10-701 - Project: footprints

Rogerio Bonatti
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
rbonatti@andrew.cmu.edu

Abstract

Footprints are usually critical evidences in the crime scenes. In this project, I will help the police deal with footprints using machine learning techniques. First, I classified 10,000 footprints between left and right foot with 98.1% precision. Then I classified another 10,000 footprints within 1000 classes with 99.0% precision, where each class represents a foot from a unique suspect. In the last section I group footprints into five different clusters based on their geometric features.

1 [40 (+10) points] Left/Right Footprint Detection

I will explain my methodology for classifying the footprints into left and right foot:

1.1 Preprocessing data

The initial data provided for this question, both for training and validation/testing contained footprints that had noise in the background, and that were rotated in an arbitrary angle. If the rotation angle with respect to the principal axis of the footprint was greater different than 0 or 180 (positions where the footprint was vertical), then part of the foot was missing from the pictures. Examples of both training and validation data can be found in Figures 1.

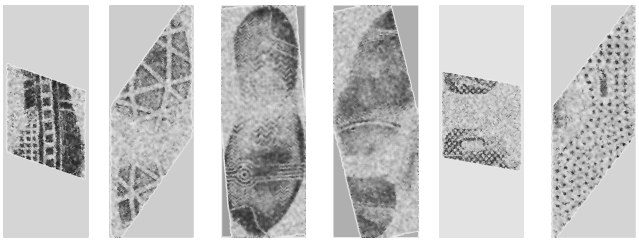
(a) Train (b) Train (c) Train (d) Test (e) Test (f) Test

Figure 1: Original training and testing images for Q1

My initial step was to pre-process the images (both training and test set) such that all images:

- had the feed aligned in the vertical orientation (pointing either up or down)
- were blurred with a Gaussian filter to minimize the impact of the noise

10807 Class Project (2016).



(a) Train (b) Train (c) Train (d) Test (e) Test (f) Test

Figure 2: Processed training and test images for Q1

- had the background without the actual foot filled with a gray color, that I arbitrarily chose to be the average pixel intensity for that image

The resulting images are shown in Figures 2.

To rotate the image to the vertical position I binarized the images and discovered the angles of the main edges.

In addition, I augmented the processed images in two ways:

- Flip image from left / right to obtain the opposite foot
- Flip image from up / down to obtain the opposite foot in opposite orientation
- Rotated image 180 deg to obtain the same foot in opposite orientation

By the end of the augmentation process, I had 20,000 images representing left feet and 20,000 images representing right feet (basically, each original image resulted in 4 processed images).

1.2 Classification algorithm

I initially thought of different possibilities for solving this classification problem, including SVMs and convolutional neural networks. I opted to use neural networks due to their role as state of the art classifiers for images.

I used the Keras package to implement the neural network due to its simple interface for both the architecture side and handling the data files. This blog by Francois Chollet was a very good guide on how to setup a network on Keras.

I decided to use a standard convolutional architecture for this project, following papers on image classification. I opted for the VGG16 architecture [2] due to the balance it presents between having a network complex enough that can capture a big variety of details in the different classes, while having a small enough size so that it could be trained on the computational resources I had available for this project.

Note: another viable alternative for the VGG16 network would be, for example, AlexNet [1]. I could have also designed my own architecture to solve this problem (with a larger or smaller number of parameters), but I did not see a true necessity of doing so given that AlexNet and VGG16 have been extensively tested and validated in the scientific community.

My VGG architecture contained an input layer of size 224x224, 10 convolutional filter layers, and 2 fully connected layers with 4096 neurons each. Between each of the layers I used batch normalization, dropout (value of 0.5), and RelU units. My loss function was binary cross-entropy and I used Adam optimizer. I loaded images as grayscale, and had batch sizes of 32.

1.3 Training

I trained my network on the Field Robotics Center cluster, using a Nvidia TitanX GPU. The loss value with respect to epochs can be found in Figure 3.

Figure 3: Loss value with respect to number of epochs for Q1

1.4 Results

I obtained good results in the left / right foot classification for both validation and test sets. For validation I obtained 98.12% precision, and for testing I obtained 98.14% precision.

