THE UNIVERSITY OF MELBOURNE

Cloud Classification through Deep Learning

COMP90055 Computing Project (25 credits)

Type of Project: Software Development Project

Supervisor: Prof. Richard Sinnott

Student: Yixiang Yang 791962

Summer Semester, 2019

Abstract

Deep learning has grown huge in recent years; it allows computational models that include different processing layers to learn and solve real problems like a human. This type of idea has already used in various areas, such as image recognition, natural language processing, recommendation system and financial fraud detection.

In this project, deep learning based methods will be used for establishing a reasonable model to detection the types of clouds in the images. There will be six trained object detection models, and after comparing these, we finally select SSD_Resnet50_v1_FPN as the final model mainly because of the high accuracy and less training steps. There will be more discussion about the result and possible optimization methods about the final model.

To demonstrate the prediction of the cloud's types, a website will be deployed on the Google cloud. The website will show the location box and what kind of clouds in an uploaded picture.

Keywords: Deep Learning, Clouds, TensorFlow, Object Detection API, SSD, Resnet50

Acknowledgement

Thanks to my supervisor Professor Richard Sinnott for trusting and giving me an opportunity to access this challenging project. During the project period, he always gives wonderful instruction for the problems that I have meet in the project. The questions usually answered in a few hours which helps me a lot to implement the software on time.

Thanks to the TensorFlow community which provides wonderful API, solutions for the issues and tutorials materials. Besides, I would like to thank the Google company open source TensorFlow which allows all developer to contribute.

Thanks to my family and girlfriend that support me when I felt disappointed about the project progress. Without their understanding, I would not finish the project on time.

Finally, thanks to the ImageNet community, Advanced Digital Sciences Centre (ADSC) in Singapore and other owner of the images used in the dataset and report.

Table of Content

1. Introduction

1.1 Deep Learning

As one of the latest technologies, deep learning has applied in many different fields, such as image recognition, automatic speech recognition, drug discovery and financial fraud detection. This deep learning term was first introduced to the machine learning community by Rina Dechter in 1986 [1], and after a few years, the concept of the artificial neural network was put forward by Igor Aizenberg in 2000 [2]. However, it has met some computational issues that the current hardware is not able to finish this job in a reasonable time. While it still provides the chance of extending the boundary of computer science. During the last two decades, the capacity of the computational ability of the computer has got an explosive breakthrough. Nowadays, even the mobile device could run deep learning based software or component. For example, the GPU Turbo technology which deployed in the Hua Wei's Kirin 980 processor. It uses TensorFlow to build a new model to learn and find the best balance points between CPU, GPU, RAM and other components. This technology brings about 10 per cent of GPU performance improvement, 30 per cent of power consumption decrease and almost full FPS and FPS stability in large mobile games.

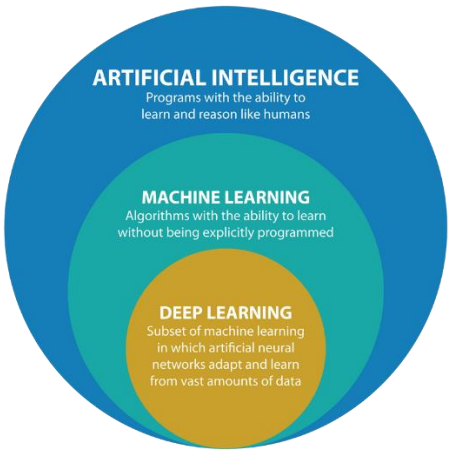


Figure 1 Artificial Intelligence Structure

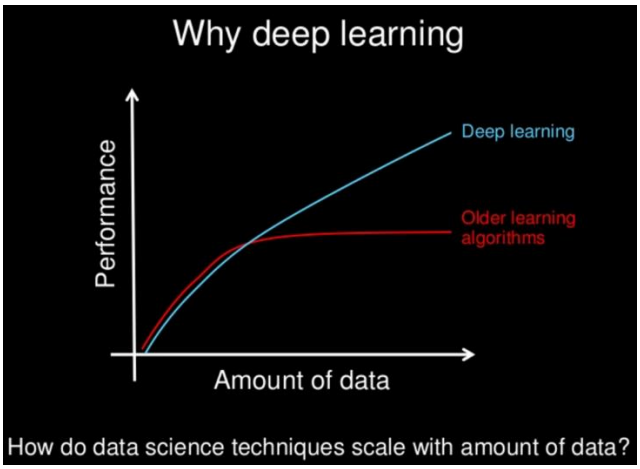


Figure 2 Performance with the amount of data

As Figure 1 shows, deep learning methods is a part of the machine learning area, most of the modern models are based on Artificial Neural Networks (ANN). The traditional machine learning methods cannot handle more complex problems very well, especially when the input data is various. Especially, for the image recognition problems, it is hard to extract useful

features and get higher accuracy. While the neural network could deal with this problem better than the traditional machine learning models. The model allows the system to learn what kind of features are more important by itself rather than manual selection. Because human being also does not know which feature could arrive a better result. With deep learning models, it is much more convenient and faster to extract useful features compared with the traditional machine learning methods.

As Figure 2 shows, deep learning methods achieve much better performance than the older machine learning algorithm and especially when the increase of the input data size. Deep learning brings remarkable progress in terms of achieving artificial intelligence.

1.2 Image Classification

Image classification is one of the popular topics in deep learning. It is a very complex process that might be influenced by various factors [3]. In brief, the process will take the images as input data, and the output is a class label which indicates the image belongs. The whole process is showing in Figure 3. Normally, the classification algorithm will give one specific class which means the image is a laptop,

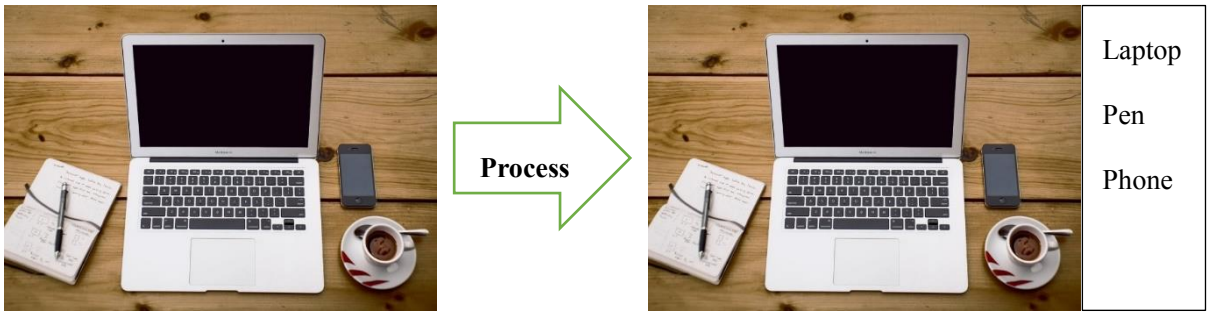


Figure 3 Image classification process

1.3 Image Classification with Localization

Image classification with localization is an upgrade version of the result. It is also taking an image as input and the difference is outputting the class label around with a bounding box to locate the object in the image. Figure 4 shows the output difference in this process.

