DEEP MULTILABEL CNN FOR FORENSIC FOOTWEAR IMPRESSION DESCRIPTOR IDENTIFICATION

Marcin Budka, Akanda Wahid -Ul- Ashraf, Matthew Bennett

Faculty of Science and Technology

Bournemouth University

Fern Barrow

Poole BH12 5BB, UK

e-mail: {aashraf, mbudka, mbennett}@bournemouth.ac.uk

Scott Neville, Alun Mackrill

Bluestar Software Ltd.

Fair Cross Offices

Stratfield Saye RG7 2BT,

e-mail: {scott.neville, alun.mackrill}@bluestar-software.co.uk

February 11, 2021

ABSTRACT

In recent years deep neural networks have become the workhorse of computer vision. In this paper, we employ a deep learning approach to classify footwear impression's features known as *descriptors* for forensic use cases. Within this process, we develop and evaluate an effective technique for feeding downsampled greyscale impressions to a neural network pre-trained on data from a different domain. Our approach relies on learnable preprocessing layer paired with multiple interpolation methods used in parallel. We empirically show that this technique outperforms using a single type of interpolated image without learnable preprocessing, and can help to avoid the computational penalty related to using high resolution inputs, by making more efficient use of the low resolution inputs. We also investigate the effect of preserving the aspect ratio of the inputs, which leads to considerable boost in accuracy without increasing the computational budget with respect to squished rectangular images. Finally, we formulate a set of best practices for transfer learning with greyscale inputs, potentially widely applicable in computer vision tasks ranging from footwear impression classification to medical imaging.

1 Introduction

In this work we develop an approach to train a deep Convolutional Neural Network (CNN) to classify features in footwear impressions for use in forensic applications. The features we classify are known as *descriptors* within the UK footwear forensic units [1, 2, 3, 4] and can be defined as recognisable units within a footwear pattern which can be classified. The *descriptors* are used by forensic practitioners to describe the makeup of a footwear pattern.

Every footwear impression added to the UK's National Footwear Reference Collection (NFRC) [1] is manually labelled with the *descriptors* [1]. The NFRC is built on an agreed standard for coding footwear patterns for different forces in

---

[1]The National Footwear Reference Collection (NFRC) and The National Footwear Database (NFD) are developed and maintained by Bluestar Software Ltd (BSL) [1]

the UK and at the time of writing, to the best of our knowledge, is the biggest police-owned collection of footwear impressions in the world. The NFRC footwear pattern collection is updated on a regular basis [5].

The NFD is a successor of the NFRC where footwear labels are maintained and added regularly. The NFRC records the custody and crime scene marks while the NFD facilitates matching with the NFRC footwear patterns. Currently, around 30 out of 43 police forces in England and Wales, continuously send or update data in the NFD [5].

The NFRC uses a total of 17 *descriptors* to identify a footwear impression. Each of the *descriptors* is assigned a unique name and code. A shoe print or footwear impression may contain any subset of these *descriptors*. The location of the *descriptors* are divided into two parts: 1) the heel / instep, and 2) the main sole (i.e. top). In this study we do not exploit this location information in any way. A single *descriptor* can exist multiple times in a shoe print, however, the specific location (other than the heal/instep or main sole) and frequency of the *descriptor* is not identified and counted.

Each of the 17 *descriptors* (Table 1) has specific semantics (for the purpose of quick identification by forensics practitioner rather than a computer), which relate to the name of the *descriptor*. For example, *descriptor D05: 5 sided*, contains all shapes which are 5 sided; *descriptor D09: Text* indicates any text that can be found on a shoe print. The number of possible geometric variations that are usually found can potentially be infinite. For example, *descriptor D09: Text* can be any combination of characters and fonts, while *descriptor D05: 5 sided* can be a rough pentagon of any shape and form. Two *descriptors* can overlap, resulting in a multiple *descriptors* from a single topological subpattern on a footwear impression. For example, *descriptor D09: Text* and *D10: Logo* usually overlap, as many logos contain text. Additionally, among the 17 *descriptors*, three are subcategories of two main/parent *descriptors*: *D01-01: Wavy*, *D01-D02: Curved-wavy* are the subcategories of *D01: Bar*, and *D02-01: Target* is a single subcategory of *D02: Circular*. While labelling with the *descriptors* for a footwear impression, the microscopic patterns of the impressions are not usually considered. For example, *D12: Texture* can contain microscopic patterns which are also *D06: 6 sided* but usually *D06: 6 sided* is not labelled in such cases as these microscopic patterns are often not reliable and persistent [6]. All the sided shaped *descriptors* (e.g. *D03*, *D04*, etc.) do not necessarily have very precise straight lines as sides but some curves and deformations are ubiquitous.

1.1 Footwear Impression Imaging Methods

In UK policing collection of footwear evidence is normally done in two scenarios: 1) collection of detainee footwear in custody, and 2) collection of crime scene marks. The vast majority of the footwear impressions captured from detainees in custody usually follow one of the below processes [7]:

- *Inked Impressions:* The inked impressions are captured using a specialist pad and paper kit (sometimes called a 'Bigfoot Kit') [1, 6, 2]. The kit uses a pad with a reactive chemical and specialist paper. The impressions can then be digitised using an office document scanner, if required.
- *Ink-less Impressions:* A specialised footwear impression digital scanner is used in this case to capture the footwear impression without any use of ink. This process produces only a digital copy of the impression whereas the inked impression also produces a physical copy on paper [1].

Additionally, some UK forces use coloured photographs of the shoe sole as opposed to using one of the impression capturing methods described above [1].

1.2 Identifying Descriptors

In practice to date, the *descriptors* are manually identified by experts and are only used as an intermediate step to identify a pattern. Processes vary between police forces, however when adding an impression to the NFRC, two independent experts individually identify the *descriptors*. If both experts agree on the set of identified *descriptors*, the footwear impression image is labelled with the identified *descriptors*. However, when there is a disagreement between the two experts, the labelling process involves a panel of experts for further analysis. The accuracy of identifying the *descriptors* by experts are thought to be 'very high', however, to the best of our knowledge, there was no empirical study to quantify this accuracy [1].

1.3 Limitations

The main limitation associated with manually identifying *descriptors* is the time and cost of human expertise. Although forensic practitioners are able to directly identify many common footwear impressions without the need for classification against the *descriptors*, classifying rare or new shoe models takes longer. As there are tens of thousands of footwear models, it is impractical for a human expert to be able to accurately identify a specific model with only the *descriptors*. The NFRC/NFD provides a number of additional searching and ordering features to make identification possible in a
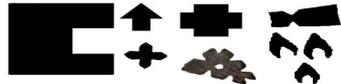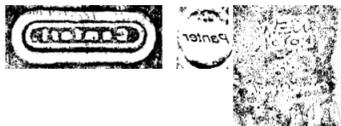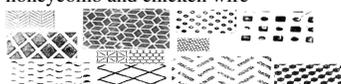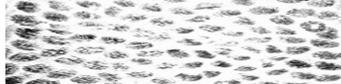
D01 **Bar**	A bar of any type such as straight, angled, curved, including chevrons	**D07** **Complex**	Shapes such as star, arrow, waisted bar, heart and cross, and any other shape with more than six sides
D01-01 **Wavy**	A bar element with more than one directional change	**D08** **Zigzag**	A broken or continuous line that changes direction repeatedly with abrupt right and left turns
D01-02 **Curved-wavy**	Any bar shape/ element deviating from a straight line with a single rounded directional change however small the angle of the curved section	**D09** **Text**	Any alpha-numeric characters; may overlap with D10
D02 **Circular**	Includes circle, semi-circle, oval, semi-oval, concentric circles, target, tear-drop, stud, crescent	**D10** **Logo**	A brand or trademark incorporating a symbol, badge, emblem or picture; may overlap with D09
D02-01 **Target**	Any concentric circle arrangement whether the centre-most circle is hollow or solid	**D11** **Lattice**	A regular, interlocking and/or repeated pattern (aka network, web or trellis); includes brickwork, herring-bone, honeycomb and chicken wire
D03 **3 sided**	All types of triangle including those with one rounded side such as a pie-segment	**D12** **Textured**	This includes pre-dominant stippling, crepe or random patterns added by the manufacturer as part of their design
D04 **4 sided**	Square, rectangle, oblong, paralle-logram, rhombus, diamond, arrowhead	**D13** **Hollow**	A pattern that has the appearance of a hollow shape, such as a doughnut or frame
D05 **5 sided**	Usually a regular shaped pentagon, but includes all five-sided shapes	**D14** **Plain**	A plain surface with no patterns or texture
D06 **6 sided**	Usually a regular shaped hexagon, but includes all six-sided shapes		

