PinPlace: CNN based location image search and its adaptation to social network

Chae seungyun, Lee jiseop, Jeong chaewon, Uhm jiyoung, Hong seongjun

[1] Sungkyunkwan University, 2066, SEOBU-RO, JANGAN-GU, SUWON-SI, GYEONGGI-DO, KOREA
[2] Capstone project by prof. Hyung joon Koo

Abstract. We can get geographic location from place pictures. CNN based place recognition already exists, but there is little in domestic research and they are not practical in that they gives only place recognition function. Therefore, we developed CNN based place recognition in 10 of hot places which are targeted as MZ generation preference. In addition, we added posting function. By this function, people can comment places which someone want to know from picture. This function can help sharing information about places. Finally, this would be merits in commercial or tourism industry. About the development process, We developed various CNN models and choose the best model on performance. With this model, we made web application which has several pages which is necessary for our web program. The UI was designed through a program called Figma, and functions were dynamically implemented using CSS and JS. Also, we made main server code router, page, and controller. The sql is used for database generation script and the views is used for rendering pages.

Keywords: CNN · Place recognition · Social networking · MZ targeting · Confusion matrix

1 Introduction

We often see photos posted by others and want to know the place or information about the place. In this case, we wondered how we can know the location of the photo. There are two ways to find the location of a photo through a photo.

The first is to use the location tag. When taking a photo, the location tracking function is used to record latitude and longitude information into the photo, which is called a location tag. This allows us to know where the photo is taken. However, in this case, the location tag disappears if it is photo that you did not take, and you cannot use it in the case of a place photo saved by capturing. App camera is also capturing way.

The second method is the image search function. A typical example is Google's image search function. This function search images by using photos you already have, but it was not efficient for our purpose because it supports not only place photos but also various other kinds of photos.

To supplement the problems of the methods above, we decided to use a technology called CNN among deep learning tech. It is to identify and inform the location information with the photo that the users want to know where, and further apply it to the social network.

2 Related work

2.1 What is CNN?

Let me explain the CNN first. CNN stands for Convolutional Neural Network, and the principle of this technology is to first automatically create a filter to extract features from the image. This process is called convolution. In this way, we generate feature maps.

Next process is called pooling that reduces the number of dimensions and extract only important features. By repeating this process, feature maps with more complex patterns are formed. And Through these features, specific photo information is mapped and learned. This process is called a fully connected.

2.2 Related work

Binary feature detectors and descriptors(No CNN) First, it is an example of using a technology called binary feature detectors and descriptors rather than CNN technology. With this technology, visual place recognition can be implemented with high accuracy regardless of location or seasonal changes. However, at present, CNN technology is more accessible, so we plan to proceed with our project using CNN.

Russian tourism apps based on CNN Second, there is a study of tourism apps in Russia using CNN. This app has an AR-based location marking module and a CNN-based location-specific module. Uncomplicated CNN models are used to minimize the size of the app. However, there was a limitation that the AR module and the CNN module were completely separated and operated. we can just use CNN only. This way is more simple.

Google image search Third one is Google image search, which is an existing service. Google Image Search provides images that are similar to keywords associated with that image when a user uploads an image file. However, as I said before it provides not only photos of places. So we thought it is not fit into our project

Research on Seoul landmark prediction Lastly, a study similar to the direction we want to proceed. In this study, they implement CNN based on location info search service, when you input a photo of a landmark in Seoul, information about the place is output, they used about 9000 photos for data source. And they showed pretty good performance. It seems that our project can also expect high performance.

The position of our work CNN based place recognition already exists, but there is little in domestic research. Also, they gives only place recognition function. As this reason, they are not practical.

So, we developed CNN based place recognition on 10 classes in Seoul and we added posting function just like sns form such as Facebook, Instagram. We think this function can help sharing information about places. And this would be merits in commercial or tourism industry. This can be differentiation compared with existing research.

