Computer Vision Lab5

Group 12 | 0786031 廖俊凱 0516044 陳思妤 0856733 黃明翰

Introduction

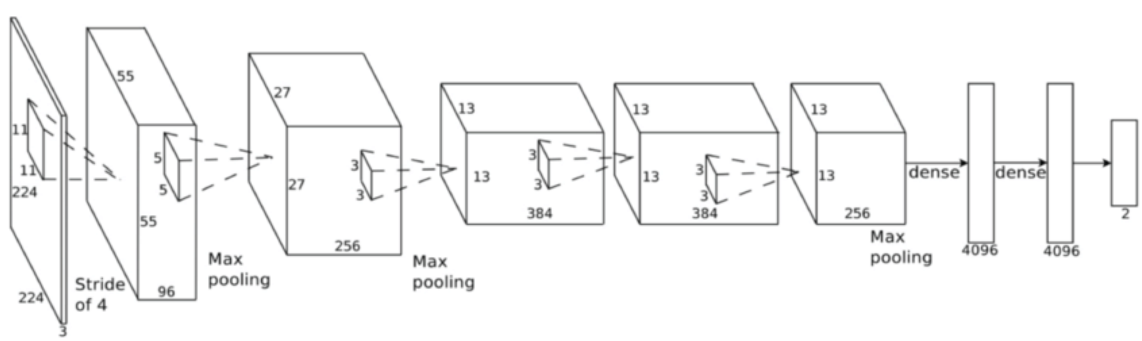
The goal of this lab is to solve a classic computer vision problem, image recognition. In this lab, we conduct different methods, from basic machine learning techniques such as nearest neighbor classifier and linear SVM classifier to state-of-the-art deep learning models, to categorize images into 15 different scenes.

Bag of SIFT

Bag of features is an efficient way to describe the feature of an image. After an image is described by bag of features, it can be seen as a data point. With multiple training images (data points), we can use different classifier to do classification on the testing images. We use nearest neighbor classifier in task 2 and use linear SVM classifier in task 3.

Bonus

In the bonus task, we try to build AlexNet as our CNN model. AlexNet was introduced in 2012, named after Alex Krizhevsky, the first author of the breakthrough ImageNet classification paper [Krizhevsky et al., 2012]. AlexNet, which employed an 8-layer convolutional neural network, won the ImageNet Large Scale Visual Recognition Challenge 2012 by a phenomenally large margin. This network proved, for the first time, that the features obtained by learning can transcend manually-design features, breaking the previous paradigm in computer vision. The architectures of AlexNet and LeNet are very similar.



Example AlexNet architecture diagram

Implementation Procedure

1. Tiny images representation + nearest neighbor classifier

A. Resize the image into a small, fix resolution (16x16) using cv2.resize(). To maintain the image size as a vector of 16 * 16, use cv2.cvtColor() function to transfer the original images into grayscale images.

B. Implement K-nearest neighbor classification by using cv2.ml.KNearest_create(). We also implement KNN classifier with Manhattan distance. (KNN classifier implemented in OpenCV uses Euclidean distance) To find out the reasonable K value with the highest accuracy, we tried 1 neighbor to 5 neighbors to compare their accuracies. (voting policy is used when choosing k value bigger than 1)

2. Bag of SIFT representation + nearest neighbor classifier

A. Bag of features

I. Extract features

We use SIFT to extract features for both training and testing images.

II. Learn "visual vocabulary"

After getting features from training images, we use k-means clustering to get k codevectors. The visual vocabulary can be built by the combination of k codevectors.

III. Quantize features using visual vocabulary

A feature vector can be mapped to the nearest codevector by vector quantization. Therefore, an image (many features) can be represented by a visual vocabulary.

B. Nearest neighbor classifier

I. For each testing image:

i. Calculate the distance between these training images.

- Euclidean distance
- Manhattan distance
- Person correlation

ii. Voting (without/ with weights)

- Without weights
- With weights by using the ranking
- With weights by using the distance

iii. Compare the result with the truth.

