Background: Footwear and Forensic Analysis

In forensic science, shoe prints and outsole characteristics fall into the category of pattern evidence. When a shoe print or impression is found at a crime scene, the investigator may ask a series of questions. Initially, it may be important to determine the make and model of the shoe, which may help detectives locate comparison shoes from suspects. Later in the investigation, the forensic examiner may consider individualizing characteristics found in the print; that is, small defects that make it possible to tie a specific shoe to the print left at the scene. In cases where such individualizing characteristics are not considered (estimated at 95% of cases in the United States according to some experts[1]), it is important to be able to assess the probability that the specific model of shoe which made the print would be found in the suspect's posessions. This question is much more difficult than identifying the make and model of the shoe, because it requires that the forensic examiner have access to a database containing information about the frequency of shoes in the local population, where the local population itself may be difficult to define. Any tractable solution to the problem of assessing the random match probability of a shoeprint based only on class characteristics[Bodziak, 2000] (make, model, and other characteristics determined during the manufacturing process) requires a way to assemble this database: an automated solution to efficiently classify many types of shoes within a common system.

Classification

Visual classification is a complex task that our brains have been trained to do very well. Our eyes detect a large variety of features of an object, including color, shape, and texture, and send that information to our brain. Our brain then learns which combinations of features to associate with a given label, and should be able to apply those rules to future objects with similar characteristics. For example, an orange caterpillar and a baby carrot may be of similar color, shape, and size, but one is distinctly more fuzzy than the other. Thus, our brains learn that when faced with a small, cylindrical orange object, texture becomes an important feature when assigning a label to that object (which keeps us from accidentally ingesting caterpillars).

Convolutional Neural Networks (CNNs)

While our brains are adept at parsing images and classifying the objects within them, the task has proved much more difficult for computers. Convolutional neural networks (CNNs) are a tool for supervised deep-learning that have become standard in recent years for automatic image classification. CNNS use combinations of convolutional and pooling hidden layers to filter raw information into features, which are then fed into densely connected layers which are trained to associate given sets of features with their desired labels. This translation-invariant automated classification mimics the human eye-to-brain classification process and has become one of the most widely used machine learning techniques for image classification.

Pre-trained CNNs

[1] Leslie Hammer, presentation to CSAFE on March 5, 2018

I don't love this paragraph. Not sure how much to include, awkward flow. Pre-trained CNNs are CNNs that have been trained on a standard data set. The standard data set comes from ImageNet, a database containing over 14 million images in about 22,000 categories (called "synsets", short for "synonym sets"). The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) was established in 2010 as a contest for CNN accuracy on a specific subset of ImageNet. Various CNN structures are tested on about 1.2 million images spanning 1,00 categories. These categories range from natural and man-made objects (e.g., daisy, chainsaw) to living creatures (e.g., ring-tailed lemur, sea lion, and dingo). There are also many categories which require subtle distinctions, such as differentiating between a grass snake and a vine snake. *Something about the best CNNs for this task are the ones that are famous.* Some of the most well-known pre-trained CNNs include AlexNet, GoogLeNet/Inception, VGG, and ResNet. *Can use just structure and train weights yourself, use fully trained model to reproduce ILSVRC results, or just use pre-trained weights for feature detection*

VGG16 Architecture The main difference between different CNNs is their structure, meaning the number of layers they contain and the pattern those layers are in. In our research, we have tested a few pre-trained CNNs, and we are currently using VGG16. Developed by Oxford's Visual Graphics Group, VGG16 has 16 "functional" (i.e., convolutional and densely connected) layers and 5 max-pooling layers, which function more to alter the structure of the information at each step.

Filters and Convolution Convolutional Neural Networks are named to highlight their use of convolution to extract information from an image. To a computer, an image is stored as a 3-dimensional array with a length and width corresponding to its number of pixels and a depth of 3 to represent the typical RBG color channels. A single convolutional filter is a small array (say 5x5x3) of real valued weights that represents some feature of the image. When a filter is applied to a portion of the image, the weights are multiplied with the image values and all values are summed, which returns a single value associated with how strong the presence of the feature is for that part of the image. When applied over an entire image, the resulting matrix of values maps the strength of the feature across the entire image. A convolutional layer of a CNN takes a large number of these filters and passes them over the image to return one feature map per filter.

In the case of VGG16, early convolutional layers contain 64 features that primarily detect colors and edge patterns. Later convolutional layers of VGG16, in contrast, contain 512 filters that represent much more complex features, like animal fur patterns or distinct bird heads.

Max-Pooling Max-pooling is a technique to reduce the size, and therefore computational load, of feature maps through structured down-sampling. Max-pooling layers apply a maximum function over adjacent regions of a feature map (like using a sliding window) to encode the important information of how strongly a feature was activated in a given region of the image while simultaneously reducing redundant or unneccessary information about smaller

activations. For example, taking 2x2 pieces of a feature map and keeping only the largest of the four values reduces the size of the feature map by a factor of 4! Max-pooling is also beneficial in that it allows CNN "vision" to be translation invariant, because it emphasizes the relative position of a feature rather than its absolute position. VGG16 follows groups of 2 or 3 convolutional layers with a max-pooling layer, which ultimately takes in initial feature maps of size 224x224 and ends with maps of size 7x7.

