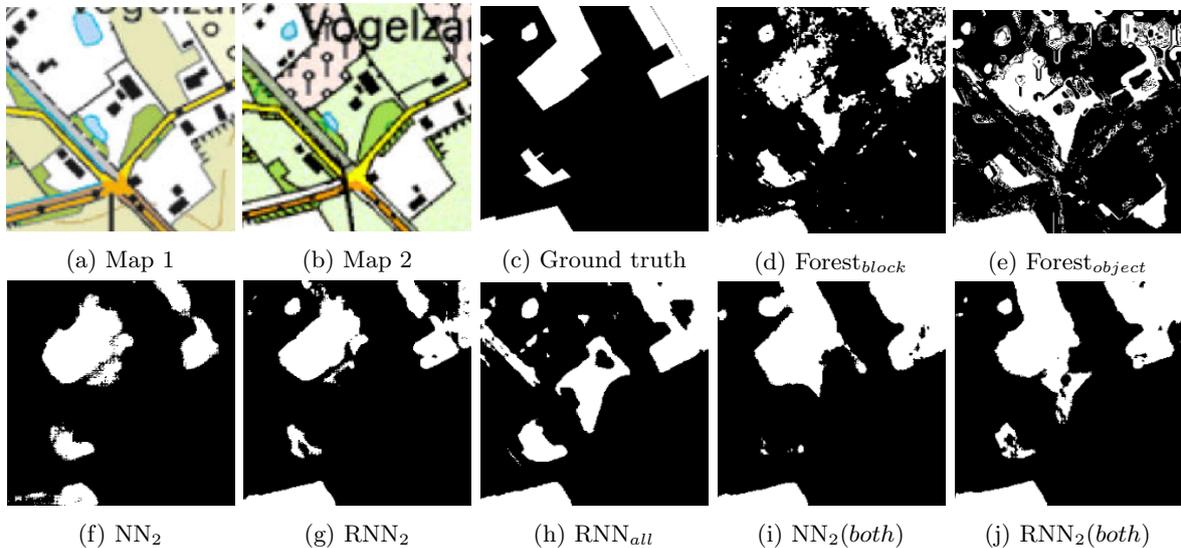


Figure 17: Example outputs. Note the different segmentation of the orchard in the upper centre between the models trained on both tasks and those trained only on change detection.

5.3.1 Yearly Performance

It is interesting to see whether there are differences between the performances in change detection over the years, seeing as the similarity and deviations in georeferencing of neighbouring decades are not stable over the years. The performances of the models trained only on change detection are shown in Figure 18. The models that were trained on both tasks are not shown because they show the same pattern as their counterparts, only with slightly higher scores. The change detection is done between a year and the decade after it; for example, the scores in 1940 denote the differences between 1940 and 1950. A general pattern that can be seen is that the performances of all models improve over the years, with the lowest scores in 1930 and the highest in 2000. The scores in 2010 are slightly lower than those of 1990 and 2000. The RNN_{all} model performs much worse than the other neural networks in the early years. This difference diminishes over the years, and its performance is comparable to the other two in 2000 and 2010. The $\text{Forest}_{\text{block}}$ model shows a sharper increase in performance over the years than the neural networks; in 1929 the difference between them is almost 20%, while in later years it is closer to 5%.

The lower performance of all models in the earlier maps can be traced back to the fact that the georeferencing of those maps often does not match well. The changing sizes and locations of features make it difficult to discern whether a feature has changed or whether only its representation has changed. The drop in performance in 2010 cannot be explained as easily; the georeferencing is very similar between 2010 and 2019, and the styles are almost the same. It could be that because

of these properties the ground truth contains smaller differences, as it was easier for the author to detect changes in this year. However, 2010 does not contain more changes than the years before it, as can be seen in Table 2. As the objective number of changes is unknown, this therefore remains speculation. The changing difference between the Forest_{block} model and the neural networks can be traced back to the fact that it classifies changes based on the graphical differences between the maps. The earlier maps differ more in graphical style than the later maps, which makes it harder to discern for the model whether something is an actual change in the landscape or not.

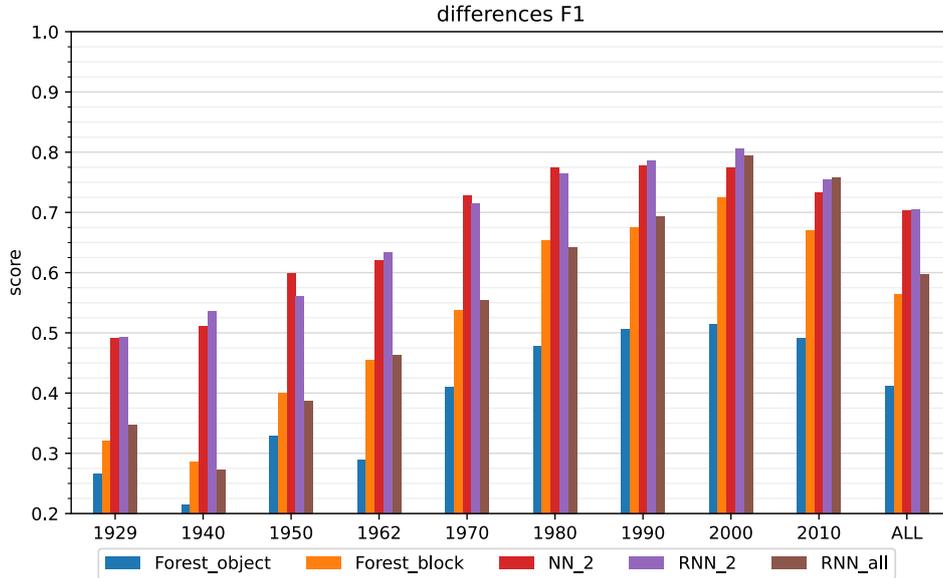


Figure 18: F1 scores over time for change detection.

5.3.2 Buffered Performance

Imprecision is especially difficult for change detection, where imprecise georeferencing can lead to uncertainty as to whether and where the landscape has changed. The impact of such problems cannot be traced completely, but the performance of the models with a buffered prediction can give some insight into how many mistakes were made due to slight imprecision. The buffered performances can be seen in Table 9. The buffer only improves the scores because the additional false positives are not counted; this improvement is shown in brackets behind the scores. The increase is mostly seen in the recall scores, as the buffer increases the number of true positives and reduces the number of false negatives. A general trend that can be seen is that the models that scored lower in the regular results have a larger gain in the buffered results. This can be seen in the difference between the neural networks trained on the change detection task and their counterparts that are trained on both tasks. Especially the Forest_{block} model has a large increase in performance, from an F1 of 56% to one of 67%. This is followed by the Forest_{object} model which has an increased performance from 41% to 48%. It should be noted that these models have a relatively low precision in the normal condition, which means that many pixels are classified as changed when they are not. This gives them a larger buffer than the other models, which can aid the detection of truly changed pixels without the additional penalty of decreased precision. This is similar to the pattern that was shown for orchards in Section 5.2.2. The increases for the neural networks are similar to their improvements in the segmentation macro F1 scores seen in Table 7.

All changes are grouped together in a single class, meaning they cannot be separated by shape, size and land use like the segmentation classes can. It can therefore not be said with certainty whether these characteristics influence the performances of the models. From observing the predictions it seems that it matters which type of land use is changed to which, and how large the change is. For example, a ditch that changes to a meadow is more easily missed than a meadow that is replaced with a residential area. Exact data on these differences cannot be given, but some examples can be found in Appendix C.

Task	Model	Accuracy	Recall	Precision	IoU	F1
CD	Forest _{block}	91.2 (1.9)	67.9 (14.6)	65.5 (5.6)	50.0 (10.8)	66.7 (10.3)
	Forest _{object}	82.0 (1.5)	64.7 (11.8)	38.4 (4.6)	31.7 (5.8)	48.1 (7.0)
	NN ₂	93.6 (0.6)	69.0 (4.6)	78.5 (1.2)	58.0 (3.9)	73.4 (3.2)
	RNN ₂	93.8 (0.7)	68.6 (5.1)	80.3 (1.2)	58.8 (4.3)	74.0 (3.6)
	RNN _{all}	91.9 (0.7)	56.3 (5.5)	74.9 (2.0)	47.3 (4.6)	64.1 (4.4)
Both	NN ₂	93.9 (0.6)	72.3 (4.5)	78.8 (1.1)	60.5 (3.8)	75.4 (3.0)
	RNN ₂	94.0 (0.6)	69.6 (4.6)	81.4 (1.1)	60.0 (4.0)	75.0 (3.2)
	RNN _{all}	93.6 (0.7)	69.4 (5.5)	78.6 (1.4)	58.3 (4.6)	73.6 (3.8)

Table 9: Change detection scores with a buffer of 2 pixels, given in percentages. The improvement with respect to the non-buffered score is given in brackets.

5.4 Segmentation and Change Detection

The scores of the combined task are shown in Table 10. The macro score refers to the average of the macro segmentation score and the score of the change detection. The individual scores can be found in Tables 6 and 8, respectively. The micro score is computed over all pixels, meaning that the segmentation score has a higher weight. A repeated measures ANOVA is performed on the macro F1 score to see whether there are any differences between the performances of the models. This shows that there is no significant difference between them [$F(2, 8)=1.48$, $p=0.28$, $\eta^2=0.27$]. The η^2 is also quite low, suggesting that any effect that could be there is very small. With this being the case, there is no reason to perform additional T-tests on the individual models.

The roughly equal performances are line with the results of the separate tasks, where the models did not perform significantly different from one another either. The NN₂ and the RNN₂ models have negligible differences in their performance. The models that receive input from two maps perform slightly better than the RNN_{all} model, with macro F1 scores of 79.5% and 79.3% compared to one of 77.5%. This effect is more prominent in the recall scores, which can be seen in both the segmentation scores in Table 6 and in the change detection scores in Table 8. The performance of these models compared to their counterparts, which are only trained on a single task, differs per task. While on the segmentation task they perform worse, on the change detection task they perform better. It can therefore not be said one way or the other whether they are better or worse than their counterparts; it depends on which task is deemed more important.

Class	Model	Accuracy	Recall	Precision	IoU	F1
Macro	NN ₂	95.8 (0.1)	76.8 (0.8)	82.7 (1.9)	66.7 (0.8)	79.5 (0.6)
	RNN ₂	95.9 (0.2)	75.5 (2.9)	84.0 (1.9)	66.4 (2.1)	79.3 (1.6)
	RNN _{all}	95.5 (0.5)	73.3 (2.3)	82.5 (1.7)	64.0 (1.9)	77.5 (1.4)
Micro	NN ₂	97.1 (0.1)	79.5 (0.9)	84.7 (2.1)	69.5 (1.4)	82.0 (1.0)
	RNN ₂	97.2 (0.1)	78.8 (2.5)	85.8 (2.0)	69.7 (2.0)	82.1 (1.4)
	RNN _{all}	96.8 (0.5)	76.0 (3.2)	84.6 (1.6)	66.7 (2.7)	80.0 (2.0)

Table 10: Averaged scores for both tasks. The macro score averages the macro segmentation score and the change detection score, while the micro score weighs all pixels equally.

5.5 Overarching Results

Having seen the performances of the models on all test conditions, it is now possible to say something about the performances of the models in general. Firstly, the Forest_{block} model performs relatively well on small features, and performs worse on larger ones that require the recognition of a pattern. This is in line with expectations, as it only has a perceptive field of 7×7 pixels. It is also relatively slow to train compared to the fastest neural network models. Secondly, the Forest_{object} models consistently performs the worst out of all models. Although the OBIA technique works well for satellite images, it seems to have difficulty with drawn map data. Aside from distracting factors and the low graphical quality, the method also suffers from the type of features that need to be identified. Normally a segmentation algorithm does not count borders of large patches as separate objects, but it is forced to do so in this case because otherwise none of the waterways

would be detected. This high sensitivity leads to small object sizes that do not reflect the size and shape of the actual object. An example of this can be seen in Figure 19, which shows that the segmentation contains many more objects than necessary. It can also be seen that parts of the ditches are not classified correctly because they are part of too large an object, and that parts of the orchard are not classified because their objects are too small.

