UNIVERSIDADE
BEIRA INTERIOR

Deep Adversarial Frameworks for Visually Explainable Periocular Recognition

João Pedro da Cruz Brito

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Dr. Hugo Pedro Martins Carriço Proença

Covilhã, junho 2021

Acknowledgements

Throughout the past years, a select group of people became increasingly more fundamental to my success. Therefore, this section recognises the many ways in which I received valuable help and, despite not fully grasping these peoples' contributions, the timeless nature of written words will perpetuate my utmost gratitude.

First and foremost, the role played by my supervisor can not go unnoticed. From the original proposal, through countless revisions of the work plan, until the final, stable version, Professor Hugo Proença made sure to accompany his sharp recommendations with adequate doses of patience and understanding. I always felt more enlightened and with a clearer sense of direction after our meetings. For the knowledge, principles and methodologies passed on to me, I express my deepest appreciation.

To my wonderful parents and brother, I express the most profound sentiment of gratitude. Never, in my academic life, did I feel like my potential successes were being limited by external factors. All the conditions that lay the foundation for academic success were made available to me, without hesitation. By adding the love and support bestowed on to my brother and me, this journey was made undeniably easier. For that, and so much more, I convey the most sincere feelings of gratitude.

To my friends, I appreciate the ways in which my mind was taken off work, thanks to the jokes and funny episodes I fondly look back to. Despite the unexpected outbreak of a pandemic, they always managed to make me feel better, even during the most sensitive periods of the past year. For embodying the values of friendship, I have nothing but admiration.

Lastly, but definitely not least, my colleagues at Socialab deserve some recognition for the productive and peaceful environment they made possible in the lab. Some constructive criticism and cheerful interactions made my trips to the workplace a worthy experience.

Abstract

Machine Learning (ML) models have pushed state-of-the-art performance closer to (and even beyond) human level. However, the core of such algorithms is usually latent and hardly understandable. Thus, the field of Explainability focuses on researching and adopting techniques that can *explain* the reasons that support a model's predictions. Such explanations of the decision-making process would help to build trust between said model and the human(s) using it. An explainable system also allows for better debugging, during the training phase, and fixing, upon deployment. But why should a developer devote time and effort into refactoring or rethinking Artificial Intelligence (AI) systems, to make them more transparent? Don't they work just fine?

Despite the temptation to answer "yes", are we *really* considering the cases where these systems fail? Are we assuming that "almost perfect" accuracy is good enough? What if, some of the cases where these systems get it right, were just a small margin away from a complete miss? Does that even matter? Considering the ever-growing presence of ML models in crucial areas like forensics, security and healthcare services, it clearly does. Motivating these concerns is the fact that powerful systems often operate as black-boxes, hiding the core reasoning underneath layers of abstraction [Gue16]. In this scenario, there could be some seriously negative outcomes if opaque algorithms gamble on the presence of tumours in X-ray images or the way autonomous vehicles behave in traffic.

It becomes clear, then, that incorporating explainability with AI is imperative. More recently, the politicians have addressed this urgency through the General Data Protection Regulation (GDPR) [Com18]. With this document, the European Union (EU) brings forward several important concepts, amongst which, the "right to an explanation". The definition and scope are still subject to debate [MF17], but these are definite strides to formally regulate the explainable depth of autonomous systems.

Based on the preface above, this work describes a periocular recognition framework that not only performs biometric recognition but also provides clear representations of the features/regions that support a prediction. Being particularly designed to explain non-match ("impostors") decisions, our solution uses adversarial generative techniques to synthesise a large set of "genuine" image pairs, from where the most similar elements with respect to a query are retrieved. Then, assuming the alignment between the query/retrieved pairs, the element-wise differences between the query and a weighted average of the retrieved elements yields a visual explanation of the regions in the query pair that *would have to be different* to transform it into a "genuine" pair. Our quantitative and qualitative experiments validate the proposed solution, which is generic enough to be applied to other scenarios. Great benefits could arise from AI systems being more transparent when discriminating between disease/non-disease decisions.

