Crime Scene footprint matching for recidivism prediction

Chirag Nagpal
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
`chiragn@cs.cmu.edu`

Abstract

Footprints form a crucial piece of evidence in crime scenes, which forensic experts and law enforcement agencies depend on to build incriminating evidence against a suspect, especially in cases of recidivism. In this paper we describe a system to perform 2 machine learning tasks on footprint data that are of application to law enforcement tasks. 1) footprint detection, to detect if the print belongs to the right or left foot, and 2) Footprint matching, which given prior footprint data aims to match newer footprints. We further extend this work by performing some analysis and exploration, and pattern discover using some Machine Learning Techniques.

1 Dataset

Our dataset consists of multiple footprints, belonging to either left/right foot. The footprints are not crisp, and have at times multiple different orientations along the principal axis, along with non trivial form of noise. The random degrees of orientation, make it hard to use simple computer vision techniques, since a lot of standard models are not rotation invariant, at the same time the footprint is occluded due to its rotation, making it hard to get the entire template of the footprint. For the task 1, our dataset consists of 5000 images of each foot. For Task 2 , we have 1000 gold standard images to which we need to match a set of noisy 10000 images.

1.1 Preprocessing

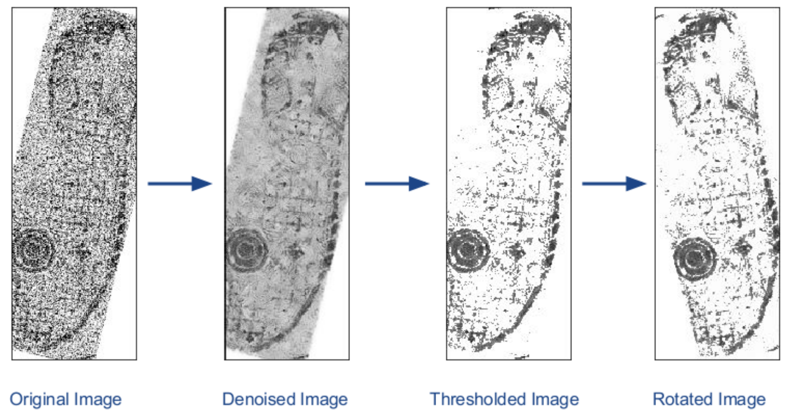

Figure 1: The Preprocessing pipeline for the Raw images.

In order to reduce the noise, we apply a Non Local Means denoising filter [1]. From the denoised image we then threshold the image in order to extract the image parts of interest for our downstream task.

Next we perform rotation inorder to reorient the principle axis of the foot with the vertical axis. We again apply a gaussian filter to find the region of the image with information, then depending on the amount of white space in the image, we perform a linear regression by keeping either the x or y axis as the dependent variable. This gives us the primary orientation of the image. (Note, we do not perform Hough transform here, inorder to explicitly use tools and techniques learnt in class)

We then compute the angle of rotation of the image using the previously computed value of the regression line and its intercept with the y axis. We then rotate the image using an affine transformation, that rotates the image the desired angle. Note, the given method will reorient the image along its principal axis, however it will not ensure the image from being upright. This is not a very significant issue in the downstream task and we will solve this using data augmentation.

2 Learning

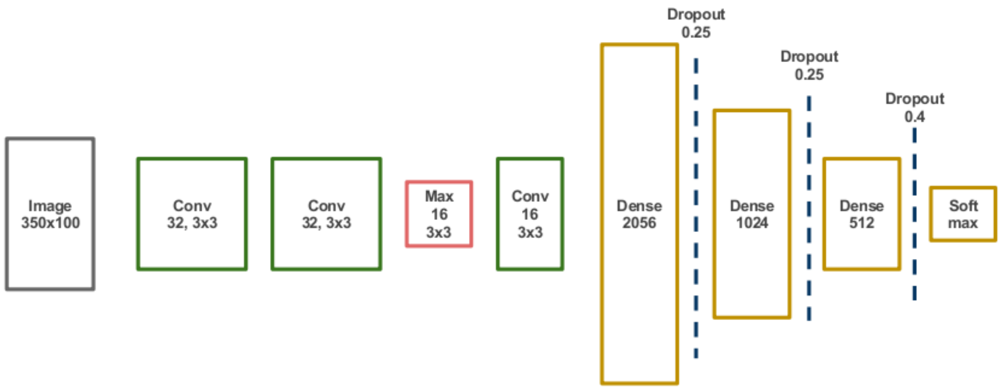

Figure 2: The proposed Neural Model for the Left Right detection task.

2.1 Task 1: Left/Right Detection

Given the dataset we, first train a deep convolution neural model and treating the left & right footprint detection as a binary classification. Our last layer consists of a sigmoid activation which we train to optimize the cross entropy loss, using the modification of SGD that allows faster convergence, called Adam [2], using the software package, Keras [4]. All the layers use Rectified Linear Units [5] as activations. After the convolution layers, we have a series of dense, fully connected layers along with different values of dropout to ensure robustness by preventing overfitting [3].

