Analysis of Street Recognition

Yun-Ping, Huang
Department of Computer Science
National Tsing Hua University
107062105

Tzu-Yu, Chuang
Department of Computer Science
National Tsing Hua University
108062315

Tun-Yuan, Chang
Department of Computer Science
National Tsing Hua University
109062211

Li-Ting, Huang
Department of Computer Science
National Tsing Hua University
109062224

Ju-Hsuan, Yu
Department of Computer Science
National Tsing Hua University
109062227

Si-Hong, Chen
Department of Computer Science
National Tsing Hua University
109062332

***Abstract*—In this project, we aim to explore the similarities and differences between how machine and human recognize street view images from the inspiration of the game - GeoGuessr.**

Four Convolutional Neural Network (CNN) models are trained using a dataset of street view images from four cities. The models are then evaluated and visualized to understand the features that the machine pays attention to. Furthermore, various experiments are conducted to enhance the performance of the models.

The objective of this project is to draw conclusions on the similarities and differences between machine and human street view images recognition.

I. Introduction

GeoGuessr is a game that players need to find the clues in the random pictures from `Google Street View` and guess actual locations. There are many strategies, like reading the traffic signs, identifying the buildings, etc, players will make use of. As we know, machine can be trained to classify the pictures. Therefore, we wonder whether machine would take the same features as human do to recognize the images of different cities and also compare the accuracy between human and machine.

II. Methods

A. Data Collection

The primary objective of our project is to develop models capable of recognizing street images from four distinct cities: Taipei, London, Bangkok, and Washington. To acquire a representative dataset, 2000 street images were uniformly collected from `iStreetView.com`.

The images were obtained by selecting specific streets, adjusting the angle of the camera and setting the image size before downloading. Each city was represented by 500 images, with a focus on selecting powerful features as identified by human observation, such as roads, traffic signs, notable buildings and representative objects.

B. Convolutional Neural Network Models

To classify the street view, we use two self-build models and two pre-trained models.

- self-build Convolutional Neural Network(CNN)
- self-build ResNet18
- Xception[1]
- DenseNet121[2]

C. Visualization

Given that Convolutional Neural Networks (CNNs) lack interpretability and can be perceived as a 'black box', it is necessary to employ visualization techniques to increase the concreteness of machine learning. To this end, we utilize the Grad-CAM[3] technique, which utilizes gradient information from the final convolutional layer of the CNN to assign importance values to each neuron in relation to a specific decision of interest. The greater the weight, the more significant the corresponding feature map is considered to be. Additionally, we employ the "jet" colormap, with red indicating a higher level of importance and blue indicating a lower level of importance.

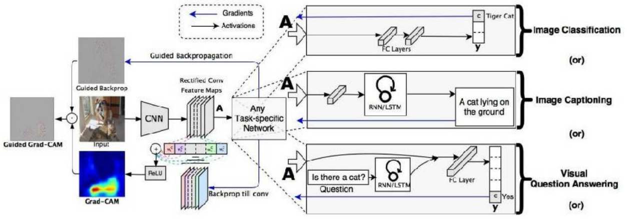


Fig. 1. Grad-CAM

D. Experiments

- Crop images
  One of the factors that affects the prediction is mosaic. Since Google Street View obtain their photos with special cameras on top of the cars, they apply special image processing algorithms to avoid gaps or seams. Mosaic is used to create smooth transitions for the images. However, the mosaic part of images is not helpful for recognition. To remove the effect of mosaic, which often appears at the bottom of the images, we crop the images to remove the lower portion of the image.