Table 1: Footwear *descriptors* for the UK's National Footwear Reference Collection (NFRC)

practical time span. These features are generally used in the same way for all searches (looking at frequency/geography of distribution) and therefore take little time to use compared with the time taken to identify descriptors. However, the most frequently worn footwear are very well known to the forensic practitioners thus are easily labelled by them, without the need of any computer system, or the *descriptors*. Since the accuracy of human footwear forensics experts are not empirically evaluated, the automated process cannot be argued to be same or better than human experts. Despite this, clear use cases for an automatic *descriptor* identification exist. For example, when a new footwear model is captured, labelling would be completed by an expert, then blindly verified by another. An automatic *descriptor* identification will be faster and have higher availability for the second check as the number of human experts available is limited. Automatic *descriptor* identification could potentially replace the second opinion when adding patterns to the NFRC (see Section 1.2).

2   Automated Descriptor Inference

The automation of the *descriptor* analysis can provide rapid identification of the *descriptors* in a given impression, which in turn will result in faster identification of a shoe model from its print, especially for an untrained (in terms of footwear analysis) personnel. Additionally, the identified *descriptors* can be used to narrow down the search in the database with thousands of footwear impressions. The automated approach, which is not only capable of identifying the *descriptors* but also infer their topological location (e.g. using Grad-CAM [8]) can be further beneficial for training police users. Rapid automatic *descriptor* identification can be achieved without involving a forensic expert, resulting in faster determination of intelligence. The latter is particularly important as the suspect will then have little time to destroy the evidence and can be questioned sooner (ideally before leaving custody), resulting in a plausibly increased detection rate. In England and Wales, there are around 25-30 (an estimation without an official source) human experts who can identify the *descriptors* currently, whereas there are 123,171[2] law enforcement personnel [9] who may handle a case where identification of the *descriptors* may be necessary. Automated *descriptor* identification can potentially provide such expertise to all the law enforcement personnel in the UK.

Due to large variability in the complex geometric shapes and patterns of a *descriptor*, a simple template matching algorithm [10] would be suboptimal. Each of the *descriptors* has an apparent but variable high-level geometric semantics.
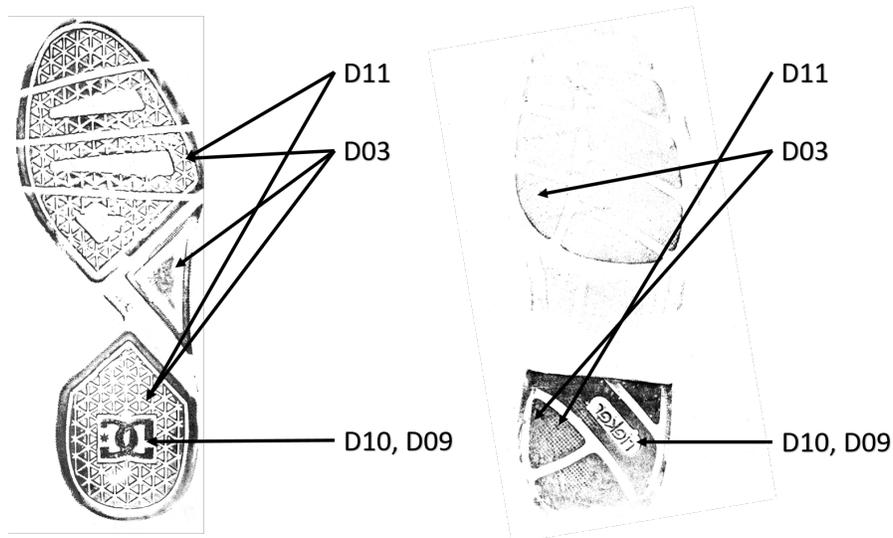

Figure 1: Different types of *descriptors*: D10, D11, D03 on two separate real inked impressions

Figure 1 shows two real-world inked impression with four *descriptors* each; *D03*, *D09*, *D10*, *D11*. As it can be seen, although the same descriptors appear on both of the impressions, their patterns are very distinct. While *D10: Logo* is an obvious example, a more 'stable' *D03: 3 sided* also looks quite different. The impression on the right has *D03* with smother edges, and also bigger in size than *D03* found in the left impression. Also note that although both the impressions have *D11: Lattice*, its appearance is very distinct.

---

[2]According to a statistical bulletin published on the 18th July 2019 by the Home Office for England and Wales. This number of officers does not include the British Transport Police

As a result, designing filters to identify the *descriptors* is not practical. Instead, a deep learning based approach has been taken, able to automatically learn the filters from the already existing manually labelled dataset.

3   Input Image Resolution in Deep Neural Networks

Training a deep neural network requires estimating a large number of parameters in the order of hundreds of millions. The matrix arithmetic operation performed to estimate these parameters are best suited for a Graphics Processing Unit (GPU) due to a GPU's better ability to perform highly parallel floating-point operations when compared with a CPU (Central Processing Unit). The GPU computational power are still limited however and there are other bottlenecks like moving data between the main memory and the GPU. As a result, a smaller model with a lower number of parameters is computationally more efficient than a bigger model with a larger number of parameters.

Apart from the base architecture (number of layers and units per layer) of a neural network, the number of computations grows approximately quadratically with the resolution of the input image. Higher resolution images also take up more space in the GPU memory, which tends to be smaller than system memory, limiting the batch size and further reducing the overall training speed. In order to reduce the computational cost and facilitate faster training, the input images are usually downscaled [11, 12]. However, the performance/accuracy of a neural network tends to suffer when image resolution is reduced.

It should also be noted that the theoretical benefit of a higher resolution image may not always increase with an ever increasing resolution of that image, e.g. once we have already achieved the theoretical upper bound of the accuracy for a specific domain. In our case, we have very small and complex features defining a class (see Section 1), thus, the upper bound of the resolution with a beneficial impact on the model is assumed to be higher than in the classification based tasks where the classes are usually more apparent.

4   Image Interpolation Techniques

Our dataset consist of high-resolution footwear impressions that have been captured via the means discussed in Section 1.1. As deliberated on in Section 3, in practice the images resolution need to be reduced and there are a number of different image interpolation [3] techniques that can be used here. In our experiments, we investigate and benchmark various combinations of image interpolation techniques, including:

- *Nearest Neighbour interpolation (N)*, which is the least computationally expensive and does not insert new colours in the result. In this interpolation, only the nearest neighbour's pixel intensity is considered. The estimation function f on a point (x, y) becomes a piecewise function with constant value [13, 14].