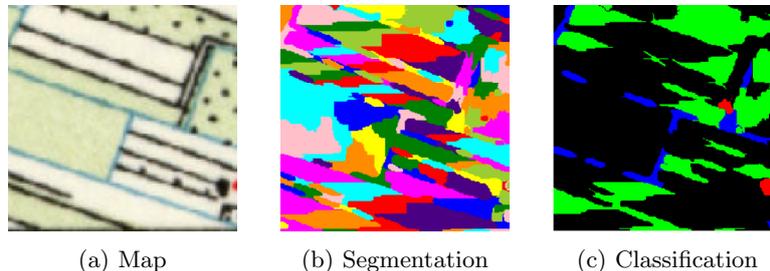


Figure 19: Example segmentation and classification of the *Forest_{object}* model. The colours in (b) only show the different objects, not classes.

Let us now look at the neural networks, starting with the NN_1 model. This model can only be used for semantic segmentation, and is the fastest neural network for that task. Its performance is a little worse than the other neural networks when those perform only the segmentation task, but not significantly. This dip in performance is mostly seen in specific combinations of classes and years where the features are not presented clearly, such as the waterways in 1990.

The NN_2 model, which combines multiple maps with concatenation, performs well at both individual tasks and the combined task. Its training time increases as the task gets more complicated: while the segmentation task took 20 minutes on average, the change detection task took 30 and the combined task 40. Compared to the other neural networks it tends to have a slightly higher recall, but a lower precision.

The RNN_2 model, which combines two maps with a CGRU, performs equally well on the individual tasks and the combined task. Because the CGRU component has more parameters, it is slower to train; about 55 minutes for each task. There is only a few minutes difference between the different tasks. It does not have any particular trends in its precision or recall scores, remaining close to the scores of the other models with multiple maps as input. There is no significant difference between its performance and that of the NN_2 model.

The RNN_{all} model receives input from all maps that are more recent than the segmented map, in addition to the segmented map. It is relatively slow to train, with 2 and a half hours training time for the individual tasks and almost 3 and a half hours for the combined task. This makes it slower than both the baselines and the other neural networks. In the semantic segmentation task it is one of the best models, although its performance gets comparatively slightly worse as the maps get older. In the change detection task it performs much worse than the neural networks when trained on only the change detection task, while it performs almost as good as the other two neural networks when trained on both tasks.

6 Discussion

The results show that the models are able to generalise features on maps spanning 90 years. The scores for semantic segmentation are quite high at a macro F1 score of more than 88%, which increases to 91.5% when the evaluation is set to be more lenient by using a buffer around the predictions. The change detection scores are lower than the segmentation scores but still usable with an F1 score of 72%. This is also lower than the change detection scores that are achieved with remote sensing data [40][12][42], likely due to the additional challenges of imprecise georeferencing and changing graphical styles. The scores are not equal over the years; the earlier years have lower scores, particularly for change detection and the segmentation of water. The impact of the timeframe is not the same for all classes, meaning that this should be examined individually for each class that is added to the program in the future. This study set out to examine three main questions about the influences on the models: whether input from maps from different timeframes supports the semantic segmentation of a single map, whether there is a difference between the two methods of combining the maps, and whether performing both tasks has any influence on the general performance of the tasks. We will go over the answers to these questions in sequence.

Multi-temporal Input

The first matter of interest is to see whether input from other timeframes of the same location has a positive influence on the semantic segmentation of a map. The models that received both their own map and the map from ten years later perform somewhat better than the model that receives only one map as input, although this difference is not significant. From the yearly results it can be seen that the advantage of two-map models is mostly present in years where the representation of a feature is unclear, such as years where ditches are not coloured as water. Contrary to expectations, the input of multiple years does not help substantially with the avoidance of distractors, as that would have resulted in a general increase in performance. This means that the use of multiple years as input will mostly be helpful for similar projects when segmenting maps on which the visibility of features differs between the years. The models that received additional input from all years before the segmented map perform worse on maps from long ago, which have many maps as input. This means that the input from many maps may in fact be detrimental to the performance of a model, possibly because changes in the landscape over the years provide too much contradicting information. Another cause for this behaviour could be the fact that a GRU variant is used in the RNN model; this component has no way to separate information for later maps in the chain and information that is relevant for the current output. It could be that using an LSTM variant will reduce the unwanted side effects of using many maps as input, as such a unit can separate the two streams of information better. Comparing these two variants in further studies should shed more light on what causes the decreasing performance. Such comparisons have been done in studies with two maps, like the work of Mou et al. [52] and that of Chen et al. [12]. Nevertheless, it has not been done before with more than two maps, where the difference between the two architectures should be more apparent. Another point of interest is that the additional maps that were provided in the experiments were always more recent than the segmented map. For further studies it could be interesting to see whether input from the decade before the segmented map could have some positive influence as well. Having both the map from a decade before and the one from a decade after the segmented map could help to filter out which features are relevant, as features that are present both a decade before and a decade later are more likely to be present on the segmented map. This was not tried in this thesis, as the sequential nature of the GRU elements does not lend itself to output about the middle of the sequence.

Combining Maps

This brings us to the second research question pertaining to the difference between the models with a CGRU and those that concatenate the two feature maps. There is no discernable difference in performance between these models, in none of the test conditions. While the CGRU has more flexibility in its options to filter out information from other maps, the convolutions in later parts of the model seem to perform this task properly as well. As mentioned previously the addition of more than two maps does not improve the performance either, which was deemed as a possible perk of using a CGRU. This result brings more clarity to the question of which architecture is more effective for the combination of two images. Where most papers that discuss such an

architecture do compare it to other models, they tend not to compare it to the same architecture with and without the RNN component. This trend can be seen in papers which test different RNN architectures [52][12], as well as those that test fully convolutional networks [10][40][70]. Several of these do compare different variants of combination, but limit it to either types of RNN or ‘simple’ types of combination like concatenation and subtraction. With the results of this study in mind, it can be said that the addition of a computationally expensive RNN component is not a necessity for either semantic segmentation or change detection in the presented conditions.

Combining Tasks

Aside from combining the maps, the potential of combining tasks has also been studied. Training on both the segmentation task and the change detection task improves the performance of the models on the change detection task, although this improvement is not significant for most models. It is probable that the improvement mostly comes from the changes that are related to the segmented features. The changes that need to be detected are not limited to the features that need to be segmented, which might explain why the improvement is only mild. It would therefore be interesting to see whether an increase in the number of features that are segmented would result in a larger improvement as well. Such an effect could be seen in the work of Caye Daudt et al. [10], who segment the whole map into land-use types and use that segmentation to inform their change detection. They conclude that that approach performs much better than the model that only performs change detection. The structure of their models differs from the presented models, as the semantic segmentation of the two maps is done separately; they do not receive information about the other map. Based on the results of this study, a more compact model which uses shared paths could be used without impacting the performance and may aid the segmentation of difficult features. The semantic segmentation score did decrease slightly when performing both tasks compared to the models that only segment. Again, this difference was only significant for one of the models. There was no significant difference between the different models that performed both tasks, meaning that concatenating and using a CGRU are equally viable methods to perform the tasks simultaneously.

6.1 Limitations

There are some aspects of this study which were held back either by practical restrictions, or by the choices that were made in the process. These involve either the way in which the experiments were conducted, or the data that was used for training and testing the models.

Experimental Conditions

There are some practical limitations to the experiments, which may have hampered the results. Firstly, the RNN_{all} model does not have the exact same training conditions as the other neural networks. It has a smaller batch size compared to the other models, because insufficient memory was available for the desired batch size. Furthermore, each batch only contains maps of the same year because the training architecture cannot handle items of different sizes in the same batch. Especially the last point may have been detrimental to its learning process, as it makes each batch less representative of the whole dataset.

Secondly, the models were only run five times for the statistical tests due to limited resources. This is quite a low number for ordinary statistical testing, which also means that the statistical power of the comparisons is relatively low. It should be noted that running large neural networks for more than 5 times is not customary due to the long training times and expensive resources. Any effects that were not found to be significant may therefore be still be worth exploring more in future studies, to see whether the insignificance was truly due to there being no difference, or due to the limited statistical power of the tests.

Thirdly, the results of the models are compared to relatively simple baseline models only. The state of the art in semantic segmentation features much larger and advanced neural networks than those presented in this study. However, such models require much more training data and computational power to be retrained than the created models, and as such they are not shown here. Aside from such large models, neural networks from papers that perform similar tasks to the ones shown here could have been used for a comparison. As there was no clear consensus on which was the best-performing model, however, this would have required rebuilding and training

multiple additional models. The resources needed for this were not realistic for the current study. Instead, the models presented aim to represent the different modelling choices that others have used, to provide a clear comparison between the techniques rather than comparing whole models. This does mean that the overall performance of the models cannot be said to be better or worse than that of any state of the art model.

Finally, the way in which the buffers have been implemented may not be ideal. If a model has incorrectly classified a random pixel and correctly classifies a nearby pixel because of the buffer, this leads to a misleading improvement in its score. As the buffer is only two pixels wide this is not likely to happen often, except for cases where a model has very low precision. Still, other kinds of buffer could have been used, such as the approach of Kampffmeyer et al. [33]. They leave the borders of ground truth objects out of the evaluation, meaning that it does not matter whether the model classified it correctly or not. A downside to such an approach is that small objects only have very few pixels left to be classified. The current study has many small or narrow objects, and as such the chosen method of buffering was thought to be more fitting. In retrospect it might have been better to implement the buffers as Kampffmeyer et al. did, considering the low precision of some of the models.

Data

The data was selected and annotated completely by the author. This is a time-consuming task, which led to several choices to keep the work within reasonable bounds. First of all, the test data consists of the same tile size as the training data. One of the advantages of a CNN is that it is able to handle images of different sizes, and the presented models will likely be used for much larger images than those in the training data. The models were not tested on larger maps, however, because this would require annotating much more land to get enough diversity in the landscapes. Another choice that was made was between which years the differences would be annotated. For the purposes of the soil advisors at TAUW it was deemed convenient to know when changes had taken place, which made annotating the differences between adjacent decades the logical choice. However, it does mean that the models at present cannot generalise to differences between any two maps, temporally adjacent or not. This would be an interesting functionality to have if the model is to be used in more general contexts.

There are also some faults to be found in the data that was used. While much effort was made to annotate it as accurate as possible, there are most certainly mistakes in the annotations, especially in the changes. These mistakes may have influenced the results, as it makes training more difficult and the evaluation less accurate. Neural networks are able to work with noisy data, and as such the impact on the training is not assumed to be very large. The evaluation is more problematic; it means that the scores that are given in the results should be taken as indications rather than exact scores. Further improvements to the datasets could help to eliminate this imprecision. Not only the annotations could use improvement; more accurate georeferencing of the maps would facilitate the both the annotation and the training of the networks greatly.

Finally, it cannot be said for certain whether the data is representative of the Dutch landscape in general. The locations that were chosen for annotation are from all over the country, but care was taken to ensure that the segmentation classes were present on enough tiles. Orchards, for example, are much more common in the data than they are in the actual landscape. The only way to ascertain whether it generalises well is to use the program in practical applications.

6.2 Practical Recommendations

Based on the capabilities and limitations of the models that we have seen above, an estimation can be made of how these models could be used in practice; specifically, which model is preferable for which task and timeframe. This is based on the models as they are currently, without any additions or changes.

Let us first look at the case where only semantic segmentation is needed. If orchards or buildings are important, all years can be used reliably without too many changes in the performance. For the water class, on the other hand, the performance drops in 1929 and 1940 to an F1 score of 65% to 70% respectively. This is still usable, but it needs more thorough checking by the user. Regarding the choice of model, the NN₂ model is preferable when features need to be segmented which are unclear in isolated years, such as the water class in 1950 and 1990. If such features are not important or computation time is of the essence, the NN₁ model is the better choice.