3 Proposed system/solution/service

3.1 Overall architecture

We developed various CNN models and choose the best model on performance. With this model, we made web application which has several pages which is necessary for our web program. The UI was designed through a program called Figma, and functions were dynamically implemented using CSS and JS. Also, we made main server code router, page, and controller. The sql is used for database generation script and the views is used for rendering pages.

3.2 Core skill

This project is CNN based place recognition. Our core skill is CNN. We provide where the picture is by using CNN.

3.3 General challenges and overcome

CNN Our data set is not sufficient to run CNN model. So, we tried cross validation during learning. However, this way could not improve the accuracy enough. Then we found the way of increasing data set, and we tried data augmentation. This helped increasing the accuracy.

At first, we could get about 60% accuracy on CNN models. We wanted to get better accuracy, then tried ResNet50 model. By using this model, we could get better accuracy about 90

Front end We had some difficulties working with the front end. It is difficult to create a user-friendly design because we have not all experienced UI/UX design. However, we were able to do a fancy design by referring to design references of various commercialized applications. In addition, we had to test with several mobile devices to ensure that no errors occurred and that it worked properly in various environments. Finally, we had difficulty handling backend related matters for functions interacting with servers, such as getting a list of locations from a database. In fact, there were too many files when linked to the backend part, so it was difficult to set the wrong path, but it was solved immediately.

Back end Developing backend had few challenges. Implementing image retrieval had certain limitation in implementation level. The database used for this project was MySQL. The SNS feature required the images to be saved in the server hence saving these images into the database was the first solution to this problem. MySQL also offered Blob type data to store the images. Few problem arose in this case first the transmission of image data in the web javascript to the server javascript made it difficult to handle the transmitted image. Second saving large data into the database did seem inefficient in working with a database like mySQL. Hence the backend server saves the image in a separate resource folder with the static files. The path to this image is saved into the mySQL database and the naming of these images take the user id, data and the original image name to not allow duplicates.

The second challenge was when combining the backend code with the front end code. Due to different styles in coding the general code readability was very low. This challenge did not require a solution but was rather time consuming to setup the front end pages such that it makes more sense in a back end server standpoint.

4 Design

The design of our project divided three parts: CNN, Front end, Back end

4.1 CNN

ML model / algorithm At first we choose 10 classes. We choose classes for MZ generation recommend hot places. We searched mass communication on goole or Naver then we agreed with 10 classes : “Dongdaemun Design Plaza”, “Gyeongui Line Forest Park”, “Naksan Park”, “Namsan Seoul Tower”, “The Hyundai Seoul Mall”, “Myeongdong Cathedral”, “Ikseon Dong Hanok Village”, “Jamsil Lotte Tower”, “Han River Sebitseom”, “Haebangchon”.

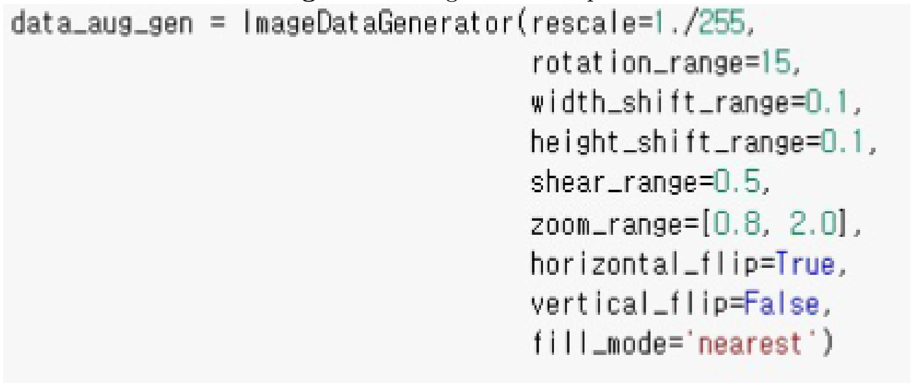
We choose CNN algorithm to predict the pictures users want to know. There are various CNN models. We select Lenet-5, AlexNet, VGG16, ResNet50 for architecture. This is representative models on CNN.