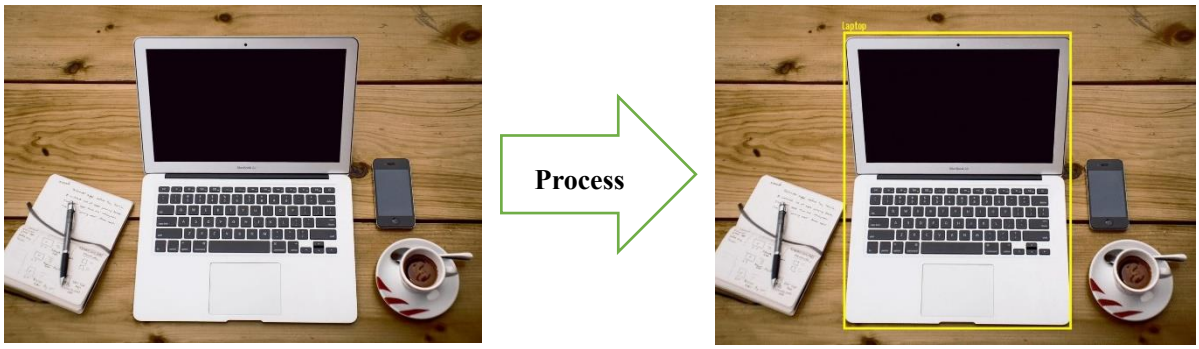


Figure 4 Image classification with localization process

1.4 Object Detection

Object detection can be simply regarded as the combination of image classification and localization. The image localization process will be applied to each detected object and draw the bounding boxes. The detected object could belong to different classes which means it allows a costumed object in different projects. Currently, object detection has widely used in face detection, skin cancer detection [4] and other fields. The typical result of object detection shows in Figure 5; multiple bounding boxes have been drawn in the output image.

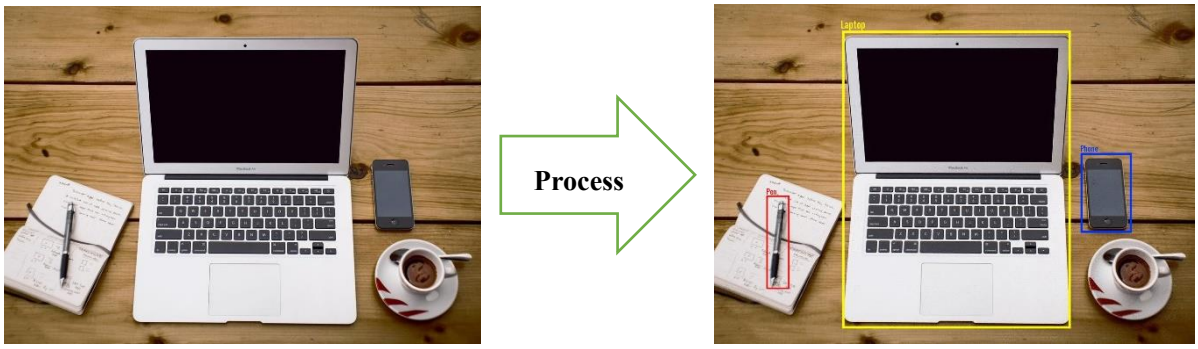


Figure 5 Object detection process

Considering the demonstration of the output result and the structure of the clouds, we finally choose object detection method as the resolution of this project. Because different clouds usually appear in the sky at the same time, it is good to show the location of all detected clouds. A website will receive an uploaded image and send to the server to process, then there will several labels listed on the right side of the image. The user can click the label to see what kind of clouds in the image and the corresponding boxes which show the location of the cloud. Besides, we will discuss several object detection frameworks in this project, such as Faster RCNN, YOLOv3, SSD, RetinaNet and others. These frameworks could be categorized into one stage process and two stages process, with different feature extract methods, it could have

different performance on the same dataset [5]. The details of these frameworks used in this project will be introduced in Section 3.2. Apart from this, the evaluation metrics are mainly focused on the accuracy and training steps; it will be discussed in Section 5.

1.5 TensorFlow

To implement a stable and scalable system, a suitable deep learning framework is necessary. Currently, there are serval frameworks, such as TensorFlow, Caffe2, PyTorch and others [6]. Some of these are open source platform and delivered by different companies and academic organizations. Compared with scalability, portability and recourse, TensorFlow would be the most reasonable for this project. Firstly, it is very easy to install on both Windows or Linux system. Secondly, it has two available versions, one is the GPU version, the other is CPU. Thus, if you do not have an NVIDIA graph card with CUDA compute capability 3.5 or higher, the TensorFlow CPU version is also enough to finish the whole development cycle. Thirdly, it is driven by the Google company which is the owner of the Android system. Thus, it is to transform a model from a desktop platform to a mobile device because 70 per cent of mobiles is running the Android system. Last but not least, there are thousands of contributors to the community which provides lost of solutions for the possible issues that you might meet in the developing progress.

2. Dataset

For deep learning based object detection in images, the quality of the training data is very important. A reasonable dataset should include the images with suitable size, correct class label and the correspond bounding box. Since there is no existing cloud dataset, I decided to collect the images from the Internet and make a new dataset for this project.

According to the UCAR Centre for Science Education (UCAR SciEd), there are mainly 16 different types of clouds. Considered some clouds are very hard to collect and the main focus is not perfect detect all types of clouds, so we narrow the categories to 10. There will be 10 different clouds in the dataset, cirrocumulus, cirrostratus, cirrus, cumulonimbus, altocumulus, altostratus, cumulus, nimbostratus, stratocumulus and lenticularis respectively. These clouds could divide into four groups, high-level clouds, middle-level clouds, low-level clouds and

special. Clouds are categorised by their shapes and their height in the sky [7]. The high-level clouds mean that the cloud bases could be observed from 4,500m to 12,000m height in the sky. The height of middle level and low-level are 2000m to 6,000m and 200m to 1,500m respectively. The special clouds are a little bit different, it can easily recognise from the shape and it could be observed from 200m to 9,000m. The shape of different clouds shows in Figure 6.

Figure 6 Different types of clouds

The following sections will illustrate the process of creating the dataset and how we transform the original images to the input data of the TensorFlow. The main effort is choosing the high-quality images and labelling the different clouds in every image.

2.1 Dataset Sources

There are five dataset sources, the Singapore whole sky imaging dataset, ImageNet, Google, Bing and Flicker. The Singapore whole sky imaging dataset called SWIMSEG, it contains 1013 images which captured in Singapore using WAHRSIS, a calibrated ground-based whole sky imager. This academic community collects these images from October 2013 to July 2015, each of images covers 60 to 70 per cent of the sky with 600x600 pixels and has labelled by experts from Singapore meteorological services [8]. According to the label, we need to draw the corresponding boxes on each of the images.

The second source is from ImageNet [9], an image database organized according to the specific hierarchy. It contains 14,197,122 images with 21841 subnets which represents a large variety of objects in the real world. For the cloud subnet, it has 1,119 images and some of these already labelled with bounding boxes. However, the quality is not very good, and some images were broken and labelled with the wrong label or incorrect bounding boxes. Thus, we review every image that downloads from the ImageNet and make sure all images labelled with correct information.

The last three sources are Google, Bing and Flicker, basically by using their interface, we can use the python script to craw images automatically. The download images must have relevant tags, title or description which belong to one of the detected cloud types. For a specific cloud, the script will craw 1,000 images on Google, Bing and Flicker separately. During each collecting iteration, the program will remove the duplicated image address. The initial dataset contains 9,406 images, and we will process these images before training a model. All scripts are uploaded to the GitHub.

2.2 Dataset Pre-processing

The pre-process includes rename images, resize images, remove duplicated images, remove overlooking images which means the photo might take from the airplane, draw the corresponding boxes and transform the dataset to TF record file. We also use the python script

to modify the original image name to a more meaningful one, then use OpenCV library to resize the images and ensure each of them under 230 kb size. After that we remove the duplicated images to improve the quality of the whole dataset. Following this, drawing the bounding boxes is a meticulous process that takes a long time, but it is a necessary part of training an object detection classifier. Figure 7 shows an example of the bounding boxes in an image. Overall, it took us about 10 days to label all the images. There will be two formats of
files after these processes, .JPG image files and the corresponding XML files. The XML files store the location information of the bounding boxes and other supplementary information. Then, we divide the dataset into two groups, test and train. Train group contains 80 per cent of the whole image dataset, and its use to train a new model. The rest of the images belongs to test group which used for evaluating the performance of the trained model. The final process is transforming the whole dataset into TF record file which is the input data format of the TensorFlow object detection API. All scripts are uploaded to the GitHub.

