Classifying deep features in an urban layout

Qi Chen

MSc in Data Science
The University of Bath
September 2019

Classifying deep features in an urban layout

Submitted by: Qi Chen

for the degree of MSc in Data Science of the
University of Bath
September 2019

COPYRIGHT

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of MSc in Data Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Signed:

Abstract

Deep learning architectures can simulate human brain system to recognize patterns which is extract features for clustering and classification. This dissertation aims to implement one end to end method to realize urban classification based on deep features. It is offered a type of street network to capture the character of urban layout and category different places around the world. Deep learning algorithm is been study performances of hyper-parameters modify. Specifically, it is focused on find the way to reach a high accuracy in a short time. The procedure is been measured by a choice of cities in different network types and display the suggested network scale descriptors of various cities is predictable. Furthermore, activation function of Convolution Neural network can affect convergence rate of the model.

Contents

List of Figures

List of Tables

Acknowledgements

Throughout the production of this dissertation I have received many great advice and assistance. I would first like to thank my supervisor Dr. Yongliang (Mac) Yang, thank you for your guidance and expertise.

I would like to express my gratitude to my family and friends. Thanks to my parents for always unconditional support, I love you. Thanks to Meghna, Leyan, Sergio and Tze-yi of inspiration and motivation. We have had a lot of happy time together. Finally, thanks to the classmates and teachers I met in the past year, this will be a good memory of my life.

Chapter 1

Introduction

3.1 Problem Description

Aspects of city planning, traffic management and infrastructural problems rely on specific data types. Urban data include different types, images, laser scans and representative data. Images and laser scans, in particular, can be used to analyze traffic flows and improve traffic design. Feature data can be used as a reference for designing new urban planning. However, there is considerable data processing work involved in urban planning and it is important to identify the effect of feature data and devise a clear way to recognize and visualize the whole structure of data set.

The transportation network of an urban layout constitutes the main distribution of a city. This determines the main transportation type for citizens, such as a Constellation layout, Ring layout or Block layout. The radial layout (Figure 1.1 and 1.2) is developed from city center lead to a shaped building block.

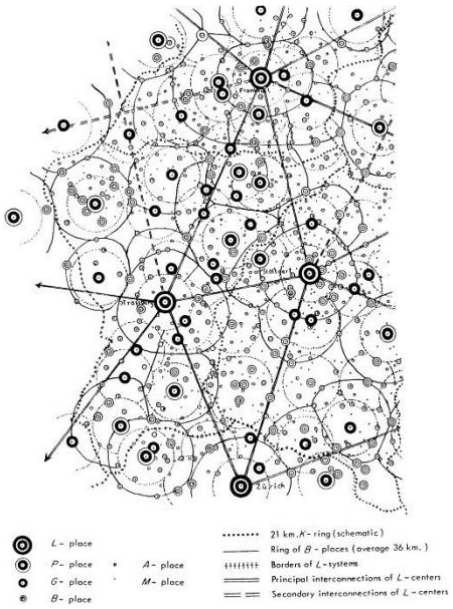


Figure 1. 1 A typical urban radial layout map diagram

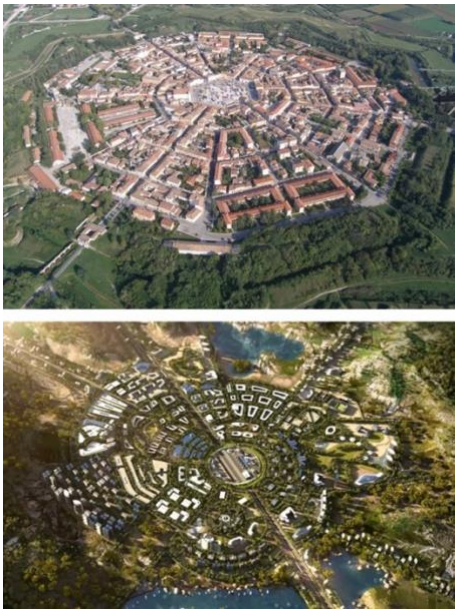


Figure 1. 2 City radial layout image

3.1.1 Main task

In this project, a model will be created to identify characteristics of different cities based on their street network design. In order to realize this, feature extraction needs to be carried out from raw data collection. There should be a focus on the urban functionality rather than two-dimensional image expression, because applying feature data to generate an urban framework directly can be more effective way than proposing descriptors. This can be evidenced from past research; functional features correspond directly to paths and transport networks. In this project, we will focus on a large-scale analysis, and investigate the effectiveness of different street network features.

This project process can be divided into 4 parts. The first is to generate descriptors to characterize the geometric and layout of street networks. The second is to use a convolutional neural network method to calculate and pick out effective influencing factors. These effective descriptors and the values can supply insights into city characteristics and be used to rank the importance of each descriptor. Thirdly, a convolutional neural network method (CNN) is applied to generate the classification types and at the same time, generate a turban rank list based on functional level. Finally, the model will be evaluated and recommendations for modifications will be presented.

3.2 Specific street network distinguishes

In internal city classification, the density and connectivity index of four-way intersections play a leading role. In intra city classification, the best way to make distinctions is to start with intersection and street densities. The convolutional neural network (CNN) method is recommended for processing, since it extracts image information directly from the street maps. The CNN method generates a number of parameters related to the size of the original image. If the same CNN runs on a 300 × 300 image, the number of parameters remain at the same number in the convolution layer. This will not increase the computation, and the resultant CNN can be applied in processing different sizes of an image.

In traditional machine learning algorithms, a large number of features are required, and the weighting of such features has an important effect, and so it is easy to make mistakes. As a result, feature engineering requires time and energy, in order to obtain a good model training outcome. However, the emergence of a neural network means that classification does not need features to be previously identified or weighted. The data itself trains the model and there is a process of self-correction. This process can achieve better results. If the number of original street network databases is large enough and the number of network feature values general enough, then the spatial hierarchy of features can be obtained by a trained algorithm model. In this way, a new generic model can be effectively generated in the pattern of the parsed OSM street network image. Moreover, the nonlinear activation data process is essential at the filter output of all intermediate layers for an application.

Chapter 2

Literature Survey

2.1 Image classification

Image classification have been widely used in modern industry, such as image automatic recognition of face systems on photograph, Biometric technology, Monitoring System, autonomous vehicle navigation (Kh Tohidul, 2018), and so on. The traditional methods are to find feature description and detection. Instead of pointing out each type of appearance directly in code, machine learning usually used to providing the features or label for computer and developing learning algorithms to obtain the characters from this feature. Finally, integrate similar images or distinguish different categories of images.

2.1.1 Street network data sources

- GIS data source
- Satellite image (google map)
- EarthExplorer
- European Space Agency
- NASA Reverb
- Global Land Cover Facility
- Digital Globe
- Geographic information system (GIS)
- OSM street network image

2.1.2 Data features

Features extracted from image data are required distinctly translated into semantic description. Image features consist of Physical and geometric features. Street network features including geometric, radiative, topological and contextual features. (Jiang, 2019). Most road characteristics can be described as follows (Table 2.1):

Table 2. 1 Road characteristics description

Features	Characteristics
Geometric features	The visualization of geometric characteristics image clearly shows the relationship between physical quantities, and makes clear the corresponding physical meaning of geometric characteristics of image
Radiation characteristics	The road surface and background have a certain angle and directly connected to city center
Topological characteristics	In the related fields of topology and mathematics, topological properties or topological invariants are properties in which the topological space is invariant under the same embryo. That is to say, one property of space is a topological property. When a space X possesses this property, each spatial homeomorphism has this property. In layman's terms, topological properties are spatial properties that can be represented by open sets.
Contextual features	Network block and street setting have been relatively single, never covered the street surface. This feature always be used to check if the main street or the rural.

There are two methods used for feature extraction, which is semi-automated and automated methods. These methods could help reduce time at a certain label for the street network data and they can be directly used at Satellite image. There are many methods for the street network layout classification. Active contour models which is the semi-automatic extraction of street.

Feature extraction is provided for understanding the shape and position of network layout. This method can ignore some noise at image and supply a relatively robust performance.

Detection the road area is a novel method. It can track roads through analysing the morphological parameters and dynamic planning way. Shi's (2014) work has used the centreline extraction of the main road to make classification at spatial and spectral level, and segment images to different parts based on road and non-road group.

Edge detection techniques is to extract the right road label. The labelling of road pixels is subject to errors and takes a very long time. A multi-stage framework is proposed to extract road networks based on salient features. The idea is to use two road features, linear trajectory and spectral contrast method. Statistical inference method designed with such linear characteristics is modelled as a Markov process and a geometric stochastic model that can be applied to the width, direction and background strength of the street and the maximum posterior probability to help in assessing the street network.

Linear feature extractions were given by Quackenbush (2004), Image from which the street can be extracted. The importance of using automated methods lies in saving time and cost factors and improving the accuracy of details.

Road intersections extracted from satellite images were identified and studied. This suggests that pure road intersection extraction does not represent a complete road network, but it helps to understand the road network topology that can be used for higher processing.

2.1.3 Image classification methods example

There is a list of classification methods and discusses their strength and weakness.

1. Support vector machine (SVM) (Melgani 2004)

Support vector machines is a binary classification model. It is a linear classifier with the largest interval defined in the feature space. The interval distinguishes it from perceptron. The basic idea of SVM learning is to solve the separation hyperplane and correctly divide the training data set, at the same time, maximize the geometric interval. The property of SVM: after the

training is completed, most training samples do not need to be retained, and the final model is only related to the support vector. SVM has been widely used in many industries, for example intrusion detection, hand wring recognition, breast cancer diagnosis and hyperspectral image classification Guo's (2019) work possessively. Figure 2.1 shows the regression results of SVM algorithm, Table 2.2 displays SVM algorithm advantages and disadvantages.

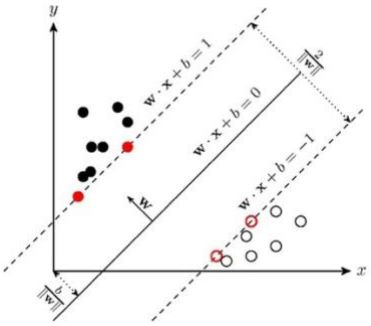


Figure 2. 1 SVM algorithm principle

Table 2. 2 SVM algorithm advantages and disadvantages

Algorithm	Advantages	Disadvantages
SVM	The optimal hyperplane of feature space partition is the goal of SVM, and the idea of maximizing classification margin is the core of SVM method	Variable weights and different impact
	Suitable for small number of sample data, can solve high-dimensional problems	It's hard to find a good kernel function value
	The final decision function of SVM is determined by only a few support vectors	Difficult to choose the value of parameters

2. K-nearest neighbours (KNN)

K -Nearest Neighbour (KNN) algorithm is a classical machine learning supervised learning algorithm. The main idea is given a test data, if most of the K training data closest to centre

(always choose mean value as centre or choose target value), this can belong to a certain category, other data can be divided to another to category.

If we choose a smaller value of k, it will mean the model will become complex and performance overfitting. Overfitting means that high accuracy was obtained in the training data set and low accuracy was obtained in the test data set. If a larger k value has been chosen, it is equivalent to using the training data in a larger neighbourhood for prediction. The training instance that is far away from the input instance will also play a role in prediction, making the prediction wrong. With the increase of k value means that the simply the model. The key point to select k value is to adjust experimental parameters and obtain a better result by adjusting hyperparameter. In summary, KNN has several advantages and disadvantages (Table 2.3)as following:

Table 2. 3 KNN algorithm advantages and disadvantages

Algorithm	Advantages	Disadvantages
KNN	easy to implement	Spending long time on training large data sets
	Effective if the training data is large	Parameter K value required to be decided
	Has no assumptions	Imbalanced data can cause bias

3. Back Propagation neural network

The learning process of BP (Back Propagation) neural network consists of two processes: signal forward Propagation and error Back Propagation. In the forward propagation, input samples are transmitted from the input layer to the output layer after being processed layer by layer by the hidden layer. If the actual output of the output layer is inconsistent with the expected output, the reverse propagation stage of the error will be turned. The back propagation of error is to transmit the output error to the input layer by layer through the hidden layer in some form and distribute the error to all the units in each layer, so as to obtain the error signal of the units in each layer. This error signal is the basis for correcting the weight of

each unit. Learning process: Under the stimulation of external input samples, the neural network constantly changes the connection weight of the network, so that the output of the network is constantly close to the desired output. The nature of learning: Dynamic adjustment of each connection weight. Learning rules: Weight adjustment rule refers to a certain adjustment rule based on which the connection weight of each neuron in the network changes during the learning process. Table 2.4 shows advantages and disadvantage of BP neural network algorithm:

Table 2. 4 BP neural network algorithm advantages and disadvantages

Algorithm	Advantages	Disadvantages
BP neural network	The network essentially implements a mapping function from input to output	The learning speed of BP algorithm is very slow
	No need for impose restrictions on the input variables	It is difficult to solve the contradiction between instance scale and network scale of application problems
	The network can automatically extract "reasonable" solution rules by learning the instance set with correct answers, that is, it has self-learning ability	The newly added samples must affect the learned successful networks and the number of characteristics that characterize each input sample must be the same

2.2Neural networks

2.2.1 Image Brief history of neural network development

The development history of neural network (shown on Figure 2.2) can be describe as following: History traces back to the 1943, McCulloch & Pitts introduced a logical calculus in nervous activity. Neural network became popular in the 80's with work by Rumelhart, Hinton, and Mclelland. Neocognition and Convolutional neural networks were proposed in 1980 and 1998

respectively. Neural network was reached peak in the 90's. Nowadays, neural network model own hundreds of variants and some aspects are less competitive than actual brain but turn into a useful tool.

