10-701 Final Project Report

Yuan Liu

Abstract

This project consists of three tasks. The major machine learning technique utilized in the first two phases is Convolutional Neural Networks (CNNs). Also, various image preprocessing approaches are implemented to help CNN to find the features and converge faster. In the second phase, strong data augmentation methods are used to generate enough randomized images to feed into CNN and 99.2% accuracy on test set is achieved.

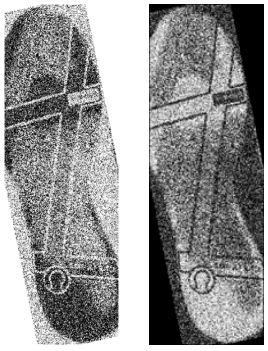
1 Left/Right Footprint Detection

1.1 Data processing

Since the images have been added with heavy noise and then rotated, those images should be rotated back and the noise should be removed as much as possible.

1.1.1 Color Inversion

Since CNN will do convolution on the images, it should be assumed that the pixels which contains less information should have values closer to 0. Yet the original images have white backgrounds which are not suitable to feed into CNN. So I take a grayscale color inversion on images.



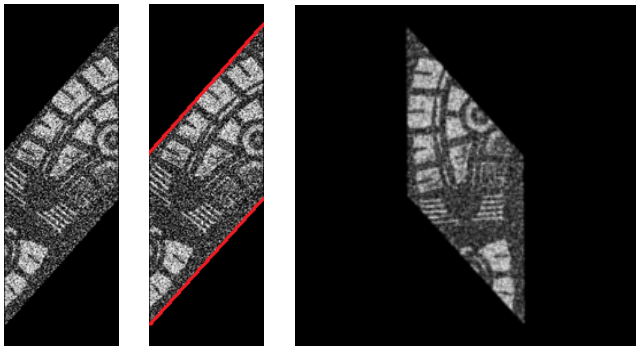
(a) (b)

Figure 1: Color inversion of images. (a) Image before inversion. (b) Image after inversion.

1.1.2 Rotation

Look at a sample in the train set, the image appears to be randomly rotated. Due to the lack of ability in fully understanding rotation invariance, the most straightforward way for CNN to decrease its loss is "learning by rote" [1]. However, this approach will consume a huge amount of time thus not feasible in a limited time and computing resources.
Although convolutional neural networks have invariance property to rotation due to its max pooling layers. It is not quite robust to large angle rotation. So, Canny edges detection and Hough transform are used to determine the noise boundary. Then the images is extended to square and rotated back as figure 5 shows.

10701 Class Project (2016).

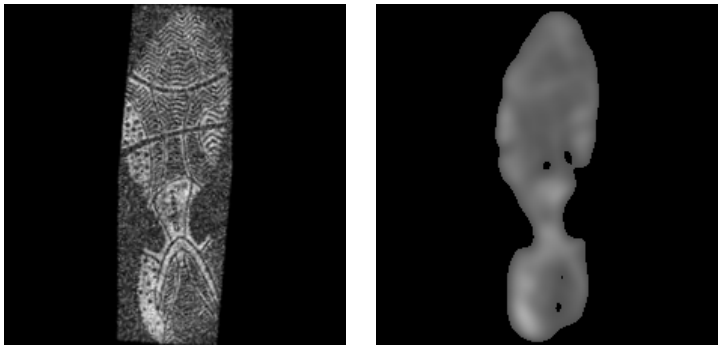


(a) (b) (c)

Figure 2: Rotation of images. (a) Image before rotation. (b) The noise boundary denoted by red lines. (c) Image after padding to square and rotation.

1.1.3 Noise removal

The most important feature should be extracted from the edge of the shoes, thus in order to extract the edge of the shoes, the background noise should be reduced.
My approach is to first use Gaussian filter to blur the image in order to decrease the grayscale of the noise. Then use find the mean value of all the non-zero pixels, denote the mean value as m. Finally, set all pixels below $k \cdot m$ to zero. In task 1, k is set to 1.15.



(a) (b)

Figure 3: Noise removal of images. (a) Image before noise removal. (b) Image after noise removal

After being processed through the above methods, the images are suitable for learning.

1.2 Machine learning techniques involved

SVM, PCA,K-means Clustering, CNN, data augmentation

1.3 How and why I choose these techniques and result

The initial attempt is throw the unprocessed images (but resized to 50x50) directly into a CNN with 5 convolutional layers. During training phase, the converge speed is pretty slow (It takes over 50 epochs to converge to 90%+ accuracy on training set). The validation accuracy is not correctly estimated since the label order is not consistent and I didn't notice that at that time.

For problem 1, since the problem is not as complex as problem 2 and there are only two possible labels, that is, 'left' and 'right', a SVM-based approach is first considered. So I switched to directly apply kerneized SVM to classify the images. To my surprise, the RBF-SVM in a relatively large dataset is quite slow. And I dropped this approach since it seemed to run infinitely.