1. Lenet-5 has three convolution layer, two pooling layer, one fully-connected layer and this have about 60,000 parameter to learn. This model is basic model of CNN.
2. AlexNet model has five convolution layer, three pooling layer, two local response normalization layer, one fully-connected layer and this have about 62,000,000 parameter to learn.
3. VGG16 model has 13 convolution layer, 5 pooling layer, three fully-connected layer and this have about 138,000,000 parameter to learn. This is much deeper model than AlexNet.
4. ResNet model used idea of ‘skip connection‘ which solve the gradient vanishing problem which happens when model is deeper. It has 49 convolution layer with pooling layer and one fully-connected layer.

We run all this models and compare the accuracy and choose the best one.

We collect the data from google by crawling using selenium library. We tried Korean place name and English name for various data. We can collect almost 1,000 images in each class, but we remove same images or irrelevant images. Then we can get total 4,684 images. This is too small size of dataset to run CNN models. So, we use data augmentation by using ImageDataGenerator on keras library. Fig. 1 is the options

Fig. 1. Data augmentation options



```
data_aug_gen = ImageDataGenerator(rescale=1./255,
                                  rotation_range=15,
                                  width_shift_range=0.1,
                                  height_shift_range=0.1,
                                  shear_range=0.5,
                                  zoom_range=[0.8, 2.0],
                                  horizontal_flip=True,
                                  vertical_flip=False,
                                  fill_mode='nearest')
```

We generate 4 images for each image. So, we can get total 25,450 images Then we split train set, validation set, test set as 5:2:3. Batch size is 32 and epoch is 80. Also optimizer is set as Nadam.

To summarize,

- Total image data : 25,450
- Training validation data: 17,815
- Input Size : 128 * 128
- Train set, Validation set, Test set : 5:2:3
- Classes : 10
- Batch size : 32 epoch : 80
- Optimizer : Nadam

4.2 Front end

Reason about design choice We decided to create an application with a web-based application because we wanted to provide it in a form that ordinary people could easily use. Therefore, the UI was designed in an intuitive and comfortable form. In fact, the application UI that won the award at the Design Awards was referenced as a reference, and free 3d graphics were also used. Finally, the UI was designed primarily through a program called Figma, and functions were dynamically implemented using CSS and JS.

The main server code opens the webserver using express. If a request is received the server code should handle these requests. However in implementing a web app, multiple requests would come and having all these handling function in one code decreases code readability. Hence, the server page routes certain requests to the router.

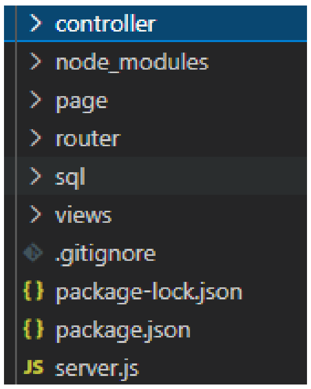
Design from UX/UI viewpoints First, we implemented it as a "Responsive web-based application" and designed it to be most optimized for the representative mobile phone model, iPhone X (375 * 812).

We initially designed a total of 8 pages, and the feedback of the professor, we added pages that users can sign up and login to implement a total of 10 pages. The following is a list of 10 pages that we finally designed.

Cover page / Start Page / User Guide Page / Signup Page / Login Page / Find Location Page / List Up Page / Upload Picture Page / SNS Page / My Page

When we think of the order of designing and linking pages, we constantly thought of user flow. We thought about how users can use the application comfortable, and even thought about the location of a small button. In other words, our ultimate goal in the FE part was to provide optimal UX to our potential customers that will use our platform.