Figure 7 Demo for the bounding box

2.3 Dataset Summary

After the image pre-processing, there are 7,014 images in the dataset, but we use the number of bounding boxes to represent the number of images. Because one image could contain various type of clouds, so it is better to describe the dataset size with different bounding boxes. Figure 8 lists the number of different clouds in the dataset. As it shows in Figure 8, most of the clouds

have more than 500 bounding boxes, but the lenticularis cloud only has 128 boxes. This is because this kind of cloud is really few to see, it only appears near the mountains with special airflow. Thus, the number of this cloud is much lower than the other clouds.

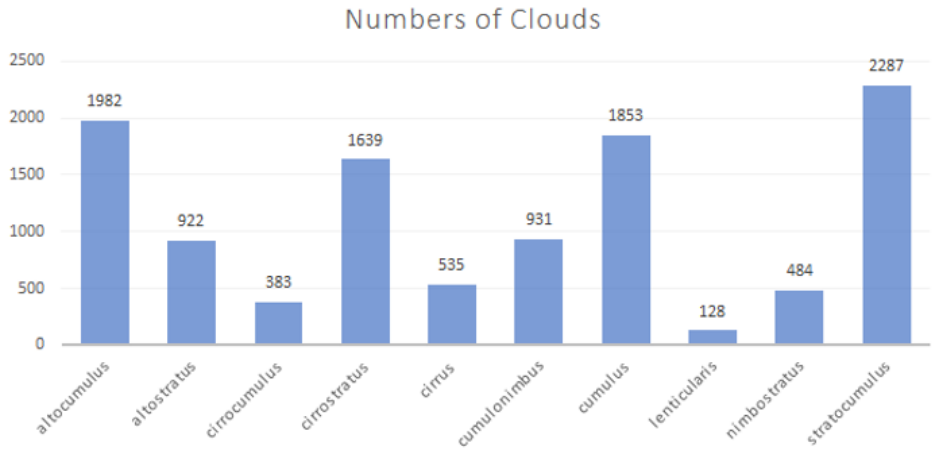


Figure 8 The Number of bounding boxes for each type of clouds

3. Training

This part demonstrates what kind of models used in the project and explain the decision of the final chosen model. Section 3.1 will introduce the related knowledge and Section 3.2 lists the different object detection frameworks. Then Section 3.3 shows the results of each model, the evaluation metric and the final model.

3.1 Related Work

In this part, we will introduce the concept of transfer learning and how to avoid issues brought by the overfitting and underfitting.

3.1.1 Transfer Learning

Transfer learning is one of the effective methods that use pre-trained models for training a new mode with less time and less dataset size. This technology has been successfully applied in many real-world applications [10]. Raina has proposed to use it to learn text data across domains [11]. Wu and Dietterich plan to use both inadequate source dataset and low-quality source dataset to solve the image classification problems [12]. According to the research of Torrey and Shavlik, we can get benefits from transfer learning in the following three points, a good start, higher slope and higher asymptote. Figure 9 shows these three possible advantages during

the training process. When we meet the real world problems, inadequate dataset is the most common issues, such as ore classification and pathological tissue detection. It is difficult to collect enough dataset for training higher performance models. Therefore, transfer learning becomes a trend in deep learning fields gradually in recent years. In this project, only six types of clouds have more than 1,000 bounding boxes, the numbers of other four genus clouds are under 500 bounding boxes. Thus, transfer learning based methods are the most reasonable solution to this problem.

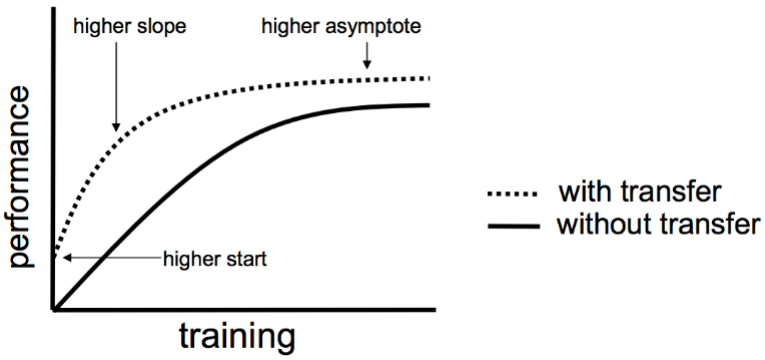


Figure 9 Three possible advantage of transfer learning

Besides, TensorFlow object detection library supports various pre-trained models with COCO dataset which contains a wide range of object. It helps us to analysis which model has the best performance on the cloud dataset.

3.1.2 Overfitting and Underfitting

Overfitting and underfitting are the common challenges when doing machine learning process [13]. Figure 10 is a diagram which shows the loss initially decrease on both training and validation data, but after a bit, the training loss value would continue to go down while the validation loss would start to increase. This is called overfitting. The model became too specialized on solving the training data and starts to perform worse when validated on the test data. In other words, the model memorizes the answers and parameters on the training dataset and does not generalize to the test dataset. The opposite situation is called underfitting where the model does have enough variables to solve the training dataset. Compared with overfitting, a more advanced model with more parameters and variables would perform better on the same dataset. Figure 11 shows the possible results of these two issues.

Luckily, TensorFlow supports a wonderful visualization tool called Tensorboard. We can use

this tool to observe the real-time training process and the trend of the loss function. If overfitting or underfitting was detected in the training phase, we could try to solve or minimize the influence. Such as reducing the training steps, change different models or apply regularization techniques.

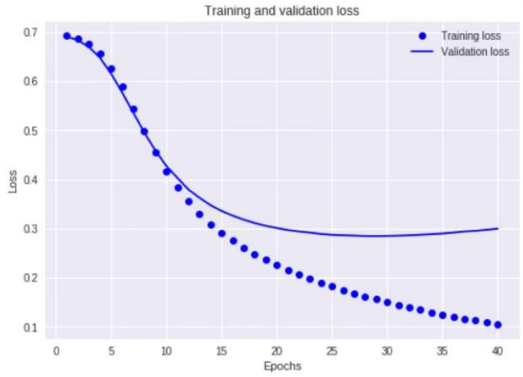


Figure 10 Overfitting problem

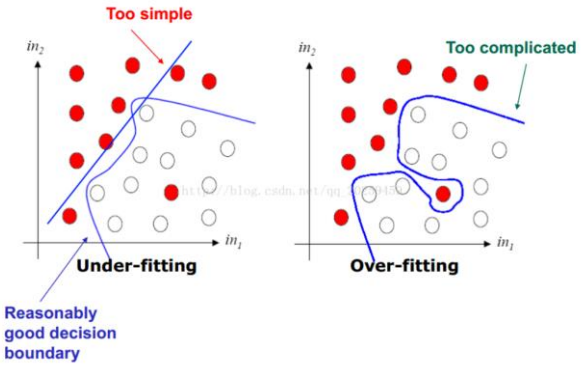


Figure 11 Results of overfitting and underfitting

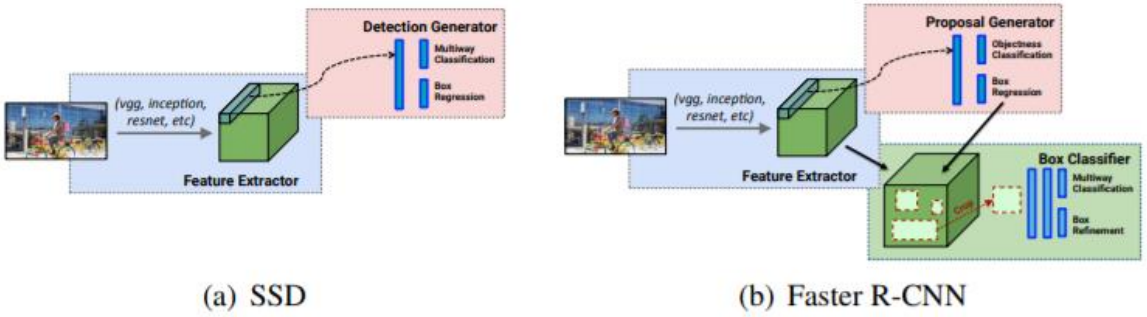
3.2 Object Detection Models

Since Alex Krizhevsky proposed AlexNet and win the champing of ILSVRC 2012 with significant lead [14]. There are a lot of object detection resolutions has been proposed [15], each of them contains two parts, the meta architecture and feature extractors methods. Taking Faster RCNN Inception V2 as an example, the meta architecture is Faster RCNN and it combined with Inception V2 feature extractors method.