The code is publicly available at https://github.com/ojoaobrito/ExplainablePR.git

Keywords

Artificial Intelligence, Convolutional Neural Networks, Deep Learning, Explainability, Generative Adversarial Networks, Image Synthesis, Instance Segmentation, Machine Learning, Periocular Recognition

Contents

List of Figures

List of Tables

Acronyms List

ALE	Accumulated Local Effects
AdaIN	Adaptive Instance Normalisation
API	Application Programming Interface
AUC	Area Under the Curve
AI	Artificial Intelligence
ANN	Artificial Neural Network
BN	Batch Normalisation
COCO	Common Objects in COntext
CNN	Convolutional Neural Network
DL	Deep Learning
EER	Equal Error Rate
EU	European Union
FPR	False Positive Rate
FFHQ	Flickr Faces High Quality
FC	Fully Connected
FCN	Fully Convolutional Network
GDPR	General Data Protection Regulation
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
HOG	Histogram of Oriented Gradients
ILSVRC	Imagenet Large Scale Visual Recognition Challenge
IoU	Intersection over Union
LBP	Local Binary Pattern
LIME	Local Interpretable Model-agnostic Explanations
LSTM	Long Short-Term Memory
ML	Machine Learning

MSE Mean Squared Error

MLP MultiLayer Perceptron

PDP Partial Dependence Plot

ROC Receiver Operating Characteristic

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

RGB Red Green Blue

RoI Region(s) of Interest

RoIAlign Region(s) of Interest Align

RoIPool Region(s) of Interest Pooling

RPN Region Proposal Network

R-CNN Region based Convolutional Neural Network

SIFT Scale-Invariant Feature Transform

SHAP SHapley Additive exPlanations

SVM Support Vector Machine

TPR True Positive Rate

UBIPr University of Beira Interior Periocular

VOC Visual Object Classes

Chapter 1

Introduction

In recent years, few areas of research have enjoyed similar manifestations of interest and enthusiasm as DL. Staggering amounts of progress have been made, culminating in models that can do seemingly everything: image classification, object detection, image synthesis, amongst many others. As the remainder of this document will make clear, DL models (and other ML techniques) can be used in creative pipelines, incorporating each architecture's strengths. Such wide availability of solutions proves how accessible the field has become.

Fortunately for the research community, within the field of DL, there is still some ground to cover in emerging areas. Explainability is one of those areas, with active lines of research. Most of the techniques found in literature find their way into existing systems due to a need for explanations. Not by design, but by requirement. Usually, this need comes after the system's deployment, as a quick fix. We argue that this mindset is no longer realistic, especially in areas where decisions need to be accurate and frequently audited. Even if the system's environment does not require intensive validation, ensuring some degree of explainability should always be desirable (it could, perhaps, be considered a "good practice").

Based on what was stated above, in this work we are interested in incorporating explainable components into a system, that given a pair of images from the periocular region, is capable of delivering a twofold output: a binary decision ("yes" or "no"), supported by a pleasing (and visual) explanation. The former is similar to a classical approach, while the latter serves the emerging need for decision validation.

1.1 Motivations and Objectives

The present document aims at presenting a view of the state of ML Explainability, as well as, describing ways to merge such techniques with standard DL methods. By compiling several approaches with adequate descriptions, we hope to clarify the concepts behind explainable AI and how it actually works. Such knowledge is becoming increasingly more valuable as ML systems find their way into real world scenarios.

Regarding the specific goals of this work, we aim at developing an integrated framework that performs periocular recognition and automatically produces easy to understand explanations. By doing so, we are not only accomplishing a useful task but also bridging the gap between our system's reasoning and the users that may end up using it.

1.2 Document Organisation

In order to provide an intuitive reading experience, this document is divided according to the following chapters:

1. **Introduction** - provides a general overview of this work's motivations and objectives, as well as, the structure by which this document is organised.
2. **Related Work** - presents an extensive look at widely used algorithms and techniques in the fields of interest.
3. **Proposed Methods** - showcases the methods and intuitions developed to tackle the problem at hand.
4. **Results and Discussion** - contains the experiments performed to validate the proposed methods, with accompanying results and discussions.
5. **Conclusions and Further Work** - concludes the present document with both a review of the work and its potential improvements in the future.

