Student thesis series INES nr 579

NGEM01-Final Report

Automatic text placement on maps using deep learning keypoint detection models

Author: Azin Khosravi

Supervisor: Rachid Oucheikh

2022
Department of
Physical Geography and Ecosystem Science
Lund University
Sölvegatan 12
S-223 62 Lund
Sweden

Azin Khosravi Khorashad (2022).

Automatic text placement on maps using deep learning keypoint detection models

Master degree thesis, 30 credits in *Geomatics*

Department of Physical Geography and Ecosystem Science, Lund University

Level: Master of Science (MSc)

Course duration: *January* 2022 until *June* 2022

Automatic text placement on maps using deep learning keypoint detection models

---

Azin Khosravi Khorashad

Master thesis, 30 credits, in *Geomatics*

Supervisor:

Rachid Oucheikh

Dep. of Physical Geography and Ecosystem Science, Lund University

Exam committee:

Micael Runnström

Dep. of Physical Geography and Ecosystem Science, Lund University

Abstract

Labeling the map is one of the most important parts of the cartographic process that requires huge time and energy. It is proven that the automation of the map labeling is a NP-hard problem. There have been many research studies that tried to solve it such as rule-based methods, metaheuristics, integer programming. However, the results achieved so far are not satisfactory and requires much manual processing. In fact, many cartographic rules are hard to quantify or formulate as objective function or to include as a constraint. The purpose of this master thesis was to find a new way for text placement and introduce a method based on keypoint detection using deep learning. For this goal, a workflow is designed and consists of rasterization of the manually labelled data, followed by data augmentation and shaping. Then, based on the experiments, the architecture and the parameters of the Stacked Hourglass Networks are determined based on the evaluated performance. The best-found architectures achieved an accuracy of 60.7%. Furthermore, with the use of an attention mechanism, the model can achieve an accuracy of 63.1%.

Keywords: machine learning, Stacked hourglass networks, attention mechanism, CNN.

Acknowledgment

I would like to thank my supervisor Rachid Oucheikh for all his help, support, and patience during the thesis work. Without his assistance and dedicated involvement in every step through the process, this thesis would have never been accomplished. I would like also to thank Prof. Micael Runnström for his professional and constructive comments. Many thanks to Åsa Nilsson from T-Kartor for their technical help and data.

I am also grateful to my best classmates and friends Soraya and Kui for their help and all support during my master's course. Also, my family encourages and motivations.

Azin Khosravi Khorashad, May 2020

Contents

figures

Abbreviations

CNN	Convolutional Neural Networks
GAN	Generative Adversarial Networks
GIS	Geographical Information System
HGN	Hourglass Network
LSTM	Long Short-Term Memory networks
NLP	Neural Machine Translation System
PCK	Percentage of Correct Key-point
PCP	Percentage of Correctly estimated body Parts
PDJ	Percentage of Detected Joints
RNN	Recurrent Neural Networks

1. Introduction

Maps convey spatial geographic information to a reader, and they are considered among the most information-dense graphical artifacts designed by humans. In the history of mapping, science art has always been interconnected. To create unique map aesthetics containing geospatial information, cartographers employ a set of design criteria that integrate human creativity and experience.

Maps are an important medium of communication and very crucial for navigation and decision-making. Thus, they should be clear and legible. The effectiveness and functionality of a map depend hugely on the way it is annotated, and high-quality label placement is a key element in cartographic representation (Imhof 1972; Bertin 1983; Robinson and others 1995). Manual map labelling is a tedious and time-consuming task. Consequently, the new wave in research is to elaborate new techniques for semi-automatic or fully automatic solutions for the tasks required in the production of the maps and, particularly, the labelling task. So, technologies have been developing to blur the line between humans and machines (Kang, 2020).

The map labeling problem includes four questions: 1) issues concerning toponyms (e.g., exonyms vs. endonyms, multilingual labelling), 2) toponym selection (quantity and type), the choice of labels to display and their classification, 3) typeface handling (e.g., font, layout, font size), and 4) geometric placement of labels (Rylov & Reimer,2014). In the automation of labelling, researchers focused on the geometric placement assuming that the first three issues are solved and require only parameterization. Thus, for the geometric issue, three major labelling types are identified: area labelling (e.g., buildings, lakes, parks), line labelling (e.g., roads, rivers, boundaries), and point labelling (e.g., bus stop, mountain peaks). The rules and the optimal placement of labels for each of these types are different (Christensen & Stuart,1992). Consequently, it is challenging to quantify their rules through a unique measure. The general requirements that the labels should follow are to not overlap with each other and to have a clear association with the objects or the features labeled (Chamra, 2017), but the specific rules are various and depend on the nature of the map, its objective and the specification given by the user. To have a clear idea about the rules, some of them will be explained in the second section. since it is one of the most time-consuming and difficult tasks.

To label the maps properly and easily, many methods have been proposed to automate the application of the specified rules. The map described in this thesis is a general-reference map and this thesis will discuss how to implement accurate label geometries in automated methods to have a good connection between text labels and icons. For this aim, a deep learning approach based on stacked hourglass networks is proposed as a solution.

1.1. Background

Map labeling is an essential task in the field of cartography and geographic information systems. The labeling task aims to place text or graphic labels on maps while avoiding overlaps, improving map visualization, and respecting predefined rules. These labels must be assigned to the corresponding map features (Chamra, 2017) and should be readable, legible, and aesthetically magnificent. The problem of map labeling is known to be NP-hard and finding optimal positions of all map labels, even for the simplest maps, is highly computationally expensive. In addition, manual map labeling could consume up to 50% of the total map production time (Morrison, 1980).

To reduce this workload and ensure high-quality maps, several studies have been conducted in the last three decenies on automatic map labeling (Wolff and Strijk, 2009, Oeltze-Jafra et al., 2014).

Most of the elaborated techniques have adopted rule-based algorithms. This kind of method requires careful formulation of certain rules and criteria and should also design specific objective functions. The constrained objective functions are then optimized using metaheuristics such as mathematical programming, genetic algorithms, particle optimization, tabu search, and many others. The output of these methods is candidate positions indicating the possible best positions of the labels. These methods are essentially based on the quantification of handcrafted rules found in the cartographic literature and thus are limited in expressing the characteristics of label placement for different features and shapes. Thus, the results are generally not sufficient and require further manual post-processing and fine-tuning.

Hence, recent research contributions aim to automate this process and get rid of heavy rule formulation. In this context, we propose to follow a novel track by using a deep learning technique to tackle the automatic label placement on maps. The motivation of this choice is to digest the complex and implicit characteristics of label placement which cannot be captured using static rules. In addition, deep learning does not require the human design of rules and features. The

ultimate goal is to provide a fully automatic solution as machine learning achieved a high level of accuracy and automation in other fields. Around this time some basic logic for labeling points, line, and are a feature introduced at the tree-step level including selection, layout, and final placement.

Most studies regarding the Label placement topic are about point-feature and line-feature. But Krumpe and Mendel (2020) presented a near-real-time method to automatically label areas with curved labels. They chose the input area as a boundary polygon and computed the labeling for large data sets with their algorithm in just a few seconds.

Kang (2020) examines two important topics in cartography that use machine learning in cartography style transfer and map generalization. A large-scaled tiled map using GIS vector data was the focus. Training generative adversarial networks (GAN), deep neural networks, and convolutional neural networks were used to transfer cartographic knowledge across multiple scales, including stylistic elements and generalization rules. Some studies in this direction are the exploration of deep learning for mountain road generalization using the U-Net network (Courtial et al., 2020).

There is only one study that used machine learning to solve this problem. Li et al., (2020) tackled the automatic label placement of area-feature based on the key-points detection model by developing deep learning. They proposed a stacked hourglass network to produce a heatmap of the position candidate before selecting the best position for the area label.

1.2. Research Aim

Generally, two main types of maps exist. A general-reference map that is intended to tell you where the actual functionality such as the location of mountains, lakes, and buildings is located. Topographic maps are a common type of general-reference map. The other type is thematic maps which emphasize one or more attributes and the patterns of the two-dimensional plane that convey that data (Cederholm, 2020). We are interested in the first type. Our interest in this project is to use machine learning, particularly deep learning, to solve the task of map labelling which we will formulate as a keypoint detection in images. This is motivated by success in many application domains. As stated above machine learning is not extensively explored for this issue and it is used

in cartography mainly for map generalization and style transfer (Touya et al. 2019; Feng, Thiemann and Sester 2019; Courtial et al. 2020).

1.3. Research Questions

In the scope of this study, two research questions has been formulated:

I. What are the pre-processing steps required for the application of deep learning for feature labelling?
II. Using the keypoint detection models, what is the model architecture which helps to achieve the best performance?

To answer the second question, experiments will be performed using a stacked hourglass network and will focus on the ResNets and convolutional layers efficiency and the improvement that the attention mechanism can present.

1.4. Delimitations

The thesis has focused on map labelling and precisely learning the positions where to place labels, called keypoints. So, all other map requirements such as icon placements, text settings such as font or size, etc. have been excluded. The ultimate objective is to learn the locations that will serve as candidate positions for labels and that will be used for further processing and text fitting. In addition, this thesis emphasizes a type of deep keypoints detection model called stacked hourglass networks.

1.5. Outline

In the second section, we will provide the theoretical background of the project by formulating the map labelling problem, and the set of techniques for solving it with a focus on machine learning and keypoint detection method. Then in the third section, we will explain the methodology followed in the project and give details of the implementation. Finally, in the last section, we will present the experiments and show the obtained results.

2. Theoretical background

In this section, we present the context and the theoretical basis for our work. First, we elucidate the problem of map labelling. Then, we categorize the techniques invented to automate this process and we review the most significant of them. Afterward, we focus on the use of machine learning for this task, and we dive into the proposed deep learning method presenting the theoretical computation performed for the training of the deep learning model and its components.

2.1. Cartography and label placement

Feature annotation in general is an important, yet complex task in the domain of spatial data presentation and visualization. Particularly, the information displayed on maps competes for the limited space and their placement requires cartographic rules and space management expertise. Manual label placement is a time-consuming process and highly laborious task. With the emergence of computers, cartographers start to think about the possible practical methods to automate the process of positioning labels and thus cope with the problem of space management on maps.

