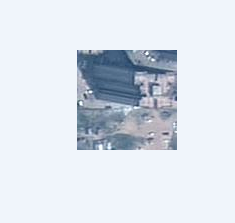
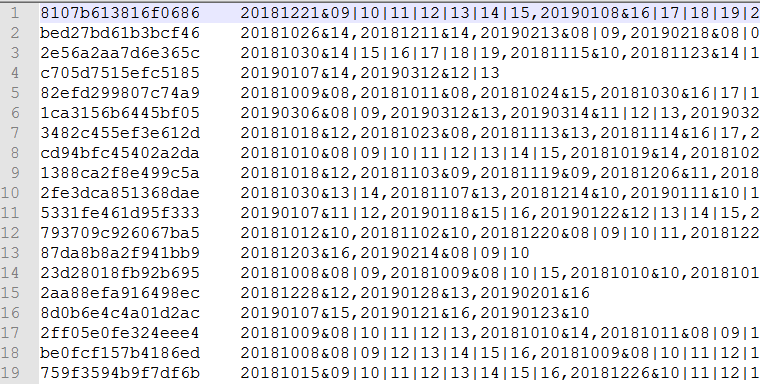
# Urban Region Function Classification

Website: <https://dianshi.baidu.com/competition/30/question>

## The goal:

Class the image (100\*100 pixels)/User visit time data to 9 classes.

|  |  |
| --- | --- |
| CategoryID | Functions of Areas |
| 001 | Residential area |
| 002 | School |
| 003 | Industrial park |
| 004 | Railway station |
| 005 | Airport |
| 006 | Park |
| 007 | Shopping area |
| 008 | Administrative district |
| 009 | Hospital |

### Initial analyses:

1. Neural network must be used (huge data, no hard rule to classify them).
2. Images can only offer limited information (textures are very similar, such as residential area, school, shopping area and others). The User visits information is very important.

### Things need to do:

1. How to extract the user visit information?
2. How to combine the image and visit information together?
3. Data augmentation?

## 5/4/19

Data preparing.

1. Extract the visit information.
2. Total visitor (the number of records) (170000 -1, how to normalize it?)
3. 24/12 hours, the number of visits.
4. ~~12 months, the number of visits. (September,2018 ----February, 2019, no enough data)~~
5. The return times of each visitor.
6. The average stay time of every day.
7. The size of data of the record. (9MB-1k)

## 5/7/19

Normalization: assume all data are following the normal distributing, then estimate the parameters, then normalize them to 0-1.

Mean-std:

1. 1401.671325,3315.93847939731🡪 /1400-2
2. 3.5713300572068145,1.469428439705176🡪/3.57-2
3. 0.3036313519477844,0.4598593726143714🡪/0.30-2

## 5/8/19

1. Do the training sample balance.
2. Prepare the CNN.
3. Train the CNN.

5/10/19

1. Retrain the CNN, added the image augmentation.

5/13/19

The first try results.



Be careful about the space in the txt.

5/14/19

1. Include the raw visit data for the training.
2. Train the network without visit data and generate the confusion matrix.
3. Generate the CNN image features.

The test class spy:

1: 0.2354

2: 0.1833

3: 0.0923

1. Generate the confusion matrix.
2. Check the combination of the image and visit features.

The features from image: most of them are 0 and the range is around(0,8);

1. Try re-normalize the visit features into 0-8.// no obvious improvement.
2. Try add one fcl before merge.
3. Reduce the image feature dimension. // no obvious improvement.
4. Try different combination length.
5. Try the other baseline.

**Baseline( bad image data removed, online image augmentation, 90% training sample)**



**Baseline + weighted loss. 80% training samples**

1.

weights = [0.76145,0.81155,0.91025,0.96605,0.9134,0.862325,0.912075,0.934575,0.928325]



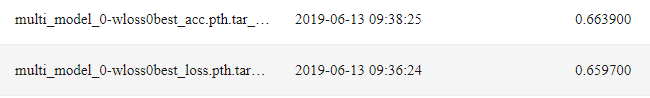
2.

weights = [0.142318,0.180154,0.378273,0.99,0.392032,0.246595,0.386125,0.518915,0.473666]

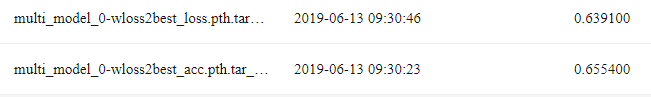


**Baseline + weighted loss (bad image data removed, online image augmentation, 90% training sample).**

1. Weight 1:



1. Weight 2:



6/13/19

Try data augment.

the data can be easily overfitting. To overcome this, we must do data augment. For images, we can do regular changes, but, for the visit data, it is hard to do such changes. However, we think the visit data is independent to the image, which means, the visit data can be attached to any image path in the same function class. Hence, to do the augment, we try random select the visit data to the images that in the same class.