3. **Bag of SIFT representation + linear SVM classifier**

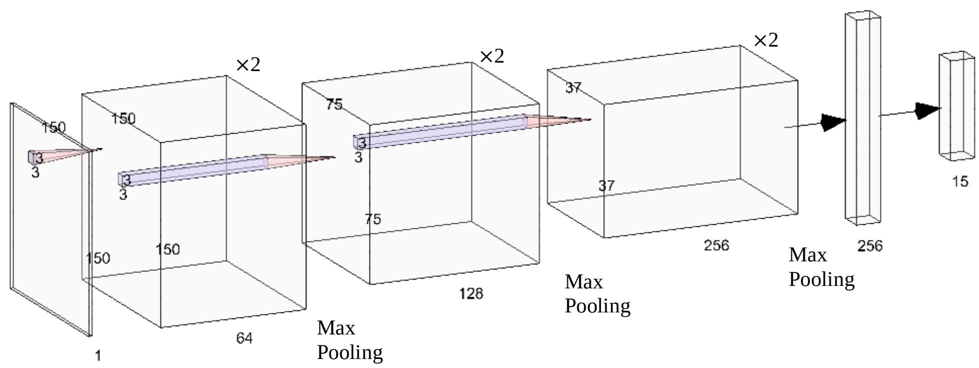
In task3, we use the features collected from the task2 as our training set and testing set. we conduct **libsvm** to build our linear SVM model and train the model using our training vectors.

SVM is the most powerful classifier before deep learning which uses kernel functions to map data points to high dimension and find a proper decision boundary. However, the hyperparameters of SVM model will severely affect the performance. Thus, we also conduct grid search to find the best hyperparameters for our linear SVM.

4. **Bonus**

A. CNN

First, we try to build a CNN model, which is similar to AlexNet, using Keras library. This model takes in images shape = (150,150,1), contains 8 layers and outputs 15 probabilities with activation function softmax.



Our first CNN model

We set the parameters as following:

optimizer	loss function	metrics	epochs	Batch size
Adam	categorical cross-entropy	accuracy	20	32

In the end, our model suffers from serious overfitting. We think the cause of overfitting is mainly due to that the training set is not large enough. Thus, we conduct two ways to solve the overfitting problem: Data argumentation & Using pre-trained model with feature extraction.

B. AlexNet + Data argumentation

First, due to our hardware constraint, we reduce the depth of our CNN model to half, then we can enlarge our input image to (256, 256, 1).

We use data generator to feed images, which will also randomly shift images horizontally or vertically, rotate or flip images. Here, we also modified our optimizer, learning rate and epochs for better result, the details are as following:

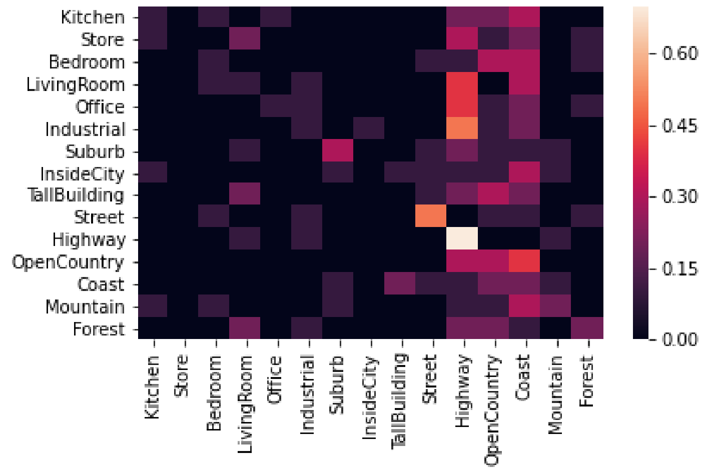
optimizer	loss function	metrics	epochs	Batch size
Rmsprop (lr=1e-4)	categorical cross-entropy	accuracy	**300**	32

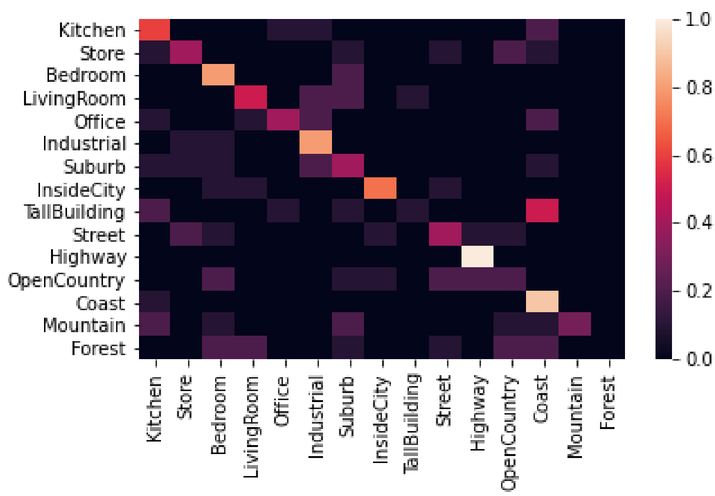
After training, our model performance on testing set is 76%, which has already outperformance linear SVM. However, the training accuracy can reach to almost 1, this means our model can still be improved. Suffering from the little number of training images, we try to use the pre-train network VGG16 Place 365 to help us.