Automation of label placement started with the work of Imhof and its guidelines and has evolved significantly since the 1970s (Kern & Brewer, 2008) yielding numerous and various research endeavors. Around that time, some basic logic for labelling points, line, and are a feature introduced at the tree-step level including selection, layout, and final placement.

2.1.1 Formulation of the rules

Cartographic rules aim to produce high-quality maps by both fulfilling the quality functions and satisfying the users. The general properties of the map should be summarized as followed:

- Legibility: a label must not overlap another label.
- Association: The relationship between the label and its referent object should be clear and easy to interpret, hence avoid placing labels too close to other objects.

- Map readability: Labels should not overlap other significant features of the map background. If it is necessary to place on top of map objects, they should not cover important features of those objects.
- Aesthetics: The labeling should not reduce the aesthetic map, but it should improve it.

The label placement problem is considered by many researchers as a problem of combinatorial optimization. Therefore, the proposed solutions are based on mathematical optimization algorithms and thus define two components: a discrete search space and an objective function stating the properties and rules explained above. The search space consists of candidate label positions of which the objective quality function tries to measure the goodness with respect to some predefined criteria. The objective function returns as output a numerical score that indicates the quality of labeling.

That is to say, the quality function tries to formulate and imitate the process, employed by a cartographer, for finding a trade-off between various informal and competing requirements for good label combinations.

2.1.2 Types of labelling

Some rules are universal and should be respected by all kinds of features. For example, the labels should be associated with their objects while avoiding overlap with other labels or point features. However, each feature has its own requirements and involves its own challenges. An extensive and comprehensive list of cartographic rules for all label types is detailed by Imhof 1975; Wood 2000; van Dijk 2002; Rylov and Reimer 2015. In the following, we only mention the most important in a compact way.

- ***Point features***

- The label should be placed closely around the corresponding point feature to avoid ambiguity in the relationships between features and their names.
- Placement of the label on the right of the point is preferred over on the left of the point, and a position above the point is prioritized.
- Labels should be placed horizontally as it is more convenient for the users' visual habits.

- Labels should be placed entirely on the water surface or entirely on the land and in the case of coastal settlements, names should be written on the water surface.
- Type arrangement should reflect the classification, importance, and hierarchy of objects.
- 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.

➢ ***Line features***

- A label is placed along with the object and adapts to the object's curviness (Chirié, 2000; Freeman, 2005). However, straight parts are preferable for the sake of readability. For big curvatures, a label could be placed parallel to the tangent line of the feature.
- For long lines, text can be repeated.

➢ ***Area features***

- A label is ideally placed completely within the polygon feature they represent, unless there is not enough space inside the area feature, then text can be wrapped into several lines or if necessary allow the cross of the polygon boundary. The alternative is to place 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 alternatives under space limitations (Freeman, 2005).
- A label should not overlap with other labels or point features. Practically, this is hard to comply with, then the overlap is allowed if it does not impact the map readability.
- A text should be labeled more than one time if the area feature has several disconnected parts.

2.1.3. Map labelling techniques

As mentioned before, since 1970 works of Imhof for map labelling has been progressed significantly. In the next steps, different techniques for map lebelling have been presented. It started with rule-based algorithms then followed by slider-Based algorithms around the 1990s using particularly genetic algorithms, finally recent methods have been focused on force-direction to prevent some conflict of labels and prevent them from not having an interfering together.

- **Rule-Based Label Placement Automation:**

Between 1984 and 1987, an automatic name placement system was developed in Fortran, called AUTOMAP which progressively labels map items. Besides, some rule-based expert systems have been developed and were able to automatically place labels on maps, graphs, or diagrams. (Yoeli 1972; Hirsch 1982; Ahn and Freeman 1984). Afterward, researchers managed to formalize and classify most requirements of good label placement and developed a general function that numerically measures the quality of label placement. However, the label placement problem has been proven to be NP-hard. This is demonstrated in the case of point labeling by Kato and Imai (1988) and Marks and Shieber (1991).

Yoeli (1972) used a depth-first search approach for point labelling and Hirsch (1982) applied a discrete gradient descent method. Those techniques do not achieve high performance, but they were just an initial step in developing more performant labelling algorithms. In 1986 and 1990 Zoraster used a variant of 0-1 integer programming to reduce the PFLP problem. At the same time, other researchers were using exhaustive search algorithms (Ahn and Freeman 1984; Freeman and Ahn 1987; Jones 1989; Cook and Jones 1990; Doerschler and Freeman 1992).
First, all possible placements for each map object are identified. The annotations of polygon features, point features, and line features are then ordered in order depending on the identification of conflicts. However, the method is limited in its flexibility since it lacks retracing, which restricts its performance on dense maps. (Wei L, 2020)

- **Simulated Annealing**

The simulated annealing has been used for the map labeling placement by Christensen, Marks, and Shieber (1994, 1995), Edmondson et al. (1996), and Zoraster (1997). The approach uses an

iterative or recursive strategy to achieve locally optimum label placement, but it does not allow for momentarily inferior label placement to obtain a superior global answer. The National Institute of Standards and Technology (NIST) defines simulated annealing as "a strategy for finding the best remedy to an optimization issue by attempting random variants of the present answer."

Wagner et al. (2001) proposed a label placement strategy that was unaffected by feature type, label size, or shape. Their method used a set of criteria to label as many features as feasible while lowering candidate-label sets for those that remained; it then decreased the number of candidates to one per feature. A comparison of their rules-based method with five other approaches utilizing datasets up to 3000 points revealed that it was like simulated annealing in terms of label placement quantity, but significantly quicker, indicating that it may be used for fast Internet labeling. (kern& brewer, 2008)

- **Slider-Based Label Placement Algorithms:**

This approach is designed around 1990 and focuses on the requirement of limiting point feature label positions to a few fixed locations, allowing continuously sliding labels. The algorithms aimed to optimize the number of points receiving non-overlapping labels. An improved slider algorithm was provided by Kameda and Imai (2003) that separated labels as much as possible within a continuous labeling space for each point or line feature to avoid packing labels so closely that they were difficult to read. With the goal of increasing the number of labels placed, they discovered that continuous labeling spaces allowed for more labels to be placed. In heavily featured areas where there is no labeling space for a given point, the authors used an extra method for labels with leader lines. (kern& brewer, 2008).

- **Force-Directed Label Placement Algorithms**

These algorithms were designed for maximizing the number of labels placed in 2003. The method gained a perfect distribution of the labels in the current space.

Despite these efforts in automating the map labeling, the automation level in production is still low and encounters many challenges in big scales maps and high-density areas. One of the reasons behind this low automation is the lack of adequate methods to solve the labeling challenges and the difficulty of precisely formulating some required rules. In this context, machine learning can

help improve the automation process as it can learn implicit features in the human-labeled data and does not need an explicit statement of the requirements.

- **Genetic Label Placement Algorithms**

The problem of map labeling is also being tackled via genetic algorithms. The genetic algorithm (GA) functions as an iterative technique on a fixed size population or pool of possible solutions. The proposed solutions provide an encoding of the issue into a form comparable to biological systems' chromosomes. These techniques beat existing labeling algorithms, including simulated annealing. In a comparison study, after that a set of design rules has been proposed, the authors compared the performance of simple genetic algorithms and other types of labeling algorithms like simulated annealing, finding that GA scale-up behavior matched that predicted by theoretical models (kern& brewer, 2008).

2.2. Machine learning in cartography

Artificial intelligence (AI) is a branch of science that develops a set of techniques that simulates human intelligence and aims to create intelligent agents able to act and perform tasks autonomously and rationally. In recent decencies, AI knew a quick development, and its techniques were being applied in different disciplines to solve various real issues. Particularly, machine learning which is an area of AI has succeeded and outperformed classical techniques in many tasks such as image recognition (Ohri et al. 2015), image classification (Zhao and Du 2016), and robot technology (Levine et al. 2018). Particularly, vision-based techniques have received increasing attention and are used in a wide range of applications as they are more accurate, flexible, and adaptable to complex and diverse tasks than conventional techniques.

In the wave of this evolution, 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 fields, such as deep learning, evolutionary algorithm, and Bayesian network. In fact, classical machine learning techniques require substantial efforts in feature engineering and achieve generally lower performance for the large datasets than the deep learning

techniques and particularly with the introduction of convolutional neural networks (CNN), Recurrent Neural networks, and other learning mechanisms. In addition, many research efforts have been provided to optimize the network design, increase the performance of learning specific tasks, and solve some technical issues such as overfitting, vanishing gradient problems, and under-specification. This leads to efficient model architectures such as Faster-R-CNN, U-Net, YOLO, SSD, FPN, or Inception (Dhilon and Verma 2020).

Researchers used a random forest classifier to allow 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). Kang (2020) examines two important topics in cartography that integrate AI in map style transfer and map generalization. A large-scale tiled map using GIS vector data was the focus. Training generative adversarial networks (GAN), deep neural networks, and convolutional neural networks were used to transfer cartographic knowledge across multiple scales, including stylistic elements and generalization rules.

Regarding the specific issue of map labelling, some metaheuristics classified as AI techniques are used. As an example, genetic algorithms served in the change of map placement. It was based on selecting and combining building blocks and considered competent (Kern& Brewer, 2008). Krumpe and Mendel (2020) presented a near-real-time method to automatically label areas with curved labels. It is a method that examines the search space more comprehensively and efficiently. It is done by computing the polygon's skeleton. The skeleton is trimmed to eliminate edges that are near the border polygon. Then a set of candidate pathways was identified from the trimmed skeleton to be the graph's longest distinct subpaths. The label support lines are computed, and the label placements are assessed based on these candidates. Li et al. (2020) focused on automatic label placement of area-feature based on a key-point detection model by developing deep learning for the first time. They suggested a stacked hourglass network and producing a heatmap to suggest the best position for the area label.

2.2.1. Key-points detection models

Keypoint detection models aim to both detect objects and localize their keypoints. The keypoints are the interest points that are highlighted in the training data and they are positions or spatial

locations in the multi-dimensional image which reflect what is interesting. The model should learn these keypoints regardless of the image variance or transformations by rotation, shrinkage, translation, distortion, etc. In our case, the key points are the positions of the labels which have specific characteristics regarding the map's features and objects. So, the vector data, which we have access to, will be converted and rasterized to images. In addition, for the training data, the centroids of the manual labels which present the keypoints will be extracted and used as inputs to the model. Briefly, a key-point detection model concurrently detects and localizes certain key points from an image. Then, from many points that the model has found as possible key points the best one should be chosen which has the most accuracy in comparison to other points. So, key-point techniques help to find the candidate positions of labels, which in past was a time-consuming and challenging process. The keypoint detection model that we have chosen to use is the stacked hourglass network.