1. Keep the original data load method. But load all data in the memory.
2. Store them by the class.
3. For each image, random find a visit data.
4. Radom change the image.

Note: there is no original partner.



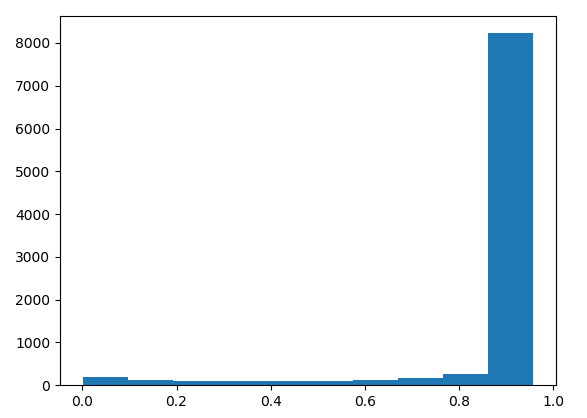
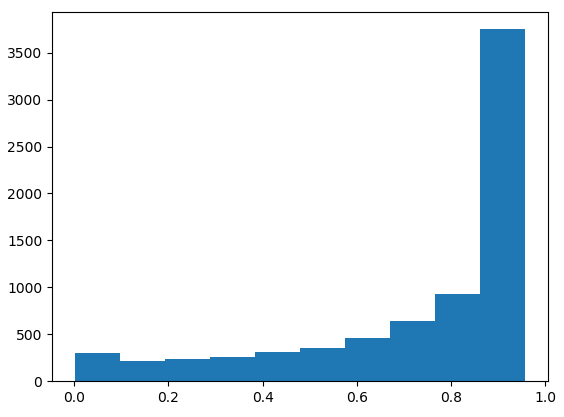
As we can see, there is no improvement and from the training processing, we can see the overfitting is still very bad. this usually means that the visit data didn’t change a lot. The reasons may include: 1. The number is not normalized, the residential and part have the bigger values and the CNN only capture that the how big the number is instead of the change patterns. 2. The visit matrix is too spare to offer enough information. I will check these reasons one by one, but at first, I will try to do the number normalization.

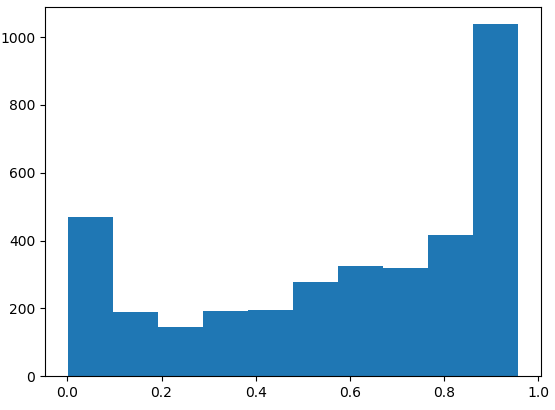
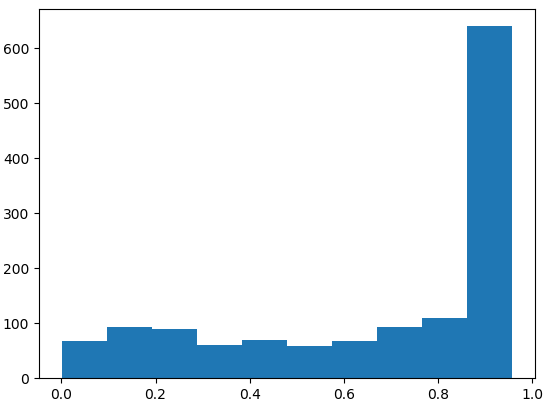
1. Normalize all the number class by class.