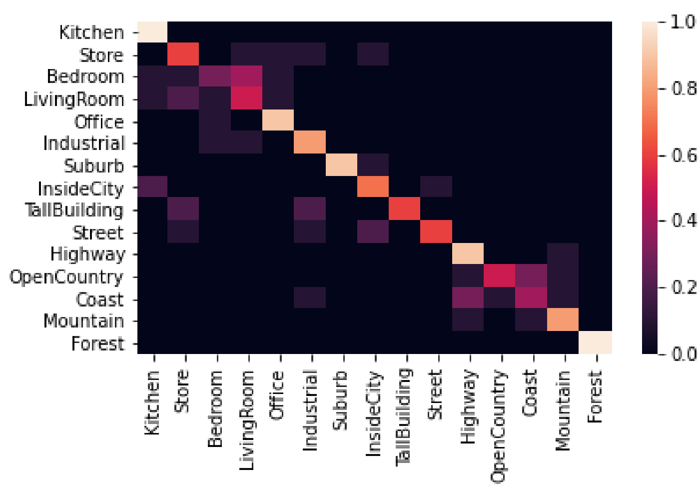
C. VGG_Place365

VGG16 Place 365 is a CNN model pre-trained on Places365-Standard, which contains ~1.8 million images from 365 scene categories, for scene classification [1]. We load the VGG16 model without top layers, freeze the convolutional base and attach our own Dense layers, which will output 15 probabilities with activation function softmax. We than train the model with data generator as we used in the last step. The evaluation result on testing set is **90.67%** after 50 epochs, which is our best score!

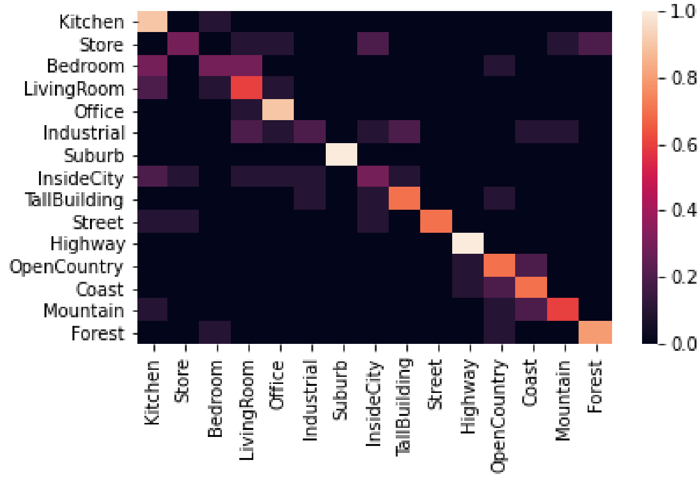
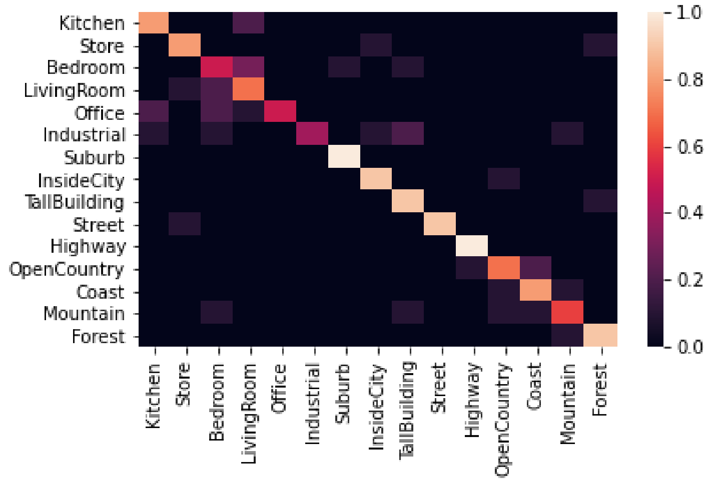
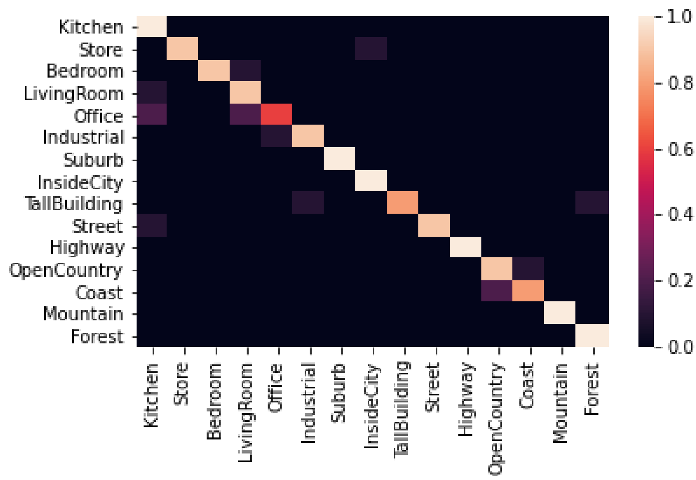
Experiment Result