4.3 Back end

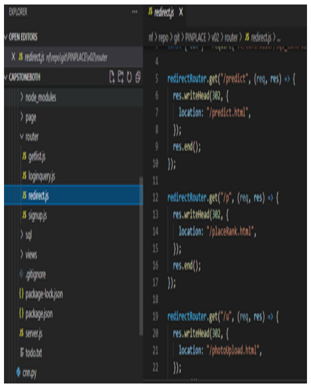


```
> controller
> node_modules
> page
> router
> sql
> views
.gitignore
package-lock.json
package.json
server.js
```

Fig. 2. server code structure

Server code structure The General structure include the main server code, router, page, and controller. The sql is used for database generation script and the views is used for rendering pages. The reason the server code is split into 3 parts: server code, router, and controller is to improve code readability and allow for flexible debugging if a bug were to occur.

The main server code opens the webserver using express. If a request is received the server code should handle these requests. However in implementing a web app, multiple requests would come and having all these handling function in one code decreases code readability. Hence, the server page routes certain requests to the router.



```
redirectRouter.get("/predict", (req, res) => {
  res.writeHead(302, {
    location: "/predict.html",
  });
  res.end();
});

redirectRouter.get("/p", (req, res) => {
  res.writeHead(302, {
    location: "/placeRank.html",
  });
  res.end();
});

redirectRouter.get("/u", (req, res) => {
  res.writeHead(302, {
    location: "/photoUpload.html",
  });
```

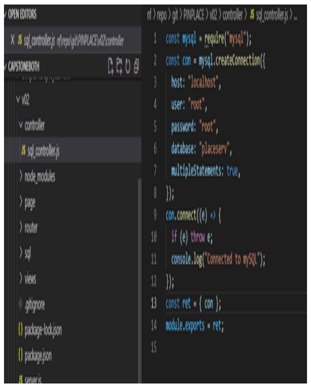
Fig. 3. router structure

Router basic structure The router will receive its signal via app.use function in the server code. For example the redirect.js would receive its signal via: app.use("/redirect", redirectRouter) where redirectRouter is the router in redirect.js which was exported as a module. These redirect signals will come to the server as "/redirect/predict" or "/redirect/p" where it would be routed to the router as "/predict" or "/p". This allows new implementation or debugging easier when a new redirect protocol is needed or if the redirect protocol is broken.

Although not utilized as much in this project, the controller holds callback functions that the rest of the code can use.

controller structure The controller currently holds the sql connection and exports the connection such that the server or routers can use it. The controller is where the overall code can be shortened immensely as it can store all the functions used by all the other codes.

The overall aim of the backend design is to improve code readability. Unlike normal programs webapps would require new features to be implemented. Hence a clean structure would allow the overall code functionality easier to visualize.


OPEN EDITORS
X JS sql_controller.js nf\repo\git\PINPLACE\v02\controller
CAPSTONEBOTH
v02
controller
sql_controller.js
node_modules
page
router
sql
views
.gitignore
package-lock.json
package.json
server.js
nf › repo › git › PINPLACE › v02 › controller › JS sql_controller.js › ...

```
const mysql = require("mysql");
const con = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "root",
  database: "placeserv",
  multipleStatements: true,
});
con.connect((e) => {
  if (e) throw e;
  console.log("Connected to mySQL");
});
const ret = { con };
module.exports = ret;
```

Fig. 4. controller structure

5 Implementation

5.1 CNN

We tried 4 different CNN models. There are Lenet-5, AlexNet, VGG16, ResNet50. We ran all these models in the same circumstance and then compare the accuracy of them. Table 2. is the results of CNN models

Table 1. Accuracy of CNN models

Lenet-5	AlexNet	VGG16	ResNet50
66.3%	13.31%	12.79%	91.12%

AlexNet and VGG16 model has remarkably low accuracy. Although dataset is not enough to run them, they have so deeper layers and so many parameters to learn.