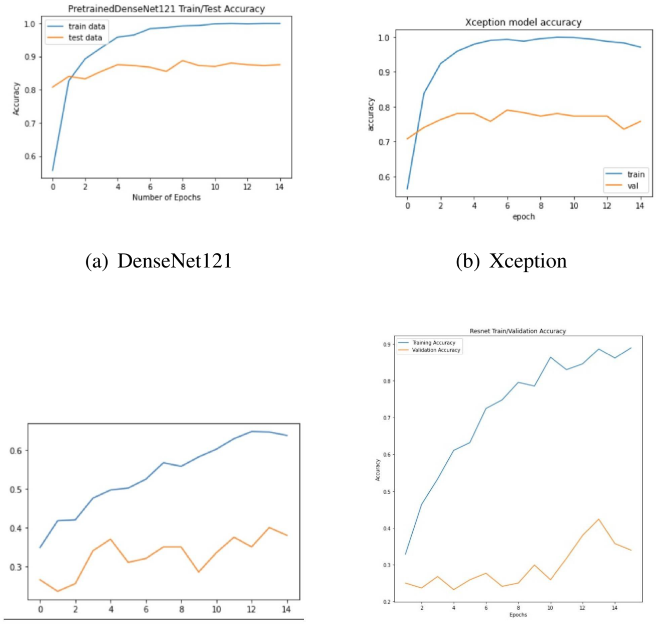
- Canny edge detector
  Light is another factor that may also affects the prediction. The exposure of photos may make the edge in the photos blurred and hard to recognition. To focus on the edges of the images and reduce the effect of light, we apply Canny edge detector to the images.
- Competition
  We download additional 25 images from `iStreetView.com` for test, choosing the model which has good performance as one candidate. Then we conducted the test ourselves and invited some of our friends to participate in the experiments. (The model developer and data collector may be seen as human with domain-knowledge.)

III. Result

A. Models performance comparison

TABLE I
MODEL PERFORMANCE

Table	**Models**			
Head	DenseNet121	Xception	CNN	ResNet18
training accuracy	0.999	0.970	0.648	0.447
validation accuracy	0.875	0.750	0.400	0.230



(a) DenseNet121

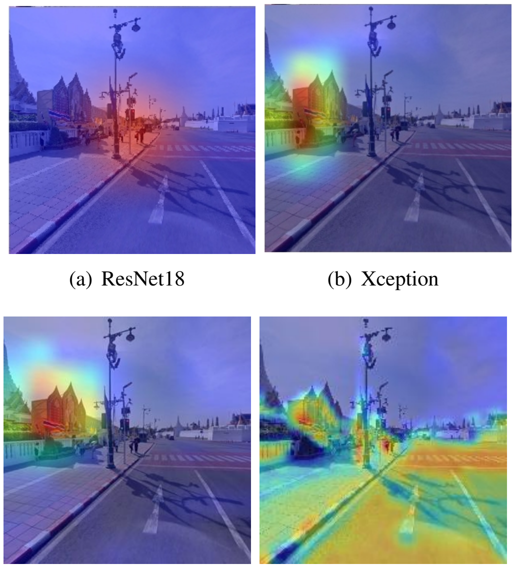
(b) Xception

(c) CNN

(d) ResNet18

Fig. 2. Accuracy curve

After training, we got better performance on two pre-trained models. Xception and DenseNet121 both have higher accuracy, more stable loss curve and less training time. The Table. I contain the accuracy of 4 models, respectively.



