Final Report for Machine Learning Course

Map identification and classification based on projection and region using Machine Learning methods

Jialin Li

Ziyu Guo

Rui Li

Luyu Liu

April. 28. 2019

Map identification and classification based on projection and region using Machine Learning methods

1. Introduction

Artificial Intelligence (AI) has been applied to plenty of fields to solve the problems in different areas, aiming at making machine act as human beings. AI has been commonly used in almost every field in our daily life, for example, self-driving car, speech recognition, game playing and robotics. Several Artificial Intelligence-based techniques have been applied to improve Intelligence Transportation Systems for three main areas: vehicle control, traffic control and prediction as well as road safety and accident prediction (Machin et al., 2018). Silver et al. (2017) trained a neural network by reinforcement learning without human data, guidance or domain knowledge beyond game rules to predict AlphaGo's own move selections which became the first program to defeat a world champion in the game of Go. Sengorur, Koklu and Ates (2015) used self-organizing map- artificial neural networks (SOM-ANNs) to evaluate the high-low flow period correlations in terms of water-quality parameters. And the SOM supported by ANN, is applied to provide a nonlinear relationship between input and output variables in order to determine the most significant parameters in each group. Seo et al. in 2015 developed and applied two hybrid models for daily water level forecasting and investigate their accuracy. The two hybrid models are wavelet-based artificial neural network (WANN) and wavelet-based adaptive neuro-fuzzy inference system (WANFIS).

However, there is little application of AI in cartography or the process of map making. Some questions in cartography can be possibly solved by methods in AI. If an image is given to AI, is it able to tell us some information about this image? We think AI can answer these questions as follows. First, Can AI tell whether this image is a map? Second, which region is the map about? Then, what is the projection used in this map? Next, can we extract the point, line and polygon features and their attribute value for further analysis? Last but not least, is this map a good map from the perspectives of color use, label placement and so on?

In this project, we focus on the first questions as my research question: 1. Can AI figure out whether the given image is a map? 2. If this image is a map, is AI able to find which region this map is about? 3. What is the projection used in this map? The three research questions are object detection and classification problems, which can be solved by some machine learning methods. The process of find solutions to the questions is optimization of parameters in the methods in essence. In this project, we used each of the three machine learning methods, MultiLayer Perceptions (MLPs), Support vector machines (SVMs) and Convolutional Neural Networks (CNNs) to solve map identification, region classification and projection classification problems.

In 2016, He et al. developed a system for automatically mapping and inspecting power line components using CNN-based object detection and classification models. This system is capable of mapping basic power components including poles, toppads, crossarms in insulators. Hughes and Salathe in 2015 and Mohanty et al. in 2016 utilized

54306 images to train DCNN (AlexNet and GoogLeNet architectures) to identify 26 diseases from 14 distinct crop species. According to the assessment metric F1 score, GoogLeNet outperforms AlexNet. VGG CNN was used by Ferentinos (2018) to work on the same Plant Village dataset and the success rate of identifying plant stress was very high. The results indicated that the model performance was better when original images were used and the use of preprocessed images would lead to significantly reduced computational time. Oh et al. (2013) applied Differential Evolution (DE) to optimize the parameters of a Radial Basis Function Neural Network (RBFNN) and also for feature selection of PCA and Linear Discriminant analysis (LDA) features. Ashok et al. (2016) compared feature selection methods for diagnosis of cervical cancer using a support vector machine (SVM) classifier. Feature selection was achieved using mutual information. Sequential forward search, sequential floating forward search and random subset feature selection methods. An accuracy of 98.5% was obtained using the sequential floating forward selection method.

In this study, we first used MLPs, SVMs and CNNs to identify whether an image is a map. Then we also used these methods to classify the regions and the projections of the map. Additionally, transfer learning using three popular CNN architectures is used for map classification based on region. The three architectures are GoogLeNet, VGG16 and VGG19.

2. Methodology

In this section, we will briefly introduce these three machine learning methods and three powerful CNN architectures which will be used in transfer learning for region classification.

2.1 Multiplayer perceptrons (MLPs)

MLPs here mean the traditional multi-layer neural networks, which is the basis of deep neural networks including CNNs. MLPs have been successfully used for image recognition in many fields (Sibanda and Pretorius, 2011).

MLPs are a class of feedforward artificial neural networks, the general architecture of MLPs can be shown in Fig.1 below. Backpropagation, which is an important supervised learning technique in artificial neural networks, is used in MLPs for the process of training parameters.

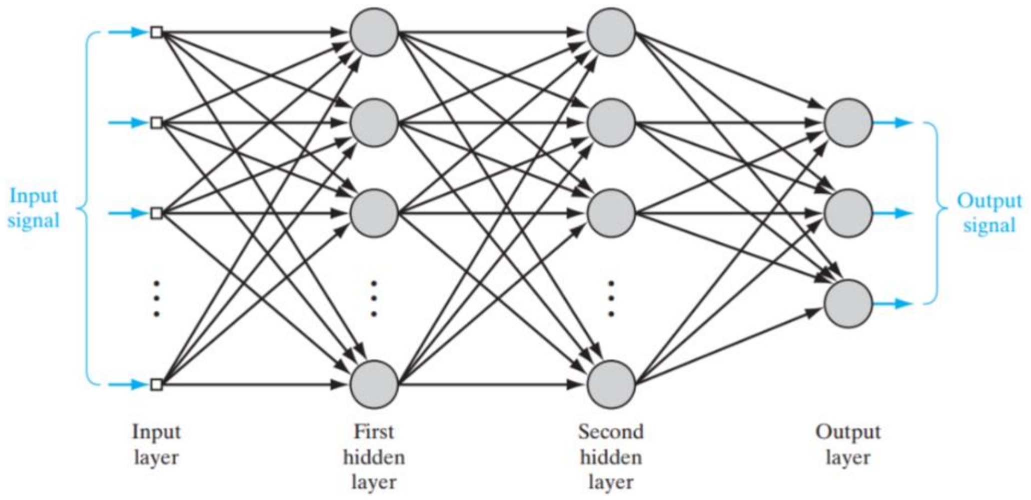


Fig.1 General architecture of MLPs

2.2 Support vector machines (SVMs)

SVMs are one of the most effective machine learning algorithms in object detection and classification problems in various fields (Oleszkiewicz et al., 2016). They are supervised learning models with associated learning algorithm. For a linear separable classification task, there are infinitely many separating hyperplanes as shown in Fig.2 below. SVMs aim at finding the optimal separating hyperplanes. For nonlinearly separable case, SVMs map the input space onto a high-dimensional feature space. The structure of SVMs is shown below in Fig.3. The reason is that according to Cover's theorem, a complex classification problem, cast in a high-dimensional space nonlinearly, is more likely to be linearly separable than in the low-dimensional input space. For example, input patterns cannot be separated in 2-dimensional space, but they can be separable in 3-dimensional space as shown in Fig.4.

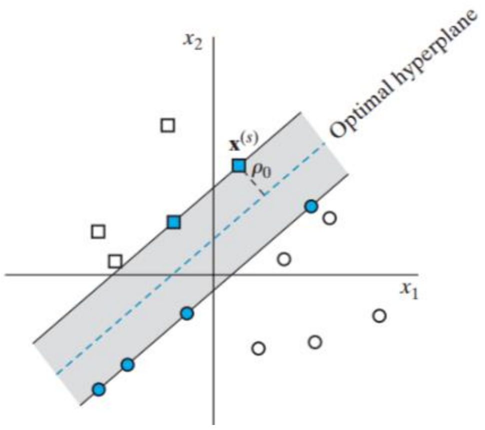
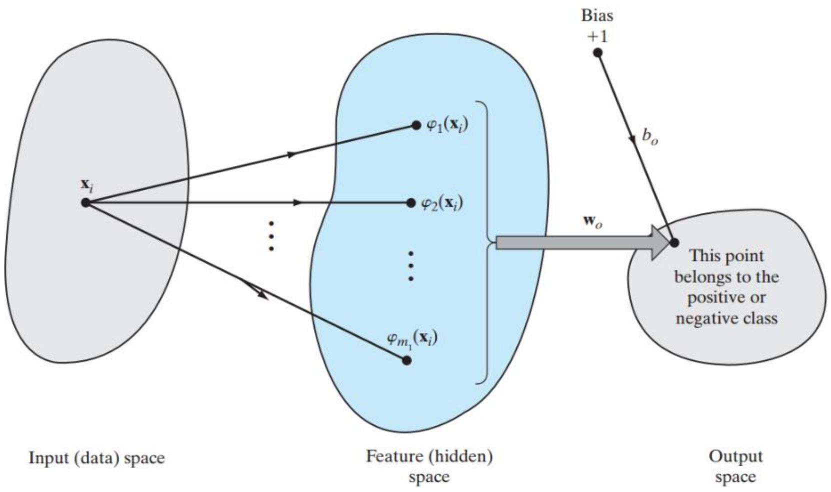


Fig.2 Linear separable classification task



Bias
+1
b_o
$\mathbf{x}_i$
$\varphi_1(\mathbf{x}_i)$
$\varphi_2(\mathbf{x}_i)$
$\varphi_{m_1}(\mathbf{x}_i)$
$\mathbf{w}_o$
This point belongs to the positive or negative class
Input (data) space
Feature (hidden) space
Output space

Fig.3 Structures of SVMs for non-linear classification

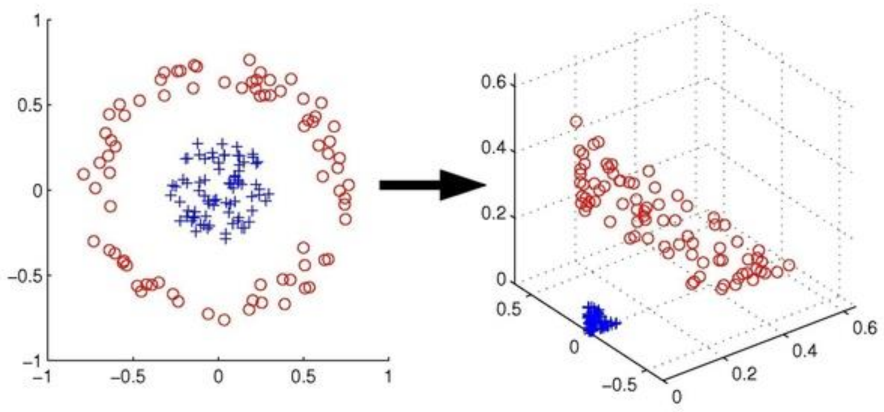


Fig.4 Example of separation of patterns in high-dimensional space

2.3 Convolutional neural networks

CNNs have been proved highly effective in large-scale image recognition and object detection (Zhu et al.,2017). CNNs are a class of deep, feed-forward artificial neural networks. Traditional MLPs cannot go deeper because of vanishing or exploding gradient problem. CNNs can solve the problems of traditional MLPs by reducing the number of free parameters and allowing the network to be deeper with fewer parameters. CNNs are the most commonly applied method for analyzing visual imagery. An example of the network architecture of CNNs is shown in Fig.5 below.

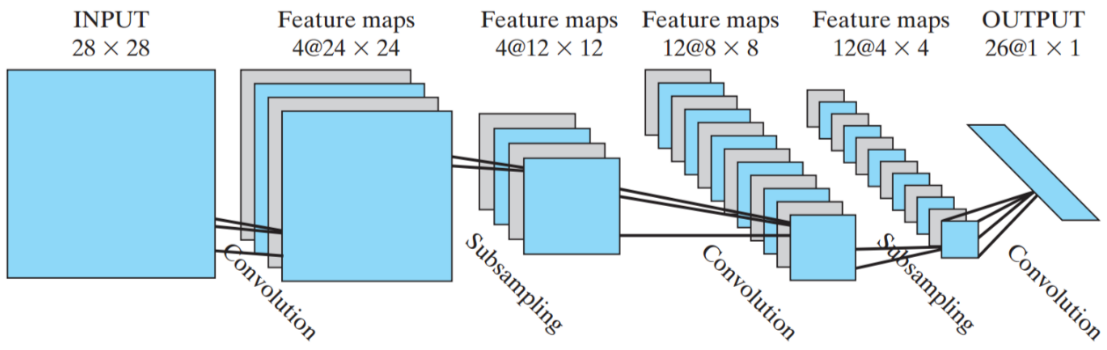


Fig.5 An example of the architecture of CNNs

2.4 Transfer Learning

In addition to training a CNN architecture by ourselves, pre-trained deep learning architectures have also been widely used and able to achieve significant success in many fields in recent years (Hinton, Osindero and Teh 2006; Kavukcuoglu et al., 2010; Farabet et al., 2013; Antonellis et al., 2015). To solve the problems of the requirements for large amount of training data and high-performance computing resources, pre-trained networks are adapted to an application by fine-tuning the parameters of the networks with the domain specific data (Mehdipour, Yanikoglu and Aptoula, 2017;Torrey and Shavlik, 2010). This is a form of transfer learning, aiming at utilizing knowledge learned from a problem to another relevant problem (Pan and Yang, 2010; Zhou et al., 2014). There are several commonly-used and powerful pre-trained architectures of CNNs in many other fields, including AlexNet, GoogLeNet VGG16 and VGG19. The later three architectures will be used in our research because they are relatively new developed and can achieve more accurate prediction.

