

CV HW5

Group 18

Introduction

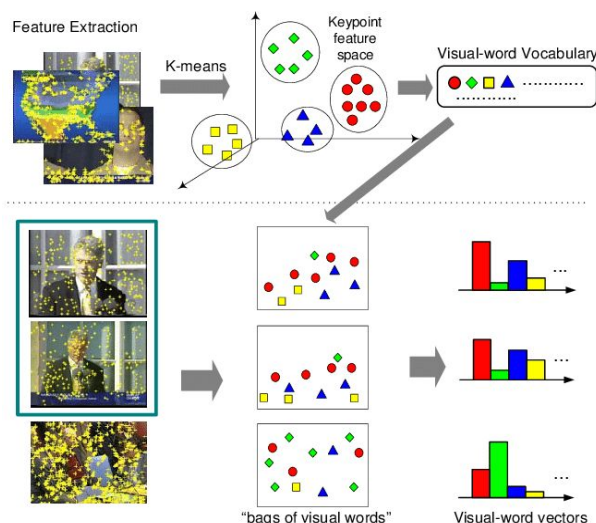
The homework is doing the classification on 15 scenes dataset. The dataset has 15 different scenes, and each scenes has 100 images, which means the total number of data is 1500. Most of the images are gray image and the resolution is about 256 by 256. Our task is to use 3 different methods to validate their performance.

Tiny images representation + nearest neighbor classifier

The first method is called Tiny image representation with KNN classifier. Tiny images means images with 16 by 16 resolution, and each tiny images are a template to their category. Therefore we have to compare each pixel value, and find the closest one as a candidate. KNN is doing the campaign within the selected candidate images. Quite straightforward and easy to implement with a few codes, but the performance too low in reality.

Bag of SIFT representation + nearest neighbor classifier

The second method is similar to the above, but the feature are replaced by SIFT descriptors. The SIFT we called here is actually Dense SIFT. The difference between these two terms is the application. The original SIFT mostly used on Image Comparison. Nevertheless, Dense SIFT usually applied on classification task. Dense SIFT will get a descriptor at every location. On the contrary, original SIFT only get descriptor determined by Lowe's algorithm. That is some of our image may only got few descriptors, which making the image fruitless. With Dense SIFT, we will got 128 descriptors in each image. And with all descriptors, we doing k-means clustering to combine the descriptors to defined number of clusters. Finally doing KNN to make a decision.



Bag of SIFT representation + linear SVM classifier

The third task is quite the same to the second one. But we have to replaced KNN with SVM classifier. SVM classifier is draw a line (hyperplane) to divided two classes points. And the linear hyperplane has to maximize the margin between the closest points.

Implementation procedure

Tiny images representation + nearest neighbor classifier

1. Load images
2. Resize to tiny images (16x16)
3. Normalize or Standardize
4. Distance counting (cdist)
5. KNN
6. Accuracy calculation

Bag of SIFT representation + nearest neighbor classifier

1. Load images
2. Dense SIFT
3. K-means (Bag of visual words)
4. Histogram accumlation on training and testing data (Dense SIFT)
5. Distance calculation. Training feature(1500, dim) and testing feature(150, dim) = (1500,150)
6. KNN
7. Accuracy calculation

Bag of SIFT representation + linear SVM classifier

1. Load images
2. Dense SIFT
3. K-means(Bag of visual words)
4. Histogram accumulation on training and testing data (Dense SIFT)
5. Train SVM linear model
6. Test image classification and accuracy calculation

Bonus: Deep Learning - Convolution Neural Network

1. Hyper-parameters settings
2. Data generator & Data augmentation
3. Model selection and freeze layers
4. Learning rate & Model checkpoint callbacks settings
5. Optimizer settings
6. Training
7. Best weight loading & Accuracy calculation