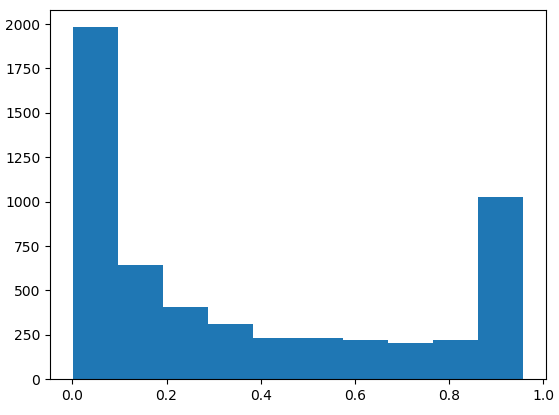
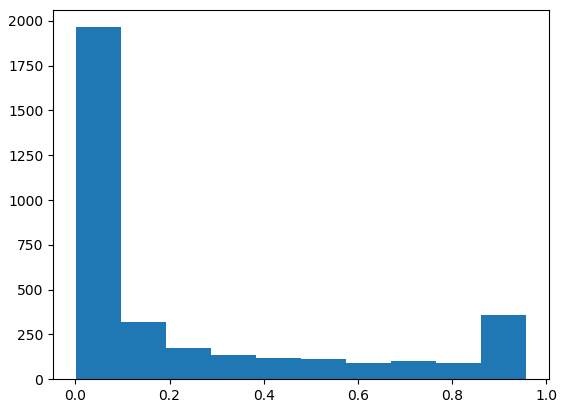
, where is the mean value of feature and is the standard deviation of the

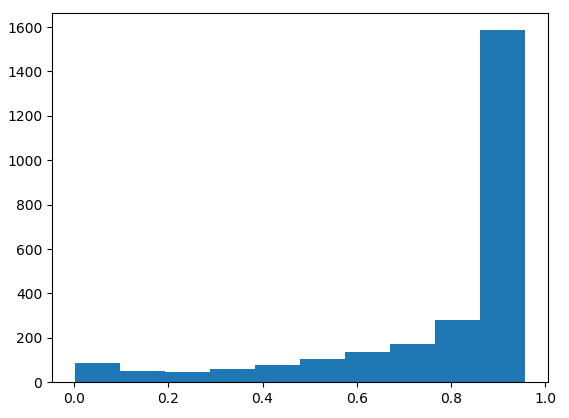
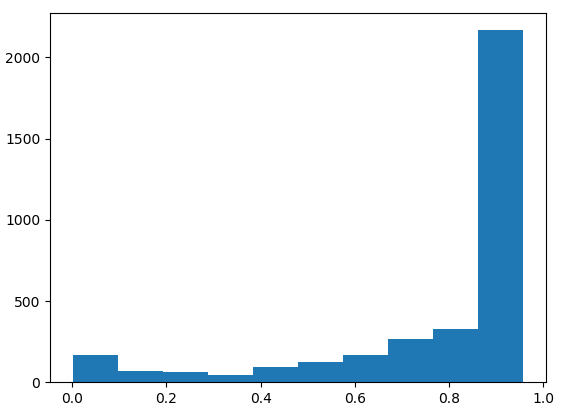
I didn’t see improvement .

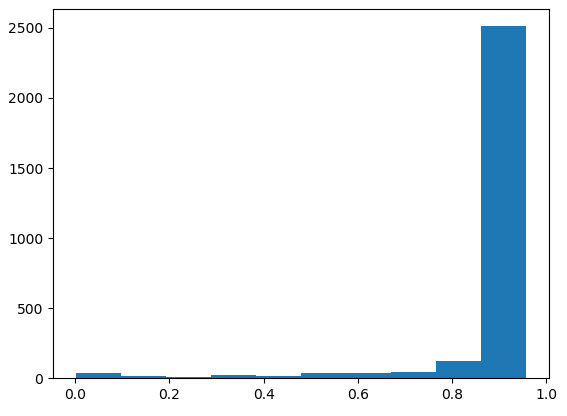
1. Test the distribution of the visit data.



