CS5340 Group 12: Hand Gesture Recognition

Qi Miao **A0159327X**
Yifan Lu **A0159305E**
Longwei Zhu **A0148601L**

Abstract

To classify hand images into predefined gesture categories, we use two kinds of methods: the first is using SIFT, K-Means, SVM and the second is the deep learning way. The first method has low accuracy in classifying images of the target datasets, but still shows potential in specialized circumstances such as classifying images of different objects. The second method shows much higher accuracy, which is 77.0% in the target datasets. In this report, we illustrate the procedure and results of both two ways explicitly.

1 Introduction

The task of the project is to classify hand images into predefined gesture categories. The hand image dataset has 30,003 images in total and 28 categories (We combined some similar categories into one such as 2 and V).

To solve the problem, we implement two kinds of ways: the first is using SIFT, K-Means, SVM and the second is the deep learning way.

The experiment result shows that the first method does not work well in this dataset. But we think it still has optimistic potential in classifying images of different objects.

The second method gets relatively higher accuracy of 77.0% by using AlexNet. We also tried some different models such as VGG and GoogLeNet, but they are not the main focus in our work.

2 Related Work & Background

2.1 SIFT & K-Means & SVM

SIFT. Scale-invariant feature transform, also called SIFT, is used to describe local features in image. Its widely used in computer version applications such as recognition, robotic mapping, video tracking. SIFT uses Gaussian filter to blur the image, then detects the interest point to construct the descriptor.

K-means. Its a vector quantization algorithm which comes from signal processing at the first time, its widely used in cluster analysis and data mining now. This algorithm simply separate the data into k clusters by the nearest mean.

SVM. Support vector machine is a supervised learning model that can separate data into two classes by a obvious gap, then map the new data to the same space and decides which class the data is in based on the gap sides.

2.2 Deep Learning Methods

In this part, we will introduce the main features of the three models we use in this task. Our main focus is on AlexNet, so we will discuss more on it and talk about the other two models just briefly.

2.2.1 AlexNet

AlexNet was first introduced in the ImageNet Competition in 2012, designed by Alex Krizhevsky. Compared with LeNet and other Nets introduced before, it has several innovations which lead to a better performance. Its main structure is 8 layers, including 5 convolutional layers and 3 full-connected layers. The output is 1000 softmax and the optimization target is to maximize the average multinomial logistic regression.

Following the first and second convolutional layers are two local response normalization layers. This method uses the neighbor pixels to perform the normalization which means a kind of "brightness normalization", aiming to aid generalization.

Activation layers follow each convolutional layer and full-connected layer. In AlexNet it applies the ReLU nonlinearity $f(x) = max(0, x)$, which trains faster and models a neuron's output ?? more easily than the standard way $f(x) = tanh(x)$ and $f(x) = (1 + e^{-x})^{-1}$.

Max pooling layers follow the first and second normalization layers and the fifth convolutional layers. Overlapping Pooling is adopted in AlexNet to avoid overfitting. This means that the neighborhoods summarized by adjacent pooling units will overlap, making over fit more difficult.

The most innovative part in AlexNet is the dropout layer, considering the situation of simulating the running status of human brain. The dropout operation is performed following the last two full-connected layers. The method is realized by set to 0 the output of each hidden neuron with probability 0.5, and the "dropped out" neurons won't participate in back propagation. By use of dropout layers, we avoid overfitting and faster the calculation, which makes great contributions in deep learning.

In general, AlexNet is a breakthrough in deep learning method and we certainly considered to try this method for our classification purpose.

2.2.2 VGG

VGG was first introduced in year 2014. It is much deeper than AlexNet with 19 layers. It uses very small $3 * 3$ filters in all convolutional layers instead and doesn't adopt local response normalization method.

2.2.3 GoogLeNet

GoogLeNet was presented in 2014 as well. And it is even deeper with 22 layers. The innovative part of GoogLeNet is that it introduces a concept called Inception which is frequently used in the future net structure.