1.3 Dissertation Outline

In order to perceive the workflow that this dissertation encompassed, Fig. 1.1 depicts a simple Gantt diagram with the tasks performed and how long they took to complete. Naturally, the work started by devoting several months towards research on the topics of interest: Biometric Recognition, DL and ML Explainability. Then, this foundation was put to use during the development of both proposed methods (while the first one consumed a lengthier, uninterrupted time span, the second method required additional research for adequate DL models). Finally, the results conveyed by our experiments were analysed so as to better understand if the original goal had been achieved:

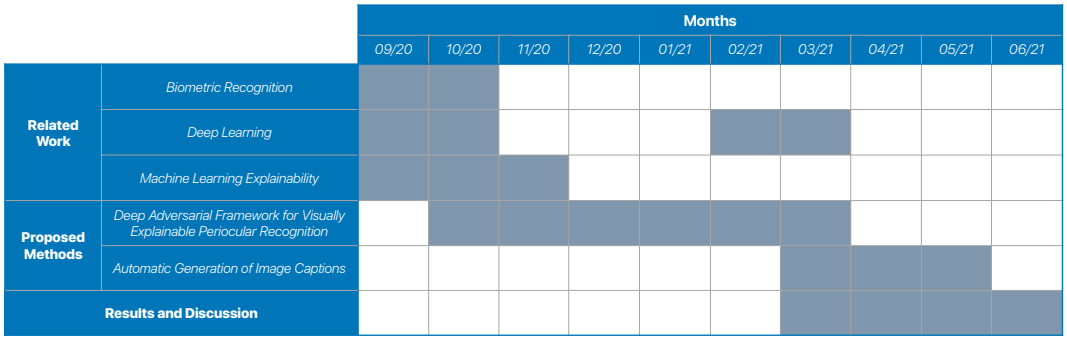

Figure 1.1: Gantt diagram with the dissertation outline. With the exception of additional DL research, the work went through a standard pipeline: research, development and testing.

Chapter 2

Related Work

The goal of the present chapter is to provide a review of the most relevant ideas and techniques in the areas of interest. To that end, the following sections are organised as follows: section 2.1 starts by acknowledging and describing our domain (i.e., biometric recognition); section 2.2 describes the concepts behind DL, as well as, its most popular architectures (like CNNs and GANs); section 2.3 follows up by compiling a list of techniques that are commonly cited in the field of Explainability; section 2.4 summarises the key aspects of this chapter.

2.1 Biometric Recognition

Typically, a recognition problem involves a set of unique and non-transferable features that can unmistakably identify a subject. Biometrics, as they are usually designated, serve such purposes, as long as they are universal, distinguishable, resilient to changes and relatively easy to collect [RP04].
Upon proving their compliance with these requirements, biometrics can be used to uniquely identify instances of data. Naturally, not every biometric suits every use case, and it is an engineering problem to determine which traits will likely convey the highest success rates. Amongst the varied set of possible biometric signals, the following remain the most popular:

- **Face**: face recognition has the natural advantage of being largely non-intrusive (i.e., it can work in both controlled and uncontrolled environments), by usually focusing on the location and shape of facial attributes. Unfortunately, certain lighting conditions or unfavourable capture angles make it more challenging to identify people by exclusively relying on facial data.
- **Iris**: iris recognition relies on the complex, yet unique patterns of an individual's iris. These can be captured with both visible and near infrared wavelength cameras, to highlight the colour and structure of the iris, respectively. Additionally, it is quite difficult to wittingly alter an iris' textures or use an artificial iris altogether, making this trait a solid candidate for many real world applications.
- **Fingerprint**: fingerprint recognition has also been preferred for a wide range of applications, by asking the user to place one or multiple fingers on top of a dedicated scanner. The ridges and valleys on the fingertip are so unique that each finger from the same person has different information. Although more common and less expensive, fingerprint based solutions are still sensitive to external conditions like debris, cuts or bruises on one's fingertip (not to mention gloves).

