Student thesis series INES nr 524

An artificial intelligence method for text placement evaluation in maps

Lai Wei

2020
Department of
Physical Geography and Ecosystem Science
Lund University
Sölvegatan 12
S-223 62 Lund
Sweden

Lai Wei (2020).

An artificial intelligence method for text placement evaluation in maps

Master degree thesis, 30 credits in *Geomatics*
Department of Physical Geography and Ecosystem Science, Lund University

Level: Master of Science (MSc)

Course duration: *January* 2020 until *June* 2020

Disclaimer

This document describes work undertaken as part of a program of study at the University of Lund. All views and opinions expressed herein remain the sole responsibility of the author, and do not necessarily represent those of the institute.

An artificial intelligence method for text placement evaluation in maps

---

Lai Wei

Master thesis, 30 credits, in *Geomatics*

Supervisor:
Lars Harrie
Department of Physical Geography and Ecosystem Science
Lund University

Exam Committee:
Micael Runnström and Per-Ola Olsson
Department of Physical Geography and Ecosystem Science
Lund University

Abstract

A map is a combinatorial presentation of map objects and labels providing users with sufficient and useful information. Text placement is the most time-consuming and cumbersome work in a map production process, and many methods have been proposed to automate this process. However, text placement evaluation lacks exploration, especially when artificial intelligence (AI) methods are experiencing prosperous development.

To bridge the gap, this master thesis proposed an AI method for a text placement evaluation. Firstly, London street maps were simplified and rasterized based on cartographic rules formulated by experienced cartographers. Secondly, a text placement quality measurement schema was designed. Thirdly, datasets were prepared by manually classified rasterized maps into different quality levels. Fourthly, *GoogLeNet*, a widely accepted convolutional neural network, was trained, and its performance was validated.

The results showed the *GoogLeNet* model's poor training accuracy and validation accuracy in the training process and bad performance in evaluating text placement. All the test images were classified as having bad text placement. These pieces of evidence indicated that the trained *GoogLeNet* model was not reliable and qualified for evaluating text placement at this stage. The negative results might be caused by the capacity issue, dataset issues and model issues. These issues were discussed in the latter part of the report. Finally, some limitations and perspectives of using AI for text evaluation were listed in the end for further enlightenment.

Keywords: Text placement evaluation; Artificial intelligence; GoogLeNet.

Acknowledgement

Foremost, I wish to express my sincere gratitude to my supervisor Lars Harrie for his guidance, support and patience. His constructive ideas and profound knowledge help me a lot to deal with this master thesis. I would also thank Micael Runnström and Per-Ola Olsson for their professional comments and Pavlos for his suggestion on writings.

Great thanks to Andreas Oxenstierna and Åsa Nilsson from T-Kartor for their technical help and data. It will be much more challenging to accomplish this master thesis without their support and advice.

Special thanks to my friend Jasmine Lu for lending me her laptop to continue my experiment.

Moreover, I would like to acknowledge Jianjun Pan for his GoogLeNet codes on GitHub.

Finally, I would express my profound gratitude to my family and my friends. Their patient listening and gentle comforting helped me get through the anxiety and panic during this surreal period. It was their understanding and support that got me where I am today.

Table of Content

1 Introduction

1.1 Background

A map is a presentation of real-world objects with pictorial or literal labels to provide users with sufficient and useful information (Haunert & Wolff, 2017). Labels are annotations associated with the corresponding objects. There is a complicated relationship between labels and map features, and such relationship restricts the label placement in the map. These relationships are defined as cartographic rules, which are sensitive to many factors, such as user preference, type of map objects (Freeman, 2005), and scales (van Dijk et al., 2002). Labels should follow specific cartographic rules for better readability. For example, labels should avoid obscuring each other and other map objects. Not only cartographic rules but also aesthetic requirements should be satisfied. Quality of a map, however, highly depends on cartographers' experiences, especially if a generalization is needed. Such complexity and ambiguity make text placement become the most time-consuming and skilled work in a map production process, which takes nearly 50% of the work (Yoeli, 1972). Therefore, how to automate text placement has been drawing much attention in cartography area (Yoeli, 1972; Hirsch, 1982; Mower, 1993; van Kreveld, Strijk, & Wolff, 1999; van Dijk, Steven, & Dirk, 2004; Revell, Regnauld, & Bulbrooke, 2011).

Many previous studies have focused on text placement related issues (Imhof, 1975; Dent, 1996). Text label placement was initially manually accomplished by highly qualified experts; however, it takes considerable time and costs to achieve high-quality results. Geographic information system and remote sensing technology development enrich technologies of observation and storage of geographic data. The development brings an increasing need for various types of maps, such as a topographic map, national map, and transportation map. Consequently, many automated labelling methods are developed for efficiency and advancement in map-making (Yoeli, 1972; Jones, 1989; Zoraster, 1997; van Kreveld, Strijk, & Wolff, 1999; Freeman, 2005). Sequential placement method (Jones, 1989), combinatorial optimization method (Zoraster, 1997) and slider model (van Kreveld, Strijk, & Wolff, 1999) were widely discussed and used in related literature. Based on these algorithms, some automated text placement tools or systems were produced. Cook and Jones (1990) proposed a

prolog rule-based automated text placement system. Freeman (2004) designed a new text placement model called Label-EZ, which could place texts according to users' preference.

However, cartographic rules change with purpose, target area and map objects, which leads to the problem that automated text placement approaches are not suitable for all scenarios. Therefore, map evaluation is critical in map production, especially for commercial mapping companies. According to evaluation results, subsequent manual modification is performed to complete text placement. Besides, numerous automated text labelling methods have been developed, then how to compare their performance is important. However, rather few studies focused on this topic (van Dijk et al., 2002). Evaluation is integrated into a map production process since the manual work after automated text placement process essentially tackles the work of it. Therefore, map evaluation is of significance for improvements since its results should be a guidance of subsequent processing.

Artificial intelligence (AI) has become popular in many areas in recent decades, such as image recognition (Wu et al., 2015), image classification (Zhao & Du, 2016) and robot technology (Levine et al., 2018). Many researchers advanced previous methods in cartography with deep learning approaches. For example, some researchers discussed deep learning methods for map generalization (Sester, Feng, & Thiemann, 2018; Touya, Zhang, & Lokhat, 2019) and point labelling (Haunert & Wolff, 2017). The study on map labelling using AI methods, however, remains as a gap and requires more exploration. The achievements of deep learning in recent years show more possibilities for its application in text placement domain, but integrating AI into text placement encounters several challenges. On the one hand, how to quantify relationships between labels and objects into a model is an essential problem. On the other hand, AI methods need quite a large size of datasets for training, which could ensure a satisfactory accuracy.

1.2 Aim

To fill the gap in related work, this master thesis proposes a text placement evaluation method based on an AI network. The aim is to design, implement and assess the AI method for evaluating text placement.

1.3 Disposition

The remainder of this master thesis is arranged as follows. Section 2 provides a review of previous studies on text placement rules, automated text placement method, map evaluation and AI techniques. Section 3 presents a theoretical framework for integrating AI into text placement evaluation process. Section 4 presents the text placement evaluation results. The discussions on the performance of the proposed method, encountered issues and limitations are provided in Section 5. Section 6 summarizes the main conclusions.

2 Literature review

2.1 Text placement rules

A map contains substantial information that competes for the limited space on the map; therefore, some cartographic rules are necessary for managing placement. Cartographic rules strongly associate annotations with map objects in all forms of maps, such as web maps and paper maps. There are some slight differences between a traditional paper map and an interactive web map, while in this section only the rules for paper maps are discussed. Many studies have proposed and summarized labelling rules to achieve a high-quality map (Alinhac, 1962; Yoeli, 1972; Imhof, 1975). Cartographic rules for different types of objects (i.e., point, line, and area feature) are separately explained. However, there are some foremost and universal rules that apply to all objects. Labels should have distinct and close associations with their referents; on the other hand, labels should not overlap with other labels or point features. Peculiar rules for different features are listed as following.

Point features

- A label should be closely around the corresponding point feature.
- A label on the right of the point is preferred than on the left of the point, and a label above the point is better than below the point (Fig. 2.1).
- A horizontal label is more suitable for users' visual habits.
- A label should align on the same side of the point when the point and a line are placed too close (Freeman, 2005).
- A polygon boundary should not split a label and its point referent when the point is located within an area feature.
- Leader lines and numbers are also used for labelling in some maps with dense features.

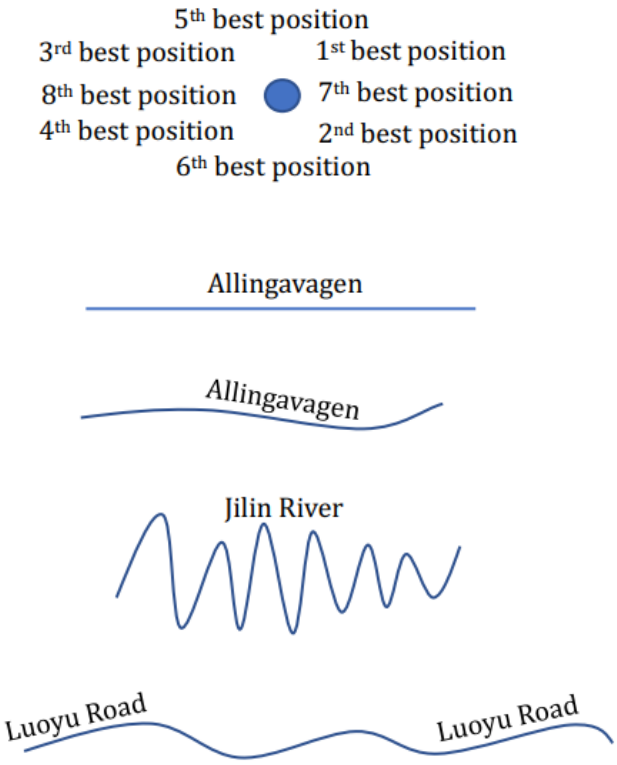


Fig. 2.1. Text placement position of a point feature in order of priority and some examples of text placement of line features (Modified from Slocum et al., 2005, p. 216-217).

Line features

- A label is generally placed along the object and conforming to the object's curviness.
- Especially for city maps, the placement of street names can be along with the street object or within the objects (Chirié, 2000; Freeman, 2005).
- Some special scenarios are worth consideration (Fig. 2.1). A text could be placed more than one time for a long line feature. Besides, a label could be placed parallel to the trend line of the line feature if it has intensive curves.

Area features

- A label is ideally positioned inside the polygon unless there is not enough space inside the area feature for the label. The other choice is placing the text label outside the feature with a leader line or numbering the feature with a legend (Freeman, 2005).
- Horizontal text placement is preferred for better legibility. Vertical and angled placement are options under space limitation (Freeman, 2005).
- A label should not overlap with other labels or point features.

- A text should be labelled more than one time if the area feature has several disconnected parts.

2.2 Automated text placement methods

Many studies have been carried out to improve automated text placement methods. Scholars explored different algorithms for automating text placement, and some methods were implemented into software since the 1970s (Kern & Brewer, 2008). The automatic text placement logic comprises three steps, namely, candidate selection, layout evaluation, and placement decision (Yoeli, 1972; Hirsch, 1982). The first step identifies several possible positions for each map object. A global evaluation is performed through all candidates in the second step, and the third step makes the final decision based on evaluation results. Many proposed methods are based on this idea.

A rule-based algorithm is a basic form of automatic text placement methods. It labels map objects based on defined cartographic rules and weights. For example, Freeman and Ahn (1984) proposed a rule-based system called AUTONAP, which labels map objects sequentially. All candidate positions for each map object are identified at first. Then the annotations of polygon features, point features, and line features are sequentially placed based on conflict detection. However, the system is not flexible to some extent since there is no backtracking which limits its performance on dense maps. Some advanced methods are developed to improve flexibility, such as Jones (1989) who developed a sequential placement method with backtracking, Freeman (2005) who updated the initial rule-based algorithm. Later, some studies progressed the rule-based algorithms by programming cartographic rules into a language (Jones, 1993). Prolog, a logic programming language, is used to define cartographic rules for identification of trial positions. A subsequent conflict detection finds a satisfiable combination for all objects, and then labels are placed.

Along with technology development, an expert system is introduced, which is a system that consists of numerous rules and knowledge from cartographers and skilled-experts. It was widely accepted in applications; for example, an expert system was used in the United States census map production (Ebinger & Goulette, 1990). Besides, there are also some improvements proposed for certain types of features. Candidate positions for a point were changed from only eight fixed positions (Yoeli, 1972) to any position around the point (Hirsch, 1982). These improvements make text placement problem easier for a map with high-density features.

An optimization algorithm is used in automated text placement methods for better quality (Christensen, Marks, & Shieber, 1994; Edmondson et al., 1996; Zoraster, 1997). This method's framework consists of two main processes. The first process is a mathematical evaluation for each label. This process considers many factors, such as disturbance, legibility. Cartographic rules abovementioned could also be criteria for the evaluation. For example, Zoraster (1997) defined a combinatorial optimization method for point labelling. A cost function was developed measuring the cost from four aspects (i.e., properties, disturbance, overlap, and removal), where the better result is the solution with the lowest cost. The second step generates an optimal solution by comparing different position combinations in each iteration. Some studies used local optimization methods to solve label conflicts iteratively (Christensen, Marks, & Shieber, 1994, 1995). It starts with a selected solution and updates the solution in each iteration if a new position of text could improve the solution. It stops when there are no changes after a specified number of iterations.