The following parts will introduce different meta-architectures and feature extractors methods used in this project.

3.2.1 Meta Architecture

Currently, there are various meta architecture has been proposed and implemented, such as YOLO, SSD, Faster RCNN, Mask RCNN. In this project, we decided to choose SSD and Faster RCNN as the meta-architecture of the model. The main reason is that it could successfully combined with other feature extractors techniques. Besides, a lot of pre-trained models has already released by the TensorFlow object detection API, which based on the Faster RCNN and SSD. Figure 12 is a high-level diagram of the object detection meta-architectures.



(a) SSD (b) Faster R-CNN

Figure 12 A high-level diagrams of the meta-architectures

Faster RCNN

As it shows in Figure 12 (b), Faster RCNN has two stages during the detection process. The first stage is called RPN (region proposal network), images could be processed by a specific feature extractor. These features from some selected intermediate layer level will be used to predict nonrelevant class box proposal. In the second stage, the box proposals are used to predict a class label and upgrade the performance of each proposal [15]. The details about RPN and implantation could be found in [16].

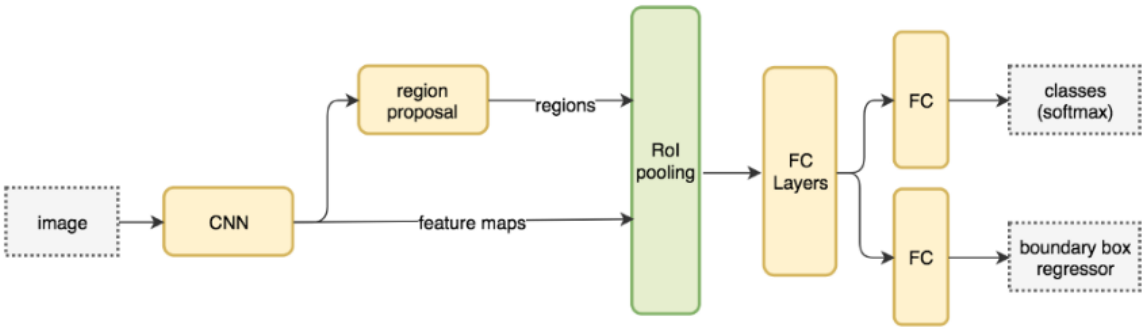


Figure 13 Faster RCNN flow

SSD (Single show detector)

SSD has been describing in the [17], it removes the RPN process used in the Faster RCNN. However, it has some improvement in a model the space of possible box shapes [17]. The whole workflow shows in Figure 14, it uses the VGG19 as the fundamental feature extractor (similar to the CNN in Faster RCNN). Then, custom convolution layers (the blue parts in the figure) will be added after the feature extractor process and apply convolution filters (the green pars in the figure) to make predictions [18].

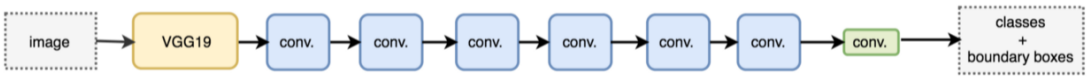


Figure 14 Single shot prediction for both classification and location

3.2.2 Feature Extractor

In this project, we will discuss four feature extractors. The following sections will briefly introduce the concept of these methods.

Inception

Inception network is one of the most important milestones in the development of image classifiers. Currently, there are four versions, Inception v1, Inception v2, Inception v3, Inception v4 and Inception with ResNet. In this project, we use the Inception v2, an upgrade version of Inception v1 [19]. This new model proposes a solution that how to scale up the size of neural nets without extra computational cost [21]. Compared with Inception v1, in order to improve computing speed, the new version factorizes 5*5 convolution to two 3*3 convolution operation. Besides, the model transforms the convolutions of size n*n to a combination of 1*n and n*1 convolutions [19]. For example, transfer a 5*5 convolution to process a 1*3 convolution first, then performing a 3*1 convolution on its output. Moreover, the transformations on different layers can be stacked in parallel after reducing the input convolution [20]. This details about how to implement this structure and the relevant analysis could be found in [20].

ResNet

If the Inception network is willing to go wider, the ResNet is about being deeper. ResNet called Residual Neural Network, as one of the breakthroughs in the development of the deep network, it has solved the notorious vanishing gradient problem which means it is hard to back-propagated to the earlier layers, because of the repeated multiplication iteration, the gradient might become extremely small. As a result, the model cannot learn the small size of the area [22]. The key part of ResNet is applying a method called identity shortcut connection which

means the training process might skip one or more layers [23]. Through this method, it is possible to establish enough layers to solve complex problems. The details about how to implement this structure and the relevant analysis could be found in [23].

Mobilenet

Mobilenet is a small but powerful convolutional network, it was first introduced by Howard in 2017. This network uses depthwise separable convolutions that factorize a standard convolution into two parts, depthwise convolution and pointwise convolution [24]. Compared to the normal convolution, this method reduces 8 times training operations. As a result, it becomes one of the most effective models when applied to various tasks. The details about how to implement this structure and the relevant analysis could be found in [24].

FPN (Feature Pyramid Network)

FPN (Feature Pyramid Network) is designed for a pyramid concept feature extractor composed by a top-down architecture with lateral connection. This architecture is developed for building the high-level semantic feature maps at all scales [25]. There have two pathway, bottom-up pathway and top-down pathway. The former one has many convolution modules that each of them has many convolution layers as well. The output of these modules will be labelled and used in the top-down pathway. The top-down pathway will output final predictions and after serval iterations, it will have d output channels while d is feature dimension in all the feature maps. Besides, the FPN network can be combined with Fast or Faster RCNN, it also improves the benchmark of these two models on the COCO dataset. The details about how to implement this structure and the relevant analysis could be found in [25].

3.3 Compare Different Detection Models

In this section, we will combine the meta-architecture with different feature extractors mentioned before. The following are the six object detection models used in the project and the results will be discussed in this part.

- Faster_rcnn_inception_v2
- Faster_rcnn_resnet50
- SSD_inception_v2

- SSD_mobilenet_v2
- SSD_mobilenet_v1_FPN
- SSD_resnet50_v1_FPN

3.3.1 Training Configuration

This project will implement a web application to demonstrate the cloud classification, there are two important indicators, accuracy and detection time. As one of the concerns, it is easy to understand high accuracy means that the website could detect different types of clouds correctly since some typical cloud has a function of predicting the upcoming weather in a few hours which could impact the decision of going outside or find shelters. The importance of detection time is that we might create an off-line mobile application on Android or IOS platform which only supports limited computational capability. Moreover, a long detection time will give a bad user experience and the target might think the application is not functional well. Thus, it is a trade-off decision between accuracy and time.

The configuration files of each model are listed on the TensorFlow detection model zoo website which supports a wide range of pre-trained model [26]. We use these models as the beginning of the transfer learning and set all parameters as the same as the default figuration files.