Let us now look at the case where only change detection is needed. The performance of all models is worse the earlier in the timeline the maps are; in 1929 the highest F1 score is 50%, which turns into 60% by 1950 and 70% by 1970. This needs to be taken into consideration when using the models for older maps, especially when a high recall is important. The models are better at finding large changes than smaller ones; if only large changes are needed, the models need less supervision. As for the choice of model, the ones that are trained on both the segmentation task and the change detection task perform the best at this task. These are especially recommended if the changes that need to be found are related to the segmented features. The NN₂ model is preferred because of its faster computation times.

Finally, there is the case where both tasks are needed. The same recommendations for time-frames and class importance hold here as in the individual tasks. If the slight decrease in the segmentation scores is not a problem, the models trained on both tasks can be used. In this case the NN₂ model would be preferable as well. The decrease amounts to 1% in the F1 score. If this decrease in performance is not acceptable and computational resources are not an issue, a model trained only on the segmentation task can be used for segmentation, while the both-task model is used for change detection. In this case, the recommendations that are given above hold as well.

6.3 Future Research

Some ideas for future research have been suggested in the paragraphs above; the ones that are deemed to be most interesting are included here. There are three global directions that future research could take based on the results of this study. The first is to shed more light on how the current models work, and the influences on this performance. The second is to improve the performance of the models for the current tasks. The third is to expand the functionality of the models. As for the further study of the current models, the two most important options pertain to the data. First of all, the models could be tested on maps from other parts of the country on which they are not trained. The tiles of these areas can also be larger than those that were used for testing in the current study. Such a test could show whether the models can generalise to other landscapes, which is valuable to know for practical use. A larger step would be to see whether it can generalise to maps of other countries without additional training. Secondly, the georeferencing of the maps could be documented, such that imprecision due to georeferencing can be distinguished. Showing which mistakes are related to the misalignment of features would help to differentiate them from mistakes due to graphical style or other influences, which the current approach with a buffer cannot do. This in turn would give an indication of the importance of improving the georeferencing of the maps. Such work is especially important for the change detection task, but the models which receive multiple maps as input could benefit from it as well. The imprecision that was not caused by imprecise georeferencing can still be studied with a buffer as was proposed in this study. Finally, the influences of RNN components on the performance could be explored further. The RNN_{all} model did not perform well on older maps; it could be that replacing the GRU with an LSTM results in a different pattern due to the separation of the output and the internal state.

There are many ways to improve the current models' performance. Without changing their architecture, they would already benefit from a more accurate training dataset. The changes between maps in particular could be made more consistent and complete, and their relevancy for the desired application could be checked by an expert to avoid trivial changes. The earlier years from 1930 to 1950 have more variability in their graphical styles, and could therefore benefit even more from additional training data. As an addition to the current procedure some post-processing could be used to refine the results, such as Conditional Random Fields to make the model outputs resemble the shape of the features more closely [50]. Finally, the loss function could be modified. The models often have difficulty with smaller changes and objects, such as narrow ditches that have disappeared. If these are given larger weights when computing the loss, the models will learn to discern these features better.

In order to expand upon the existing models, several options can be explored. The most straightforward option is to include more classes for semantic segmentation. For example, roads are useful to identify in many applications, and are frequently involved in landscape changes. An interesting option would be to include distractors such as text as an explicit class, to see whether this helps to counteract their influence. An expansion of classes could also be applied to change detection; rather than a binary classification of changed or not, different classes could represent

which feature has changed into which. A higher number of segmentation classes could introduce an imbalance in the models if the classes are not equally common, as each class has the same weight in the loss function. Other design choices are more appropriate with a high number of classes, such a weighted loss to inflate the importance of rare classes. Another option for expanding the capabilities could be to train them to detect changes between any two maps, not just temporally adjacent ones. This would expand the number of applications for which the models can be used.

7 Conclusion

The original motivation of this study was to automate the retrieval of indicators of soil contamination from historical maps spanning 90 years. As a representative set of the most important indicators, three features were chosen for semantic segmentation: water, buildings and orchards. Additionally, the changes in the landscape had to be determined for each decade. Multiple CNN models were designed to accomplish this, building upon previous work on both historical maps and satellite images. The results show that the models can learn to generalise the features of maps from different decades, and that changes in the landscape can be separated from changes that are caused by graphical styles. This demonstrates that CNNs are a useful tool for extracting information from historical maps. To put their performance into perspective, two baseline models were constructed as well. Both were based on random forests, with either a patch-based or an object-based input. In comparison with the baseline models the CNNs performed significantly better, especially in cases where a larger pattern such as an orchard had to be identified. This shows that using CNNs is especially useful in cases where context information is important.

The goal was not just to build suitable models, but also to examine which kinds of methods work best for these tasks. Three main influences were investigated: multi-temporal inputs, ways to combine multiple maps, and combining the two tasks. The multi-temporal inputs were either an additional map from a decade later, or all maps that were more recent than the segmented one. Giving this additional input for semantic segmentation was shown to be useful only in specific cases, where the segmented feature is drawn ambiguously while it is drawn clearly in the decade after it. The usefulness of the additional input hence depends on the needs of the user. Having more than two maps as input did not result in a larger improvement. For both multi-temporal segmentation and change detection there were two methods of combining multiple maps that were tried; concatenation and a CGRU. There was no significant difference between the two methods, which gives preference to the concatenation method as that one is faster. This resolves some disagreement in the community as to which method of combination should be preferred with image data. The tasks of segmentation and change detection were performed both separately and simultaneously. Performing both tasks slightly increases the performance in change detection, and slightly decreases the performance in segmentation. These effects were not significant for all models, but they do warrant further investigation. The increased performance in change detection in particular could be valuable, especially with the further addition of segmentation classes.

This study has given more insight into which approaches are useful for the analysis of drawn maps. It is not often that a collection of maps spanning such a large area and timespan is available for studies. As such, this provided a unique opportunity to examine both the capabilities for generalisation of CNNs, and the influence of multiple maps on the segmentation of single maps. Furthermore, this study has performed the first step to fully automate the change detection of drawn maps, which was previously either partially automated or done by hand. The promising results of this study encourage further development of convolutional neural networks for both semantic segmentation and change detection. The digitisation of historical maps is an arduous task of which this study is only a small part, but each automation step can contribute to the availability of a wealth of knowledge to the general public.

References

- [1] Hafez A. Affy. “Evaluation of change detection techniques for monitoring land-cover changes: A case study in new Burg El-Arab area”. In: *Alexandria Engineering Journal* 50.2 (2011), pp. 187–195. ISSN: 1110-0168. DOI: 10.1016/j.aej.2011.06.001.
- [2] Abdullah Almutairi and Timothy A. Warner. “Change Detection Accuracy and Image Properties: A Study Using Simulated Data”. In: *Remote Sensing* 2.6 (2010), pp. 1508–1529. ISSN: 2072-4292. DOI: 10.3390/rs2061508.
- [3] Anju Asokan, J. Anitha, Monica Ciobanu, Andrei Gabor, Antoanela Naaji, and D. Jude Hemanth. “Image Processing Techniques for Analysis of Satellite Images for Historical Maps Classification—An Overview”. In: *Applied Sciences* 10.12 (2020). ISSN: 2076-3417. DOI: 10.3390/app10124207.
- [4] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. *Delving Deeper into Convolutional Networks for Learning Video Representations*. 2016. arXiv: 1511.06432.
- [5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “SURF: Speeded Up Robust Features”. In: *Computer Vision – ECCV 2006*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33833-8.
- [6] Cong Cao, Suzana Dragičević, and Songnian Li. “Land-Use Change Detection with Convolutional Neural Network Methods”. In: *Environments* 6.2 (2019). ISSN: 2076-3298. DOI: 10.3390/environments6020025.
- [7] Cong Cao, Suzana Dragičević, and Songnian Li. “Short-Term Forecasting of Land Use Change Using Recurrent Neural Network Models”. In: *Sustainability* 11.19 (2019). ISSN: 2071-1050. DOI: 10.3390/su11195376.
- [8] Alejandro Coca Castro. “Pan-tropical modelling of land cover and land-use change trajectories for newly deforested areas”. PhD thesis. King’s College London, 2020.
- [9] R. Caye Daudt, B. Le Saux, and A. Boulch. “Fully Convolutional Siamese Networks for Change Detection”. In: *IEEE International Conference on Image Processing (ICIP)* (Athens, Greece). 2018.
- [10] Rodrigo Caye Daudt, Bertrand Le Saux, Alexandre Boulch, and Yann Gousseau. “Multi-task learning for large-scale semantic change detection”. In: *Computer Vision and Image Understanding* 187 (2019), p. 102783. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2019.07.003.
- [11] Joseph Chazalon, Edwin Carlinet, Yizi Chen, Julien Perret, Bertrand Duménieu, Clément Mallet, Thierry Géraud, Vincent Nguyen, Nam Nguyen, Josef Baloun, Ladislav Lenc, and Pavel Král. *ICDAR 2021 Competition on Historical Map Segmentation*. 2021. arXiv: 2105.13265.
- [12] Hongruixuan Chen, Chen Wu, Bo Du, Liangpei Zhang, and Le Wang. “Change Detection in Multisource VHR Images via Deep Siamese Convolutional Multiple-Layers Recurrent Neural Network”. In: *IEEE Transactions on Geoscience and Remote Sensing* 58.4 (2020), pp. 2848–2864. DOI: 10.1109/TGRS.2019.2956756.
- [13] Yizi Chen, Edwin Carlinet, Joseph Chazalon, Clément Mallet, Bertrand Duménieu, and Julien Perret. “Combining Deep Learning and Mathematical Morphology for Historical Map Segmentation”. In: *Discrete Geometry and Mathematical Morphology*. Ed. by Joakim Lindblad, Filip Malmberg, and Nataša Sladoje. Cham: Springer International Publishing, 2021, pp. 79–92. ISBN: 978-3-030-76657-3.
- [14] Zhanzhan Cheng, Fan Bai, Yunlu Xu, Gang Zheng, Shiliang Pu, and Shuigeng Zhou. “Focusing Attention: Towards Accurate Text Recognition in Natural Images”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.
- [15] Yao-Yi Chiang, Weiwei Duan, Stefan Leyk, Johannes Uhl, and Craig Knoblock. *Using Historical Maps in Scientific Studies: Applications, Challenges, and Best Practices*. 2020. ISBN: 978-3-319-66907-6. DOI: 10.1007/978-3-319-66908-3.
- [16] Yao-Yi Chiang, Stefan Leyk, and Craig A. Knoblock. “A Survey of Digital Map Processing Techniques”. In: *ACM Comput. Surv.* 47.1 (2014). ISSN: 0360-0300. DOI: 10.1145/2557423.