- *Bilinear interpolation (B)* is a linear interpolation over all non-channel dimensions of an image, i.e. for a two dimensional image it is the interpolation over both the X and Y dimensions [15]. A straight line passing through two points (x_1, y_2) and (x_2, y_2) between range x_1 and x_2 is the linear interpolant of these two points. For a range of (x_1, x_2), the slopes of the interpolant from both of these points (x_1 and x_2) should be exactly the same, hence the following equation of slopes can be formulated:

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1} \tag{1}$$

Solving Equation 1 for y gives:

$$y = y_1 \left(\frac{x_2 - x}{x_2 - x_1} \right) + y_2 \left(\frac{x - x_1}{x_2 - x_1} \right) \tag{2}$$

Equation 2 produces interpolation over the X direction. In case of a two dimensional image for four different points on the image, $Q_{11} = (x_1, y_1)$, $Q_{12} = (x_1, y_2)$, $Q_{21} = (x_2, y_1)$, $Q_{22} = (x_2, y_2)$, the task is to estimate the function f at a point (x, y). In this four points scenario, the linear interpolation on the X direction using Equation 2 gives us the following:

$$f(x, y_1) = \left(\frac{x_2 - x}{x_2 - x_1} \right) f(Q_{11}) + \left(\frac{x - x_1}{x_2 - x_1} \right) f(Q_{21}) \tag{3}$$

$$f(x, y_2) = \left(\frac{x_2 - x}{x_2 - x_1} \right) f(Q_{12}) + \left(\frac{x - x_1}{x_2 - x_1} \right) f(Q_{22}) \tag{4}$$

---

[3]We use the terms interpolation, resampling, downscaling, and resizing interchangeably

We can then use Equations 3 and 4 to interpolate on the Y direction in order to estimate $f(x,y)$:

$$f(x,y) = \frac{y_2 - y}{y_2 - y_1}\left(\frac{x_2 - x}{x_2 - x_1}f(Q_{11}) + \frac{x - x_1}{x_2 - x_1}f(Q_{21})\right)$$

$$+ \frac{y - y_1}{y_2 - y_1}\left(\frac{x_2 - x}{x_2 - x_1}f(Q_{12}) + \frac{x - x_1}{x_2 - x_1}f(Q_{22})\right) \quad (5)$$

- *Hamming (**H**)* interpolation technique uses a *sinc* approximating kernel by multiplying (convolution operation as its in the frequency domain) the well-known *sinc* [16] function with the hamming [17] window function [18]. Equation 7 is the *sinc* function and Equation 6 is the Hamming window function with the window interval $(-m, m)$

$$W_{hamming} = 0.54 + 0.46\cos\left(\frac{\pi x}{m}\right) \quad (6)$$

$$sinc(x) = \frac{\sin(\pi x)}{\pi x} \quad (7)$$

Although an ideal interpolation technique is expected not to alter any pattern within the image or introduce any artefact, most of the interpolation techniques usually alter some image features and also introduce artefacts when interpolated to reduce image resolution [14, 19]. Figures 3, 4, and 5 shows how the interpolation techniques discussed above can affect the features of a footwear impression image at different resolutions[4]. It is apparent from the undersampled (Figure 3) images that using different interpolation techniques produce slightly different images. Although these discrepancies are aesthetically undesirable and can hamper the performance of the model, we leverage such differences as an effective image augmentation technique as described in Section 5.

Figure 2: Original image without any interpolation. Zoom in to circumvent distortion introduced by the interpolation applied from the medium where this paper is being viewed

As we can see, all three interpolated images closely resemble the original (Figure 2) at a higher resolution (Figure 5) and at the same time their differences reduces. Comparing between the lowest resolution images (Figure 3), it is

---

[4]Please zoom in to see how the interpolation pattern gradually resembles the original image (Figure 2) with increasing resolution. Zooming is required as the images embedded in this paper go through arbitrary interpolation applied by your browser or PDF reader.

apparent that Nearest Neighbour (N) produces the most different looking downsampled image. Additionally, all the lower resolution impression images produce *descriptor D01: Bars* which are angled whereas the original (Figure 2) and higher resolution (zoom in for Figure 5) impressions have *descriptor D01: Bars* which are straight lines with an angle of $0°$. A study by [11] found that even a mildest quality loss of input images can greatly hamper the performance of a deep learning model. Glorot and Bengio [20] too found neural networks to be susceptible to image noise.

(a) $N_{(100 \times 262)}$ (b) $B_{(100 \times 262)}$ (c) $H_{(100 \times 262)}$

(d) $N_{(300 \times 786)}$ (e) $B_{(300 \times 786)}$ (f) $H_{(300 \times 786)}$

Figure 3: Interpolation samples with fixed aspect ratio and varying sizes

(a) $N_{(500 \times 1310)}$ (b) $B_{(500 \times 1310)}$ (c) $H_{(500 \times 1310)}$

(d) $N_{(700 \times 1834)}$ (e) $B_{(700 \times 1834)}$ (f) $H_{(700 \times 1834)}$

Figure 4: Interpolation samples with fixed aspect ratio and varying sizes (zoom in to see the original pattern)

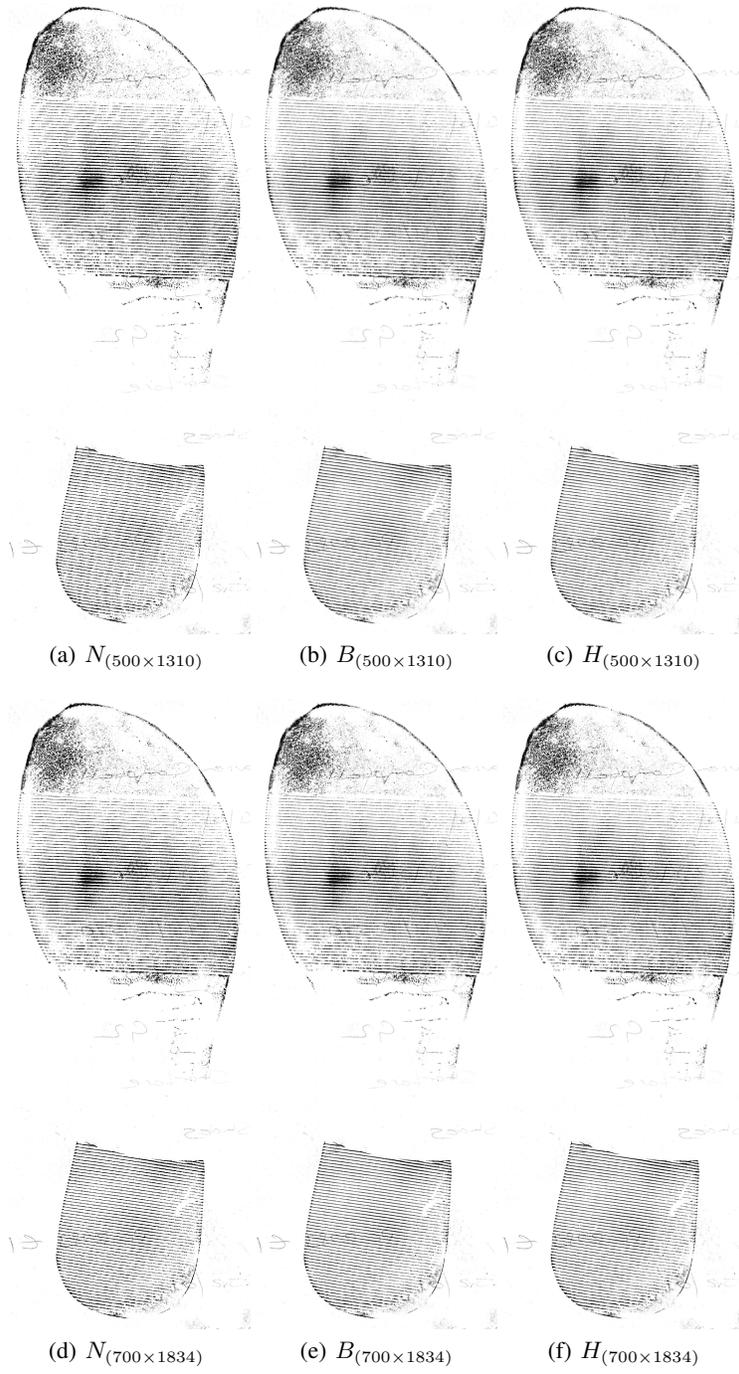(a) $N_{(1000 \times 2620)}$ (b) $B_{(1000 \times 2620)}$ (c) $H_{(1000 \times 2620)}$
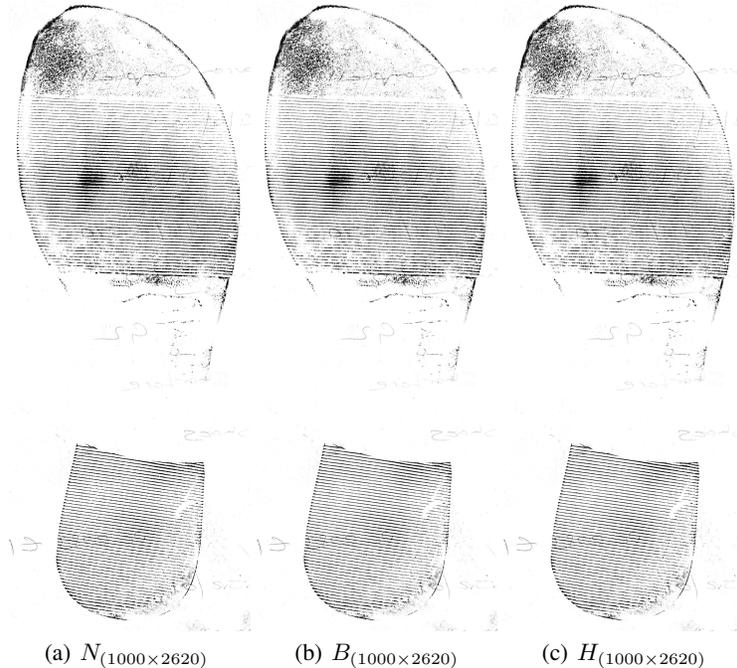
Figure 5: Interpolation samples with fixed aspect ratio and varying sizes (zoom in to see the original pattern)