<table>
<tr><th>Method</th><th>Acc</th><th>Confusion Matrix</th></tr>
<tr><td>Tiny + NN</td><td>19.33%</td><td>


</td></tr>
<tr><td>BoS + NN</td><td>50%</td><td>


</td></tr>
<tr><td>BoS +
linear SVM</td><td>70%</td><td>


</td></tr>
</table>

CNN	64.67%	
CNN(AlexNet) + Data Augumentation	76%	
CNN (VGG16_Place_365)	90.67%	

Tiny + NN (Task1)

Accuracy without image normalization:

Neighbors	cv2.ml.KNearest function	KNN with Manhattan distance
1	**14%**	**14%**
2	10.67%	**14%**
3	**14%**	13.33%
4	11.33%	13.33%
5	11.33%	12.67%

From the Table, we can find out that the accuracies are lower than the result requested in spec, so we tried to normalize the image before classification.

Accuracy with image normalization:

Neighbors	cv2.ml.KNearest function	KNN with Manhattan distance
1	18.67%	**19.33%**
2	14%	18%
3	14.67%	15.33%
4	15.33%	12.67%
5	14.67%	11.33%

Bag of SIFT + NN (Task2)

Set K vary from 5 to 100 to find a better classification result.

K	Accuracy	K	Accuracy
0	6.7%	50	44%
5	41.3%	55	47.33%
10	43.33%	60	48%
15	44%	65	47.33%
20	44%	70	48%
25	46.7%	75	48%
30	44.67%	80	47.33%
35	46.67%	85	**50%**
40	44%	90	**50%**
45	47.33%	95	**50%**

Bag of SIFT +linear SVM (Task3)

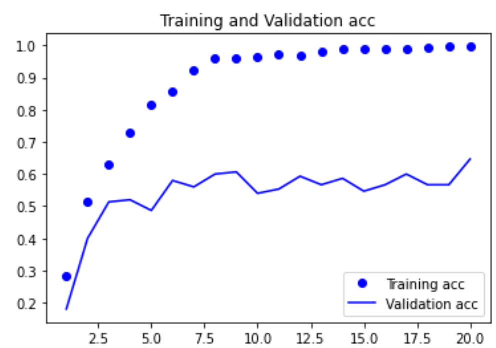
In the SVM method, the hyper-parameters will severely affect the accuracy. Thus, we conduct grid search on 5-fold cross-validation to find the best hyper-parameters.

Cost (2**c) c	Accuracy
-1	34.67%
0	33.13%
1	34.4%
2	35.4%
3	45.2%
4	56.47%
5	60.73%
6	65.4%
7	67.6%
8	68.47%
9	**68.8%**
10	67.4%

In the end, we choose cost=2^9=512 and get the accuracy on testing set = **70%**.

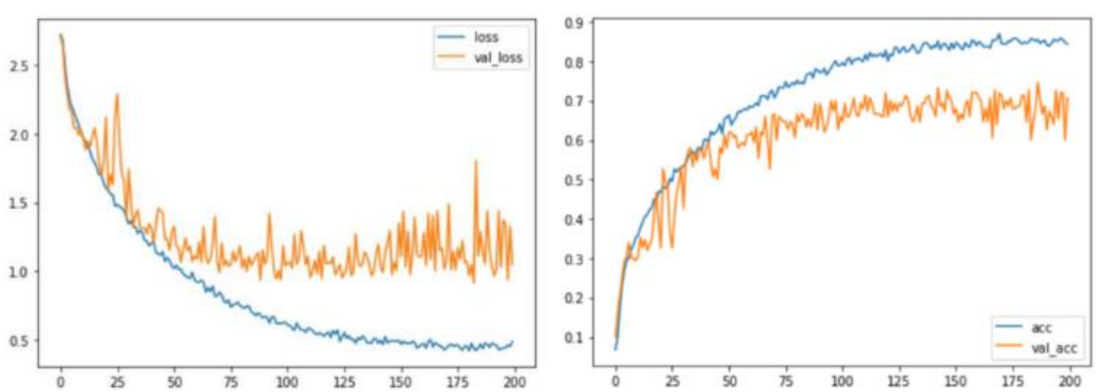
Bonus

CNN:



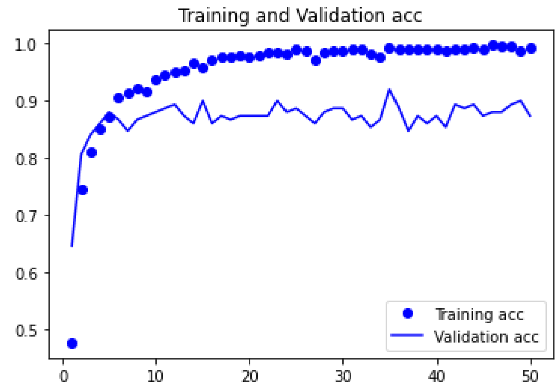
From the plot, we can find out that the accuracies can reach to almost 1.0 in the training set when epoch reaches 20, but it experiences a serious overfitting in validation set although we have already put many dropout layers to avoid overfitting. Thus, we try to use some **data argumentation** and **pre-train model** to help us with better validation accuracy.

CNN (AlexNet) + Data Argumentation:



Training and Validation losses / accuracy with AlexNet + Data Argumentation

CNN (VGG16_Place365):



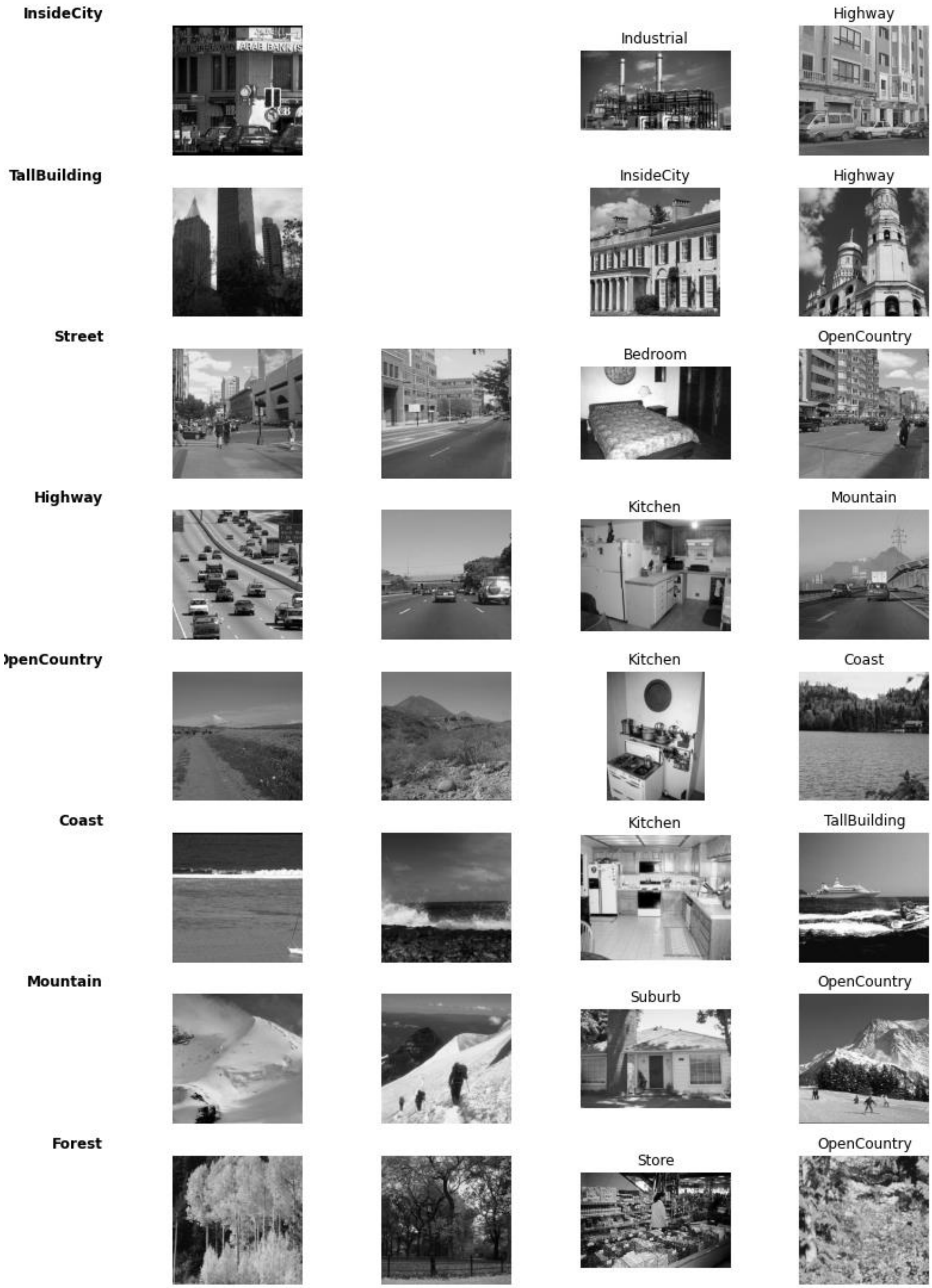
Training and Validation accuracy with VGG16 Place 365