Then I switched to trying to PCA the images, and project them onto the first 225 principle components. and still use a RBF SVM to classfify them. Only 81% accuracy on validation set is achieved.
Since the images are rotated by an angle, I tried K-means to cluster them into several group of images. But the different angle of images are not correctly clustered together.

I utilized several image preprocessing and augmentation techniques which I have demonstrated in data processing section to reduce the noise and keep the contour of the footwear prints. Then I used the same CNN as my initial attempt and achieved 92% accuracy. In this attempt, the images were resized to 50x50, and the overall training time is about 30 seconds. It converges in 3 epochs to over 90% accuracy on mini-batch accuracy. Quite fast and achieved a relatively high accuracy.

Although this approach is very fast and the accuracy is also enough, I wonder that if I use a classic CNN like AlexNet [2] whether the overall accuracy will increase. So I constructed a neural network based on AlexNet, but the input layer is modified to accept only one channel instead of three channels as AlexNet. Using the same preprocessing techniques as the former step but resized the images to 227x227, the result is pretty good. 96.1% on validation set and 96.5% on test set.

1.4 Parameter setting

25x1 Layer array with layers:

1	Image Input	227x227x1 images with 'zerocenter' normalization
2	Convolution	96 11x11x3 convolutions with stride [4 4] and padding [0 0]
3	ReLU	
4	Cross Channel Normalization	cross channel normalization with 5 channels per element
5	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0]
6	Convolution	256 5x5x48 convolutions with stride [1 1] and padding [2 2]
7	ReLU	
8	Cross Channel Normalization	cross channel normalization with 5 channels per element
9	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0]
10	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1]
11	ReLU	
12	Convolution	384 3x3x192 convolutions with stride [1 1] and padding [1 1]
13	ReLU	
14	Convolution	256 3x3x192 convolutions with stride [1 1] and padding [1 1]
15	ReLU	
16	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0]
17	Fully Connected	4096 fully connected layer
18	ReLU	
19	Dropout	50% dropout
20	Fully Connected	4096 fully connected layer
21	ReLU	
22	Dropout	50% dropout
23	Fully Connected	2 fully connected layer
24	Softmax	
25	Classification Output	2 class classfication layer, 'left' and 'right'

1.5 Analysis

Convolutional Neural Network reduced the work on feature extraction since it automatically finds the features. The parameters of CNN is not optimal for this problem but yet achieved great accuracy. The first convolutional layer use a 11x11 filter, the intrinsic feature of left/right feet could be a larger area than a 11x11 window. So this feature is left for the deeper layers to find.

Image preprocessing in this projects plays a critical role. It not only makes the overall accuracy of classifiers much higher, also reduces the convergence time drastically.

Since I use matlab to do all my work in this project, and SVM in matlab only utilizes the

CPU while neural networks use GPU for training, the training speed of CNN is drastically faster than kernelized (RBF) SVM. SVM based on the PCA reducted data could be much faster.

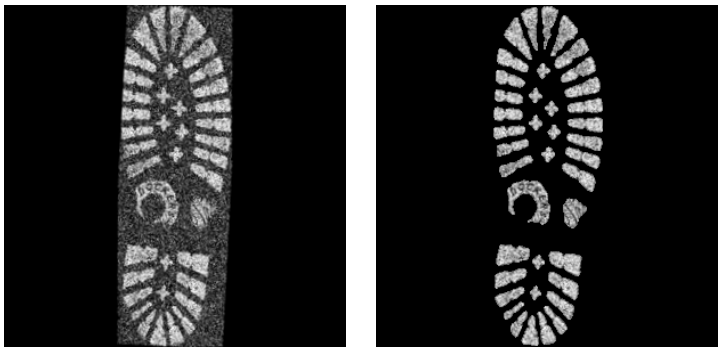
2 Footprint Matching

2.1 Data processing

Most processing techniques are identical to Task 1. In additional to the data processing techniques in Task 1, the texture of footwears are essential in Task 2. Here only different processing techniques than Task 1 are introduced. Limited by the length of this report, there are still various preprocessing techniques I have utilized in this project yet left not introduced.

2.1.1 Gamma adjust and threshold

For train image, data are augmented and their gamma is randomly adjusted between 1.6 and 2.4. For test and validation image, gamma is fixed to adjust to 2.2. The threshold is set to 0.1, pixels whose grayscale are below the threshold is set to 0. A gaussian blur filter is used on an identical image as reference to determine if a pixel needs to be set to 0.