2.2.2. Stacked hourglasses networks

Stacked hourglass networks (Newell et al., 2017) are a key-point detection model based on deep learning which is originally proposed for human pose estimation. It is a special type of Fully Convolutional Network. Since the encoder-decoder structure makes it look like an hourglass, the name is "hourglass networks".

The original architecture is composed of multiple hourglass modules stacked on top of each other and the output of one is input to the next. Each hourglass module is made up of encoder and decoder parts. The encoder performs a downsampling operation ensured by a combination of different residual blocks, convolution, and max-pooling layers. These layers are designed to lower the resolution of the input image and capture various features. In the second part, upsampling and a combination of features at all scales are performed.

The upsampling is achieved by a kind of interpolation using the nearest neighbour and increases successively the image resolution. The combination of features is obtained using a shortcut connection which passes features of all resolutions from the encoder to the decoder to perform an elementwise addition and thus flows and keeps the information of the adjacent resolutions. The

network structure ensures the learning of contextual and multiscale information as it captures high-level features at an early stage and specific features in later stages through a repeated top-down to bottom-up process (Li Y, et al., 2020). It also has a strong ability to digest the spatial features and preserves the spatial location of features.

Instead of constructing a huge encoder and decoder network, the rationale behind stacking numerous HG (Hourglass) modules is that each HG module will provide a comprehensive heatmap for keypoint and also decrease the computation cost. As a result, the latter HG module can learn from the prior HG module's combined forecasts (li, 2020). Figure 1 shows the stacked hourglass process and its structure.

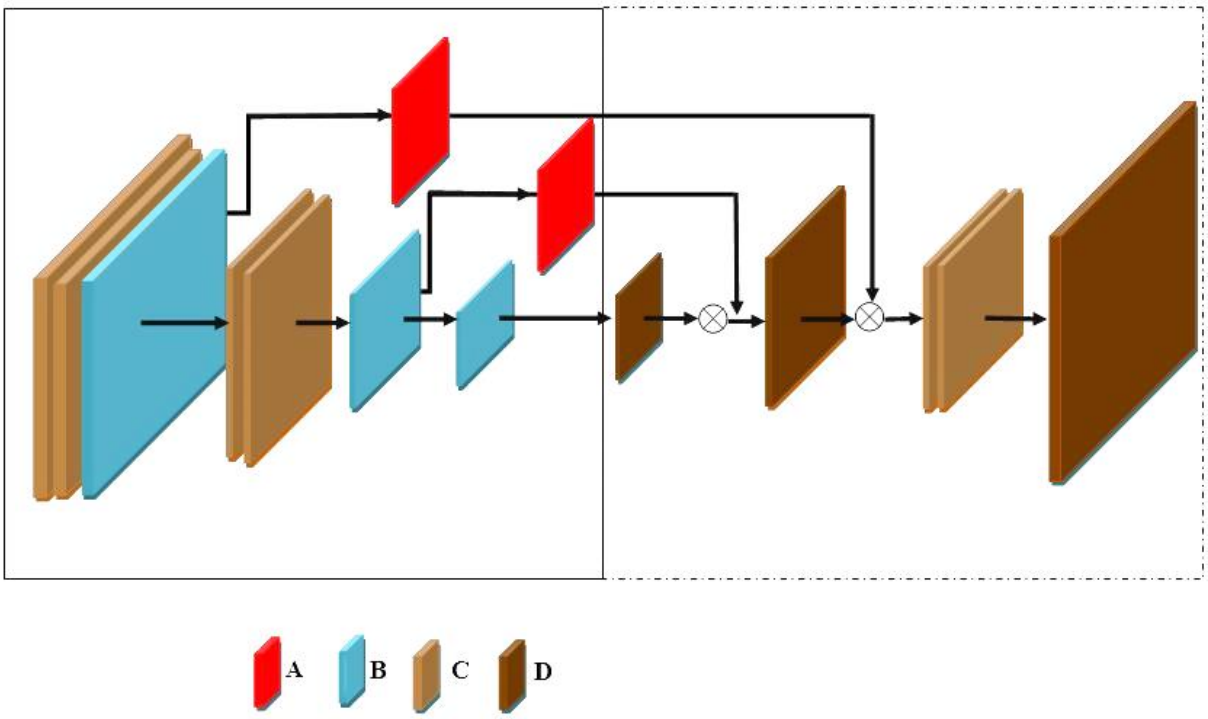


Figure 1, The proposed Stacked Hourglass network. Which contains different layer and residual modules between them

In figure 1, A is residual modules, B is Maxpooling layers, and C and D are convolutional networks. As explained, stacked hourglass networks include convolutional neural networks, residual networks, pooling layers, and some other auxiliary subnetworks. As these networks in

addition to the encoder-decoder mechanism need explanation, we will give details about them in the next paragraphs.

2.2.3. CNN (Convolutional Neural Networks)

Convolutional Neural Networks are the main part of the stacked hourglass network which learns the spatial features of the image and constructs the feature maps and help to decrease the size of the inputs. This is made by applying a kernel that is moving windows that traverses across the input data, performs a dot product with a sub-region of the input data, and outputs a matrix of dot products. In the end, CNN uses multiple filters in each layer. The layers are used to break down and rebuild inputs. They take an input (in our case image) and deconstruct the picture into a feature matrix to extract features from it. The feature matrix is then combined with prior layers that have a stronger spatial comprehension than the feature matrix in other words they have a better sense of where objects are in the image than the feature matrix.

We can learn a lot about the input, particularly what objects are in the image and where they are located, by integrating the feature matrix with early layers in the network that have better spatial knowledge (Ferdinand, 2020). The entire network works as a sophisticated non-linear function that converts inputs into goal variables.

CNNs differ from multi-perceptron networks in that they have specific layers and composing elements dedicated to specific functions, such as computing convolution, down-sampling, and up-sampling operations. In the following, we explain the four basic types of layers used in CNNs: convolutional layer, non-linear function layer, and spatial pooling layer.

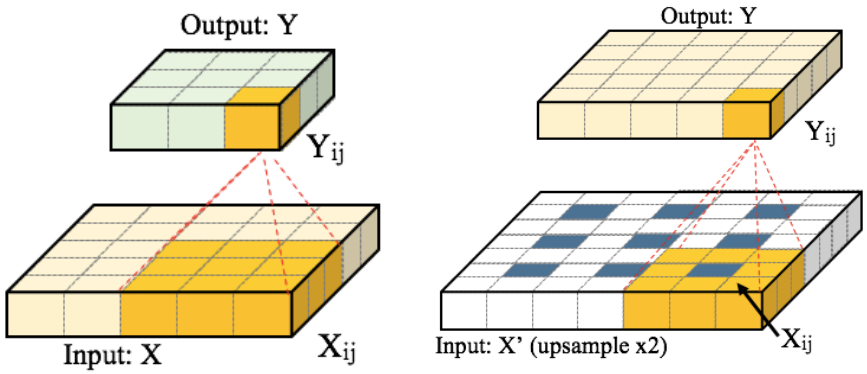
CNNs are built around the convolutional layer. It's a collection of basic filters with programmable settings. The layer convolves the input X of size W1× H1× C1 with the filter bank by sliding stride S and padding the border with P units. This process produces an output volume Y with the dimensions W2×H2×C2. The output at spatial point (i, j) is calculated using this equation:

$$Y_{ij} = \mathrm{W} \times N_{ij} + \mathrm{b}$$

(W, b) are the layer's programmable parameters (weights and bias), N_{ij} is the matching receptive field (or a window enclosing X_{ij}), and W× N is the dot product of W and N.

$$W_2 = (W_1 - F + 2P)/S + 1, H_2 = (H_1 - F + 2P)/S + 1$$

This formula is the spatial dimensions of the convolutional layer output, where F is the size of the receptive field and matches the spatial size of the filters. filters can be varied in size however most CNN systems use filters with F-dimensional square masks.



a) Convolutional layer b) transposed convolutional layer

Figure 2 The convolutional layer's basic components are illustrated. Transposed convolutional layer (b) and convolutional layer (a). modified from (Ferdinand, 2020).

Filters of size F×F×C_1 can be regarded as neurons in the output volume Y. Each neuron instinctively searches the input volume X for a given pattern. The programable weights and bias for all neurons in a channel of Y are divided because the same pattern across all spatial positions in the input volume is needed. This is known as parameter sharing, and the output volume Y is made up of the values acquired when C_2 filters are applied to the input volume X.

The parameter sharing also decreases the number of convolutional layer weights to C_2×F×F×C_1, which is significantly less.

2.2.3.1. Transposed Convolutional Layer

Often known as the deconvolution layer, this layer is frequently used in CNN for up-sampling operations. The up-sampled input is convolutionally processed with a filter bank with size F. The inverse operation of convolution is called transposed convolution. Filter settings can be configured to learn or to follow regular bilinear interpolation.

2.2.3.2. The layer of Non-Linear Functions

A non-linear function layer, also known as an activation function, is frequently used after the convolution layer. This layer serves the same purpose as a fully connected layer in classic neural networks. The Sigmoid function, the Tanh function, the rectified linear unit (ReLU) function, and the leaky ReLU function are all common activation functions. The ReLU function $f(x)=\max(0, x)$ is the most often employed in deep-learning research among these functions.

2.2.3.3. The layer of Spatial Pooling

The input volume is spatially reduced using the spatial pooling layer. To do a simple spatial analysis, a little filter (average size: 2×2 or 3×3) is pushed through the volume. The max, mean, and sum functions are all common pooling functions. It should be noted that the convolutional layer can also be used to replace the pooling layer. However, this technique does not always result in improved performance and would need additional memory and training effort (lee et al., 2016). The max function is the most often utilized pooling function in the literature.