3.3.2 Evaluation Metric

After the retraining process, we start to evaluation these new models with PASCAL VOC detection metrics and the average detection time. This former metric includes three values, mAP (the mean average precision), AP (average precision) of all class and IoU (intersection over union). mAP is used to evaluate the overall performance of the new models and AP will be calculated for every class label. The definition of IoU shows on Figure 15, it measures the size of overlap area between two regions which means how good is our prediction in the object detector with the ground truth [27]. Normally, if the value of IoU larger than 0.5 which means the intersection over union area lager than a half, the prediction could be regarded as an acceptable prediction. In this project, we will calculate the total evaluation time for the test dataset and use it to measure the average detection time for the per image. Through these metrics, it is easy to understand which model performs better on the whole dataset and which

model might have a higher accuracy on a specific type of clouds.

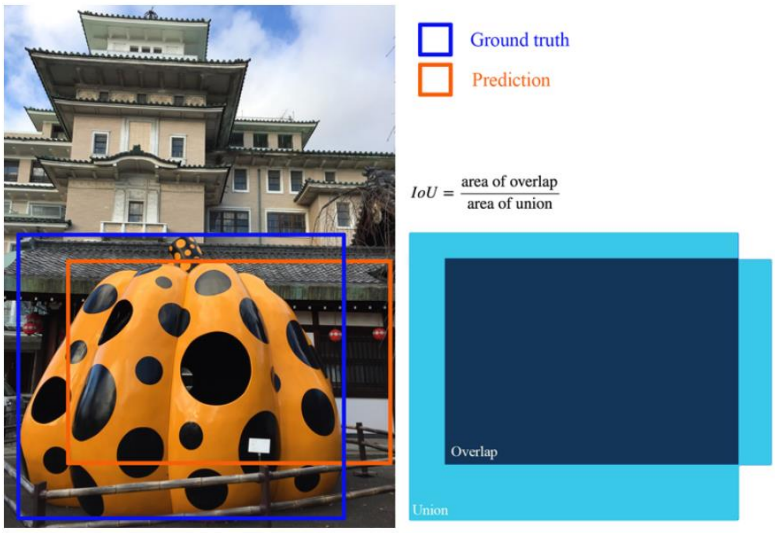


Figure 15 The definition of IoU

3.3.3 Result of Models

Figure 16 shows the results of mAP@IoU and average detection time for the six models. It is clear to see the Faster RCNN Resnet50 has the highest mAP@IoU with 0.53. Besides, most of the models with SSD meta-architecture have less detection time than the models with Faster RCNN meta-architecture. However, Faster RCNN ResNet50 models takes 1.38 second on detection process, taking the transferring latency time into consideration, it is not a good choice. Comparing to Faster RCNN ResNet50, Faster RCNN Inception v2 and SSD Resnet50 v1 FPN achieve a similar accuracy with a much lower detection time. These two models are more suitable than Faster RCNN ResNet50. Besides, since web application is our demonstration approach which means we could establish a cloud server with high performance computational capability. Thus, we tend to choose the model with higher accuracy. Overall, the SSD ResNet50 v1 FPN is the best model for this project.

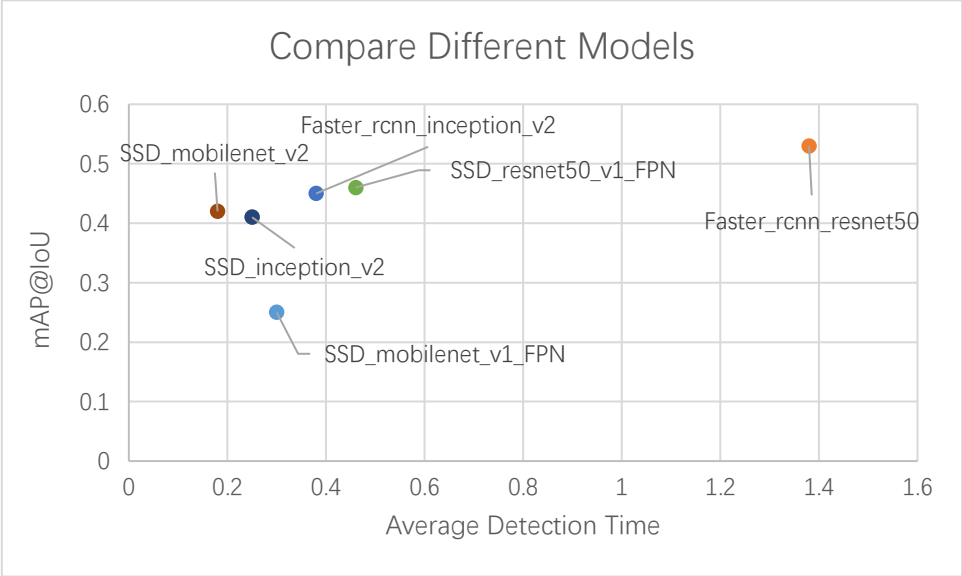


Figure 16 Compare the results of different with mAP@IoU

4 Optimization

As the result mentioned in Section 3.3.3, only one model has the value of IoU larger than 0.5 and the final choose we. However, we also mentioned in Section 3.3.2, the IoU value of an acceptable prediction should above 0.5. Thus, we still need to optimize the current model. Since we use transfer learning strategy to train the new model, it is hard to modify the actual structure of some convolution layers. Therefore, the most realistic method is to tune the hyperparameters in the configuration file which could influence the performance of a model. The parameters tuned in this project are training step, batch size and learning rate.

4.1 Batch Size

Batch size is a hyperparameter which defines the number of samples to propagate through before updating the internal model parameters [28]. It could influence the requirement of the system recourse and the time for each training step. If the batch size is too big, it requires huge recourse, especially the memory. In this project, we can only increase the batch size to 64, it consumes 4 Tesla P100 graph cards and the total memory is 48 gigabytes. If the batch size is too small, we might have a larger probability to meet underfitting problems which means the loss function would not converge in limited steps.

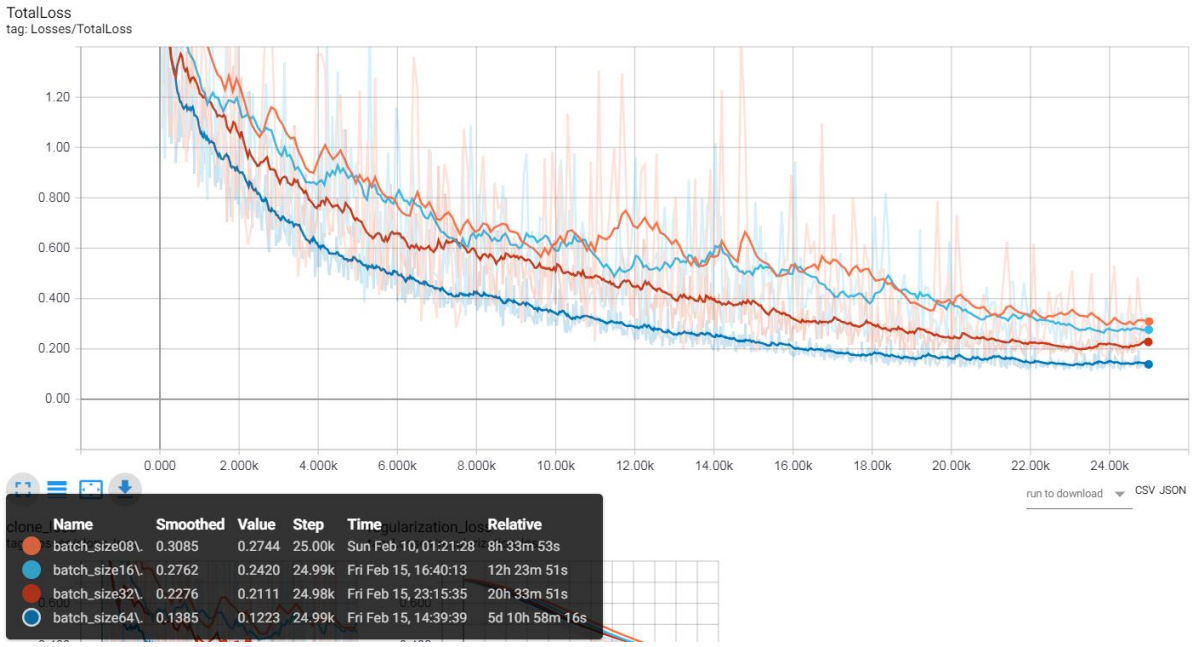


Figure 17 Loss curve with different batch size

There is a mistake about the training time of 64 batch size in Figure 17, the training process takes 36 hours. Because we have trained the model for two times, the first time we trained for 12 hours with almost 8,000 steps and the second time we finished the default training steps with 2,500 steps. However, there are three and a half days interval between two training times and the Tensorboard show the whole period based on the timestamp. Thus, the training time of 64 batch size is about 36 hours. It is clearly to observe in Figure 17, if the batch size has 1 increment, the training time will increase 2 hours. All loss curves are not converging.