- **Gait**: gait recognition differs from the biometrics above because it requires both spatial and temporal information, so as to analyse the way someone walks. In practice, the movement patterns of a person's joints, when walking, can be seen as unique and invariant enough to constitute a biometric (if not for anatomic changes that our bodies endure over time). Unfortunately, processing gait information is computationally expensive, requires more physical space and does not offer the same accuracy levels that other traits do.
- **Voice**: voice recognition is, just like gait, an action over time (i.e., a subject must record a voice clip for subsequent processing). Our ability to speak is conditioned by anatomic properties (e.g., vocal tract, mouth) and therefore contains useful information. Despite this, the widespread adoption of such systems remains largely held back by voice altering factors (e.g., catching a common cold or having a hoarse voice).

Furthermore, these unique traits can be divided into two major categories: 1) *physiological* biometrics (e.g., the iris, fingerprint and retina) that are naturally possessed by a given subject and 2) *behavioural* biometrics (e.g., the gait and handwritten signature) that yield from the interaction between a subject and the surrounding environment [BZ19].

2.1.1 Periocular Recognition

Concentrating growing interests in the biometrics domain, periocular recognition uses the information in the vicinity of the eye, in which the iris, sclera, eyebrow, eyelid and skin stand out.

The seminal breakthroughs in the aforementioned area can be traced back to a set of methods termed "feature descriptors". Methods such as Histogram of Oriented Gradients (HOG), Local Binary Pattern (LBP) and Scale-Invariant Feature Transform (SIFT) were able to produce simplified data representations by relying on edges, textures and keypoints, respectively. In [RJ09], the results from each feature descriptor were fused to faithfully discriminate between the "genuine"/"impostor" pairs. This work served as the basis for subsequent fusion-based approaches, as in [PP12b]. In [KZ14] a Restricted Boltzmann Machine was used to learn a probabilistic distribution over the input data, further discriminated with metric learning and SVMs.

With the effective application of DL solutions, researchers turned to popular architectures (in particular CNNs), to pursue ever increasing recognition accuracy. Accordingly, in [ZK17] the main concept involves the use of multiple CNNs that are specialised in classifying a particular kind of semantic information (e.g., gender or age). Then, a score fusion process yields the final response. In [PN18], authors enforce a CNN to ignore the ocular region (due to its likelihood to contain specular reflections) and rely on the eye's surrounding area (eyebrow, eyelid and skin). [RP20] created independent representations of the iris and periocular regions, that feed classification modules, whose scores are finally fused

to reach the decision. Using a multi-glance mechanism, where part of the intermediate components are configured to incorporate emphasis on the most important semantical regions (i.e., eyebrow and eye), Zhao and Kumar [ZK18] developed a recognition model that particularly focuses on these regions, enabling the deep CNN to learn additional discriminative features that improve the recognition ability of the whole model.
Recently, [SL19] attempted to bridge the gap between biometric recognition and explainability, by learning feature specific filters that respond in their preferred spatial locations.
Finally, [HL20] propose an integrated solution that leverages the discovery of parts as a form of attention. More specifically, a dictionary of object parts is learned, upon which a feature map can be grouped into part segments. Then, an attention mechanism determines the parts that should be regarded as essential to the classification process. This solution (hereby referred to as HL) is similar to our methods with respect to the integration of explainability into a recognition framework, making it a prime candidate for a direct comparison (as seen in section 4.2.1).

2.2 Deep Learning

As a subordinate area to ML, DL focuses primarily on algorithms that mimic the human brain and how the seemingly simple neurons receive, process, store and release information in a coordinated manner. Fortunately, the field has progressed immensely throughout the years, updating and rethinking its archetype (i.e., Artificial Neural Network (ANN)s) to perform a wide range of tasks. However, despite the groundbreaking success of these approaches, the field went through some really unremarkable periods before becoming so coveted as it is nowadays.