Slope descending algorithm is an example of the optimization algorithm, which could find the best result in an acceptable time (Christensen, Marks, & Shieber, 1995). However, this kind of local optimization method has a fatal weakness that the number of global optimization solution is much less than that of local optimization solutions. A solution could trap in an optimal local position if an intermediate improvement significantly changes the solution. Some algorithms are introduced to solve this problem, such as genetic algorithm (Verner et al., 1997; van Dijk, 2001; van Dijk, Steven, & Dirk, 2004; Yamamoto & Lorena, 2005; Liu & Lü, 2008). Simulated annealing algorithm (Russell & Norvig, 1995) was also widely used in many studies since it could skip a locally optimal position to the global optimized position (Zoraster, 1997). Many experiences have proven the excellent performance of optimization algorithms for automated text placement (Edmondson et al. 1996; Zoraster, 1997).

Slider model algorithm is another classic automated text placement method (van Kreveld, Strijk, & Wolff, 1999; Strijk & van Kreveld, 2002; Kameda & Imai, 2003; Zhang & Harrie, 2006a, b). It significantly extends the choices of labels' positions based on a no-conflict basis. A slider model is generally divided into three stages. Firstly, all possible labelling areas are generated for each object sequentially. Secondly, each candidate area shrinks based on the detection of obstacles for each object. Thirdly, the final position is chosen from left areas considering the legibility and association

between label and map object. Moreover, some studies also combined other algorithms, such as introducing a leader line to indicate a particular point in a dense area (Kameda & Imai, 2003).

2.3 Map evaluation

2.3.1 Map evaluation methods

Great efforts have been directed towards map evaluation process for assessing differences between desired and achieved presentation of a map (van Dijk et al., 2002; Mackaness & Ruas, 2007; Stoter et al., 2014). A map evaluation was initially performed by visual evaluation, which is considered to be the most intuitive and inefficient method since different examiners have different preferable designs. However, automatic drafting, generalization and text placement speed up the map production process and more maps are demanded nowadays. Therefore, visual evaluation is not ideal in many cases, and quantitative methods are expected.

Many quantitative map evaluation methods are proposed for assessing a generalization process (Brewer et al., 2013; Zhang et al., 2013). For example, some approaches have paid attention to an estimation of a generalized map's readability with supervised machine learning methods (Harrie, Stigmar, & Djordjevic, 2015). However, fewer approaches have been proposed for a text placement evaluation. An example is a combinatorial optimization method that integrates the map evaluation. This evaluation is replaced by a cost function, which scores different text placement combinations, and the final output is the best combination. However, such quantitative measurement only assures a theoretical satisfaction of text placement without any considerations on aesthetics or exceptional cases. To bridge this gap, van Dijk et al. (2002) made progress by proposing a quality function to evaluate text placement from four aspects, namely, aesthetics, label visibility, feature visibility and the association between label and feature based on a series of rules.

2.3.2 Text placement measurement

Text placement is a complex rule-based process that is subjective to cartographers' experiences. Therefore, no measurement can fully describe text placement, which, as noted by some scholars who summarized the criteria for text placement, there are four measurements that are important for text placement (van Dijk et al., 2002; Hirsch, 1982; Huffman & Cromley, 2002; Kern & Brewer, 2008).

Aesthetics

Design of text varies with map type, mapmaker, or other factors. Despite the fact that some parameters (e.g., the font type and font colour) are not relevant to text placement, they are still affecting the aesthetics of a map. A mapmaker's experience decides text distribution on a domain. Some cartographers tend to place texts dispersedly, while some people prefer to cluster texts.

Legibility

Many factors have impacts on legibility, for example, font type, font size, and label position. Font should be striking and easily identified. Different objects' text placement should be avoided on a horizontal level (van Dijk et al., 2002).

Disturbance

A text should not obscure and cover other map information except the corresponding line or area feature.

Association

Associations between text labels and their corresponding features should be considered, where labels should be close to their referents and not separated by other labels or features.

2.4 Artificial intelligence

AI is a technique that simulates human intelligence, and its prosperous development draws much attention from different disciplines, such as image recognition, robotics, and expert system (Hohnson & Basoglu, 1989; Durkin, 1997). For example, expert systems played important roles in many fields that require rule-based decisions, because they could provide inference based on pre-stored expert knowledge, release workload and improve work efficiency. In addition, considerable AI algorithms are also utilized in cartography related filed, such as deep learning, evolutionary algorithm, and Bayesian network. Researchers used random forest classifier to achieve the extraction of urban landmarks (Lin et al., 2019). Identification of road networks was also performed by multiple deep learning models (Fu et al., 2016; He et al., 2018).

2.4.1 Machine learning

Machine learning methods are widely utilized nowadays, and they are playing important roles in geography-related fields.

Supervised and unsupervised classification methods play important roles in remote sensing image processing. Traditional land use classifications use conventional classifiers, such as Minimum distance, Maximum-likelihood, and K-means. These methods could provide moderate accuracy since the 'curse of dimensionality' reduces its performance (Chen et al., 2014). Machine learning methods provide a possibility to achieve higher accuracy, and some studies have proven the better capability of support vector machine (SVM) in a series of common machine learning algorithms (Ambikapathi et al., 2013; Gualtieri & Chettri, 2000; Chen et al., 2014).

Regression algorithms can reveal relationships between two or more factors, and these algorithms are welcomed in the spatial analysis field. Traditional regression models, such as ordinary least squares and geographically weighted regression, can only build a linear relationship between factors; however, relationships are always more complicated (i.e., non-linear relationship). Machine learning regression methods could enrich details of such a complex relationship by a decision tree (e.g., random forest) or network (e.g., neural network). For example, Zhang et al. (2017) used random forest regression to disclose the relationship between tea expansion and local biophysical as well as socioeconomic factors in China.

2.4.2 Deep learning

One type of machine learning is denoted *deep learning*. Deep learning algorithms are highly involved in geography studies and gradually replace traditional methods. Its essence is training a model with large datasets and generating outputs based on the model. There are many types of deep learning approaches widely used in different subfields of geography, while they are all based on neural networks.

Neural network shows strong image interpretability and tackles many image recognition tasks (Almeida et al., 2015). Introducing convolutional layers and pooling layers strengthens the capability of a neural network (i.e., convolutional neural network, CNNs). Many improved networks are proposed based on these advancements, for example, SegNet (Badrinarayanan, Kendall, & Cipolla, 2017), GoogLeNet (Szegedy et al., 2015), and AlexNet (Krizhevsky, Sutskever, & Hinton, 2012).

Previous studies proved the performance of deep learning; for example, He et al. (2018) proved the feasibility of utilizing AlexNet to identify road junctions.

2.4.3 AI methods applied in cartography

Using AI methods in text placement was earlier a research hotspot in cartography domain, and many scholars attempted to achieve it before 2003 (Freeman & Ahn, 1984; Ebinger & Goulette, 1990; cf. section 2.2 above). A mature expert system technique ensured its application in solving text placement issues. Johnson and Basoglu (1989) discussed whether an expert system could be used in text placement for more flexibility. Freeman (1988) developed an expert system that could automatically enter and label texts. The capability of an expert system became insufficient and inefficient for dense-featured maps. Therefore, later studies shifted to other automated text placement methods, where some evolutionary algorithms are introduced to improve text placement approaches. Zoraster (1997) incorporated a simulated annealing algorithm for labelling point features. Yamamoto and Lorena (2005) proposed an approach to position point features using constructive genetic algorithm. However, integrating AI methods into text placement did not have significant progress in the last decade.

Using AI methods in map generalization has also drawn researchers' attention in recent years (Steiniger et al., 2008; Harrie, Stigmar, & Djordjevic, 2015; Ma, 2017; Touya, Zhang, & Lokhat, 2019). Traditional map generalization tasks are solved based on geometric considerations (Sester, Feng, & Thiemann, 2018). However, map generalization's graphic essence makes it an ideal application for deep learning (Touya, Zhang, & Lokhat, 2019). Many studies proposed AI generalization methods for different applications. Lee et al. (2017) proposed a building generalization method based on deep learning classification algorithms. They evaluated the capability of four algorithms (naive Bayes, decision tree, k-nearest neighbour, and support vector machine), and the satisfactory accuracy results proved their reliability. Similarly, Zhou and Li (2017) tested the performance of different deep learning approaches in road generalization. Multiple neural networks are applied in building generalization domain (Feng, Thiemann, & Sester, 2019). However, AI methods are only performed on how to achieve generalization instead of evaluating generalization performance.

2.5 GoogLeNet

GoogLeNet is a deep convolutional neural network used for improving object detection capability to meet growing needs (Szegedy et al., 2015), and has achieved great influence since 2014 as the winner of the ImageNet Large Scale Visual Recognition Competition 2014 (ILSVRC14). It draws increasing attention on many domains, such as image recognition (Zhong, Jin, & Xie, 2015), classification (Singla, Yuan, & Ebrahimi, 2016), and detection (Salavati & Mohammadi, 2018).

Along with increasing needs for object classification and detection, previous deep learning networks have insufficient capabilities and go into a design problem when searching for improvement. The common way to improve performance is increasing network depth and width, which refer to the number of layers and the number of neurons of a network, respectively. However, such method limits development from several aspects. Firstly, too many parameters caused by an expansion of a network scale lead to overfitting issues when the training dataset is limited (Szegedy et al., 2015). Secondly, consequent computational complexity makes usage inapplicable and unachievable. Thirdly, there is a vanishing gradient problem, which means a gradient will decrease sharply along with an increase of propagation depth, hinders the model optimization by weakening a model learning ability (Hochreiter, 1998). To break the bottleneck, GoogLeNet introduces *Inception* architecture to optimize a sparse structure to enhance network performance and resource efficiency (Szegedy et al., 2015). Such attempt makes GoogLeNet achieve a higher accuracy but with smaller model size compared to other networks (Russakovsky et al., 2015), which is suitable when there are limited computing resources (e.g., GPU, Storage).

2.5.1 Theory

GoogLeNet has experienced a development of versions of V1, V2, V3, V4, and other versions (Ioffe & Szegedy, 2015; Szegedy et al., 2015, 2016, 2017) and many adjustments are applied to its model structure. For example, GoogLeNet V1 and V3 use input images with 224 x 224 and 299 x 299 image resolution, respectively. Their core theoretical basis, however, remains the same regardless of improvements. The theory of GoogLeNet V1 is introduced below, and a brief description is based on Szegedy et al. (2015).

GoogLeNet V1 has 22 layers based on *Inception* module (Fig 2.2(a)). An *Inception* module stacks convolutions (1x1, 3x3, 5x5) and max pooling (3x3) together. A convolution is a mathematical operation on input with a moving window. Max pooling extracts the largest value from input, where a moving window covers it. The main idea of this structure is increasing computational resources utilization within the network. Convolutional layers (3x3 and 5x5) can extract every detail of input and max-pooling can reduce feature size and avoid overfitting. At the same time, to reduce the large thickness of a feature map which caused by excessive calculation, 1x1 convolution layers are added before 3x3 convolutions, before 5x5 convolutions, and after max pooling. Such a structure benefits the model from dual perspectives. On the one hand, it increases the width of the network; on the other hand, it also increases the adaptability of the network to scale. Several *Inception* modules stack upon each other to achieve the full architecture of GoogLeNet. An excerpt from GoogLeNet model is depicted in Fig 2.2(b), which demonstrates how two Inception modules are connected. There are more layers in the model, and the full architecture of GoogLeNet V1 is presented in Appendix A.

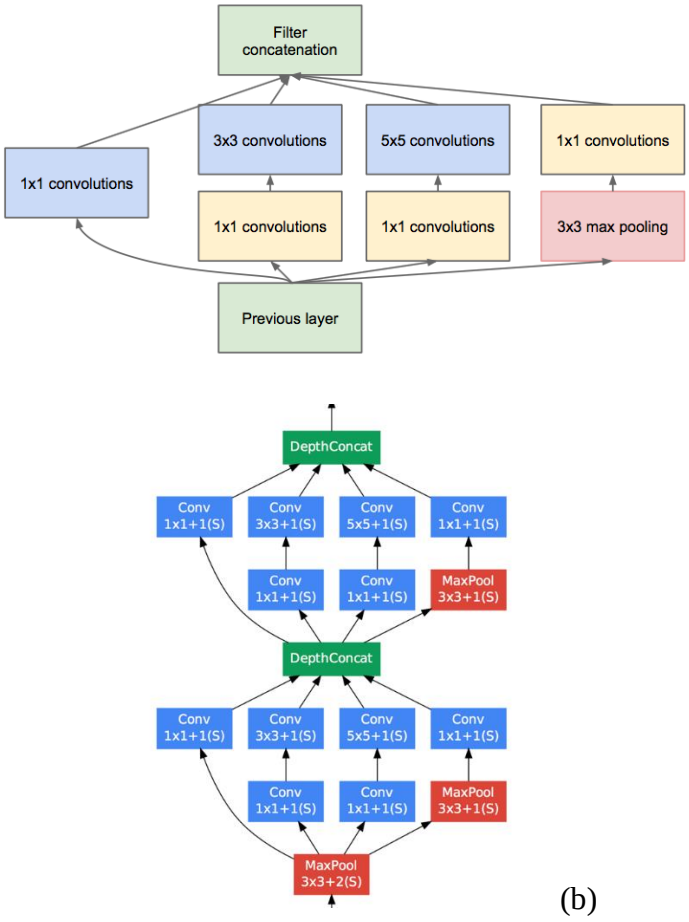


Fig. 2.2. Structure of *Inception* module in GoogLeNet (a) and an excerpt of GoogLeNet full architecture (b) (reproduced from Szegedy et al. (2015) with permission from publisher IEEE). **© Copyright [2014] IEEE**

The essence of GoogLeNet model is identifying a series of features or patterns that are hidden behind pixels. Three datasets are involved in this process, namely, training dataset, validation dataset and test dataset. GoogLeNet uses a multilayer perceptron to capture these features. The training dataset is entered into the input layer and passing through many hidden layers between the input layer and the output layer. These hidden layers have parameters whose weights are adjusted in the model through continuous learning. At the same time, GoogLeNet model verifies accuracies by the validation dataset and thus improves learning. The features are finally obtained in the output layer. Training finally stops when the number of iterations achieves the maximum iteration or specified accuracy is achieved. Then the trained model reads images from the test dataset and classifies them based on knowledge. Each test image is given one label which has the highest accuracy among all candidates.