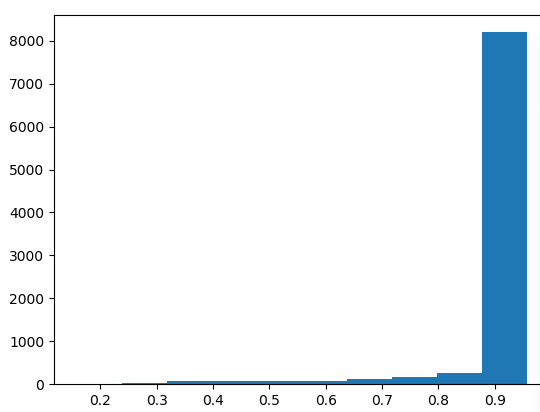
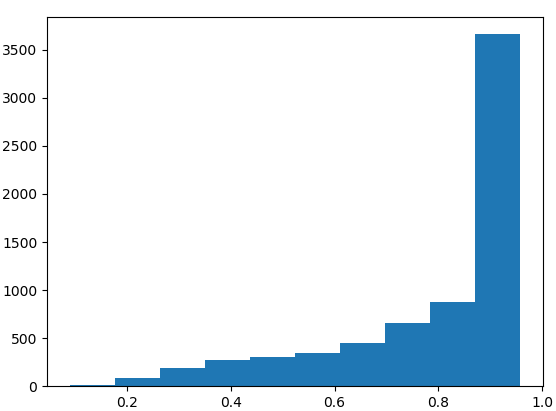


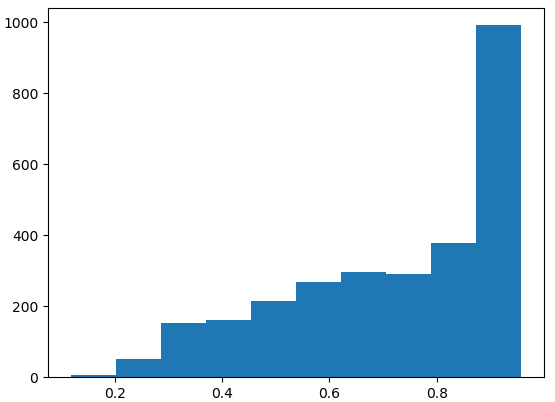
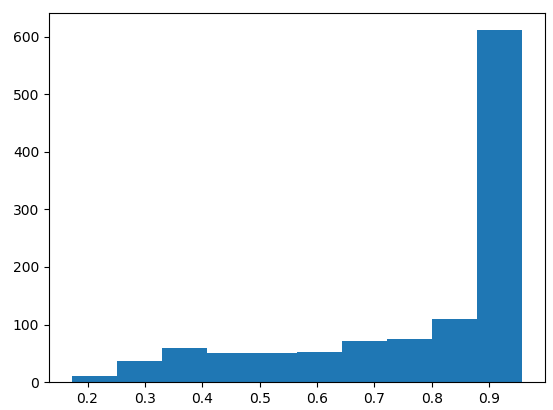


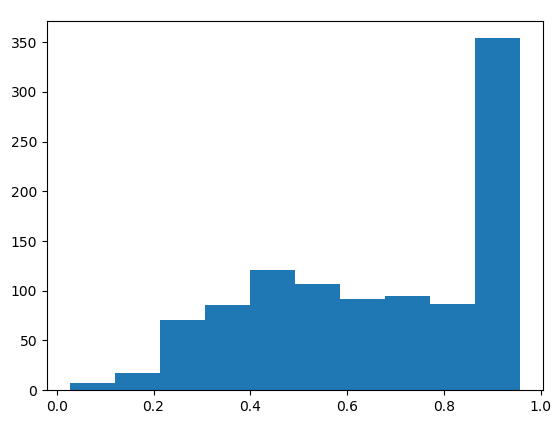
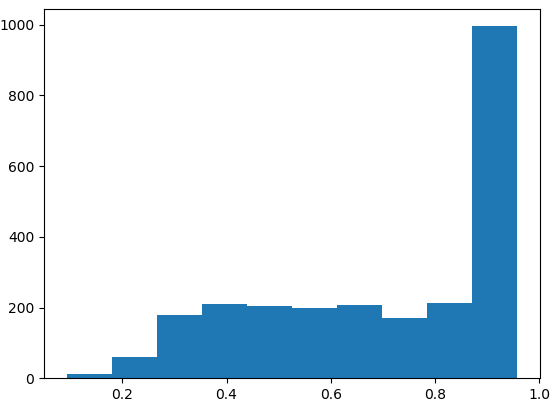
Train with different size.