In addition, as it shows in Figure 17, the loss curve of batch_size08 and batch_size16 have a significant fluctuation during the training process. With the increasing of the batch size,

the loss curve drops gradually with a stable rate. Thus, a suitable higher batch size could stabilize the performance of the mode.

Figure 18 shows the performance with different batch size, it is clear to see when the batch size increases from 8 to 16, the IoU grows from 0.15 to 0.44. When the batch size change from 16 to 32 and 64, they got a similar IoU but a more stable mode since we discuss in the last paragraph.

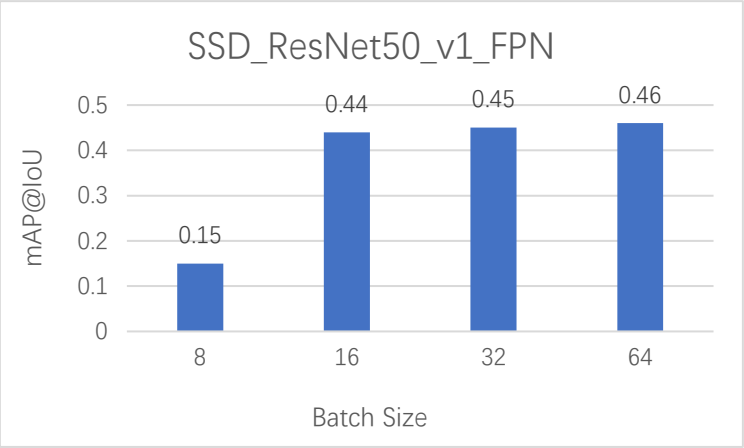


Figure 18 The performance of SSD_Resnet50_v1_FPN with different batch size

4.2 Learning Rate

Learning rate is another important hyperparameter for training deep neural network, it controls how much we are adjusting the weights of the network with respect the loss gradient [29]. As it shows in Figure 19, if the learning is too low, the process of gradient descent can be very slow. While if the learning is set too high, gradient descent could overshoot the minimum and it may fail to converge, or even diverge.

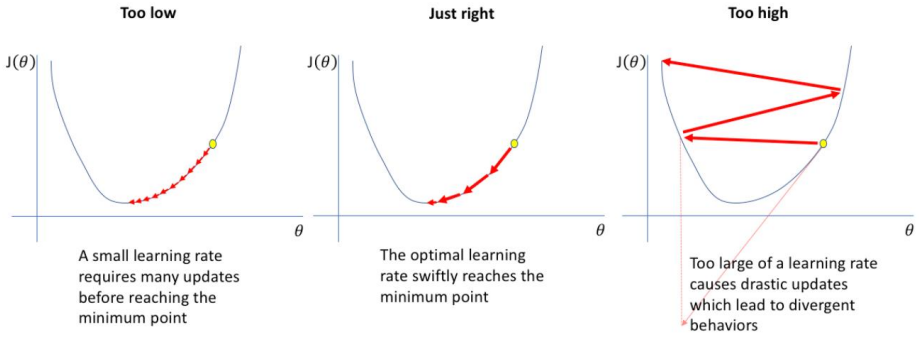


Figure 19 Performance of different learning rate

Figure 20 shows the downtrend of the loss curate with different learning rate, it seems like our model does not impact by the learning rate too much. Figure 21 shows the IoU value reach a peak at 0.46 when the learning rate equal to 4.00E-04.

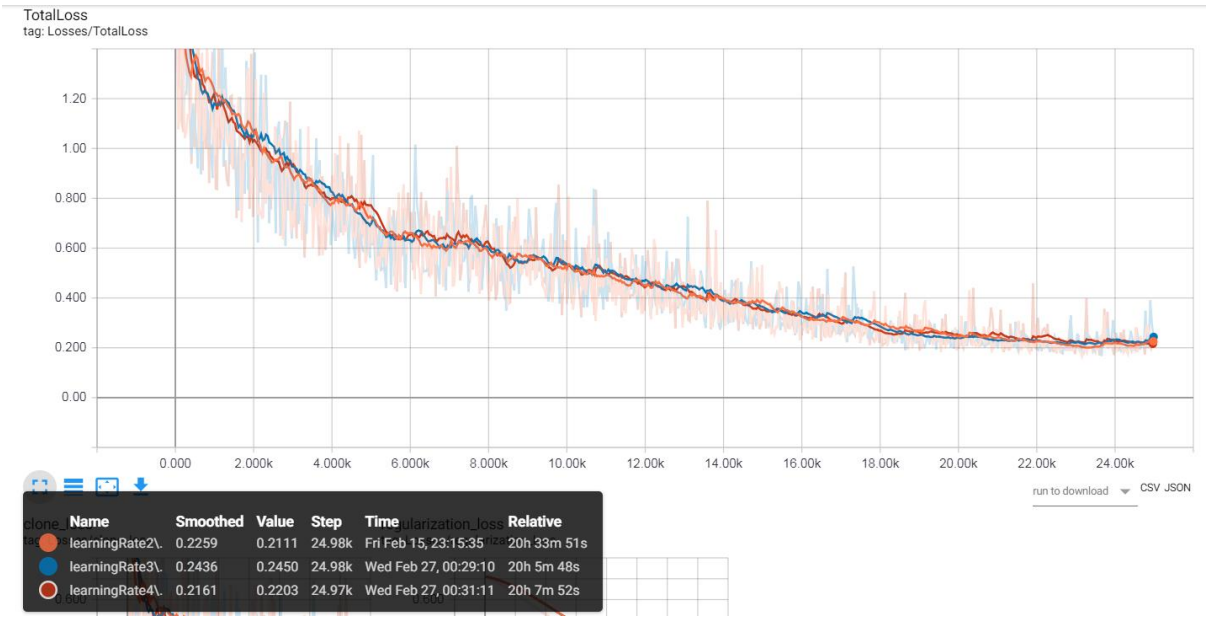


Figure 20 Loss curve with different learning rate

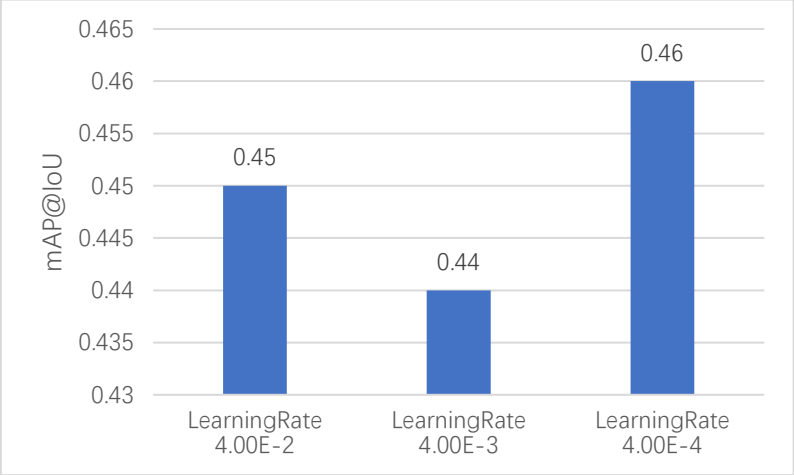


Figure 21 The performance of SSD_Resnet50_v1_FPN with different learning rate

5 Results and Analysis

After the optimization, the overall mAP@IoU is still not higher enough, but the result is acceptable in this project. In the following sections, we will release the performance of the model on different type of clouds and analysis the result, then discuss the possible reasons that influence the performance of the models.