2.4.1 GoogLeNet

GoogLeNet, the winner of ILSVRC 2014, is an inception architecture (Szegedy et al.,2015a). It combines the multi-scale idea and dimension reduction layers based on the Hebbian principle and Embedding Learning (Ghazi et al., 2017). There are 4 versions of GoogLeNet. In this project, we used Inception V3 model to do fine tuning experiment.

Inception V3 model has 42 layers and about 7 million parameters[1]. The most important feature of Inception V3 is that it uses factorizing convolutions. The factorization is aimed at reducing the number of parameters without reducing the efficiency of the neural network.

In Inception V3, the original 1 layer of 5x5 convolution layer is replaced by two 3x3 convolution layer (Fig.6). In this way, one 5x5 convolution layer with 25 parameters has been reduced to two 3x3 convolution layers with a total of 9+9=18

[1] Some layers contain different kind of parallel sublayers, and the total number of sublayers is 310.

parameters. With the parameter reduction, the network will be less likely to be overfitting. It also allows the network to go deeper.

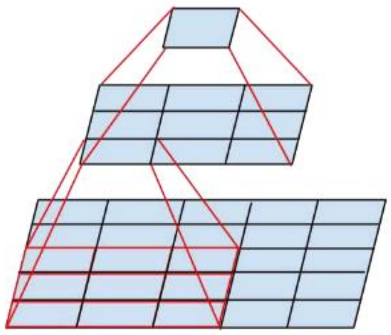


Fig.6 Mini-network replacing the 5 × 5 convolutions (source: Szegedy et al. 2015b)

Fig.7 is an architecture to describe the Inception V3 network. The model is made up of symmetric and asymmetric building blocks. There are six types of layers in this model: convolutional layer, average-pooling layer, max-pooling layer, concat layer, dropout layer, and fully-connected layer. The loss is computed using softmax[2]. It is much more efficient than VGGNet in terms of computation cost.

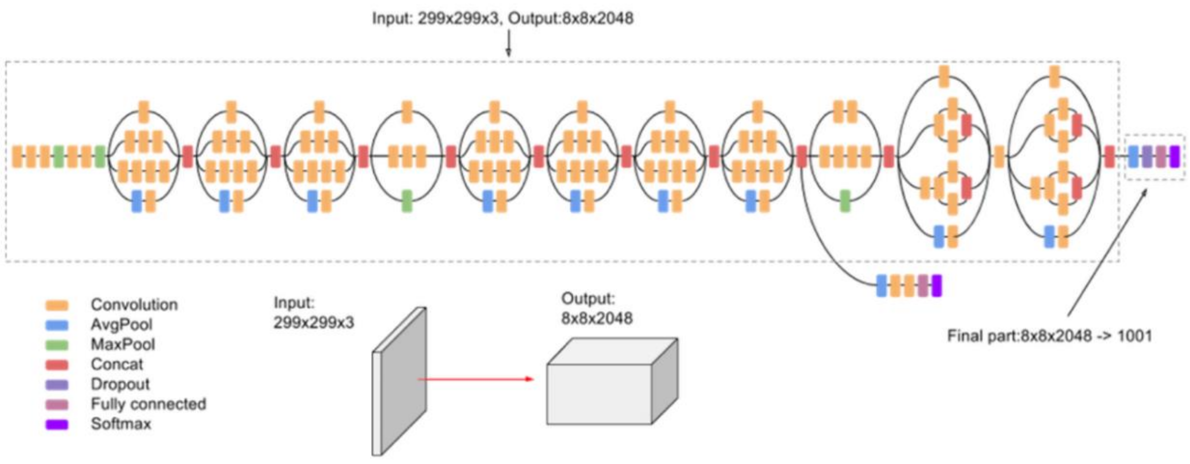


Fig.7 Inception V3 network architecture (Figure source: https://cloud.google.com/tpu/docs/inception-v3-advanced)

Inception V3 model is very complicated and contains a total of 42 layers which consisting 310 sublayers[3] and many sublayers do not have any trainable parameters. The relation of layer and sublayer in Inception V3 model is shown in Fig.8.

[2] https://cloud.google.com/tpu/docs/inception-v3-advanced

[3] Sublayers form into layers.

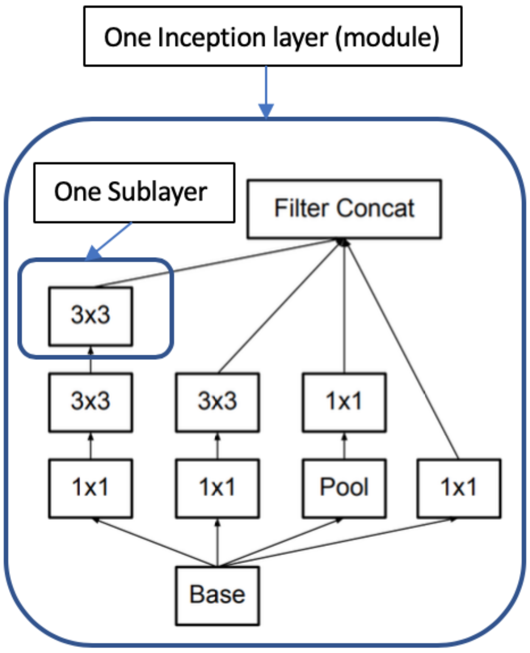


Fig.8 Relation between layers and sublayers

2.4.2 VGG16

As the first runner-up in ILSVRC 2014, VGG16Net (Simonyan and Zisserman, 2014) uses a homogeneous architecture to investigate the effects of increased convolutional network depth on performance. “16 means that there are 16 weight layers.” The input to the ConvNets is a fixed-size 224 * 224 RGB image The VGG16Net architecture involves 144 million parameters from 16 convolutional layers with very small receptive fields (3 × 3), five max-pooling layers of size 2 × 2 with stride 2, three fully-connected layers, and a linear layer with Softmax activation in the output. This model also uses dropout regularization in the fully-connected layer and applies ReLU activation to all the convolutional layers.

The configuration of VGG16Net is shown in Fig.9 below:

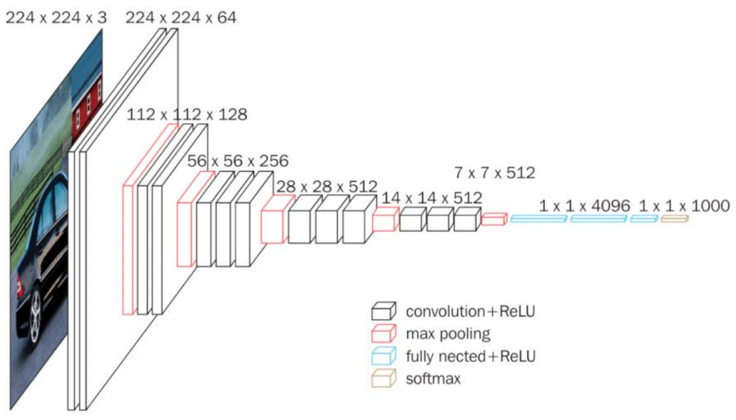


Fig.9 The configuration of VGG16Net
(Source: https://neurohive.io/en/popular-networks/vgg16/)

However, despite its satisfactory performance in multiple transfer learning tasks, this network has a greater number of parameters compared to AlexNet and GoogLeNet, which makes it computationally more expensive to evaluate and requires a considerable amount of memory in optimizing the learning parameters.

2.4.3 VGG19

Extremely similar to VGG16, the original VGG19 network has 19 weight layers and is trained on more than a million images from the ImageNet database. The network can identify 1000 categories with input size of 224*224. Fig.10 illustrates the structure of VGG19. It consists of five conventional layers and an output layer. Every conventional layer includes 2 or 4 conventional sub-layers, and there is one max-pooling layer between every two conventional layers.

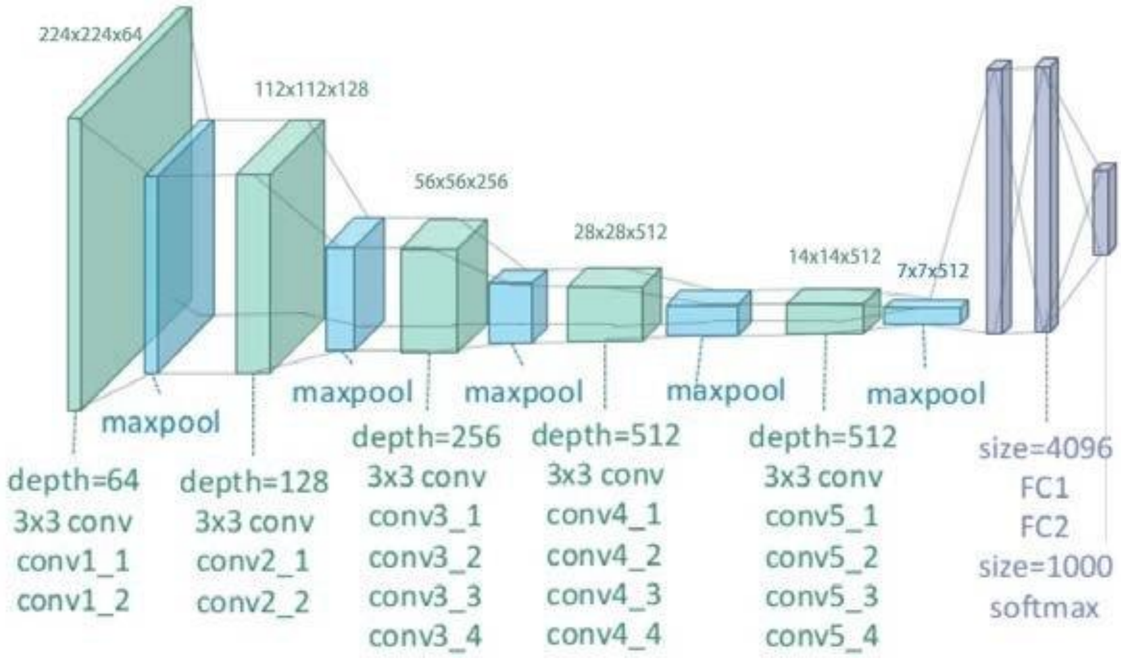


Fig.10 Structure of VGG19
(Source: https://towardsdatascience.com/transfer-learning-in-tensorflow-9e4f7eae3bb4)

In this paper, we also present a transfer learning solution to migrate the original VGG19 network to a relatively simplified map classifying network. To implement transfer learning, similar to VGG16, we loaded pre-trained weight, added a new output layer, and set the last few layers trainable during the training process.

3. Experiment and Discussion

As mentioned before, we will use MLPs, SVMs and CNNs to do the three sub-objectives here.

For MLPs, we tested the results from two hidden layers to four hidden layers. Because, we found that in general the performance of three-hidden-layer MLPs is better than two-hidden-layer MLPs and the prediction accuracy decreased as we increased the number of hidden layers to four. And for each layer in MLPs, we used similar approach to adjust the number of nodes in the layer.

Four kernel functions, linear kernel, RBF kernel, polynomial kernel and sigmoid kernel, are used in SVMs as introduced above. As for the parameters in these SVMs, each of kernel parameters varies from the exponential sequence and 5-fold cross validation is used to select the best fitting parameter combination. That is, first randomly choose one half (i.e., 200 instances) of the training images as the cross validation set. Then, divide the cross validation set into 5 subsets of equal size (i.e. each containing 40 instances). Each subset in turn is used to validate the classifier trained on the remaining 4 subsets. In this case, you have 5 trained SVMs and 5 validation subsets and the accuracy of cross validation is the mean value over the 5 validation subsets.