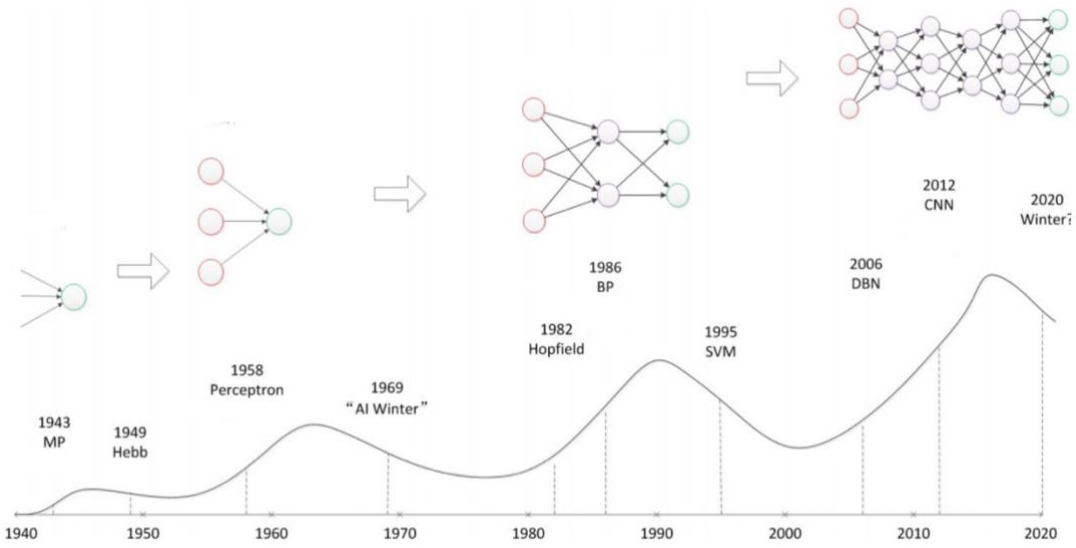


Figure 2. 2 Neural network history

2.2.2 Main application fields

Neural network is applied in different fields, its applications are used in many aspects of real life, such as sales forecasting, identify face, image recognition and risk management. The way it works is also very interesting.

- Autonomous driving cars: distinguish different types of objects, people and road signs. automatic noise removal and training a vehicle to drive by itself.
- Health care: Breast and skin cancer diagnostic have been achieved by deep learning. Artificial intelligence is completely reshaping the medicine and healthcare industries. Innovations in artificial intelligence are advancing the future of precision medicine and population health management in a brand-new way.

- Automatic hand wring generation: Given a sample handwriting corpus, generate a new handwriting for a given word or phrase. Either letter or digits can be recognized.
- Stock market prediction: Many factors influence a stock price. Neural networks can filter much information quickly and make a ranking based on the weights. So, it is used to predict stock prices.
- Retail and sales: The neural network can consider multiple variables simultaneously, such as the market demand for a product, the customer's revenue, population, and product price. It is good for predicting sales in supermarkets.
- Human face recognition: It is a biometric method for recognizing a human face. If a neural network is well trained, it can be divided into two categories, an image of a human face and an image without a human face.

2.2.3 Main Implement method and achievement

Convolutional neural network is a represent deep learning model. This model has input layer, the convolution layer, the pooling layers, the fully connected layers and the output layer. Convolutional layer is used to extract features from multi dimension data based on the convolution kernels of different size. Pooling layers is to gradually reduce the size of the representation space, reduce the parameters and calculations in the network, and thus control over-fitting. Fully connected layers combine the outputs from all previous layer input, each layer contains previous layer all information. In summary, CNN can achieve automatic high-level feature learning and provide an effective classification. (Zhou, 2019)

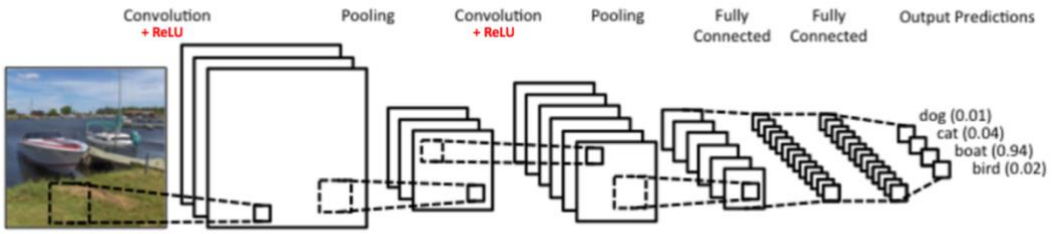


Figure 2. 3 Convolutional neural network principle

When only a small amount of data is available, it is usually used data augmentation to solve this problem. Data augmentation can generate similar data samples into the training data set by label preserving transformations. If image data set is being used, add new images can be applied by clipping, flipping, rotation, cropping and shifting the image data, sometimes add some random noise to RGB channels can increase classification accuracy as well. This process can provide vary samples for the neural network.

CNN can be divided into the following specific implementation methods.

- Siamese convolutional neural networks contain two or more identical subnet components. (Bromely, 1994) The Siam network might look like this (Figure 2.4):

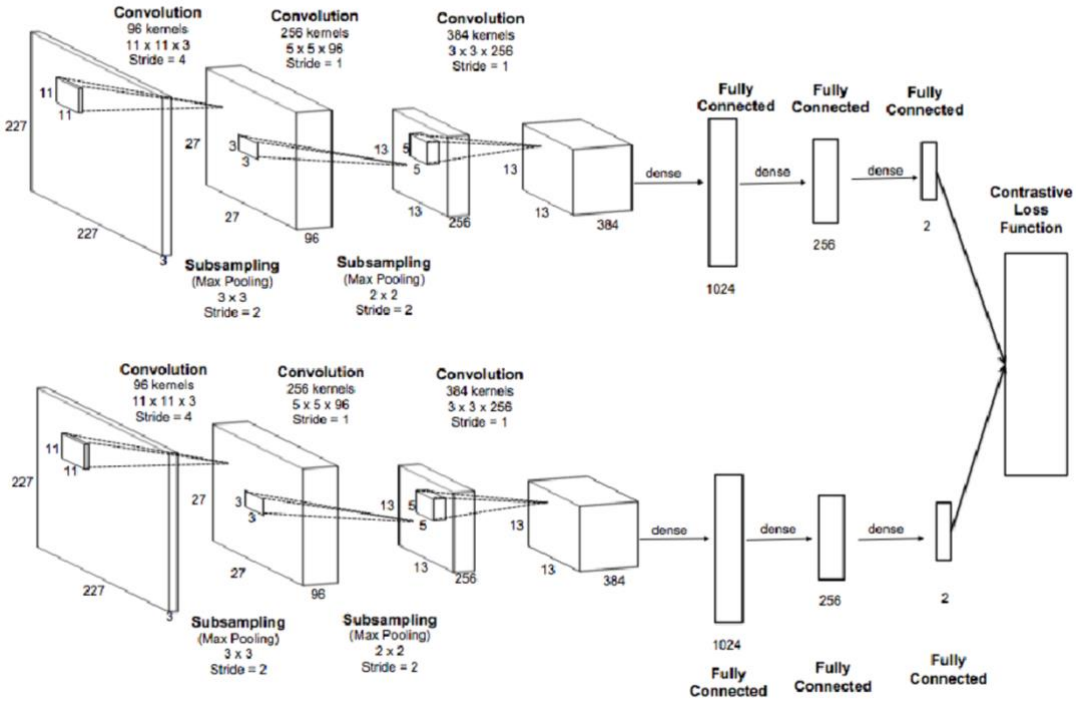


Figure 2. 4 Siamese convolutional neural network

The main idea of the Siam network is that they can learn useful data descriptors that can be further used to compare the inputs of individual subnetworks. Therefore, the input can be any digital data, image data, or even sequential data, such as sentences or time signals. This method proved the effectiveness for application at image classification task. (Zhou, 2019)

- MLP-CNN

  MLP-CNN is a new method with deep learning algorithm which employ pixel-based multilayer approach shallow structure. This model is the most advanced deep learning methods. Xin's (2019) work add decision fusion method to help increase classification accuracy. The process of this work is to obtain spatial features by use CNN and obtain spectral discrimination by use MLP method.

- Naive Bayes data fusion scheme (NB-CNN) (Huang,2018)

In order to analyze the crack detection of a single video frame. There provided a data fusion model, which is used to extract information from video frame result in improve the robustness performance of the system. A CNN to detect crack patches is produced to detect video frame, and the proposed data fusion scheme maintains the spatiotemporal consistency of cracks in the video, and the naive Bayesian decision can filter the false noise. The frame's hit rate of 0.1 false positives per frame is 98.3%, this provided this method is able to obtain higher accuracy than other method.

- Attacking convolutional neural network (Su, 2019)

  Antagonistic attacks include subtly modifying the original image in a manner that is barely noticeable to the human eye. The modified image is called an antagonistic image and is classified incorrectly when submitted to the classifier, while the original image is correctly classified. The CNN classifier can obtain either a specific or arbitrary feature through adding the adversarial perturbation. Differential evolution (DE) is the most important weight in all parameter. DE is been modified to obtain the best classification. This process obtains a success way of image classification by conducting the adversarial perturbation at a specific range value.

  There are many previous attacks which generate adversarial perturbation

  After receiving an addition number of pixels which may take a risk.

- LISA-CNN
- GTSRB-CNN
- Deep Neural Networks (DNNs)

  Neural networks are techniques that mimic the activity of the human brain, specifically, pattern recognition and input through different levels of simulated neural connections.

  A deep neural network is defined as a network having an input layer, an output layer, and at least one of the hidden layers. Each layer performs a specific type of sorting and sorting in one process, some call it an "attribute hierarchy." “A key use of these complex neural networks is to process unlabeled or unstructured data. The term “deep learning” is also used to describe these deep neural networks, as deep learning represents a specific

form of machine learning. In this form, techniques using artificial intelligence are sought to classify and rank information in a manner that goes beyond simple input/output protocols.

Dl is based on a neural network, which simply means that the input is training data, the middle is hidden node, and the output is tag. Training data can activate the hidden node with a certain weight, and the hidden node can activate the output tag. The creativity of dl lies in the self-encoding and sparsity. Self-encoding means that dl treats the input as training data and the output as training data. In this way, if the number of intermediate nodes is relatively small, it is equivalent to the compression of data. The other one is sparsity, that is, if I want more intermediate nodes but do not want the training to be too overfit, I can add a limit, assuming that each training data activates as few intermediate nodes as possible. In this way, the trained model can be unsupervised and play the role of dimensionality reduction. The trained model can express the original data well. At this time, we can meet the new demand by further training on the trained data and the supervised data.

- Recurrent Neural Networks (RNN)

  A cyclic neural network is a neural network transmitted over time. The depth of the network is the length of time. This neural network is specially used to deal with the problem of time series and can extract the information of time series. If it is a forward neural network, the neuron signals of each layer only be pass forward to the next layer, and sample processing is independent in time. For the cyclic neural network, the output of the neuron at this moment can directly affect the input at the next time point, so the neural network can deal with the problem of time series (Figure 2.5).

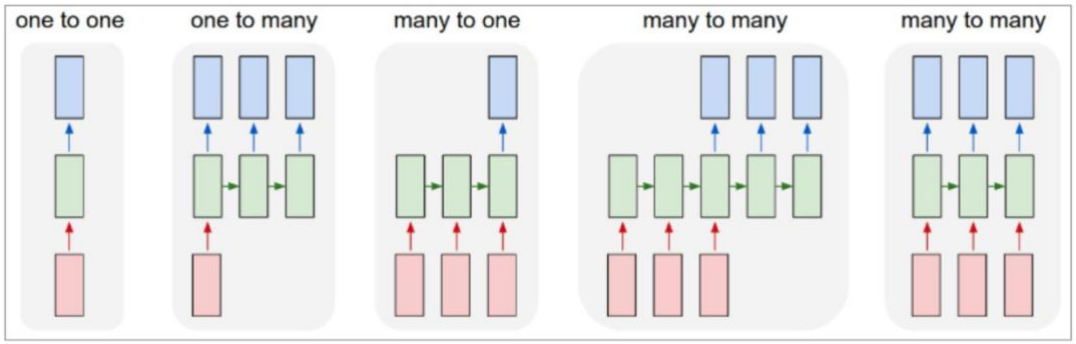


Figure 2. 5 Mapping perform at RNN

A significant limitation of common neural networks (and convolutional networks) is that their APIs are too restrictive: they accept fixed-size vectors as input (such as images) and generate fixed-size vectors as outputs. Not only that: these models perform mappings using a fixed number of computational steps, such as the number of layers in the model. The core reason why recursive networks are more exciting is that they allow us to manipulate vector sequences: input sequences, output sequences, or both in the most general case. Each rectangle is a vector, the arrow indicates that the function. The input vector is red, the output vector is blue, and the green vector keeps the state of the RNN from left to right.

2.2.4 The way to make classification on street network classification

The main objective of this work is to develop an easy - to - operate method for road extraction of satellite images. At present, all the work of road extraction is based on morphological operation of neighborhood pixel group. The limitations of these methods are a lack of generality, limited success rates for datasets, terrain, and spectral range variations. Machine learning has come a long way in recent years and is widely known. Computer vision is an area where machine learning is widely used and has a commendable success rate. Humans are able to self-identify and estimate roads when they see aerial photographs. There are certain visual patterns that distinguish pixels to be classified as part of the road. So far, there is no mathematical classifier to define the above model. In fact, these patterns are hard to define,

but are intuitive and accessible to humans. It is this intuitive model that proposes artificial intelligence. In turn, artificial neural networks serve as forward-looking solutions to roadway extraction problems. The initial method is to collect and extract the satellite view and the corresponding road view data set.

After starting the study, two questions need to be identified.

1. It is not available, and not easy to make data sets large enough, work must divide into sub-sessions.

2. CNN with input and output as complete images is too complex and expensive, which makes it difficult to represent and train neural networks with available computers. Based on the above work, there can produce a better solution. This problem has been improved to estimate whether the pixel is part of the road based on the color profile of the neighborhood of interest pixel. Either the image model is simplified, representative eigenvectors are used, or some minor factors are ignored. This simplifies computation and saves time and storage. Since the input is a small block of pixels, roughly an array of 1,000 digits, the artificial neural network has been thinned considerably. Using the new method to output the artificial neural network has become a scalar. After applied this method, a single different size image can generate about one million training cases.

Data sets are important for any classification purpose, such as road detection. The first step is to get satellite images from the repository. The next step is to take the Annotated Image, which helps generate the dataset for training purposes. Image annotations are also known as image tags, where the technique is used for systems that need to retrieve images to identify and classify images from interested databases. This is also a classification technique. Generate data sets from these image sets.

Choosing the best configuration for your network is a trade-off. Large networks take a long time to train and can become overstrained. Overtraining is a problem where networks are trained for a given data set rather than generalized learning. Small networks are often trained quickly but cannot classify complex nonlinear situations.