Since such CNN aims at simulating a human brain, it needs a substantial number of parameters. Most parameters are within layers and trained to build model, while some essential parameters are outside layers and used to define model training. Learning rate refers to the number of updated weights towards minimum loss function during iterations, and it varies from 0 to 1. The loss function is a function of evaluating how well the network accomplishes its intended work. The smaller the loss function is, the more accurate the model is. There are many loss functions for CNN, such as mean squared error, cross-entropy, and categorical cross-entropy (Zhao et al., 2016). Optimizer algorithm is an algorithm that updates parameters to achieve lower loss function. Batch size is the number of utilized images in each iteration. Maximum iteration is the maximum number of training iterations if a model does not achieve specified accuracy.

2.5.2 Applications

Many scholars have used GoogLeNet to achieve various applications, and some examples are introduced as follows.

Szegedy et al. (2015) proposed GoogLeNet and attended ILSVRC14 in object classification competition. 1.2 million images from ImageNet dataset were used to train the model. Its training was performed with CPU on DistBelief (Dean et al., 2012) distributed machine system; however, the

authors believed it could be trained within one week by using multiple high-performance GPUs. GoogLeNet won the competition with 93.33% accuracy in classification.

Zhong et al. (2015) achieved a high-performance GoogLeNet model for handwritten Chinese character recognition. They used 2.14 million characters to train the model where each character image had an image resolution of 112 x 112. The experiment was run with GTX TITAN BLACK GPU with 6 GB memory. The final testing accuracy of this model is 96.26%.

Singla et al. (2016) classified and categorized food and non-food images with pre-trained GoogLeNet. A total of 13296 images from two datasets were used for model training in the study. They trained and verified the model for food as well as non-food classification and food categorization, respectively. Food and non-food classification model obtained accuracy over 90% with 10,000 iterations training, and food categorization model achieved over 70% accuracy with 40,000 iterations.

2.6 Challenges for AI methods in cartography

As discussed in section 2.3, the AI technique includes a variety of algorithms, and some of them are implemented in cartography field. For example, expert systems were popular in automated text placement tools (Freeman, 1988; Ebinger & Goulette, 1990). Although expert systems contain explicit knowledge information of cartographic rules, its disadvantages hinder its application. A critical disadvantage is the insufficiency of knowledge acquisition. An expert system essentially provides information based on pre-stored rule-based knowledge, while it is difficult to update knowledge in time. Besides, the size and computational complexity of an expert system will decrease its performance when knowledge reaches a certain level.

Given the above disadvantages, machine learning and deep learning algorithms provide new approaches to boost AI techniques (Yanase & Triantaphyllou, 2019). Despite new machine learning and deep learning techniques are experiencing a rapid and prosperous development in recent years, their usages in cartography area are still insufficient. Only a few studies that focused on map generalization were completed (Touya, Zhang, & Lokhat, 2019; Harrie, Stigmar, & Djordjevic, 2015). Especially for the era of text placement evaluation, the vacancy needs to be filled. However, there are many challenges for the utilization of these algorithms in text placement evaluation.

- Deep learning algorithm requires a fairly large amount of datasets to train a model; however, different maps adopt dissimilar cartographic rules which leads to a lack of enough training materials where insufficient training data cannot generate a satisfying result.
- Deep learning networks consists of complex parameters and causal mechanism, which makes it difficult to achieve an excellent result by modifying parameter settings.
- The language barrier also impedes utilization. Image recognition gives preferable accuracy for different language characters; however, these characters cannot transfer between different languages. For example, a map with Swedish text labels cannot be used as training material for a map with Chinese labels.

3 Materials and methods

3.1 Conceptual framework

This master thesis is a part of a large project, and the integrated conceptual framework of the large project is presented in Fig. 3.1. This project aims to improve the map design product line for T-Kartor, a company that provides cartographic service. The current product line of text placement in T-Kartor is mainly made by an automated text placement tool and followed by manual modifications. However, massive data put heavy workloads on cartographers and consume too much time. The company is working to establish a new method to update the product line for increasing automated text placement quality. This study runs in parallel with another master thesis that focuses on text placement design with PAL labelling library in QGIS (Cederholm, 2020). The map evaluation method proposed in this thesis uses the output map from Cederholm (2020) as input, and the results give feedbacks to text placement for improvements.

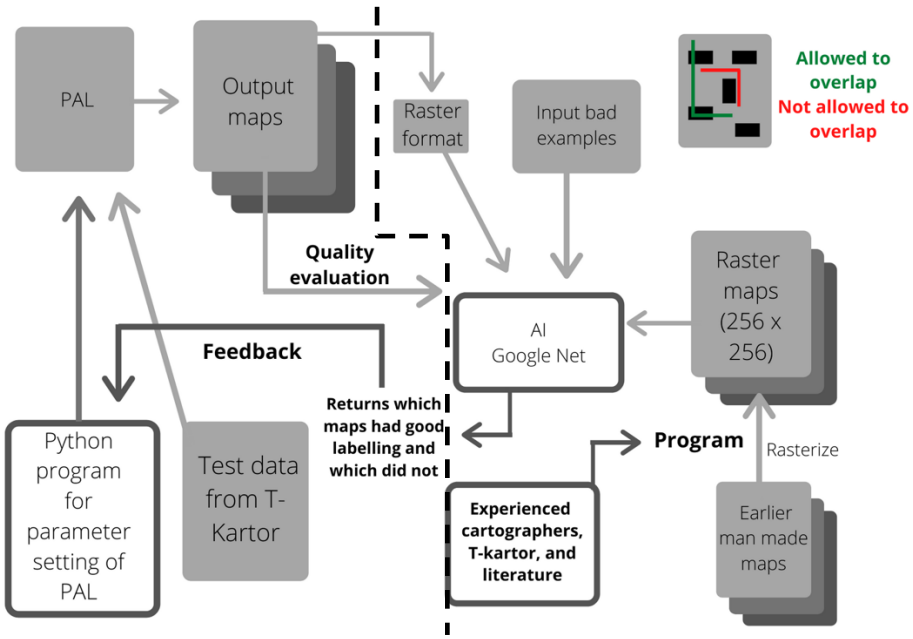


Fig. 3.1. Overall conceptual framework of this master thesis and Cederholm (2020).

The overall workflow includes two parts which are separated by the dashed line. To the left is the project about PAL parameter settings, and to the right is the project about text placement evaluation. AI model is the connection of two projects, and we choose GoogLeNet for this master thesis. Text placement evaluation consists of three subsections. Cartographic rules formulation is

needed for simplifying maps. An example of a simplified map is shown at the upper-right corner in Fig. 3.1. Based on cartographic rules, maps are processed and saved by programming for dataset preparation. Finally, GoogLeNet is trained with these datasets and is used to provide responses to the other project.

Following steps explain the detailed text placement evaluation process. Firstly, simplify previously manually produced maps from T-kartor based on some cartographic rules. The cartographic rules are summarized through an interview with experienced cartographers at T-kartor (see Cederholm, 2020). Current text labels on the map are categorized into several classes according to their corresponding feature types. For example, map features could roughly divide into two types depending on if they are allowed to be overlapped. Secondly, rasterize and clip maps. Simplified maps are exported from shapefile format and clipped into a specified size since AI methods have strong image identification ability and input requirements. Thirdly, generate bad text placement examples for training AI model. Fourthly, build an AI model with enough good and bad examples as input. Fifthly, use the trained network to evaluate map labelling conducted by the PAL program (as accomplished in Cederholm 2020). Finally, feedbacks from GoogLeNet could be used to enhance parameter settings in PAL.

3.2 Materials

T-kartor provides all materials used in this study. The data source is the London street maps project conducted by T-kartor, which is designed for Transport for London (TfL). These urban street maps provide wayfinding information for citizens and tourists, such as the metro, bus, and ferry. Many unnecessary area map objects (e.g., buildings) are simplified for a clear representation. These maps are produced at 1:2250 scale and are intended to be placed in London with printed versions, but for this study, T-kartor provides originally edited shapefiles.

3.3 Cartographic rules formulation for rasterization

Cartographic rules applied in this study are summarized from an interview with cartographers at T-kartor (for details, see Cederholm 2020). Specifically, these labelling rules only apply to the London street maps. Detailed requirements for the map design (e.g., typography, mapping scales) are provided in ‘Streetmap design standard Issue 1 – Transport for London’. Text characteristics (i.e.,

font type, size, colour) are not taken into consideration in this study as they are specified, and good legibility is the foremost rule among all cartographic rules.

3.3.1 Point features

Texts of point features should ideally be placed horizontally, and their placement around the corresponding point has a prioritization from the upper left, upper right, lower left, to lower right. This positioning rule only applies to text labels of point features, while icons represent some point features in the London street map. Such a symbol that combines the icon and the point feature clearly shows the point feature without no real point object on the map. The icon clearly indicates the position of the point. Besides, a leader line is used to clearly show the affiliation relationship in some special cases (Fig. 3.2).

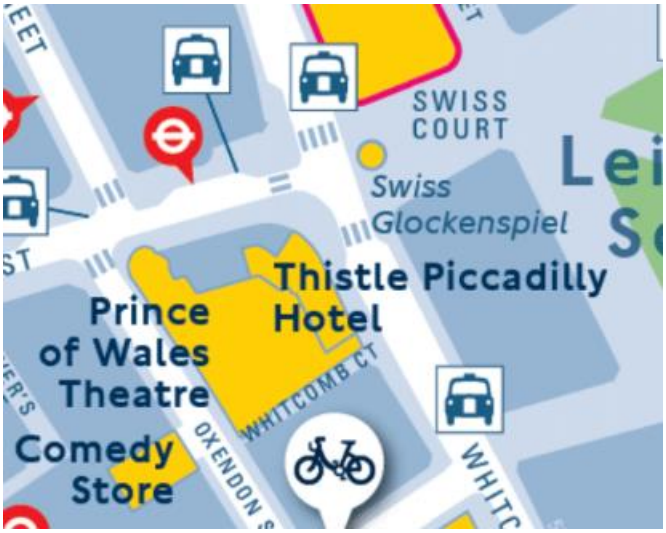


Fig. 3.2. Examples of icons representation for point features (based on data from T-kartor).

3.3.2 Line features

Different rules apply to various types of line features. Street names are positioned within line features and conform to the curviness of streets. Straight labels for line features are preferred for good legibility. On the one hand, texts should conform to the curve of features if it is significant; on the other hand, texts could keep straight when the curve is slight. Besides, texts could be split into more lines or be shortened to fit street objects. A text label should be located at the centre of the relevant feature instead of the start or the end. Labels for a long street should repeat certain times for better

recognition. When a text label meets an intersection, the text should be positioned across the intersection to explicitly indicate the street name if two segments belong to the same street.

In addition to street line features, there are also some lines representing zebra crossing and pedestrian subways. These features do not have labels for identification; however, labels for street features should not overlap them. An example is shown in Fig. 3.3.

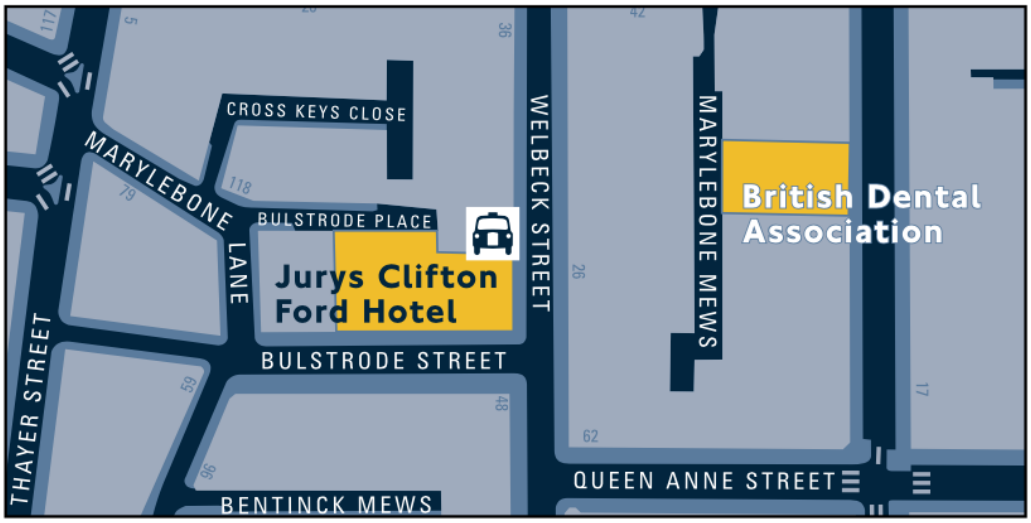


Fig. 3.3. Examples of text placement of road features, e.g., BULSTRODE STREET (based on data from T-kartor).

3.3.3 Area features

Labels of area features should ideally be located within the features and allow overlapping of the boundaries if necessary. Labels should use adaptive text alignment based on the corresponding area features' shape and position to avoid stack. For example, the 'British Dental Association' label needs to be located towards the right to avoid the overlap with the street name in Fig. 3.3. A text should always have an intersection with its relevant area feature unless the polygon is too small to contain even a fraction of the text. The prioritization of label placement around a point feature (Section 3.4.1) also applies to the centroid of an area feature. Different cases introduced in this section are shown in Fig. 3.4.