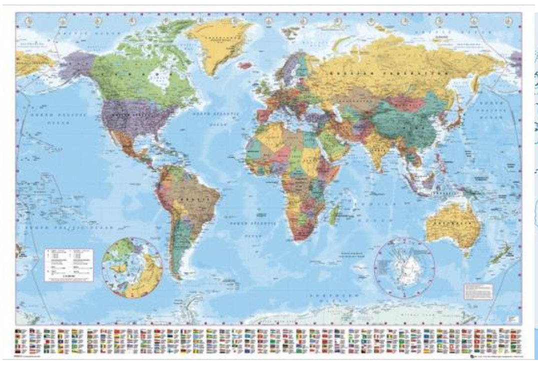
For parameter setting of CNNs, we started from an empirical model which has been proved the best parameter setting of CNNs for classical digital hand writing recognition problem, and then changed some of the parameters in CNNs to get the best results in our case. Because the initial parameter settings are different for each of the three methods and randomly selected images for training and testing are also different, which will influence the results of identification and classification, we conducted 10 sets of experiments for each of parameter setting and calculate the average testing accuracy for each parameter setting. Experiments have been conducted to test the influence of training size and batch size on the performance of map identification and classification. In our experiments, in convolutional layers and pooling layers, rectified linear unit (ReLU) will be used as the activation function, which is able to speed up the process of training deep neural network by including sparsity and preventing gradient vanishing problem (Chen et al., 2016). Stochastic gradient descent will also be used.

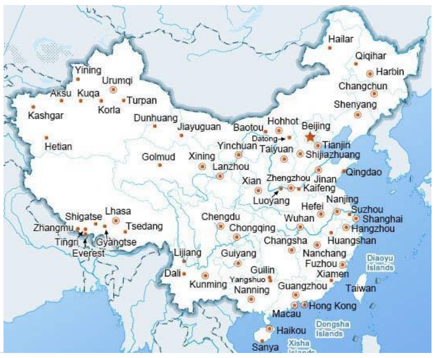
The image resolution, we used now is 100*120 pixels considering the shape of most images. More cases of image resolution will be used to test the effects of image resolution on the prediction accuracy of experiment results of the three methods. And, as we said in previous sections, we will also apply transfer learning to map identification and classification using pre-trained CNN architectures (GoogLeNet , VGG16 and VGG19) with the training data. We have and then compare the performance of pre-trained CNN architectures with the performance of the three methods we used. The performance of training weights of different number of layers in the pre-trained CNN architecture with different quantity of training data will be examined in this project.

3.1 Experiment Preparation

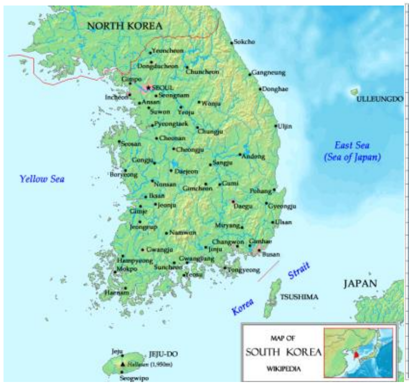
3.1.1 Data collection

The images we used in our experiments include map images and non-map images. The map images are about four regions: world map, China map, South Korea map and map of the US. In order to have a large number of region-maps for our study, we create a collection of 6000 maps which consists of 4000 synthetic generated maps and 2000 real-world maps scraped from the Internet. From the 2000 real-world maps, 400 map images (100 for each of the regions) are used for region classification without transfer learning, while in region classification using transfer learning, all of the 6000 maps are used. Then, 400 world map images (100 for each of the projections) are used for projection classification. And for map identification, 400 images including 200 map images and 200 non-map images are used. And for each of the tasks, 80% images are used for training and the other 20% images are for testing. The non-map images are manually downloaded from the Internet. Map images for four regions, for four projections and non-map images are shown below.


(a)World map



(b)China map



(c)South Korea map

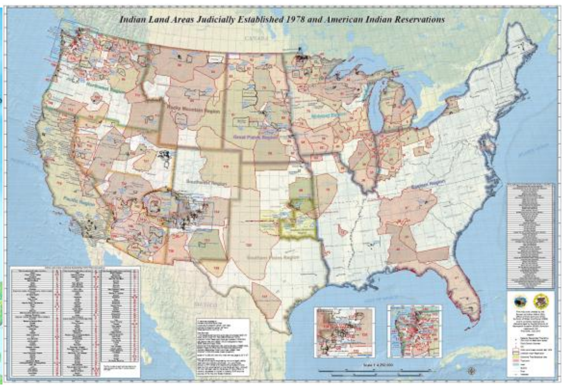

Indian Land Areas Judicially Established 1978 and American Indian Reservations
(d)US map

Fig.11 Examples of maps about four regions

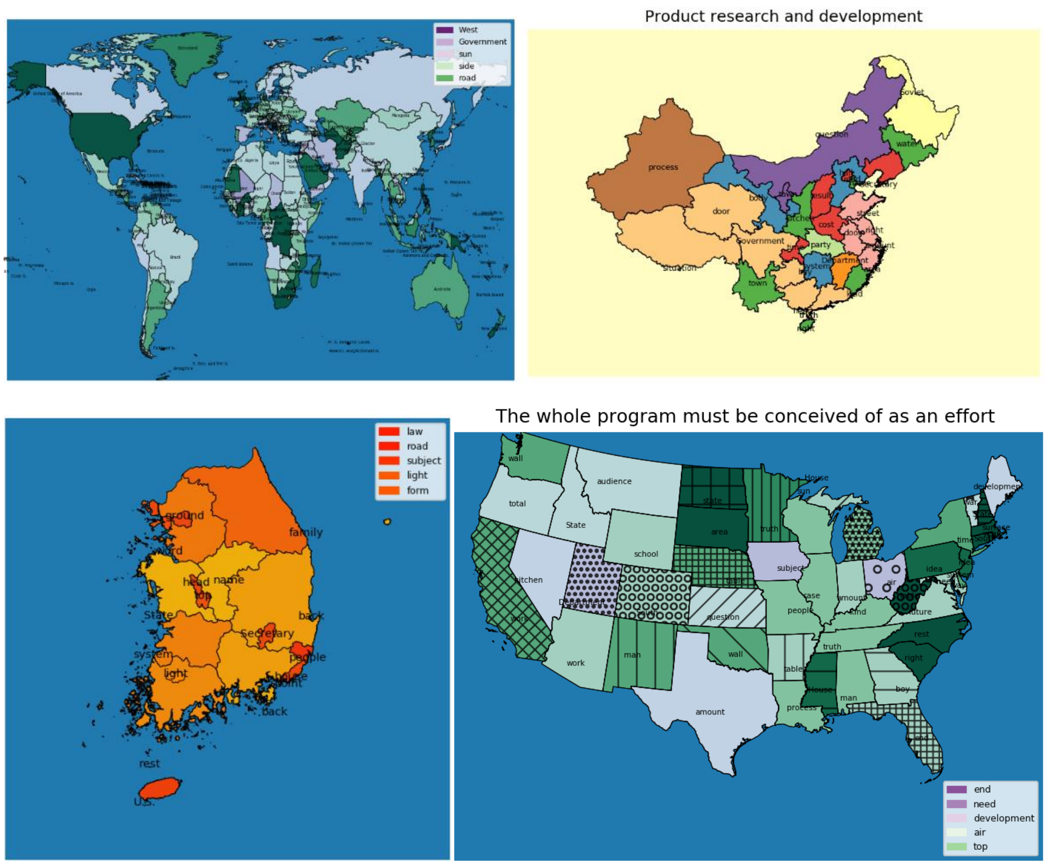
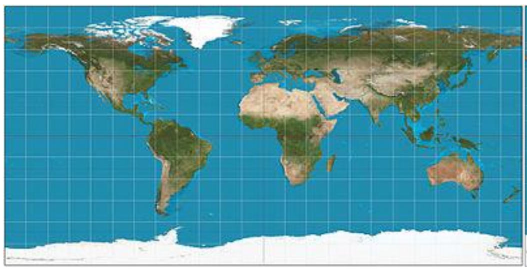
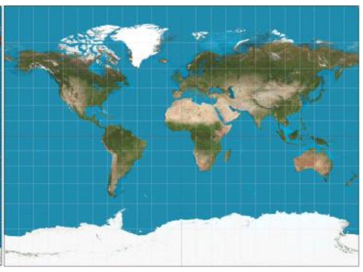
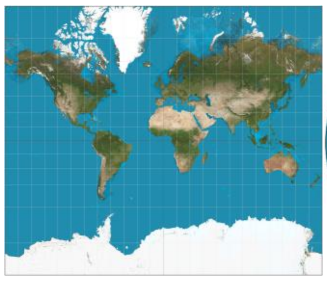
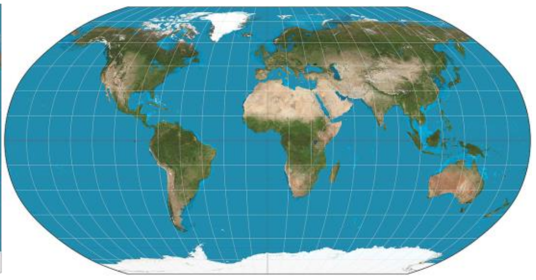

West
Government
sun
side
road
Product research and development
law
road
subject
light
form
The whole program must be conceived of as an effort
end
need
development
air
top

Fig.12 Examples of synthetic generated maps

(a)Equirectangular projection (b)Miller projection

(c)Mercator projection (d)Robinson projection

Fig.13 Examples of maps about four projections

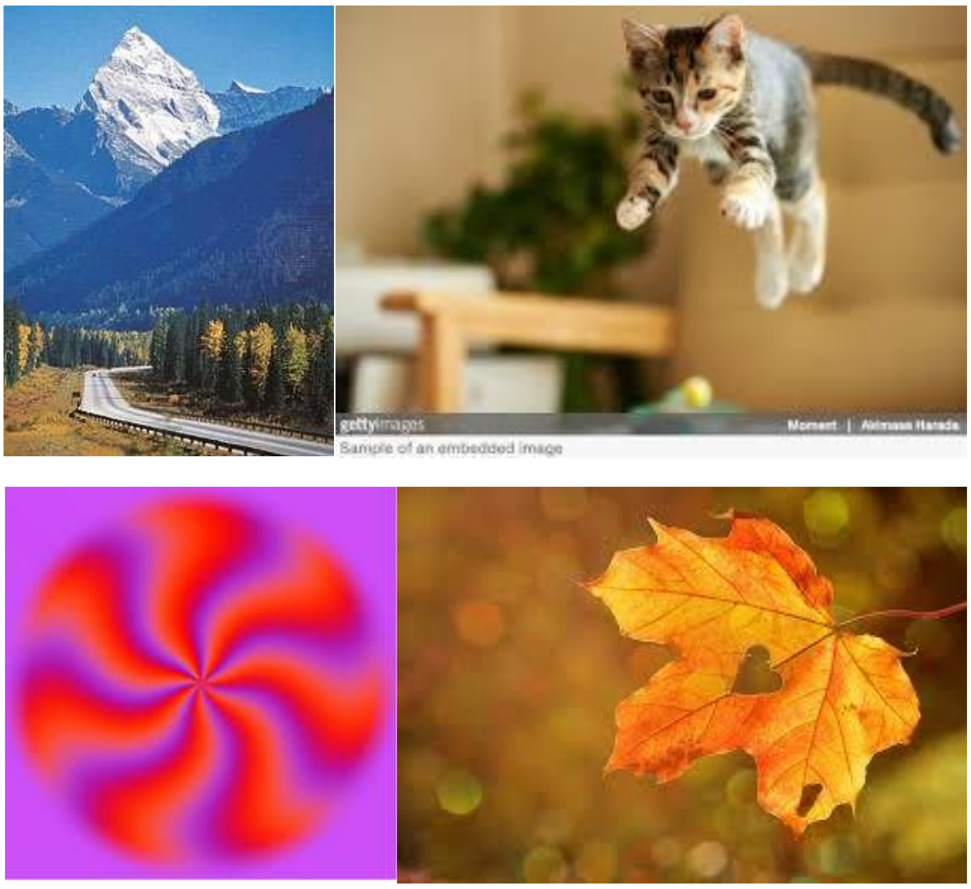


Fig.14 Examples of non-map images

The rest of this section is organized as follows. First, we introduce how to generate a synthetic map dataset following the standard visual design pipeline, including data acquisition and preparation, visual mapping, and visualization techniques. Next, we present the methods and techniques that we used to scrap real-world data from web. We discuss the limitation and future work in the end. In this study, we use Basemap toolkit for generating maps. The matplotlib Basemap toolkit is a library for plotting 2D data on maps in Python.

3.1.1.1 Synthetic map generation

Most maps in our daily life can be regard as typical type of visualization, which convey the specific spatial location and present ideas and concept within that location. In this study, we follow standard visualization design pipeline to generate number of maps automatically. This design process requires first an understanding of the underlying nature of the data to be visualized, then the selection of visual dimensions that mapping the data into visual symbols. The following figure shows the basic procedures that used to generate a region-based map.