We also augment the data by flipping the images long the vertical axis and adding them from one class to the other. We further also rotate all images 180 degrees. This is done inorder to ensure that the model can also learn from images that were not upright after the initial reorientation to the principal axis.

2.2 Task 2: Footprint matching

This task is a little more difficult as compared to the previous task, here we are given 1000 gold standard images and we are to match these to 10000 raw images. One can think of this as a 1000-way classification as opposed to the previous task which is just binary classification. However, given that we have just one gold image per class, it is difficult to train a supervised model, given the lack of training data. We thus approach this problem in a fundamentally different manner. We rather treat this as a information retrieval problem wherein we aim to find the image which is closest to the gold

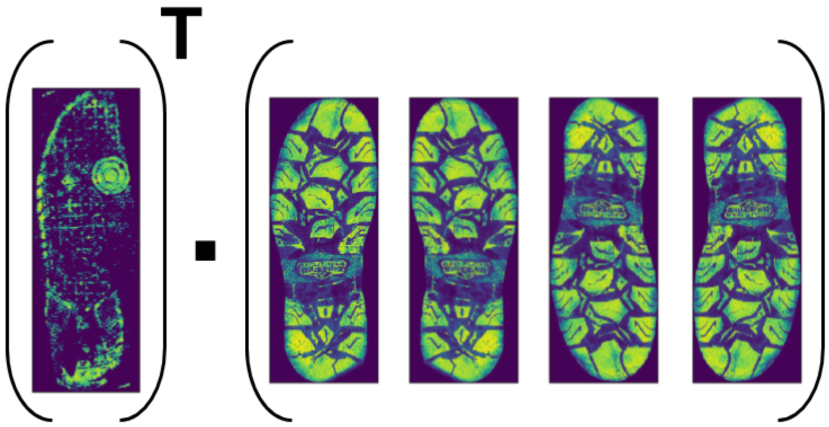

Figure 3: Illustrative example of our approach, modeled as an IR problem. the flattened vector of the images is transposed and dot product is taken with the flattened gold set.

image. Let the test image be described by the vector x_i. Here $x_i \in \mathbb{R}^{35000}$. Let the gold image be described by the vector y_j, where $y_j \in \mathbb{R}^{35000}$. Therefore,

$$j_i^* = \arg\min_j d(x_i, y_j) \tag{1}$$

Here $d(,)$ is a distance function that can compute a similarity between the two vectors in the $\mathbb{R}^{35000}$ space.

$$j_i^* = \arg\min_j x_i^\top . y_j \tag{2}$$

We first use the dot product as a measure of similarity between the images. However, we noticed that dot product had a tendency to increase when a given image had a greater amount of brighter regions. Thus , most of the images matched with brighter images from the gold set rather than aligning with the correct image. In order to mitigate this effect of the simple dot product, we normalize each vector with its l_2 norm. (This corresponds to the cosine of the angle between the two vectors, and since its normalised by the magnitude of each image vector, reduces the tendency for the brighter images to show up.)

$$j_i^* = \arg\min_j \frac{x_i^\top . y_j}{||x_i||_2 ||y_j||_2} \tag{3}$$

Just like in the previous task, we augment the dIn this section we will describe the results for the various tasks. ata, with flipped as well as images that are rotated to be upside down. Instead of augmenting the query data, which would result in 10,000 $\times$ 4 examples, we simply augment the gold images. This results, in 4000 gold images, and thus less comparisons are required to be made. (While it can be argued that we do not gain much in computational efficiency, we have a large gain in space efficiency as now, we do not need to store vector representations of the augmented query images)

3 Results

3.1 Task 1

	Development Set	Validation Set	Test Set
PP	88.67%	*	*
PP+Rot	93.23%	*	*
PP+Rot+Flip	96.65%	93.3%	93.5%

Table 1: PP: Preprocessing, Rot: Rotate by 180 degrees, Flip: Flipping images to augment each class

Table 1 shows the results of Task 1 and also how we improve by augmenting the data by rotating and flipping the images. Note the development set here was created using the provided data and was 5% of the training data. as expected due to the lesser amount of Development data we see that the model performance is lower for the Validation and Test Data, which suggests we are overfitting our Development data. It is very hard to not overfit, since a small sample from the training data is likely to be biased, while increasing Sample size leads to the model having lesser data to learn from. We do not report the performance of the models without augmentation on the Validation and Test data due to 1) Computational Costs to run the models on the entire dataset 2) Limited number of attempts at submission. In this section we will describe the results for the various tasks.

3.2 Task 2

	Validation Set	Test Set
PP	*	*
PP+Aug, Dot Product	41.3	*
PP+Aug, Cos	75.3	75.0

Table 2: PP: Preprocessing, Aug: Data Augmentation, Dot: Dot Product, Cos: Cosine Similarity

As with the prior task, we augment the training set with the rotated and flipped images. With Dot Product similarity we get a score of 41.3% on the Validation Set, this is vastly improved using Cosine Similarity, and we get a score of 75.3% on the Test Set.