5.1 Predication of Different Clouds

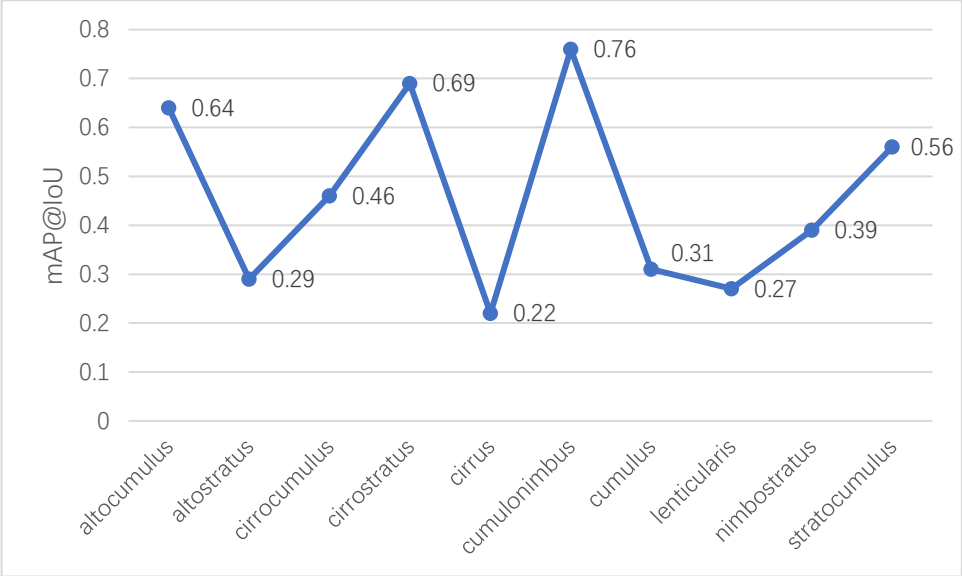


Figure 22 The IoU value of each type of clouds

As the mAP@IoU shows in Figure 22, the accuracy of altocumulus cloud, cirrostratus cloud and cumulonimbus cloud above 60 per cent. The cumulonimbus cloud reaches the highest accuracy with 76 per cent. However, there still have 6 types of clouds below 50 per cent. After analysis prediction of the test dataset, we found it always have more than one predicted bounding boxes on a single image. Some of the bounding boxes are very small but still detect the correct cloud. This situation will influence the IoU value since it is the average IoU of each bounding box. We can see from Figure 23, there are more than one bounding boxes in the images, the red colour box is the collect bounding box and others are the prediction result. All the predicted bounding boxes shows the correct type of cloud, but the area of some boxes is small and could be only 20 per cent of the correct one. As the result, the IoU value of these bounding boxes is very low and it will impact the overall IoU value of this image. Thus, for some particular clouds, the IoU value is disappointingly lower than the others. Such as altostratus, one of the important features about this cloud is it always cover the entire sky, the model may just detect some part of the cloud. On the contrary, cirrus cloud is just a wispy white streamer and if it covers the whole sky, it will be called cirrostratus clouds. Thus, if cirrus cloud appears a lot on an image and just fill up the image frame size, it might be classified as cirrostratus cloud.

Figure 23 The prediction of image with low IoU

5.2 Possible Reasons

The previous parts discuss the reason for the lower value of IoU. The following parts will illustrate the possible reasons that might influence the performance about the whole dataset.

Noising Image

Noising image means that some object with a cloud-like shape. It might be a light with cloud shape, hap or some artworks. Figures 24 shows some objects that extremely looks like cloud.

Figure 24 Noise images

Our model predicts a wrong cloud on the left and the middle image, it is because the shape, colour and structure are extremely like cloud. On the other word, these items are imitating the

nature of the clouds to manufacture. Thus, it is hard to distinguish these objects if we do not collect enough noise images. Due to the limit of the time, we do not have time to search this cloud-like images and it is a part of the future work.

Human Mistake

Human mistake happens when we manually label the type of cloud in an image. Although we have learned a lot of knowledge about how to distinguish the different clouds, it is still inevitable to label the wrong clouds in an image. As we mentioned in Section 2, one of the big differences between cloud types is the height. We can not determine the real height just according to an image, even for an experience meteorologist, it is still hard to identify altocumulus cloud and stratocumulus cloud. Since the shape of these two clouds is very similar, the most accurate method to distinguish them is measuring the height. We can see how similar it is in Figure 6

6 Web Application

As mentioned before, we will create a B/S (Brower and Server) architecture application which there will be a website deployed on a server. At the end of last year, TensorFlow delivered a demo solution for creating an object detection web application with uploaded images. In this project, we will modify the existing code and deployed our new model on the server. There are two reasons that we choose the web application as the final demonstration method. Firstly, a desktop server could have high computational capability which supports enough memory to deal with parallel requests and reduce the detection time. On the Google cloud computing platform, we can easily establish a sever with more than 48 gigabytes memory and 8 Intel Haswell CPU cores. With enough fund, a power graph card could be deployed on the server as well. Another reason is convenience. Through a simple website, the user from all over the world does not need to do anything just enter the IP address and upload an image on the website. To be more specific, the website enables the user to upload a JPEG, JPG or PNG image, then send it to the server. On the server side, it will detect the types of cloud on the image. After that, server will return an image with different cloud label and the user could click the label to see

the location of the cloud. The result shows in Figure 25.

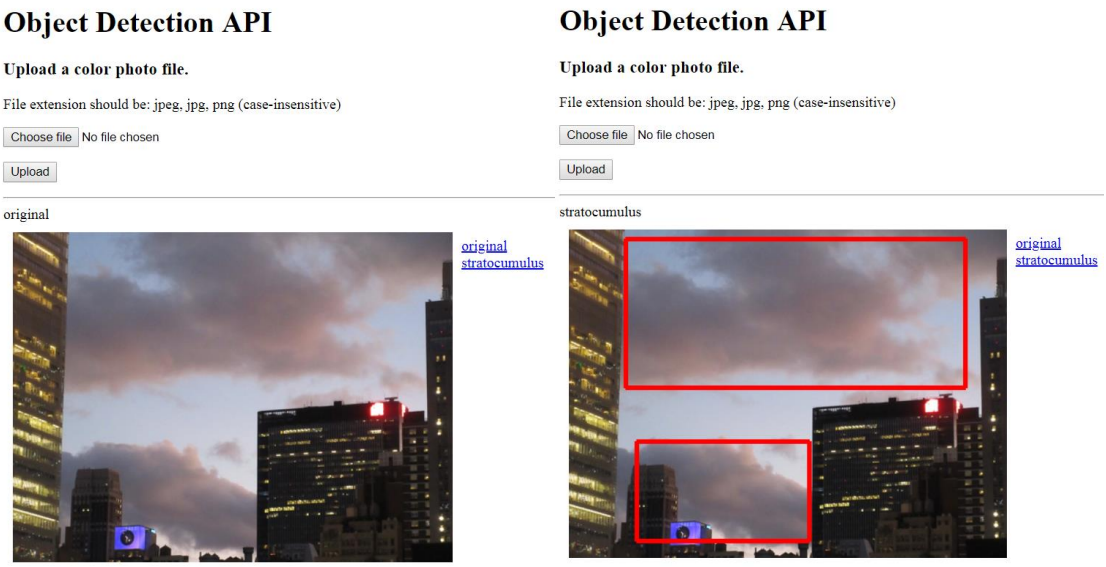


Figure 25 The result on the website

However, B/S mode brings some possible limitations which we must to declare. Firstly, the website and the server need to communicate with each other under the Internet that limits the user's usage scenes. Secondly, even we choose a model with less detection time, but take transfer latency time into consideration, the whole process might spend much more time than the theoretical situation. Especially when the server is down, the website will not able to access. Overall, a web application is the most suitable method to show the results of this project.