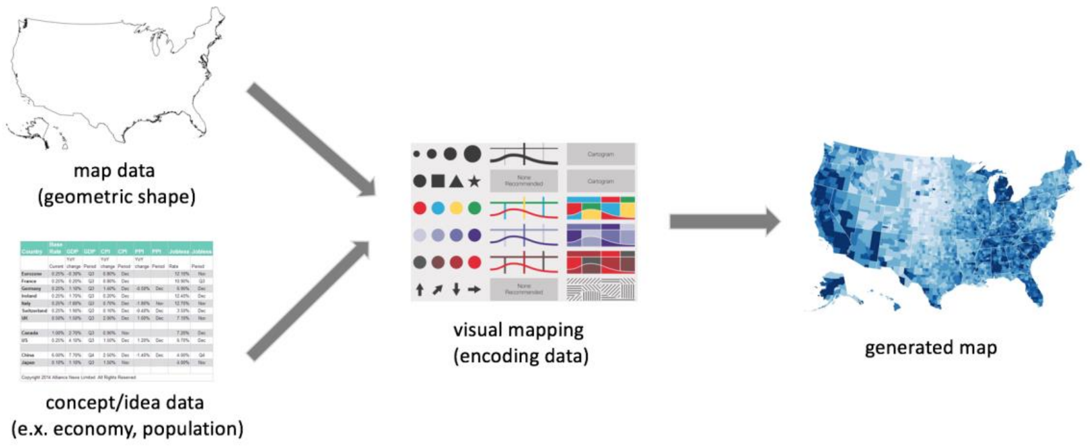


Fig.15 The basic pipeline used to generate synthetic map graph

Underlying Data: In our study, we aim to generate map of four different regions: the US, China, South Korea, and the whole world. Thus, we need to obtain the geometry (polygons) data that define the boundary of each region. In geography, shapefile format is a common standard for representing geospatial vector data (ESRI et al., 1998). The shapefile format is a digital vector storage format for storing geometric location (the coordinates of a given polygon that define the area) and associated attribute information such as the name of specific region. The dimension used in shapefile always be the 2-D, i.e., x and y coordinates. In this study, we only generated 2D map since most of real-world map are in a 2D form. Another notable truth within the geometric data is the granularity, that is, the range of area division. Generally, we define it as administrative level (admin level). Take the US as an example, level 0 correspond to the country, level 1 correspond to each state, and level 2 correspond to county side. This range can be further divided into more fine level such as census tract. In our study, we only generated maps using these three administrative levels, and most of them are belongs to level 1.

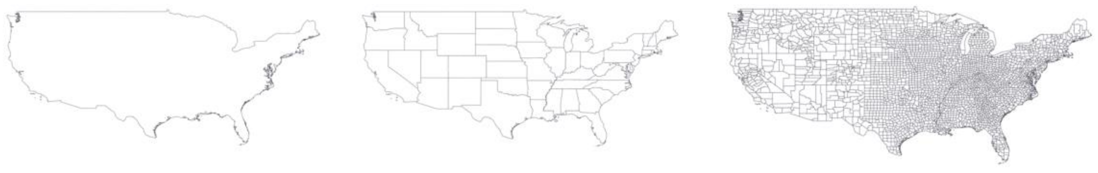


Fig.16 Three different administrative level of us map

Besides the geometry data that used to define the spatial information, real-world data also convey information about the area such as population density, house income, weather condition, and election results. The data used to store such information can be diverse, and the attributes can be continuous and discrete. In this study, we assign each level1 area (e.g. state for US) a randomly generated value. The value is further used to be mapped with different color in the visual mapping phase.

Visual Mapping: After we identify the geometric boundary and related attribute, we need to map these data into visual symbols, this stage is called visual mapping. Visual mapping stage contains a lot of actions such as color mapping, texture mapping, and size mapping. These actions can be widely divided into three categories, namely, basic visual variables (such as color, size, position) mapping, layer style (such as ocean, street map) mapping, and visual elements (such as text and legend) mapping. The figure below shows an example of these three types mapping. In the rest of this section, we will introduce these three types of mapping respectively.

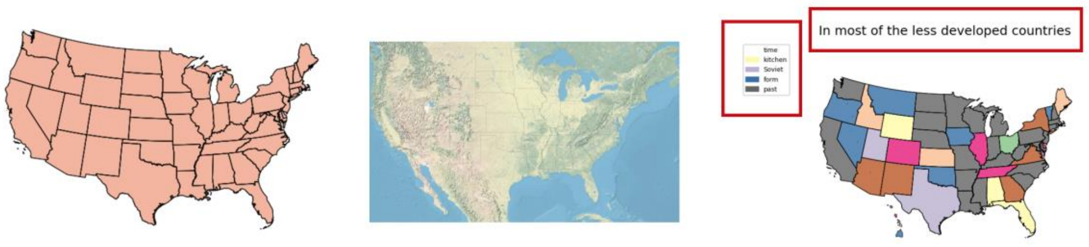
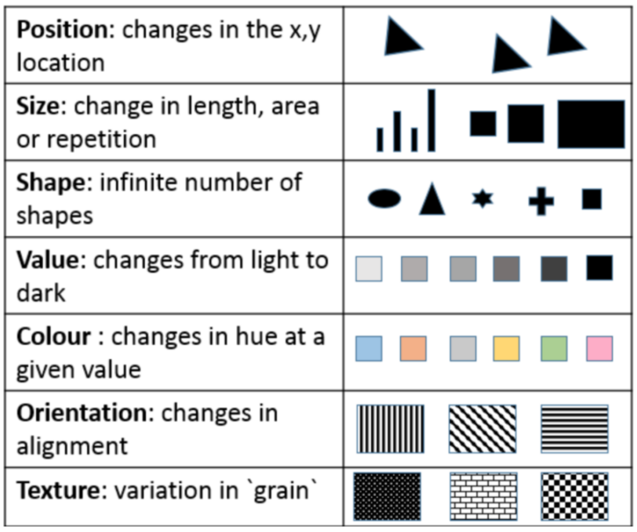


Fig.17 From left to right: basic visual variables (color) mapping, layer mapping, and visual elements mapping (title and legend)

Bertin proposed the semiology of graphics to create a mapping from the data (digital representation) to a visual representation (Bertin et al., 1983). In the original work, he introduced seven basic visual variables: position, size, shape, color, value, texture, and orientation. Fig.18 shows 7 visual variables proposed by Bertin. Over the years, the system has been further expanded by several later publications and include more static and non-static variables. Mackinlay (Card, 1999) presented a set of visual variables and sort these variables according to specific tasks. White (2017) stated the symbolization and visual variables mainly for geometric visualization. In our study, we mainly follow Bertin's visual variables and updated them to match geospatial circumstance. Besides, we add transparency and aspect ratio as the supplement of visual variables. Here we introduce the detail of how to visualize data using these visual variables.



Position: changes in the x,y location	▶ ▶ ▶
Size: change in length, area or repetition	ılı ■ ■ ■
Shape: infinite number of shapes	⬬ ▲ ✶ ✚ ■
Value: changes from light to dark	⬜ ▪ ▪ ▪ ▪ ■
Colour : changes in hue at a given value	🟦 🟧 ⬜ 🟨 🟩 🟪
Orientation: changes in alignment	▥ ▧ ▤
Texture: variation in `grain`	▩ ▦ ▒

Fig.18 Bertin's visual variables

From a viewpoint of implementation, overlayers can be regarded as the warp image background of a map. Many organizations provide overlayers and their API for us to call. In our study, we used 16 different types of overlayer by calling service from ESRI (an international supplier of geographic information system software, web GIS and geodatabase management applications).

Visual elements mapping: The last step we do for generating synthetic map is to add visual elements into map. In our study, we choose the following visual elements: titles, legend, name on each admin1 level area, and latitude and longitude lines. To generate title for each visualization, we extracted the 100 most frequent sentences in the Brown Corpus (Marcus et al., 1993) under 'government' topic using NLTK library. Similarly, we select 100 most frequent words nouns in the Brown Corpus using part of speech tagging. For legend that used to indicate the color scheme of the map, we identify the size and position of legend randomly. Moreover, for latitude and longitude lines, we selected different margin for two lines. Through these setting, we assign different visual elements for each map. Fig.19 shows a US map with all four different visual elements.

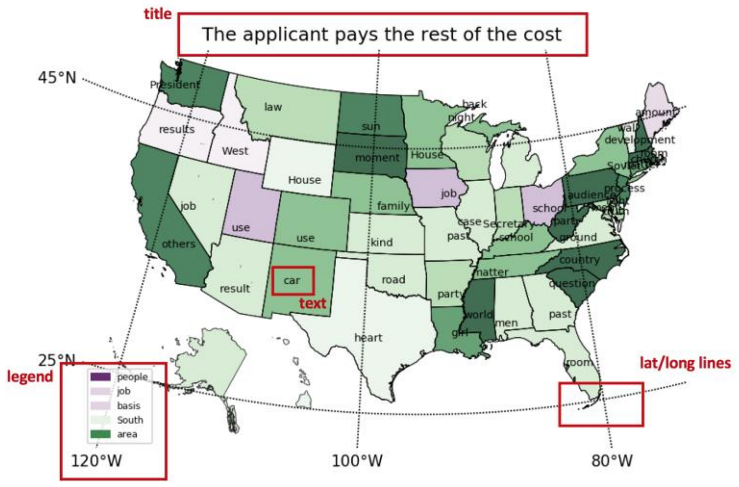


Fig.19 US map with four types of visual elements

3.1.1.2 Real world map collection

In this section, we briefly introduce how to collect real-world maps from web. First, we use the google image download toolkit to scrap images from google automatically. The basic structure of logic to download images using this toolkit can be found in Fig.20. Due to the availability of image count on google image, we search three keywords: region + map, region + visualization, region + weather using four three different language: English, Chinese, and Korean on google image. Then we download all images for each query, which provide us a lot of maps. However, due to the rough search constraints, we got a lot of noise data, i.e., the incorrect maps. For example, the US map may give us a map of Canada. Thus, we did an extensive manually review to filler those incorrect images, which results in a final map collection contain 2000 maps, with 500 maps for each region respectively.

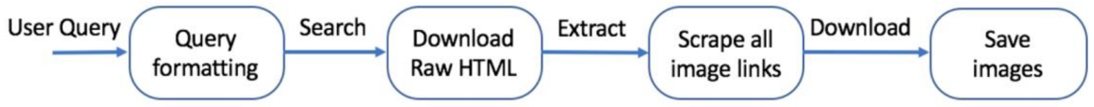


Fig.20 The algorithm logic to download images

3.1.2 Libraries and tools used

In our experiment, we used MLPs, SVMs and CNNs to solve the mentioned problem of map identification and classification. Keras package in Python is used in this project to accomplish MLPs and CNNs. Keras package is a commonly used package to implement different kinds of neural networks. It can run on top of Tensorflow, designed to enable fast experimentation. Tensorflow is currently the most popular package for deep learning developed by Google. And for SVMs, we used a package called libsvm in Python to implement four kinds of SVMs: Linear kernel SVM, RBF kernel SVM, Polynomial kernel SVM and Sigmoid kernel SVM. As for the parameters in these SVMs, cross validation is used to select the best fitting parameters. For parameter setting of CNNs, we started from an empirical model which has been proved the best parameter setting of CNNs for classical digital hand writing recognition problem, and then changed some of the parameters in CNNs to get the best results in our case. Because the initial parameter settings are different for each of the three methods and randomly selected images for training and testing are also different, we conducted 10 sets of experiments for each of each parameter setting of the three methods and calculate the average testing accuracy for each parameter setting.

If we want to train weight parameters in layers of pre-trained CNN architecture in transfer learning, much more training data of map images are needed as mentioned above. In our experiments, totally 6000 map images are used for transfer learning. But this also causes a computational problem, which will take lots of time for training the CNNs using regular computers without GPU. Therefore, Google Colaboratory is used in our transfer learning experiments. Google Colaboratory a free Jupyter notebook environment that requires no setup and runs entirely in the cloud, and with Google Colaboratory you can access powerful computing resources (e.g. TensorFlow with GPU) all for free from your browser. With the help of Google Colaboratory, much training time is saved in our experiment.

3.2 Maps Identification

For MLPs and CNNs, we compared the average results between different number of hidden layers. Because the performance difference between MLPs and CNNs with different number of hidden layers is significantly large, difference but between MLPs and CNNs with same number of hidden layers is not obvious. The testing accuracy results are shown in Table.1 and Fig.21 below. From the results, we can find that the best results are from MLPs with 3 hidden layers. The architecture of the best result is 600-200-100-1, which means that there are 600 nodes in the first hidden layer, 200 nodes in the second hidden layer and 100 nodes within the third hidden layer. For this setting, the training accuracy is 0.96 and testing accuracy is 0.97.