CNN has excelled in various contests involving computer vision and image processing. Image Classification and Segmentation, Object Detection, Video Processing, Natural Language Processing, and Speech Recognition are just a few of CNN's fascinating application areas. CNNs are one of the most effective learning algorithms for comprehending picture content, with outstanding results in image segmentation, classification, detection, and retrieval applications. CNNs include 3 layers which are the Convolutional layer, pooling layer, and fully connected layer. (Khan et al.,2020). The convolutional layer automatically learns map features that correspond best to a specific object leading to higher classification accuracy.

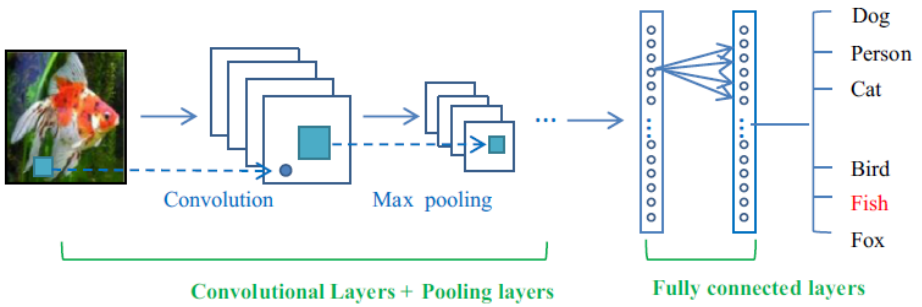


Figure 3 Convolutional neural networks' structure modified from (Rajani, 2017).it contains many layers to detect that the image is a fish

One of the important parts of CNNs is subsampling which is a method that is used because of two main reasons. First, it uses more filters, and using more filters, the better result we have. Since it increases the parameters, so it affects the result better. Second, using subsampling increases the field of view which means that we will look more globally. Figure 4 shows the subsampling methods in the LeNet model which was the first convolutional neural network introduced by LeCun (2015).

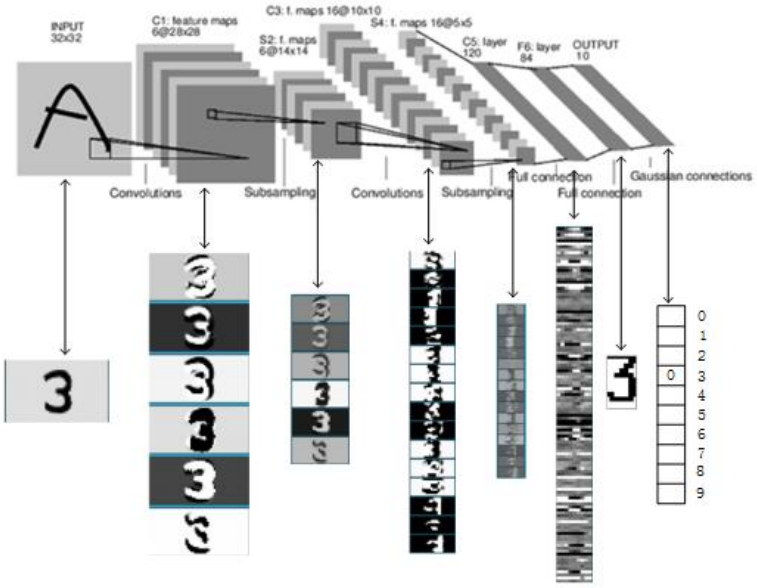


Figure 4 subsampling process in LeNet model modified from LeCun 2015 which explains all the process of models to concludes what is has written in the text

2.2.4. Encoder-Decoder Network

Encoder-Decoder Network allows input to be manipulated by extracting features and attempting to recreate them. It is a network that accepts a feature map/vector/tensor as input and outputs it as a feature map/vector/tensor. The information, or features, that represents the input is stored in these feature vectors. The decoder is a network that accepts the low dimensional feature tensor from the encoder after reducing the input to its minimum resolution. It is typically the same network topology as the encoder but in the opposite direction.

In general, the hourglass module is an encoder and decoder, in which we first downsample the features, and then collect the features to retrieve the information and form the predicted heatmap in the output. Each encoder layer will have a connection to its decoder counterpart, and we can stack as many layers as we want. In the implementation, we usually make some iterations and let this hourglass unit repeat itself (Li, 2020).

2.2.5. Residual networks

These networks introduce skip connections or shortcuts which are used to jump over some layers. Thus, the output of early layers can be fed as input to the next non-adjacent layers. Typical ResNet models are implemented with double- or triple-layer skips that contain nonlinearities (ReLU) and batch normalization. The big advantage of using such a network is the avoidance of common neural networks which is the vanishing and exploding gradient. In addition, Ferdinand (2020) also mentioned that the ResNets slow the convergence of the network's gradient in backpropagation, allowing for deeper networks. The practical benefice is also to merge spatial and feature data. A new sort of residual net that is used in our projects is the bottleneck that is efficient and computationally cheaper.

2.2.6. Spatial Bottlenecks ResNets

The input feature map X in a convolutional layer is a $W_1 \times H_1 \times D_1$ cube, with W_1,$H_1$and D_1 specifying the width, height, and depth (also known as the channels number). At the same condition, the output feature map is a cube Z that includes $W_2 \times H_2 \times D_2$ entries. D_2 are

convolutional kernels for each of S×S×D_1 cube and they are used to parameterize the convolution Z = f (X). $W_2H_2D_1D_2S_2$ is the number of floating-point operations. The kernel size (S_2), the number of connections in the channel domain (D_1D_2), and the resolution of the output feature map (W_2H_2) are the three parameters. The basic concept behind lowering W_2H_2 is to decrease the spatial resolution of the feature map first, and then save it to the appropriate size.

Different scenario to think about the spatial bottleneck is to imagine that the intermediate layer Y' has the same spatial resolution as X and Z, but only a subset of spatial positions on Y' are sampled, so convolution has calculated all coordinates (w,h) satisfying (w,h) ≡ (a, b) (mod K). Some neurons in the output feature map might share some of the calculations due to the spatial bottleneck. This method decreases the computational cost of expanding the receptive field of neurons. (Peng et al., 2018)

the channel bottleneck that has been used in this research exists as a 1×1 convolution for decreasing the channels number to 1/C, followed by a regular 3×3 convolution on the channel reduced layer, and finally, another 1×1 convolution to save the number of channels as it needed.

When there is no channel bottleneck, the residual block has two normal 3×3 convolutional layers. $r^{RB^{3\times3}}$ stands for residual block 3×3. The simple scenario is to change each of them with a spatial bottleneck module. (Peng et al., 2018)

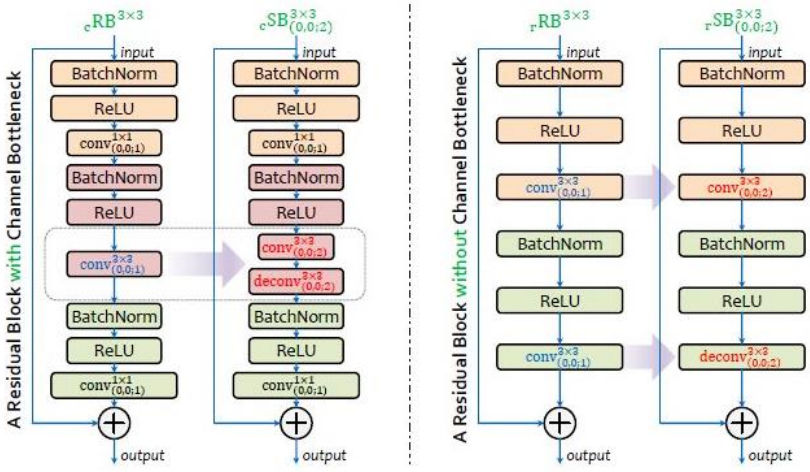


Figure 5 Improvement of a residual block with (left figure) or without (right figure) a channel bottleneck into a spatial bottleneck module. Modified from (Peng et al., 2018)

2.2.7. Attention mechanism

At this step, the attention mechanism has been incorporated into the stacked hourglass networks. The attention mechanism can focus on the important parts of the image while finding differences in it. (Xiaoxia L. 2021). The attention mechanism was introduced by Bahdanau et al. (2014), In natural language processing, the attention mechanism outperformed the neural machine translation system (NLP) which relied on encoder-decoder RNNs/LSTMs before Bahdanau et al developed the first Attention model. This approach, or adaptations of it, was later applied in various applications such as computer vision, voice processing, and so on.

Bahdanau's attention mechanism includes three steps which are alignment scores, the weight, and the context vector: (Cristina, 2021)

- The alignment model uses the encoded hidden state $\boldsymbol{h_i}$ and the previous decoder output $\boldsymbol{s_{t-1}}$ to calculate the scores $\boldsymbol{e_{t,i}}$ which defines the input sequence to match the current output at position t. The alignment model is represented by the function a (.) That can be used in the following neural network.

$$\boldsymbol{e_{t,i}} = a(\boldsymbol{s_{t-1}}, \boldsymbol{h_i})$$

- Weights: Weights $\boldsymbol{a_{t,i}}$ are calculated by using the softmax operation to the previously calculated alignment score.

$$\boldsymbol{a_{t,i}} = \text{Softmax}\ (\boldsymbol{e_{t,i}})$$

- Context vector: A unique context vector $\boldsymbol{c_t}$ is fed to the decoder at each time step. This is calculated by the hidden weighted sum of all encoders T.

$$\boldsymbol{c_t} = \sum_{i=1}^{T} \boldsymbol{a_{t,i}}\, \boldsymbol{h_i}$$

Cristina(2021).

2.2.7.1. The General Attention Mechanism

The general attention mechanism includes three main components, which are queries, Q, the keys, K, and the values, V. in comparison to Bahdanau's mechanism, the keys and values are the same vector. That can be calculated as below:

$$\boldsymbol{e}_{\boldsymbol{q},\boldsymbol{k}_i} = \text{q}.\ \boldsymbol{k}_i$$

Which the query vector (q=$\boldsymbol{s}_{\boldsymbol{t-1}}$) used to calculate a score value. Then the scores pass through a Softmax operation and compute the weights:

$$\boldsymbol{a}_{\boldsymbol{q},\boldsymbol{k}_i}=\text{Softmax}\ (\boldsymbol{e}_{\text{q},\boldsymbol{k}_i})$$

In the end, the attention is calculated by a weighted sum of the value vector and $\boldsymbol{V}_{\boldsymbol{k}_i}$:

$$\text{attention}(\text{q}, \text{K}, \text{V}) = \sum_i \boldsymbol{a}_{\boldsymbol{q},\boldsymbol{k}_i}\, \boldsymbol{V}_{\boldsymbol{k}_i}$$

Cristina (2021).