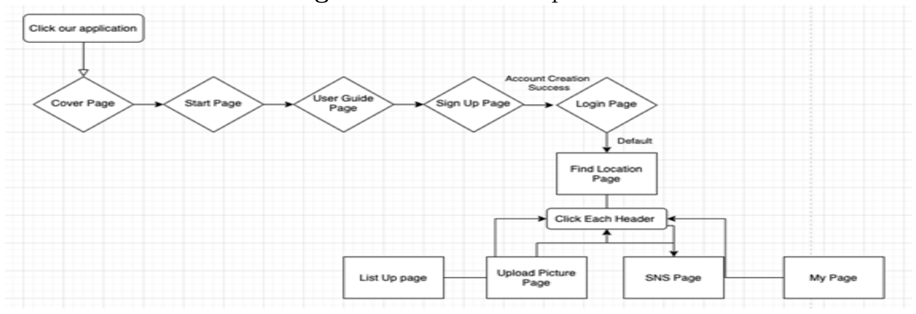
However, ResNet50 model has less parameters than other models and this model solve the gradient vanishing problem by skip connection. In this reason, this can have 50 layers which is deeper than other models.

As a result, we can decide to adopt this ResNet model which has the best accuracy.

5.2 Front end

First, we implemented our platform in the form of responsive web-based application with HTMl, CSS, Javascript. The below picture is our platform's user flow. This diagram shows the inter-page connection structure of our web-based application. Fig. 5 is a brief description of each page implemented.

Fig. 5. User flow of our platform



1. Cover page
   Since it is the first screen that users face, We designed the logo ourselves because we thought we had to firmly convey the platform brand image.
   – Top Layer
     • "PINPLACE" (Our platform's name)
   – Bottom Layer
     • 3d graphic assets (Represents our platform's brand image)
     • Include in "Onclick Function: that can move next page
2. Start Page
   This page is expressed in fancy graphics to roughly imply the functionality of our platform
   – Top Layer
     • 3D Rotated Cube (with diverse place's pictures)
     • Implemented by setting x,y, z- axis angles with css
   – Bottom Layer
     • Start Button (with onclick function)
3. User Guide Page
   For optimal UX, we made this page with Card UI. Every time user turn the page, the content and design are designed to be different.
   – In card UI
     • Put related graphics
     • Put button below 'PREV, NEXT, FINISH'
     -PREV : onclick function that move previous card
     -NEXT : onclick function that move next card
     -START: onclick function that connect sign up page
   – Bottom Layer
     • Start Button (with onclick function)
4. Sign Up Page
   This page is for new users who want to make an account for this service. Currently this includes four text or password boxes, and a submission button.
   – Submission form
     • ID, nickname, and password
     • Two password boxes that prevent mistakenly typed password

5. Login Page
   To use the service, users need to sign in via this page. Among the information provided in the sign-up page, ID is unique for each user: thus, ID and password are needed to log in. Additionally, there is the button to the sign-up page for who doesn't have an account for this service.
   - Submission form
     - ID and password
6. Find Location Page
   This page is core function page. We connect with CNN model that we made ourselves.
   - User flow
     - Click Choose File button
     - Put Input file (regardless of files' extension)
     - Click Predict Button
     - Appear Output(location)
7. List Up page
   This page shows the list of places serviced, by popularity. Popularity can be measured by daily, weekly, or monthly. Each place entry is clickable and shows a subpage for that place.
   - Place list
     - Shows top ten places; highlights top three
     - Most recently uploaded photograph for each place
   - Subpages
     - Shows uploaded photographs by recency
8. Upload Picture Page
   This page is prepared for improving AI model, so the location information for the picture is necessary. The dropdown list for locations needed is served.
   - Submission form
     - Upload button (shows preview after uploading)
     - Dropdown list of locations
     - Two buttons: Upload and Cancel
   - After uploading
     - A page saying "Thank you"
     - The user can choose to upload more or to quit from this page
9. SNS Page
   - Top Layer
     - Develop Image Slider that place's images are moved automatically
   - Bottom Layer
     - Posting function to recommend a location.
10. My Page
    This is the own user page for a user logged in, which shows pictures uploaded by that user from find location page.
    - Layout
      - Shows uploaded photographs by recency
      - User's nickname and profile photo

5.3 Back end

The backend implementation is quite simple. The server is opened in port 8898 which is an arbitrary choice and can be modified by changing the PORT variable in server.js. The server is opened through express. The server uses express.static function to redirect user to page. This implementation was done such that front end pages and resources can be added easily. A better implementation would be placing all resources, scripts, style files in the static page but leaving out the html pages which would be controlled by the main server code in redirection. However for the sake of simple implementation the html files are included in the static folder.