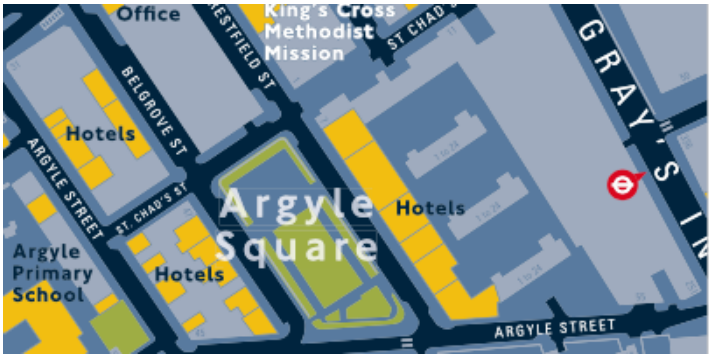


Fig. 3.4. Examples of text placement for area features, e.g., 'Hotels' refers to the surrounding yellow area features (based on data from T-kartor).

3.3.4 Additional rules

All texts and icons should generally avoid overlaps with each other for legibility. However, a special scenario for text label is the label for neighbourhood, borough or village. The text labels for such large features are allowed to overlap other texts if necessary (Fig. 3.5).

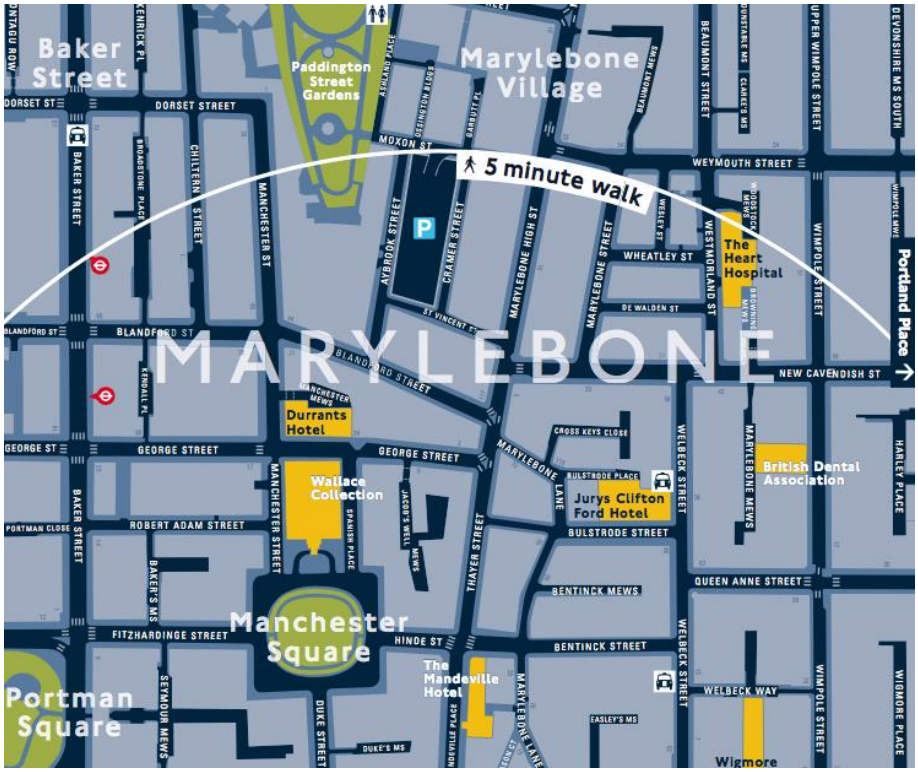


Fig. 3.5. A special scenario for a Borough text label, e.g., MARYLEBONE (based on data from T-kartor).

3.4 Methodologies

The workflow of methodologies in this master thesis is shown in Fig 3.6. Maps from T-Kartor and Cederholm (2020) are firstly simplified based on cartographic rules, and they are subject to rasterization. Manual quality level evaluation is performed for each rasterized image, and these

images are split into the training dataset, validation dataset and test dataset, respectively. Finally, GoogLeNet model is trained, verified and assessed by these datasets.

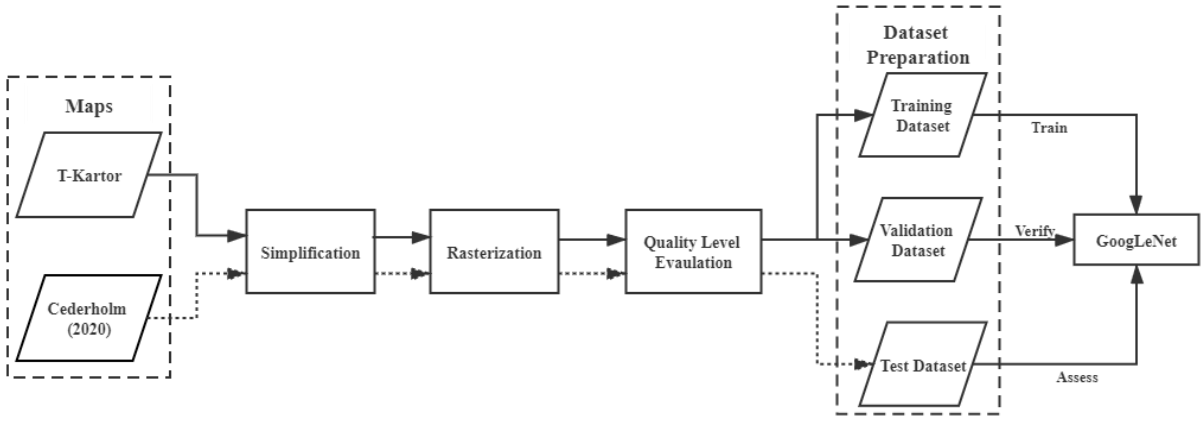


Fig. 3.6. Methodologies workflow.

3.4.1 Rasterization

3.4.1.1 Simplification

The original maps contain dense information and objects (e.g., multiple layers for different area features) which potentially weaken the performance of text placement evaluation because of the image complexity increase. Different features use dissimilar label settings (e.g., font size, font type), and this makes the text evaluation complex. Besides, the font settings for each area feature are pre-defined, which means the content of the text has no impact on placement evaluation. Therefore, it is essential to perform simplification to maximize the text placement characteristics, and relations before the map samples are used for training the AI network, and also before the evaluation.

Simplification is achieved by feature selection and label transformation. Firstly, the map features need selection. On the one hand, some features are not important to text placement evaluation, where some features use icons as symbology or some features have no label and are allowed to be overlapped by other labels. Hiding such unimportant features highlights the clear geometric and topological relationships between retained objects and the corresponding labels. On the other hand, each feature with text label and other symbology that are not allowed to be overlapped need to keep in the map. Secondly, converting labels from text to geometric figure turns the evaluation into an image recognition task. A precondition that all font characteristics are pre-defined and unchangeable

makes the transformation possible. Moreover, character recognition increases the complexity and difficulty compared to simple graphs. The principles for simplification are summarized as follows.

Point features

Pre-designed icons represent point features in this project. Some of the point features use Scalable Vector Graphics (SVG) as their representations (e.g., cycle hire station, police station), while others (e.g., info centre) integrate icons with font settings (i.e., the typeface contains the icon). An icon is always positioned on the referent's centric or marginal position (Fig. 3.7). Such placement provides a distinct association between labels and their referents, as well as good legibility. Therefore, it is reasonable to consider the combination of icon and point feature as a whole. Point features' highest rendering priority rationalize the simplification of treating them as a part of the base map instead of changing their symbology.

Fig. 3.7. Icons representations of point features. The centroid of the 'Ticket Stop' icon is the point feature's position, and the tip of the bicycle icon is the point feature's position. (based on data from T-kartor).

Line features

Line features can be classified into two categories. One of them is road features which could not be ignored. Road line features and the space between pavements (i.e., the dark space in Fig. 3.8) both indicate roads in the map. The road centre line's symbology is removed for aesthetic purposes, while the labels still locate on the line. The space between pavements accommodates the road labels. Therefore, the labels are simplified as red rectangles (Fig. 3.8) and should ideally locate within the specified room. The other category is the auxiliary road feature, such as zebra crossing and pedestrian

subways. These features are treated as a part of the base map after the symbology changes because of the lack of labels.

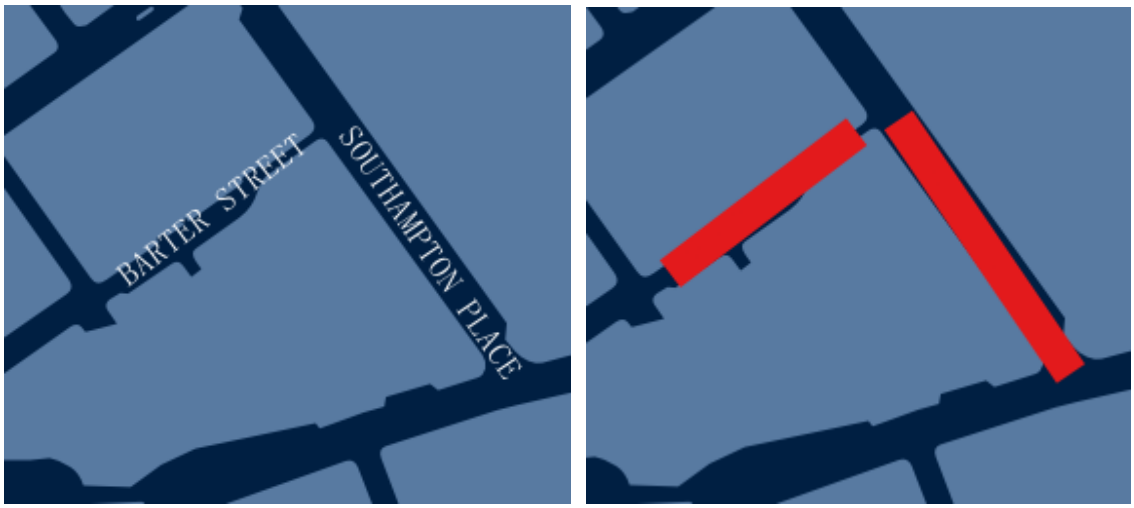


Fig. 3.8. Original road feature labels (left) and simplified road feature labels (right). (based on data from T-kartor).

Area features

There are many kinds of area features on the map, such as water bodies, landmark buildings, and green spaces. Green spaces and normal buildings are not labelled, and water bodies have quite enough room for a single label (Fig. 3.9). Therefore, only the landmark building features are retained and other buildings are hidden in the map. Landmark building features are in yellow, and their labels are simplified as black rectangles (Fig. 3.10).

Fig. 3.9. Four types of area feature in this project (1) Green spaces, (2) normal buildings, (3) water body, and (4) landmark buildings (based on data from T-kartor).

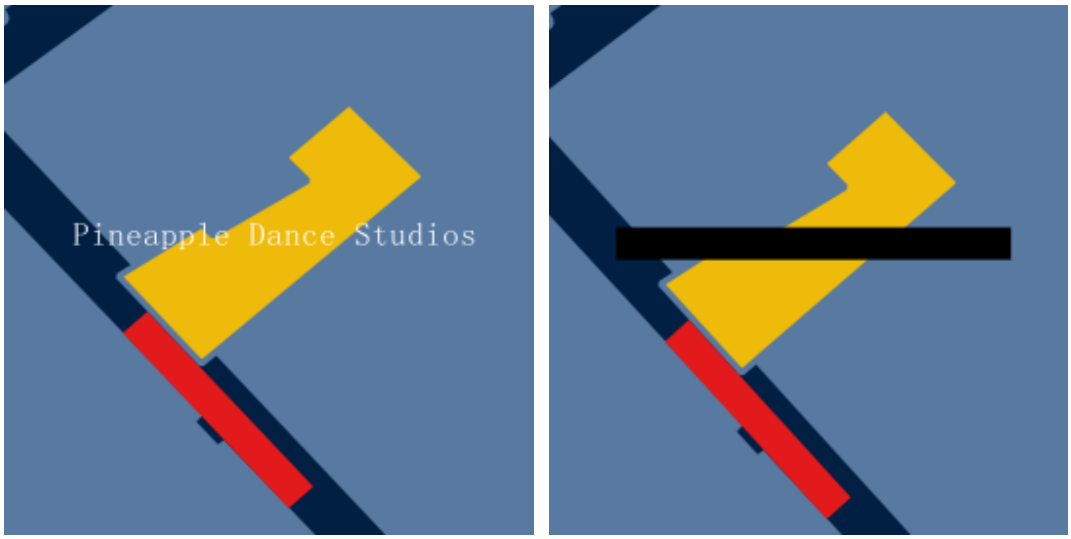


Fig. 3.10. An original area feature label (left) and a simplified area feature label (right) (based on data from T-kartor).

3.4.1.2 Rasterization

The simplified map needs to be rasterized and exported as images with certain image resolution (256 x 256 pixels) for the subsequent AI process. This operation leads to the problem that lower image resolution leads to blurry images. Clipping the map into a series of tiles and exporting them as separate images are the solutions to deal with this problem.

The rasterization python code excerpt is shown in Appendix B. The output image has an image resolution of 256 x 256, and one pixel represents one meter. This image resolution is decided after several preliminary experiments. These experiments test the balance between the amount of content in the image and the number of images. There is too much information in an image if it covers a large part of the map, and an image covering a small part of the map might lack enough information for GoogLeNet training. Therefore, the map from Cederholm (2020) is clipped into 35 images out of the concerns from the above perspectives. The traceback function is achieved by naming images with their row number and column number. An example of a rasterized image is shown in Fig 3.11.

Fig. 3.11. A rasterized map sample with 256 x 256 pixels (based on data from T-kartor).