3. Method and Implementation

3.1. Methodology

The methodology followed in this study is Design Science Research (DSR). It is a relatively new method to create a new world. This kind of research includes three steps which are investigation, induction, and deduction of the issue, context, and activities, as well as hypothesis generation; the second step is design and testing of solutions; and the third one is hypothesis verification, research validation, and generalization to different applications (Pello,2018). In this study, we found the insights and then tried to design a solution based on the findings. We wanted to design a solution to find the best place for map text. to that, we understood the problems and based on that defined objective and restrictions. Finally proposed stacked hourglass techniques.

3.2. Data processing

In this step it has defined how input has obtained and then how it was used and changed its format. The algorithms have been proposed by detailed to define all the processes in the models.

3.2.1. Data collection

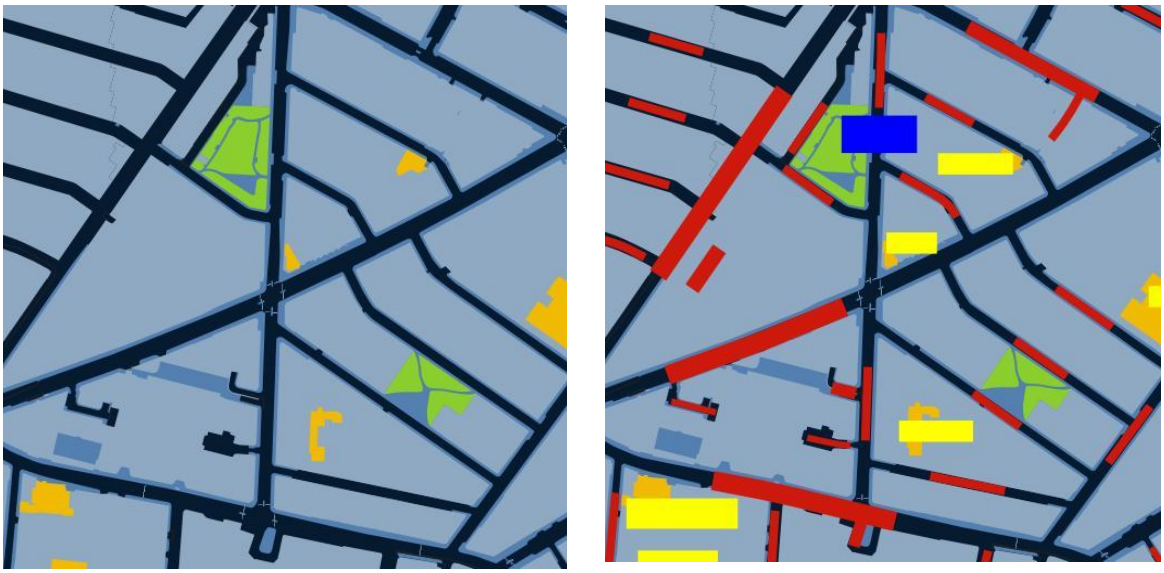
A large number of data is required so that the model can learn from them by identifying certain relations and common features related to the objects. The used data are from the London city map provided by T-Kartor and it is a vector data. These maps are produced in an ESRI ArcGIS environment with substantial manual label editing both in the ArcGIS environment and in the publishing editing tool Adobe Illustrator.

3.2.2. Rasterization

The first step is to obtain suitable data for the input of deep learning models. For this purpose, the vector format of the area features is converted into RGB images. There are all the 4390 images in .Jpg format which is (512x512) pixels. As can be seen in **Error! Reference source not found.**, the input image includes street lines, building areas, green spaces, etc.

In figure7, image (a) which was made from the dataset in a different color that define different land uses to have a better identification of different features in the maps.

Figure 6 An example of the rasterized image which will be the input of the model



a. Non labeled raster image b. Labeled image

Figure 7.A sample image from the dataset. The red bounding boxes show the road labels, yellow for landmarks, and blue for green areas.

After using the algorithms and finding some peak points, the points that have the higher score would choose as the best one. So, in image(b) the best positions to write the labels define by red colors for roads and yellow for landmarks and blue for the green space.

3.2.3. Building of key-points detection model based on stacked hourglass networks

Each image includes some coordinates data (X, Y) which are saved in an excel file under .csv format. It is a table with 4390 rows and several columns. Each row contains coordinates of the feature labels which are the centroids of the label boxes (streets, buildings, and so on). As each image contains a different number of features, the rows include a different number of coordinates. Thus, there are some missing values in the excel file.

Table1: The coordinates of 5 keypoints from the first 6 images

Image name	Coordinates									
	x_1	y_1	x_2	y_2	x_3	y_3	x_4	y_4	x_5	y_5
R60C60	135.1198	199.401	510.0968	369.8386	97.24301	101.0345	129.1652	422.0725	343.6596	0.830955
R61C60	396.7967	233.6029	4.686376	365.0821	409.5015	241.2717	257.9426	90.82879	222.0015	59.27165
R62C60	96.04328	373.6577	180.5553	263.8122	146.3695	49.27697	159.3328	156.1273	476.0289	477.2341
R63C60	149.0015	65.27165	381.3638	187.6849	131.1114	153.0642	63.516	277.1782	265.2003	127.1294
R64C60	129.7338	11.14579	358.3644	307.5612	108.2034	492.6643	300.6952	379.0912	155.2531	365.5782
R65C60	167.867	430.3015	250.6007	415.8025	88.29675	464.2381	105.0015	387.2717	29.2311	79.6996

As shown in figure 8, the background of the image has been removed to better focus on the goal features. The algorithm is tried to prevent any overlaps between the background and other significant features. And the centroid of labels which were defined manually has shown by green color.

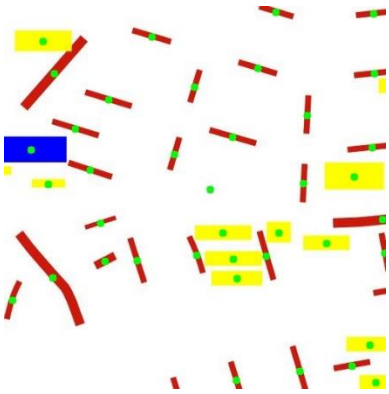


Figure 8.Background features are removed and only the boundary boxes of the manual labels are displayed with their centroids. Label centroids are the green dots.

3.2.4. Selection of the inputs to feed to the model by trying different cases.

The stack-hourglass code includes different parts. The major parts contain 3 codes which are param.py, data.py, and Train.py part.

- **TensorFlow**

In this study, we have used TensorFlow for the implementation of the deep learning models. TensorFlow is a comprehensive open-source machine learning platform, that contains a comprehensive and flexible ecosystem of tools, libraries, and community resources that allows researchers to push the latest in machine learning technology and developers to easily build and deploy machine learning-powered applications (Unruh, 2017).

- **Pandas**

Pandas is used for data manipulation and analysis. It includes data structures and methods for manipulating numerical tables and time series. It is a Python library that provides quick, versatile, and expressive data structures for working with "relational" or "labeled" data. It is based on two keyPython libraries which are matplotlib for data visualization and NumPy for mathematical operations. Before pandas, most operation has done with python and R.(https://pandas.pydata.org/2022)

3.2.5. Model Parameters

- **key-points**

20 key points have been chosen for each image which means we have 20 numbers (10 for X, and 10 for Y) for each image.

- **Image size**

As mentioned before the size of the images is 512 by 512 pixels.

- **Train Parameters**

The Batch size determines the number of samples that will be distributed over the network. In this model, images have been compared two by two. Using Batch size help to use less memory because

fewer sample would be used. While the Epoch is a hyperparameter that shows how many times the training algorithms will work. And as explained before the batch size is also a hyperparameter but it defines sample numbers that are used for learning algorithms.

3.2.6. Data Augmentation

Data augmentation is carried out to mitigate the overfitting and aims to increase the size of the dataset by performing some transformations on the dataset. Three operations are applied to the images: adding random lighting noise, random flipping, and expansion. The lighting condition of the images is varied by adding Gaussian noise to the image (Gandhi. 2021). This helps CNN not learn the irrelevant pattern and thus boosting the overall performance.

3.3. Model building

The experiment is performed using a stacked hourglass with a different number of stacks (2, 4, and 8). Each stack, i.e., single hourglass module, is composed of a bottom-up submodule on the left followed by a top-down submodule on the right as shown in Figure 9. On the encoder part, convolutional and max-pooling layers are used to process features down to a very low resolution. Each time max pooling is used, the network splits up and applies more convolutions at the original pre-pooled resolution. After achieving the lowest dimension space, the right half of the stack which is a decoder begins the top-down sequence of upsampling and fusion of features across scales through skip connections by elementwise addition. The output is then a higher resolution feature map.

The input is passed into a 7×7 convolution combined with stride 2, Batch Normalization, and ReLu layer. The first convolution layer is different as it has a kernel of size 7×7 in contrast to the others with only 3×3 or 1×1 kernels. It has been shown that applying multiple small filters instead of a larger one can improve the overall performance of the hourglass. Thus, we are using 3×3 filters instead of 5×5 or 7×7 kernels. In addition, applying convolution of a 1×1 filter to reduce the resolution improves its performance.

The input is passed into a 7×7 convolution combined with stride 2, Batch Normalization, and ReLu layer. The first convolution layer is different as it has a kernel of size 7×7 in contrast to the others

with only 3×3 or 1×1 kernels. It has been shown that applying multiple small filters instead of a larger one can improve the overall performance of the hourglass. Thus, we are using 3×3 filters instead of 5×5 or 7×7 kernels. In addition, applying convolution of a 1×1 filter to reduce the resolution improves its performance.