(a) (b)

Figure 4: Noise removal of images. (a) Image before noise removal. (b) Image after noise removal

2.2 Machine learning techniques involved

KNN, CNN, Data Augmentation, Transfer Learning

2.3 How and why I choose these techniques and result

Since in Task 1 CNN did a pretty good job, I directly choose to use it in Task 2 as well. The training accuracy on mini-batch is around 85% on heavy augmentation, and around 97% on mild augmentation. The detailed training adjustment process is demonstrated in analysis section. The **parameters** of the CNN is the same as Task 1, except for the last fully connected layer and classification layer. These two layers are adjusted to output 1000 classes.

2.4 Analysis

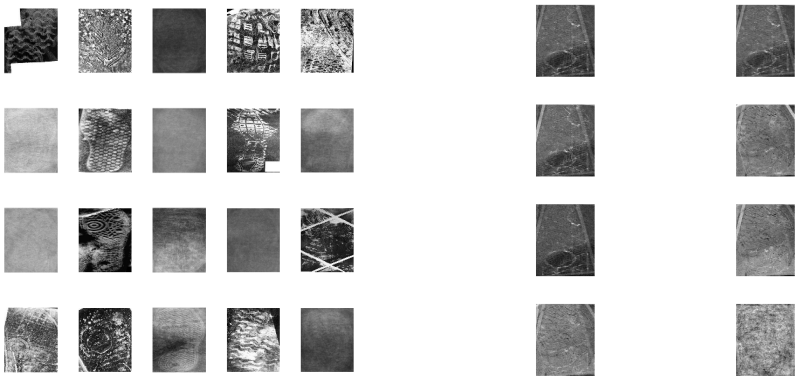
On initial training, that is, each images is rotated to 18 images with a step of 10 degrees. Then I use techniques introduced in Task 1 to rotate them back. (In order to look like the cropped effect) When the training set converges at 90%, the validation accuracy is only 22%. Then, I realized data augmentation is the key in this task. So I randomized everything in the image read function. In the training phase, the images are read from disk and augmented on the fly. As you can notice from my source code in file readAndProcessImage.m, the Gaussian noise's mean and variance is randomized from 0 to 0.7. Then the image is cropped randomly. The image has a 50% chance of being flipped, and rotated to a random angle then rotate back using my preprocessing algorithm. Then it is blurred using a Gaussian filer with coefficients randomized from 0 to 2.0. Also, the final step, the image

gamma is randomly selected from 1.6 to 2.4. All these randomizations prevent the over-fitting of CNN as the training epoch number increases. And there are actually no same image will be trained twice in the trainiing process. Under such heavy augmentation, the training accuracy is at the same level of validation accuracy. The validation accuracy is 84.0%. The test set images are less noised than validation set, the test accuracy is 99.2%.
For the size of first convolutional layer, I have tried 11x11, 7x7 and 3x3. While 3x3 keeps more details of the original image, it is less robust to heavily noised images. 7x7 seems to require more convolutional filters or deeper networks to achieve better accuracy under the 227x227 image setting. Since the data augmentation methods are built up from scratch, transfer learning are used to accelerate convergence. After each update of augmentation function, the layers are inherited from the last training. By comparing the mini-batch accuracy of last saved checkpoint and the first batch of updated augmentation, it is possible to adjust the parameters in augmentation for less over-fitting. By this method, the optimal gamma value to reduce the noise background of validation/test images is found. That is, around 2.1 to 2.2. In my approach, 2.2 is used.

3 Pattern Discovery

Data processing Images are color reversed and resized to 300x300.

Machine learning techniques involved K-means, Hierarchical clustering, Autoencoder
While without much data preprocessing, hierarchical clustering seems always cluster most of the images to a single cluster, and other images takea whole cluster. Then I used Autoencode tyring to find some pattern in the encoder weights. But it didn't work. Then I returned to the most naive way, K-means. It seems to have found some pattern in the images. Limited by the length of this report, figures of the weights of encoders in the autoencoder cannot be presented.



(a) (b)

Figure 5: Images cluster centers (a)using K-means, K=20, (b)using autoencoder, the weights of encoders, hidden size=8

Parameters Since there are only 300 images, the cluster number should not be set too large. But the images are quite different. So I selected K=20. And the autoencoder set hidden size 8. Trained 1000 epochs.

Analysis As we can see, some of the similar images has been clustered together. This is out of my expection, since I thought K-means is not very robust to scale, rotation and shift. Though I thought that autoencoder will do a better job, it turned out that in most cases encoder weights looks like an image with majorly white noise. Feature extraction is critical in this task. I have tried SURF but it didn't work as well. Since SURF only finds a series of points yet it is hard to extract features from these points with some sense of invariance across different images.

References

[1] Yanzhao Zhou, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. *Oriented Response Networks*. University of Chinese Academy of Sciences, Duke University, 2017.

[2] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton *Imagenet classification with deep convolutional neural networks.*. Advances in neural information processing systems. 2012.