The static html files will request certain redirections, images, and text to the server. The server will recognize these requests and handle this request. Data is mainly stored in the sql database. However, the images are saved within the resource folder. The sql database will hold the url to this resource folder. This was done since saving large data into the database was unnecessary and in a real webpage image will hold their own url saved in some separate database. Simulating this behavior but saving money the images are assumed to have a url while this url is the path to the image saved in the hard disk.

Login request is also something tackled in the backend implementation. Certain factors were simplified. The backend server adds id and password information to the database in registration. The password is hashed and saved with its salt value. When login request is given the password string is compared with this hashed password. Since the database has the salt as well the hash can be decoded and compared.

After login is complete the user pages should be rendered specific to the user. This section would be the frontend task so the backend should offer user data to the frontend page. The user data is saved as a cookie in the browser. The lifetime of this cookie is the session lifetime. This cookie data is used in rendering specific user pages.

In implementing the backend server multiple opensource modules were used.

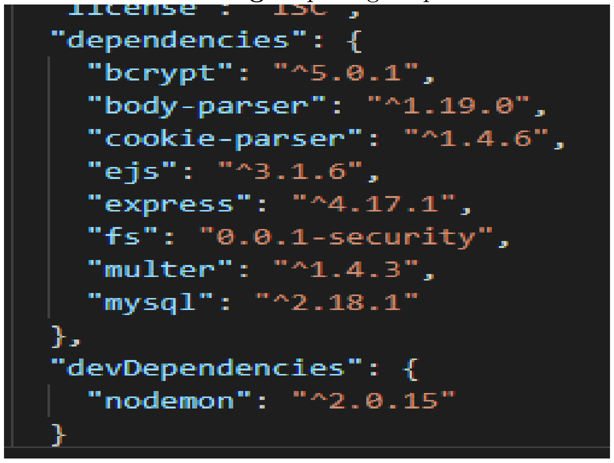
Express, mysql, body-parser is used for the webserver and database management. Express is also used in routing the signals. Bcrypt allows the password to be hashed and bcrypt also provides the salt to be used for hashing. Cookie-parser is used for cookie generation and usage for user identification data. The ejs is used for rendering the pages. Fs and multer is used for file system and image storing and loading. Finally, nodemon was used for development purposes and is not needed for the webapp to function.

6 Limitations

6.1 CNN

Although we do our best, 91.12% accuracy is not enough to users. The accuracy should be increased. If more complicate model is used, there might be better models which has accuracy more than 91.12

Fig. 6. package dependencies used in project

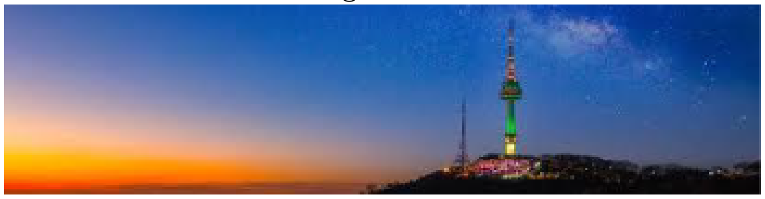


```
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^5.0.1",
    "body-parser": "^1.19.0",
    "cookie-parser": "^1.4.6",
    "ejs": "^3.1.6",
    "express": "^4.17.1",
    "fs": "0.0.1-security",
    "multer": "^1.4.3",
    "mysql": "^2.18.1"
  },
  "devDependencies": {
    "nodemon": "^2.0.15"
  }
```

Also, there are problem of range. There are wide range of classes among our 10 classes. For example, “Haebangchon”, “Ikseon Dong Hanok Village”, “Gyeongui Line Forest Park” have wide range of geographical place. Users can take photos at different places and then CNN model can not easily predict right places.