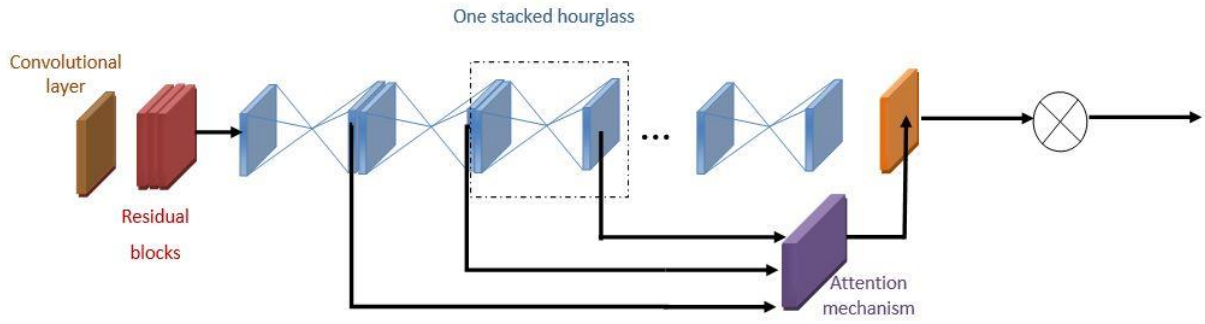


Figure 9.The proposed hourglass-shaped network architecture. it explains the thesis algorithms and the relations between different layers

Each stack is a type of convolutional encoder-decoder network which breaks down the image inputs to extract the essential features and then tries to reconstruct them faithfully. The feature matrix has low spatial understanding, meaning it does not really know the location of objects in the image. In fact, the extraction of the object's features requires the discard of all pixels which are not features of the object and this breaks down to remove all the background pixels and thus losing knowledge of the object's context. However, fusing these feature matrices with earlier layers that have higher dimension and spatial understanding help to both learn rich and precise object characteristics and their location and context. This is efficiently achieved by ResNets which combine the spatial with the feature information. So, we have used Bottlenecks including 3 convolution layers: one 1×1 convolution, one 3×3 convolution, and one 1×1 convolution. Basically, after each pooling layer, we add one of the bottleneck layers. (Peng et al., 2018)

To summarize: we have as input of SHG the non-labeled map. Then each stack has an encoder that extracts features by breaking down the input into feature tensors. Afterward, the decoding part

combines the feature information and the spatial characteristics to understand properly the input image. Convolution layers help to extract features with high quality, max-pooling layer decreases the dimension as they bring gradually the resolution down from 512 to 64 and eliminate the unnecessary parts of the image that contain redundant or low-quality information. On the other hand, the Bottleneck ResNets allow to save the spatial information and use the memory efficiently. Finally, two 1×1 convolutions are applied to the output to get the final prediction which is a heatmap that shows the locations of keypoints through the probability of occurrence of a keypoint.

3.4. The learned output: Heatmap

The heatmap is used to depict a keypoint location in an image and saves the location of the data. Once the heatmap is obtained in the output, all that is required is to determine the heatmap's peak point and utilize it as the keypoint position. In the output of each stack of the network, a heatmap is learned and the loss is computed for it.

In other words, the heatmap displays peaks at each predicted label point of the area feature, with pixel-by-pixel detection scores. A feature map with a resolution of 1/4 of the input picture is created at the start of the stacked hourglass networks using convolution and pooling layers. The heatmap is then created using a network of numerous stacked hourglasses with the same resolution as the feature maps. This shows the advantage of stacking numerous hourglasses and how it allows for more accurate final predictions as claimed in the theoretical part of this report.

3.5. Training

We train the network to optimize the mean squared error loss function using RMSprop optimizer after comparing it with the Adam optimizer. The learning rate is set to decrease 10-times from 2.5e-4 every 30 epochs. The batch size is set to 8. The image patches are extracted with size 512*256 with 20% of augmented data and 5% of overlap. The network is trained from scratch until 150 epochs, practically until the loss converges. The training and testing processes are performed on a server equipped with Nvidia GeForce Titan X (12 Gb vRAM). So, in this step, the model has been compiled and trained, and then it has been fitted.

In the previous line, optimizing algorithms ensure the minimization of the loss function and update the model weights in the backpropagation process. In the fitting part, the Epoch and batch_size have been chosen.

3.6. Implementation

Using the attention mechanism helps to concentrate on the target location. It pays attention to the most important features and uses them to optimize the stacked hourglasses network for better results. It has a sequence-to-sequence model which is called the Encoder-Decoder model. Working with long input data is hard, therefore the previous hidden state of the Encoder goes to the next Decoder as vector data. For all the output of the decoder, the model has been able to select special features to make the output result. That is the reason they called it "Attention Mechanism" (loye, 2019). The decoder can retake the encoded representation. The process of using the attention mechanism in this research is as follows:

3.7. Producing the Encoder class

First of all, the input data goes through the embedding layer which initializes the batch size and encoding units. It encloses __init__ () and call() methods. In the call() method, describe the forward propagation that has to happen through the encoder network.

3.8. Optimizer and Loss Functions

For finding the optimizer and loss function the batch_size, height, width, and n_keypoints has considered. The squared error loss function is one of the most often used loss functions in statistics, albeit its popularity originates from mathematical convenience more than real loss considerations in applications.

Carl Friedrich Gauss, the inventor of mean squared error, was aware of its arbitrariness and agreed with critics who objected on these grounds. The mathematical advantages of mean squared error are especially apparent when analyzing the performance of linear regression, as it allows one to partition the variation in a dataset into variation explained by the model and variation explained

by chance. (Wikipedia) here is the mean squared error equation where σ^2 is the population variance and $\mu\ is\ mean.$

$$MSE(\bar{\bar{X}}) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

$MSE = mean\ squared\ error$
n=numbers of data points
Y_i= observed values
$\hat{Y}_i$= predicted values

4. Results and findings

4.1. Evaluation

To evaluate the methods and algorithms the accuracy rate and loss have been considered for this aim the ground-truth value (gt) and the predicted value(pred) have been compared. The differences between them show the accuracy. If the result is higher than the threshold, it wouldn't consider. So, the lower amount, the better accuracy.

According to the tensorflow.org website tf. constant () function is used to calculate a constant tensor from an object which is tensor-like. dtype has chosen as dtype=tf. It is also creating a stack from tf. tensors into an r+1 rank tf. tensor. it also used some parameters while evaluating the model such as tensor which is a list that includes some objects that are similar in terms of shape and dtype. Another parameter is the axis which is the stack's axis. The last one is the return value which returns tf. Tensor.

4.1.1. Learning metric: Loss function

The loss function used is the Mean Squared Error which is calculated over the predicted and ground-truth heatmaps and aims to estimate the distance between them. First, we compute the difference between each pixel gray level value in the two images for the first channel, then we compute the sum of squared differences for all pixels in that channel. Afterward, this calculation is repeated for all the channels.

4.1.2. Evaluation metrics

4.1.2.1. Percentage of Correct Key-points (PCK)

PCK is a precision metric that determines if the predicted keypoint and the ground-truth centroids are within a distance threshold. The PCK is often set to the object which is included within the bounding box or to the image scale. The metric measures the percentage of detections that fall within a normalized distance of the ground truth and is calculated as the percentage of detections that fall within a normalized distance of the ground truth. It is usually used for the body joint

detection and the true detection is determined by a threshold relative to the body part. When the normalization reference is the head length, it is noted PCKh. For example, PCKh@0.5 denotes a threshold of 50% of the skull bone link and PCK@0.2 means the distance between the expected and true joint is less than 0.2 * torso diameter. In addition, PCK can be utilized in both 2D and 3D graphics (PCK3D) (Barla, 2022). In our case, the threshold is expressed in terms of the number of pixels. We set up a threshold parameter that determines the range of allowed errors. A candidate keypoint is considered correct if it falls within α pixels away from the ground-truth keypoint.

This metric is equivalent to the accuracy metric which is used for evaluating classification models. It is the fraction of predictions the model gets right. Formally, accuracy has the following expression:

Accuracy = Number of correct predictions / Total number of predictions

4.1.2.2. Percentage of Correctly estimated body Parts (PCP)

As this metric is initially developed for the evaluation of the human body joint detection, it measures the is the percentage of correctly estimated body parts. If the segment endpoints of an estimated body part are within a fraction of the length of the ground-truth segment from their annotated position, it is considered accurate. The smaller the PCP threshold is, the criterion will be harder, and the accuracy will be better (Eichner, 2010). In our case study, the prediction can be considered correct or accurate if the predicted keypoints fall inside the boundary box of the ground-truth label or a fraction of it. It can be formulated otherwise by considering the predicted keypoint correct if it lies inside the feature itself or a fraction of it. However, the first formulation is better than the second which does not guarantee the respect of the labelling rules.

y_i^{t*} is the predicted keypoint of the model and i^{th} is the centroid coordinates for the data sample t.

$$acc_i(\mathrm{r}) = \frac{100}{N} \sum_{t=1}^{N} 1\left(\frac{100 \cdot \|y_i^{t*} - y_i^t\|_2}{\|y_{imax}^t - y_{imin}^t\|_2} \ll r\right) \qquad \text{(adapted from Sapp \&Taskar,2013)}$$

4.2. Experiments and Results

To evaluate the performance of the built model and understand how keypoint detection concept will help to obtain accurate label places, we have performed 3 experiments. These experiments and their results are detailed as follows:

Experiment 1: We have used the stacked neural network without the attention mechanism and tested different parameters: number of stacks, number of blocks, and learning rate. The goal is to find the best architecture of the networks and to understand the effects of the most important parameters on the network performance.

After a comprehensive comparison, we found out that stacked hourglasses outperform a single hourglass when tested with the same number of layers and parameters. Furthermore, applying supervision at different stages is more effective, particularly when applied to the two resolutions right after upsampling and before the final output.

Table 2. Results of the first experiments

Number of stacks	Number of blocks	Accuracy
2	2	54.8
	4	56.1
4	2	56.8
	4	58.2
8	2	58.6
	4	**60.7**

The number of blocks increases slightly the performance of the model with an increase of approximately 1.2% when we use 4 blocks instead of only 2. However, the number of stacks has a higher impact as it can improve the accuracy by around 2% when using 4 blocks instead of 2 and an increase of around 6% when 8 blocks are used.