- [17] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *NIPS 2014 Workshop on Deep Learning*. 2014. arXiv: 1412.3555.
- [18] Thomas G. Dietterich. “Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms”. In: *Neural Computation* 10.7 (Oct. 1998), pp. 1895–1923. ISSN: 0899-7667. DOI: 10.1162/089976698300017197. eprint: <https://direct.mit.edu/neco/article-pdf/10/7/1895/814002/089976698300017197.pdf>.
- [19] M.R. Carter E.G. Gregorich. *Soil Sampling and Methods of Analysis*. 2nd ed. CRC Press, 2007. ISBN: 0849335868; 9780849335860. DOI: 10.1201/9781420005271.
- [20] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. “Efficient Graph-Based Image Segmentation”. In: *International Journal of Computer Vision* 59.2 (2004), pp. 167–181. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000022288.19776.77.
- [21] Richard Fuchs, Peter H. Verburg, Jan G.P.W. Clevers, and Martin Herold. “The potential of old maps and encyclopaedias for reconstructing historic European land cover/use change”. In: *Applied Geography* 59 (2015), pp. 43–55. ISSN: 0143-6228. DOI: 10.1016/j.apgeog.2015.02.013.
- [22] Arnau Garcia-Molsosa, Hector A. Orengo, Dan Lawrence, Graham Philip, Kristen Hopper, and Cameron A. Petrie. “Potential of deep learning segmentation for the extraction of archaeological features from historical map series”. In: *Archaeological Prospection* 28.2 (2021), pp. 187–199. DOI: 10.1002/arp.1807.
- [23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. “Deep Learning”. In: MIT Press, 2016. Chap. Deep Feedforward Networks. URL: <http://www.deeplearningbook.org>.
- [24] Nathan Hagen and Michael W. Kudenov. “Review of snapshot spectral imaging technologies”. In: *Optical Engineering* 52, 090901 (2013). DOI: 10.1117/1.OE.52.9.090901.
- [25] Antal Horváth, Charidimos Tsagkas, Simon Andermatt, Simon Pezold, Katrin Parmar, and Philippe Cattin. “Spinal Cord Gray Matter-White Matter Segmentation on Magnetic Resonance AMIRA Images with MD-GRU”. In: *Computational Methods and Clinical Applications for Spine Imaging*. Ed. by Guoyan Zheng, Daniel Belavy, Yunliang Cai, and Shuo Li. Cham: Springer International Publishing, 2019, pp. 3–14. ISBN: 978-3-030-13736-6.
- [26] Mohammad D. Hossain and Dongmei Chen. “Segmentation for Object-Based Image Analysis (OBIA): A review of algorithms and challenges from remote sensing perspective”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 150 (2019), pp. 115–134. ISSN: 0924-2716. DOI: 10.1016/j.isprsjprs.2019.02.009.
- [27] Bo Huang, Bei Zhao, and Yimeng Song. “Urban land-use mapping using a deep convolutional neural network with high spatial resolution multispectral remote sensing imagery”. In: *Remote Sensing of Environment* 214 (2018), pp. 73–86. ISSN: 0034-4257. DOI: 10.1016/j.rse.2018.04.050.
- [28] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. “Densely Connected Convolutional Networks”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2261–2269. DOI: 10.1109/CVPR.2017.243.
- [29] Masroor Hussain, Dongmei Chen, Angela Cheng, Hui Wei, and David Stanley. “Change detection from remotely sensed images: From pixel-based to object-based approaches”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 80 (2013), pp. 91–106. ISSN: 0924-2716. DOI: 10.1016/j.isprsjprs.2013.03.006.
- [30] Jungho Im and John Jensen. “A change detection model based on neighborhood correlation image analysis and decision tree classification”. In: *Remote Sensing of Environment* 99 (Nov. 2005), pp. 326–340. DOI: 10.1016/j.rse.2005.09.008.
- [31] Shruti Jadon. “A survey of loss functions for semantic segmentation”. In: *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)* (2020). DOI: 10.1109/cibcb48159.2020.9277638.
- [32] Kadaster. *200 Jaar Topografische Kaarten*. URL: <https://www.topotijdreis.nl/> (visited on 12/09/2021).

- [33] Michael Kampffmeyer, Arnt-Borre Salberg, and Robert Jenssen. “Semantic Segmentation of Small Objects and Modeling of Uncertainty in Urban Remote Sensing Images Using Deep Convolutional Neural Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2016.
- [34] Mustafa Kesikoglu, Ümit Atasever, and C. Özkan. “Unsupervised change detection in satellite images using Fuzzy C-Means Clustering and principal component analysis”. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XL-7/W2* (Oct. 2013), pp. 129–132. DOI: 10.5194/isprsarchives-XL-7-W2-129-2013.
- [35] Simeon Kostadinov. *Understanding GRU Networks*. 2017. URL: <https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be> (visited on 12/24/2021).
- [36] Stefan Leyk and Ruedi Boesch. “Extracting Composite Cartographic Area Features in Low-Quality Maps”. In: *Cartography and Geographic Information Science* 36.1 (2009), pp. 71–79. DOI: 10.1559/152304009787340115.
- [37] Fei-Fei Li, Justin Johnson, and Serena Yeung. *Detection and Segmentation*. 2017.
- [38] Shuang Li, Zhongqiu Sun, Yafei Wang, and Yuxia Wang. “Understanding Urban Growth in Beijing-Tianjin-Hebei Region over the Past 100 Years Using Old Maps and Landsat Data”. In: *Remote Sensing* 13.16 (2021). ISSN: 2072-4292. DOI: 10.3390/rs13163264.
- [39] Zekun Li, Yao-Yi Chiang, Sasan Tavakkol, Basel Shbita, Johannes H. Uhl, Stefan Leyk, and Craig A. Knoblock. “An Automatic Approach for Generating Rich, Linked Geo-Metadata from Historical Map Images”. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’20. Virtual Event, CA, USA: Association for Computing Machinery, 2020, pp. 3290–3298. ISBN: 9781450379984. DOI: 10.1145/3394486.3403381.
- [40] Kyungsun Lim, Dongkwon Jin, and Chang-Su Kim. “Change Detection in High Resolution Satellite Images Using an Ensemble of Convolutional Neural Networks”. In: *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. 2018, pp. 509–515. DOI: 10.23919/APSIPA.2018.8659603.
- [41] Haowen Lin and Yao-Yi Chiang. “SRC: Automatic Extraction of Phrase-level Map Labels from Historical Maps”. In: *SIGSPATIAL*. Vol. 9. 2017, pp. 14–15.
- [42] Dan Liu, Elizabeth Toman, Zane Fuller, Gang Chen, Alexis Londo, Xuesong Zhang, and Kaiguang Zhao. “Integration of historical map and aerial imagery to characterize long-term land-use change and landscape dynamics: An object-based analysis via Random Forests”. In: *Ecological Indicators* 95 (2018), pp. 595–605. ISSN: 1470-160X. DOI: 10.1016/j.ecolind.2018.08.004.
- [43] Desheng Liu and Fan Xia. “Assessing object-based classification: advantages and limitations”. In: *Remote Sensing Letters* 1.4 (2010), pp. 187–194. DOI: 10.1080/01431161003743173.
- [44] Fang Liu, Licheng Jiao, Xu Tang, Shuyuan Yang, Wenping Ma, and Biao Hou. “Local Restricted Convolutional Neural Network for Change Detection in Polarimetric SAR Images”. In: *IEEE Transactions on Neural Networks and Learning Systems* 30.3 (2019), pp. 818–833. DOI: 10.1109/TNNLS.2018.2847309.
- [45] Vladimir Loginov. “Why You Should Try the Real Data for the Scene Text Recognition”. In: *CoRR* abs/2107.13938 (2021). arXiv: 2107.13938. URL: <https://arxiv.org/abs/2107.13938>.
- [46] David Lowe. “Object Recognition from Local Scale-Invariant Features”. In: *Proceedings of the IEEE International Conference on Computer Vision 2* (2001).
- [47] Emile Marnette, Charles Pijls, Frank Volkering, Tobias Praamstra, Bert Scheffer, Marian Langevoort, Dirk Paulus, and Ronnie Berg. “In Situ Soil and Groundwater Remediation: Theory and Practice”. In: TAUW, 2018. Chap. 1.
- [48] Emile Marnette, Charles Pijls, Frank Volkering, Tobias Praamstra, Bert Scheffer, Marian Langevoort, Dirk Paulus, and Ronnie Berg. “In Situ Soil and Groundwater Remediation: Theory and Practice”. In: TAUW, 2018. Chap. 3.

- [49] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org, 2015. URL: <https://www.tensorflow.org/>.
- [50] Shervin Minaee, Yuri Y. Boykov, Fatih Porikli, Antonio J Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. “Image Segmentation Using Deep Learning: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), pp. 1–1. DOI: 10.1109/TPAMI.2021.3059968.
- [51] Divyanshu Mishra. *Transposed Convolution Demystified*. 2020. URL: <https://towardsdatascience.com/transposed-convolution-demystified-84ca81b4baba> (visited on 12/23/2021).
- [52] Lichao Mou, Lorenzo Bruzzone, and Xiao Xiang Zhu. “Learning Spectral-Spatial-Temporal Features via a Recurrent Convolutional Neural Network for Change Detection in Multi-spectral Imagery”. In: *IEEE Transactions on Geoscience and Remote Sensing* 57.2 (2019), pp. 924–935. DOI: 10.1109/TGRS.2018.2863224.
- [53] Vinod Nair and Geoffrey Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: *Proceedings of the 27th International Conference on Machine Learning*. Vol. 27. Haifa, Israel, 2010, pp. 807–814. ISBN: 9781605589077.
- [54] NEN. *NEN 5740:2009+A1:2016*. 2016. URL: www.nen.nl (visited on 12/16/2021).
- [55] Peer Neubert and Peter Protzel. “Compact Watershed and Preemptive SLIC: On Improving Trade-offs of Superpixel Segmentation Algorithms”. In: *2014 22nd International Conference on Pattern Recognition*. 2014, pp. 996–1001. DOI: 10.1109/ICPR.2014.181.
- [56] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. “Learning Deconvolution Network for Semantic Segmentation”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1520–1528. DOI: 10.1109/ICCV.2015.178.
- [57] Christopher Olah. *Understanding LSTM Networks*. 2015. URL: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/> (visited on 12/24/2021).
- [58] ReGister Historisch Onderzoeksbureau. *Toelichting op de introductie van de nieuwe UBI (UBI 2.0)*. 2004.
- [59] ReGister Historisch Onderzoeksbureau and Arcadis Heidemij Advies. *Uniforme Bron Indeling: potentieel bodemvervuilende activiteiten*. 2001.
- [60] Erting Pan, Xiaoguang Mei, Quande Wang, Yong Ma, and Jiayi Ma. “Spectral-spatial classification for hyperspectral image based on a single GRU”. In: *Neurocomputing* 387 (2020), pp. 150–160. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2020.01.029.
- [61] 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.
- [62] Pietro Picuno, Giuseppe Cillis, and Dina Statuto. “Investigating the time evolution of a rural landscape: How historical maps may provide environmental information when processed using a GIS”. In: *Ecological Engineering* 139 (2019), p. 105580. ISSN: 0925-8574. DOI: 10.1016/j.ecoleng.2019.08.010.
- [63] R.J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. “Image change detection algorithms: a systematic survey”. In: *IEEE Transactions on Image Processing* 14.3 (2005), pp. 294–307. DOI: 10.1109/TIP.2004.838698.
- [64] Rijkswaterstaat. *Typen bodeminformatie*. URL: <https://www.bodemplus.nl/onderwerpen/bodem-ondergrond/informatiebeheer/typen-bodeminformatie/> (visited on 12/14/2021).
- [65] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (2015), pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