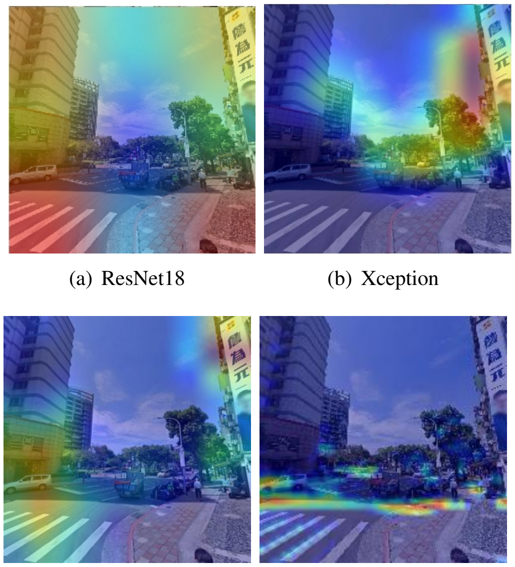
(a) ResNet18

(b) Xception

(c) DenseNet121

(d) CNN

Fig. 3. Example of visualization: Bangkok



(a) ResNet18

(b) Xception

(c) DenseNet121

(d) CNN

Fig. 4. Example of visualization: Taipei

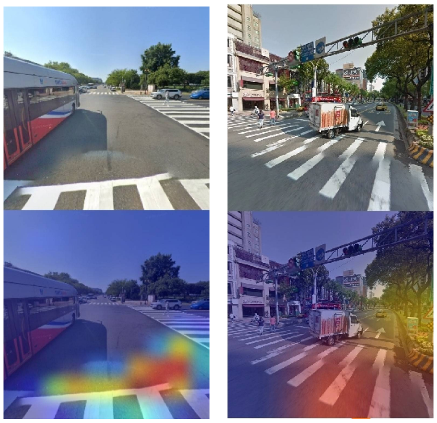
B. Visualization

Fig. 3 illustrates the result of using Grad-CAM on a street view image of Bangkok. As seen in the figure, the Resnet model focuses on the middle of the image, while the Xception and DenseNet121 models focus on the flag and religious building. The CNN model, on the other hand, pays attention to the borders of the image.

Fig. 4 illustrates the result of heatmap generated from an image of a street in Taipei. It can be seen that the Resnet model focuses on the corner of the image, while the Xception and DenseNet121 models focus on the buildings and signboards.

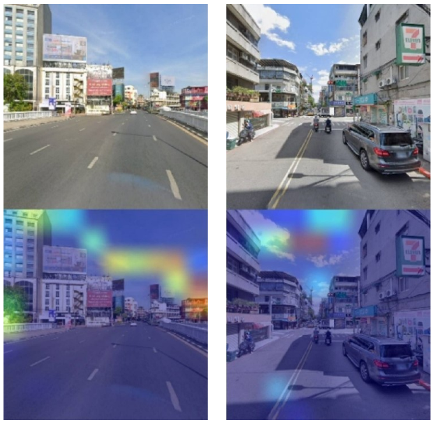
The CNN model, in this case, pays attention to the borders of shadows.

From the comparison of Fig. 3 and Fig. 4, we notice that the machine's view is similar to human's view. However, as illustrated in Fig. 5 and Fig. 6, the model sometimes pays attention to inexplicable objects. To address this issue, we carried out the experiments as mentioned in the Section. II-D.



(a) example 1 (b) example 2

Fig. 5. Prediction affected by mosaic



(a) example 1 (b) example 2

Fig. 6. Prediction affected by sky

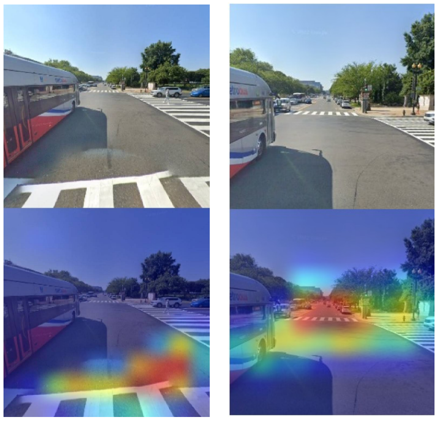
C. After croping and edge detect

TABLE II
EXPERIMENT EFFECT ON THE ACCURACY

Table	**Predict accuracy**			
Head	London	Taipei	Bangkok	Washington
Original image	0.995	1.000	0.960	0.980
Cropped image	1.000	1.000	1.000	0.998
Edge image	0.995	1.000	0.968	1.000

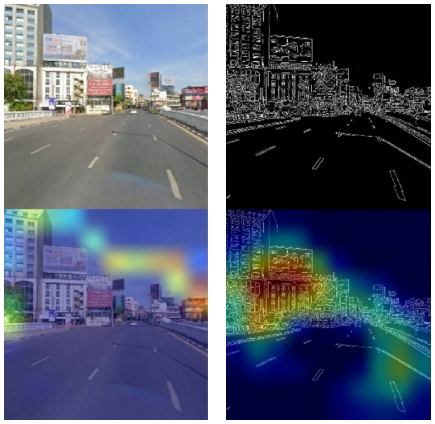
After conducting the experiments of cropping images and using Canny edge detector, we found that the accuracy of models was slightly improved, as shown in Table II. By observing the heatmaps after cropping the images and using the Canny edge detector, we observed that the trained models would focus on objects that are closer to what a human would notice.