3.4.2 Quality level schema

A text placement quality measurement schema is designed to quantify text placement quality for each text on the map. The output gives a decision about the performance of all labels in the map. It reflects if the labels are placed in a good or bad position. Input to this function is the clipped images after simplification, and the output is the scores of text placement quality for input images. The quality function is given below (Eq. 3.1).

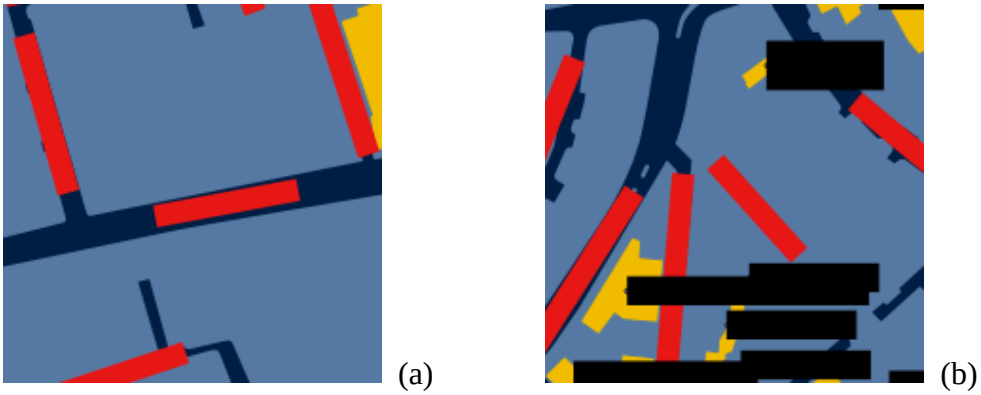
$$Placement\ Quality\ Level = \min_{L_n \in L}(Legibility, Disturbance, Association) \qquad (3.1)$$

L_n refers to the n_{th} label in a label set L. For each label $L_n \in L$, the *placement quality level* function measures text placement from three aspects. Legibility describes whether a text can be easily and clearly identified in the map. Disturbance refers to whether a text is obscuring other map features. Association measures how well a label indicates its referent. There are three quality levels: 1 (Bad), 2 (Moderate) and 3 (Good). For each label in an input image, the quality is given from abovementioned aspects. The lowest score is assigned to the corresponding label, and the lowest score among all labels is assigned to the corresponding image. However, this method has no way to trace back to the label with the lowest quality. One solution is to decrease the number of map objects in clipped images. Besides, point features represented by icons are considered as the base map as discussed in section 3.5.1.1, which omits the measurement of their labels. They are pre-placed on the map and act as an obstacle to other labels. The schema is presented below.

Legibility

Line Features:

- 3 – Labels have no conflicts with any labels and locate at the intended place (Fig. 3.12a).
- 2 – No measurement of moderate quality for line features.
- 1 – Labels have conflicts with other labels, point feature's icon or other line features (Fig. 3.12b).



(a) (b)

Fig. 3.12. Examples of legibility measurement of line feature: (a) Level 3: The middle road label locates at the intended position; (b) Level 1: Road label has a conflict with area feature's label (based on data from T-kartor).

Area Features:

- 3 – Labels have no conflicts with any labels and locate at the intended place (Fig. 3.13a).
- 2 – Labels have small proportion overlap with other landmark buildings (Fig. 3.13b).
- 1 – Labels have conflicts with other labels or point feature's icon. The percentage of a label that covers other landmark building features is greater than the percentage that covers its referent (Fig. 3.13c).

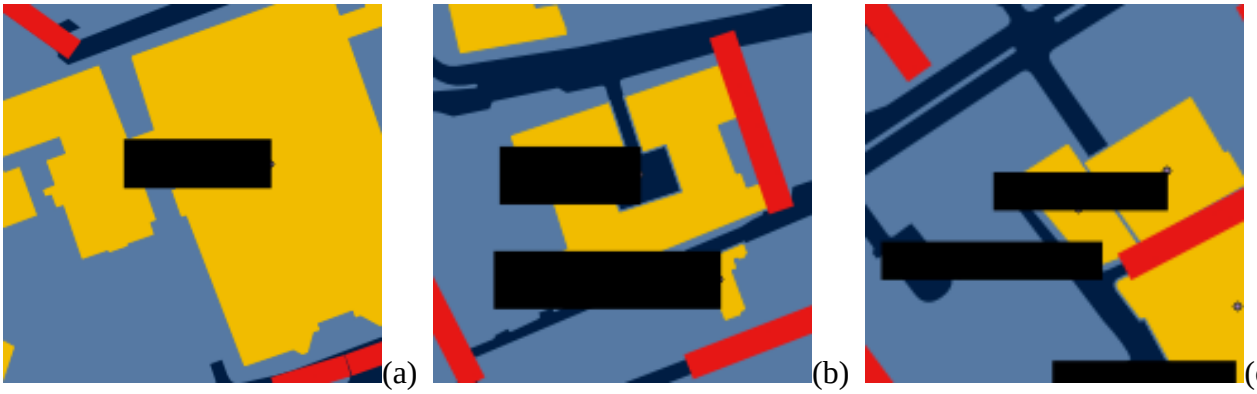


Fig. 3.13. Examples of legibility measurement of area feature: (a) Level 3: Landmark building's label locates at the intended position; (b) Level 2: The bottom label has small proportion overlap with the big landmark building; (c) Level 1: The upper label has large proportion overlap with other landmark building (based on data from T-kartor).

Disturbance

There is no measurement for line features from disturbance aspect. In a city map, the symbology of road features generally has enough width to accommodate their labels, and therefore it is easy to identify road features even if they have conflicts with other labels or icons.

Area Features:
- 3 – Area features are not covered by other labels or icons (Fig. 3.14a).
- 2 – Area features are covered small proportion by labels or icons other than their corresponding labels (Fig. 3.14b).
- 1 – Area features are covered large proportion by labels or icons other than their corresponding labels (Fig. 3.14c).

Fig. 3.14. Examples of disturbance measurement of area feature: (a) Level 3: Only landmark building's label covers the area feature itself; (b) Level 2: The lower label covers the left landmark building; (c) Level 1: The right landmark building's label covers the left small landmark building with a large proportion (based on data from T-kartor).

Association

Line Features:
- 3 – Labels are located within the intended space (Fig. 3.15a).
- 2 – Labels are partly out of the intended position (Fig. 3.15b).
- 1 – Labels are fully out of the intended position (Fig. 3.15c).

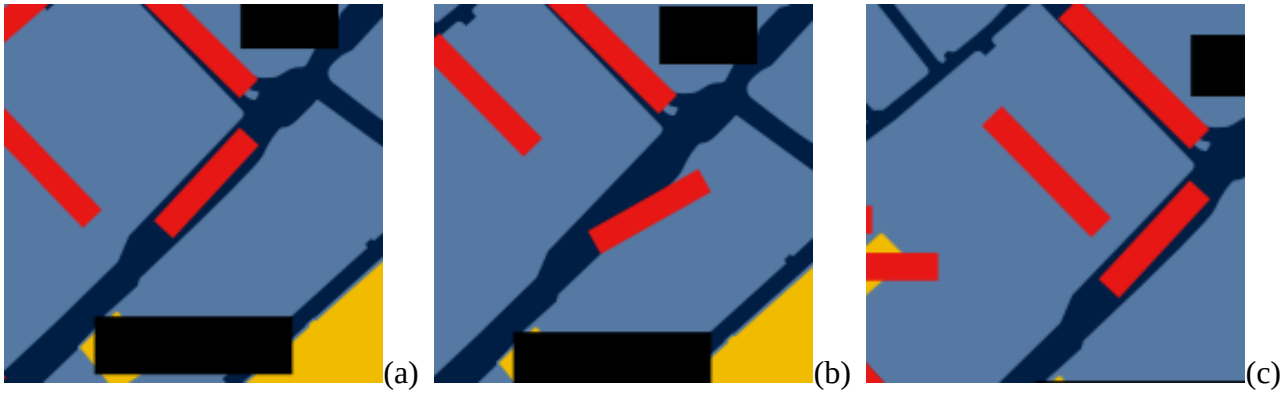


Fig. 3.15. Examples of association measurement of line feature: (a) Level 3: The middle road label locates within the intended position; (b) Level 2: The middle road label is partly out of the intended position; (c) Level 1: The middle road label locates at the wrong position (based on data from T-kartor).

Area Features:
- 3 – Area features are only covered by their labels (Fig. 3.16a).
- 2 – Area features are covered more by their labels than other labels (Fig. 3.16b).
- 1 – Area features are covered more by other labels than their labels or labels do not exist (Fig. 3.16c).

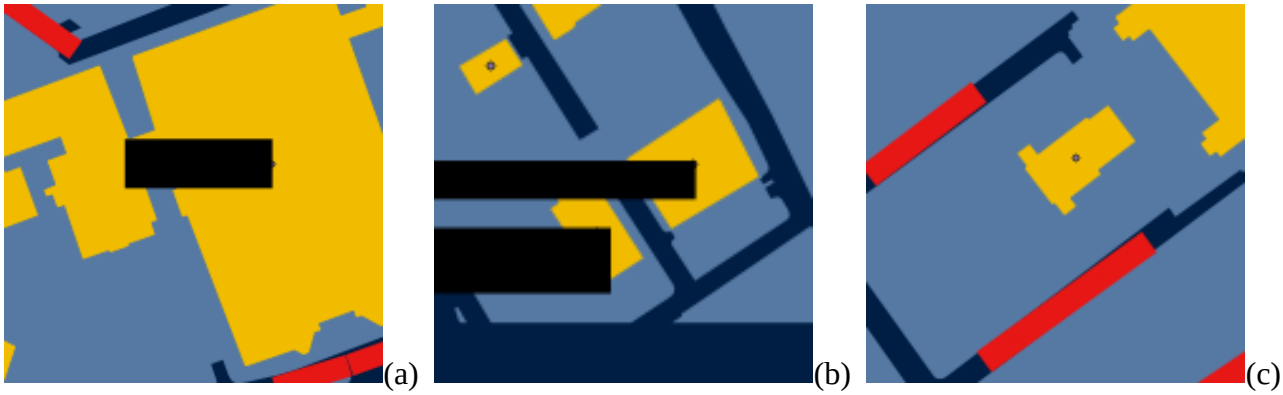


Fig. 3.16. Examples of association measurement of area feature: (a) Level 3: Landmark building is only covered by its label; (b) Level 2: The left landmark building is covered by the right landmark building's label with small proportion; (c) Level 1: The middle landmark building lacks label (based on data from T-kartor).

3.4.3 Dataset preparation

There are three datasets used for training and validating the GoogLeNet model, namely, training dataset, validation dataset and test dataset. The test dataset is acquired from the labelled map of Cederholm (2020) after simplification and rasterization as presented above. The other datasets are prepared by programming in QGIS (Appendix C) and proportionally split into the training dataset and validation dataset.

The following steps create training dataset and validation dataset. Firstly, generate multiple random points across the map's full extent on the base map, which contains all essential objects and information (e.g., icons and features). Secondly, simplify and place road labels with different placement choices (e.g., curve or straight label). Thirdly, several area features are randomly placed at one of the pre-generated random points' positions, respectively. This operation repeats numerous times, and images are saved at each iteration. The centroid of an area feature is used as the centre of an image, and the image is clipped and saved with 256 x 256 image resolution. Finally, all images are manually selected and classified into different quality levels.

In the end, 800 images of each quality level are selected, that means in total 2400 images were used. These images are randomly split into the training dataset and validation dataset, and they include 500 images and 300 images, respectively. For testing the GoogLeNet model, 35 images generated from Cederholm (2020) are chosen. Some examples of training dataset and test dataset are presented in Fig 3.17. There are some clipped labels in the image (Fig 3.17 c, d), which are caused by the automatic dataset generation process. These clipped labels are still treated as normal labels for quality evaluation.