The first efforts in the field of DL can be traced back to 1943, with the work of neuroscientist Walter Pitts and mathematician Warren McCulloch. They proposed the simplest of elements, one that did so little but, when combined with more of its kind, could achieve a whole much greater than the sum of its parts. The artificial neuron was capable of taking inputs, assigning weights to them and, through some light processing, reach an output.
More than a decade after the first steps, the next major innovation came in the form of the Perceptron [Ros57], so called by Frank Rosenblatt, its inventor. This algorithm could be used to solve linearly separable binary problems, in a supervised manner (i.e., the training samples possess ground-truth labels, against which the model's predictions are to be compared). Some relevant concepts were established, like the use of activation functions to give more expressive power. Moreover, this work started with a single layer Perceptron and naturally extended to an MultiLayer Perceptron (MLP) (i.e., multiple layers of these units). A formulation similar to this one is still used in modern architectures, whether in a feed-forward style, where connections between nodes do not form cycles, or otherwise (i.e., RNNs).
The next couple of decades marked the first really low point in the field's research interest, except for the continued work upon the ideas behind Perceptrons and the discovery of the

backpropagation algorithm in 1974 (the default procedure for weight optimisation, based on the derivative of a loss function with respect to a given weight).
After this period, research saw a rejuvenated wave of interest, with breakthroughs like the formalisation of the CNN's precursor, the Neocognitron [Fuk04]. Kunihiko Fukushima described such framework, in 1980, as having two basic types of layers: convolutional and downsampling/pooling layers. Convolutional layers would serve the purpose of applying a sliding window (i.e., a kernel with learnable weights) over the input to extract features. Downsampling layers, usually in between convolutional layers, could reduce the spatial complexity of the input and, as a side benefit, give invariance to a feature's occurrence.
In 1989, Yann LeCun, one of the better known individuals in the DL community, combined a CNN with backpropagation to classify handwritten digits. Subsequent work led to the establishment of one of the most relevant ANN designs, LeNet5 [BH98].
Several years later, the next DL revolution was made possible by the advent of Graphics Processing Unit (GPU)s, reducing training times to much more realistic amounts. Based on that, Alex Krizhevsky, under the supervision of another DL great, Geoffrey Hinton, proposed AlexNet [SH12] and achieved a record top-5 error in Imagenet Large Scale Visual Recognition Challenge (ILSVRC)-2012, proving that deep CNNs were feasible and well suited to the image classification task. Moreover, it showed that Rectified Linear Unit (ReLU) could be amongst the *de facto* activation functions.
Whilst the DL revolution was put in motion, areas like Image Segmentation saw an uprising of novel methods, diverging from the classical clustering or colour/edge based segmentation methods. Many of those methods became obsolete, as the state-of-the-art moved on to fully connected and region-based approaches [GL17].
Another recent line of work has to do with generative algorithms, brought forward in the original GAN paper [CB14], co-authored by Ian Goodfellow and Yoshua Bengio, the third pioneer of DL.

Outside of the theoretical strides, practitioners and newcomers have enjoyed the success of specialised frameworks that grant more mainstream appeal to DL as a whole. Tensorflow [YZ15] and PyTorch [BC19] are two of those tools, with similarities and disparities that make them more suited to different target audiences. Nonetheless, both include Application Programming Interface (API)s that make the coding more user friendly.

For the purposes of this work, algorithms like CNNs and GANs, amongst others, are of particular interest. Therefore, the following subsections describe those architectures both in general and, through the actual models that were chosen, in particular.