Inception is an architecture with two motivations and high level considerations. When the net structure is in bigger size it will need more parameters which means more possible to over fit, we could consider to replace the fully-connected layers by sparse ones. While it is not efficiency when calculating non-uniform sparse data structures, we could cluster sparse matrices into relative dense sub-matrices.

The architectural details of Inception is mainly 4 points:

1. To find the optimal local construction and repeat it spatially.

2. Units from an earlier layer will be grouped into filter banks.

3. Filer sizes are restricted to $1 * 1$, $3 * 3$ and $5 * 5$ to avoid patch-alignment issues.

4. $1 * 1$ filters are used as dimension reduction modules, increasing the net depth and width.

3 Implementation

3.1 SIFT & K-Means & SVM

We use Matlab 2015b on Windows 8.1 to implement our solution. Our first attempt is a classic combination of recognition, classification and learning algorithms. First is the SIFT. We use SIFT

to extract the features from the images. The feature is called SIFT descriptor, which is a vector with 128 dimensions.

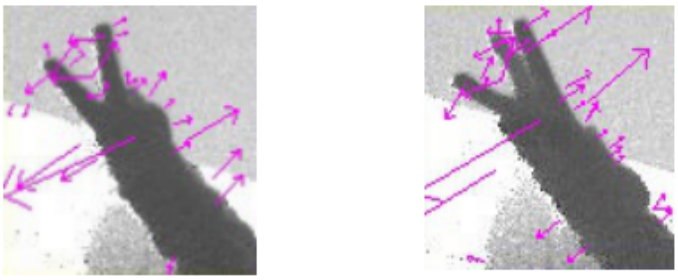


Figure 1: SIFT features

After extracting the descriptors, we use K-means cluster to classify them. K-means is a simple method to cluster data into k groups in which each data can cluster to the nearest mean.

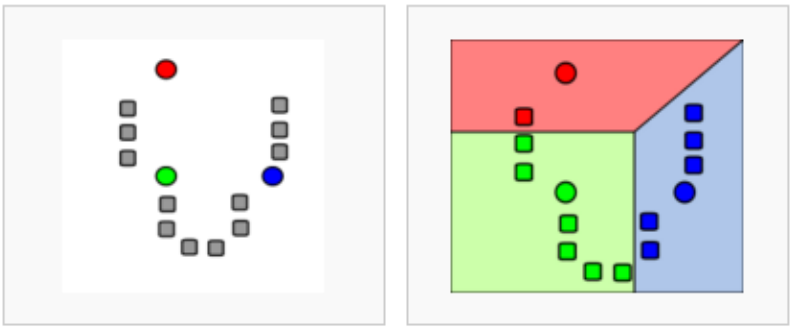


Figure 2: K-means cluster

In this step, SIFT descriptors are clustered into 28 patches, which the number of hand gestures we want to recognize. All the patches in an image are mapped to a vocabulary. Finally ,the clustering process and the image can be represented by the histogram of the vocabulary.

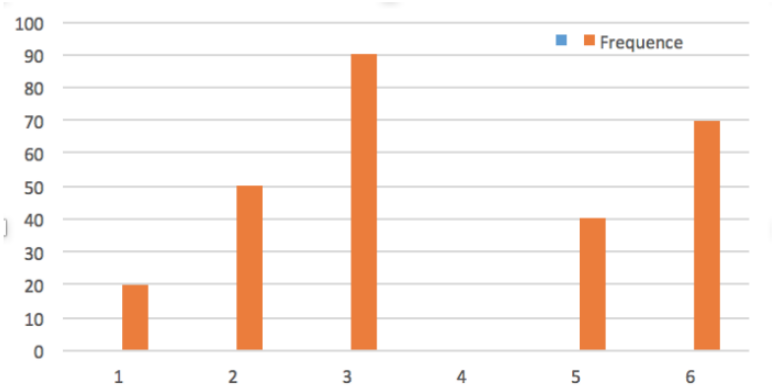


Figure 3: Histogram of vocabulary

As soon as we have the vocabulary, we consider to use one-to-all SVM to train a classifier, which one-to-all means every categories has a classifier to determine wither the input image is belong to it. 28 classifiers will test on all the histograms, and the image will be classified into the most confidently positive category.