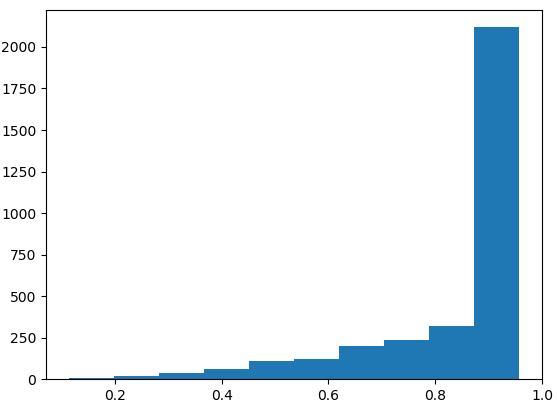
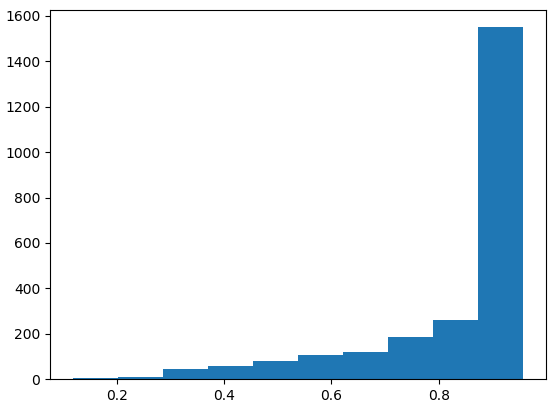
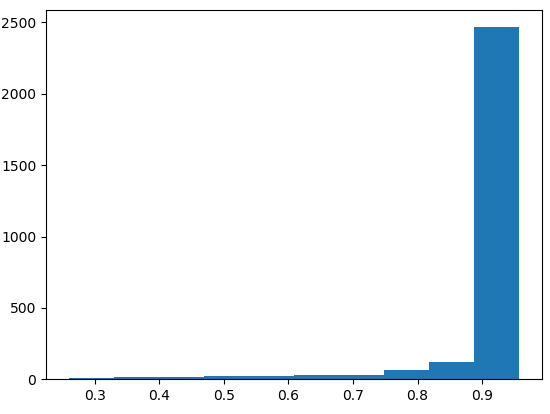
Since the size of visit data vary a lot, we are trying to train the network that only with enough data.

The data >10kB.

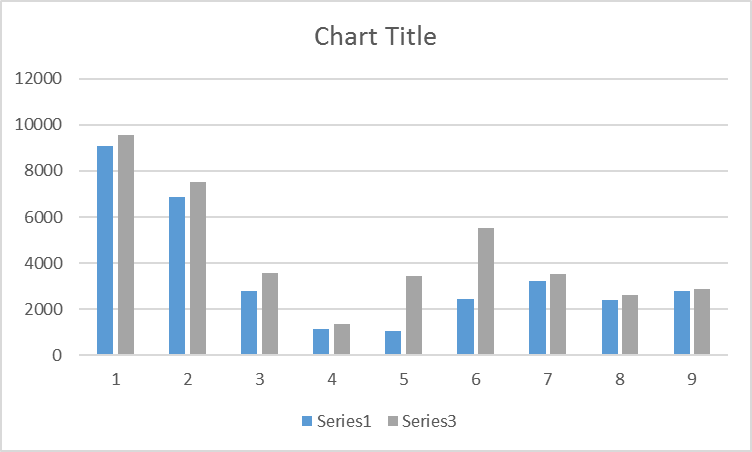
 

10KB vs all



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9091 | 6849 | 2808 | 1125 | 1037 | 2449 | 3229 | 2422 | 2794 |
|  |  |  |  |  |  |  |  |  |
| 9542 | 7538 | 3590 | 1358 | 3464 | 5507 | 3517 | 2617 | 2867 |

6/3/19

1. Check the pytorch model.
2. Output the final layer and last layer to give to HE.
3. Confusion matrix.
4. Use the multimodel as baseline.
5. Get the feature vector form CNN and combine the manual feature vector from visit data to do classification.
6. The manual features are designed by the confusion matrix, that can be used to fix the weakness of the CNN.
7. Probability of the 24\*26\*7 matrix. For test that only has limited visit data, we can try the visit time points probability test.