2.2.1 Convolutional Neural Networks

As mentioned above, CNNs have the ability to take a fixed-size image as input, apply complex operations and determine the corresponding class (i.e., the group it belongs to). A typical CNN architecture includes a feature extraction stage and a classification stage. The first analyses the image to keep the core aspects, while the second translates that infor-

mation into the predicted class. The image below is a visual representation of what was described in this paragraph:

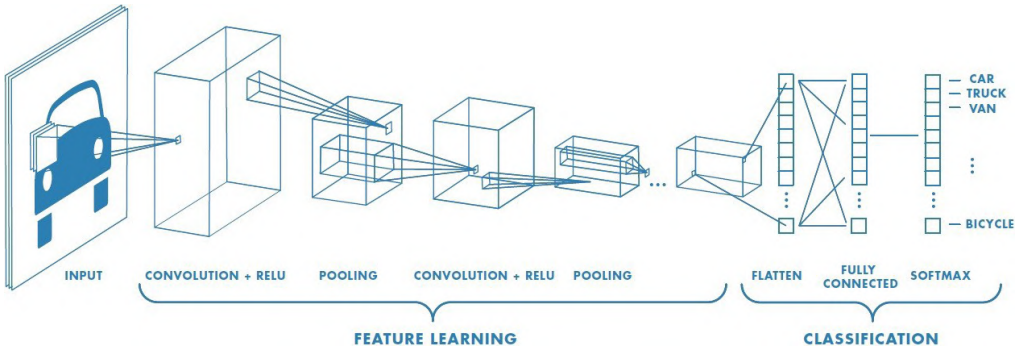

Figure 2.1: Typical CNN architecture [Pra18]. The feature extraction stage extracts as many characteristics as possible from the input image, while the classification stage gives them meaning (i.e., a class).

During the feature extraction stage, the image goes through a series of convolutional and pooling layers. A convolutional layer contains a certain number of filters/kernels of small size (e.g., 3x3 or 5x5 pixels). These filters are responsible for learning both low and high level features from the source image. The first layers usually capture lines, curves and general shapes, while the deeper ones are able to capture more and more abstract concepts. As Fig. 2.1 shows, a filter is a small window that slides over the input, multiplying its values (learned weights) by the ones from the input.

In between convolutional layers, pooling layers are responsible for reducing the dimensionality of the previous layer's output (in Fig. 2.1, every time a pooling operation is performed, the input's height and width are reduced, with the depth being determined by the number of filters). This step also ensures that a feature's occurrence is spatially invariant. Just like convolutional layers, pooling layers use a kernel of size NxN, but unlike convolutional layers, they doesn't possess learnable weights. Two commonly used forms of pooling are max-pooling and average-pooling: the former only keeps the biggest value inside the chosen kernel, while the latter keeps the mean of all the values inside the kernel.

After extracting features from the input image, the final feature maps are flattened (i.e., stretched to a vector like shape) and fed to a regular MLP. An MLP, as described earlier, is a sequence of Fully Connected (FC) layers of neurons, meaning that a neuron receives the output of every neuron, in the previous layer, and its output is sent to every neuron in the next layer.

Finally, the output layer (last layer of neurons) has, typically, a neuron for each class and the values given by those neurons are the probability of the input image belonging to the classes associated with those neurons (i.e., in Fig. 2.1, the first neuron outputs the probability of the image belonging to the "car" class).

Within convolutions and neurons, there are special functions (called "activation functions" in the literature) that usually perform non-linear operations.

In convolutions, an example could be the ReLU function, which simply maps negative

inputs to zero and behaves like the identity function for positive ones.
Neurons, on the other hand, receive several inputs and perform a weighted sum with them (once again, these weights are the parameters learned by the network). Next, the result is given to an activation function (like ReLU, sigmoid or softmax, as shown in Fig. 2.1), which determines the neuron's activation value.
These kinds of functions are crucial, without them a neuron could only process simple data, and not complex (non-linear) data, like images.

Understandably, these methods have been at the core of research efforts over the past years, not only for their ability to learn meaningful information but also due to the democratisation of GPUs. Naturally, many deep designs were proposed, including the Inception [VR15], VGG [SZ15], ResNet [RS16] and DenseNet [MW17] architectures (which subsection 2.2.1.1 covers in more detail).