5 Experimental Setup

In our experiments, we use *ResNet-50*, a popular 50 layer CNN architecture with residual connections [21], pre-trained on the ImageNet dataset [22] with a custom head initialised using the Glorot/Xavier initialisation [20] and optional, learnable preprocessing layer (see below). The head consists of an adaptive pooling layer, followed by two *BatchNorm* → *Dropout* → *Linear/Dense* blocks with *ReLU* non-linearity in between. The number of units in the non-output linear layer was set to 512, while the output layer has a total of 17 neurons with *sigmoid* activation functions, one per each *descriptor* type. The models are trained using *AdamW* [23], a stochastic gradient descent based backpropagation algorithm in two phases:

1. **Initial training**, where all the *ResNet-50* body layers are frozen and only the 2-layer head as well as the optional preprocessing layer are trained with the learning rate of $1e - 3$ and weight decay of 0.1.

2. **Fine-tuning**, where the whole network is trained using discriminative learning rates [24] of between $1e - 6$ and $1e - 4$ and weight decay of 0.1.

We have experimented with various combinations of the following:

1. **Loss function**: In addition to the default Binary Cross Entropy (BCE) loss, which in our experiments was always used in a cost-sensitive setting via class weighting (i.e. with the cost of misclassification being inversely proportional to class frequency in the training dataset), we have also used the Soft-F1 loss in an attempt to maximise both precision and recall directly within the model training process. The Soft-F1 loss is a simple generalisation of the F1 score obtained by replacing the number of True Positives (TP), False Positives (FP) and False Negatives (FN) with their probabilistic counterparts [25]:

$$TP = \sum_i y_i \hat{y}_i$$

$$FP = \sum_i (1 - y_i)\hat{y}_i$$

$$FN = \sum_i y_i(1 - \hat{y}_i)$$

9

where $y_i \in \{0, 1\}$ is the label for the ith data instance and $\hat{y}_i \in [0, 1]$ is the model prediction.

2. **Channel configuration**: All the original input images are greyscale (single-channel), yet the pre-trained model expects RGB/colour inputs (three-channels). The simplest and most popular approach to address this discrepancy is to collate three identical copies of the greyscale input. Since this approach seems wasteful, we have instead opted for various compositions of the three-channel input obtained via applying *different* interpolation techniques (see Section 4) to the high resolution input image – these are specified in Table 2.

3. **Preprocessing layer**: For the same reasons as described above, we have included a number of learnable preprocessing layers in our network. The rationale here was that the distribution of greyscale images, particularly when collating three different interpolated versions of each image into a single three-channel input, is different from the distribution of natural RGB images from the ImageNet dataset. The preprocessing layers we have used have been shown in Figure 6.

channels	R	G	B
B-B-B	Bilinear	Bilinear	Bilinear
B-H-N	Bilinear	Hamming	Nearest Neighbour
B-N-H	Bilinear	Nearest Neighbour	Hamming
H-B-N	Hamming	Bilinear	Nearest Neighbour
H-H-H	Hamming	Hamming	Hamming
H-N-B	Hamming	Nearest Neighbour	Bilinear
N-B-H	Nearest Neighbour	Bilinear	Hamming
N-H-B	Nearest Neighbour	Hamming	Bilinear
N-N-N	Nearest Neighbour	Nearest Neighbour	Nearest Neighbour

Table 2: Compositions of input channels via different combinations of interpolation techniques

Figure 6: Learnable preprocessing layers: (a) **cbn_1**: 1x1 Conv and BatchNorm, (b) **cbn_3**: 3x3 Conv and BatchNorm, (c) **inc**: inception-like transformation, (d) **inc_d**: dense inception-like transformation

The training dataset consisted of 33,757 greyscale images retrieved from the NFRC with the class distribution as shown in Figure 7a. As it can be seen, the classes are dominated by *D01*, *D02*, *D04* and *D07*, with *D01-01*, *D02-01*, *D05* and *D14* being the least frequent. The validation set consisted of 1,000 images retrieved from the same database with the class distribution as shown in Figure 7b. The image resolution ranged from 180×60 to $15,000 \times 7,000$ and has been depicted in Figure 8, where the whiskers represent Q_{05} and Q_{95}, and outliers have been omitted for presentation clarity.

(a) Training set

(b) Validation set

Figure 7: Class distribution

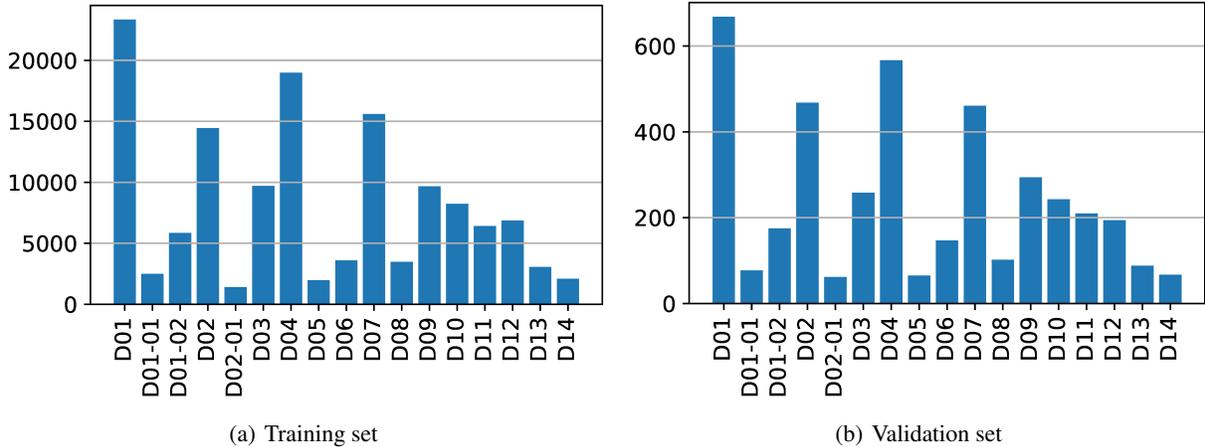Figure 8: Original input image resolution

6   Results

Table 3 contains the aggregated results of the total of 180 experiments run across 90 different combinations of hyper-parameters as specified in Table 2 and Figure 6. Each experiment has been repeated twice with random initialisation, and the average of these two runs was used to construct Table 3. In each of the experiments, we have trained a custom head on top of a fixed/frozen ImageNet pre-trained ResNet-50 for 10 epochs, followed by 40 epochs of finetuning of the whole network. We have opted for the Area Under the Precision-Recall Curve (*PRAUC*) as the performance metric in order to decouple the results from class-specific thresholds. *PRAUC* is a better measurement of performance of a binary classifier than the *AUC* of the *ROC* (receiver operating characteristic) curve [26, 27] since *ROC* is very sensitive to class imbalance and in our case class labels are heavily imbalanced.