1.5 Conclusion

The VGG16 neural network, coupled with the data augmentation processes that I used provided very good results for the left / right foot classification process.

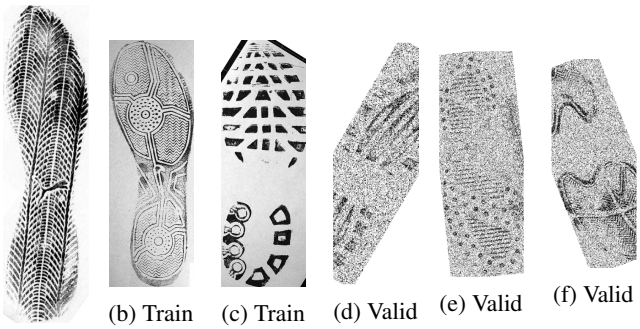
2 [40 (+10) points] Footprint matching

I will explain my methodology for classifying the footprints into the 1000 possible classes:

2.1 Preprocessing data

The initial data provided for this question for the training set had right feet in a noise-free environment. Examples of training data can be found in Figure 4.

For validation/testing contained footprints that had noise in the background, and that were rotated in an arbitrary angle. If the rotation angle with respect to the principal axis of the footprint was greater different than 0 or 180 (positions where the footprint was vertical), then part of the foot was missing from the pictures. Examples of validation data can be found in Figure **??**.



(a) Train (b) Train (c) Train (d) Valid (e) Valid (f) Valid

Figure 4: Original training and valid images for Q2

My initial step was to pre-process the training images such that all images:

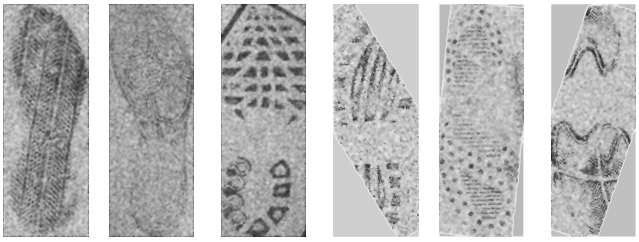
- Had noise added to them, in a intensity comparable to the noise in the validation/test data (done by visual inspection)
- After noise addition, were blurred with a Gaussian filter to minimize the impact of the noise
- had the background without the actual foot filled with a gray color, that I arbitrarily chose to be the average pixel intensity for that image

and for the validation images:

- had the feed aligned in the vertical orientation (pointing either up or down)

- were blurred with a Gaussian filter to minimize the impact of the noise
- had the background without the actual foot filled with a gray color, that I arbitrarily chose to be the average pixel intensity for that image

The resulting images are shown in Figures 5.



(a) Train (b) Train (c) Train (d) Valid (e) Valid (f) Valid

Figure 5: Processed training images for Q2

To rotate the image to the vertical position I binarized the images and discovered the angles of the main edges.

In addition, I augmented the processed images for training:

- Flip image from left / right to obtain the opposite foot
- Flip image from up / down to obtain the opposite foot in opposite orientation
- Rotate image every 11.25 deg to obtain the same foot cropped in different orientations, because the validation image can be in any orientation

By the end of the augmentation process, I had 64,000 images representing each foot (basically, each original image resulted in 64 processed images).

2.2 Classification algorithm

Just like for the first problem, I initially thought of different possibilities for solving this classification problem, including SVMs and convolutional neural networks. I opted to use neural networks due to their role as state of the art classifiers for images. I used the Keras package to implement the neural network due to its simple interface for both the architecture side and handling the data files.

Again, I decided to use a standard convolutional architecture (VGG16) for this part. My architecture and parameters were equal to Q1, with addition of one output layer with 1000 outputs and categorical-crossentropy loss.

2.3 Training

I trained my network on the Field Robotics Center cluster, using a Nvidia TitanX GPU. The loss value with respect to epochs can be found in Figure 6.

Figure 6: Loss value with respect to number of epochs for Q2

2.4 Results

I obtained good results in the left / right foot classification for both validation and test sets. For validation I obtained 98.95% precision, and for testing I obtained 98.93% precision.

2.5 Conclusion

The VGG16 neural network, coupled with the data augmentation processes that I used provided very good results for the multi-class classification process.