The overfitting situation become better and the final accuracy of testing set is **90.67%**.

Prediction Results

The **missing image** means no corresponding result.

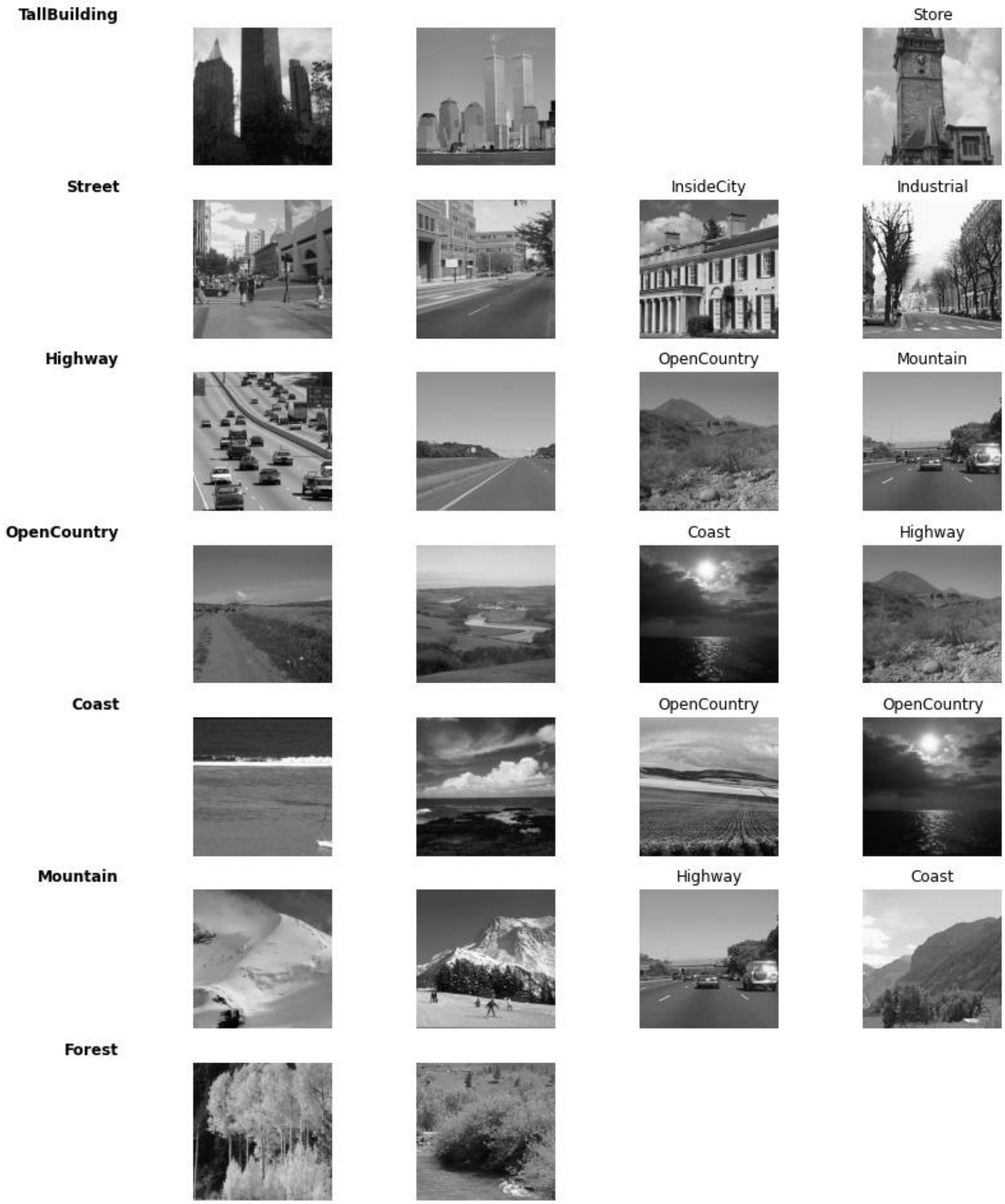
Tiny images representation + nearest neighbor classifier