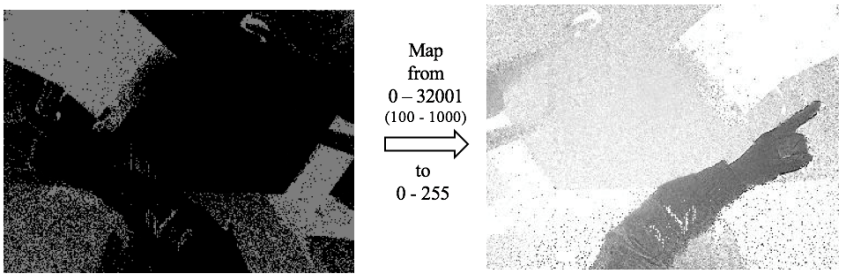
3.2 Deep Learning Methods

3.2.1 Images-Preprocessing

Before using images to train our models, we do some image-preprocessing work in order to raise accuracy and reduce computation complexity. Firstly, we convert the 1-channel, 16-bit depth image to 3-channel, 8-bit RGB image. The RGB channels hold the same value. The equation we use to is as following:

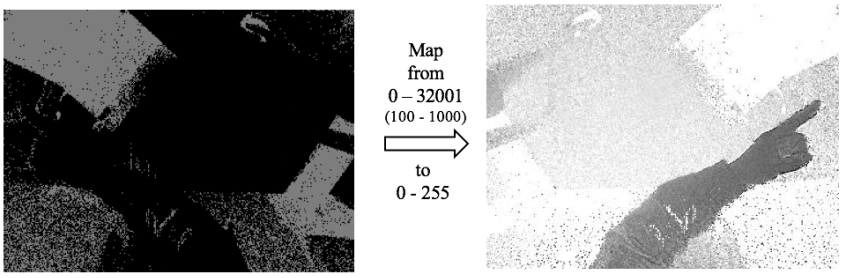
$$convertedValue = 255 * \frac{value - min}{max - min}$$

Here, the interval (min, max) represents the common value interval that the images of datasets lie in. In our experiment, we find the depth value of most pixels are between 100 and 1000, so we set min = 100, and max = 1000. Then the depth value of (100, 1000) will be mapped to (0, 255). The value out of range will be mapped to 0 or 255.



320 * 240 * 1 Channel 320 * 240 * 3 Channel

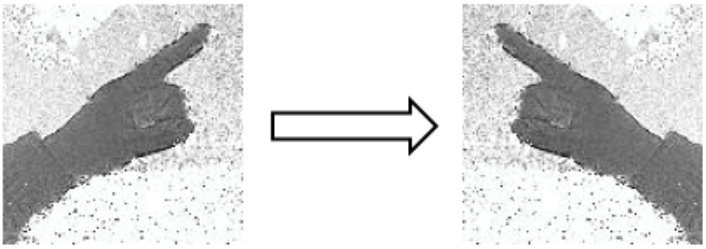
And then, to reduce computation complexity, we crop the image size according to the txt file in the data image. The four value in the txt maps after computation maps the (left, up, right, down) boundary of the cropped image.



320 * 240 * 1 Channel 320 * 240 * 3 Channel

At last, we use data augmentation, which can be seen as a kind of trick, to upgrade the size of training dataset and raise accuracy. The way we do is to generate the reflection of the image simply and add it to the training dataset.

Finally, we get all the image-preprocessing done. The whole process is implemented via Python scripts. The details can be seen in the source code of this part.



Use reflection to do data augmentation

3.2.2 AlexNet

To train and test the AlexNet model, generally we use the open source deep learning framework—-Caffe.