2.2.1.1 DenseNet

Throughout the years, CNNs got deeper and deeper, in a move that the research community saw as somewhat natural. In spite of this, the gradient based training seemed to halt this trend. That situation was addressed by the ResNet design, which introduced residual connections to create information pathways between more layers, thus controlling the vanishing gradient problem. Additionally, the work with such deep networks showed how some layers could be dropped during training while not seriously affecting the overall performance. Hence, there was evidence that deep residual networks possess a great deal of redundancy [MW17, SW16]. Based on this knowledge, the authors of DenseNet proposed an architecture similar to the following:

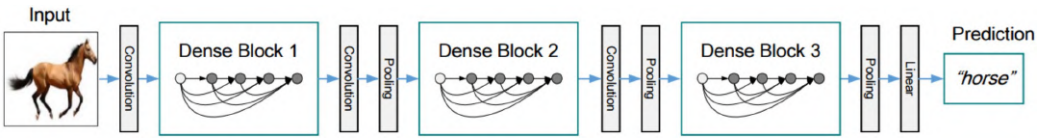

Figure 2.2: Deep DenseNet with three dense blocks [MW17]. Each one of those blocks contains a series of convolutions, pooling and/or BN layers.

Fig. 2.2 contains an example of a deep DenseNet with three blocks, each containing 5 layers. The circles represent the feature maps and the arrows send them through composite functions of operations (including BN, convolutions and pooling), which output more feature maps. Due to the many connections, each composition of operations receives a concatenation of every feature map from the previous layers. In between dense blocks, we have the so called "transition layers", which change the feature maps' sizes via convolution and pooling operations.
With this approach to network design, the authors gave the model a set of densely packed blocks to perform the feature extraction process. Naturally, many versions of the same design were created, like DenseNet-121, DenseNet-161 or DenseNet-201 (where the number is analogous to the number of layers that were included).

2.2.2 Region-based Instance Segmentation

The basic idea behind image segmentation comes from the definition of the word ”segmentation”: division into separate parts/sections. With effect, when an algorithm attempts to segment an image, it tries to determine the object that a given pixel belongs to. Notice the wording here, because, unlike object detection, a segmentation algorithm does not need to know what the objects and visual elements represent (i.e., the classes). But before we discuss instance segmentation, we should create a notion for object detection first.

2.2.2.1 R-CNN

The region-based family of object detection algorithms relies on the proposal of Region(s) of Interest (RoI) in any given image. A naive approach to this task would be to consider all possible bounding boxes, with varying dimensions, aspect ratios and positions. Unfortunately, this algorithm would rapidly become prohibitive in terms of the computational budget. So, to keep the same idea but make it more efficient, the R-CNN paper [DM14] used selective search [GS13] to consider just a few thousand region proposals. The following steps and Fig. 2.3 provide more details on how the proposals are derived:

1. Start with an initial set of seemingly random segmentation regions.
2. Group neighbouring regions together, based on a number of similarity metrics (e.g., colour, texture or size).
3. Repeat the previous step until the stopping criterion is reached (e.g., maximum number of iterations).

Figure 2.3: Visualisation of how the selective search algorithm progresses [GS13]. The top row contains the segmented regions, while the bottom row contains the corresponding bounding boxes. The algorithm works by joining similar regions together until a stopping condition is reached.

Next, the region proposals are resized to a square aspect ratio and fed to a CNN that acts as a feature extractor. Finally, for each region proposal, a trained SVM receives the corresponding feature maps (which are flattened to become a 4096-dimensional vector) and detects the presence of any object of the class it was trained on:

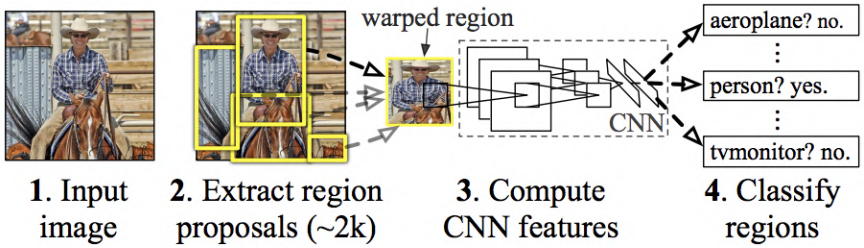

Figure 2.4: R-CNN pipeline [DM14]. The framework uses selective search to create region proposals, which are later fed to a CNN that extracts features. Then, a trained SVM classifies the feature maps as describing a known object or not.