6/6/19

1. Check the results.

6/26/19

Retrain the network with different val and train set(using random split)

version\_name = 'wloss1-10k-256'

this is the initial training with 10% as validation data.

version\_name = 'wloss1-10k-256-2'

this is the second training with 20% as random validation data. The “best\_acc” weight with

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 5 | 0.453534 | 1.537063 | 0.387189 | 0.432167 | 1.633046 | 0.362446 |

In version\_name = 'wloss1-10k-256' is selected as the pre-trained weight

version\_name = 'wloss1-10k-256-3'

this is the second training with 20% as random validation data. The “best\_acc” weight with

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 5 | 0.453534 | 1.537063 | 0.387189 | 0.432167 | 1.633046 | 0.362446 |

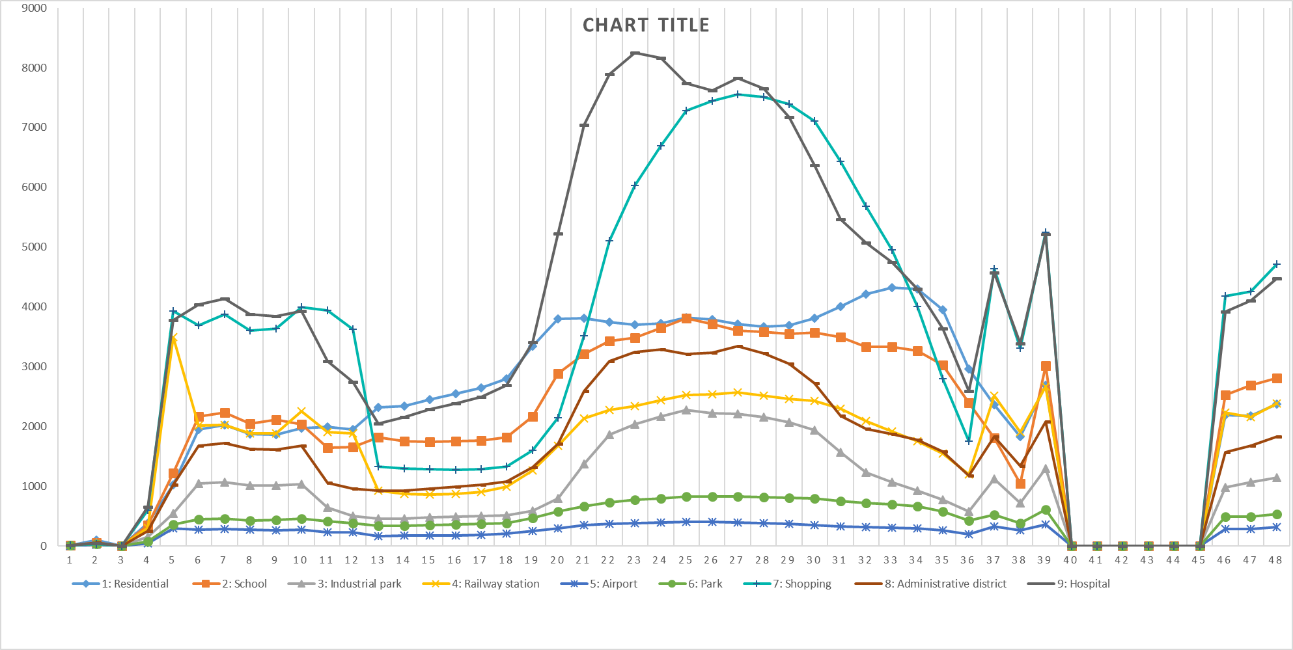
In version\_name = 'wloss1-10k-256-2' is selected as the pre-trained weight

Try to extract the visit information with manual features.

1. Design the manual features.
2. Average num. of days for all visitor with 1 dimension
3. Average num. of points for all visitor with 1 dimension
4. Average daily duration for all visitor with 1 dimension
5. File size with 1 dimension
6. Num. of total visitors with 1 dimension
7. Num. of points in each day of week with 7 dimensions
8. Num. of points in each hour of day with 24 dimensions
9. Num. of points in each month with 12 dimensions
10. Test their intra-unity and inter-otherness.

I need to test how similar the features are in the same class and how differences the features are between different classes.

6/26/19



The differences: 1. Railway station. A lot of visitors but just small num\_visitor\_days and small visitor\_points.

6/27/19

Use a three branch model include the manual features.

Begin training. With augument of image and visit data but not manual features. The weight loss is 1. “three\_branches\_wloss1-256-4”

// see obvious improvement, the validation (20%) accuracy can be 0.72.

6/28/19

1. Retrain with cross-validation datasets.