To make our datasets fit the Caffe framework, firstly, we divide the dataset into training set (with 23987 images) and test set (with 6016). The ratio of training to test is 4:1 and the selection is random.

Secondly, we generate the file list for both training and test set by bash scripts. The list stores the image name, url and label.

Thirdly, we convert all the image files into lmdb by using bash scripts as well. We convert the image into equal size of 100 * 100 or 227 * 227 for simplicity of computation and to meet the model input requirements. This may make some images lose some information because of slight distortion. However, we think it will have few effects on the classification results due that the convolutional layers extract and compress the hidden features of images.

The model we used is exactly as the one we shown in Section 2.2. So for the configuration of the file .prototxt and .solution, we do not change the function or the order of the layers. We adjust the parameters such as crop-size, batch size, output category number, monument, test interval, iteration times, learning rate and so on.

After all these preparation work is finished, we begin the training and get the test result. When the result turns out the trend of overfitting, we stop and adjust the parameter such as the one in drop out layer again until the result relatively satisfies our expectations.

For AlexNet, we use three different training set to train our models. The first is the cropped size images without data augmentation. The second is the cropped size images with data augmentation. The third is the full size images. The result of comparison can be seen in Experiment Results section.

3.2.3 VGG & GoogLeNet

For VGG and GooLeNet, we use the dataset of full size images. The preparation work is nearly the same as AlexNet. We do not devote much time to adjusting the layers or parameters. The purpose of this part work is to have trial to apply these models to our datasets and see the effects and difference from AlexNet from a general point of view.

4 Experiment Results

4.1 SIFT & K-Means & SVM

In the first experiment, we actually trained 15 categories of the hand gestures. The final result of 10 times records is 18.49% which is not a good result.

Table 1: Final result of SIFT & K-means & SVM

Times	1	2	3	4	5	6	7	8	9	10
Results	0.182	0.187	0.179	0.181	0.182	0.190	0.185	0.181	0.198	0.184

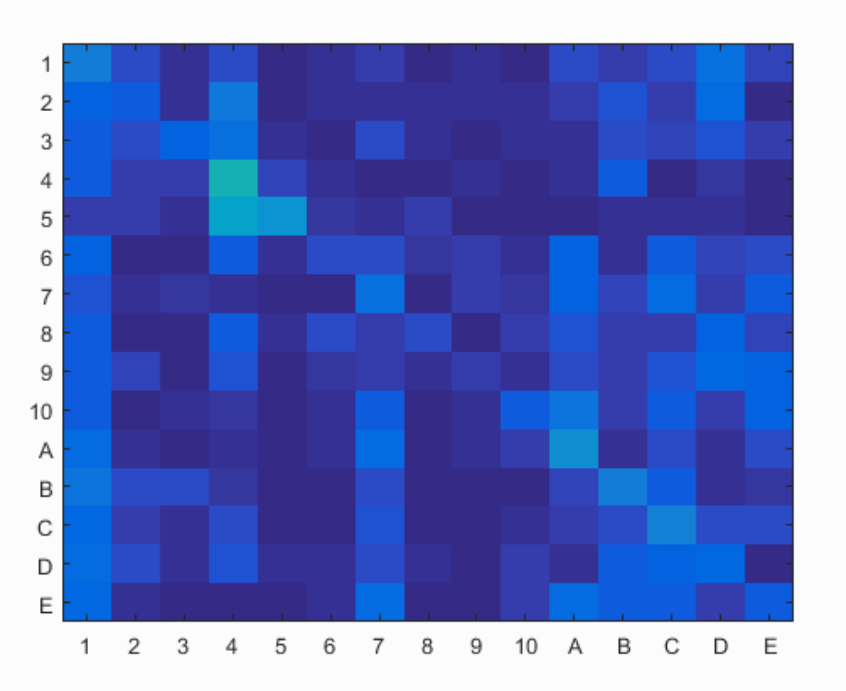


Figure 4: The accuracy distribution graph

We use graph to describe the distribution of the accuracy, which the more the color green, the higher rates we get. In this graph, we can see barely green but almost all blue, that means the accuracy is low.