Table.1 Testing accuracy results of MLPs

# Hidden Layers	Training Acc	Testing Acc
2	0.88	0.79
3	0.91	0.90
4	0.41	0.49

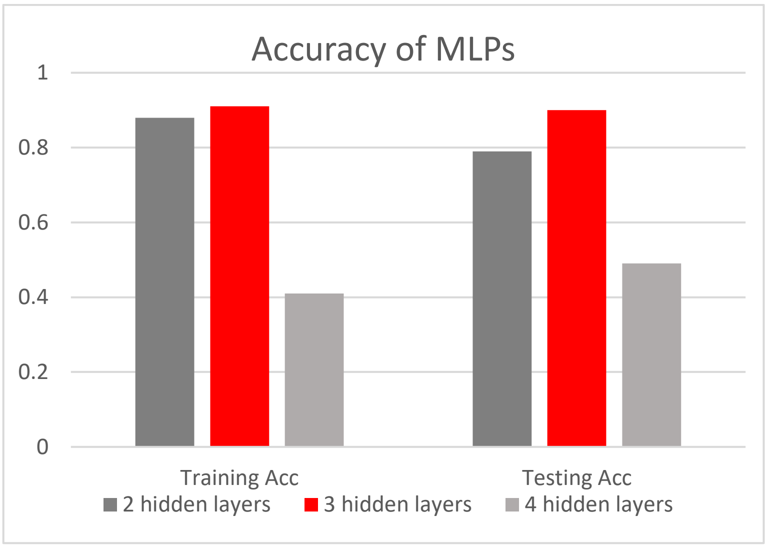


Fig.21 Testing accuracy results of MLPs

For SVMs, the testing results of four kinds of classification kernel were compared in this experiment. The results are shown in Table.2 and Fig.22 below. According the results, Linear kernel and Polynomial kernel SVMs outperforms the other two.

Table.2 Testing accuracy results of SVMs

Kernel Function	Training Acc	Testing Acc
Linear	1	0.87
RBF	0.57	0.63
Polynomial	1	0.92
sigmoid	0.58083	0.565

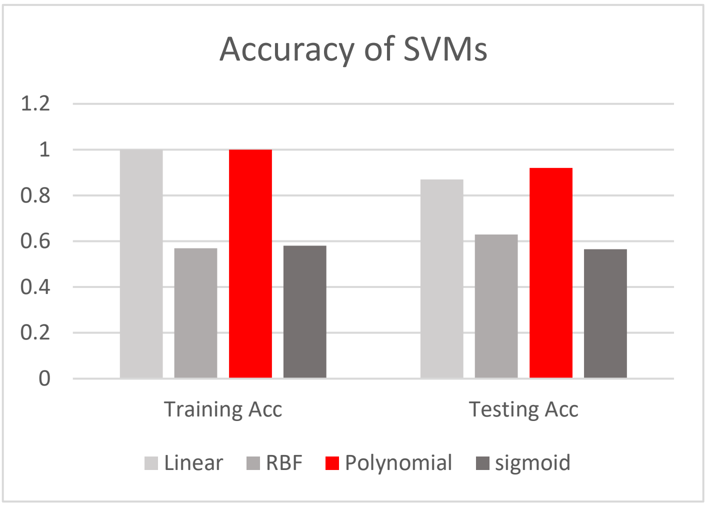


Fig.22 Testing accuracy results of SVMs

As for CNNs, the best results are from CNNs with 2 convolutional layers. The best training accuracy could reach 1, and testing accuracy can reach 0.96. The general results of CNNs for map identification is shown in Table.3 and Fig.23 below.

Table.3 Testing accuracy results of CNNs

# Hidden Layers	**Training Acc**	**Testing Acc**
2	0.99	0.93
3	0.99	0.92
4	0.94	0.90

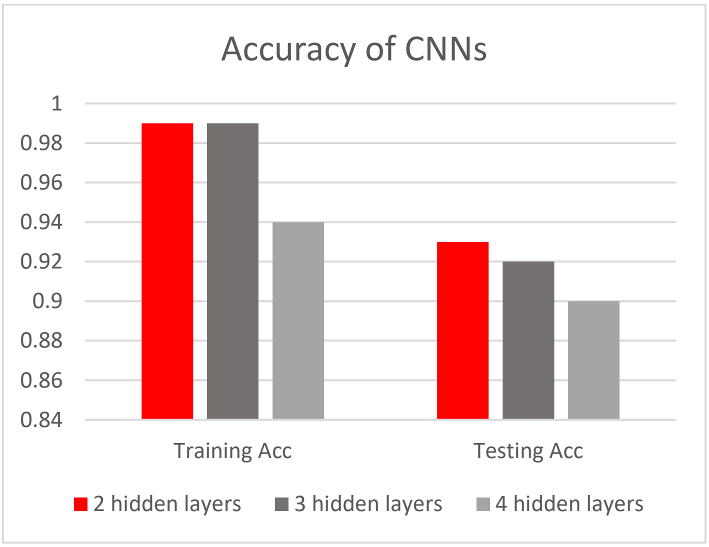


Fig.23 Testing accuracy results of CNNs

Comparing the experiment results of the three methods, we can find that all of the three methods can get relatively good results. We think that is because the difference between map images and non-map images is big enough for all of the three methods classify them correctly. And CNNs with 2 convolutional layers got the best results. The testing accuracy of best result we got is 100%. The images and corresponding predicted label are shown as follows:

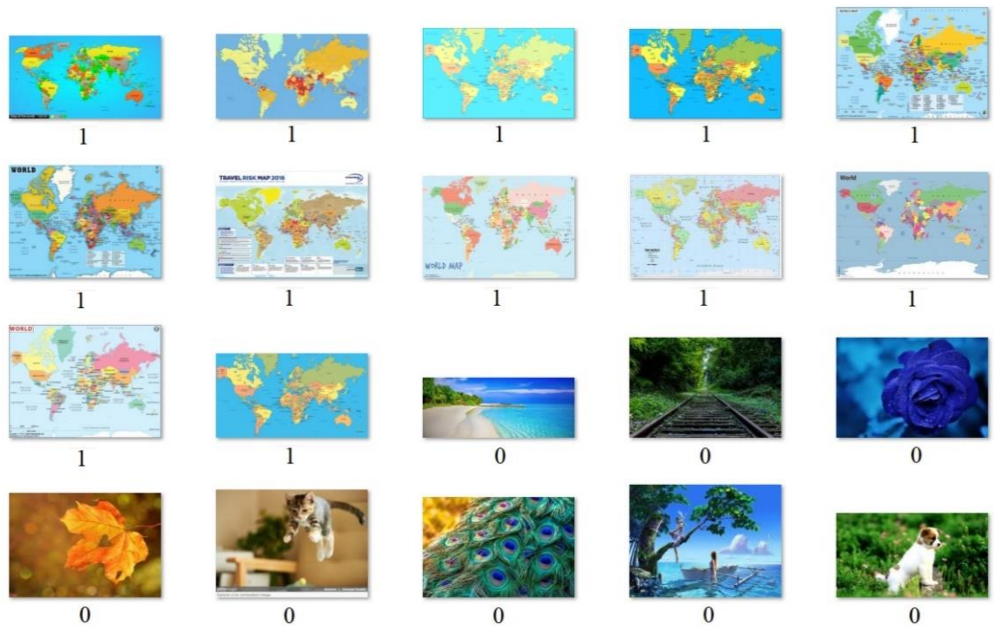


Fig.24 The images and corresponding predicted label

3.3 Region classification

The testing accuracy results of MLPs are shown in Table.4 and Fig.25 below. From the results, we can find that both training accuracy and testing accuracy of the results are all very bad for MLPs with different number of hidden layers. The accuracies are all about 0.25, which is the probability for one class in the four. The predicted value is all for one class. MLPs are not powerful enough to tell the difference between distinct regions.

Table.4 Testing accuracy results of MLPs

# Hidden Layers	Training Acc	Testing Acc
2	0.25	0.25
3	0.25	0.25
4	0.25	0.26

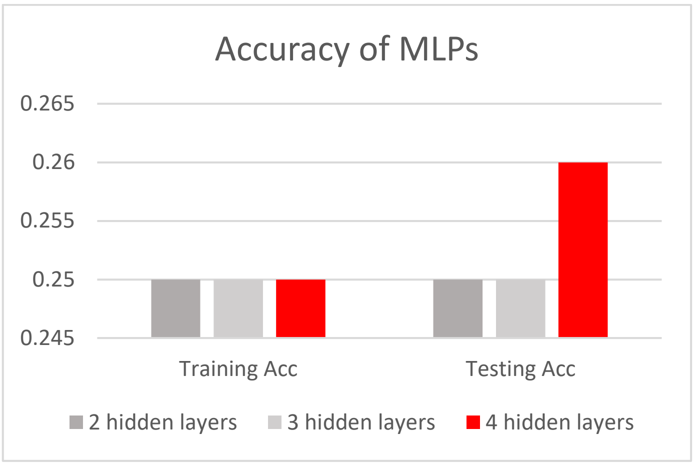


Fig.25 Testing accuracy results of MLPs

For SVMs, the testing results of four kinds of classification kernel were compared in this experiment. The results are shown in Table.5 and Fig.26 below. According to the results, Linear kernel and Polynomial kernel SVMs significantly outperforms the other two. They can classify the four classes with satisfactory results, but the other two's results are very bad. That means that the inner function of region classification fits linear and polynomial kernels well, but it cannot fit RBF and Sigmoid kernels. And we can also find that the training accuracy of RBF kernel is good, but the testing accuracy is very poor. So, the RBF kernel overfits the inner function of region classification.

Table.5 Testing accuracy results of SVMs

Kernel Function	**Training Acc**	**Testing Acc**
Linear	1	0.91
RBF	0.90	0.22
Polynomial	1	0.91
sigmoid	0.27	0.12

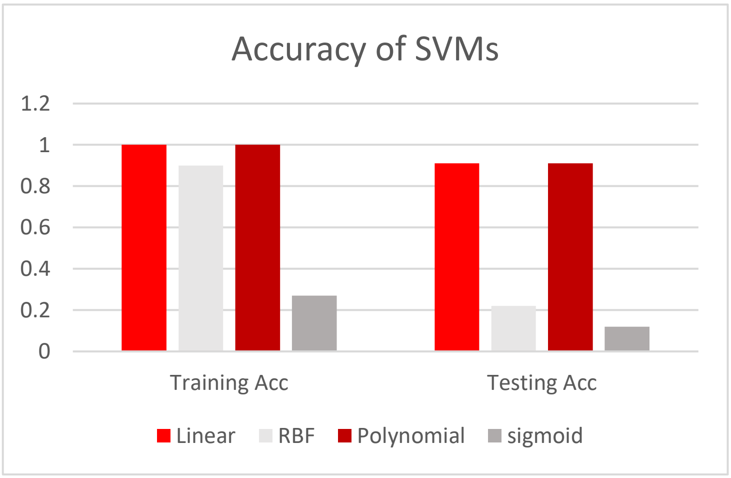


Fig.26 Testing accuracy results of SVMs

As for CNNs, the best results are from CNNs with 2 convolutional layers. The best training accuracy could reach 1, and testing accuracy can reach 0.92. The general results of CNNs for map identification is shown in Table.6 and Fig.27 below. The CNNs can also get results with high testing accuracy.

Table.6 Testing accuracy results of CNNs

# Hidden Layers	**Training Acc**	**Testing Acc**
2	1	0.92
3	1	0.87
4	0.70	0.55

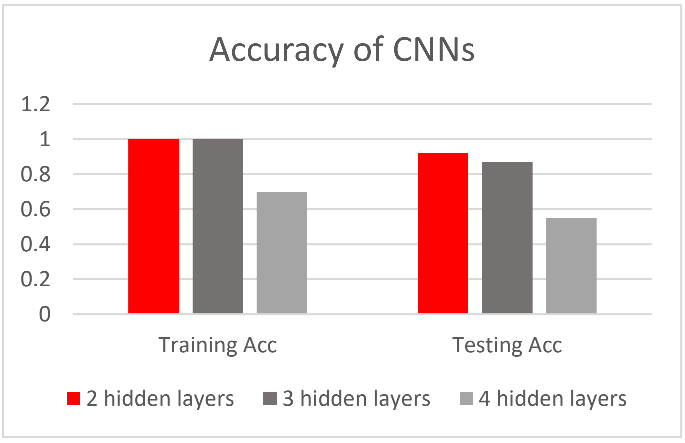


Fig.27 Testing accuracy results of CNNs

Comparing the experiment results of the three methods, we can find that SVMs and CNNs can also get very good classification results with some parameter settings but MLPs cannot get good results. We think that is because the difference between map images of different region is much smaller than the difference between map and non-map images. In this case, MLPs cannot tell the difference and get good classification results. And CNNs with 2 convolutional layers can still get the best results. The testing accuracy of best result we got is 100%. The images and corresponding predicted label are shown below in Fig.28. World map, China map, South Korea map and US map are represented by 0, 1, 2 and 3 respectively.