The first thing to notice is that according to the results, preserving the aspect ratio of the input images (i.e. 352×144 resolution) **always** leads to better performance than when using squished images (i.e. 224×224 resolution, see the 'Δ' column in Table 3). This holds regardless of the type of resampling, preprocessing and loss function used. It appears that preserving the aspect ratio of the original images matters much more than higher horizontal resolution (i.e. 144 vs 224 pixels) and is the best way of spending a fixed computational budget (the number of input pixels in both cases is approximately equal ($224 \times 224 = 50,176$, $352 \times 144 = 50,688$). Although this performance difference might be partially attributed to the nature of our inputs (shoe impressions are long and thin, so squishing can introduce significant distortions), the same can be said about many other objects like people or vehicles. It is hence somewhat surprising that 224×224 is the default transfer learning setting for popular deep learning frameworks [28, 29, 30].

The average difference in *PRAUC* between the two resolutions (everything else being equal) is 0.0189 (2.8%), while the maximum difference over all 180 runs reaches 0.0415 (6.5%). To put these numbers in context, the average difference between any two randomly initialised runs of each experiment is 0.0051, while the maximum difference is 0.0206. Thus the observed effect is unlikely to be a random fluctuation. For this reason, we limit further analysis to the results for the 352×144 (and higher) resolutions, preserving the input aspect ratio.

11

| channels | preprocessing | PRAUC | | | | | |
| | | BCE LOSS | | | F1 LOSS | | |
		352x144	224x224	Δ	352x144	224x224	Δ
N-N-N	cbn_1	0.7200	0.6969	0.0231	0.6927	0.6712	0.0215
N-N-N	cbn_3	0.7171	0.6968	0.0203	0.6839	0.6741	0.0098
N-N-N	inc	0.7205	0.6990	0.0215	0.6861	0.6706	0.0155
N-N-N	inc_d	0.7215	0.6945	0.0270	0.6938	0.6712	0.0226
N-N-N	no_tfm	0.7135	0.6940	0.0195	0.6902	0.6728	0.0174
H-H-H	cbn_1	0.7113	0.6958	0.0155	0.6876	0.6694	0.0182
H-H-H	cbn_3	0.7172	0.6939	0.0233	0.6852	**0.6778**	**__0.0074__**
H-H-H	inc	0.7133	0.6931	0.0202	0.6878	0.6721	0.0157
H-H-H	inc_d	**0.7259**	**0.7044**	0.0215	0.6923	0.6753	0.0170
H-H-H	no_tfm	0.7169	0.6995	0.0174	0.6874	0.6732	0.0142
B-B-B	cbn_1	0.7177	0.6956	0.0221	0.6872	0.6733	0.0139
B-B-B	cbn_3	0.7165	0.6974	0.0191	0.6892	0.6710	0.0182
B-B-B	inc	0.7176	0.6976	0.0200	0.6914	0.6705	0.0209
B-B-B	inc_d	0.7198	0.699	0.0208	0.6960	0.6679	**0.0281**
B-B-B	no_tfm	**__0.7061__**	0.6913	0.0148	0.6895	0.6649	0.0246
N-B-H	cbn_1	0.7198	0.6981	0.0217	0.6879	0.6702	0.0177
N-B-H	cbn_3	0.7118	0.6958	0.0160	0.6879	0.6690	0.0189
N-B-H	inc	0.7127	0.6957	0.0170	0.6851	0.6703	0.0148
N-B-H	inc_d	0.7204	0.6931	0.0273	0.6899	0.6725	0.0174
N-B-H	no_tfm	0.7158	0.6972	0.0186	0.6878	0.6705	0.0173
N-H-B	cbn_1	0.7118	0.6919	0.0199	0.6894	0.6665	0.0229
N-H-B	cbn_3	0.7201	0.7005	0.0196	0.6890	0.6732	0.0158
N-H-B	inc	0.7161	0.6973	0.0188	0.6943	0.6701	0.0242
N-H-B	inc_d	0.7189	0.6988	0.0201	0.6965	0.6750	0.0215
N-H-B	no_tfm	0.7144	0.6971	0.0173	0.6896	0.6675	0.0221
B-H-N	cbn_1	0.7167	0.6953	0.0214	0.6893	0.6698	0.0195
B-H-N	cbn_3	0.7200	0.7007	0.0193	0.6895	0.6734	0.0161
B-H-N	inc	0.7206	**__0.6907__**	**0.0299**	0.6865	0.6724	0.0141
B-H-N	inc_d	0.7156	0.6976	0.0180	0.6896	0.6697	0.0199
B-H-N	no_tfm	0.7092	0.6972	**__0.0120__**	0.6914	0.6720	0.0194
B-N-H	cbn_1	0.7169	0.7011	0.0158	0.6863	0.6709	0.0154
B-N-H	cbn_3	0.7187	0.6991	0.0196	0.6866	0.6670	0.0196
B-N-H	inc	0.7177	0.6977	0.0200	0.6854	0.6761	0.0093
B-N-H	inc_d	0.7232	0.7016	0.0216	0.6874	**__0.6642__**	0.0232
B-N-H	no_tfm	0.7149	0.6968	0.0181	0.6891	0.6711	0.0180
H-N-B	cbn_1	0.7178	0.6982	0.0196	0.6875	0.6704	0.0171
H-N-B	cbn_3	0.7124	0.6963	0.0161	0.6875	0.6667	0.0208
H-N-B	inc	0.7134	0.6985	0.0149	**__0.6849__**	0.6728	0.0121
H-N-B	inc_d	0.7176	0.6984	0.0192	0.6896	0.6693	0.0203
H-N-B	no_tfm	0.7107	0.6987	0.0120	0.6897	0.6678	0.0219
H-B-N	cbn_1	0.7224	0.6929	0.0295	0.6889	0.6698	0.0191
H-B-N	cbn_3	0.7171	0.6939	0.0232	0.6850	0.6732	0.0118
H-B-N	inc	0.7146	0.7007	0.0139	0.6861	0.6685	0.0176
H-B-N	inc_d	0.7242	0.6975	0.0267	**0.6966**	0.6751	0.0215
H-B-N	no_tfm	0.7118	0.6915	0.0203	0.6865	0.6706	0.0159
μ		0.7167	0.6969	0.0199	0.6889	0.6709	0.0180

Table 3: Performance based on the Area Under the Precision-Recall Curve (*PRAUC*). 'Δ' denotes difference between the two resolutions. *Max* for each column in **bold**, *min* in **__underline__**.

A similar observation can be made when considering the loss function. *BCE* **consistently** outperforms the *F1 LOSS*, with the average *PRAUC* difference of 0.0269 and the maximum difference of 0.0374. Despite attractive theoretical properties of the *F1 LOSS* as discussed in Section 5, the *BCE* loss proved to be a much better choice in practice. For this reason we are not considering the *F1 LOSS* in the subsequent analysis.

rank	channels	PRAUC (↓)
1	N-N-N	0.7185
2	B-N-H	0.7183
3	H-B-N	0.7180
4	H-H-H	0.7169
5	B-H-N	0.7164
6	N-H-B	0.7163
7	N-B-H	0.7161
8	B-B-B	0.7155
9	H-N-B	0.7144
max(Δ)		0.0041

Table 4: Performance for 352×144 input resolution, *BCE* loss, and various input channel configurations, averaged over preprocessing methods, sorted by *PRAUC*.

rank	preprocessing	PRAUC (↓)
1	inc_d	0.7208
2	cbn_1	0.7171
3	cbn_3	0.7168
4	inc	0.7163
5	no_tfm	0.7126
max(Δ)		0.0082

Table 5: Performance for 352×144 input resolution, *BCE* loss, and various preprocessing layers, averaged over input channel configurations, sorted by *PRAUC*.