4 Task 3: Pattern Discovery & Exploration

The final task for the challenge consists of exploring the data, and performing some patter n analysis for a given set of 300 images. We first directly use the image vectors and compute the k-NN and the k-Means using cosine distance to find certain trivial patterns, however since these images have multiple orientations, and different color gradients, these simple approaches completely fail.

We then move to use the activations of the dense layers and perform t-SNE [6], an unsupervised dimensionality reduction technique, that projects our data to a lower dimensional (typically, 2 or 3) such that the inter feature distance of the vectors is preserved. However, on manual inspection we find that the given images are not as grainy as the images prvoided for Tasks 1 & 2. We therefore, reduce the intensity at which denoising is performed to these images. We then use the first dense layer activations as the feature vectors for our images, and proceed to perform t-SNE. It is observed that the data has a tendency to separate into mostly 2 groups, this is predictable, since our model is trained on a discriminatory binary classification task.

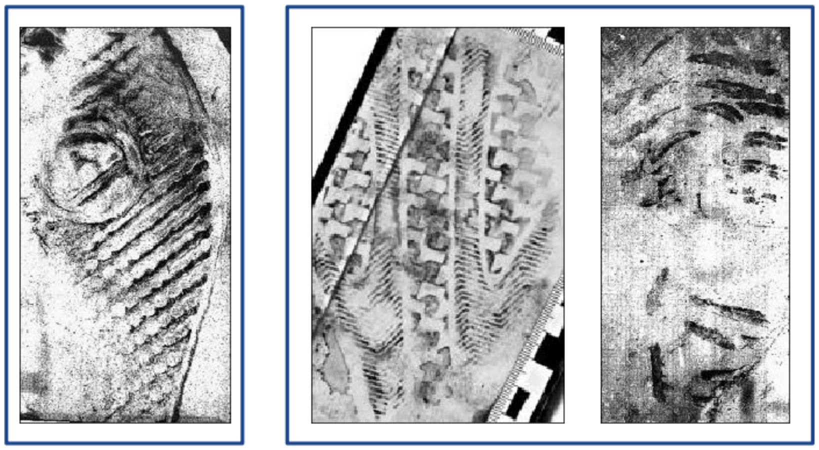

Figure 4: Results of 2-NN on a given Query Image, the noisy nature of this data, makes it hard to use standard image vectors for any task.

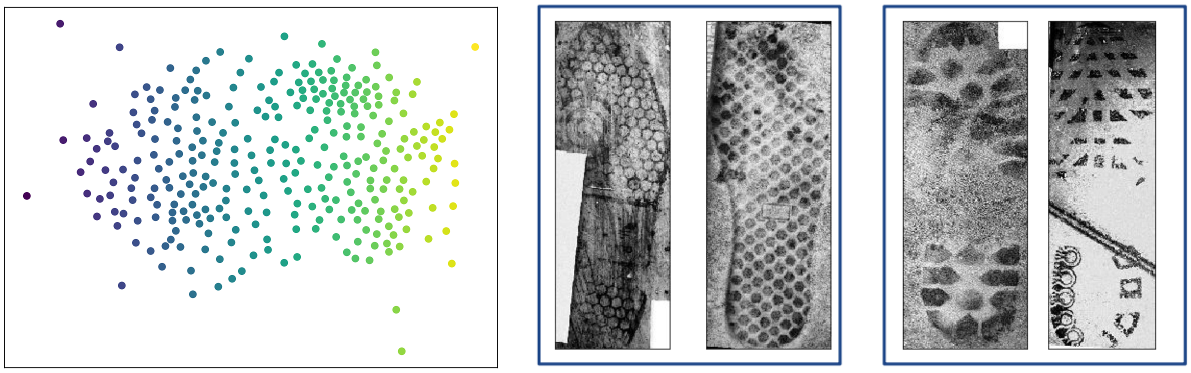
Figure 5: The figure on the left displays the tSNE representation of the feature vectors. On the right, we have images that were close in tsNE space, and also found similar upon visual inspection.

Although, certain images close in tSNE space did indeed visually seem similar, however it is difficult to make claims with certainty here. Without standard ground truth it is hard to say if the retrieved images were truly close in the latent neural space or if the visual similarity was purely out of chance. Confirmatory bias also cannot be ruled out.

5 Conclusion

In this paper we present approaches to predicting the orientation of a footprint, using a deep convolutional neural network, and report a high score of 93.4% on the first classification task . We further use standard linear algebra and model task 2, involving matching footprints, treating it as a information retrieval problem and achieve a score of 75.0% on the test set. We finally perform some standard clustering and dimensionality reduction techniques in order to extract underlying patterns from the provided footprint images.

References

[1] Nandhini, M., and T. Nalini. "Survey of Image Denoising Algorithm." International Journal of Advanced Research in Computer Science 5.3 (2014).

[2] Kingma, Diederik, and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[3] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[4] Chollet, François. "Keras." (2015).

[5] Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." Proceedings of the 27th international conference on machine learning (ICML-10). 2010.

[6] Maaten, Laurens van der, and Geoffrey Hinton. "Visualizing data using t-SNE." Journal of Machine Learning Research 9.Nov (2008): 2579-2605.