Although the result on the given dataset is not good, we still hold positive opinion on this method. To prove its efficiency on classifying different objects, we created another dataset which consist of houses, rooms, highways and so on, and then train and test on this new dataset.

Figure 5: Examples of Dataset 2

The final result is 66.88% after 10 times records, that means this model can work well in some situation. For we analysis, we believe it because the hand gestures graphs are so familiar in most part, for the traditional algorithm, its hard to extract the feature that have obviously different.

Table 2: Final result of SIFT & K-means & SVM

Times	1	2	3	4	5	6	7	8	9	10
Results	0.677	0.653	0.643	0.681	0.665	0.690	0.674	0.680	0.654	0.671

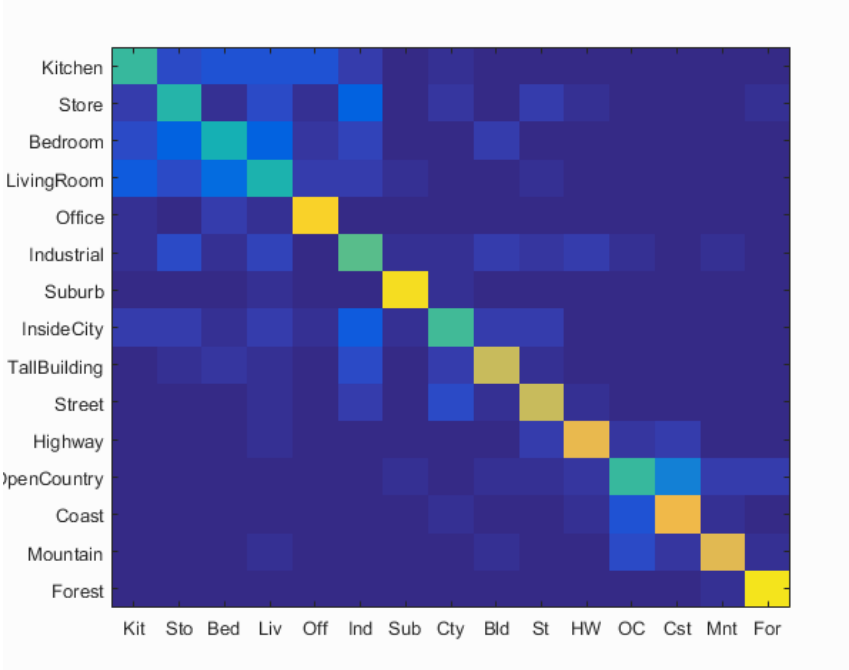
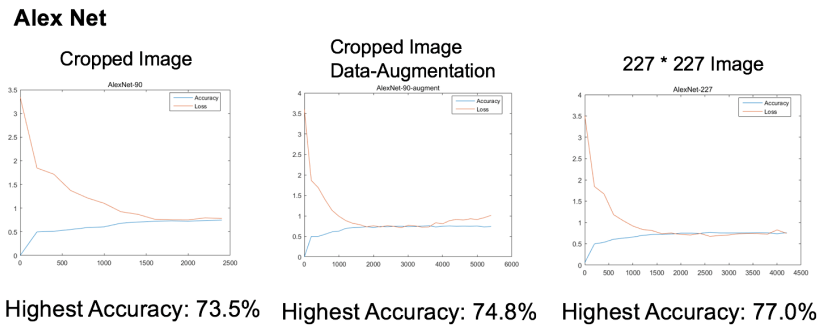


Figure 6: The accuracy distribution graph in dataset 2

4.2 Deep Learning Methods

Generally, we train our models on the Ubuntu 16.04 server, which is equipped with GTX-Titan. We use cuda and cudnn to accelerate the computation process. The time of training for each model varies from 10 minutes to 2 hours, which depend on whether the images are cropped and the complexity of the models. We iterate each models about 3000 times where they have obviously converged.