- [66] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2017. arXiv: 1609.04747.
- [67] Mahmoud Saeedimoghaddam and T. F. Stepinski. “Automatic extraction of road intersection points from USGS historical map series using deep convolutional neural networks”. In: *International Journal of Geographical Information Science* 34.5 (2020), pp. 947–968. DOI: 10.1080/13658816.2019.1696968.
- [68] Robin M. Schmidt. “Recurrent Neural Networks (RNNs): A gentle Introduction and Overview”. In: *CoRR* abs/1912.05911 (2019). arXiv: 1912.05911.
- [69] Steven E. Sesnie, Paul E. Gessler, Bryan Finegan, and Sirpa Thessler. “Integrating Landsat TM and SRTM-DEM derived variables with decision trees for habitat classification and change detection in complex neotropical environments”. In: *Remote Sensing of Environment* 112.5 (2008). Earth Observations for Terrestrial Biodiversity and Ecosystems Special Issue, pp. 2145–2159. ISSN: 0034-4257. DOI: 10.1016/j.rse.2007.08.025.
- [70] Seyd Teymoor Seydi, Mahdi Hasanlou, and Meisam Amani. “A New End-to-End Multi-Dimensional CNN Framework for Land Cover/Land Use Change Detection in Multi-Source Remote Sensing Datasets”. In: *Remote Sensing* 12.12 (2020). ISSN: 2072-4292. DOI: 10.3390/rs12122010. URL: <https://www.mdpi.com/2072-4292/12/12/2010>.
- [71] Tenzing W. Shaw and Peter Bajcsy. “Automation of digital historical map analyses”. In: *Computer Vision and Image Analysis of Art II*. Ed. by David G. Stork, Jim Coddington, and Anna Bentkowska-Kafel. Vol. 7869. International Society for Optics and Photonics. SPIE, 2011, pp. 75–84. DOI: 10.1117/12.872875.
- [72] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. “Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting”. In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc., 2015. ISBN: 978-1-55860-274-8.
- [73] Mennatullah Siam, Sepehr Valipour, Martin Jagersand, and Nilanjan Ray. “Convolutional Gated Recurrent Networks for Video Segmentation”. In: *2017 IEEE International Conference on Image Processing (ICIP)* (2017), pp. 3090–3094. DOI: 10.48550/ARXIV.1611.05435.
- [74] Md. Sadman Siraj, Omar Shahid, and Md Atiqur Rahman Ahad. “Cooking Activity Recognition with Varying Sampling Rates Using Deep Convolutional GRU Framework”. In: *Human Activity Recognition Challenge*. Ed. by Md Atiqur Rahman Ahad, Paula Lago, and Sozo Inoue. Singapore: Springer Singapore, 2021, pp. 115–126. ISBN: 978-981-15-8269-1. DOI: 10.1007/978-981-15-8269-1_10.
- [75] A. Smith. “Image segmentation scale parameter optimization and land cover classification using the Random Forest algorithm”. In: *Journal of Spatial Science* 55.1 (2010), pp. 69–79. DOI: 10.1080/14498596.2010.487851.
- [76] *Soil Sampling for Environmental Contaminants*. TECDOC Series 1415. Vienna: International Atomic Energy Agency, 2004. ISBN: 92-0-111504-0.
- [77] Charlotte E Stancioff, Robert G Pontius Jr, Scott Mandry, and Elizabeth Jones. “Historic Forest Change: New Approaches to Land Use Land Cover”. In: *CAA2014: 21st Century Archaeology: Concepts, methods and tools. Proceedings of the 42nd Annual Conference on Computer Applications and Quantitative Methods in Archaeology*. Archaeopress Publishing Ltd. 2015.
- [78] Ziheng Sun, Liping Di, and Hui Fang. “Using long short-term memory recurrent neural network in land cover classification on Landsat and Cropland data layer time series”. In: *International Journal of Remote Sensing* 40 (2018), pp. 1–22. DOI: 10.1080/01431161.2018.1516313.
- [79] Shirisa Timilsina, Jagannath Aryal, and Jamie B. Kirkpatrick. “Mapping Urban Tree Cover Changes Using Object-Based Convolution Neural Network (OB-CNN)”. In: *Remote Sensing* 12.18 (2020). ISSN: 2072-4292. DOI: 10.3390/rs12183017.
- [80] Sik-Ho Tsang. *Review: DilatedNet — Dilated Convolution (Semantic Segmentation)*. 2018. URL: <https://towardsdatascience.com/review-dilated-convolution-semantic-segmentation-9d5a5bd768f5> (visited on 12/24/2021).

- [81] Johannes H. Uhl, Stefan Leyk, Yao-Yi Chiang, Weiwei Duan, and Craig A. Knoblock. “Extracting human settlement footprint from historical topographic map series using context-based machine learning”. In: *8th International Conference of Pattern Recognition Systems (ICPRS 2017)*. 2017, pp. 1–6. DOI: 10.1049/cp.2017.0144.
- [82] Johannes H. Uhl, Stefan Leyk, Yao-Yi Chiang, Weiwei Duan, and Craig A. Knoblock. “Map Archive Mining: Visual-Analytical Approaches to Explore Large Historical Map Collections”. In: *ISPRS International Journal of Geo-Information* 7.4 (2018). ISSN: 2220-9964. DOI: 10.3390/ijgi7040148.
- [83] Andrea Vedaldi and Stefano Soatto. “Quick Shift and Kernel Methods for Mode Seeking”. In: *Computer Vision – ECCV 2008*. Ed. by David Forsyth, Philip Torr, and Andrew Zisserman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 705–718. ISBN: 978-3-540-88693-8.
- [84] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. “scikit-image: image processing in Python”. In: *PeerJ* 2 (June 2014), e453. ISSN: 2167-8359. DOI: 10.7717/peerj.453.
- [85] Shuhang Wang, Szu-Yeu Hu, Eugene Cheah, Xiaohong Wang, Jingchao Wang, Lei Chen, Masoud Baikpour, Arinc Ozturk, Qian Li, Shinn-Huey Chou, Constance D. Lehman, Viksit Kumar, and Anthony Samir. *U-Net Using Stacked Dilated Convolutions for Medical Image Segmentation*. 2020. DOI: 10.48550/ARXIV.2004.03466.
- [86] Jerod Weinman. “Geographic and Style Models for Historical Map Alignment and Toponym Recognition”. In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 01. 2017, pp. 957–964. DOI: 10.1109/ICDAR.2017.160.
- [87] Jerod Weinman, Ziwen Chen, Ben Gafford, Nathan Gifford, Abyaya Lamsal, and Liam Niehus-Staab. “Deep Neural Networks for Text Detection and Recognition in Historical Maps”. In: *Proc. IAPR International Conference on Document Analysis and Recognition*. Sydney, Australia, 2019. DOI: 10.1109/ICDAR.2019.00149.
- [88] Tzu-Tsung Wong and Nai-Yu Yang. “Dependency Analysis of Accuracy Estimates in k-Fold Cross Validation”. In: *IEEE Transactions on Knowledge and Data Engineering* 29.11 (2017), pp. 2417–2427. DOI: 10.1109/TKDE.2017.2740926.
- [89] Hongtao Xie, Shancheng Fang, Zheng-Jun Zha, Yating Yang, Yan Li, and Yongdong Zhang. “Convolutional Attention Networks for Scene Text Recognition”. In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 15.1 (2019). ISSN: 1551-6857. DOI: 10.1145/3231737.
- [90] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. “Convolutional neural networks: an overview and application in radiology”. In: *Insights into Imaging* 9.4 (2018), pp. 611–629. ISSN: 1869-4101. DOI: 10.1007/s13244-018-0639-9.
- [91] Fisher Yu and Vladlen Koltun. “Multi-Scale Context Aggregation by Dilated Convolutions”. In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, Conference Track Proceedings*. 2016. URL: <http://arxiv.org/abs/1511.07122>.
- [92] P. Zatelli, S. Gobbi, C. Tattoni, N. La Porta, and M. Ciolli. “Object-based image analysis for historic maps classification”. In: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* 42 (2019), pp. 247–254.
- [93] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. *Pyramid Scene Parsing Network*. 2017. DOI: 10.1109/CVPR.2017.660.
- [94] Zhedong Zheng, Liang Zheng, and Yi Yang. “A Discriminatively Learned CNN Embedding for Person Reidentification”. In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 14.1 (2017). ISSN: 1551-6857. DOI: 10.1145/3159171.

A Tables

Model 1	Model 2	T	p-corr	Hedge's g
Forest _{block}	NN ₁	-25.9	< 0.001	-11.0
	NN ₂	-32.5	< 0.001	-10.0
	RNN ₂	-16.9	< 0.001	-9.9
	RNN _{all}	-28.9	< 0.001	-10.3
	NN ₂ (both)	-27.1	< 0.001	-10.3
	RNN ₂ (both)	-21.9	< 0.001	-8.5
	RNN _{all} (both)	-12.0	< 0.001	-6.5
Forest _{object}	NN ₁	-55.7	< 0.001	-21.2
	NN ₂	-96.8	< 0.001	-19.0
	RNN ₂	-39.2	< 0.001	-18.9
	RNN _{all}	-72.0	< 0.001	-19.6
	NN ₂ (both)	-62.2	< 0.001	-20.8
	RNN ₂ (both)	-68.1	< 0.001	-17.0
	RNN _{all} (both)	-26.3	< 0.001	-13.9

Table 11: One-tailed T-test results for the segmentation task of the baseline models against the neural network models. All tests have 4 degrees of freedom.

Model 1	Model 2	T	p-corr	Hedge's g
Forest _{block}	NN ₂	-11.9	< 0.001	-5.1
	RNN ₂	-13.9	< 0.001	-5.6
	NN ₂ (both)	-44.8	< 0.001	-9.3
	RNN ₂ (both)	-22.3	< 0.001	-7.1
	RNN _{all} (both)	-11.2	< 0.001	-6.8
Forest _{object}	NN ₂	-31.0	< 0.001	-9.7
	RNN ₂	-30.2	< 0.001	-10.3
	RNN _{all}	-13.7	< 0.001	-5.1
	NN ₂ (both)	-55.4	< 0.001	-14.4
	RNN ₂ (both)	-30.0	< 0.001	-12.1
	RNN _{all}	-21.0	< 0.001	-12.0

Table 12: One-tailed T-test results for the change detection task of the baseline models against the neural network models. All tests have 4 degrees of freedom.

B Semantic Segmentation

The segmented map is always the second map; the first is the map from one decade after it, to which only the models with multiple maps as input have access. The classes are coloured as follows: red for buildings, green for orchards, and blue for water.

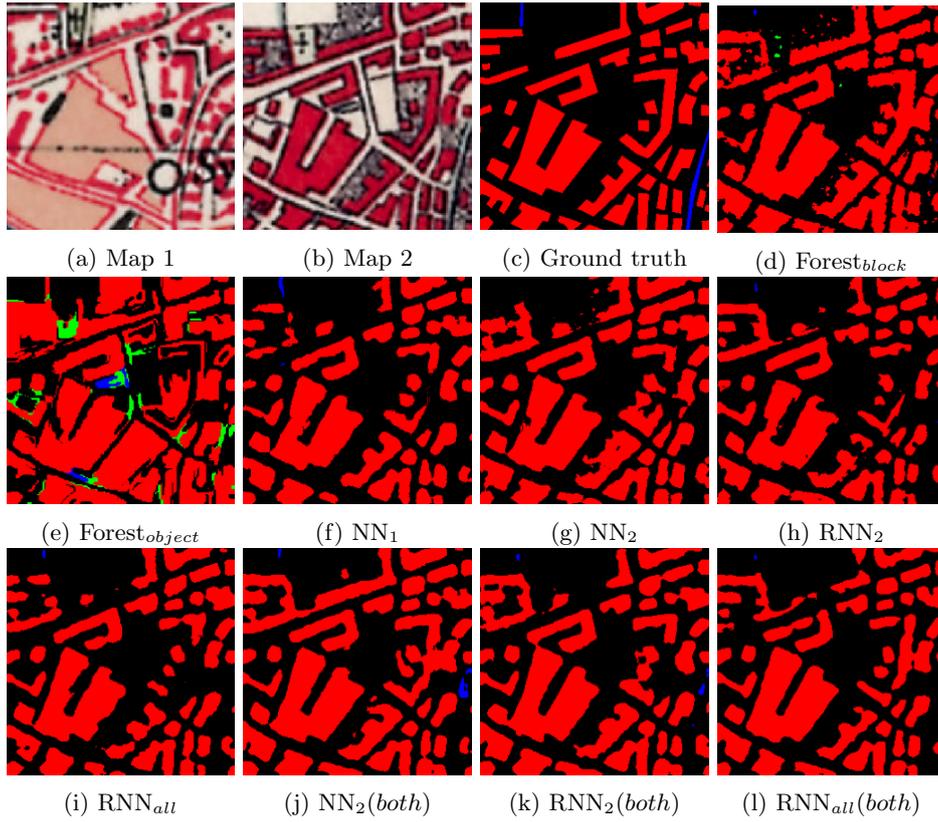


Figure 20: Note the long and narrow residential blocks. Only *Forest_block* and *NN₂(both)* identify them correctly; the others likely identified them as roads because they are also coloured red in this time period.