3 [20 points] Pattern Discovery

For this question I tried two different approaches for unsupervised clustering, both of them following the same principle:

- Extract features from the images of the dataset, and generate a feature vector for each image
- Cluster the feature vectors using K-Means, varying the value of K until I find clusters that qualitatively make sense for the case of footprints

Now I will describe both approaches I tried for feature extraction:

3.1 Feature extraction using the convolutional neural network

The first method I tried for feature extraction was using the convolutional neural network that I trained for Q2. The idea was load the weights from the neural network from Q2 following the approach proposed by [3] right after the last convolutional layer and flatten the output, resulting in one feature vector for each image.

To reduce the dimensionality of the data (which was about 5000 in the flattened vector), I tried using PCA to reduce it to about 100. After this I clustered the data using K-Means, varying the value of K and observing the images of the resulting clusters.

3.2 Feature extraction using SURF and bags of features

A second method that I tried for feature extraction was creating bags-of-features with SURF features. First I extracted SURF features from all the 300 images, then clustered these feature vectors into a vocabulary of 500 "words" (using K-means), and then for each image I extracted SURF features and counted the number of features of each of the 500 different types, creating a feature vector. Lastly, I clustered the feature vectors for the images using K-means, varying the size of K and observing the resulting clusters.

3.3 Results

Here I present the clustering results with both methods.

First, the results obtained from features extracted from the neural network did not present satisfactory clusters. Images with very different types of features were grouped together, and no significant characteristics between different shoe types could be distinguished from those resulting clusters. For the sake of brevity I will not include images from the resulting clusters. One interesting result though was that by plotting the within-cluster sum of squares, I could observe that a good number of clusters to choose from was 5 (where the "knee" of the curve occurred).

I tried several combinations of values for PCA (larger and smaller final size of feature vectors) and even tried leaving the first fully connected layer of the neural network as the feature vector, but all these approaches resulted in clusters that did not present any significance for a human evaluator.

I have some hypothesis on why this clustering with neural network did not work:

- All images used for training in Q2 had one characteristic scale, while images in Q3 had very different scales. The neural network learned to identify features solely in one scale for Q2, and in Q3 we have images that are much more zoomed in
- Noise was very different in Q2 and Q3. The images in Q2 were cropped, rotated and had a very particular type of noise, whereas images in Q3 had very different types of noise, weird cropping, and very different types of noise. Therefore the network was not trained to identify these differences

Secondly, the results using features extracted with the SURF and "bag of words" were interesting, and make sense from the point of view of a police investigation. These features were able to, with reasonable precision, separate footprints that contain different patterns in the sole of the shoes such as squares, strides, circles. I arbitrarily selected a number of clusters K=5 after trying a few possibilities and evaluating them qualitatively. Results from the clusters, including positive cases and failure cases are shown in Figures 7-11. Here follows a brief explanation of each cluster:

- Cluster 1: contains images that have large circular features
- Cluster 2: contains images that have strides
- Cluster 3: contains images that have relatively dark rectangles / squares
- Cluster 4: contains images that have small circular features
- Cluster 5: contains images that have elongated rectangles

By clustering the shoe-prints into these categories the police officers can discover patterns in the data to help investigations, grouping similar suspects.

Note: in each of the clusters there is a minority of images that do not follow the general trend of that cluster, but for the report I selected images that represented the general trend of the cluster well.

Figure 7: Cluster 1: zoom in to see!

Figure 8: Cluster 2: zoom in to see!

Figure 9: Cluster 3: zoom in to see!

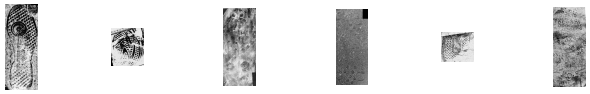


Figure 10: Cluster 4: zoom in to see!

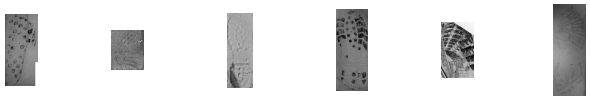


Figure 11: Cluster 5: zoom in to see!

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[2] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[3] Yichuan Tang. Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239*, 2013.