Deep convolutional neural networks provide efficient and highly accurate results for road classification from satellite images. It's also an efficient way to do classification in satellite imagery. Roads in satellite images are identified at the pixel level. The convolutional neural network can process images of any size, whether it represents a small area or a very large area, so it can be tested under images of different sizes. Depending on the type of satellite sensor, the type of image obtained varies widely. Convolutional neural networks are flexible in that they have adaptive layers that can connect the major components that exist in the network. It can analyze and identify satellite images and output a set of predictions. The original satellite image set will be sampled down with the help of a pool layer associated with the set. The last output in the summary layer output is sorted with the help of a fully connected fixed-size layer. Since the size of the output image must be consistent with the input data, this may limit the application in network training, as well as the size of the input and application images. With the help of the convolutional layer, the restrictions on fully connected layers can be lifted.

The gradient descent algorithm is easy to implement, and the convergence speed is fast, but it is easy to concentrate on an optimization result, which lead to gradient come close to the saddle point drops slowly, which influence the training of the network model. the training of Newton's algorithm can prevent this kind of things happened, but this method needs to calculate the Hessian matrix to provide its non-negative positive definite and bunch of storage space and have conditional restrictions.

Two methods are used to classify the output. One is the binary partition method, the other is the regression method. In the first method, zero space tolerance is associated with binary segmentation. The scheme of this method is to identify the pixels in the image as roads or backgrounds. In the second method, the regression method is associated with an adjustable space tolerance. The idea is to give a uniform target distribution along the center of the road tag, which has a maximum value of 1 allocated to the road tag, and which uniformly reduces to zero for the background other than the road. Wherever the regression method is applied, the original prediction can be classified by SoftMax function when binary partition method or s-shaped function is executed. Either of these methods produces a map of each data point with a confidence score varying between the values 0 and 1. The confidence score value indicates

that the value closer to 1 indicates that the corresponding pixel execution really belongs to the road, while at the other extreme, the confidence score value is close to 0, indicating that the corresponding pixel represents the background respectively. The SoftMax function mentioned above does provide two different confidence values for each pixel.

The image can be observed from different multi layers. This reflects the real area from the original image and provides good revolution for the picture. The way to extract features from the image which will be based on the wavelet packet feature. There are two potential methods for fusion of wavelet transform. The first method is the pixel fusion. The second is thronging local window measurement. Large absolute value coefficient corresponds to the image, which always own large amount of constant data. For human, it is quietly easy to identify such the edge features of image. Therefore, the maximum selection least squares method is adopted to solve the problem.

It must be noted that when the road network in the satellite image is small enough, it is likely to be assimilated compared to the background category. This is a real problem that must be overcome. Appropriate steps must therefore be taken to limit class imbalances at the training stage. This means that the label must completely cover the outline of the road and its embankment. Therefore, the effective way to reduce the imbalance is to increase the spatial tolerance of the corresponding regression model. Another way to handle this problem is to recalculate each class based on the loss calculation. The idea is to multiply the predicted loss of pixels by the value of the coefficient inversely proportional to the real level of ground reality. Note that these coefficient values are calculated using the median class frequency values. Pixels exist in ground truth.

Chapter 3

Methodology

3.1 Introduction

In this chapter, the Path Model will be presented first, followed by the introduction of the core project objectives. The path model is divided into four parts, targeting problems, analysis, test, and evaluation. This can help approach problems of map image recognition, such as the map recognition input system. An important step in processing the image data is to understand the neural network system output and the pattern of each layer processed that results in the final prediction and accuracy. In the beginning, there is a problem statement from the literature review and research questions.

3.2 Problem statement

Traditionally, the structure of the street network is analyzed with a supervised machine learning method, which needs to collect various features by calculation and modification of the parameters and weights related to these features, and finally to use regression algorithms to calculate and generate the classification results. In this process, errors can be caused by the way of feature numbering, collection, choice, and management. Moreover, the application of some algorithms may lead to an overfitting outcome. Correspondingly, this procedure requires a large amount of manual time.

3.3 Target solution

The establishment of the Path model can help the carding process to approach the target problem. It is considered that the target results can reach a higher accuracy and minimize the error. For this project, deep learning models are adopted to tackle this problem. Some people strongly advocate that the use of deep learning methods can manage these problems. First, deep learning methods can deal with raw pictures and require no labeling for data, thereby reducing the error from features caused by human factors. Second, it is easy to avoid overfitting by using a deep learning model. Third, training for a certain period of time can achieve results with high accuracy. The following Path model will describe the framework of the entire process.

3.4 PATH model

3.4.1 Development Environment

This experiment was programming under the Jupyter notebook using python3 environment. The followings list (Table 3.1) is the correlated hardware which supports the model running:

Table 3. 1 Software and hardware required

Name of Hardware/Software	Version
Python3	3.7
Central Processing Unit (CPU)	
Graphics Processing Unit (GPU)	
Tensorflow-gpu	1.13.1
GCC	4.8
Bazel	0.19.2
CUDA	7.4
cuDNN	10.0

NVIDIA-SMI	418.67

3.4.2 Data generation

OSM API packages allow python3 to access the OpenStreetMap database directly. Figure 3.1 shows the procedure of database establishment. First, it confirms the locations of 10 cities, which should be some representative cities, such as national capital city, large area cities, large population cities, or some cities with special geographical locations. Second, based on the city locations, each city has its own boundary range, and some coordinates from each city are selected. Meanwhile, the values of the parameters are set, such as the scale of city figure, map layers and the numbers of map figure. Third, these coordinates and parameters are input to the OpenStreetMap API package to generate the target database. Finally, according to the visualized figures, parameters are modified to collect valid map figures.

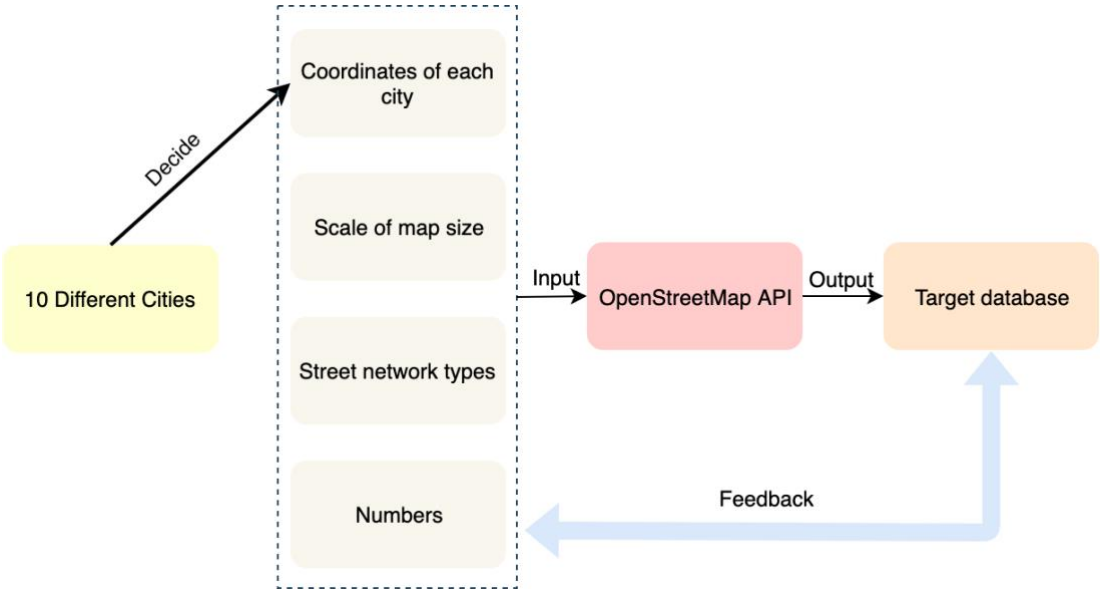


Figure 3. 1 Procedures of database establishment

3.4.3 Training model

Deep learning (DL) model has been widely used for image classification, object detection, automatic handwriting generation and music composition and so on. It can process raw data like pictures or videos directly. Especially when facing a large amount of feature introspection, it can reduce much time and data engineer work.

Nowadays, deep learning has been applied in many fields, such as human face skin disease recognition (Chen et al, 2020). This system is not only applied to professional medical services but also for private use by personal mobiles. First, it is to access the skin database and then finish skin analysis, emotion analysis and health analysis through the training model and finally, a conclusion is drawn by the doctor. A recent wildlife classification case by using deep learning (Miao et al, 2019) provides realistic steps for helping ecologists, which is based on detecting edge from visual features on each image. Moreover, deep learning has been used for recommendation engine or text sentiment analysis through regression algorithms.

Application steps

CNN is not only applied in the field of the image but also in speech and natural language processing. The common feature of these data is the local correlation. It can encompass both local features and global features. Moreover, it can learn new features from the image database as well. The way how the human brain works is shown in Figure 3.2.

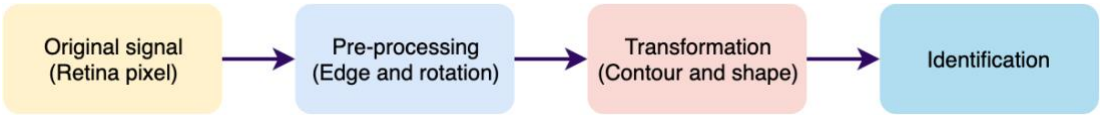


Figure 3. 2 The way how human beings recognize objects

Consequently, simulating the working mechanism of the human nervous system has been proved an effective method in machine learning. The feature learning system of the neural network routine adopts the multi-layer network structure to deal with the original database, by employing the end-to-end training, which can learn the desired

result from the raw sample dataset directly, and each layer extracts some features from the upper output. Figure 3.3 shows the classification procedures developed from traditional machine learning method to neural network system.

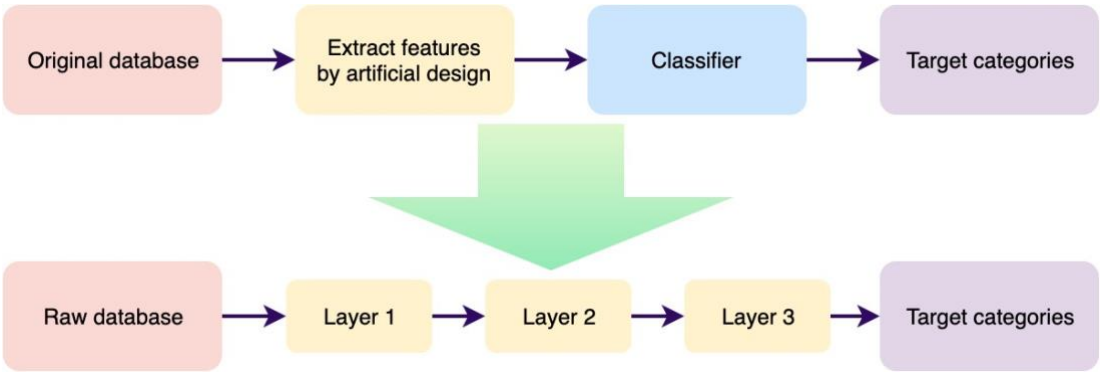


Figure 3. 3 Classification procedures developed from traditional machine learning method to the neural network system

In computer language, an image can be treated as a grid of numbers which represent the color of each pixel. A typical example is the MNIST database (Figure 3.4). If the figure size is 18×18 pixels, it can be regarded as an array of 324 numbers. After the input of these 324 numbers into the neural network nodes, an output of number values can be obtained. When the requirement to judge a certain number value is met, there will be two output outcomes, namely ‘True’ or ‘False’, and objects will be classified into groups (Figure 3.5).

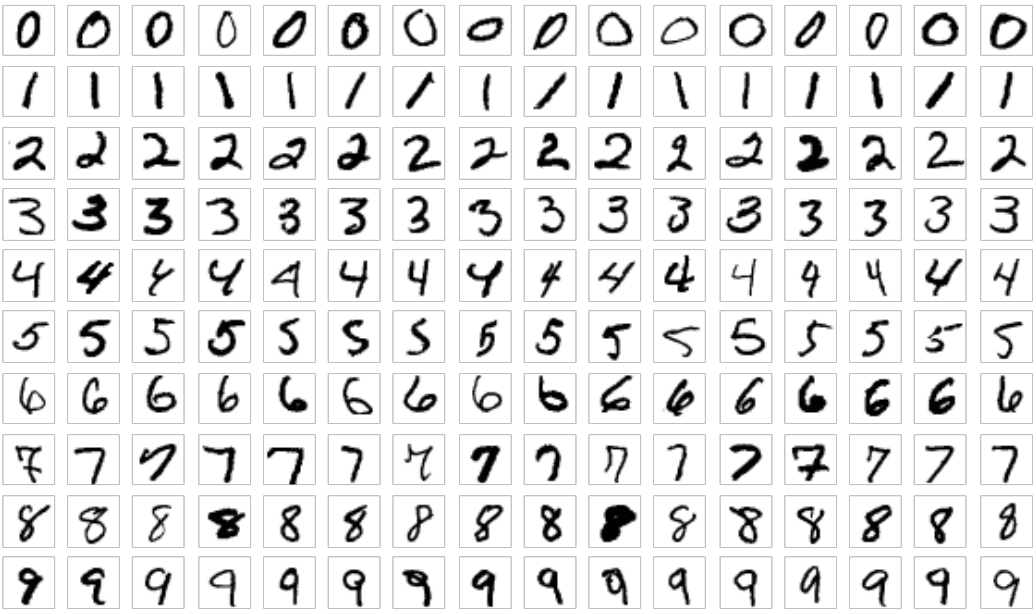


Figure 3. 4 MNIST database

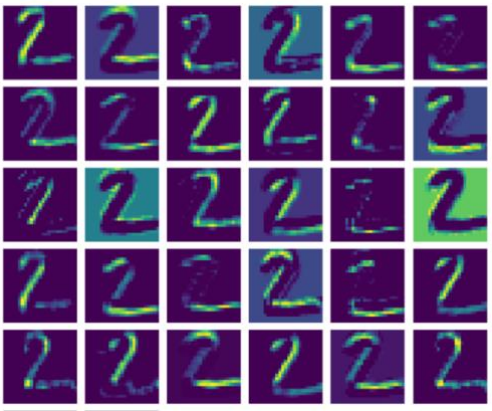


Figure 3. 5 Visualization of output results

Convolution

In practical application, it always needs data argument to increase dataset distinctiveness, such as flip, rotation, rescale, and mirror flip and so on. Besides, the whole figure can be searched with a sliding window of a certain size for identifying useful edges. By using these methods, convolution can do better effective work on image recognition (Figure 3.6). First, convolution

divides original images into sub-images with determined padding size. Second, each single sub-image is input into the neural network for pattern analysis. Third, analysis results from all sub-images are recorded as arrays or matrixes. Forth, because the array obtained will be huge, in order to simplify the following calculation, the max value within the settled grid square is chosen. This step can help maintain the most important information while reducing the size of the output array. Fifth, the final neural network needs to judge the image real number value, and sometimes it is called 'fully connected' while making a decision based on all sub-image calculation results.