Concerning the learning rate, we have experimented with different values: 5e-5, 1e-4, 5e-4, and 1e-3 and we achieved respectively the following maximum accuracies 57.3%, 59.6%, 60.7%, and 59.3%. Therefore, the learning rate of 5e-4 seems to be more adequate and ensures a good convergence of the learning. It is worthy to note that this is tested with the best-found architecture of SHGN of 8 stacks and 4 blocks. Choosing the learning rate of 5e-4 can achieve higher accuracy, but at the expense of slower learning as the convergence is achieved after 85 epochs in contrast to the case were using for example a learning rate of 1e-3 in which the model start to achieve a convergence state after only 64 epochs.

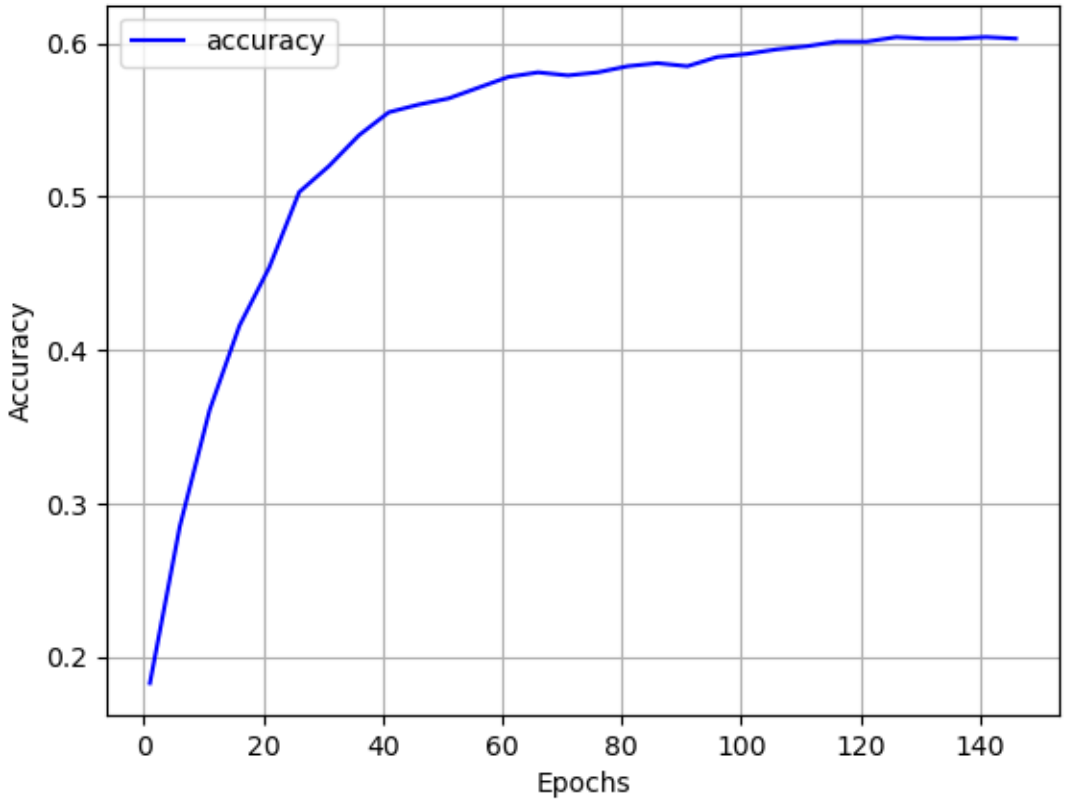


Figure 10 Accuracy of SHGN during the training phase.

Figure 10 shows the progress of the accuracy during the training of the SHGN with the best configuration: 6 stacks, 4 blocks, and a learning rate of 5e-4. It starts to converge after 80 epochs and achieves a maximum of 60.72%.

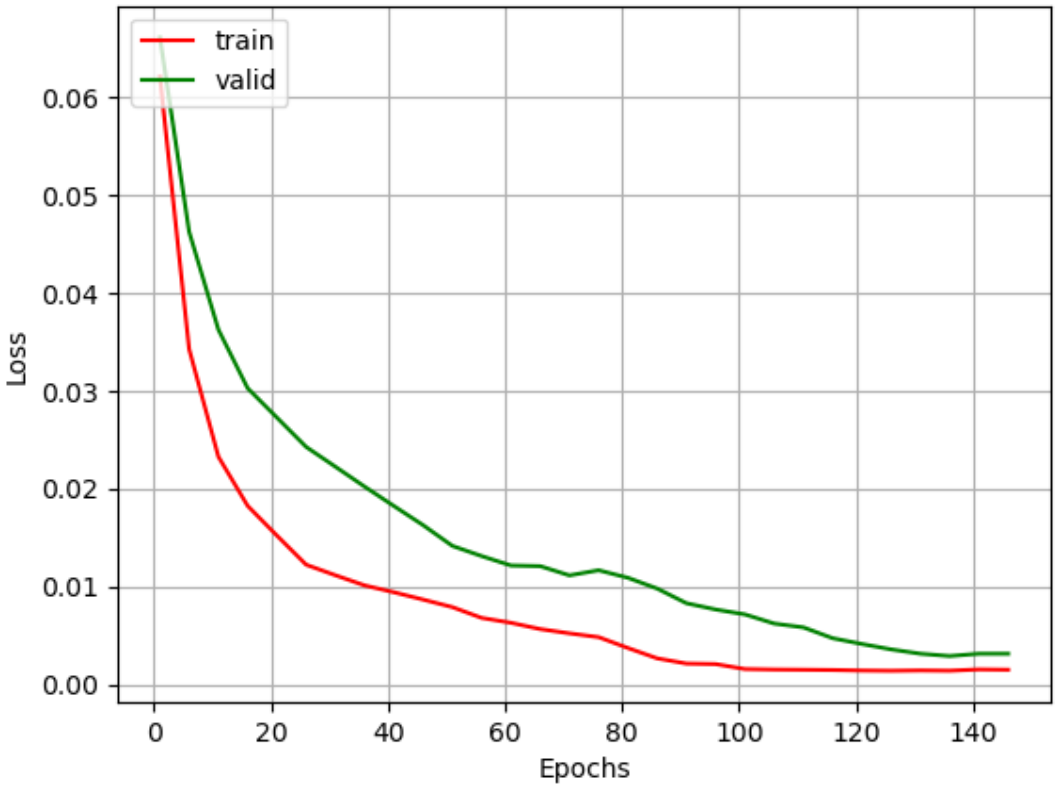


Figure 11.Training and validation loss during the training of SHGN

Experiment 2: Using the best architecture and parameters found in the first experiments, we have included the attention mechanism with the hourglass network. Then, we train the entire network again. The model achieved an accuracy of 63.1% with an increase of 1.4%. This shows that this mechanism can improve the overall performance of the model. Once the model is fully trained, we run it on test data. Fig. 12 shows the predictions obtained for one of the raster images using an error threshold of 20pixels.

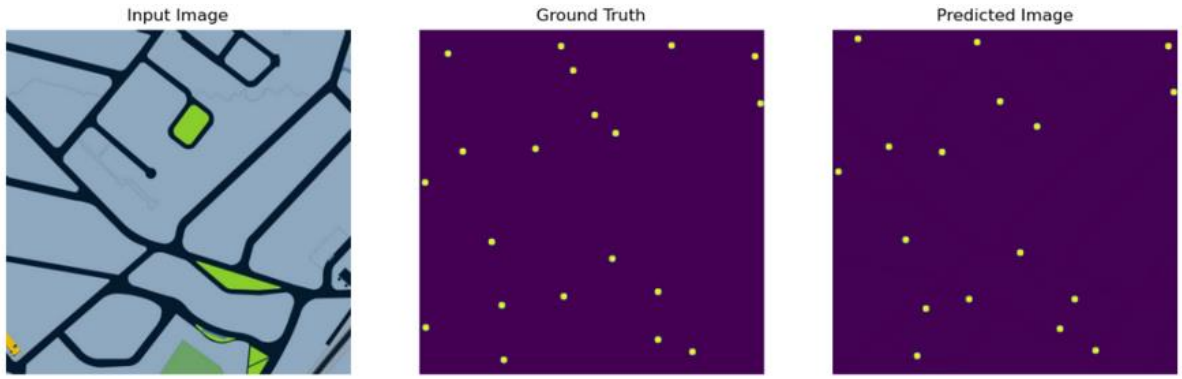


Figure 12, Visualization of the prediction obtained by the SHGN for one raster map sample, displaying the input image, the ground truth keypoints, and the predicted keypoint positions.

Experiment 3: We tested different margin errors by varying the parameter α controlling the threshold of correctness. So, we used a smaller value of 10. When we decrease the threshold to α = 10 pixels, approximately 60% of the keypoints disappear from the predicted heatmap. This indicates that those predictions are far away from their correct positions, i. e. from the centroids of the manual labels. Figure 13 depicts a prediction of the trained model of a sample image from the test set. When we decrease the threshold to α = 10 pixels, approximately 60% of the keypoints disappear from the predicted heatmap and the accuracy falls down to 42%. This indicates that those predictions are far away from their correct positions, i. e. from the centroids of the manual labels.

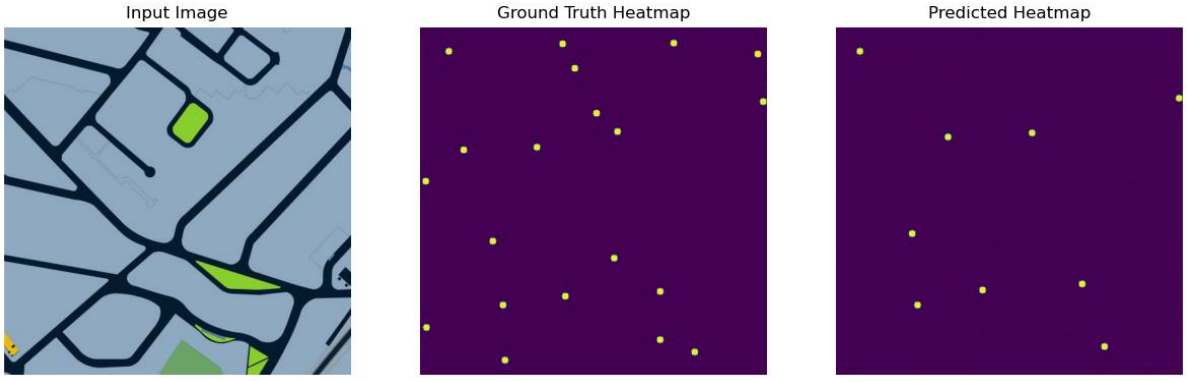


Figure 13.The prediction obtained by the SHGN showing the input image, the ground truth keypoints and the predicted keypoint positions, with a threshold α = 10.