4.2.1 Results on AlexNet with different methods of image-prepocessing

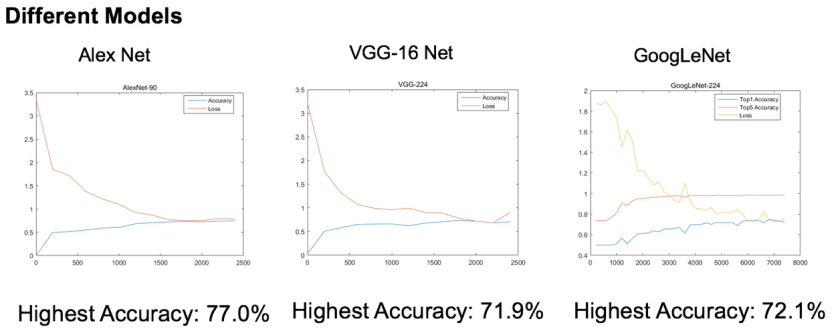


As shown in the figure, the highest accuracy of AlexNet with different training datasets is 77.0%, 74.8% and 77.0%. The comparison between them is interpretable and satisfy our common sense.

For the crop size images, the data augmentation successfully improved the accuracy by roughly 1.5%. Because the data augmentation operation doubles the size of training image and make the features of each hand gesture captured with higher probability, the result improves and it is within our expectations.

For the crop size images and full size images, the full size dataset gets higher accuracy, which is 77% and it is the highest result of all our experiments. This is because the crop size images lose some information and features during the cropping procedure. And another reason is, the coordinates given by the txt file may not be accurate and resulting in some distortion that can not be ignored. In other words, the cropping operation makes our dataset unclean. If we improve the algorithm in deciding the cropping coordinates and do better work in capturing the main area of hand, we believe that the difference of the two training datasets will be smaller.

4.2.2 Results on the three models



As shown in the figure, the highest accuracies of AlexNet, VGG and GooLeNet are 77.0%, 71.9% and 72.1% respectively. The result is somehow out of expectation. We expect that VGG and GooLeNet performs better than AlexNet is they are more optimized and have more powerful ability in computation and classification. However, they provide relatively lower accuracy compared to AlexNet.

The possible reasons we analyzed can be as following: The dataset we provide is not large enough. Both VGG and GooLeNet has more layers than AlexNet. So small dataset will result in overfitting problem easily, which will definitely reduce the accuracy of final result.

Since we do not devote much time in the two models, the configuration of some parameters and layer may be not suitable for our datasets. There may exist some unnecessary layers or the order of layers need to adjusted. As we get deeper understanding of the two models and adjust them respectively, the result will improve.

What needs to mention is the top-5 accuracy of GooLeNet is around 100%, which may indicate that some categories of gestures are closely similar to each other. If we combine them, we can get higher accuracy further.

5 Conclusion

In this project, our group implement two methods to classify hand images into predefined gesture categories: using SIFT, K-Means, SVM and using deep learning way.

The experiment result shows that the first method does not work well in this dataset. But we think it still has optimistic potential in classifying images of different objects.

The second method gets relatively higher accuracy of 77.0% by using AlexNet. We also tried some different models such as VGG and GoogLeNet, but the accuracies of these two methods are not ideal.

In the future, we get long way to learn and improve. With better understanding of classification and clustering algorithms and deep learning knowledge especially in models, and with more

cleaned data and better image-preprocessing methods, we optimistically believe that we can get better classification accuracy.

References

[1] Lowe D G. Distinctive image features from scale-invariant keypoints[J]. International journal of computer vision, 2004, 60(2): 91-110.

[2] Cortes C, Vapnik V. Support-vector networks[J]. Machine learning, 1995, 20(3): 273-297.

[3] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.

[4] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[5] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015: 1-9.