The other problem is about distance. For instance, users take photos like towers which is “Namsan Seoul Tower”, “Jamsil Lotte Tower” among our classes. If the picture is taken far from the tower, the shapes of towers look so similar. Actually, our CNN model can confuse the “Namsan Seoul Tower” and “Jamsil Lotte Tower” which are taken far from towers. You can comprehend this as comparing following two pictures.

Fig. 7. Namsan seoul tower



6.2 Front end

As the pages are designed by non-professionals, stylings are minimal and visual is sometimes awkward, despite the struggle to catch that. Mostly, however, the design is user-friendly from its simplicity, and the user experience is clean.

Fig. 8. Jamsil Lotte Tower

There are features planned but not made or implemented ad hoc. Firstly, the List-up page intends to show places by popularity, but because the popularity data are unavailable for pages with low access frequency, those are substituted by arbitrary numbers. Secondly, it lacks user information-related pages. The user information like the profile photo should be able to be changed, but there is no such page. The service claims to be a social network, but the actual posting function is not prepared. Additionally, a function that a user can favorite a place have been a consideration but is not implemented. Despite that, the initially planned page is all done by their design and functions, and the aforementioned tasks are remaining as future work.

6.3 Back end

The biggest limitation would be the weak security the page has. As mentioned before using express.static for html files is fine if the webpage has no login feature. Unfortunately, when login is available allowing user to enter webpages via static calls is problematic. The better solution would be to leave the html files out of the static folder. Furthermore, the best solution would be to render all pages like the user page after login. Rendering pages would require user token cookie which could only be acquired if the user logs in.

The assumption was that the user would not use the url to enter the website without logging in. If this is done that the user can enter the website without the login cookie which causes problems when the user attempts to enter the user page. Also, the cookie should be encrypted and checked closer as it is currently acting as the key into the user page. Although the overall webserver has security issues most of these problems would be fixable.

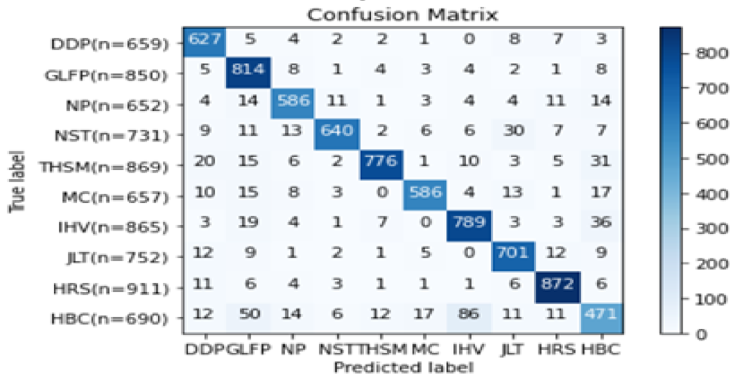
7 Evaluation

We evaluate the CNN model by accuracy and confusion matrix. About the accuracy we compared each CNN models which we tried. ResNet50 is the best ones. However, we tried confusion matrix to improve the accuracy. Confusion matrix is the matrix which shows the number of correct and incorrect predictions briefly.

The following figure is first confusion matrix we tested. X axis means the class which model predicted, and Y axis means the class which is the real class. And the class is notated by acronym. For example, the case that prediction is

"Ikseon Dong Hanok Village", but real data is "Haebangchon" is 86. We can realize that the places which have wide range of place such as "Haebangchon", "Ikseon Dong Hanok Village", "Gyeongui Line Forest Park" are confusing to model.