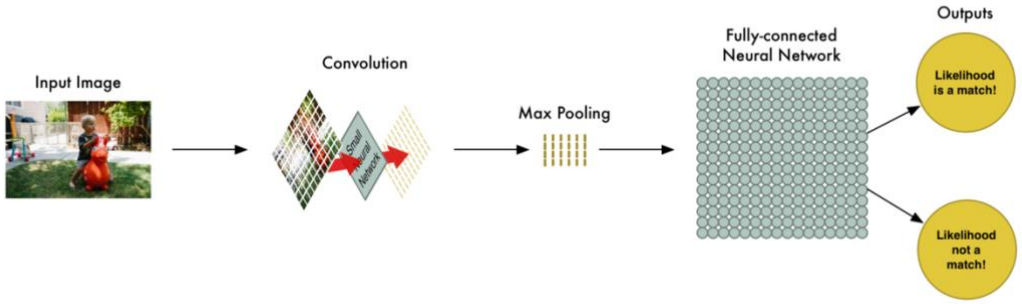


Figure 3. 6 Convolution procedure

Internal structure (layers) of deep learning

At this project, the classification model will be built based on the Convolutional Neural Network (CNN) algorithm. Generally, CNN has three functional layers:

1. Input layer
2. Feature-extraction layers
3. Classification layers

Feature-extraction can be divided into four parts:

1. Convolution layer (CONV layer)
2. Rectified Linear Units layer (RELU layer)

3. Pooling layer (POOL layer)
4. Fully-connected (FC) layer

Specifically, the input layer always accepts the original pixel values of an image, if the size of image is 32*32*3 in width, height and color channel values separately.

The CONV layer handles a convolution filter (kernel) of size 5×5×3 to and forward pass until all images are convolved and a group of feature maps are generated. The size of the convolution filter is set in a specific height and width value, with the depth value of the convolution filter equal to the image depth value. Hence, the convolution filter has the same dimension as the input image. One feature map size will be 32×32×1, with the height and width value depending on padding. If there are n numbers filters, the output will be a volume of 32×32×n feature maps. The following list shows the hyper-parameter in the convolution filter.

- Filter (Kernel) size (field size)
- Output depth
- Stride
- Padding

The RELU layer is always used for nonlinearly mapping the results of the convolutional layer, and the resulting values can be 0 to 1 or -1 to 1. Basically, there are two categories of the activation function:

1) Linear activation function
2) Non-linear activation functions

Activation function and Loss function definition (Table 3.2)

The following table provides some commonly used activation functions and their mathematical expression. Figure 3.7 shows activation function curves.

Table 3. 2 Activation function equation and range scale

Name	Equation	Range
Linear	$R(z,m) = \{z \times m\}$	$(-\infty, \infty)$
Sigmoid	$S(z) = \frac{1}{1+e^{-z}}$	(0,1)
Tanh	Tanh(z) = $\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$	$(-1,1)$
ReLu	$R(z) = \begin{cases} z, & z > 0 \\ 0, & z <= 0 \end{cases}$	$[0, \infty)$
LeakyReLu	$R(z) = \begin{cases} z, & z > 0 \\ az, & z <= 0 \end{cases}$	$(-\infty, \infty)$
SoftMax	$f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \ for\ i = 1, \dots, J$	$(0, 1)$
Gaussian	$f(x) = e^{-x^2}$	$(0, 1]$

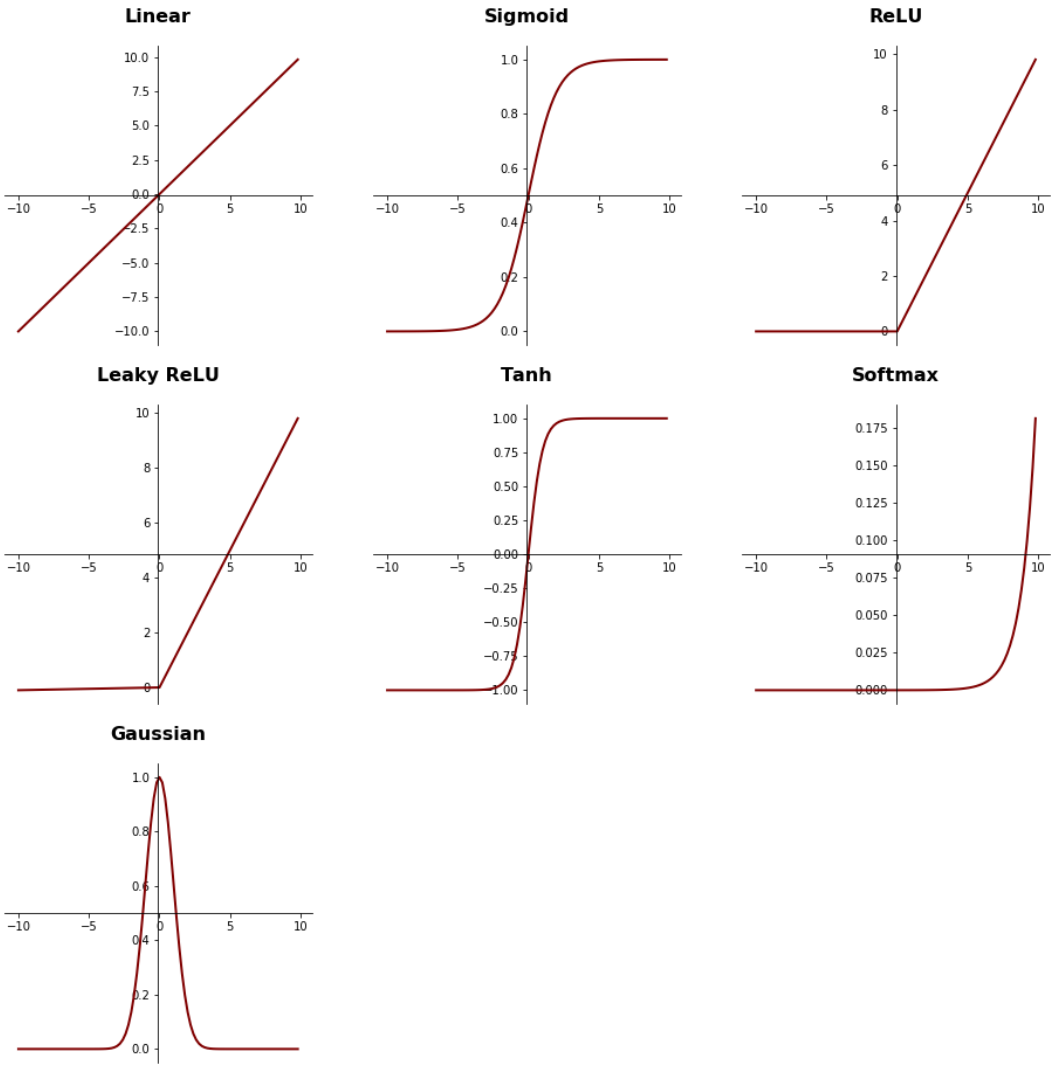


Figure 3. 7 Activation function curves

The pooling layer is used for down sampling and decreasing the spatial dimensions. Overall, the location feature has lower importance than other features, which is to summaries the average or max value from the output of the last layer and keep the most important features of the input image. Hence, the output of the pooling layer can gain fewer spatial features, resulting in reduced over-fitting chances and reduced numbers of parameters.

There are 3 common methods on the Pooling layer:

1) General Pooling

   It is defined that the horizontal displacement/vertical move of two adjacent pooled windows is named stride. The pooling region has the same size as stride, which can prevent overlapping. There are two ways for calculating the pooling value, mean pooling and max pooling.

   - Mean pooling: The average value of the selected feature map area as the pooled value of the area.
   - Maximum pooling: The maximum value of the selected feature map area as the pooled value of the area.

2) Overlapping Pooling (Krizhevsky, Sutskever and E. Hinton, 2012)

   If the pooling region is set smaller than the stride size, there is an overlapping pooling layer. It shows that the overlapping pooling can make overfitting difficult.

3) Spatial Pyramid Pooling (He et al, 2015)

   Spatial pyramid pooling can transform the convolution feature maps from images of any size into the same dimension, which can avoid cropping and warping and allow CNN to process images of any scale. Images are convoluted and converted into feature maps of the same dimension into the next fully connected layer, which expands the limits of CNN use in any size of images.

Finally, fully connected layers accept the output of the last layer and make the classification, which can be the class probabilities or accuracy scores. These layers are fully connected to every neuron from the last layer. The outcome of fully connected layers brings out two-dimensional output in the dimensions b×N, where b represents the number of sample batch and N is the number of target classes. It can be classified by SoftMax regression, which can be called a SoftMax layer (SoftMax layer) as well.

Forward propagation procedures (Figure 3. 8)

It is related to the image propagation which passes the features from the input layer to the output layer, that is, the order of occurrence is to pass the stimulus from the previous layer to the next layer.

It can be expressed as equation 3.1

$$a^2 = \sigma(z^2) = \sigma(a^1 \times W^2 + b^2) \tag{3.1}$$

where superscript represents the number of layers, the asterisk indicates the convolution, b represents the bias term bias, and σ represents the activation function.

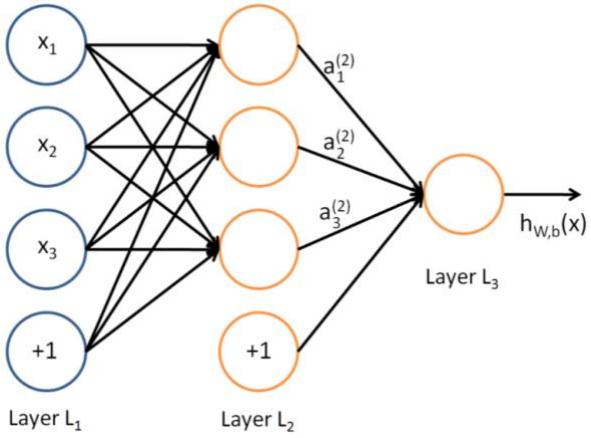


Figure 3. 8 Forward propagation procedures

3.4.4 Classic deep learning models

The convolutional neural network has advantages in speech and image recognition with its special structure of local weight sharing. It comes closer to the practical biological neural network, and weight sharing reduces the complexity of the network, thereby avoiding the complexity of feature extraction and classification. The following are several classic CNN architecture algorithms.

1. LeNet-5 (1998) (Figure 3.9)

   It is used to classify hand-written digits on bank cheques. LeNet-5 is one of the earliest convolution neural networks. It has seven layers, including three convolutional layers

(C1, C3 and C5), two pooling layers (S2 and S4), one fully connected layer (F6), and lastly, one output layer.

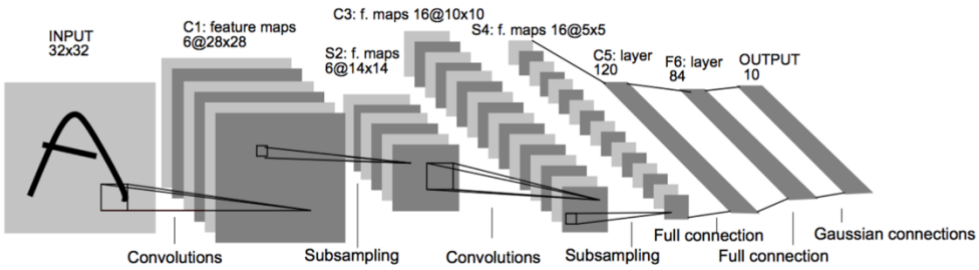


Figure 3. 9 An illustration of the LeNet-5 architecture (LeCun et al, 1998)

2. AlexNet (2012) (Figure 3.10) (Krizhevsky, Sutskever and E. Hinton, 2012)

   AlexNet finished the ImageNet Large Scale Visual Recognition Challenge in 2012 (AlexNet on Wikipedia, 2019, para.2). It realized the reduction of the top-5 error rate by 10% when compared with the winner in the last year.

   The network of AlexNet is increased (5 convolutional layers + 3 fully connected layers + 1 SoftMax layer), while over-fitting is solved through data augmentation and the dropout method. Furthermore, AlexNet can be accelerated on multi-GPU.

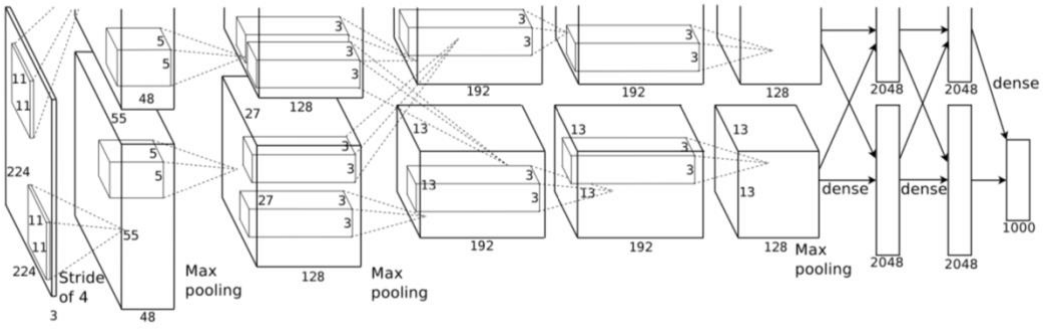


Figure 3. 10 An example of AlexNet structure

3. Visual Geometry Group (VGG16/VGG19) (Simonyan and Zisserman, 2015)

The original purpose of VGG is to study the depth of the convolutional network and find the way depth affects the accuracy of large-scale image classification. VGG is called a very deep convolutional network (GG-Very-Deep-CNN). To avoid too many parameters in the calculation, VGG adopts 3x3 convolution kernels for all layers and sets the stride equal to 1.