The average difference in *PRAUC* among various combinations of input channels (Table 4) is much less pronounced. The approach of creating an input to the network by 'sandwiching' the outputs of three different interpolation methods rather than simply creating three identical channels (i.e. *B-N-H* vs *N-N-N*), doesn't seem to affect the *PRAUC* much, with the difference between the best and worst performing approach being 0.0041. Nevertheless, it's worth noting that the *B-B-B* approach, which is the default in the existing deep learning frameworks [28, 29, 30], is one of the worst performing. This seems to confirm the intuition that the blurring effect characteristic for bilinear interpolation, tends to make discrimination between different types of descriptors more challenging. Another observation is that the influence of input channel ordering on *PRAUC* can be almost as big as the difference between the best and worst performing approach (the difference between *B-N-H* and *H-N-B* is 0.0039). This is somewhat surprising as in theory, the learnable preprocessing layer should be able to 'swap' the input channel order if needed. However, in the context of the average difference between any two randomly initialised runs of each experiment, which as mentioned earlier was 0.0051, the results given in Table 4 need to be declared inconclusive.

The influence of the learnable preprocessing layer on *PRAUC* is more substantial. As it can be seen in Table 5, the difference between the dense inception-like transformation (*inc_d*) and no transformation at all (*no_tfm*) reaches 0.0082. Since *no_tfm* is the worst performing approach in our experiments, we conclude that using some kind of learnable preprocessing is beneficial.

channels	preprocessing					μ	max(Δ)
	cbn_1	cbn_3	inc	inc_d	no_tfm		
B-B-B	0.7177	0.7165	0.7176	<u>0.7198</u>	0.7061	0.7155	0.0137
B-H-N	0.7167	0.7200	**0.7206**	0.7156	0.7092	0.7164	0.0114
B-N-H	0.7169	0.7187	0.7177	<u>0.7232</u>	0.7149	0.7183	0.0083
H-B-N	**0.7224**	0.7171	0.7146	<u>0.7242</u>	0.7118	0.7180	0.0124
H-H-H	0.7113	0.7172	0.7133	<u>**0.7259**</u>	**0.7169**	0.7169	0.0146
H-N-B	<u>0.7178</u>	0.7124	0.7134	0.7176	0.7107	0.7144	0.0071
N-B-H	0.7198	0.7118	0.7127	<u>0.7204</u>	0.7158	0.7161	0.0086
N-H-B	0.7118	<u>**0.7201**</u>	0.7161	0.7189	0.7144	0.7163	0.0083
N-N-N	0.7200	0.7171	0.7205	<u>0.7215</u>	0.7135	**0.7185**	0.0080
μ	0.7171	0.7168	0.7163	<u>0.7208</u>	0.7126	0.7167	
max(Δ)	0.0111	0.0083	0.0079	0.0103	0.0076		

Table 6: *PRAUC* for 352×144 input resolution, BCE loss, and various combinations of pre-first layer transformations and input channel configurations. The highest score in each column in **bold**. The highest *PRAUC* in each row in <u>underline</u>.

Table 6 presents the breakdown of the results by preprocessing layers and input channel configurations. As it can be seen, *inc_d* gives the highest PRAUC on average (0.7208), is the best preprocessing method for 6 out of 9 input channel configurations, and second best for additional 2. It is harder to identify the best performing channel configuration as none of them seems to be dominating across different preprocessing layers. However, looking at preprocessing and channel configuration jointly, *inc_d* with *H-H-H* gives the highest *PRAUC* of 0.7259, which is 0.0198 more than the worst performing combination (*no_tfm* with *B-B-B*) and 0.0092 more than the average across all the entries in Table 6.

	preprocessing					
channels	**cbn_1**	**cbn_3**	**inc**	**inc_d**	μ	$\max(\Delta)$
B-B-B	<u>**0.7450**</u>	0.7407	0.7363	0.7369	0.7397	0.0086
B-H-N	0.7394	0.7334	0.7371	<u>0.7400</u>	0.7375	0.0067
B-N-H	<u>0.7399</u>	0.7365	0.7375	<u>0.7399</u>	0.7384	0.0035
H-B-N	0.7407	0.7409	0.7377	<u>**0.7477**</u>	0.7410	0.0070
H-H-H	**0.7450**	0.7404	0.7361	0.7441	0.7414	0.0089
H-N-B	0.7405	<u>0.7425</u>	0.7363	0.7411	0.7401	0.0062
N-B-H	0.7383	0.7311	**0.7436**	0.7432	0.7390	0.0125
N-H-B	0.7367	**0.7429**	0.7415	<u>0.7451</u>	**0.7415**	0.0084
N-N-N	0.7440	0.7303	0.7399	<u>0.7459</u>	0.7400	0.0156
μ	0.7410	0.7376	0.7385	<u>0.7423</u>	0.7399	
$\max(\Delta)$	0.0083	0.0126	0.0075	0.0090		

Table 7: *PRAUC* for 464×192 input resolution, BCE loss, and various combinations of preprocessing layers and input channel configurations. The highest score in each column in **bold**. The highest *PRAUC* in each row in <u>underline</u>.

In Table 7 we report the results of a similar experiment, this time with the resolution of the input images increased to 464×192. This lead to a significant increase of the *PRAUC* across all tested combinations of preprocessing layers and input channel configurations, with the minimum, average and maximum difference of 0.0132, 0.0221 and 0.0337 respectively. As before, *inc_d* is the dominating preprocessing method, while none of the input channel configurations seems to be a clear winner. Note, that in Table 7, *B-B-B* is no longer as strongly dominated by other input channel configurations as it was the case at lower input resolutions due to the blurring effect now being less severe.

	preprocessing
channels	**inc_d**
B-B-B	0.7715
B-H-N	**0.7736**
B-N-H	0.7718
H-B-N	0.7676
H-H-H	0.7681
H-N-B	0.7722
N-B-H	0.7695
N-H-B	0.7687
N-N-N	0.7696
μ	0.7703
$\max(\Delta)$	0.0060

Table 8: *PRAUC* for 928×384 input resolution, BCE loss, and various combinations of input channel configurations. The highest *PRAUC* in **bold**.

In out final experiment, we have investigated increasing the input resolution to 928×384. As shown in Table 8, this has resulted in further improvement in terms of *PRAUC* reaching 0.0280 on average. It is also apparent that with the increase in the input resolution, the importance of interpolation diminishes – the maximum difference among all the interpolation methods is 0.0060, albeit achieved at significantly increased computational cost.

6.1 Error analysis

In Figure 9 we depict the *PRAUC* of our best model from Table 8, broken down by class/descriptor. As it can be seen, some descriptors seem to be relatively easy to classify; *D01: Bar*, *D02: Circular*, *D04: 4 sided*, *D07: Complex* and

D08: Zigzag all have $PRAUC > 0.9$, which is to be expected as these descriptors are relatively clear cut (see Table 1). At the other end of the spectrum, *D05: 5 sided* followed by *D13: Hollow* and *D01-02: Curved-wavy* are the most challenging. Note, that *D05* is not only the least frequent in the dataset as per Table 7 (we've counteracted this by using class-weighting in the *BCE* loss), but it is also one of the subtler descriptors in general. As it can be seen in Table 1, *D05* can for example be a rectangle with one of the corners 'cut off', hence easy to confuse with *D04: 4 sided*. In a similar vein, *D13: Hollow* can easily be confused with a circle (*D02: Circular*), triangle (*D03: 3 sided*), square/rectangle (*D04: 4 sided*) etc. Some shapes on an impression can also represent multiple descriptors, for example *D09: Text* and *D10: Logo* will often overlap. Some overlays may be more complex such as *D03: 3 sided* and *D13: Hollow*. There may even be some examples of nested overlap such as *D02-01: Target* (which implies *D02: Circular*) and *D13: Hollow*, as a circle with the centre missing is both a target and hollow. *D14: Plain* is unusual in that when it applies to part of the shoe, it excludes the other descriptors from that area.