- DDP : Dongdaemun Design Plaza
- GLFP : Gyeongui Line Forest Park
- NP : Naksan Park
- NST : Namsan Seoul Tower
- THSM : The Hyundai Seoul Mall
- MC : Myeongdong Cathedral
- IHV : Ikseon Dong Hanok Village
- JLT : Jamsil Lotte Tower
- HRS : Han River Sebitseom
- HBC : Haebangchon



	DDP	GLFP	NP	NST	THSM	MC	IHV	JLT	HRS	HBC
DDP(n=659)	627	5	4	2	2	1	0	8	7	3
GLFP(n=850)	5	814	8	1	4	3	4	2	1	8
NP(n=652)	4	14	586	11	1	3	4	4	11	14
NST(n=731)	9	11	13	640	2	6	6	30	7	7
THSM(n=869)	20	15	6	2	776	1	10	3	5	31
MC(n=657)	10	15	8	3	0	586	4	13	1	17
IHV(n=865)	3	19	4	1	7	0	789	3	3	36
JLT(n=752)	12	9	1	2	1	5	0	701	12	9
HRS(n=911)	11	6	4	3	1	1	1	6	872	6
HBC(n=690)	12	50	14	6	12	17	86	11	11	471

Fig. 9. confusion matrix

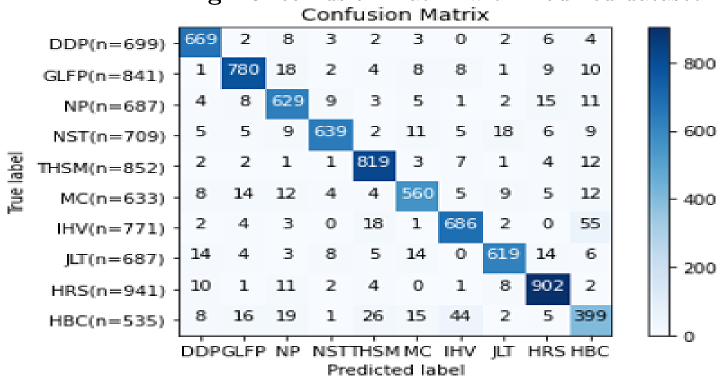
To remove the confusion, we discard some ambiguous pictures in each class. Then we can better confusion matrix as bellow and get 91.12% as accuracy.

8 Conclusion

Our project is complete for producing CNN based place recognition function and SNS function. Although this model has 91% accuracy and only have 10 classes, this web application can accept the new class images from users, and we can updated the model. In this reason, our program has expansion to process various places.

Also, we developed successfully SNS features. Although this web needs more classes, this can help communicate each other with information about places. Then this would enhance community development. In this point, this program has commercial trait.

Fig. 10. confusion matrix with modified dataset



Confusion Matrix

True label \ Predicted label	DDP	GLFP	NP	NST	THSM	MC	IHV	JLT	HRS	HBC
DDP(n=699)	669	2	8	3	2	3	0	2	6	4
GLFP(n=841)	1	780	18	2	4	8	8	1	9	10
NP(n=687)	4	8	629	9	3	5	1	2	15	11
NST(n=709)	5	5	9	639	2	11	5	18	6	9
THSM(n=852)	2	2	1	1	819	3	7	1	4	12
MC(n=633)	8	14	12	4	4	560	5	9	5	12
IHV(n=771)	2	4	3	0	18	1	686	2	0	55
JLT(n=687)	14	4	3	8	5	14	0	619	14	6
HRS(n=941)	10	1	11	2	4	0	1	8	902	2
HBC(n=535)	8	16	19	1	26	15	44	2	5	399

References

1. 2020 20th International Conference on Control, Automation and Systems (ICCAS 2020) Oct. 13 16, 2020; BEXCO, Busan, Korea
2. O. S. Laptev and I. I. Bikmullina, "Sightseeing Application Based on Location Marking and Convolutional Neural Network Building Recognition," 2020 International Russian Automation Conference (RusAutoCon), 2020, pp. 209-214, doi: 10.1109/RusAutoCon49822.2020.9208062.
3. Journal of The Korea Society of Computer and Information Vol. 25 No. 9, pp. 31-36, September 2020
4. Yoonah Lee, Jongho Nang (2018). Building Training Image Dataset and Experiments with Convolutional Neural Network for Seoul Landmark Recognition. KOREA INFORMATION SCIENCE SOCIETY Academic Presentation Paper Collection, 845-847