Experiment result

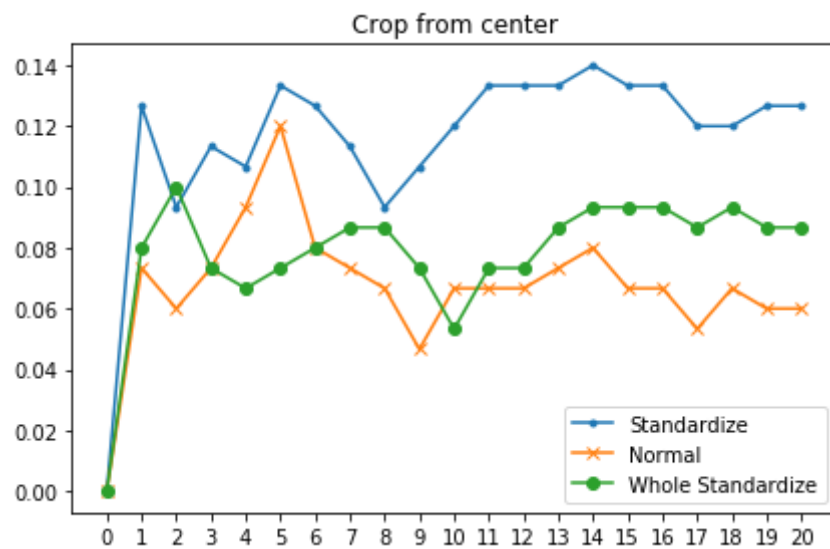
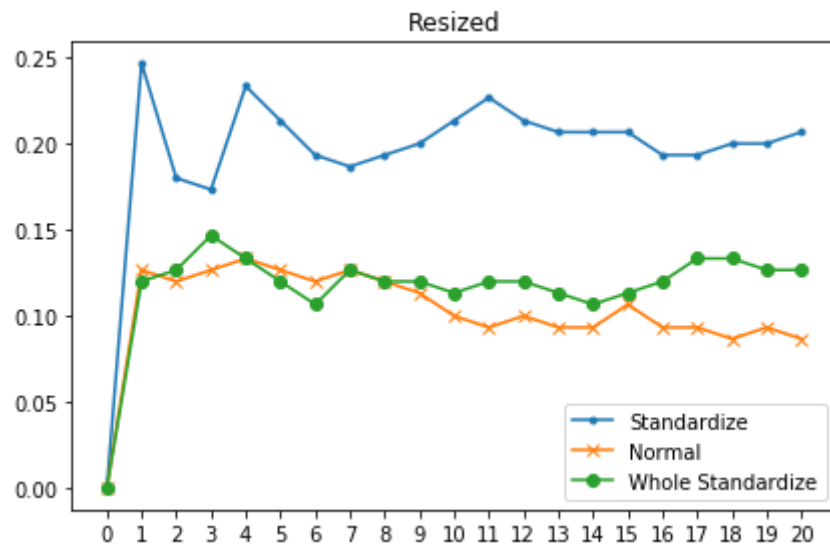
Tiny images representation + nearest neighbor classifier

The best result is **22%** with tiny image representation and kNN.

With k nearest neighbor classifier k = 1.

k	accuracy without nomalization	accuracy with nomalization
1	0.14	0.22
2	0.09	0.16
3	0.13	0.19
4	0.12	0.19
5	0.12	0.19
6	0.13	0.19
7	0.12	0.20
8	0.12	0.21
9	0.10	0.21
10	0.10	0.21
11	0.09	0.19
12	0.09	0.19
13	0.09	0.20
14	0.09	0.21
15	0.09	0.22
16	0.09	0.22
17	0.10	0.22
18	0.09	0.21
19	0.09	0.21
20	0.09	0.22
21	0.09	0.21
22	0.10	0.20
23	0.11	0.19

24	0.11	0.20
25	0.11	0.20



Bag of SIFT representation + nearest neighbor classifier

The best result is **59.3%** with Bag of SIFT representation and kNN.

Best parameters are: dsift step = 10, key point pick in sift = 5, vocabulary size = 300.