VGG16/VGG19 represents the numbers of weight layers. VGG16 (shows on Figure 3.11) has 16 layers including 13 convolution layers and 3 fully connected layers, and VGG19 has 16 convolution layers and 3 fully connected layers. The following picture shows the structure of the VGG model.

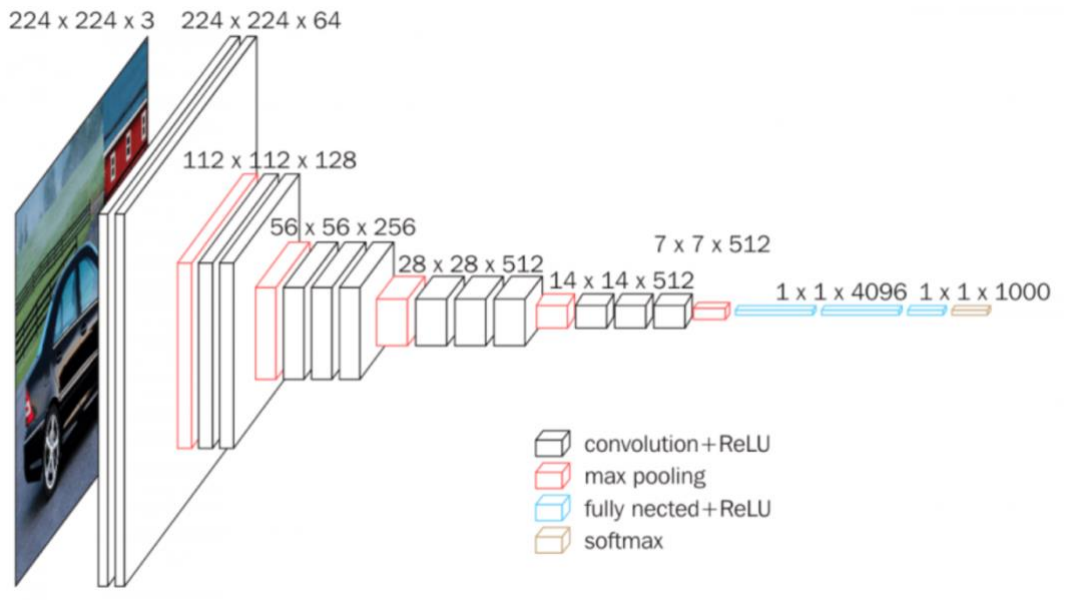


Figure 3. 11 The architecture of VGG16

4. Residual Neural Network (ResNet) (He et al, 2015)

Sometimes, as the network goes deeper, the accuracy of the training dataset comes down. ResNet is introduced to deal with this problem by applying skip connections or short-cuts to abandon some layers (Figure 3.12). The aim for skipping layers is to avoid the disappearance of gradients by repeating prior layer activations until neighboring layers gain their weights. By passing the input information to the output

and protecting information integrity, the entire network only needs to learn the part of the input and output differences, thereby simplifying the objective and procedure.

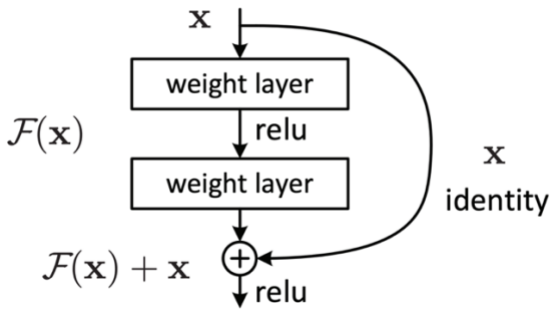


Figure 3. 12 The process of Residual Neural Network

3.5 Research Objectives

This project is intended to pick up ten cities and generate their street network pictures for recognition. The deep learning model is built for realizing identification and classification of different cities around the world based on the deep feature pattern. In the meantime, this project will try to evaluate the applied method on a different scale of cities in distinctive styles and determine the performance of these image levels.

Chapter 4

Experiment design and implementation

This chapter will describe the way of approaching the objectives of this project. There will be four main parts: first, requirement analysis and specification; second, the introduction of the detailed design and implementation of each step; third, scientific control for establishing several compared groups; last, conclusion and critical evaluation.

4.1 Neural networks

4.1.1 Jupyter Notebook

The Jupyter Notebook is an open source web application which allows editing and creating notebook documents. It provides over an environment with 40 programming languages, which contains Python, R and Scala. Once Jupyter Notebook App has been installed, it can be carried out on local operating systems without an internet connection. Jupter Notebook provides an environment which allows users to add text, mathematical equations, code and visualized data into one share document. This makes it a handy tool for executing end to end jobs comprising data engineering, statistical modeling, producing and training machine learning models and so on.

4.1.2 OpenStreetMap

OpenStreetMap (OSM) is an online map collaboration program that aims to create a free world map which can be edited and accessed by everyone. At the beginning of OpenStreetMap, many

data are collected by volunteers using GPS tracking records and other digital equipment in field exploration. The geodata produced by the map is considered as the main output of the OpenStreetMap.

OpenStreetMap uses topological data structures with four core elements: Nodes, Ways, Relations and Tags (pairs of a key and a value). Nodes define the geographic positions as coordinates (a couple of latitude and longitude). Ways define polyline and polygon which can represent linear features like roads, railway tracks or rivers, and serve like farmland or forests which denote areas with specific boundaries. Relations describe the relationship between Nodes and Ways in an order list, such as a number of existing ways for the turn limit extent of long-distance highways. Tags describe the precise features of Nodes, Ways or Relations and change sets which are stored as metadata.

4.1.3 TensorFlow and Keras package

TensorFlow is an open source library that provides an environment to build and test machine learning algorithms. TensorFlow has become the most popular open source matching learning framework. It owns fast, flexible and highly extensible characters which make it convenient for developers and researchers to deal with new challenges. It has a multi-layered design that can be arranged on multi-choice servers, such as PC, web pages and high-performance CPU, GPU and TPU for numerical calculations. Currently, it has been widely used in machine learning and artificial intelligence research in product development and scientific research in numerous fields.

Keras is a neural network API library in Python which can be used quickly. It is able to run under the TensorFlow. Keras also brings a straightforward approach to build models for beginners and researchers. Users can execute common operations of neural network building, containing layers, activation functions, loss functions, optimizers and other tools, such as wrappers for the Scikit-Learn to allow working with big data in much easier way.

4.2 Detailed design and implementation

The implementations can be divided into 3 modules.

4.2.1 Street network data generation

OpenStreetMap provides API for accessing the OSM database. It is stored in an XML format. OSMnx is a python package that allows users to download spatial geometries and core elements.

It is created based on geopandas, network and matplotlib. As a result, it enables users to analyze and visualize street network features from OpenStreetMap's API.

Street network data can be downloaded through the OSMnx including the following information:

- Place boundary definition, such as city, country, province, zone, and street block
- Polygon of the specific street network boundary
- List of place names

OSMnx also allows users to customize the network types (shown on Figure 4.1):

- drive - public roads
- drive_service – all drivable roads
- walk – all roads and routes all people walk
- bike - cycle track
- all - all roads and routes
- all_private - all roads and routes, including private-owned paths

For instance, the centre of London, Westminster Block Street can be visualised in different network types.

Listing 1 Westminster street network visualisation using OSMnx library

```
G = ox.graph_from_place(
'Westminster, Greater London, United Kingdom', network_type='drive'
)
fig, ax = ox.plot_graph(
G,fig_height=8,bgcolor='white',node_color='lightpink',edge_color='li
ghtslategrey', save=True, show=False,close = False,
file_format='png', filename= 'Westminster_drive', edge_linewidth=0.9
)
```

Figure 4. 1 Westminster of London street network visualized in different types of the street (top to bottom, left to right: drive, drive_service, bike, walk, all, all_private)

The Interactive Nolli Map Website can be used to explain in detail the map itself and serve as an important tool for cue measurements, presentation of the accurate line of sight and a full view of the map, in addition to displaying various map modes. Figure-ground diagram presents connections of different places. It is similar to the Nolli map, displaying the network pattern.

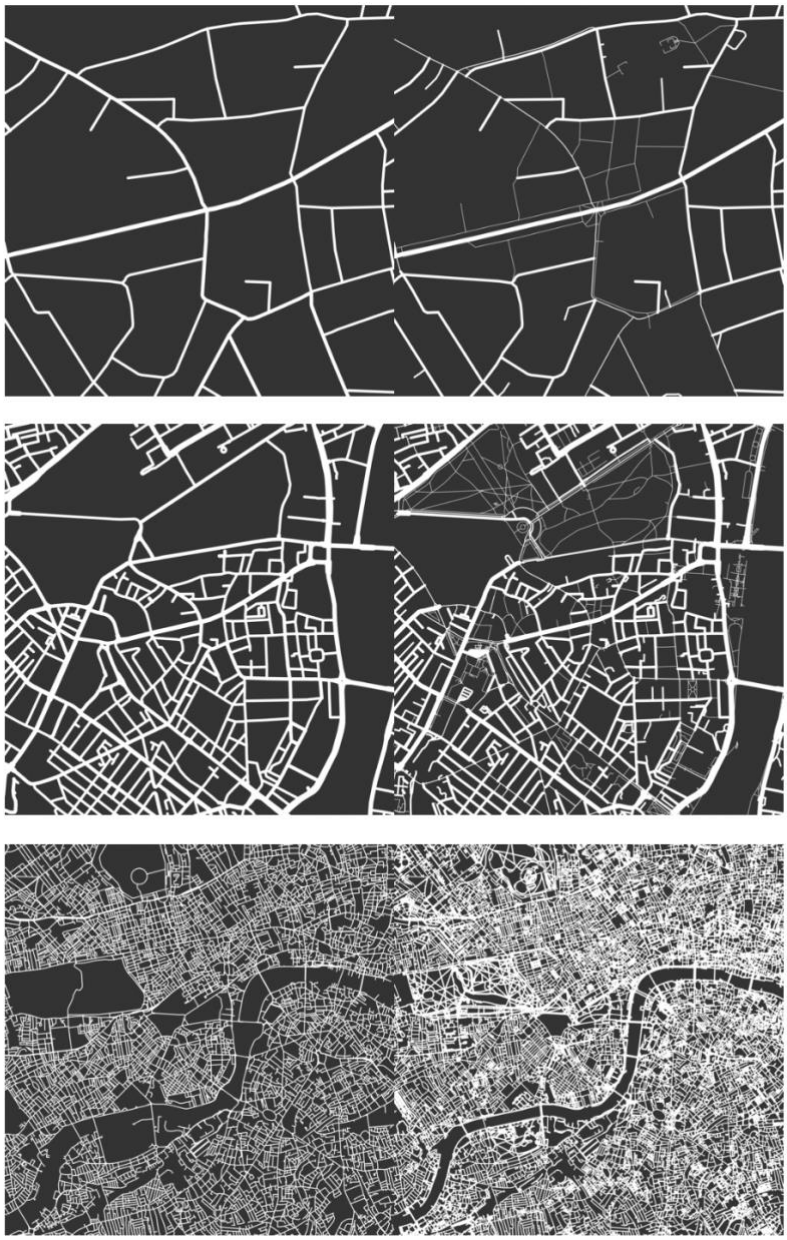


Figure 4. 2 Westminster street network in 'drive' (left) and 'all' (right) types of street in figure-ground paradigm (top to bottom: the scale of 0.25km, 1km, 4km)

As chapter 3 in section 3.4.2 states, generating the target map data requires several elements. After the modification of several hyperparameters, there is a need to find the most proper network map structure, as shown below:

- City address/coordinate
- Scale
- Street network type
- Street widths
- Figure size
- Street network type
- Pixel

4.2.2 City internal location selection

Ten different cities are selected to make the layout classification represent different urban structures. They are London, Beijing, Paris, Moscow, Los Angeles, Bangkok, New York, Sydney, Mumbai and Amsterdam (Figure 4. 3)

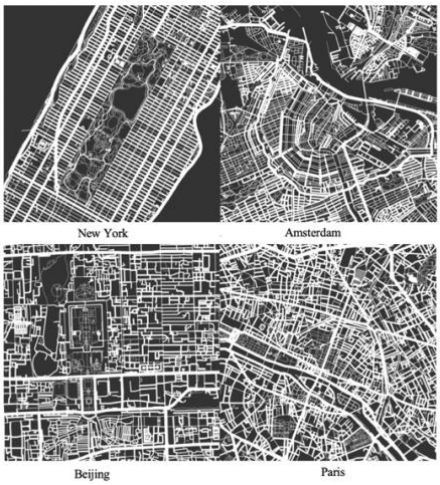


Figure 4. 3 Four cities' centre layout with different characteristics

OpenStreetMap file is saved as '.osm' file, which can be downloaded as 'OSM XMl' in xml format extended by the API. Alternatively, the Overpass JSON file can be read by the OSMnx and save the layout to disk as shapefile, such as 'multidigraph'.

100 layout figures of each city will be generated in different scales, which are 0.25km, 1km, and 4km individually. Statistical methods are needed for sampling locations in cities.

1) Create a dictionary that stores city name and order list
2) Access cities boundary and polygon information through the geopandas and OSMnx.
3) Random sampling in uniform distribution of both latitude and longitude under the range of city boundary
4) Check and visualize random coordinate distribution (Figure 4.4) and save random points in csv file
5) Generate figures through coordinate and parameter setting

The whole procedure under the same city order list at the first step is created. Layout figures are named with city name plus coordinate values and scale size number. For instance, the top left figure of the Figure 4. 3 will be named as 'New York, 40.78 -73.96, 2.5.png'. Precisely, coordinate values retain two digits. Therefore, it can be easy to identify each figure in the training procedure.