Densely Connected Layers Densely connected layers are typically the final layers in a CNN. These layers form the meaningful connection between the features of an image (detected by convolutional and max-pooling layers) and the corresponding labels associated with the image. These layers act like the human brain: just as we learned which combinations of features should be associated with a given label, densely connected layers use real-valued weights to represent these associations. For example, if we see an item that is orange, small, and fuzzy, we are taught to call it "caterpillar". Fuzzzy is not a feature we meaningfully associate with a baby carrot, so there is little connection between the feature "fuzzy" and the label "carrot". Similarly, in CNNs, each final feature is connected to each label through a weight (hence the name "densely connected"), and those weights are learned through the training process (using an algorithm called back-propogation) to minimize loss and thus improve classification accuracy.

Using a Pre-Trained CNN for a New Task *I went off on a tangent and am purposely not restructuring this paragraph yet. Sorry.* As we have just seen, convolutional layers and max-pooling layers in a CNN are analogous to the human visual perception process, and densely connected layers behave like the human brain. In short, the approach to classifying an image is to detect the features in the image (like our eyes) and then assign labels to combinations of those features (brain). This analogy is also appropriate because it reflects the difficulty of the task: it takes many years and a significant amount of effort for humans to learn how to distinguish a large variety of features and also to connect those features to labels that are often complex, hierarchical, and subtle. Similarly, training a CNN is no small task. VGG16, in particular, has over 14.7 million trainable parameters in its "eyes" alone. Luckily, CNNs offer one benefit that humans do not: you can utilize the eyes and replace the brain for new tasks. In terms of CNNs, it is possible to build a CNN that uses the weights already trained on over 1.2 million images in the convolutional layers, and then only retrain a new classifier for any new classification task. This reduced task brings the number of required training images down from millions to only thousands. Furthermore, this kind of approach is quite reasonable when considering what the CNN was originally trained to classsify. Since the 1,000 categories from the ILSVRC span a huge variety of natural and unnatural objects, we can likely trust that the features detected by the pre-trained CNN to be diverse enough to be applied to a new task.

Outsole Class Characteristics *Include something that explains what class characteristics are?* *Transition from CNN to shoes* In the beginning of this project, the literature (source IDK) indicated that geometric shapes are

unique and well-defined enough to provide most of the necessary information to classify a shoe outsole. We have since settled on nine geometric categories, modified from (source): bowtie, chevron, circle, line, polygon, quadrilateral, star, text, and triangle. Many of these categories are self-explanatory, such as circle and triangle. Polygon is a catch-all for pentagons, hexagons, and octagons. Star is any concave shape, including X- or plus-shapes. Line is a difficult distinction, as most shapes are simply a combination of lines, but in this case it is reserved for cases where the other categies do not readily apply, such as repeating patterns of lines that are not distinctly quadrilaterals.

Defining categories this way does not remove all ambiguities. The best example lies in considering text. The letter "v" can easily be considered a chevron, and the letter "o" is clearly a circle. However, text is also an important category to encompass the variety of ways text appears on footwear outsoles, and it is not necessarily helpful (or possible) to try to categorize every shape in text into another category. Many of the ambiguities that arise can be solved by applying multiple labels to an image, but some shapes also do not fit into any categories. Applying comprehensive and consistent labels to difficult or ambiguous shapes is the most difficult part of this process.

Data Collection and Structure

Pick a date cut-off and use information true as of that date. *Build up to final model somehow? Start with

Thousands of outsole images were web-scraped from Zappos.com, a large online shoe retailer. These images were then uploaded for use in a tool called LabelMe, a labeling/annotating interface [connected to] Matlab, which allows users to easily select and label regions of an image. To date, about [2,200] shoes have been labeled, yielding about [24,000] multi-label images. *Image processing?* *Number of images per category?* To train the CNN, data was split such that 70% went to training, and 15% each to validation and test data.

Model Training

Model training was conducted using the *keras* package in R, which provides an interface to the neural network API of the same name which is written in Python, with a TensorFlow computational backend.

I want to include the following ideas, but just don't want to spend the time writing them out when I'm unsure about the format.

-Use VGG16 Convolutional base and train new classifier, done by getting features from conv base and training only dense layers -Model training parameters (e.g., augmentation parameters, drop-out rate) -Model predictions are multi-label binary, probabilities don't add to one

Model Performance

I feel like graphs will speak more here -Accuracy and loss during training -Examples of prediction -Ways to measure accuracy (TPR, FPR, ROC/AUC) -Interesting case studies

References

William J. Bodziak. *Footwear Impression Evidence: Detection, Recovery, and Examination*. CRC Press, Boca Raton, Florida, 2000. ISBN 0-8493-1045-8. 00319.