# of key point pick in sift	accuracy
5	0.593
8	0.567
10	0.560
15	0.553

with dense sift step = 10, vocabulary size = 300

# of dsift step	accuracy
5	0.553
10	0.593
15	0.480

with key point pick in sift = 5, vocabulary size = 300

vocabulary size	accuracy
100	0.587
300	0.593
500	0.533

with dense sift step = 10, key point pick in sift = 5

k	accuracy
1	0.493
2	0.473
3	0.507
4	0.560
5	0.580

6	0.553
7	0.540
8	0.500
9	0.533
10	0.533
11	0.573
12	0.567
13	0.580
14	0.567
15	0.567
16	0.593
17	0.573
18	0.580
19	0.593
20	0.567
21	0.560
22	0.573
23	0.587
24	0.580
25	0.573

with dense sift step = 10, key point pick in sift = 5, vocabulary size = 300

Bag of SIFT representation + linear SVM classifier

We get the best result - **63.3%** with Bag of SIFT representation and linear SVM classifier.

The parameter in Dense SIFT of training step is:

step : 15 (the horizontal and vertical displacement of each feature center to the next)

size : 8 (the width in pixels of a spatial bin)

The parameter in Dense SIFT of test step is:

step : 10

size : 8

Vocabulary size : 400.

Fast mode : be used to turn on an variant of the descriptor which, while not strictly equivalent, is much faster.

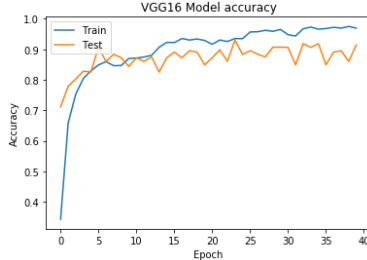
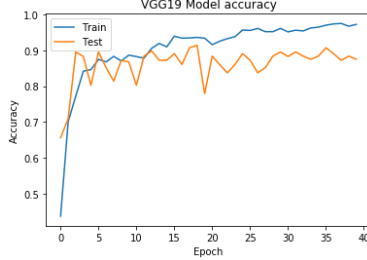
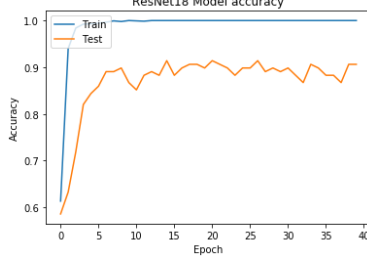
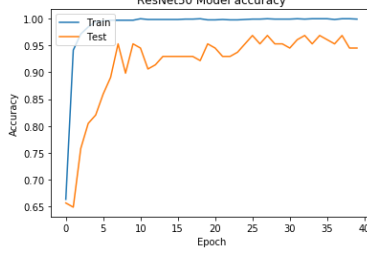
Bonus: Deep Learning - Convolution Neural Network

Our VGG model is testing on 16 and 19 with ImageNet pretrained. Input image size is 256x256. The top is replaced by a global average pooling and 3 fully-connected layers (including the output softmax classifier) with Dropout. We freeze all the original layers with pretrained weights, and only train the finally FC layers we added.

Our ResNet model is testing on 18 and 50 with ImageNet pretrained. The top is replaced by a global average pooling and classification softmax layers. These model without any dropout but batch normalization. We freeze the early 7 layers for the GPU memory efficiency and feature extraction reason.

About data augmentation, we applied horizontal flip and brightness change as well as zoom range. Some of other methods are worth trying, such as shearing, rotation, cropping.