4.3 Discussion

In this thesis, a deep learning model based on stacked hourglass networks and attention mechanisms is implied to tackle text labelling problems. The model is designed to learn how to predict the best places for the labelling in the maps. So, the input data has been used then the algorithms try to learn the input by iterating over batched data which contains both the map images and coordination information. The model is trained for many epochs until they achieve the best accuracy or ensures some convergence. To answer the research questions, we tried to find the best architecture of the model. We initially led a benchmarking of the literature, thus we used bottleneck ResNets instead of regular ResNets. Then, we tried the different numbers of blocks and stacks. It yields that the best architecture is to use 8 stacked hourglasses with 4 blocks. In addition, we tuned the learning rate, and it turns out that 5e-4 is the best value ensuring a good convergence. Afterward, an attention mechanism has been employed to improve the stacked hourglass network.

Limitations and perspective

This study is subject to some limitations, and further improvements are expected to encounter and fill the gap. First, the used approach only determines the positions where the labels should be placed and do not deal with the size of the bounding boxes of the labels and the overlapping that may occur between them. In addition, fitting the text in their exact positions is considered as a postprocessing step. One of the important weak points of this approach is the number of keypoints that should be fixed. The model is trained only for a determined number of keypoints. In our experiments, we used mainly 20 keypoints for each raster image.

This study opens many doors as perspectives. First, the evaluation of the proposed approach is done only using deep learning metrics, namely the metrics used for keypoit detectors, using accuracy-related measurement. However, evaluation using quality functions of map labelling is more valuable and reflects accurately the performance of the approach.

In addition, the only loss used is the squared mean errors which are general. However, designing losses related to cartography is an asset. Actually, the advantage of using deep learning is to learn implicit features of the data and get rid of manual processing, function objective definition or rule design. However, the use of a simple loss function that includes general requirement of map

labelling (for example overlap measurement) will guide the learning and thus help to achieve a near-optimal solution. In all cases, the use of deep learning help to avoid quantification and formulation of rules and exhaustive optimization.

4.4. Conclusion

This thesis focused on the following steps: 1) it has used deep learning and machine learning algorithms to suggest a novel technique for automating maple belling. And it was the first time that the stacked hourglass and attention mechanism has used for this purpose.

2) one of the most important achievements of this proposed technique is to reduce the time in comparison to the previous methods.

3) there were three experiments to find the highest accuracy and it figured out that by increasing the number of stacked, the accuracy will be more precise. Also, it has been understood that attention mechanisms are useful techniques to have a better result in map labelling. Finally, by choosing a threshold, it can be so effective on accuracy.

Bibliography

Bahdanau. D, Cho. K, Bengio. Y, (2015) Neural machine translation by jointly learning to align and translate, in: ICLR

Barla. N (2022) A Comprehensive Guide to Human Pose Estimation.https://www.v7labs.com /blog /human-pose-estimation-guide

Bertin, J.(1983) Semiology of Graphics, University of Wisconsin Press

Eichner. M., Marin-Jimenez .M, Zisserman. A, and Ferrari. V. Articulated human pose estimation and search in (almost) unconstrained still images. Technical report, ETH Zurich, D-ITET, BIWI, 2010

Freeman, Lance. (2005). Displacement or Succession?: Residential Mobility in Gentrifying Neighborhoods. Urban Affairs Review - URBAN AFF REV. 40. 463-491. 10.1177/1078087404273341.

Cederholm, P. (2020). Automatic label placement for city maps with the labelling library PAL. Student thesis series Ines.

Chamra, Tomáš (2017) Algorithms for Automatic Label Placement, M.Sc. thesis in Computer Science (Department of Computer Science) Czech Technical University

Christensen, Jon, Joe Marks, and Stuart Shieber. 1992. Labeling Point Features on Maps and Diagrams. Harvard Computer Science Group Technical Report TR-25-92.

Courtial A, El Ayedi A, Touya G, Zhang X (2020) Exploring the Potential of Deep Learning Segmentation for Mountain Roads Generalisation. ISPRS International Journal of Geo-Information 9(5): 338.https://doi.org/10.3390/ ijgi9050338

Cristina. S, (2021),https://machinelearningmastery.com/the-attention-mechanism-from-scratch/

Elhagry, A., Saeed, M., & Araia, M. (2021). Lighter Stacked Hourglass Human Pose Estimation. arXiv preprint arXiv:2107.13643.

Ferdinand, (2020 may30) Using Hourglass Networks to Understand Human Poses. https://towardsdatascience.com/using-hourglass-networks-to-understand-human-poses-1e40e349fa15)

Gandhi,2021 Data Augmentation: How to use Deep Learning when you have Limited Data.https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/

Imhof, E. (1975). Positioning Names on Maps.American Cartographer2(2):128-144.

Jun94,2020, Convolutional Neural Network Basics, https://medium.com/jun94-devpblog/dl-8-cnn-1-convolutional-neural-network-basics-2f60c93d7d22

Kato,T., and Imai,H. (1988). The NP=completeness of the character pacement problem of 2 or 3 degrees of freedom. In Record of Joint Conference of Electrical and Electronic Engineersin Kushu, page 1138.

Kang Yuhao (2020) Transferring Multiscale Map Styles Using Generative Adversarial Networks. M.Sc. thesis in Cartography and GIS (Department of Geography) University Of Wisconsin–Madison

Kern, J.P., & Brewer, C.A. (2008). Automation and the Map Label Placement Problem: A Comparison of Two GIS Implementations of Label Placement. Journal of the Brazilian Computer Society, 22-45.

Khan, A., Sohail, A., Zahoora, U., & Qureshi, A.S. (2020). A survey of the recent architectures of deep convolutional neural networks. Artificial Intelligence Review, 1 - 62.

Krumpe, F. and Mendel, T., 2020. Computing Curved Area Labels in Near-Real Time, arXiv:2001.02938.

Lee,C.Y.; Gallagher,P.W.;Tu,Z,(2016) Generalizing pooling function in convolutional neural networks: Mixed, gated and tree. In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS), Cadis, Spain,9-11 May 2016

Li Y, Sakamoto M, Shinohara T, Satoh T (2020) Automatic label placement of area-features using deep learning. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLIII-B4-2020:117–122. https://doi.org/10.5194/isprs-archives-xliii-b4-2020-117-2020

Li ,2020,Human Pose Estimation with Stacked Hourglass Network and TensorFlow.https://towardsdatascience.com/human-pose-estimation-with-stacked-hourglass-network-and-tensorflow-c4e9f84fd3ce

License – Package overview – pandas 1.0.0 documentation. pandas. 28 January 2020. Retrieved 30 January 2020.

Liu, Yu, Duc Minh Nguyen, Nikos Deligiannis, Wenrui Ding, and Adrian Munteanu. 2017. "Hourglass-ShapeNetwork Based Semantic Segmentation for High Resolution Aerial Imagery" *Remote Sensing* 9, no. 6: 522. https://doi.org/10.3390/rs9060522

LeCun, Yann & Bengio, Y. & Hinton, Geoffrey. (2015). Deep Learning. Nature. 521. 436-44. 10.1038/nature14539.

Loye, Gabriel .(2019), attention mechanism, https://blog.floydhub.com/attention-mechanism

Marks, Joe and Stuart Shieber. (1991). The Computational Complexity of Cartographic Label Placement. Harvard Computer Science Group Technical Report TR-05-91

Njima, Wafa & Ahriz, Iness & Zayani, Rafik & Terre, M. & Bouallegue, Ridha. (2019). Deep CNN for Indoor Localization in IoT-Sensor Systems. Sensors. 19. 3127. 10.3390/s19143127.

Oeltze-Jafra, S., Preim, B., (2014). Survey of Labeling Techniques in Medical Visualizations. Visual Computing for Biomedicine (VCBM'14), Eurographics Association, 199-208.

Pello, rauno (2018), Design science research-asummary. https://medium.com/ @pello/ design-science-research-a-summary-bb538a40f669

Peng, J., Xie, L., Zhang, Z., Tan, T., & Wang, J. (2018). Accelerating Deep Neural Networks with Spatial Bottleneck Modules. *ArXiv, abs/1809.02601*.

Rajan, Vani. (2017). IJARCCE Towards Efficient Intrusion Detection using Deep Learning Techniques: A Review. 6. 375-384. 10.17148/IJARCCE.2017.61066.

Robinson, Arthur Howard (1995) Elements of Cartography, 6th Edition. New York: John Wiley & Sons

Rylov, M & Reimer, A. (2014). A Comprehensive Multi-Criteria Model for High Cartographic Quality Point-Feature Label Placement. Cartographica: The International Journal for Geographic Information and Geovisualization. 49. 52-68. 10.3138/carto.49.1.2137.

Sapp.B and Taskar.B (2013), "MODEC: Multimodal Decomposable Models for Human Pose Estimation," *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3674-3681, doi: 10.1109/CVPR.2013.471.

Unruh, (2017, November 9), What is the TensorFlow machine intelligence platform?.https://opensource.com/article/17/11/intro-tensorflow

van Dijk, Teun. (2002). Discourse, Ideology and Context. Journal of Asian Economics. 35. 11-40. 10.1515/flin.2001.35.1-2.11.

Wei L (2020) An artificial intelligence method for text placement evaluation in maps. M.Sc. thesis series INES nr 524, Lund University, Sweden

Wolff, A., Strijk, T.(2009), The Map Labeling Bibliography

Xiaoxia Luo and Feibiao Li(2021), "Stacked hourglass networks based on polarized self-attention for human pose estimation", Proc. SPIE 12079, Second IYSF Academic Symposium on Artificial Intelligence and Computer Engineering, 120792A (1 December 2021); https://doi.org/10.1117/12.2622889

Yoeli,p. 1972. The logic of automated map lettering.The Cartographic Journal 9(2):99-108,December.

Zhang, C., Xu, Y. and Shen, Y., 2021. Decorating Your Own Bedroom: Locally Controlling Image Generation with Generative Adversarial Networks. arXiv preprint arXiv:2105.08222.https://towardsdatascience.com/using-hourglass-networks-to-understand-human-poses-1e40e349fa15