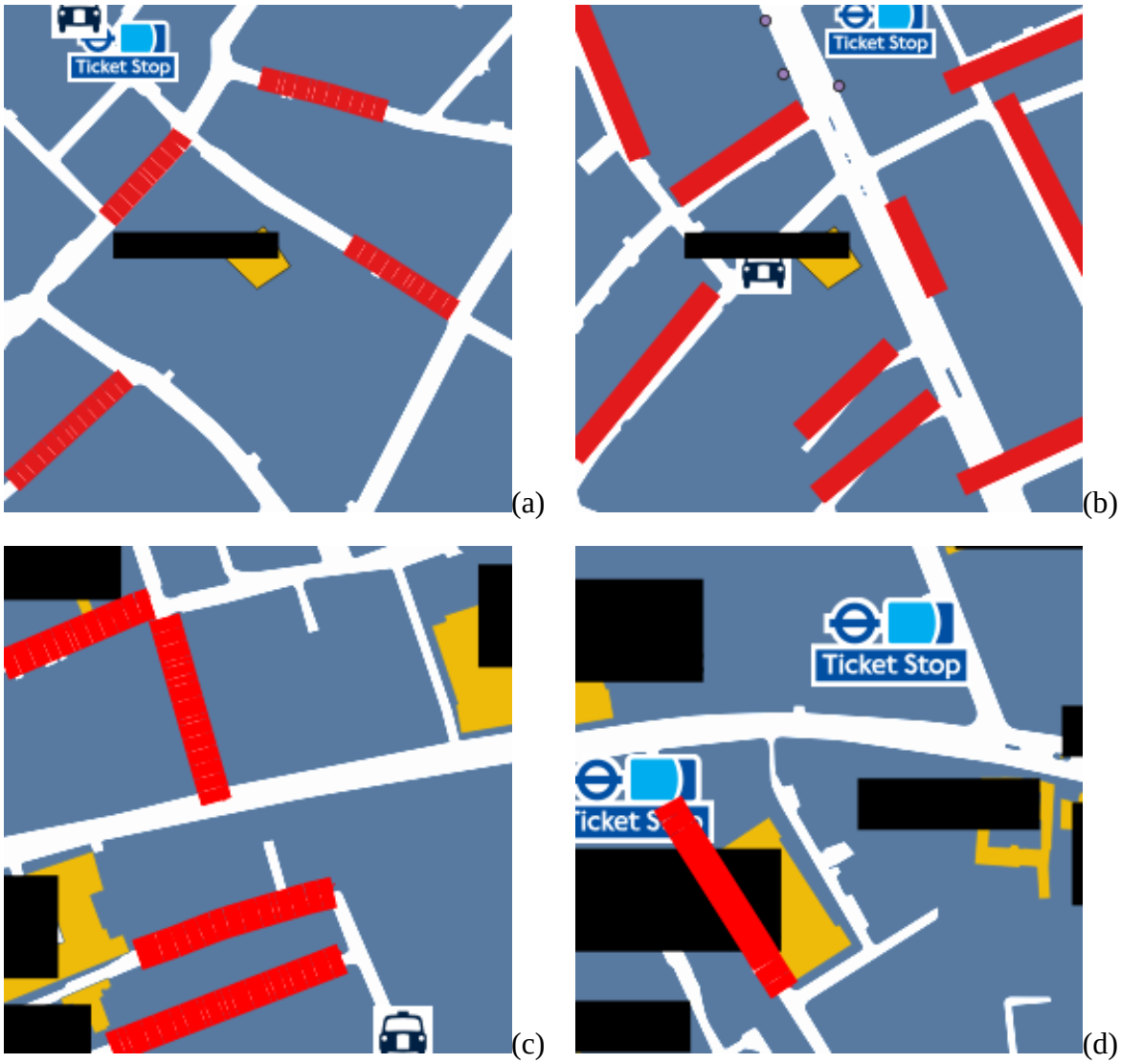


Fig. 3.17. Examples of the prepared dataset. Training dataset and validation dataset: (a) Level 3 - Good; (b) Level 1 – Bad. Test dataset: (c) Level 2 - Moderate; (d) Level 1 – Bad (based on data from T-kartor).

3.4.4 GoogLeNet construction

GoogLeNet V1 is adopted between V1 and V3 in this study. On the one hand, GoogLeNet V1 is the most classic version for introducing a novel structure; on the other hand, some technical problems lead to a failure of GoogLeNet V3 implementation. Original GoogLeNet V1 codes are cloned from GitHub (https://github.com/PanJinquan/tensorflow_models_learning), while several training parameters are adjusted to suit this project. Parameter settings for the experiment are presented in Table 3.1, and these parameters are explained in section 2.5.1. For loss function and optimizer, cross-entropy and gradient descent optimizer (GD) are the most common algorithms for a multi-classification problem. Learning rate was set according to best performance in preliminary experiments. Batch size and maximum iterations are set according to hardware capacity limitations.

Table. 3.1. Parameter settings of GoogLeNet V1 for training.

Parameter	Value
Loss function	Cross-entropy
Optimizer	Gradient descent optimizer
Learning rate	0.001
Batch size	8
Maximum iterations	800

Lastly, hardware capacity also has a strong influence on the training process. This experiment runs on an XPS 15 laptop with 64-bit Windows 10 operating system. Its hardware capacity includes Intel® Core™ i7-6700HQ central processing unit (CPU), NVIDIA GTX960M graphics processing unit (GPU) with 2G memory and 2133MHz dual-core processer.

4 Results

The results indicate that the current GoogLeNet model cannot tackle map evaluation tasks with good performance. Changes in training accuracy and validation accuracy in the training process are drawn in Fig. 4.1. On the one hand, it can be seen that training accuracy has a significant fluctuation within 800 iterations. Training accuracy sharply declines to 0.15 over 100 iterations, which followed by a drastic increase to 0.5 after 400 iterations. Being stable for a short while, training accuracy has a slight descent to 0.31 and rebound to 0.5 at 800 iterations. Training accuracy fitting line indicates training accuracy's uptrend during iterations. On the other hand, validation accuracy does not have a large difference during the process and slightly fluctuates around 0.33. There is no evidence of how accuracy will change. Besides, the loss function remains at a huge number during the whole process.

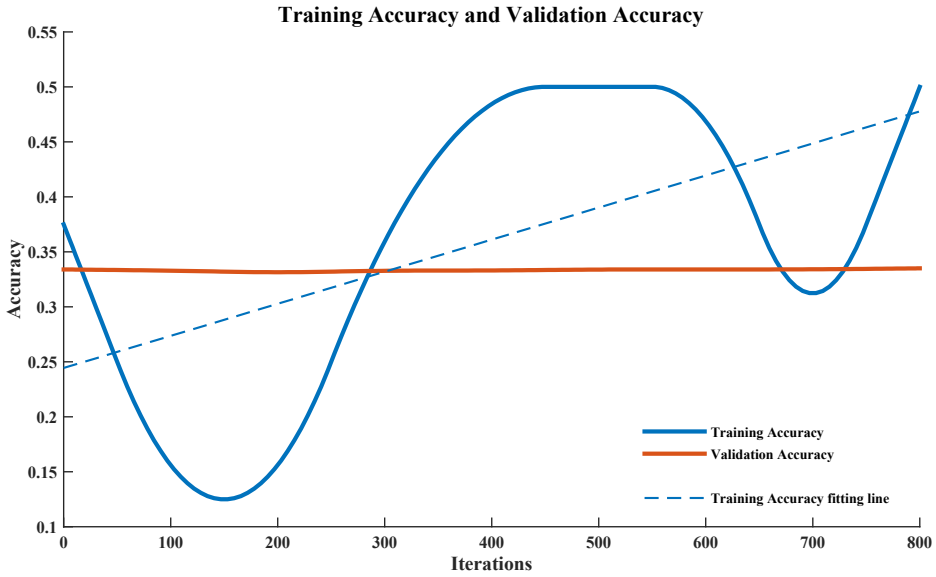


Fig. 4.1. Training accuracy and validation accuracy changes of GoogLeNet in the training process.

The performance of GoogLeNet model is evaluated by the test dataset. The evaluation result is shown in Table 4.1. It can be found that a total of 35 images are entered into the model. The manual classification of them before input identifies 28 images with bad quality, two images with moderate quality and five images with good quality. These images are all classified as bad quality by GoogLeNet model. However, the number of bad quality images in the test dataset is much larger than others, and this imbalance could affect the evaluation result.

Table. 4.1. The evaluation result of the test dataset. A comparison between manual quality level evaluation and GoogLeNet quality level evaluation.

	GoogLeNet Evaluation Quality Level			
Manual Evaluation Quality Level	**Bad**	**Moderate**	**Good**	**Total**
Bad	28	0	0	28
Moderate	2	0	0	2
Good	5	0	0	5
Total	35	0	0	35

To avoid sample bias in the test dataset, another assessment of the GoogLeNet model performance is conducted. A new dataset (TEST2) is generated with 25 images in moderate quality and 15 images in good quality by manual quality level evaluation. The results are shown in Table 4.2 that all images are classified as bad quality.

Table. 4.2. The evaluation result of TEST2. A comparison between manual quality level evaluation and GoogLeNet quality level evaluation.

	GoogLeNet Evaluation Quality Level			
Manual Evaluation Quality Level	**Bad**	**Moderate**	**Good**	**Total**
Bad	0	0	0	0
Moderate	25	0	0	25
Good	15	0	0	15
Total	40	0	0	40

5 Discussion

5.1 GoogLeNet performance

The results of this study imply the bad performance of the GoogLeNet model, which not only because of continuously large loss function values but also all test images are classified as bad quality. These results reveal that the current GoogLeNet model cannot tackle the text placement evaluation task, where it just classifies all inputs as bad quality in the results so far.

On the other hand, training accuracy, validation accuracy and classification accuracy are lower than expected. Validation accuracy and classification accuracy are close to 1/3, which implies that the classification result is randomly generated from three choices. There is a huge difference between the achieved accuracy in other successful studies and this study. As presented in section 2.5.2, the lowest accuracy among all those applications is larger than 0.7, and most of their mean accuracies vary between 0.9 and 1.0.

However, the current failure does not imply that GoogLeNet is not applicable to text placement evaluation. Although the training accuracy is not greater than 0.5 with 800 iterations, its fitting line has a significant upward trend which suggests a potential increase in accuracy if there are more iterations than the current. GoogLeNet learns and discards knowledge from the datasets during the training process where could be seen from the wave of training accuracy. Along with iterations, more knowledge is acquired than that of discarded, which explains the upward trend of training accuracy. This uptrend could be achieved by improving and perfecting the model settings as well as the datasets.

5.2 Encountered issues

During the training process, there is a large number of issues affecting the GoogLeNet model from many perspectives. These encountered issues could be solved to reduce or improve the model in different scenarios, which could contribute to further studies.

5.2.1 Capacity issue

One of the fundamental and unchangeable factors that affect the model training process in this project is the hardware capacity issue. This issue restricts maximum iteration and batch size. In this study,

we only conduct 800 iterations training with a batch size of 8. Maximum iteration cannot exceed 800 times after many tests. This result is a consequence of insufficient graphics memory. If there could be more iterations in training, the training accuracy is believed to increase and converge to a satisfactory value. For example, Singla et al. (2016) trained model to achieve food classification and categorization by 10,000 and 40,000 iterations, respectively. Their accuracy has a general trend that increases along with the iterations and converges to a certain high accuracy value in the end. Besides, the batch size is also restricted by the memory since a larger number of processed images at one iteration needs larger memory. The GPU used, GTX960M with 2GB memory, cannot accommodate batch size over 8.

Introducing multiple high-end GPUs with large memory is a direct and straightforward solution of the capacity issue. GoogLeNet conducts image recognition tasks and thus has high graphics processing requirements. Common GPUs are not qualified, but only those GPUs that support NVIDIA CUDA toolkit are qualified. NVIDIA CUDA is a toolkit that provides the environment to accelerating GPU for a high-performance application. It is an exclusive computing platform of NVIDIA which means only NVIDIA GPUs are CUDA-enabled.

5.2.2 Dataset issues

The training dataset is the sole learning source for GoogLeNet model; therefore, dataset related issues have a much stronger influence on model performance than capacity issues. Data size, data quality and data content are discussed in this section.

First of all, the size of the dataset matters. When facing a complex problem, a small dataset may result in unfitting or overfitting. Overlearning image features that caused by high feature similarity in images in the dataset could lead to overfitting. On the contrary, small feature similarity could contribute to unfitting due to the inability to understand features. In addition, dataset needs to be not only quantitatively rich but also dimensionally rich. There are no specific requirements on the size of the dataset, while the model's good quality is proved by having as many high-quality datasets as possible. Facebook created a face-recognition system; for example, using more than four million portrait images (Hurwitz et al., 2015). GoogLeNet participated in ILSVRC14 using 1.2 million images for training (Szegedy et al., 2015). However, such a large amount of work is impossible to

achieve with manual quality level classification since the number of images classified is much larger than the number of images used to ensure the balance between and within text placement quality levels.

Our project is different from the above examples because of the characteristics of map data. Firstly, the size of map datasets is limited. The number of publicly available map images is much smaller than that of traditional objects (e.g., face images and animals). Secondly, different types of maps (e.g., topographic maps, thematic maps, and city maps) have different cartographic rules and designs, which reduces the commonality of data. Finally, it is hard to manipulate data in a certain format. Vector data are easily operated simplification or rasterization. For raster data, it is a big challenge to change its content or design.

Dataset quality issues are also a concern. The quality of a dataset determines the quality of the model. It is not possible to construct a high-quality model from low-quality learning materials. For example, unclear differences between two quality level may confuse the GoogLeNet model.

Image content is highly related to data quality. The content of each image in this project is controlled within a 256 x 256-meter range at 1:2250 scale. A large value of loss function implies that each image in the dataset contains too much information which makes it difficult to identify features. This issue increases the difficulty of model training and weakens the quality when using small dataset. However, this problem cannot be simply solved by narrowing the range in images. Reducing the range of a single image increases the number of images a map can segment. Therefore, it is worth considering how to choose a delicate balance between image size and the total amount. Besides, discrete characteristics of map images also increase the amount of information for a model to learn. GoogLeNet can only identify learned features but can do nothing about the unknown features. For example, when different street blocks that are separated by the road network change in shape or size, they are unknown features to GoogLeNet (Fig. 5.1). Thirdly, the essence of the proposed quality measurement schema is that one bad label results in one image with bad quality. This fact could make the training process complicated. A slight difference in one label among multiple labels in a single image can lead to entirely different quality levels. This phenomenon hides the explicit features in the

image and makes it hard for GoogLeNet to understand. Besides, the quality measurement schema is qualitative rather than quantitative.

Consequently, subjective preferences have an impact on the results in manual classification. Some oversight in this process is bound to affect the model performance. Therefore, reducing the amount of information in the image is worthy of consideration to solve the abovementioned issues.

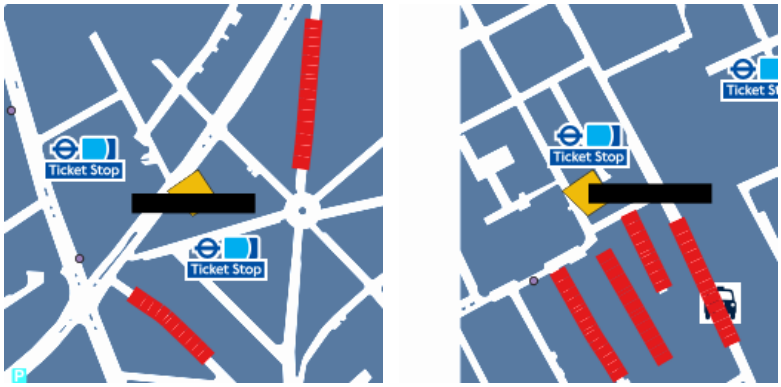


Fig. 5.1. A comparison of different street blocks features. It could be found that different road features create different street block features.