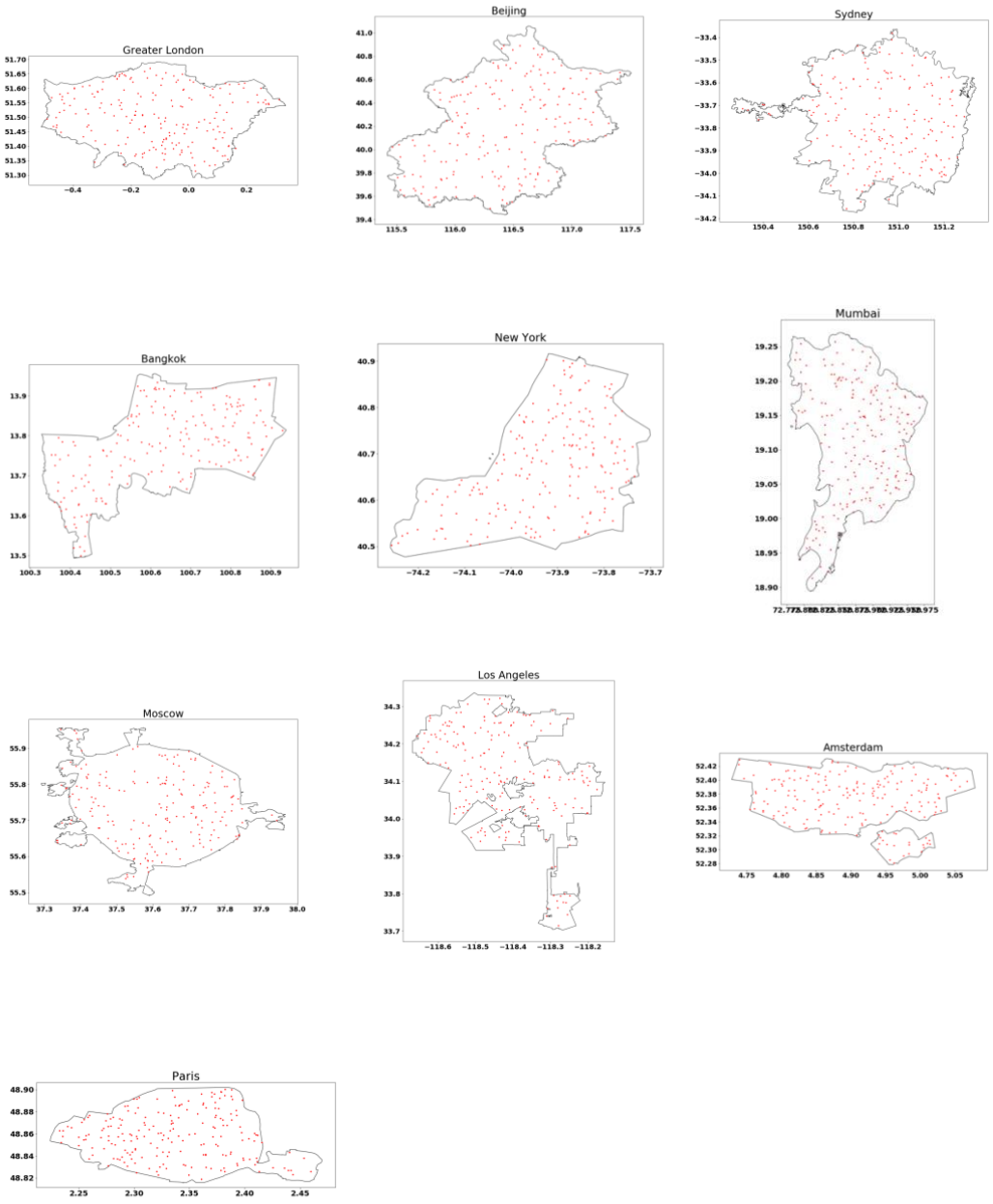


Figure 4. 4 City boundaries and random coordinates

4.1.4 Deep learning model building

The experiments of this project are designed based on the CNN model. The architecture of CNN is shown in Table 4. 1. Figure 4. 5 shows CNN output parameter status of each layer.

Table 4. 1 Details of the architecture of the convolutional neural network

Layer Type	Size	Output Shape
Input	(240,240,3)	None
Convolution + ReLu	32 3×3 filter	(238,238,32)
Max-Pooling +Dropout	2×2 filter, s=2	(119,119,32)
Convolution + ReLu	64 3×3 filter	(117,117,64)
Max-Pooling +Dropout	2×2 filter, s=2	(58,58,64)
Convolution + ReLu	128 3×3 filter	(56,56,128)
Max-Pooling +Flatten +Dropout	2×2 filter, s=2	100352
Hidden (ReLu or Tanh or Sigmoid)	512	512
SoftMax	10	10

Layer (type)	Output Shape	Param #
conv2d_10 (Conv2D)	(None, 238, 238, 32)	896
batch_normalization_13 (Batc	(None, 238, 238, 32)	128
max_pooling2d_10 (MaxPooling	(None, 119, 119, 32)	0
dropout_13 (Dropout)	(None, 119, 119, 32)	0
conv2d_11 (Conv2D)	(None, 117, 117, 64)	18496
batch_normalization_14 (Batc	(None, 117, 117, 64)	256
max_pooling2d_11 (MaxPooling	(None, 58, 58, 64)	0
dropout_14 (Dropout)	(None, 58, 58, 64)	0
conv2d_12 (Conv2D)	(None, 56, 56, 128)	73856
batch_normalization_15 (Batc	(None, 56, 56, 128)	512
max_pooling2d_12 (MaxPooling	(None, 28, 28, 128)	0
dropout_15 (Dropout)	(None, 28, 28, 128)	0
flatten_4 (Flatten)	(None, 100352)	0
dense_7 (Dense)	(None, 512)	51380736
batch_normalization_16 (Batc	(None, 512)	2048
dropout_16 (Dropout)	(None, 512)	0
dense_8 (Dense)	(None, 10)	5130

Total params: 51,482,058
Trainable params: 51,480,586
Non-trainable params: 1,472

Figure 4. 5 Output parameters status of each layer

4.3 Experiment

Experiments for this project are designed to observe which choices of activation functions and scales of layout figures can perform better on classification. The database will be split into dataset training, validation and test, which take 60%, 20% and 20% separately. In the beginning, ReLu activation function will be used to test the most effective scale layout figure. Then, the activation function is changed, in which alternative activation functions are Tanh and Sigmoid. Finally, the results will be analyzed.

Chapter 5

Experiment Results and Discussion

5.1 System Testing

Based on the complexity of layout figures, figures with bad performance are removed. According to map features,

- numbers of nodes and ways
- The distance between the two points is too close
- Whether the geographic information can be downloaded, at some coordinate point the json file does not exit

At the scale of 0.25km, drive type street network is quite simple. For example, for the coordinate at the edge of the city, areas out of city boundary perform as blank, which leaves tiny geographic information. Considering outskirts or geographical condition limits, some figures only present a simple line. This happens in each city, and thus these repetitive and confusing figures are deleted for reducing the error. Consequently, around 200 – 600 coordinates are obtained for each city. Firstly, points which come too close are removed to keep every point unique. Then, the coordinates without json file information are skipped. Next, images with insufficient geographic information are deleted. Finally, 100 layout figures of each city in each scale are picked up.

- **Street network type choice**

Drive network types meet the needs of geographic information classification. However, at the scale of 0.25km, great numbers of figures cannot supply a good quantity of geographic details.

For this reason, at the 0.25 km scale, an extra group of the dataset is added, which changes street network types to ‘all’.

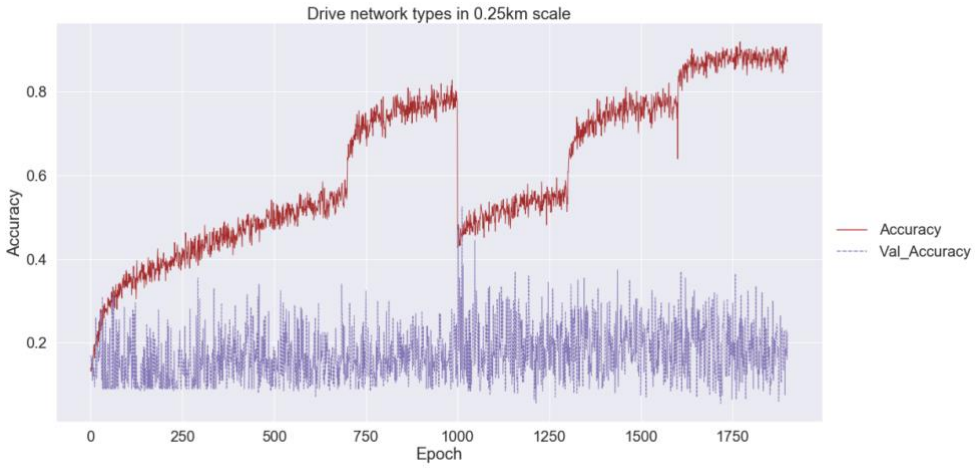
In conclusion, there are four group datasets. The groups with drive network types are in 0.25km, 1km, and 4km scales independently. In addition, there is one group with all network types in the 0.25km scale.

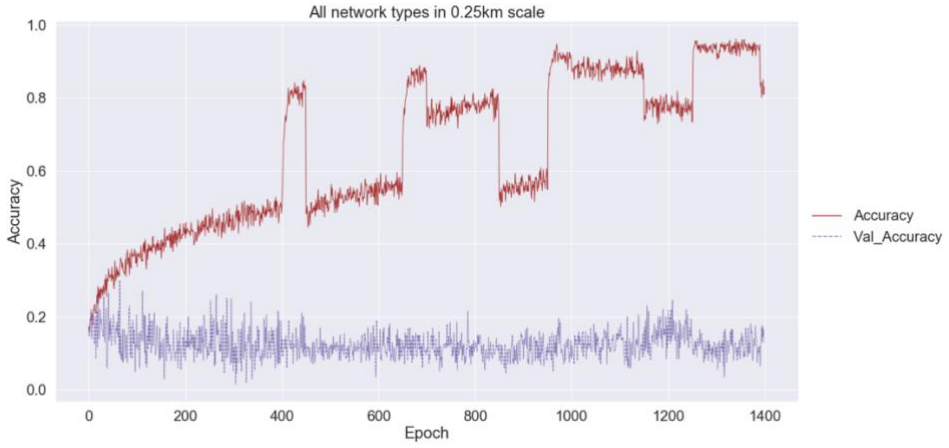
5.2 Experiment Results

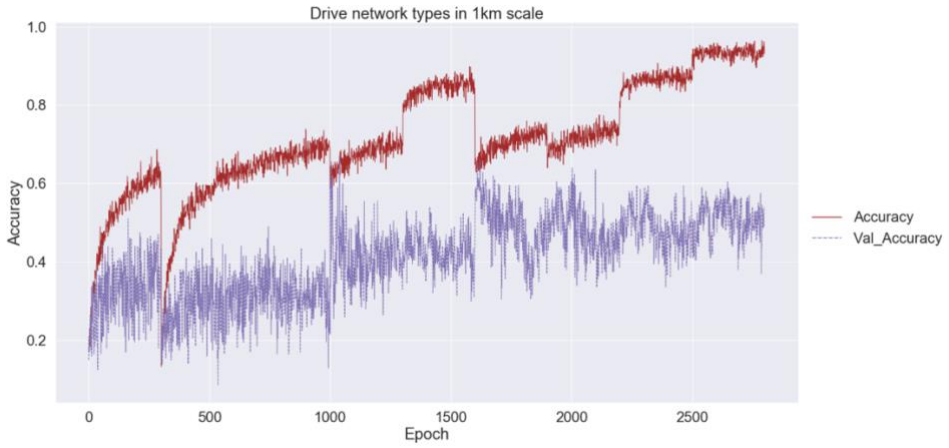
5.2.1 Scale performance

The initial number of training Epoch is set to 400. Then, the number of training Epoch number should be increased according to the learning speed of the training model, and 300 Epoch is added each time until model accuracy has first reached over 90%.

The following (Figure 5-1) presents accuracy changes with training Epoch. Table 5-1 shows final accuracy and total Epoch numbers of each scale layout figures.




All network types in 0.25km scale
Accuracy
Epoch
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
1200
1400
Accuracy
Val_Accuracy


Drive network types in 1km scale
Accuracy
Epoch
1.0
0.8
0.6
0.4
0.2
0
500
1000
1500
2000
2500
Accuracy
Val_Accuracy

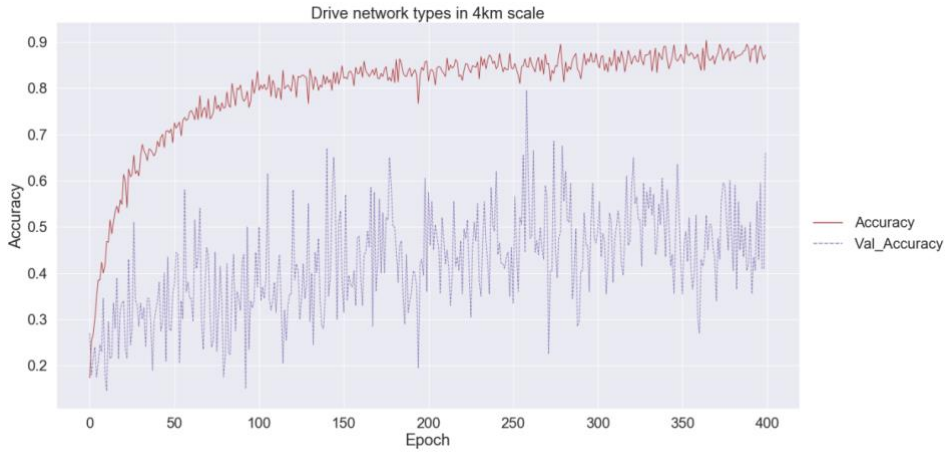


Figure 5. 1 Accuracy curves of different street network scales

Table 5. 1 Trained Epoch numbers and final accuracy of each scale

Scales	Street Network types	Epoch	Output Accuracy
0.25 km	Drive	1900	87.7%
0.25 km	All	1400	83%
1 km	Drive	2800	93.2%
4 km	Drive	400	87.2%

It can be seen that the scale of 4km is the most powerful layout map figure for classification. There are 3 reasons for selecting the scale of 4km to continue training for the next step. First, compared to other 3 group layout figures, model learning is fast under this scale. Second, it takes the shortest time to reach high accuracy. It spends around 750 – 900ms on each step, resulting in significant differences in the consumption of time. Its epoch investment is 2200 less than that at the scale of 1km, and the accuracy is only 5% less than the scale of 1km. Third, 4 km has the best stability during training. Especially, at the scale of 0.25 km with all street network types, there are many fluctuations with significantly changing magnitude. Even though this scale can achieve high accuracy, the outcome is unstable.