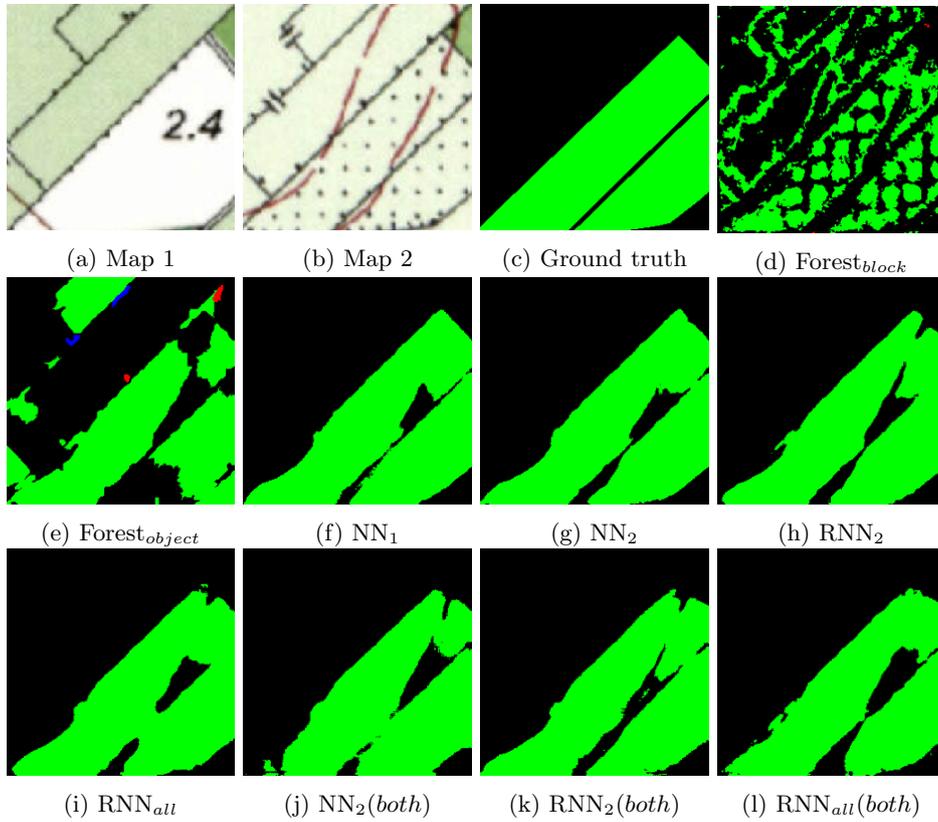


Figure 21: Note how the contour lines interrupt the segmentations of all models.

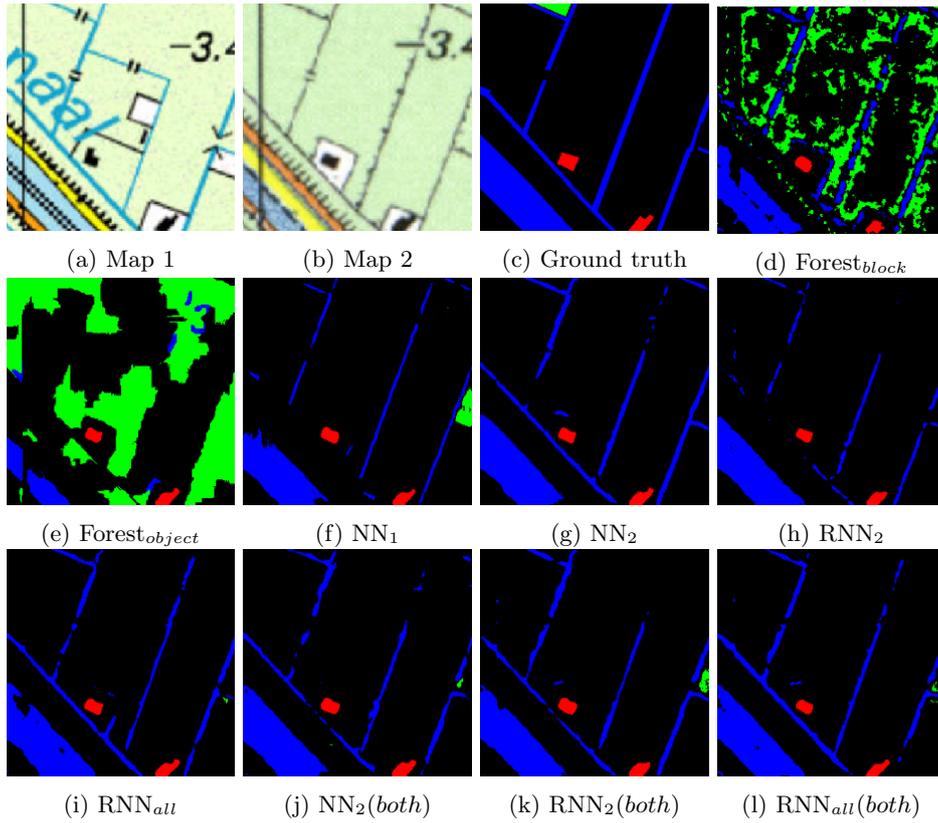


Figure 22: Note the ditch at the bottom, which is missed by some models. Also note that most models correctly include the yellow province border as water.

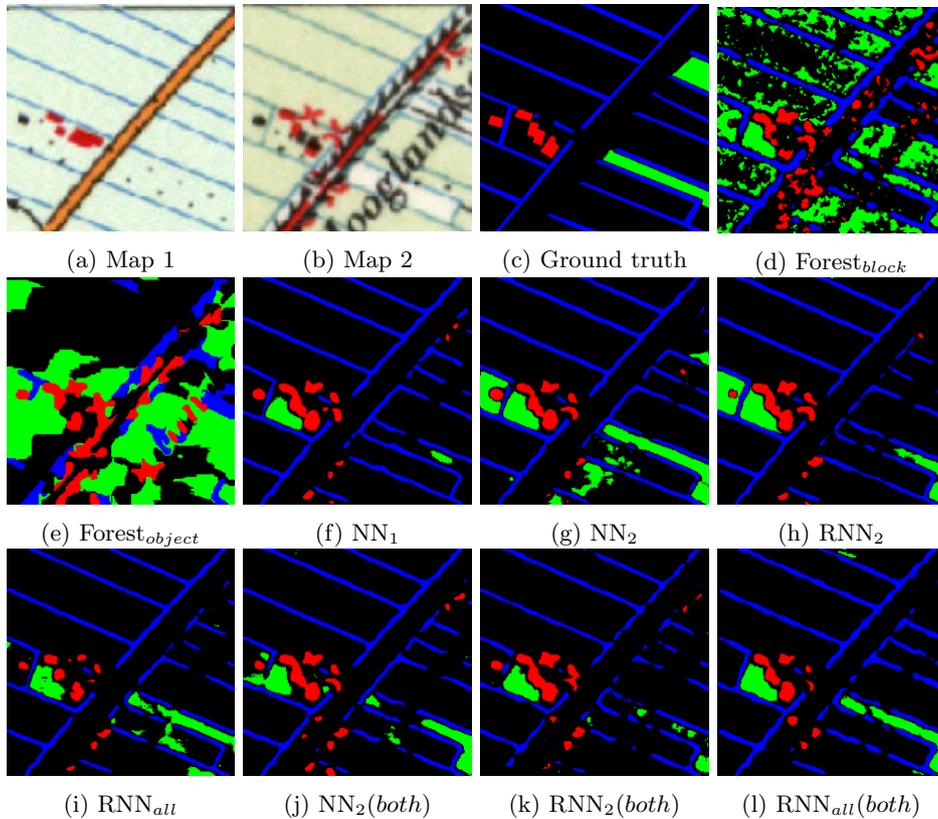


Figure 23: The letters interrupt the ditches of all models, even though the additional map has uninterrupted ditches. The orchard in the bottom right is identified mostly by models with both maps as input, presumably because the other map also has an orchard there.

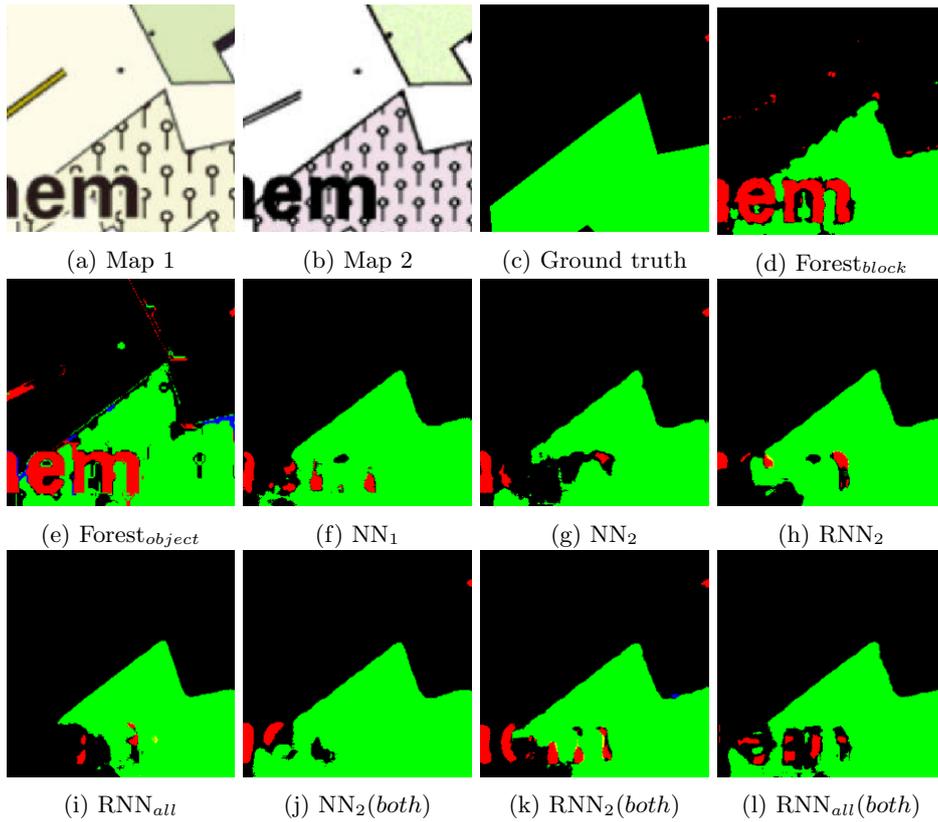


Figure 24: Big black letters are sometimes confused with buildings.

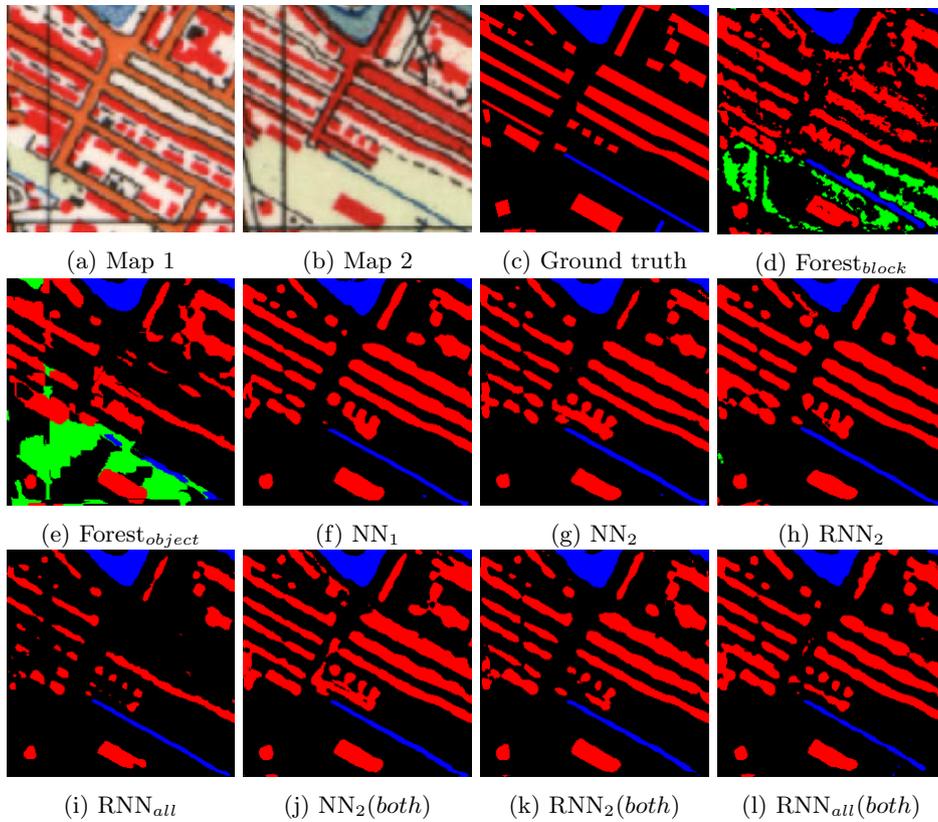


Figure 25: The buildings here are very similar to the roads, but most models separate them well. Only the road in the middle underneath four individual buildings is seen as a building by multiple models.