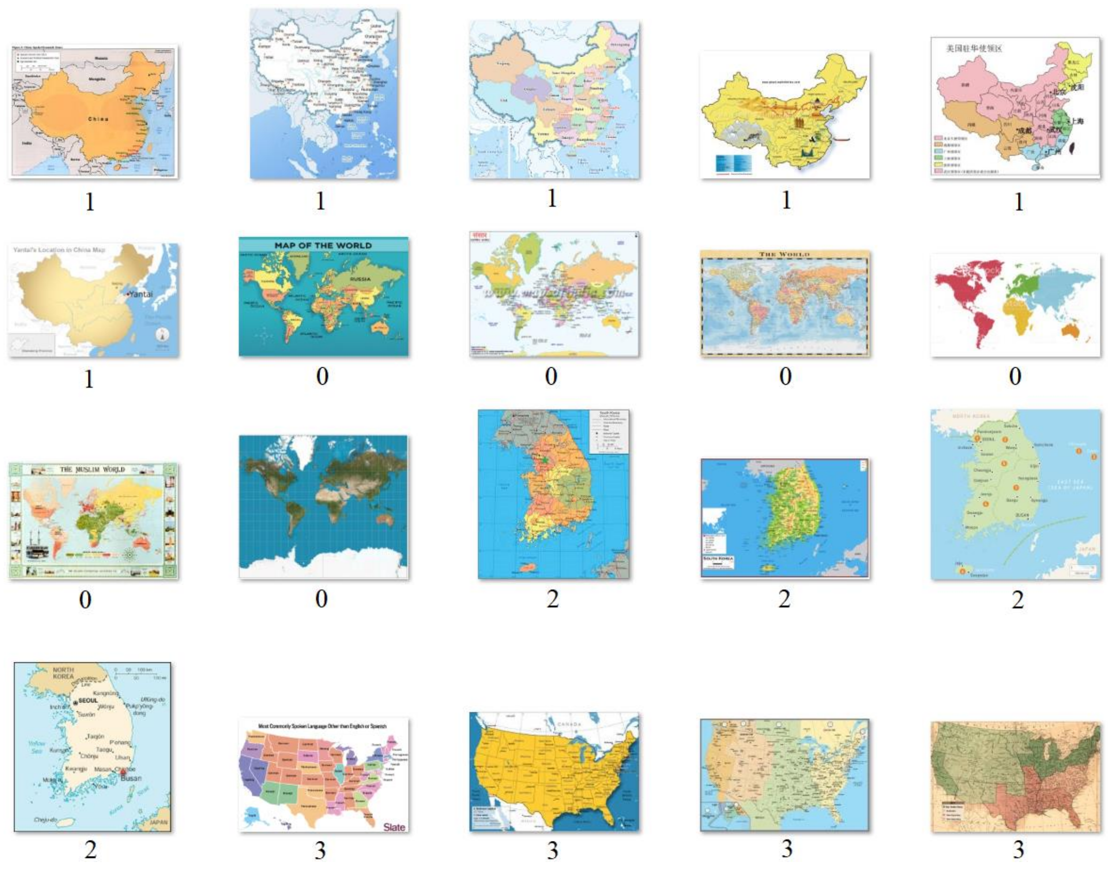


Fig.28 The images and corresponding predicted label

3.4 Region classification with transfer learning

For region classification, transfer learning is also used as mentioned before. The experiment design and experiment results are shown below for the three CNN architectures respectively.

3.4.1 GoogLeNet

There are 8 groups of experiments with different sample size, 400, 1200, 2000 …, 6000. For each sample size, we use 80% of data for training and 20% of data for testing. In each experiment, we try to train 9 different sets of parameters (9 sets of sublayers) to see how the number of trainable parameters can affect the testing accuracy. In the experiments, we use batch size = 40 for all sample sizes. We choose 20 epochs and 0.0004 as our learning rate.

When sample size is bigger than 2000, the experiment become very time consuming. Therefore, we only train 5 different set of parameters for sample size = 2800, 3600, 4400, 5200 and 6000, to see how the number of trainable parameters can affect the testing accuracy. The last fully connected layers and output layers are replaced by new ones based on our application.

In order to do fine tuning with Inception V3 model, we exclude the output layer in the original model and add three layers, including two fully connected layers and one output layer. With the change of the model, the output will be four-dimensional vector corresponding to four classes.

As mentioned, the configuration of Inception V3 model is very complex, and many layers don’t have the weight parameters to be trained. Therefore, we only count the sublayers as one sublayer when its type named ‘Conv2D’ (see Table.7). In our nine sets of trainable sublayers, the number of trainable layers and number of trainable parameters are given in Table.8. Sublayers 310-312 refer to the 3 new added layers.

Table.7 Sublayers from 283-310 in original Inception V3 model

Sublayer No.	**Sublayer name**	**Sublayer No.**	**Sublayer name**
283	conv2d_87	**297**	batch_normalization_92
284	conv2d_91	**298**	batch_normalization_93
285	batch_normalization_87	**299**	conv2d_94
286	batch_normalization_91	**300**	batch_normalization_86
287	activation_87	**301**	activation_88
288	activation_91	**302**	activation_89
289	conv2d_88	**303**	activation_92
290	conv2d_89	**304**	activation_93
291	conv2d_92	**305**	batch_normalization_94
292	conv2d_93	**306**	activation_86
293	average_pooling2d_9	**307**	mixed9_1
294	conv2d_86	**308**	concatenate_2
295	batch_normalization_88	**309**	activation_94
296	batch_normalization_89	**310**	mixed10

Table.8 Trainable number of layers and trainable parameters

# Trainable sublayers	**Corresponding sublayer No.**	**# Trainable parameters**	**# Nontrainable parameters**
0	310-312	279,300	21,802,784
1	299-312	673,028	21,409,056
2	294-312	1,329,924	20,752,160
3	292-312	1,772,292	20,309,792
4	291-312	2,214,660	19,867,424
5	290-312	2,657,028	19,425,056
6	289-312	3,099,396	18,982,688
7	284-312	4,648,452	17,433,632
8	283-312	5,434,884	16,647,200

In this part, we show the experimental results. In Table.9, we show the experiment results for different sample sizes with different set of number of trainable sublayers. Fig.29 shows the results for sample size = 400, 1200 and 2000 with 9 different sets of trainable layers. Fig.30 shows the results for sample size =400, 1200, 2000, 2800, 3600, 4400, 5200 and 6000 with 5 different sets of trainable layers.

Table.9 Testing accuracy of different sample size and number of trainable layers

# Trainable layers / # Images used	0	1	2	3	4	5	6	7	8
400	0.36	0.58	0.56	0.51	0.54	0.42	0.55	0.53	0.56
1200	0.45	0.67	0.66	0.70	0.65	0.68	0.60	0.69	0.63
2000	0.43	0.66	0.71	0.74	0.70	0.70	0.80	0.77	0.79
2800	0.48	N/A	N/A	N/A	N/A	0.73	0.69	0.68	0.70
3600	0.52	N/A	N/A	N/A	N/A	0.70	0.72	0.62	0.67
4400	0.44	N/A	N/A	N/A	N/A	0.70	0.69	0.65	0.74
5200	0.43	N/A	N/A	N/A	N/A	0.64	0.72	0.79	0.77
6000	0.45	N/A	N/A	N/A	N/A	0.66	0.76	0.74	0.75

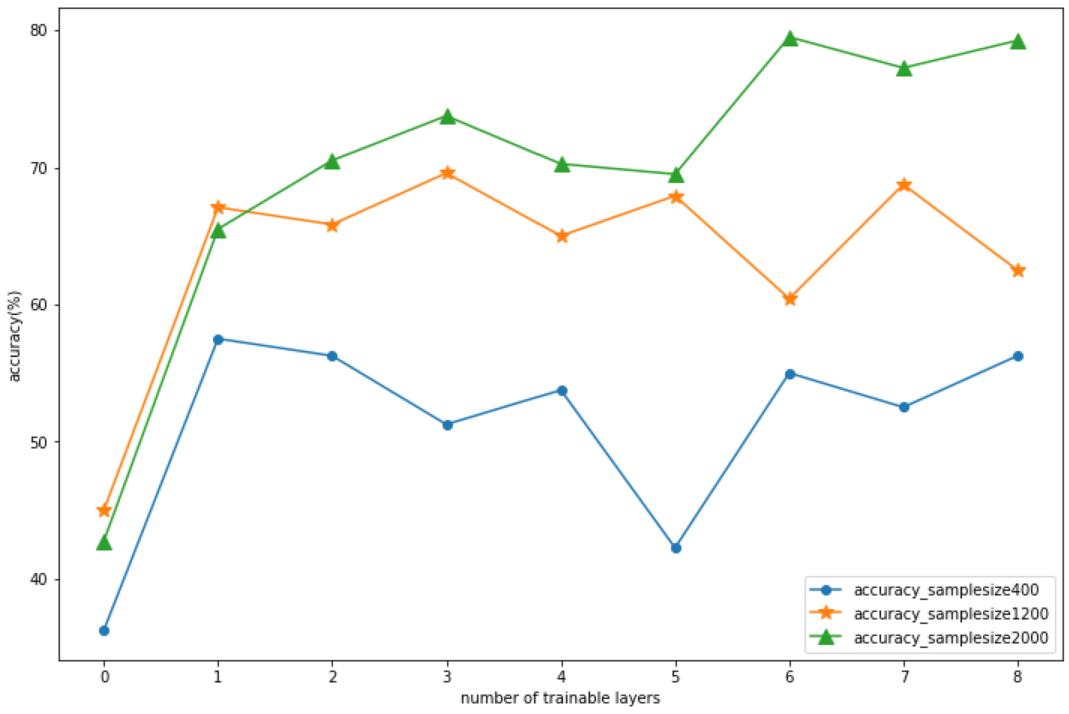


Fig.29 Sample size and training accuracy (number of layers: 0, 1, 2 …, 8)

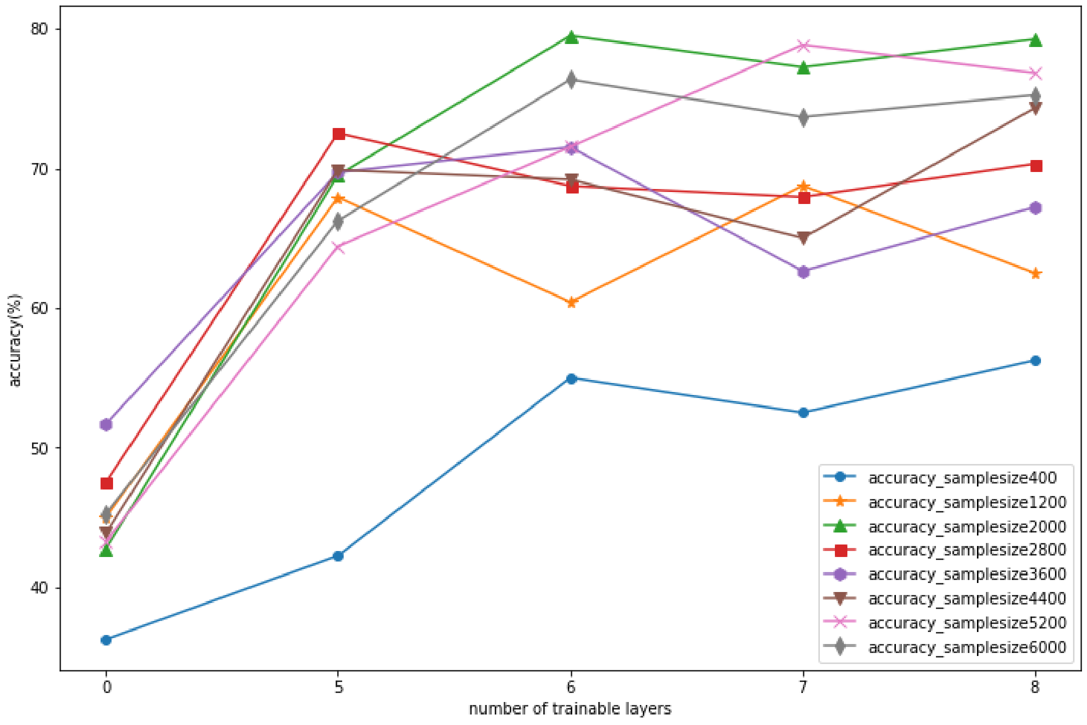


Fig.30 Sample size and training accuracy (number of layers: 0, 5, 6, 7, 8)

From Table.9 and Fig.29, we can see the sample size affects the testing accuracy. When sample size = 400, the testing accuracy is around 35% for layer = 0 and 52% for layers between 1-8. When sample size =1200, the testing accuracy is around 45% for layer = 0 and 65% for layers between 1-8. When sample size = 2000, the testing accuracy is around 42% for layer = 0 and 70% for layers between 1-8. Therefore, larger sample size will have higher testing accuracy. However, according to Fig.30, the testing accuracy did not have significant improvement when sample size increase from 2000 to 6000. This is probably because there is no big difference for such a large neural network to train on the level of 'thousand' number of pictures. It is probably that when the sample size increase to more than ten thousand, the accuracy will change.