5.2.2 Activation function results

Figure 5. 2 shows the training results of selecting Tanh as the activation. In comparison with ReLu, it has a lower learning rate and the highest accuracy is 67.7% and 71.5% separately within the same training epoch.

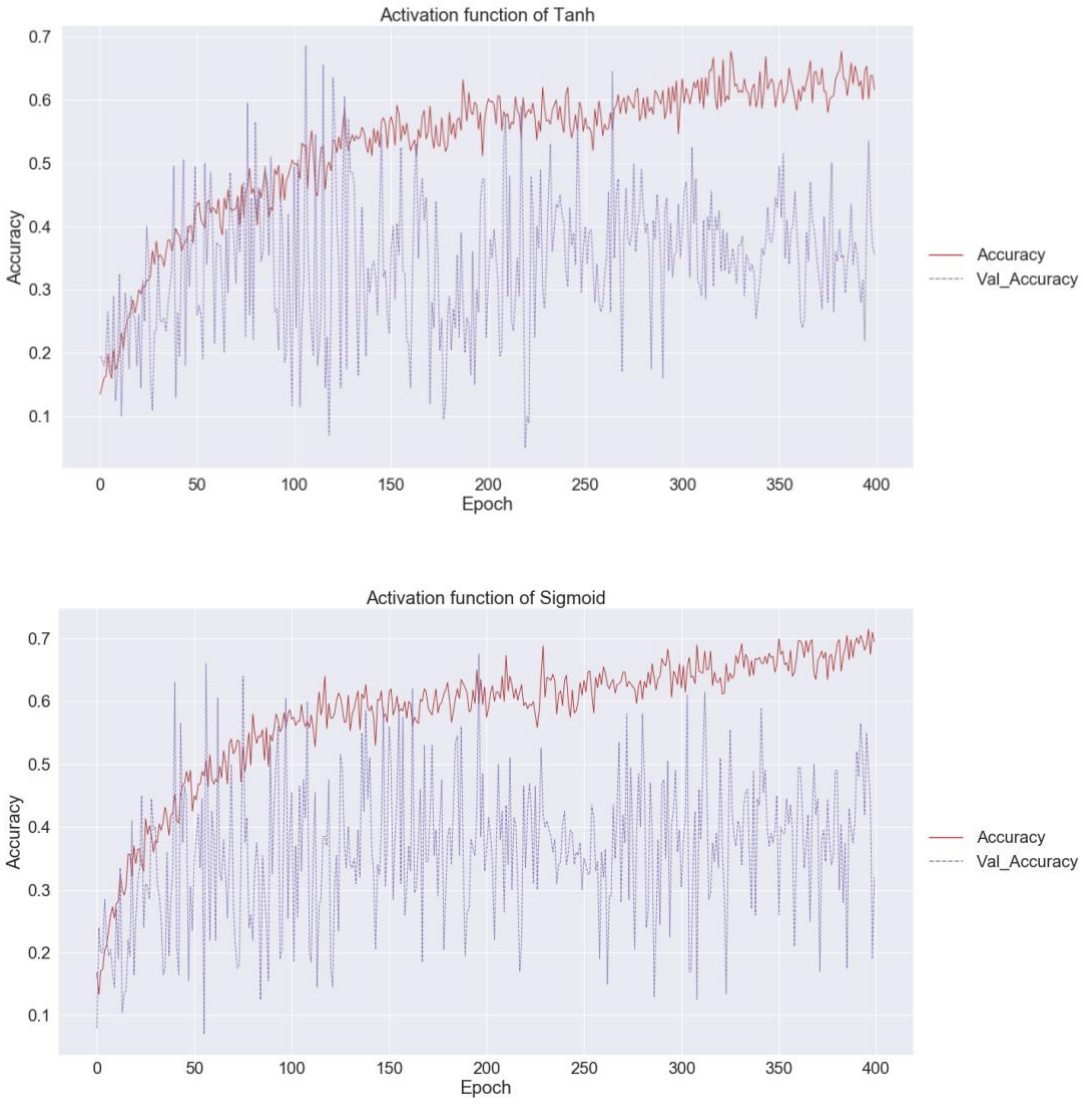


Figure 5. 2 Accuracy curves of activation function as Tanh and Sigmoid

The following confusion matrix (Figure 5. 3) shows the prediction and real label relationship. If the activation function is ReLu, confounded pairs are easy to happen between London and Sydney, New York and Sydney. On the other hand, Mumbai, Amsterdam and Beijing layout figures have a high accuracy, which can be distinguished from other cities easily. If the activation function is Tanh, top confusing combination are Amsterdam and Moscow, Beijing and Sydney. If the activation function is Sigmoid, top confusing couple are Bangkok and Los Angeles, Bangkok and Sydney. It reached the highest prediction accuracy in Moscow by using Than function, and in Bangkok by using Sigmoid function.

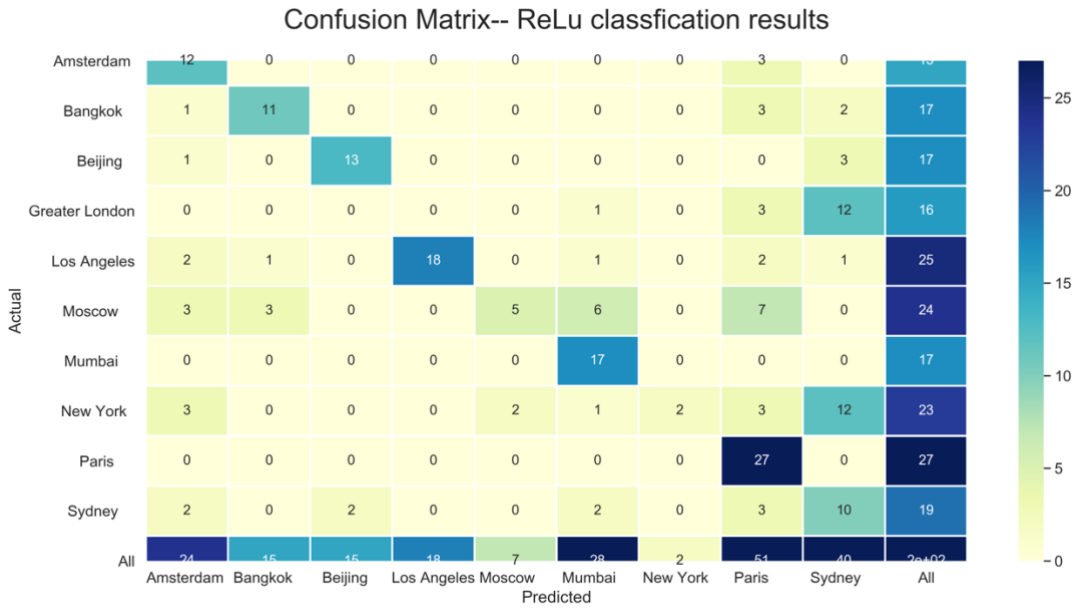


Figure 5. 3 Confusion matrix of 10 cities classification (top to bottom: Activation functions are ReLu, Tanh and Sigmoid)

5.3 Discussion and Analysis

5.3.1 Gradient vanishment

Deep networks are difficult to train due to the notorious vanishing gradient problem - because gradients propagate back to the earlier layers, repeated multiplications can make gradients quite tiny. Assuming that the same layer, called the identity mapping layer, is superimposed on the shallow network, even if the network depth increases, the training error should not be higher than the original shallow network. As it is more difficult to directly fit some overlays to a potential identity mapping function H(x) = x, some nonlinear layers are used to fit another residual map F(x):= H(x)-x, and the original mapping becomes H(x) = F(x) + x. Resnet learns the residual function F(x) = H(x) - x, and it is easier to optimize this residual mapping than to optimize the original mapping.

Consequently, as the network goes deeper, its performance becomes saturated and even begins to degrade rapidly. This is why it happens several times and is mutable under the scale of 0.25km with all street network types.

5.3.2 Activation function output

Each activation function has different gradient characteristics. The gradients of sigmoid and tanh are very gentle in the saturation area and close to 0, which may easily cause the vanishing gradient problem and reduce the convergence rate. On the contrary, Relu gradient is constant in most cases, which helps solve the convergence problem of the deep network. Another advantage of Relu is its biological rationality. It is unilateral and more in line with the features of biological neurons than sigmoid and tanh.

Sigmoid and tanh are proposed mainly because they are differentiable. There is also a problem of expression interval. Sigmoid and tanh interval is 0 to 1, or -1 to 1, which has advantages in expression, especially in the expression of the output layer.

5.3.3 Similarities between cities

There will be some similarities between different cities, perhaps results from factors like the local geographical environment. From the results, Sydney is most likely to be confused with other cities. No matter what activation equation is, it appears in top confusing combination. After examining the pictures, it can be found that grid structures accounted for a large proportion in the Sydney, and this structure also appears in other cities to varying levels. Therefore, the similarity between cities affects the accuracy of the prediction among different cities.

Chapter 6

Conclusions and Critical Evaluation

This project illustrates that the application of the end to end method can realize city classification beyond using a figure-ground diagram of the street network map. It is examined that in the process of generating layout figures from coordinate points in 3 different scales of 10 cities, there are 100 diagrams of each city in each scale and street network types. Besides, this project also investigated the way how hyper-parameters influence model performance and proposed map scales on the same model.

The results show that figures have relatively complex features in the per unit area, such as nodes density and ways density, ensuring that the CNN model has a more powerful way to learn in terms of training time and episodes. The recommended alternative activation function introduced criteria to process calculation value, which can increase or decrease the amount of computation, thereby influencing the speed of convergence of the model. After this, it will determine whether the current neuron should be activated.

6.1 Limitations

1. Image access

Some official storage files are incomplete and not clear.

Map files for China are not new enough and sufficient. When generating a street network of Beijing, it takes 600 coordinates to obtain the target 100 figures in the scale of 0.25km due to the absence of json file.

2. Gradient vanishing

It happens that as the number of neural network layers increases, regardless of the application of the activation function, gradient explosion and vanishing always exist. These problems will restrict the performance of outcomes.

3. Disadvantages of Keras

For Keras acts like a frontend framework, it lacks substantial functionality since it requires backend to finish the calculation. As a result, it reduces calculation speed. Moreover, programming based on Keras occupies more GPU memory.

4. Hardware challenges

The resource limitations of single-machine single-cell (such as GPU) often fail to meet the processing requirements for large-scale data and models. In the meantime, the cost of calculation is expensive.

6.2 Future Work

There are several aspects that can be improved for future work.

1. The way of street network figure design
    1) Design better precise ways for presenting features of the street network structure.
        - Change weights
        - Colored mark area or present different road types in color
    2) Add extra features
        - Population density
        - footprint
        - Measurement of scales, such as walking distance in 5, 10, and 15 minutes, which can enhance applicability

2. Location sampling way
    - Gibbs sampling
    - Markov chain Monte Carlo (MCMC)
    - Nested sampling
3. Deep learning algorithms

    To solve the gradient vanishing problem, ResNet architecture can be applied after checking whether it works.
4. Software or package of building deep learning architecture
    - Theano
    - Caffe
    - Torch

Bibliography

AlHalawani, S., Yang, Y., Wonka, P. and Mitra, N. (2014). What Makes London Work Like London? *Computer Graphics Forum*, 33(5), pp.157-165.

Arietta, S., Efros, A., Ramamoorthi, R. and Agrawala, M. (2014). City Forensics: Using Visual Elements to Predict Non-Visual City Attributes. *IEEE Transactions on Visualization and Computer Graphics*, 20(12), pp.2624-2633.

Chen, M., Zhou, P., Wu, D., Hu, L., Hassan, M. M., & Alamri, A. (2020). AI-Skin: Skin disease recognition based on self-learning and wide data collection through a closed-loop framework. *Information Fusion*, 54, 1–9.

Dewa, C. K., & Afiahayati. (2018). Suitable CNN Weight Initialization and Activation Function for Javanese Vowels Classification. *Procedia Computer Science*, 144, pp.124–132.

Elboushaki, A., Hannane, R., Afdel, K., and Koutti, L. (2020). MultiD-CNN: A multi-dimensional feature learning approach based on deep convolutional networks for gesture recognition in RGB-D image sequences. *Expert Systems with Applications, 139,* 112829.

Gonzalez, T. F. (2007). Approximation algorithms for multilevel graph partitioning. In *Handbook of Approximation Algorithms and Metaheuristics* (pp. 943-958). Chapman and Hall/CRC.

Guo, Q., Liang, Z. and Hu, J. (2017). Vehicle Classification with Convolutional Neural Network on Motion Blurred Images. *DEStech Transactions on Computer Science and Engineering*, (aiea).

Guo, Y., Yin, X., Zhao, X., Yang, D. and Bai, Y. (2019). Hyperspectral image classification with SVM and guided filter. *EURASIP Journal on Wireless Communications and Networking*, 2019(1).

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9), pp.1904–1916.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778.

Hormese, J., Saravanan, C., A convolutional neural network approach to road classification from satellite images. *Journal of Theoretical and Applied Information Technology, 96(23),* pp.7917-7927.

Islam, K.T,. Pervez, M., Wijewickrema, S., O'leary, S. (2018) Road trial classification using color images for autonomous vehicle navigation. *IEEE.*pp.1-5.

Jiang, Y. (2019). Research on road extraction of remote sensing image based on convolutional neural network. *EURASIP Journal on Image and Video Processing*, *2019(1),* 31.

L. J. Quackenbush. (2004) A review of techniques for extracting linear features from imagery Photogram. Eng. *Remote sens., 70(12),* pp. 1383-1392.

Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *Proceedings of the IEEE Computer. Society Conference on Computer Vision and Pattern Recognition*, 2, pp. 2169–2178.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), pp. 2278-2324.

Liang, X., & Wang, G. (2017, October). A convolutional neural network for transportation mode detection based on smartphone platform. In *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pp. 338-342.

Lu, J., Ye, Y., Xu, X. and Li, Q. (2019). Application research of convolution neural network in image classification of icing monitoring in power grid. *EURASIP Journal on Image and Video Processing*, 2019(1), 49.

Miao, Z., Gaynor, K. M., Wang, J., Liu, Z., Muellerklein, O., Norouzzadeh, M. S., … Getz, W. M. (2019). Insights and approaches using deep learning to classify wildlife. *Scientific Reports*, 9(1), 1–9.