Optimizer	sgd	loss	cross entropy	strategy	Cosine aneling
Learning rate	1e-3	Momentum	0.9	Weight decay	1e-3
Epochs	40	Batch size	64	Input size	256x256

Model	Test Accuracy	
VGG16	0.91	 <p>VGG16 Model accuracy plot showing Train and Test accuracy over 40 epochs. The Train accuracy (blue line) starts at ~0.35 and rises to ~0.95. The Test accuracy (orange line) starts at ~0.75 and rises to ~0.91.</p>
VGG19	0.87	 <p>VGG19 Model accuracy plot showing Train and Test accuracy over 40 epochs. The Train accuracy (blue line) starts at ~0.45 and rises to ~0.95. The Test accuracy (orange line) starts at ~0.65 and rises to ~0.87.</p>
ResNet18	0.89	 <p>ResNet18 Model accuracy plot showing Train and Test accuracy over 40 epochs. The Train accuracy (blue line) starts at ~0.6 and rises to ~1.0. The Test accuracy (orange line) starts at ~0.6 and rises to ~0.89.</p>
ResNet50	0.97	 <p>ResNet50 Model accuracy plot showing Train and Test accuracy over 40 epochs. The Train accuracy (blue line) starts at ~0.65 and rises to ~1.0. The Test accuracy (orange line) starts at ~0.65 and rises to ~0.97.</p>

Discussion

Normalization?

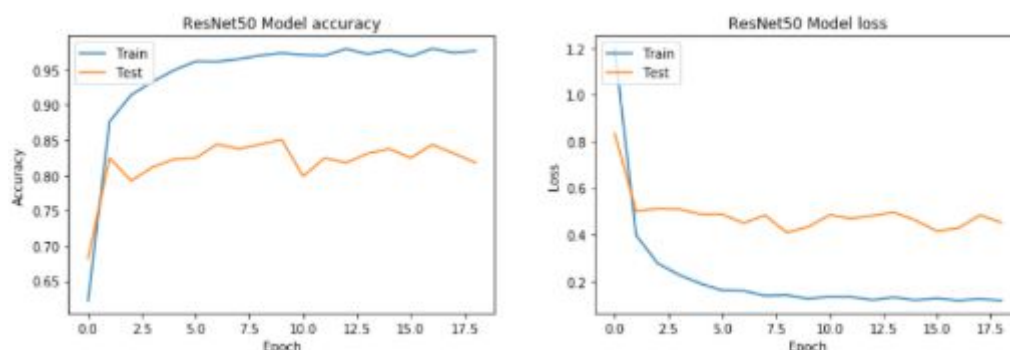
In the first part of the homework, after we normalize the tiny image features, the result improved from 14% to 22%. But in the second and third part, the normalization for the bag of SIFT features actually leads to a much poor accuracy. We guess that it is because the features in the SIFT means orientations in the image, so if we normalize it, the descriptor features will not be that clear.

Randomly pick features or use them all?

In the second part of our homework, when building the vocabulary of the features, we tried to use two methods, first method is randomly pick x features in the image, second method is build the vocabulary using all the features found in images. The results is surprisingly very similar. So we choose randomly pick x features to reduce our execution time.

Keras Batch Normalization layer is broken?

In the Keras with Tensorflow 1.x backend, the BN layer training has some problem when we freeze them. In the previous version, the training parameters (gamma, beta) in the BN are different when using on training and inference. It's the reason why when doing transfer learning the performance of models with BN (e.g. ResNet, DenseNet) are worse than the old ones (VGG). Thankfully this problem has been solved in Tensorflow 2. In our experiment, the ResNet performance is about 77% accuracy in Tensorflow 1. Yet, in Tensorflow 2, the accuracy boosted to 93%.



Conclusion

We got accuracy 22% from the tiny image representation & kNN, and 59.3 from the bag of SIFT & kNN, and 63.3% from the bag of SIFT & linear SVM, and 97% from the deep learning.

Actually there are still lots of hyperparameters we can adjust in the homework. Like how to choose the feature to train or those parameters in the dsift and svm.

Deep learning model are the hottest techniques recently. The result also out performs the hand craft features. And there are so many different models that we can try by ourselves.

Work assignment plan between team members

We finish this homework together.

Additional references

1. [Dense sift vs sift?](#)
2. [图像检索：BoW图像检索原理与实战](#)
3. <https://github.com/menpo/cyvlfeat/blob/master/cyvlfeat/sift/dsift.py>
4. [Keras的BN你真的冻结对了吗](#)