Another interesting result is that when the number of trainable layers = 0, i.e. when the number of trainable parameters is small, the testing accuracy keeps around 45% when sample size is bigger than 1200. But when the number of trainable layers is larger than 0, the testing accuracy is around 70% when sample size is bigger than 1200. This means that only fine tuning a small set of trainable parameters in Inception V3 model cannot generate good classification performance. It also confirms that with factorization convolution, Inception V3 model can deal with overfitting problem really well. Even with 5 million trainable parameters, the performance is still good. When compare with the results from the VGGNets, we can see the benefit of the Inception V3 model more clearly.

3.4.2 VGG16

The original configuration of VGG16Net is shown above. In our experiments, the initial two fully-connected layers with 4096 nodes are replaced by two fully-connected layers with 128 nodes. That is because our task only has 4 classes while the original task of VGG16Net has more than one thousand classes. And our output layer has four nodes corresponding to four classes. We first conduct transfer learning with only the three added layers trainable and then then fine-tune the weights in former layers. From the perspective of trainable layers, we do eight sets of experiments, training 3 to 10 layers of this architecture to test the influence of number of trainable layers. And we also examine the influence of training data size. Similar with GoogLeNet, we have 6000 map images, and 1500 for each of the regions. We conduct the 8 sets of experiments using 100, 300, 500 …, 1500 map images for each region. And 80% of the images used are training data, while the rest 20% images are testing data. That means, we have 64 sets of experiment testing the influence of number of trainable layers and input data size. Because we don't have much time for training, for each of the experiment set, the number of epochs is 20.

The numbers of parameters in each of the trainable layer in the experiments are shown in the following table.

Table.10 Layer type and number of parameters

Trainable Layers	**Number of parameters**
conv1 (Conv2D)	2359808
conv2 (Conv2D)	2359808
conv3 (Conv2D)	2359808
MaxPooling2D	0
Flatten	0
Fully-Connected Layer 1	262272
Fully-Connected Layer 2	16512
Output Layer	516

In Table.11, we show the experiment results for different sample sizes with different set of number of trainable layers. Fig.31 shows the results for sample size = 400, 1200, 2000, 2800, 3600, 4400, 5200 and 6000 with 8 different sets of trainable layers.

Table.11 Testing accuracy of different sample size and number of trainable layers

# Trainable layers / # Images used	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
400	0.73	0.26	1.00	0.26	0.21	0.21	0.26	0.26
1200	1.00	0.76	1.00	0.25	0.23	0.27	0.27	0.27
2000	0.73	0.77	1.00	0.27	0.27	0.27	0.27	0.23
2800	0.27	1.00	0.51	0.27	0.27	0.27	0.24	0.24
3600	0.99	0.99	1.00	0.25	0.23	0.25	0.26	0.23
4400	0.25	0.99	0.25	0.25	0.25	0.25	0.25	0.24
5200	0.72	0.98	0.72	0.24	0.26	0.24	0.24	0.24
6000	0.72	0.27	0.74	0.24	0.27	0.24	0.24	0.24

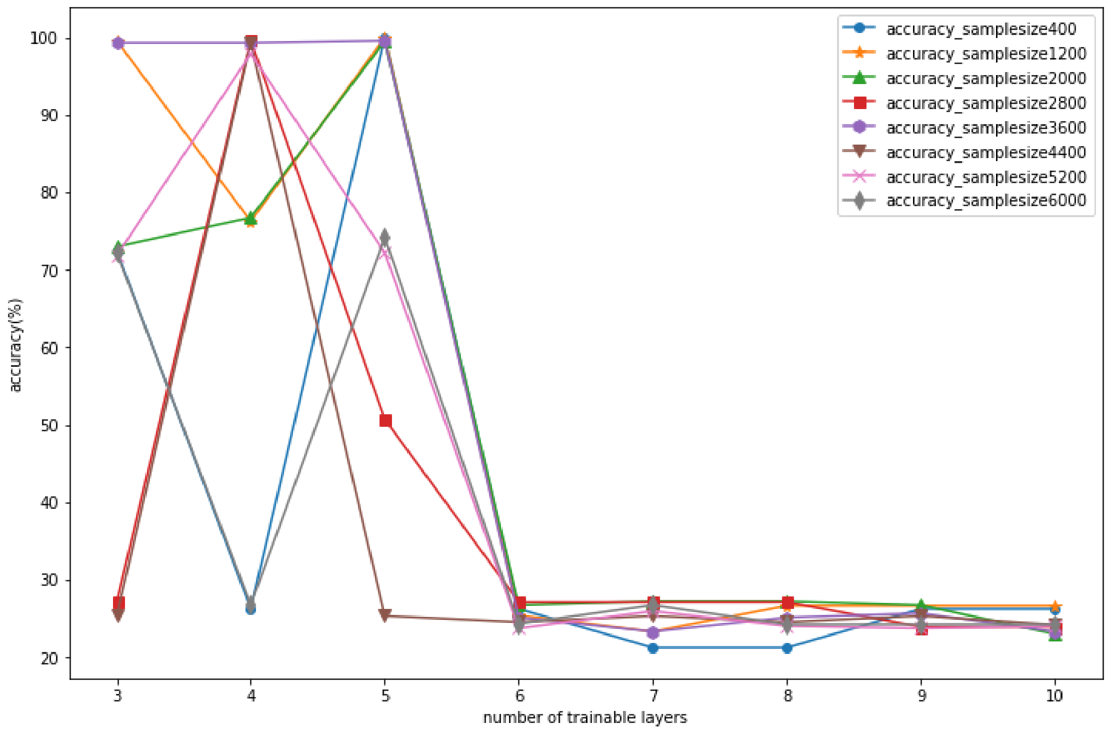


Fig.31 Accuracy rate of different sample size and number of trainable layers

From the results shown above, in many cases, transfer learning using VGG16 can get very good classification results, which means that this method can be a good solution to region classification problem. In general, we can see the accuracy rate decreases as number of trainable layers increases. That is because, when more layers are trainable, there are more weight parameters in VGG16 to be trained. And the training data we have is much less than the original training data for VGG16. And the fourth and fifth layer counting from the output layer in VGG16 have no weight parameter to be trained, the numbers of trainable parameters for 3, 4 and 5 trainable layers are the same. Therefore, the situation for them should be the same. We can see that, in these three cases, the accuracy rates are not stable and change a lot. Sometimes, it is 100% while it can be 25% for some other cases. I think the reason is that our epoch number is only 20 may not always enough for our model to get good classification results.

And with the increase of number of map images used, there is no significant improvement of classification accuracy. The most possible for this is that the generated map images don't contain the complex features compared with real world maps from the Internet. So, training more data cannot provide our model more information about the four classes.

3.4.3 VGG19

Considering the extreme similarity between VGG16 and VGG19, we constructed the output layers and parameters just like VGG16.

Likewise, we investigated training/validation accuracy's relationship with epochs, and trainable layer number.

To see the trainable layers' impact on the final accuracy, we conducted a series of experiment with different trainable layers' number from 3 to 10.

Table.12 and Fig.32 show the convergent accuracy's relationship with trainable layers for different input size. Larger dataset can also achieve higher accuracy in a good sense.

Table.12 The convergent accuracy's relationship with input data size

# Trainable layers / # Images used	3	4	5	6	7	8	9	10
400	0.20	0.80	0.80	0.38	0.23	0.23	0.20	0.20
1200	0.92	0.74	0.53	0.25	0.20	0.20	0.29	0.29
2000	0.71	0.33	0.87	0.20	0.33	0.33	0.33	0.20
2800	0.24	0.46	0.71	0.24	0.30	0.24	0.30	0.24
3600	0.81	0.27	0.23	0.24	0.27	0.26	0.24	0.23
4400	0.23	0.80	0.63	0.23	0.28	0.23	0.23	0.23
5200	0.74	0.66	0.76	0.25	0.23	0.23	0.26	0.26
6000	0.60	0.69	0.57	0.27	0.27	0.24	0.26	0.22

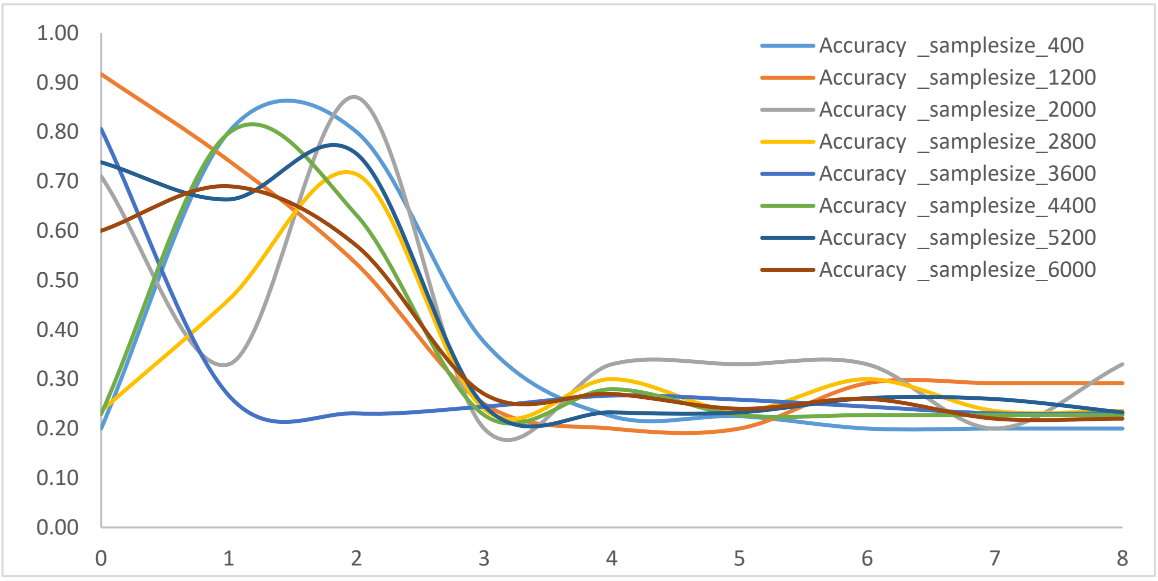


Fig.32 Accuracy rate of different sample size and number of trainable layers

As we can see from the figure above, same as VGG16, the first 3 conditions are the same, because the 4th and 5th trainable layers have no weight parameter. So, the first three conditions have high relatively accuracy. However, with more trainable layers, the accuracy is lower. After 6 trainable layers, the convergent accuracy is 0.25, which is equal to random guessing. This is probably because of overfitting with more weight parameters to be trained. More parameters make the optimizer harder to achieve a better solution other than a local one.

VGG19 has more parameters than VGG16 and need more input data for training, therefore, VGG19's performance of transfer learning is accordingly worse than VGG16. And comparing the results of Inception V3 and VGGNets, there is a very interesting trend. As the number of trainable parameters increases in Inception V3, its performance improves, while in VGGNet, the condition is just opposite with Inception V3. We think that is because compared with VGGNet, Inception V3 has much fewer parameters. The training data can be used to train more layers in Inception V3 than in VGGNets.

To be honest, the experiments for transfer learning are not complete enough because of the issue of time. We will conduct transfer learning to map identification and classification based on projection in the future. But we want to make it a good beginning work and meaningful try for transfer learning.

3.5 Projection classification

The testing accuracy results of MLPs are shown in Table.13 and Fig.33 below. From the results, we can find that both training accuracy and testing accuracy of the results are all very bad for MLPs with different number of hidden layers. The accuracies are all about 0.25, which is the probability for one class in the four. The predicted value is all for one class. MLPs are not powerful enough to tell the difference between distinct regions.