7 Future Work

With limited time and human resources, there still have some possible methods to improve the performance of the model. The following sections will propose the available suggestions which could be applied in the future.

7.1 Extend Dataset

In this project, we have collected 7,012 images with 11,144 different bounding boxes. However, it is still not enough to train a model. Besides, as we mention in Section 2.2, we removed the overlooking clouds image and only keep the ground look clouds image. In the future work, this restriction will be cancelled in order to get more data. Moreover, collecting enough noise image is another method to make the current model stronger when it meets more type of noise image,

such as cotton candy.

7.2 Mask RCNN

Mask RCNN is another complex and advanced object detect framework released by the Facebook AI Research. Expect the bounding box and label, it also generates a high-quality segmentation mask for each instance [30]. This model performs better on detect the large scale objects in the images, such as clouds, trees and sky. However, the pre-process image needs much more time compared with other models since we need to draw the mask area of each target. Besides, the detection time usually is 5 times greater than a normal framework. The demonstration of the result shows on Figure 26.

Figure 26 Mask RCNN result

7.3 Ceilometer

As we have emphasized many times, one of the most important factors to categorise a cloud type is height. If we could add a relevant height information as a parameter or the output of an activation function, the model will easy to exclude the cloud with a similar shape. Such as altocumulus cloud and stratocumulus cloud, altocumulus cloud and cirrocumulus. In the future work, we could cooperate with the Meteorological research community and use professional ceilometer equipment to measure the practical height during taking a photo of clouds. Figure

27 shows the normal ceilometer for calculating the cloud height.

Figure 27 The professional ceilometer

7.4 Mobile Application

According to the statistics research, the number of mobile phone users in 2019 will reach about 4.68 billion [31]. It is an extremely large market and we can implement an application on mobile platform. Mobile phone has two main advantage, internal camera and off-line mode. Nowadays, every mobile phone has at least one camera which could be used as the input data port for real-time detection. Once we deploy the model on the mobile platform, the real-time detection result will be displayed on the screen. Compared with a website, mobile phone is more intuitive and interesting. Another advantage is that mobile phone could be used without any signal. It just regards as a functional machine that used for identifying the cloud types. Even if the computational capability of a mobile phone is limited, we could chose a light model, such as Faster RCNN Mobilenet.

8 Conclusion

In summary, in this project, we use deep learning based technologies to solve the cloud classification problem. The trained model could detect 10 different types of clouds with overall IoU value of 0.46. We also compared 6 different object detection framework and choose the SSD ResNet50 v1 FPN as the final model. In this paper, we have introduced the steps of collecting and pre-processing the dataset. Then, the article illustrates the concept of transfer

learning and the possible issues that might appear during the training process. According to the result, we analysis the possible reason and try to optimize the model through two hyperparameters. Finally, we introduce the architecture of the web application and the future work for the next phase.

Appendix

Source code link: https://github.com/ClarkYang91/Computing_Project_90055

Demo video link: https://youtu.be/6D-H27kvrco

Reference

[1] Dechter, R., 1986. *Learning while searching in constraint-satisfaction problems* (pp. 178-183). University of California, Computer Science Department, Cognitive Systems Laboratory.

[2] Aizenberg, I.N., Aizenberg, N.N. and Vandewalle, J., 2001. Multi-Valued and Universal Binary Neurons: Theory, Learning, and Applications. *IEEE Transactions on Neural Networks*, *12*(3), p.647.

[3] Nath, S.S., Mishra, G., Kar, J., Chakraborty, S. and Dey, N., 2014, July. A survey of image classification methods and techniques. In *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)* (pp. 554-557). IEEE.

[4] Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M. and Thrun, S., 2017. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, *542*(7639), p.115.

[5] Medium. (2019). *Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and…*. [online] Available at: https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359 [Accessed 16 Feb. 2019].

[6] Skymind. (2019). *Comparison of AI Frameworks*. [online] Available at: https://skymind.ai/wiki/comparison-frameworks-dl4j-tensorflow-pytorch [Accessed 16 Feb. 2019].

[7] Scied.ucar.edu. (2019). *Cloud Types | UCAR Center for Science Education*. [online] Available at: https://scied.ucar.edu/webweather/clouds/cloud-types [Accessed 16 Feb. 2019].

[8] Dev, S., Lee, Y.H. and Winkler, S., 2017. Color-based segmentation of sky/cloud images from ground-based cameras. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *10*(1), pp.231-242.

[9] Image-net.org. (2019). [online] Available at: http://www.image-

net.org/api/text/imagenet.synset.geturls?wnid=n09247410 [Accessed 19 Feb. 2019].

[10] Pan, S.J. and Yang, Q., 2010. A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10), pp.1345-1359.

[11] Raina, R., Ng, A.Y. and Koller, D., 2006, June. Constructing informative priors using transfer learning. In *Proceedings of the 23rd international conference on Machine learning* (pp. 713-720). ACM.

[12] Wu, P. and Dietterich, T.G., 2004, July. Improving SVM accuracy by training on auxiliary data sources. In Proceedings of the twenty-first international conference on Machine learning (p. 110). ACM.

[13] Jabbar, H. and Khan, D.R.Z., 2015. Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). Computer Science, Communication and Instrumentation Devices.

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

[15] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S. and Murphy, K., 2017. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7310-7311).

[16] Ren, S., He, K., Girshick, R. and Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).

[17] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. and Berg, A.C., 2016, October. Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham.

[18] Medium. (2019). *What do we learn from single shot object detectors (SSD, YOLOv3), FPN & Focal loss (RetinaNet)?*. [online] Available at: https://medium.com/@jonathan_hui/what-do-we-learn-from-single-shot-object-

detectors-ssd-yolo-fpn-focal-loss-3888677c5f4d [Accessed 21 Feb. 2019].

[19] Towards Data Science. (2019). *A Simple Guide to the Versions of the Inception Network*. [online] Available at: https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202 [Accessed 21 Feb. 2019].

[20] 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).

[21] Towards Data Science. (2019). *An Intuitive Guide to Deep Network Architectures – Towards Data Science*. [online] Available at: https://towardsdatascience.com/an-intuitive-guide-to-deep-network-architectures-65fdc477db41 [Accessed 21 Feb. 2019].

[22] Towards Data Science. (2019). An Overview of ResNet and its Variants – Towards Data Science. [online] Available at: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035 [Accessed 21 Feb. 2019].

[23] 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).

[24] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[25] Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S., 2017. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2117-2125).

[26] GitHub. (2019). *tensorflow/models*. [online] Available at: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md [Accessed 22 Feb. 2019].

[27] Medium. (2019). *mAP (mean Average Precision) for Object Detection – Jonathan Hui – Medium*. [online] Available at: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173 [Accessed 22 Feb. 2019].

[28] Brownlee, J. (2019). *What is the Difference Between a Batch and an Epoch in a Neural Network?*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/ [Accessed 23 Feb. 2019].

[29] Towards Data Science. (2019). *Understanding Learning Rates and How It Improves Performance in Deep Learning*. [online] Available at: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10 [Accessed 23 Feb. 2019].

[30] He, K., Gkioxari, G., Dollár, P. and Girshick, R., 2017. Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).

[31] Statista. (2019). Number of mobile phone users worldwide 2015-2020 | Statista. [online] Available at: https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/ [Accessed 26 Feb. 2019].