InsideCity
Industrial
Highway
TallBuilding
InsideCity
Highway
Street
Bedroom
OpenCountry
Highway
Kitchen
Mountain
OpenCountry
Kitchen
Coast
Coast
Kitchen
TallBuilding
Mountain
Suburb
OpenCountry
Forest
Store
OpenCountry

Bag of SIFT representation + nearest neighbor classifier

TallBuilding
LivingRoom
Coast
Street
Store
Store
Highway
Street
OpenCountry
Store
Bedroom
Coast
Kitchen
Kitchen
Mountain
Bedroom
Forest
OpenCountry

Bag of SIFT representation + linear SVM classifier


TallBuilding
Store
Street
InsideCity
Industrial
Highway
OpenCountry
Mountain
OpenCountry
Coast
Highway
Coast
OpenCountry
OpenCountry
Mountain
Highway
Coast
Forest

General CNN:

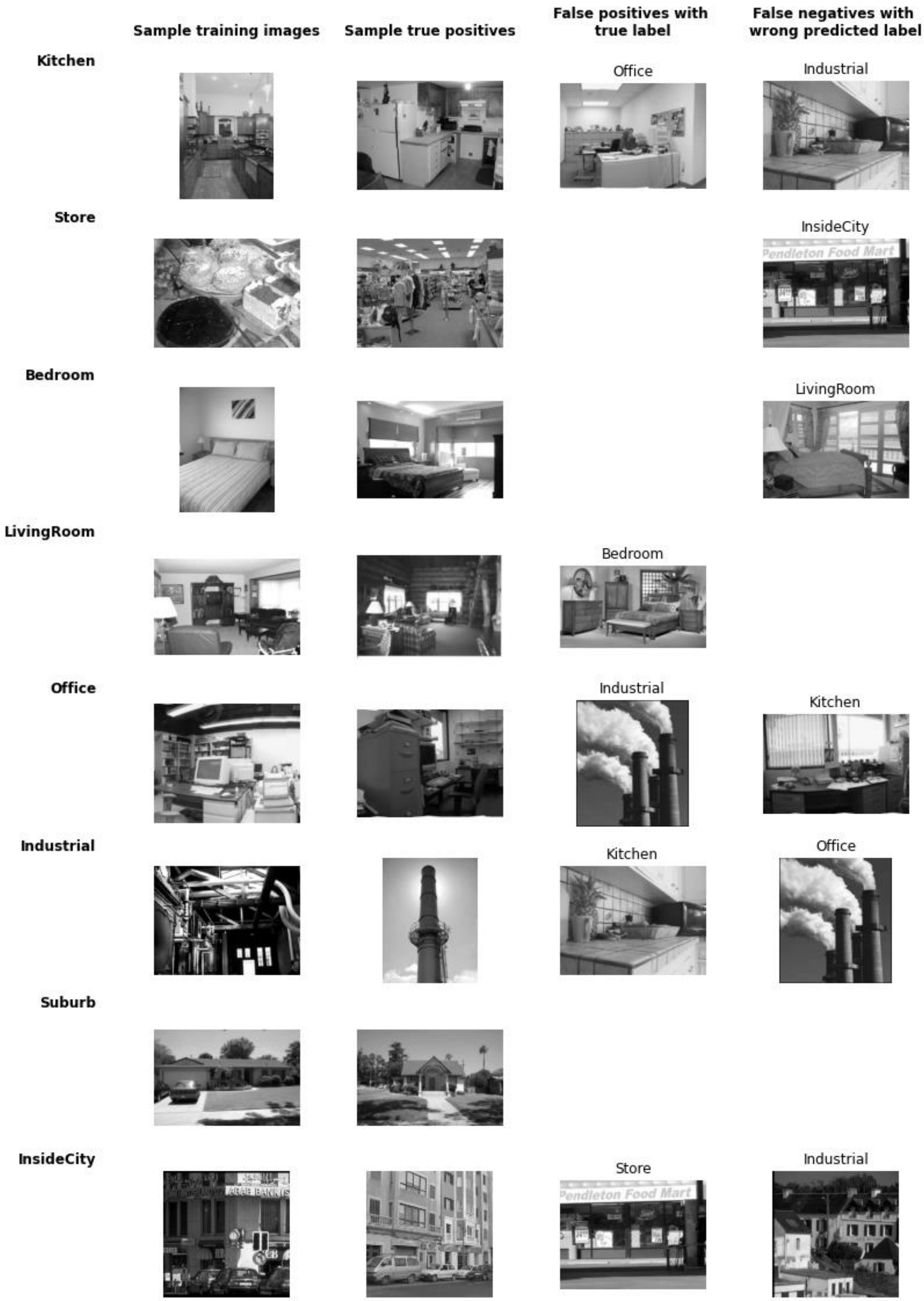

Sample training images
Sample true positives
False positives with true label
False negatives with wrong predicted label
Kitchen
Bedroom
Bedroom
Store
InsideCity
LivingRoom
Bedroom
Kitchen
Kitchen
LivingRoom
Store
Bedroom
Office
Store
LivingRoom
Industrial
InsideCity
Coast
Suburb
InsideCity
Store
LivingRoom


TallBuilding
Industrial
Industrial
Street
Kitchen
Highway
OpenCountry
OpenCountry
Bedroom
Coast
Coast
Industrial
OpenCountry
Mountain
Store
OpenCountry
Forest
Store
OpenCountry

AlexNet + Data Argumentation:


TallBuilding
Bedroom
Forest
Street
Store
Highway
OpenCountry
OpenCountry
InsideCity
Coast
Coast
OpenCountry
Mountain
Mountain
Industrial
OpenCountry
Forest
Store
Mountain

CNN + VGG16 Place 365:




TallBuilding
Industrial
Street
Kitchen
Highway
OpenCountry
TallBuilding
Coast
Coast
OpenCountry
OpenCountry
Mountain
Forest

Discussion

1. Tiny image representations

In task 1, the accuracy is not ideal. The reason is that this method can only compare the pixel-by-pixel similarity between two images. Since this method can not find out the interest features in the images, any changes in the images could lead to misclassification. For example, if you take an image of the same bedroom with totally different angles, the possibility of misclassification could be really high.

2. Bag of SIFT representation

There are many design decisions and free parameters for the bag of SIFT representation (number of clusters, times of running K-means, SIFT parameters, etc.) so performance might vary from 50% to 60% accuracy. At the beginning, we get the accuracy about 0.4. After using some improved methods as shown below, we get the accuracy about 0.5. For different number of clusters, we find that more the number of clusters, more the accuracy we get. We get our best result based on setting number of clusters equals to 295. For times of running K-means, we know