Table.13 Testing accuracy results of MLPs

# Hidden Layers	Training Acc	Testing Acc
2	0.25	0.24
3	0.25	0.26
4	0.25	0.25

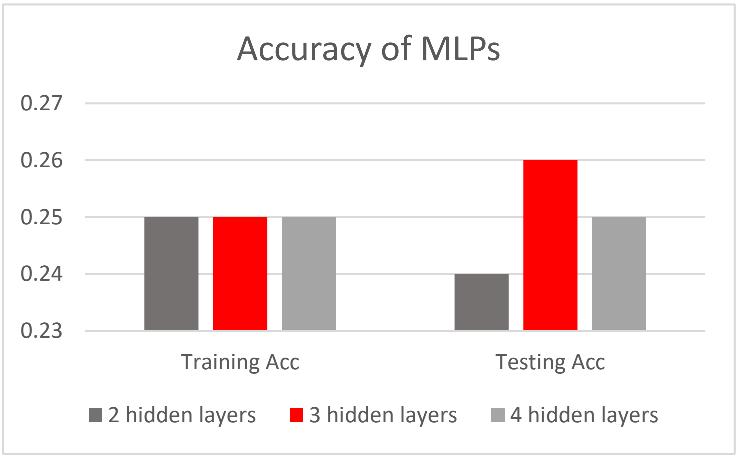


Fig.33 Testing accuracy results of MLPs

For SVMs, the testing results of four kinds of classification kernel were compared in this experiment. The results are shown in Table.14 and Fig.34 below. According the results, Linear kernel and Polynomial kernel SVMs significantly outperforms the other two. But the testing accuracies are also not good enough for use. They can classify the four classes with relatively satisfactory results, but the other two's results are very bad. That means that the inner function of region classification fits linear and polynomial kernels to some extent, but it cannot fit RBF and Sigmoid kernels. And we can also find that the training accuracy of RBF kernel is 1, but the testing accuracy is very poor. So, It implies that the RBF kernel overfits the inner function of region classification.

Table.14 Testing accuracy results of SVMs

Kernel Function	Training Acc	Testing Acc
Linear	1	0.715
RBF	1	0.34
Polynomial	1	0.715
sigmoid	0.27	0.19

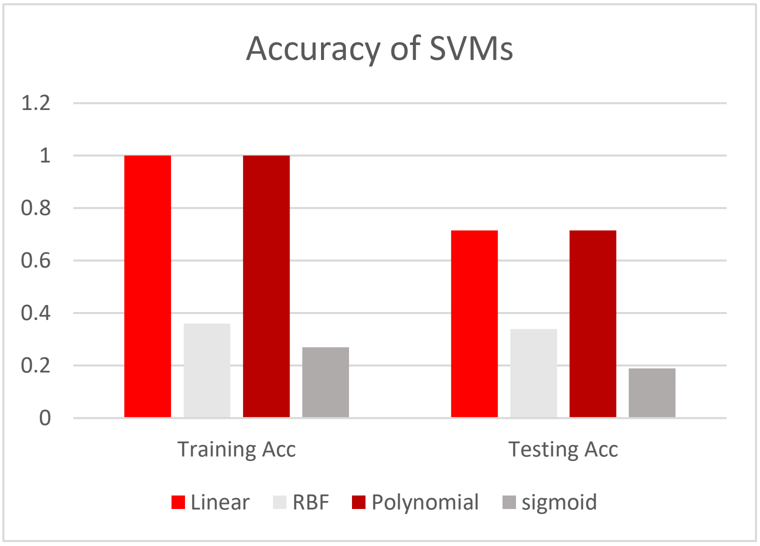


Fig.34 Testing accuracy results of SVMs

As for CNNs, the best results are from CNNs with 3 convolutional layers. The best training accuracy can reach 100%, while testing accuracy can only reach about 80%. The general results of CNNs for map identification is shown in Table.15 and Fig.35 below. The CNNs can also get results with high testing accuracy compared with SVMs and MLPs, but the results are not as good as in region classification.

Table.15 Testing accuracy results of CNNs

# Hidden Layers	Training Acc	Testing Acc
2	1	0.75
3	1	0.79
4	0.84	0.65

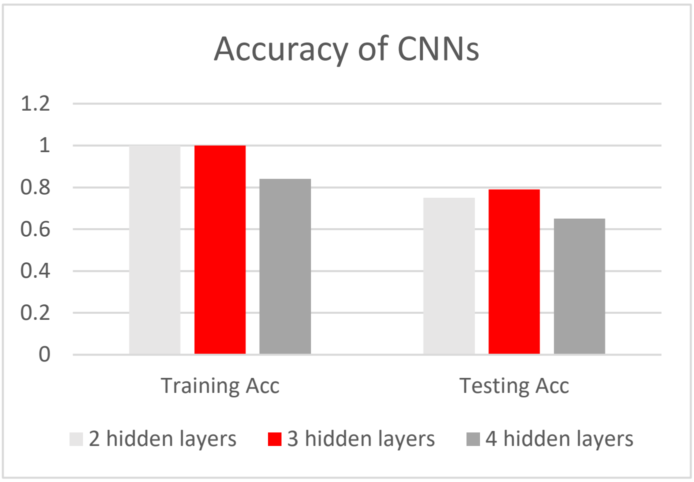


Fig.35 Testing accuracy results of CNNs

The testing accuracy of best result we got is 95%. The images and corresponding predicted label are shown as follows. Equirectangular projection, Mercator projection, Miller projection and Robinson projection are represented by 0, 1, 2 and 3 respectively. The red label of 3 is the wrong predicted label. It should be 0.

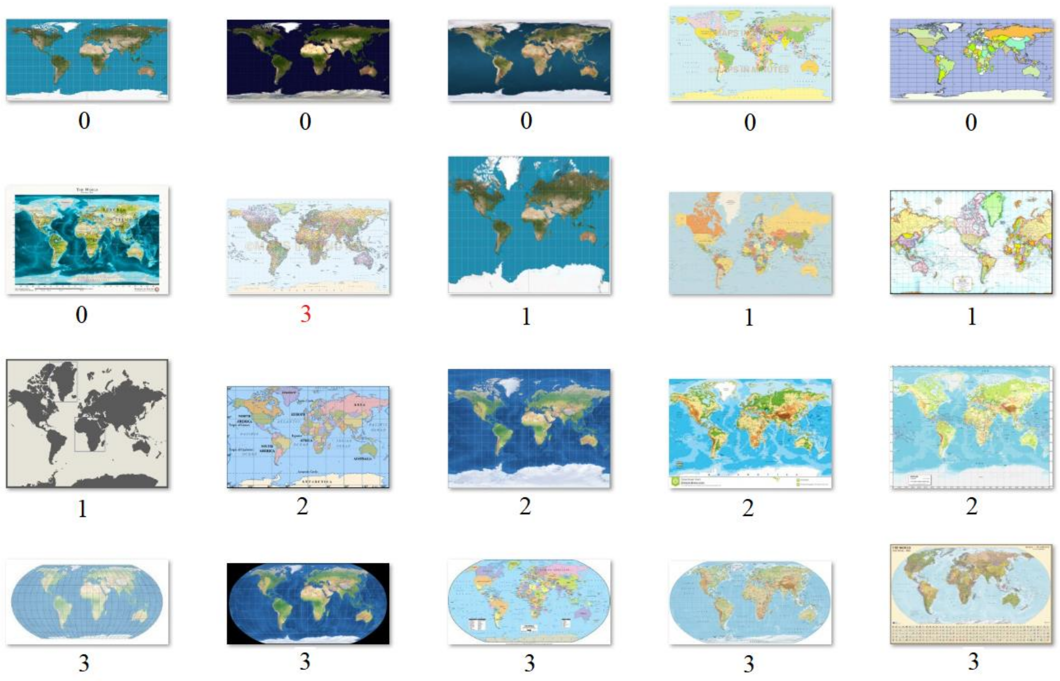


Fig.36 The images and corresponding predicted label

Comparing the experiment results of the three methods, we can find that SVMs and CNNs can also get relatively good classification results with some parameter settings but MLPs cannot get good results. The reason is that is the difference between map images of different projection is much smaller than the difference between map and non-map images and different map regions. Different projections of same region can be very similar for machine to classification. In this case, MLPs cannot tell the difference and get good classification results. CNNs and SVMs can get relatively good results but still not very good.

4. Summary

In summary, three machine learning methods are used to solve the map identification, region classification and projection classification problems in this study. Additionally, three CNN architectures are used in transfer learning for region classification. In general, CNNs and SVMs can always generate relatively good results and CNNs outperforms SVMs a little bit. For map identification problem, all of the three methods can get good results because of the large difference between map images and non-map images. For region classification and projection classification problems, SVMs and CNNs can also get good results but MLPs' performance is quite poor. And for projection classification, because of much smaller difference between different projections, even SVMs and CNNs cannot get a testing accuracy above 90%.

And even though the results of transfer learning are not very stable in different cases, we believe it can be a good solution to region classification. As mentioned, because of the time limitation, we only use transfer learning into region classification. In the future, we will also conduct transfer learning to other tasks, especially projection classification, because our current methods cannot get very high classification accuracy.

This project is the beginning work of AI application for cartography. There are lots of other works to be done here. For example, more parameter settings should be tried for systematically analysis of parameter influence, including the influence of batch size, input size of images. And the training time of different methods should also be examined, and high-performance computation techniques can be used to reduce the training time. All of these works mean to make the conclusion complete and more convincing.

Reference List

Ashok, B., & Aruna, P. (2016). Comparison of Feature selection methods for diagnosis of cervical cancer using SVM classifier. Int. J. Eng. Res. Appl, 6, 94-99.

Bertin, J., Berg, W. J., and Wainer, H. (1983). Semiology of graphics: diagrams, networks, maps (Vol. 1, No. 0). Madison: University of Wisconsin press. Card, M. (1999). Readings in information visualization: using vision to think. Morgan Kaufmann.

ESRI, U., and PaperdJuly, W. (1998). ESRI shapefile technical description. Comput. Stat, 16, 370-371.

Ferentinos, K. P. (2018). Deep learning models for plant disease detection and diagnosis. *Computers and Electronics in Agriculture*, *145*, 311-318.

Ghazi, M.M., Yanikoglu, B. and Aptoula, E., 2017. Plant identification using deep neural networks via optimization of transfer learning parameters. Neurocomputing, 235, pp.228-235.

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

Hughes, D., and Salathé, M. (2015). An open access repository of images on plant health to enable the development of mobile disease diagnostics. *arXiv preprint arXiv:1511.08060*.

Leborg, C. (2006). Visual grammar. Princeton Architectural Press.

Machin, M., Sanguesa, J. A., Garrido, P., and Martinez, F. J. (2018, April). On the use of artificial intelligence techniques in intelligent transportation systems. In *Wireless Communications and Networking Conference Workshops (WCNCW), 2018 IEEE* (pp. 332-337). IEEE.

Marcus, M., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank.

Mohanty, S. P., Hughes, D. P., and Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in plant science*, *7*, 1419.

Oh, S. K., Yoo, S. H., and Pedrycz, W. (2013). Design of face recognition algorithm using PCA-LDA combined for hybrid data pre-processing and polynomial-based RBF neural networks: Design and its application. *Expert Systems with Applications*, *40*(5), 1451-1466.

Oleszkiewicz, W., Cichosz, P., Jagodziński, D., Matysiewicz, M., Neumann, Ł., Nowak, R. M., and Okuniewski, R. (2016, September). Application of SVM classifier in thermographic image classification for early detection of breast cancer. In *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2016* (Vol. 10031, p. 100312T). International Society for Optics and Photonics.

Sengorur, B., Koklu, R., and Ates, A. (2015). Water quality assessment using artificial intelligence techniques: SOM and ANN—A case study of Melen River Turkey. *Water Quality, Exposure and Health*, *7*(4), 469-490.

Seo, Y., Kim, S., Kisi, O., and Singh, V. P. (2015). Daily water level forecasting using wavelet decomposition and artificial intelligence techniques. *Journal of Hydrology*, *520*, 224-243.

Sibanda, W., and Pretorius, P. (2011). Novel application of Multi-Layer Perceptrons (MLP) neural networks to model HIV in South Africa using Seroprevalence data from antenatal clinics. *International Journal of Computer Applications*, *35*(5).

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A. and Chen,Y. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*(7676), 354.

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

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

White, T. (2017). Symbolization and the Visual Variables. The Geographic Information Science and Technology Body of Knowledge (2nd Quarter 2017 Edition), John P. Wilson (ed.).

Zhu, X. X., Tuia, D., Mou, L., Xia, G. S., Zhang, L., Xu, F., and Fraundorfer, F. (2017). Deep learning in remote sensing: a comprehensive review and list of resources. *IEEE Geoscience and Remote Sensing Magazine*, *5*(4), 8-36.