In addition, the data resolution (256 x 256 pixels) is compressed to 224 x 224 pixels at the initial step in GoogLeNet for the subsequent processes. Whether this compress influences the training is unknown since the Szegedy et al. (2015) also trained model with substantial images with diverse resolutions. The discrete essence of a map image, however, might be affected.

There are some solutions to dataset issues. As for data quality issues, it is necessary to automate the dataset classification process. Automating this process can not only avoid the mistakes in manual classification but also can quantify the quality measurement schema to ensure data quality. The data resolution should also be the same as model input resolution to avoid problems. Concerning data amount issues, several general approaches are taken into consideration. One method is data augmentation which can enhance the dataset from amount perspective and dimension perspective. Operations such as mirroring, rotation, scaling, and flipping are widely adopted. The other way is reducing the amount of content in each image.

5.2.3 Model issues

GoogLeNet has substantial parameters for training network, and parameter settings changes have strong influences on the model itself. Some parameters are especially important in the training process and need to be discussed.

The first influential parameter is the learning rate. Fig 5.2 shows the relationship between loss function and learning rate under different scenarios. $L(\theta)$ is a simple example of the loss function of parameter θ (Jordan, 2018). The value of $L(\theta)$ indicates the amount of error in the model; therefore, the smallest value is, the better the model's performance is. If the learning rate is set too low, the training will progress slowly because of slight adjustment of network weight parameters (Fig. 5.2). The loss function may decrease but at a very slow pace. However, if the learning rate is set too high, it may lead to undesirable consequences in the loss function. Loss function may fluctuate greatly or even diverge near the lowest point (Jordan, 2018). Therefore, the optimal learning rate can achieve the fastest loss function decline. The optimal learning rate depends on the specific model structure and dataset. In this project, 0.001 is found to be an appropriate learning rate after several attempts with 0.1, 0.01, and 0.0001. High learning rates 0.1 and 0.01 result in terrible learning performance because the model cannot give an accuracy for its classification result, and small learning rate 0.0001 leads to incomplete learning.

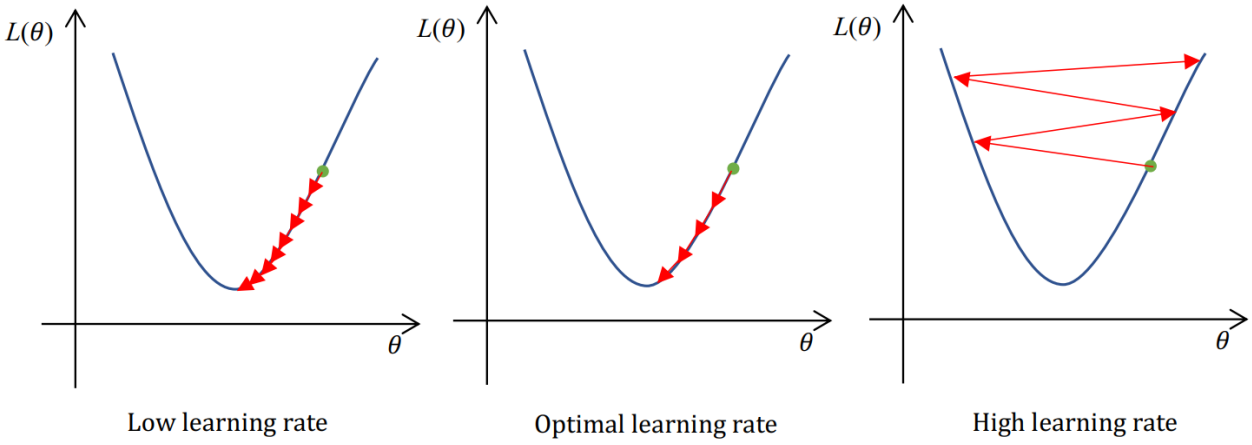


Fig. 5.2. Different learning rate choices and corresponding consequences in GoogLeNet training process (Modified from Jordan (2018)). $L(\theta)$ is a loss function to θ, and the minimum value of $L(\theta)$ is the optimal value; the green dot is the current loss function value; red arrows refer to the improvement during iterations.

When the constant learning rate provides a moderate result, the model performance can be improved by adjusting to the optimal learning rate. The widely accepted method to search the best value is learning rate annealing, which gradually decreases the learning rate from a high value to a

low value during training (Jordan, 2018). The most common implementation of this approach is step decay. The learning rate drops by a fixed percentage after a specific iteration of training. Besides, the periodic learning rate table proposed by Smith can also complete the same task (Smith, 2017).

The second topic is the optimizer algorithm. GD updates and adjusts model parameters to minimize loss function by decreasing the model parameters along the direction of the gradient. However, this algorithm has two disadvantages, namely, slow training speed and locally optimal solution. On the one hand, each training iteration requires a new calculation to adjust the direction of the next iteration. Model parameters are updated after each input, and all inputs are traversed at each iteration. It delays the training process and takes a long time to convergence, especially for large dataset. On the other hand, it is easy to trap into locally optimal solution since the search of the gradient within a limited scope obscures the global vision. When it found a locally optimal solution, the search stops because of the 0 gradients at the saddle point. Therefore, the model parameters are no longer updated.

More advanced optimizer algorithms could be candidates for better performance. For example, momentum optimizer is an advance based on GD optimizer, and its improvement is the ability to accelerate the gradient descent. The main idea is to introduce a momentum to accumulate historical gradient information to accelerate gradient descent (Duda, 2019). Adaptive learning rate optimizer algorithms are also popular, for example, AdaGrad algorithm. It independently adapts to the learning rate of all model parameters and scales each parameter inversely to the square root of the sum of the historical averages of all its gradients (Goodfellow, Bengio, & Courville, 2016). The parameters with the large gradient of the loss function have a large learning rate for a fast descent, while the parameters with small gradient have a small learning rate.

However, there is no consensus on which optimizer is best. Wierstra et al. (2014) showed a comparison of many optimizer algorithms given a large number of learning tasks. The results do not significantly support any optimizer. The choice of optimizer should be tested and considered with specific problems.

5.3 Limitations and perspective

Some limitations are impeding the study, and further works are expected for improvement. Firstly, the number of model training iteration is not enough. Due to hardware capacity limitations, we only perform 800 times of training. However, it is common to train such CNN over 10,000 times. The accuracy and loss will change and finally converge during iterations. More iterations are expected for GoogLeNet's better performance. Secondly, model training is a time-consuming process. For example, GoogLeNet was trained with multiple high-end GPUs within one week (Szegedy et al., 2015). For poor hardware capacity, the time will be longer. Since more iterations than this study are required for GoogLeNet, a high-performance machine is necessary for further studies. Thirdly, manually preparing training dataset is a cumbersome work. The workload of the current preparation method is similar to that of manual text placement. Automating the dataset classification process must be implemented for further studies, which could thus improve the dataset quality and amount. Fourthly, too much content in the image limits the model training performance. In a single image, there are multiple labels in an image and one bad label results in a bad image. One bad label among multiple labels in an image leads to the image will be classified as bad quality, even though the others are of good quality. This fact increases the difficulty for GoogLeNet to identify features in the image. Diminishing the number of labels in an image could make the features explicit and easy for GoogLeNet to learn. Fifthly, the multi-classification task influences the performance. Classifying images into multiple classes is harder than into binary classes for GoogLeNet. Singla et al. (2016) successfully classified food and non-food objects with 90% accuracy, while only achieved food categorization with 70% accuracy. Such a result also evidences changing the quality measurement schema to two levels (i.e., good and bad) might reduce the difficulty and increase the accuracy for GoogLeNet.

6 Conclusion

This master thesis proposes, implements, and assesses an AI method for text placement evaluation which is, to the author's knowledge, novel use of AI in cartography. Simplified and rasterized maps based on formulated cartographic rules are used to train a GoogLeNet model. This model is tested by some test images from Cederholm (2020). The results suggest that the current GoogLeNet model is not qualified for our aim. Training accuracy, validation accuracy and the value of loss function are far from satisfactory. All test images are classified into bad quality, which is proved to be an unsuccessful classification.

However, the unsuccess in GoogLeNet is highly limited by some solvable factors. Firstly, hardware capacity imposes restrictions on training process which weakens the performance, including maximum iterations, training speed and batch size. Secondly, dataset quality and size limits the training process of the model. Thirdly, model parameters have strong impacts on the model.

Although the experiment in this master thesis does not achieve success, this method still worth further exploration and could be improved by solving the abovementioned issues. High-performance hardware is expected for improvement, and automated implementation of dataset classification is required for better quality and larger data amount. Diminishing the number of labels in an image and quality levels of labels for better learning performance are also taken into consideration. Besides, more tests on parameter settings should be considered to achieve better performance, given the specific purpose and datasets.

References

Alinhac, G. (1962). *Cartographie Théorique et Technique*, chapter IV. Institut Géographique National, Paris.

Almeida, L. G., Backović, M., Cliche, M., Lee, S. J., & Perelstein, M. (2015). Playing tag with ANN: boosted top identification with pattern recognition. *Journal of High Energy Physics*, *2015*(7), 86.

Ambikapathi, A., Chan, T. H., Lin, C. H., & Chi, C. Y. (2013, June). Convex geometry based outlier-insensitive estimation of number of endmembers in hyperspectral images. In *2013 5th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)* (pp. 1-4). IEEE.

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, *39*(12), 2481-2495.

Brewer, C. A., Stanislawski, L. V., Buttenfield, B. P., Sparks, K. A., McGilloway, J., & Howard, M. A. (2013). Automated thinning of road networks and road labels for multiscale design of The National Map of the United States. *Cartography and Geographic Information Science*, *40*(4), 259-270.

Cederholm, P. (2020). Producing automatic label placement for city maps with the labelling library PAL. *Student thesis series INES*.

Chen, Y., Lin, Z., Zhao, X., Wang, G., & Gu, Y. (2014). Deep learning-based classification of hyperspectral data. *IEEE Journal of Selected topics in applied earth observations and remote sensing*, *7*(6), 2094-2107.

Chirié, F. (2000). Automated name placement with high cartographic quality: City street maps. *Cartography and Geographic information science*, *27*(2), 101-110.

Christensen, J., Marks, J., & Shieber, S. (1994). Placing text labels on maps and diagrams. *Graphic Gems*, *4*, 497-504.

Christensen, J., Marks, J., & Shieber, S. (1995). An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics (TOG)*, *14*(3), 203-232.

Cook, A. C. and Jones, C. B. (1990). A Prolog interface to a cartographic database for name placement. *Proceedings of the International Symposium on Spatial Data Handling*: 701-710.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., ... & Le, Q. V. (2012). Large scale distributed deep networks. In *Advances in neural information processing systems* (pp. 1223-1231).

Dent, B.D. (1996). *Cartography*, chapter 14. Wm. C. Brown Publishers.

Duda, J. (2019). SGD momentum optimizer with step estimation by online parabola model. *arXiv preprint arXiv:1907.07063*.

Durkin, J. (1997). 4 Expert System. *The Handbook of Applied Expert Systems*, *4*, 15.

Ebinger, L. R., & Goulette, A. M. (1990). Noninteractive automated names placement for the 1990 decennial census. *Cartography and Geographic Information Systems*, *17*(1), 69-78.

Edmondson, S., Christensen, J., Marks, J., & Shieber, S. (1996). A general cartographic labelling algorithm. *Cartographica: The International Journal for Geographic Information and Geovisualization*, *33*(4), 13-24.

Feng, Y., Thiemann, F., & Sester, M. (2019). Learning cartographic building generalization with deep convolutional neural networks. *ISPRS International Journal of Geo-Information*, *8*(6), 258.

Freeman, H., & Ahn, J. (1984). AUTONAP-an expert system for automatic map name placement. In *Proc. 1st International Symp. on Spatial Data Handling* (pp. 544-569).

Freeman, H. (1988). An expert system for the automatic placement of names on a geographic map. *Information Sciences*, *45*(3), 367-378.

Freeman, H. (2004). Label-EZ™--Software for Automated Cartographic Text Placement.

Freeman, H. (2005). Automated cartographic text placement. *Pattern Recognition Letters*, *26*(3), 287-297.

Fu, Z., Yang, Y., Gao, X. (2016). An optimization algorithm for multi-characteristic road network matching. *Acta Geodaetica et Cartographica Sinica*, 45(05): 608-615.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Gualtieri, J. A., & Chettri, S. (2000, July). Support vector machines for classification of hyperspectral data. In *IGARSS 2000. IEEE 2000 International Geoscience and Remote Sensing Symposium. Taking the Pulse of the Planet: The Role of Remote Sensing in Managing the Environment. Proceedings (Cat. No. 00CH37120)* (Vol. 2, pp. 813-815). IEEE.

Harrie, L., Stigmar, H., & Djordjevic, M. (2015). Analytical estimation of map readability. *ISPRS International Journal of Geo-Information*, *4*(2), 418-446.

Haunert, J. H., & Wolff, A. (2017). Beyond maximum independent set: an extended integer programming formulation for point labeling. *ISPRS International Journal of Geo-Information*, *6*(11), 342.

He, H., Qian, H., Xie, L., Duan, P. (2018). Interchange recognition method based on CNN. *Acta Geodaetica et Cartographica Sinica*, 47(03): 385-395.

Hirsch, S. A. (1982). An algorithm for automatic name placement around point data. *The American Cartographer*, *9*(1), 5-17.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *6*(02), 107-116.