Nakajima, Y. and Saito, H. (2016). Robust camera pose estimation by viewpoint classification using deep learning. *Computational Visual Media*, 3(2), pp.189-198.

Peng, C., Yang, Y., Bao, F., Fink, D., Yan, D., Wonka, P. and Mitra, N. (2016). Computational network design from functional specifications. *ACM Transactions on Graphics*, 35(4), pp.1-12.

Shi, W., Miao, Z., & Debayle, J. (2013). An integrated method for urban main-road centerline extraction from optical remotely sensed imagery. *IEEE Transactions on Geoscience and Remote Sensing*, *52*(6), pp. 3359-3372.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Su, J., Vargas, D. and Sakurai, K. (2019). Attacking convolutional neural network using differential evolution. *IPSJ Transactions on Computer Vision and Applications*, 11(1), pp. 1.

Sui, K. and Kim, H. (2019). Research on application of multimedia image processing technology based on wavelet transform. *EURASIP Journal on Image and Video Processing*, 2019(1), pp. 24.

Xin, M. and Wang, Y. (2019). Research on image classification model based on deep convolution neural network. *EURASIP Journal on Image and Video Processing*, 2019(1), pp. 40.

Yang, S., Gong, L. and Qiao, D. (2019). Image offset density distribution model and recognition of hand knuckle. *EURASIP Journal on Image and Video Processing*, 2019(1), pp.23.

Appendix A

Code

Figure generate code

[1]:
```
import osmnx as ox
import geopandas as gpd
import pandas as pd
import matplotlib.cm as cm
import matplotlib.colors as colors
import networkx as nx
from IPython.display import Image
# from keys import google_elevation_api_key #replace this with your own API key!

import matplotlib.pyplot as plt
from descartes import PolygonPatch
from shapely.geometry import Point, LineString, Polygon
import numpy as np
import igraph as ig
import operator

from geopandas.tools import overlay

import random
```

[2]:
```
%matplotlib inline
ox.config(log_console=True, use_cache=True)
ox.__version__
```

[2]:
```
'0.10'
```

1 1. Input 10 city names, saved as dictionary

[5]:
```
places ={'London'  : 'Greater London, United Kingdom',
         'Beijing' : 'Beijing,China',
         'Paris'   : 'Paris,France',
         'Moscow'  : 'Moscow,Rusia',
         'Los Angeles'   : {'city':'Los Angeles', 'state':'CA', 'country':
 ↪'USA'},
         'Bangkok':'Bangkok, Thailand',
         'New York': 'New York,USA',
         'Sydney'  : 'Sydney,Australia',
         'Mumbai'  : 'Mumbai, India',
         'Amsterdam': 'Amsterdam,Netherlands'}
```

2 2. Access city boundary value and polygon information

[7]:
```
gdf = ox.gdf_from_places(places.values())
gdf
```

3 3. Generate random points

[15]:
```
def random_point(gdf,index,num):
    north = gdf[index:index+1].bbox_north.values[0]
    south = gdf[index:index+1].bbox_south.values[0]
    west = gdf[index:index+1].bbox_west.values[0]
    east = gdf[index:index+1].bbox_east.values[0]

    lagitude = []
    long = []

    for j in range(num):
        lagitude.append(random.uniform(south,north))
        long.append(random.uniform(west, east))

    random_point = pd.DataFrame({'Latitude': lagitude,'Longitude': long})
    gdf_random = gpd.GeoDataFrame(
        random_point, geometry=gpd.points_from_xy(random_point.Longitude,␣
↪random_point.Latitude))

    return gdf_random
```

4 4. Select coordinates within the boundary of each city

[ ]:
```
def overlay_point(gdf,gdf_random,index):
    polygon = gdf.geometry[index]
#     gdf_random = random_point(index)
    gdf_random.within(polygon)
    subset = gdf_random[gdf_random.within(polygon)]
    print(subset.shape)
    new_point = subset.reset_index(drop=True)
    new_point = new_point.iloc[:600]
    print('city new point:',new_point.shape)

    plt.rcParams["font.weight"] = "bold"
    base = gdf[index:index+1].plot(color='white',␣
↪edgecolor='black',figsize=(14,14))
    new_point.plot(ax=base, marker='o', color='red', markersize=8)
    name = gdf.place_name[index].split(',')[0]

    plt.title(name, fontdict = {'fontsize' : 30})

    plt.xticks(size = 20)
    plt.yticks(size = 20)
    plt.rcParams["axes.labelweight"] = "bold"
```

```
    plt.savefig('random_fig/'+'{:>10}'.format(name)+'.png')
    plt.show()
    return new_point
```

4.1 Save coordinates as dataframe with latiude, longitude value

[16]:
```
df = pd.DataFrame(columns=['Latitude', 'Longitude'])
all_point = gpd.GeoDataFrame(
    df, geometry=gpd.points_from_xy(df.Longitude, df.Latitude))
all_point.shape
```

[16]:
```
(0, 3)
```

4.2 Visualize coordinate under the range of each city

[17]:
```
for i in range (10):
    print(gdf.place_name[i])
    gdf_random = random_point(gdf,i,1200)

    new_point = overlay_point(gdf,gdf_random,i)
    print('output new point',new_point.shape)
    all_point = pd.concat([all_point, new_point])
```

5 5. Generate street network figure based on coordinate

5.1 Modify network_type and dist

[10]:
```
for i in range(5,10):
    city = gdf.place_name[i].split(',')[0]
    print(city)
    for j in range(200):
        index = 200*i+j
#         print('index',index)
        point = (samp.Latitude.iloc[index],samp.Longitude.iloc[index])

        location = filename='{} {}'.format(round(samp.Latitude.iloc[index],2),␣
↪round(samp.Longitude.iloc[index],2))+ ','+'1'
#         print('location',location)
        try:
            fig, ax = ox.plot_figure_ground(point=point, dist=1000,␣
↪network_type='drive_service',
                            street_widths=None, default_width=2, fig_length=10,
                            edge_color='w', bgcolor='#333333',␣
↪smooth_joints=True,
                            filename='drive_1/'+city + ','+ location,␣
↪file_format='png',
                            show=False, save=True, close=True, dpi=300)
            print('index',index)

        except:
            pass
```

Convolution neural network code

[12]:
```
import numpy as np
import pandas as pd
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator, load_img
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import random
import os
import json

from keras.models import load_model
```

[13]:
```
import tensorflow as tf
print(tf.__version__)
```

```
1.14.0
```

[15]:
```
from PIL import Image, ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True
```

[16]:
```
os.environ['PYTHONHASHSEEED'] = '0'
tf.set_random_seed(37)
np.random.seed(37)
random.seed(37)
```

[18]:
```
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
sess = tf.Session(config=config)
```

1 1. Open image file

[19]:
```
filenames = os.listdir("4_all")
```

[20]:
```
len(filenames)
```

[20]:
```
1001
```

macOS system creat '.DS_Store' automatically

[21]:
```
filenames.remove('.DS_Store')
```

1.1 set up image size

[22]:
```
FAST_RUN = False
IMAGE_WIDTH=240
IMAGE_HEIGHT=240
IMAGE_SIZE=(IMAGE_WIDTH, IMAGE_HEIGHT)
IMAGE_CHANNELS=3
```

1.2 2. Read image data with pandas

[23]:
```
categories = []
for filename in filenames:
    category = filename.split(',')[0]
    categories.append('{:10}'.format(category))
```

[24]:
```
df = pd.DataFrame({
    'filename': filenames,
    'category': categories})
print('Dataset shape: ',df.shape)
```

```
Dataset shape:  (1000, 2)
```

[25]:
```
df['category'].value_counts()
```

[25]:
```
Paris             100
Mumbai            100
Los Angeles       100
Moscow            100
Bangkok           100
New York          100
Sydney            100
Amsterdam         100
Greater London    100
Beijing           100
Name: category, dtype: int64
```

[17]:
```
df['category'].value_counts().plot.bar()
```

[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a382e52e8>
```

2 3. Buidling CNN architecture

[26]:
```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense,␣
 ↪Activation, BatchNormalization

model = Sequential()
# Convolution + ReLu layer
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(IMAGE_WIDTH,␣
 ↪IMAGE_HEIGHT, IMAGE_CHANNELS)))
model.add(BatchNormalization())
# Max-Pooling +Dropout layer
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

```
# Hidden layer
model.add(Flatten())
model.add(Dense(512, activation='tanh'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))


model.compile(loss='categorical_crossentropy', optimizer='rmsprop',␣
 ↪metrics=['accuracy'])

model.summary()
```

2.1 Split dataset into train, test and validate

dataset of train: test: validate = 6 : 2 : 2

[29]:
```
train_df, test_df = train_test_split(df, test_size=0.2, random_state=42)
train_df, validate_df = train_test_split(train_df, test_size=0.25,␣
 ↪random_state=42)
```

[30]:
```
train_df = train_df.reset_index(drop=True)
test_df = test_df.reset_index(drop=True)
validate_df = validate_df.reset_index(drop=True)
```

[31]:
```
print('Training dataset shape: ',train_df.shape)
print('Validation dataset shape: ',validate_df.shape)
print('Test dataset shape: ',test_df.shape)
```

```
Training dataset shape:  (600, 2)
Validation dataset shape:  (200, 2)
Test dataset shape:  (200, 2)
```

[32]:
```
total_train = train_df.shape[0]
total_validate = validate_df.shape[0]
batch_size= 5
```

[33]:
```
train_df.head()
```

[33]:

	filename	category
0	Paris,48.83 2.3,4.png	Paris
1	Moscow,55.75 37.5,4.png	Moscow
2	Amsterdam,52.39 4.9,4.png	Amsterdam
3	Greater London,51.41 0.14,4.png	Greater London
4	Paris,48.85 2.41,4.png	Paris

[34]:
```
train_datagen = ImageDataGenerator(
    rotation_range=15,
    rescale=1./255,
    shear_range=0.1,
    zoom_range=0.2,
    horizontal_flip=True,
    width_shift_range=0.1,
    height_shift_range=0.1
)
```

```
train_generator = train_datagen.flow_from_dataframe(
    train_df,
    "4_all/",
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='categorical',
    batch_size=batch_size
)
```

```
Found 600 validated image filenames belonging to 10 classes.
```

[35]:
```
validation_datagen = ImageDataGenerator(rescale=1./255)
validation_generator = validation_datagen.flow_from_dataframe(
    validate_df,
    "4_all/",
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='categorical',
    batch_size=batch_size
)
```

```
Found 200 validated image filenames belonging to 10 classes.
```

[36]:
```
example_df = train_df.sample(n=1).reset_index(drop=True)
example_generator = train_datagen.flow_from_dataframe(
    example_df,
    "4_all/",
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='categorical'
)
```

```
Found 1 validated image filenames belonging to 1 classes.
```

[37]:
```
total_validate//batch_size,total_train//batch_size
```

[37]:
```
(40, 120)
```

3 4. Training model

[38]:
```
epochs=3 if FAST_RUN else 300
history = model.fit_generator(
    train_generator,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=total_validate//batch_size,
    steps_per_epoch=total_train//batch_size
#     callbacks=callbacks
)
```

```
W0907 23:17:33.560070 4732810688 deprecation.py:323] From
//anaconda3/lib/python3.7/site-packages/tensorflow/python/ops/math_grad.py:1250:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where

Epoch 1/300
120/120 [==============================] - 233s 2s/step - loss: 2.7719 - acc:
0.1400 - val_loss: 2.5962 - val_acc: 0.0950
```

3.1 Visualize accuracy and loss curve

[39]:
```
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 12))
ax1.plot(history.history['loss'], color='b', label="Training loss")
ax1.plot(history.history['val_loss'], color='r', label="validation loss")
ax1.set_xticks(np.arange(1, epochs, 1))
ax1.set_yticks(np.arange(0, 1, 0.1))

ax2.plot(history.history['acc'], color='b', label="Training accuracy")
ax2.plot(history.history['val_acc'], color='r',label="Validation accuracy")
ax2.set_xticks(np.arange(1, epochs, 1))

legend = plt.legend(loc='best', shadow=True)
plt.tight_layout()
plt.savefig('TanhSoftmax_loss_acc_curve.png')
plt.show()
```

Save model, ensure model can be load and continue training

[45]:
```
model.save('TanhSoftmax_training_model.h5')
```

3.2 Save accuracy and loss into csv file

[40]:
```
Val_Loss = pd.DataFrame({"Val_Loss":history.history['val_loss']})
Loss = pd.DataFrame({"Loss":history.history['loss']})
Val_Loss.shape,Loss.shape
```

[40]:
```
((300, 1), (300, 1))
```

[41]:
```
Val_Accuracy = pd.DataFrame({"Val_Accuracy":history.history['val_acc']})
Accuracy = pd.DataFrame({"Accuracy":history.history['acc']})
Accuracy.shape,Val_Accuracy.shape
```

[41]:
```
((300, 1), (300, 1))
```

[42]:
```
frames = [Val_Accuracy, Accuracy,Val_Loss,Loss]
result = pd.concat(frames, axis=1, join='inner')
result.head()
```

[42]:

	Val_Accuracy	Accuracy	Val_Loss	Loss
0	0.095	0.140000	2.596214	2.771913
1	0.100	0.163333	3.154372	2.553566
2	0.255	0.216667	2.623200	2.470449
3	0.135	0.180000	3.358853	2.607753
4	0.205	0.163333	2.604756	2.537889

[43]:
```
result.to_csv('TanhSoftmax_loss_acc_curve.csv', index=False)
```

4 5. Make prediction

[44]:
```
test_gen = ImageDataGenerator(rescale=1./255)
test_generator = test_gen.flow_from_dataframe(
    test_df,
    "4_all/",
    x_col='filename',
    y_col=None,
    class_mode=None,
    target_size=IMAGE_SIZE,
    batch_size=batch_size,
    shuffle=False
)
```

```
Found 200 validated image filenames.
```

[46]:
```
nb_samples = test_df.shape[0]
predict = model.predict_generator(test_generator, steps=np.ceil(nb_samples/
 ↪batch_size))
```

[48]:
```
test_df['prediction'] = np.argmax(predict, axis=-1)
```

4.1 Change prediction value into labels, city names

[49]:
```
label_map = dict((v,k) for k,v in train_generator.class_indices.items())
test_df['prediction'] = test_df['prediction'].replace(label_map)
test_df['prediction'].head()
```