that more times of running K-means, more accurate the centers will be. Since K-means algorithm may converge into different local minimum due to different initial seed, the solution for this problem is to run several times of K-means with different initial centers, and then choose the best one, or, use the K-means++ method. Obviously, this solution needs more time to compute clustering centers.

For SIFT parameter, we find that we can set the number of SIFT feature we want. In SIFT function, if you set the number of SIFT feature, the function will return the number of best features to retain. The features are ranked by their scores (measured in SIFT algorithm as the local contrast). From different images, we detect different number of features vary from tens to thousands. However, when we set the number of best features equals to a number, take 100 for example, the result looks poor. We guess that we need more features for better clustering centers. Thus, we remain all the SIFT features for K-means algorithm.

3. Nearest Neighbors

For nearest neighbor classifier, you can find a better K nearest neighbors for a better classification result. we set K vary from 5 to 100 to find a better classification result. For histogram of vector quantization, we need to normalize the histogram for a better result, since each image will get different number of features. Different normalization methods will cause variant result. For example, we had tried

- probability-like normalization (histogram / sum (histogram))
- unit-vector-like normalization (histogram / sqrt (sum (histogram^2)))
- mean normalization ((histogram – mean (histogram)) / std (histogram)).

By experiment, we find that mean normalization produces better result.

Conclusion

In this assignment, we used six different methods to implement the image classifier which classify images into 15 different scene types. With each method, we tried to make improvements to obtain better accuracy. At the final setting of this project, we got the accuracy of 76% using our own CNN and the accuracy of 90.67%. For further improvement of the performance, we can:

- Try out more different models or kernels such as RBF kernel for svm
- Model tuning to find the best parameter setting

Reference

[1] https://github.com/GKalliatakis/Keras-VGG16-places365

[2] https://www.cc.gatech.edu/~hays/compvision2017/proj4/

Work Assignment Plan Between Team Members

0516044 陳思妤	Tiny + NN
0786031 廖俊凱	Bag of Sift + NN / CNN (AlexNet) + Data Argumentation
0856733 黃明翰	Bag of Sift + SVM / CNN (General) / CNN (VGG16_Place_365)