C Change Detection

The changes between the two given maps are detected, with the first map being the more recent decade. White denotes changed areas, black not changed.

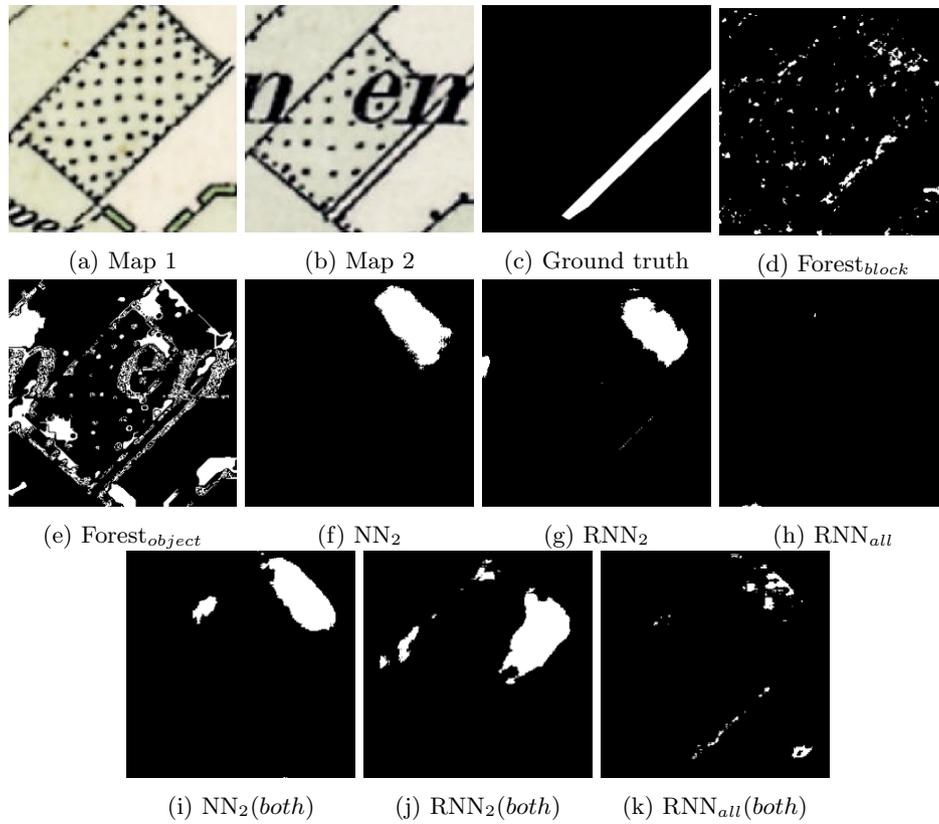


Figure 26: Note that multiple models identify the upper part of the orchard as changed, because it was longer in the newer map. This change is not in the ground truth because of the imprecise georeferencing of the older map, but this is not visible to the models.

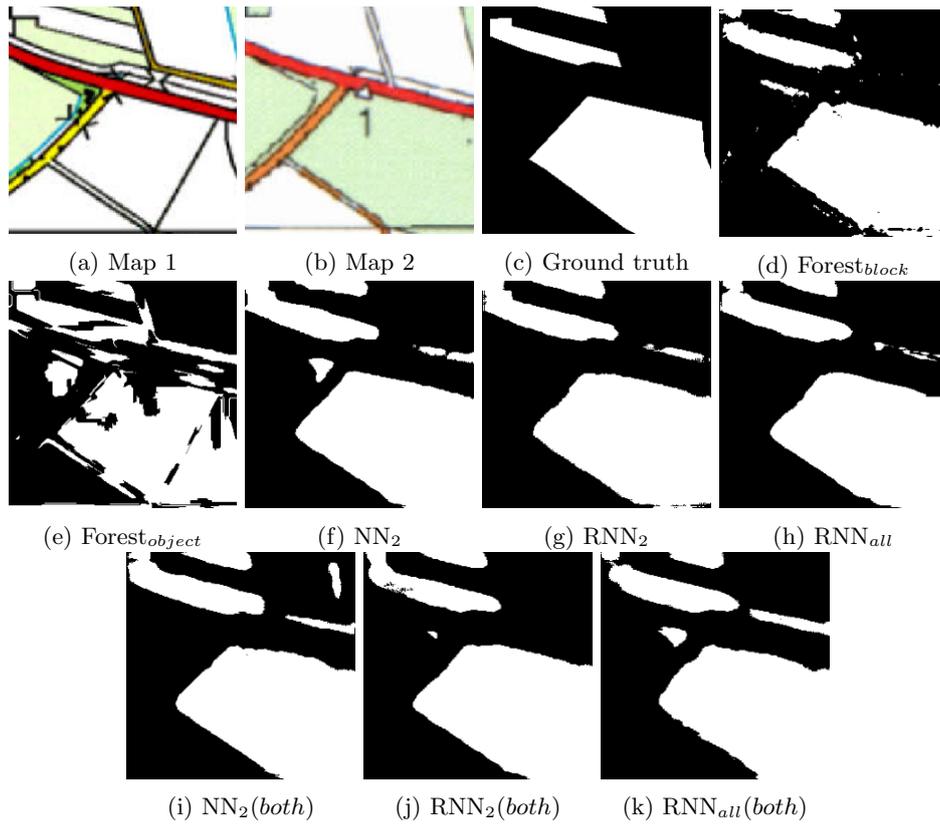


Figure 27: Large patches of single colours are easily identified as changed. Note the yellow road at the middle right; while it is missed in the ground truth, some models pick up on it.

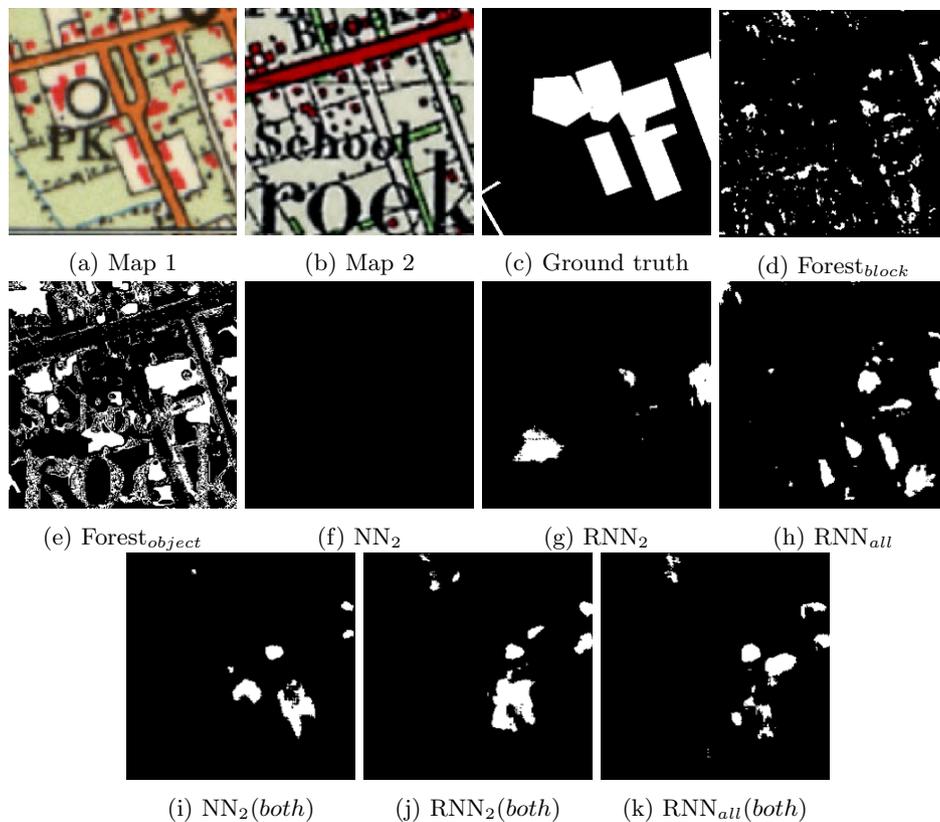


Figure 28: Hardly any change is found here by the models; the georeferencing and the graphical style differ too much between the two maps.

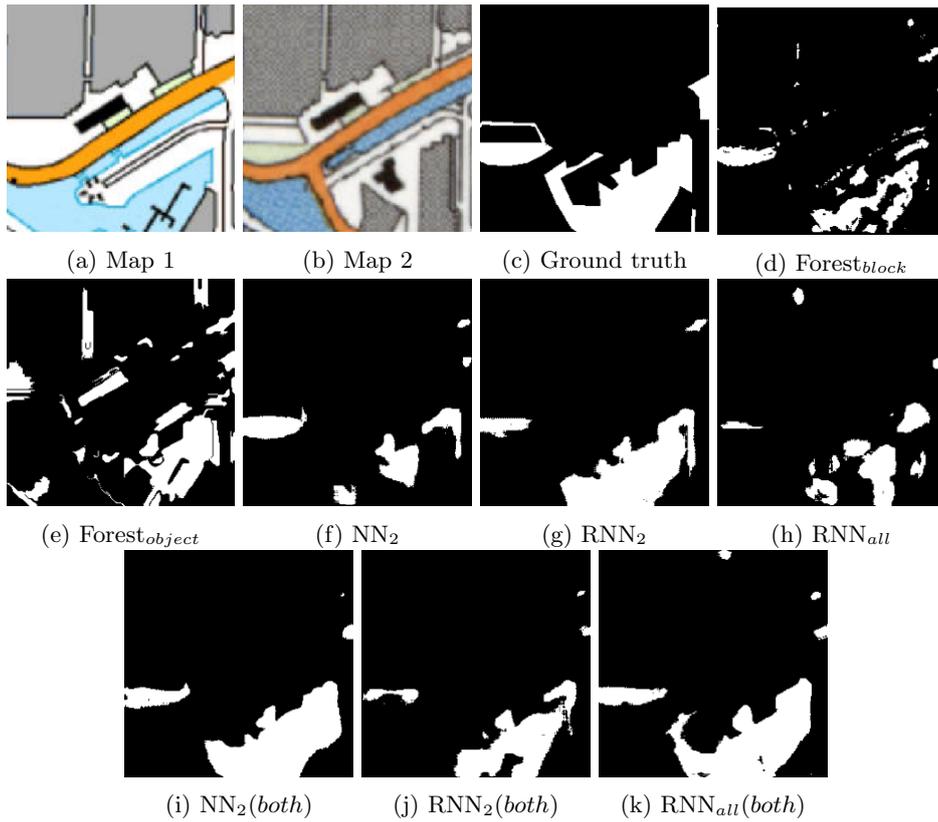


Figure 29: Note the difference between the models trained on both tasks and those trained on CD only; they identify the new harbour more completely.