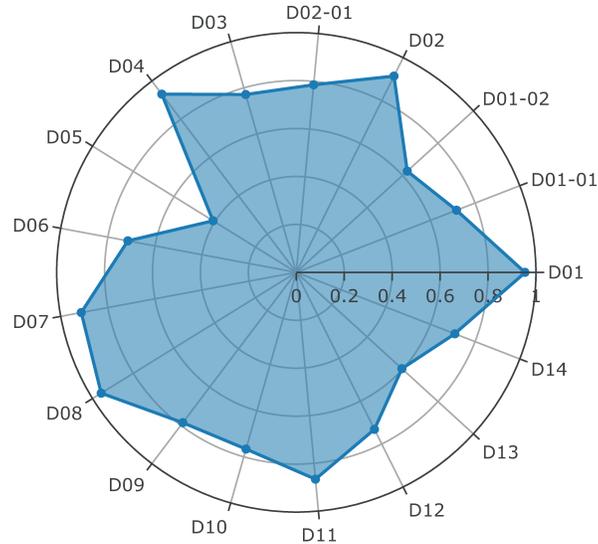| Descriptor	PRAUC
D01	0.9541
D01-01	0.7172
D01-02	0.6267
D02	0.9145
D02-01	0.7856
D03	0.7713
D04	0.9314
D05	0.4086
D06	0.7151
D07	0.9129
D08	0.9575
D09	0.7859
D10	0.7664
D11	0.8665
D12	0.7315
D13	0.5964
D14	0.7097

Figure 9: Per class PRAUC

In order to investigate this issue further, in Figure 10 we show the confusion matrix generated for the validation dataset. For each validation image, if the predicted score for a descriptor which is not present in the image (false positive) exceeds the score for a descriptor which is in the image (true positive), then the two are considered confused. An example is given in Table 9. The actual labels are *D02*, *D05* and *D06*. Since the score for *D02* is the highest, 1 would be added to the diagonal entry for this descriptor. However, as the score for *D04* (false positive) is higher than that for *D05* and *D06*, these are considered confused (i.e. either or both of *D05* and *D06* are classified as *D04*) and hence $1/2$ (i.e. one over the number of potentially misclassified descriptors) is added to the entries *D05-D04* and *D06-D04* of the confusion matrix.

Descriptor	D01	D02	D03	D04	D05	D06
Label	0	1	0	0	1	1
Score	0.3	0.9	0.2	0.8	0.6	0.6

Table 9: Example prediction to illustrate confusion matrix calculation

As it can be seen in Figure 10, *D01* is the most frequently misclassified descriptor which can be partially explained by its prevalence in the dataset. In Figure 11(a) we show an example of a *D01: Bar* in the top right corner of the print, which has not been detected by our model. At the same time, the model detected *D13: Hollow* in the locations shaded in orange, although this particular shoeprint impression has not been labelled with *D13* by the human expert. However, due to wear, some of the 6 sided shapes (*D06*) closed, and indeed now fit the description of *D13*. Another example of undetected *D01* is given in Figure 11(b), where in addition *D11: Lattice* has been misclassified as *D04: 4 sided* in the areas highlighted by the heatmap – the lattice indeed consists of 4 sided 'cells'. In both of these examples, it is actually difficult to state which descriptor *D01* was confused with; it appears that *D01* was simply not detected, yet this would still be recorded in the confusion matrix due to the way in which the matrix was derived.
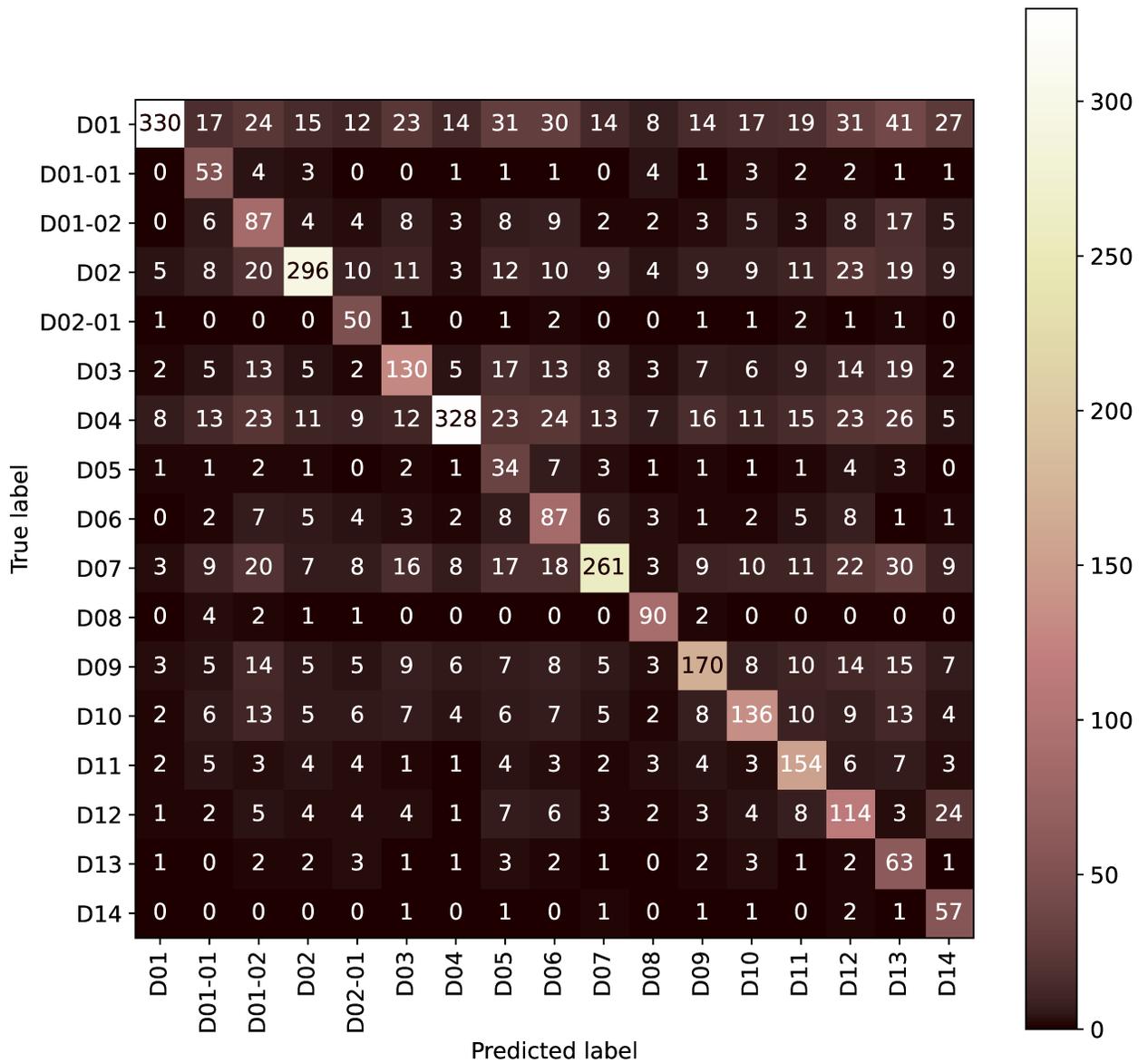
Figure 10: Confusion matrix

Another descriptor worth looking at is *D05*, which as mentioned before is the most rare in the dataset and has the lowest *PRAUC*. *D05* is most often misclassified as *D06* or *D12*, while at the same time *D03*, *D04* and *D07* are most often misclassified as *D05*. An example can be seen in Figure 11(c).