As can be seen in Fig. 7 and Fig. 8, the utilization of the techniques for removing the effect of mosaic and light have been successfully implemented..



(a) before (b) after

Fig. 7. Result of cropping image



(a) before (b) after

Fig. 8. Result of using edge detector

D. Competition

TABLE III
COMEPTITION RESULT

Table	**Testers**		
Subject	machine	normal players	top players
Average Score	76	68	92

Our subjects were divided into three groups: machine, normal players (with minimal domain-knowledge), and top players (with full domain-knowledge). As shown in Table. III, the performance of the machine was better than that of the normal players but worse than that of the top players.

According to the feedback from the players, the features they focused on may have been the national text on the road sign or signboard of the shops. Additionally, due to the limitations of our data resources, the resolution of images is not high enough for the machine to capture the features identified in the players' feedback. As a result, the machine we trained was not able to consistently outperform human players in this game.

IV. Discussion / Conclusion

A. Confusion matrix

TABLE IV
Confusion Matrix

True Label / Prediction	Taipei	Bangkok	London	Washington
Taipei	0.964	0.012	0.014	0.010
Bangkok	0.068	0.904	0.010	0.018
London	0.030	0.014	0.950	0.006
Washington	0.028	0.024	0.038	0.910

As Table. IV shows, in our results, we found that Bangkok was more easily recognizable as Taipei than Washington or London. The possible reason for this is that both Taipei and Bangkok have a high density of cables and motorcycles, which may be similar visual features that the model is identifying. Additionally, Washington was more easily recognizable as London. This could be because the culture backgrounds and building styles of these two cities are quite similar, making them difficult to distinguish for the model. However, London typically features iconic objects such as red double-decker buses and red telephone boxes, which could be easily recognizable and help to differentiate it from Washington.

B. Factors affecting prediction

After conducting a series of experiments, we have identified several factors that play a role in how both humans and machines identify cities. For both the machine and human view, they tend to focus on distinctive features such as traffic signs and unique objects.

In the case of the machine view, it can be affected by factors that are not present in human view, such as camera differences and computational photography algorithms. These can result in variations in image quality and edge detection, and can be further impacted by factors such as mosaic and lighting.

In contrast, human view is often shaped by domain knowledge, and individuals tend to focus on specific features such as national flags and license plates. Additionally, the ability to read and understand text, especially when it is too small for machines to recognize, is a significant advantage for humans.

C. Future plan

We believe that this project has the potential for future development and application in various fields. For example, it could be used for verifying the authenticity of pictures and determining the location of photos. Furthermore, it could be applied in the field of tourism, by recognizing and providing information about the landmarks in a photo. These are just a few examples of the many potential applications that could benefit from the technology developed in this project.

V. Author Contribution Statements

- Yun-Ping, Huang (18%): self-build ResNet18 model, analysis of visualization & cropping image, presentation, PPT, report.
- Tzu-Yu, Chuang (16.5%): self-build CNN model, Canny edge detector, analysis of visualization & cropping image & texture(GLCM).
- Tun-Yuan, Chang (18%): Xception model, Grad-CAM visualization, analysis of visualization & cropping image & edge detector, report.
- Li-Ting, Huang (17%): report, data collection, analysis of edge detector & texture(GLCM).
- Ju-Hsuan, Yu (16.5%): data collection, presentation, PPT, analysis of edge detector.
- Si-Hong, Chen (14%): DenseNet121 model, analysis of visualization & cropping image,

References

[1] François Chollet. (Apr. 2017). ”Xception: Deep Learning with Depthwise Separable Convolutions”.

[2] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q Weinberger. (Jan. 2018). ”Densely Connected Convolutional Networks”.

[3] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. (October 2017). ”Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization”.

[4] K Scott Mader. (2019). ”Introduction to Texture Analysis”. [Online]. Available: https://www.kaggle.com/code/kmader/introduction-to-texture-analysis

[5] dshahid380. (2019). ”Convolutional Neural Network”. [Online]. Available: https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529

[6] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra. (2017). ”Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization”. 2017 IEEE International Conference on Computer Vision (ICCV)

[7] Sofiane Sahir. (2019). ”Canny Edge Detection Step by Step in Python — Computer Vision”. [Online]. Available:https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123