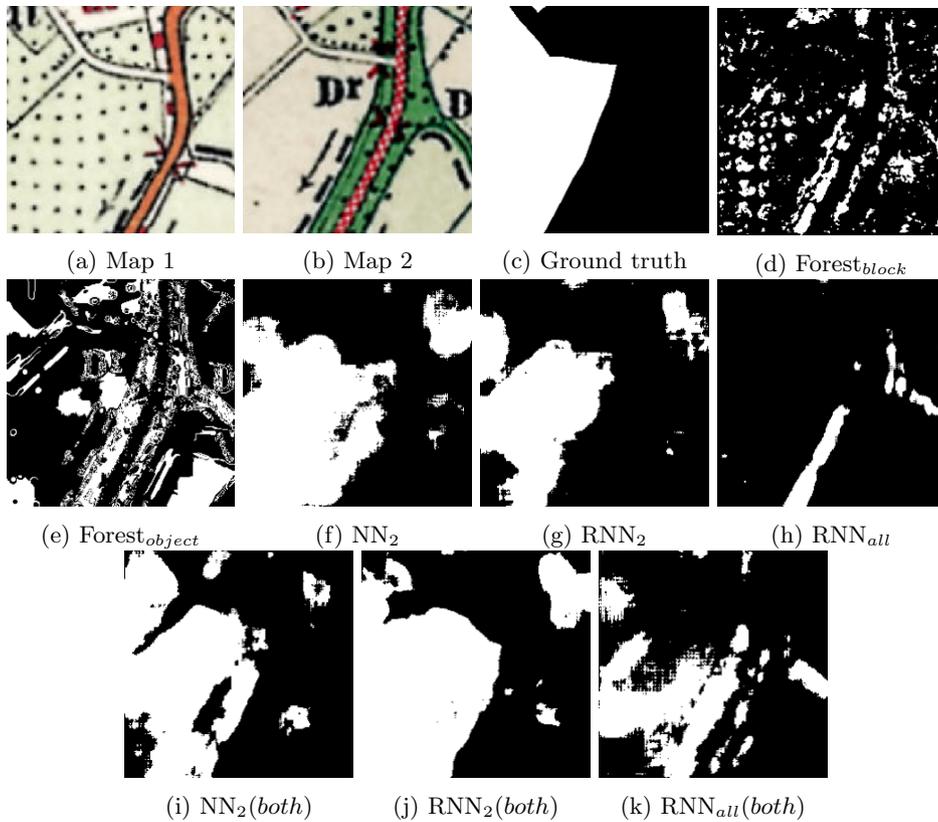


Figure 30: The ground truth misses the orchard in the upper right corner, but is (partially) spotted by several models. The orchard on the left is correctly found by most.

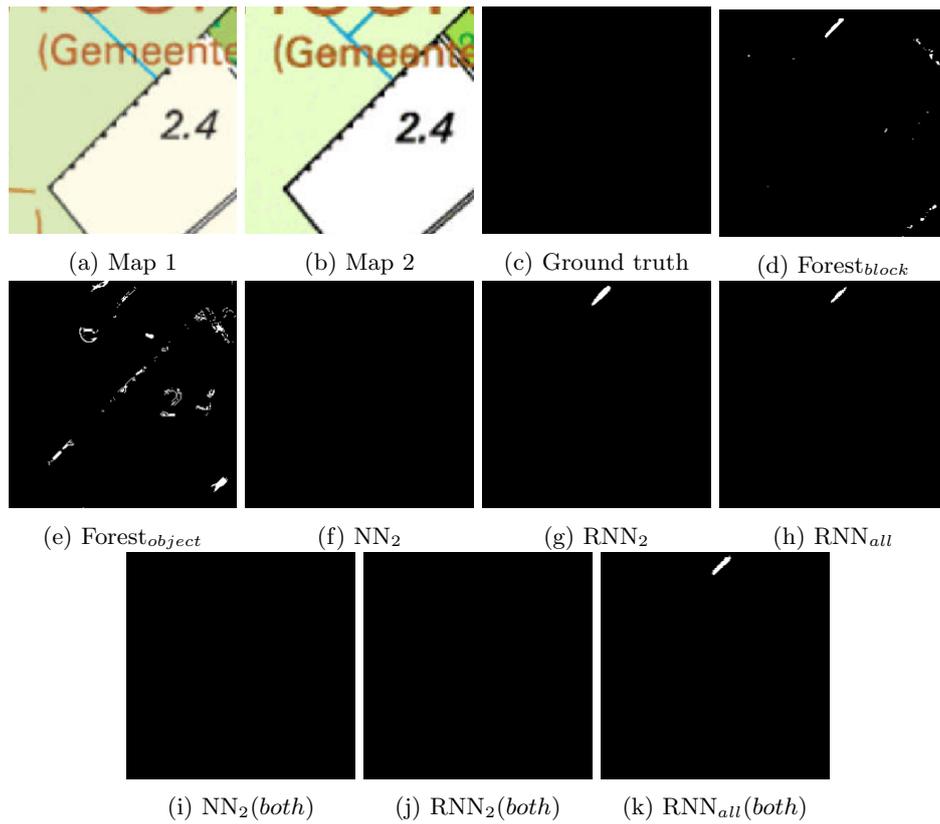


Figure 31: The ground truth misses the small ditch in the upper middle. Several models do spot it correctly. Note that the text is on almost exactly the same spot; this is often the case

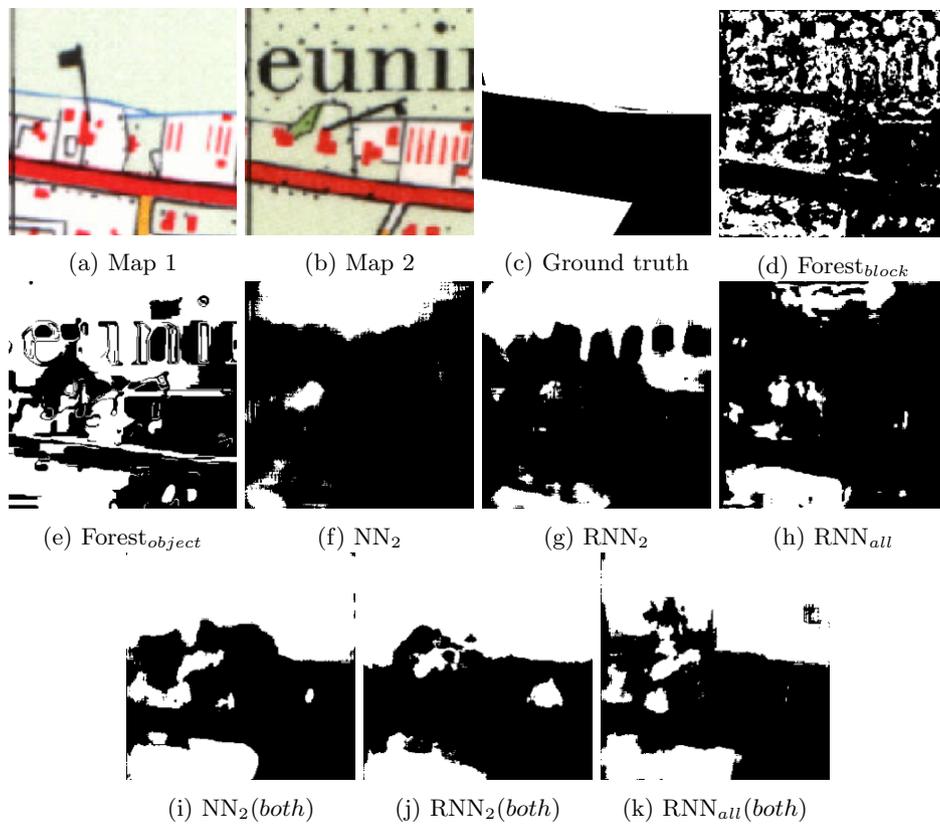


Figure 32: The letters hinder the change detection of most models, although those trained on both tasks are less affected.

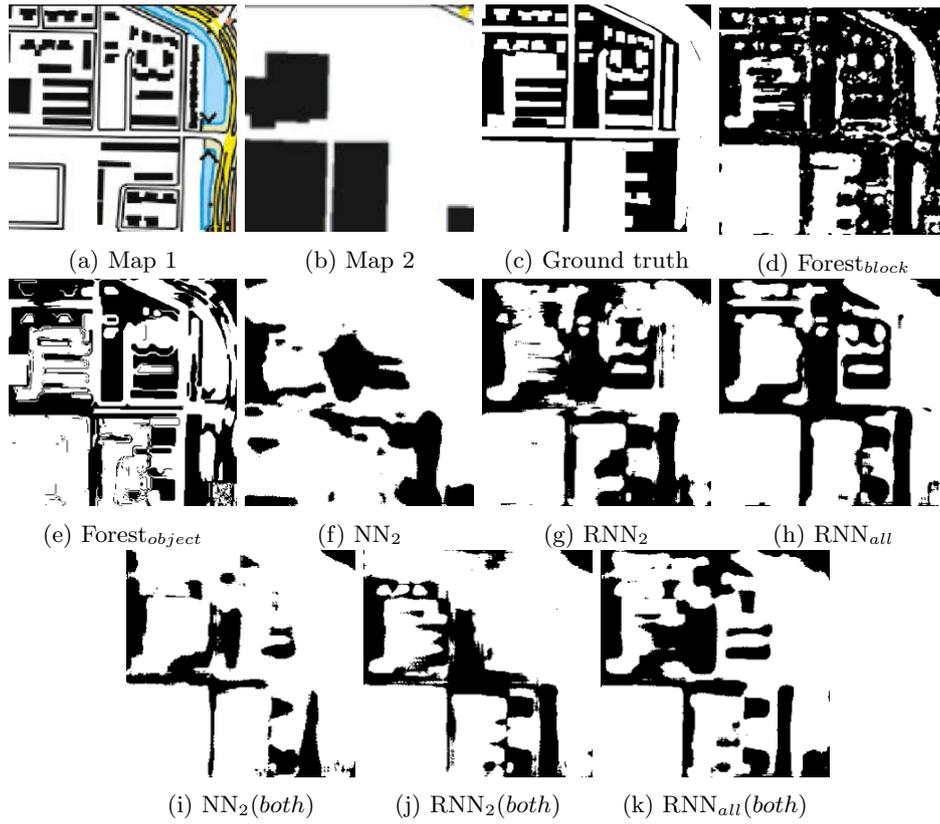


Figure 33: Large changed areas are not marked in as much detail as smaller changes.

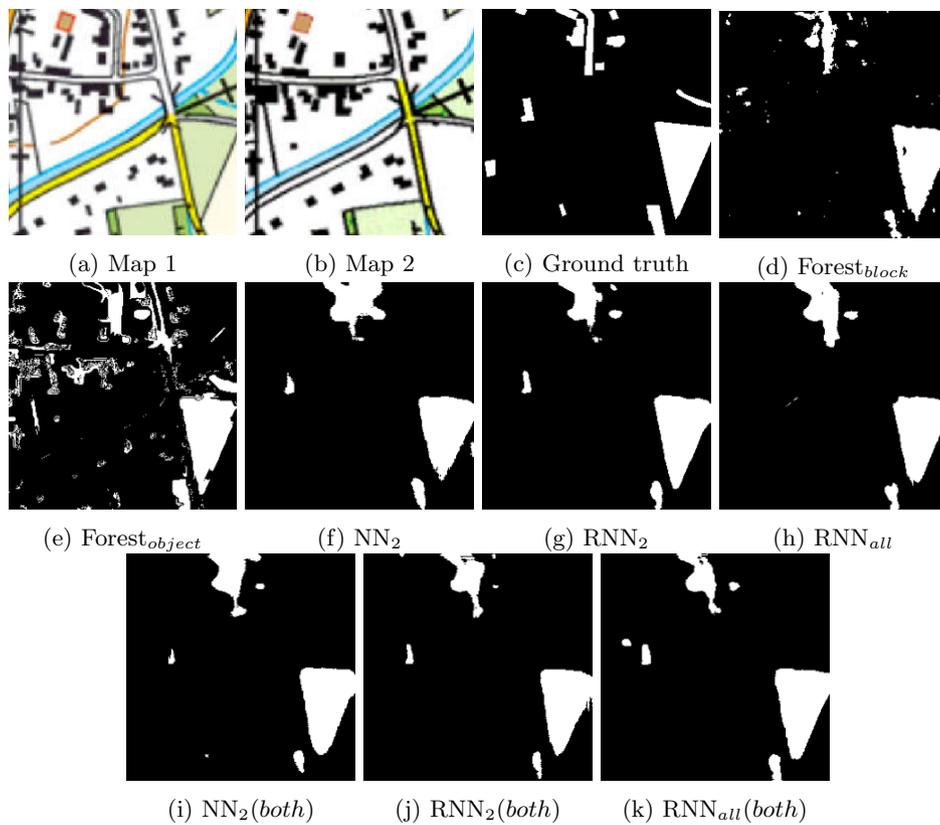


Figure 34: Small changes such as individual buildings are more easily missed.