Huffman, F. T., & Cromley, R. G. (2002). An automated multi-criteria cartographic aid for point annotation. *The Cartographic Journal*, *39*(1), 51-64.

Hurwitz, J., Kaufman, M., Bowles, A., Nugent, A., Kobielus, J. G., & Kowolenko, M. D. (2015). *Cognitive computing and big data analytics*. Indianapolis: Wiley.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Imhof, E. (1975). Positioning names on maps. *The American Cartographer*, *2*(2), 128-144.

Johnson, D. S., & Basoglu, U. (1989). The use of artificial intelligence in the automated placement of cartographic names. In *Proceedings Auto-Carto* (Vol. 9).

Jones, C. B. (1989). Cartographic name placement with Prolog. *IEEE Computer Graphics and Applications*, (5), 36-47.

Jones, C. B. (1993). Placenames, cartographic generalisation and deductive databases. *Formalising Cartographic Knowledge, Buffalo*.

Jordan, J. (2018). Setting the learning rate of your neural network. Retrieved from https://www.jeremyjordan.me/nn-learning-rate/ on May 17th 2020.

Kameda, T., & Imai, K. (2003). Map label placement for points and curves. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, *86*(4), 835-840.

Kern, J. P., & Brewer, C. A. (2008). Automation and the map label placement problem: A comparison of two GIS implementations of label placement. *Cartographic Perspectives*, (60), 22-45.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Lee, J., Jang, H., Yang, J., & Yu, K. (2017). Machine learning classification of buildings for map generalization. *ISPRS International Journal of Geo-Information*, *6*(10), 309.

Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., & Quillen, D. (2018). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, *37*(4-5), 421-436.

Lin, Y., Cai, Y., Gong, Y., Kang, M., & Li, L. (2019). Extracting urban landmarks from geographical datasets using a random forests classifier. *International Journal of Geographical Information Science*, *33*(12), 2406-2423.

Ma, L. (2017). Features extraction of buildings and generalization using deep learning. In *Proceedings of 28th International Cartographic Conference, Washington, DC, USA.*

Mackaness, W. A., & Ruas, A. (2007). Evaluation in the map generalisation process. In *Generalisation of geographic information* (pp. 89-111). Elsevier Science BV.

Mower, J. E. (1993). Automated feature and name placement on parallel computers. *Cartography and Geographic Information Systems*, *20*(2), 69-82.

Revell, P., Regnauld, N., & Bulbrooke, G. (2011, July). OS Vectormap District: Automated Generalisation, Text Placement and Conflation in Support of Making Public Data Public. In *25th International Cartographic Conference* (pp. 3-8).

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Berg, A. C. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*(3), 211-252.

Russell, S., & Norvig, P. (1995). Artificial intelligence: A modern approach prentice-hall. *Englewood cliffs, NJ*, *26*.

Salavati, P., & Mohammadi, H. M. (2018, October). Obstacle detection using GoogleNet. In *2018 8th International Conference on Computer and Knowledge Engineering (ICCKE)* (pp. 326-332). IEEE.

Sester, M., Feng, Y., & Thiemann, F. (2018). Building generalization using deep learning. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLII-4 (2018)*, *42*, 565-572.

Singla, A., Yuan, L., & Ebrahimi, T. (2016, October). Food/non-food image classification and food categorization using pre-trained googlenet model. In *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management* (pp. 3-11).

Slocum, T. A., McMaster, R. B., Kessler, F. C. och Howard, H. H. (2005). Thematic Cartography and Geovisulization, Second edition. Upper Saddle River, NJ, USA: Pearson Prentice Hall. ISBN 9780132298346.

Smith, L. N. (2017). Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 464-472). IEEE.

Steiniger, S., Lange, T., Burghardt, D., & Weibel, R. (2008). An approach for the classification of urban building structures based on discriminant analysis techniques. *Transactions in GIS*, *12*(1), 31-59.

Stoter, J., Zhang, X., Stigmar, H., & Harrie, L. (2014). Evaluation in generalisation. In *Abstracting Geographic Information in a Data Rich World* (pp. 259-297). Springer, Cham.

Strijk, T., & Van Kreveld, M. (2002). Practical extensions of point labeling in the slider model. *GeoInformatica*, *6*(2), 181-197.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017, February). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*.

Touya, G., Zhang, X., & Lokhat, I. (2019). Is deep learning the new agent for map generalization?. *International Journal of Cartography*, *5*(2-3), 142-157.

van Dijk, S. (2001). *Genetic algorithms for map labeling*. Utrecht University, Netherlands.

van Dijk, S., Thierens, D., & Berg, M. D. (2004). On the design and analysis of competent selecto-recombinative GAs. *Evolutionary computation*, *12*(2), 243-267.

van Dijk, S., van Kreveld, M., Strijk, T., & Wolff, A. (2002). Towards an evaluation of quality for names placement methods. *International Journal of Geographical Information Science*, *16*(7), 641-661.

van Kreveld, M., Strijk, T., & Wolff, A. (1999). Point labeling with sliding labels. *Computational Geometry*, *13*(1), 21-47.

Verner, O. V., Wainwright, R. L., & Schoenefeld, D. A. (1997). Placing text labels on maps and diagrams using genetic algorithms with masking. *INFORMS Journal on Computing*, *9*(3), 266-275.

Wierstra, D., Schaul, T., Glasmachers, T., Sun, Y., Peters, J., & Schmidhuber, J. (2014). Natural evolution strategies. *The Journal of Machine Learning Research*, *15*(1), 949-980.

Wu, R., Yan, S., Shan, Y., Dang, Q., & Sun, G. (2015). Deep image: Scaling up image recognition. *arXiv preprint arXiv:1501.02876*, *7*(8).

Yamamoto, M., & Lorena, L. A. (2005). A constructive genetic approach to point-feature cartographic label placement. In *Metaheuristics: Progress as real problem solvers* (pp. 287-302). Springer, Boston, MA.

Yanase, J., & Triantaphyllou, E. (2019). A systematic survey of computer-aided diagnosis in medicine: Past and present developments. *Expert Systems with Applications*, 112821.

Yoeli, P. (1972). The logic of automated map lettering. *The Cartographic Journal*, *9*(2), 99-108.

Zhang, Q., Gao, W., Su, S., Weng, M., & Cai, Z. (2017). Biophysical and socioeconomic determinants of tea expansion: Apportioning their relative importance for sustainable land use policy. *Land Use Policy*, *68*, 438-447.

Zhang, Q., & Harrie, L. (2006a). A real-time method of placing text and icon labels simultaneously. *Cartography and Geographic Information Science*, *33*(1), 53-64.

Zhang, Q., & Harrie, L. (2006b). Real-time map labelling for mobile applications. *Computers, environment and urban systems*, *30*(6), 773-783.

Zhang, X., Stoter, J., Ai, T., Kraak, M. J., & Molenaar, M. (2013). Automated evaluation of building alignments in generalized maps. *International journal of geographical information science*, *27*(8), 1550-1571.

Zhao, W., & Du, S. (2016). Spectral–spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach. *IEEE Transactions on Geoscience and Remote Sensing*, *54*(8), 4544-4554.

Zhao, H., Gallo, O., Frosio, I., & Kautz, J. (2016). Loss functions for image restoration with neural networks. *IEEE Transactions on computational imaging*, *3*(1), 47-57.

Zhong, Z., Jin, L., & Xie, Z. (2015, August). High performance offline handwritten chinese character recognition using googlenet and directional feature maps. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)* (pp. 846-850). IEEE.

Zhou, Q., & Li, Z. (2017). A comparative study of various supervised learning approaches to selective omission in a road network. *The Cartographic Journal*, *54*(3), 254-264.

Zoraster, S. (1997). Practical results using simulated annealing for point feature label placement. *Cartography and Geographic Information Systems*, *24*(4), 228-238.

Appendix A

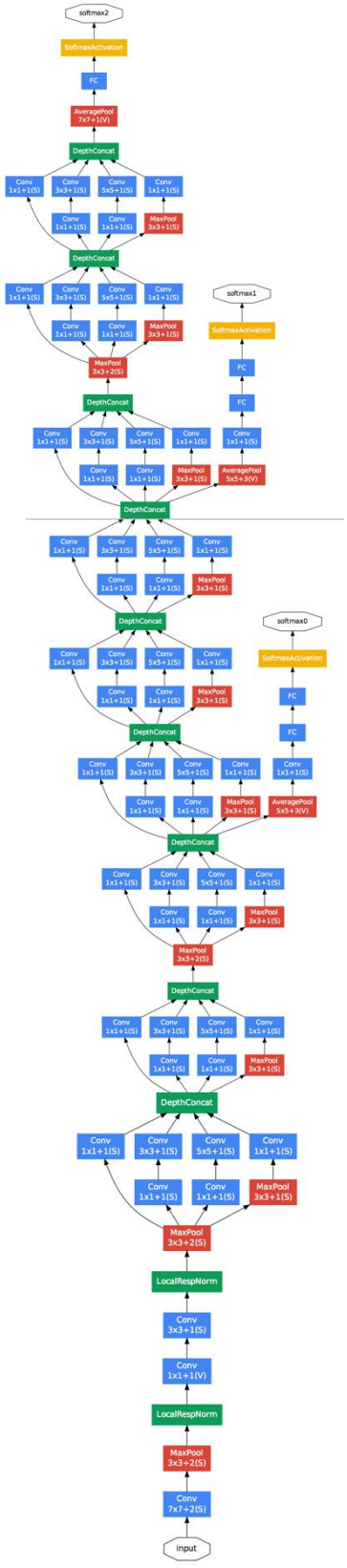


Fig. A. GoogLeNet structure (re-produced from Szegedy et al. (2015) with permission from publisher IEEE). **© Copyright [2014] IEEE**

Appendix B

```
# Export
# Get full extent
ext = canvas.fullExtent()
xmin = ext.xMinimum()
ymin = ext.yMinimum()
xmax = ext.xMaximum()
ymax = ext.yMaximum()

# Define output resolution
resolution = 256

# Calculate how many images
row = ceil((xmax - xmin)/resolution)
column = ceil((ymax - ymin)/resolution)

# Define rasterization function
def rasterize(resolution, row, column):
    # Create Image
    img = []
    color = QColor(255, 255, 255, 100) # Last argument represents transparency

    for i in range(1, (row+1)*(column+1)):
        img.append(QImage(QSize(resolution, resolution), QImage.Format_ARGB32_Premultiplied))

    for i in range(1, (row+1)*(column+1)):
        img[i-1].fill(color.rgba())

    # Create map settings
    ms = QgsMapSettings()
    ms.setBackgroundColor(color)

    # Set layers to render
    ms.setLayers([taxi, carPark, policeSt, post, ticket, info, roads, buildings, pavement])

    # Create output folder
    newpath = '/Users/weilai/Desktop/QGIS_test/' + 'Rasterization Output' # change the first directory to the place you want to save
    if not os.path.exists(newpath):
        os.makedirs(newpath)

    for n in range(1, column+1):
        for m in range(1, row+1):
            # Set extent
            rect = QgsRectangle(xmin+(m-1)*resolution, ymax-n*resolution, xmin+m*resolution, ymax-(n-1)*resolution)
            ms.setExtent(rect)

            # Set output size
            ms.setOutputSize(img[0].size())

            # Create painter
            p = QPainter()
            p.begin(img[(n-1)*int(row)+m])
            p.setRenderHint(QPainter.Antialiasing)

            # Setup qgis map renderer
            render = QgsMapRendererCustomPainterJob(ms, p)
            render.start()
            render.waitForFinished()
            p.end()

            # Save the image (Change the path)
            img[(n-1)*int(row)+m].save(str(newpath) + '/R' + str(m) + 'C' + str(n) +'.png')
    return;

# Export the map
rasterize(resolution, row, column)
```

Fig. B. Codes for rasterization process: the map is rasterized as images with 256 x 256 image resolution and 1 pixel refers to 1 meter.

Appendix C

```
# Get feature
for num in range(len(RandomPoint)):
    RP = RandomPoint.getFeature(num)
    AP = AreaPoly.getFeature(0)

    #polygons geometry
    AP_mulpoly = AP.geometry().asMultiPolygon() #polygon1

    #centroid coordinates
    RP_centroid = RP.geometry().asPoint() #centroid1
    AP_centroid = AP.geometry().centroid().asPoint() #centroid2

    pol_t = AP_mulpoly
    n = len(AP_mulpoly[0][0])

    #calculating displacement based in centroids
    px = RP_centroid.x() - AP_centroid.x()
    py = RP_centroid.y() - AP_centroid.y()

    #translation as an affine transformation
    for i in range(n):
        pol_t[0][0][i].setX(AP_mulpoly[0][0][i].x() + px)
        pol_t[0][0][i].setY(AP_mulpoly[0][0][i].y() + py)

    #creating a memory layer for displaced polygon (building1)
    crs = RandomPoint.crs()
    epsg = crs.postgisSrid()

    uri = "MultiPolygon?crs=epsg:" + str(epsg) + "&field=id:integer&field=name""&index=yes"

    mem_layer = QgsVectorLayer(uri, "building1_displaced", "memory")

    project.addMapLayer(mem_layer)

    mem_layer.startEditing()

    feat = QgsFeature()

    #Set geometry
    feat.setGeometry(QgsGeometry.fromMultiPolygonXY(pol_t))

    #set attributes values
    feat.setAttributes([1,'Rathbone Hotel'])
    mem_layer.addFeature(feat)

    #stop editing and save changes
    mem_layer.commitChanges()
```

Fig. C. Codes excerpt for training and validation dataset preparation: randomly placed an area feature across the map extent and extract images with 256 x 256 image resolution (1 pixel refers to 1 meter).