It is worth noting, that all three examples of errors in Figure 11 have been selected on the basis of the highest loss (i.e. they are as bad as it gets). It is reassuring that these do not result from the model behaving in an unexpected fashion, but are rather due to ambiguity in the inputs that may even cause disagreement between expert users, requiring resolving by expert panel.

(a) $D01, D06 \rightarrow$ (b) $D01, D11 \rightarrow D04$ (c) $D05 \rightarrow D06$
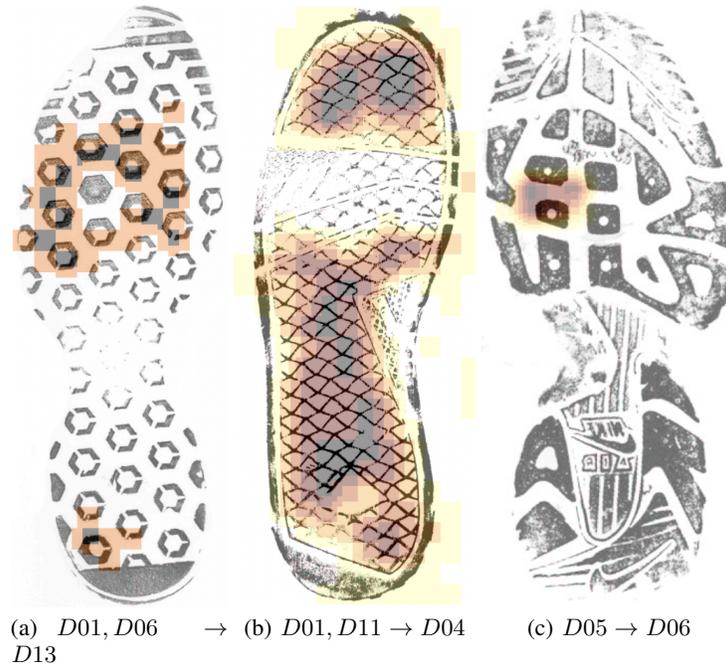$D13$

Figure 11: High loss misclassification examples

7 Conclusions

The *descriptor* identification task we have approached in this study is of great significance for the forensic practitioners in the UK and beyond. The *descriptors* are an agreed standard for coding footwear patterns for different forces in the UK are in active use. Although a human performance benchmark is not available at this time, our model performs well with the *PRAUC* of over 0.77. The mistakes that the model tends to make are mostly justifiable, either by ambiguity or by overlaps in the input patterns, and are not unlike what an inexperienced human would make. The system that we have built has been deployed for testing by selected police forces.

In the process of building the model, we have experimented with a number of ways of feeding greyscale impressions to the ImageNet (RGB) pre-trained network. Our findings can be summarised as the following 'best practices':

- Preserve the aspect ratio of the input images. This seems particularly important if the object of interest (a shoeprint in our case) has aspect ratio significantly different than $1:1$ (i.e. 'long and thin' or 'short and fat'). This advice goes against the common practice in the computer vision community of 'squishing' the input images to make them square in order to use ImageNet pre-trained models. This is unnecessary as current deep learning frameworks allow one to feed rectangular images to the ImageNet pre-trained models out of the box.

- Use as high input resolution as practical. In our experiments increasing the input resolution always led to higher *PRAUC* albeit at the cost of significantly increased computations, which is an obvious constraint. The original resolution of the input images can also be a limitation as there's little point in upscaling such images.

- Use different interpolation methods to construct the three input channels from greyscale images. Although the effect of this approach that we have observed was modest, it was positive nevertheless. It also seems that using the Nearest Neighbour interpolation as one of the input channels is beneficial, while using three identical channels obtained via Bilinear interpolation is detrimental, particularly at lower input resolutions.

- Use a learnable preprocessing layer. In our experiments, these additional computations played a crucial role in the process of adapting greyscale inputs to be used with a colour-image pre-trained network, regardless of the interpolation method used. Learnable preprocessing combined with different interpolation methods to construct the three input channels gave the best results.

Acknowledgements

References

[1] Bluestar Software Limited. National Footwear Solutions (the NFRC and NFD), 2020. URL `https://bluestar-software.co.uk/products/forensic-intelligence/`.

[2] Robert Milne. *Forensic intelligence*. CRC Press, 2012.

[3] Hannah Larsen and Matthew R Bennett. Recovery of 3d footwear impressions using a range of different techniques. *Journal of Forensic Sciences*, 2020.

[4] Matthew R Bennett and Marcin Budka. *Digital technology for forensic footwear analysis and vertebrate ichnology*. Springer, 2018.

[5] Bluestar Software Limited. National Footwear Solutions (the NFRC and NFD), 2020. URL `https://bluestar-software.co.uk/products/uk-national-solutions/`.

[6] William J Bodziak. *Footwear impression evidence: detection, recovery and examination*. CRC Press, 1999.

[7] National Policing Improvement Agency. Footwear marks recovery manual, 2007. URL `http://library.college.police.uk/docs/appref/NPIA-(2007)-Footwear-Marks-Recovery-Manual.pdf`.

[8] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[9] Home Office. Home Office, Police Workforce, England and Wales, 2019. URL `https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/831726/police-workforce-mar19-hosb1119.pdf`.

[10] Roberto Brunelli. *Template matching techniques in computer vision: theory and practice*. John Wiley & Sons, 2009.

[11] Michal Koziarski and Boguslaw Cyganek. Impact of low resolution on image recognition with deep neural networks: An experimental study. *International Journal of Applied Mathematics and Computer Science*, 28(4), 2018.

[12] Suresh Prasad Kannojia and Gaurav Jaiswal. Effects of varying resolution on performance of cnn based image classification: An experimental study. *International Journal of Computer Sciences and Engineering*, 6(9):451–56p, 2018.

[13] Philippe Thévenaz, Thierry Blu, and Michael Unser. Image interpolation and resampling. *Handbook of medical imaging, processing and analysis*, 1(1):393–420, 2000.

[14] Philippe Thévenaz, Thierry Blu, and Michael Unser. Interpolation revisited [medical images application]. *IEEE Transactions on medical imaging*, 19(7):739–758, 2000.

[15] PR Smith. Bilinear interpolation of digital images. *Ultramicroscopy*, 6(2):201–204, 1981.

[16] Philip M Woodward and Ian L Davies. Information theory and inverse probability in telecommunication. *Proceedings of the IEE-Part III: Radio and Communication Engineering*, 99(58):37–44, 1952.

[17] Ralph Beebe Blackman and John Wilder Tukey. The measurement of power spectra from the point of view of communications engineering—part i. *Bell System Technical Journal*, 37(1):185–282, 1958.

[18] Erik HW Meijering, Wiro J Niessen, and Max A Viergever. Quantitative evaluation of convolution-based methods for medical image interpolation. *Medical image analysis*, 5(2):111–126, 2001.

[19] Jeffrey Tsao. Interpolation artifacts in multimodality image registration based on maximization of mutual information. *IEEE transactions on medical imaging*, 22(7):854–864, 2003.

[20] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[22] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

[24] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, 2018.

[25] Elad Eban, Mariano Schain, Alan Mackey, Ariel Gordon, Ryan Rifkin, and Gal Elidan. Scalable learning of non-decomposable objectives. In *Artificial intelligence and statistics*, pages 832–840. PMLR, 2017.

[26] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432, 2015.

[27] Akanda Wahid-Ul-Ashraf, Marcin Budka, and Katarzyna Musial. How to predict social relationships—physics-inspired approach to link prediction. *Physica A: Statistical Mechanics and its Applications*, 523:1110–1129, 2019.

[28] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[29] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[30] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.