2.2.2.2 Fast R-CNN

After the publication of the R-CNN paper, one of the authors improved the architecture with Fast R-CNN [Gir15], a much faster and accurate approach. This new version still relies on selective search to generate region proposals but includes some key improvements.

Once the region proposals (i.e., RoI) are generated, the image is fed to a CNN responsible for extracting a single set of feature maps. Then, each RoI is projected in the common feature maps, effectively slicing it into smaller feature maps. Next, a new layer called Region(s) of Interest Pooling (RoIPool) computes a fixed-length vector for each RoI: it applies max-pooling to each slice of feature maps, ensuring a pre-defined HxW size, thus creating a small feature map for every RoI that still relies on shared computations. Here we start to see the gains in speed and space by only extracting features once. The pipeline continues by mapping the smaller, RoI specific, feature maps to a feature vector, using FC layers. Finally, the architecture bifurcates into two branches with additional FC layers: one serves the classification purposes and has a "$K + 1$"-way softmax, with probabilities for each of the K classes and the "background" class; and another performs regression by outputting four values for each class, representing the bounding box corners:

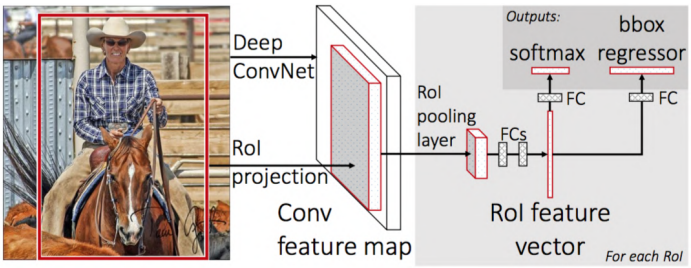

Figure 2.5: Fast R-CNN pipeline [Gir15]. The revision of R-CNN uses a common feature map, shared by all region proposals, instead of extracting features for each and every proposal.

2.2.2.3 Faster R-CNN

Making further improvements, the Faster R-CNN [GS15] architecture extends the Fast R-CNN predecessor by replacing the mechanism that proposes the object regions. Faster inference times and accuracy are, once again, made possible by some rethinking of the core aspects. Two modules serve as the basis for this unified framework: RPN and Fast R-CNN detector. The following figure contains a good visual approximation of the Faster R-CNN architecture:

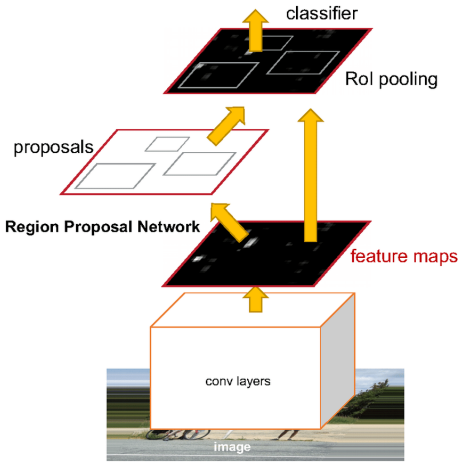

Figure 2.6: Faster R-CNN pipeline [GS15]. The two main modules are the RPN, to predict where an object might be, and the R-CNN detector, to classify the proposed regions.

The advantages brought forward by the RPN include its speed and the ability to be tuned to a specific task (i.e., because it is a trainable component, it can become better suited to the types of images and biases in the training set, unlike generic algorithms like selective search). The RPN works by using k anchor boxes (which are, essentially, matrices of varying aspect ratios) that slide over the image's feature maps, followed by an intermediate layer that produces a fixed-length vector (e.g., 256-dimensional) for each anchor. This vector is then given to two separate FC branches. On one hand, the first branch outputs $2k$ scores (two for each of the k anchors) which are the probabilities of an object's presence in each proposal. On the other hand, the second branch outputs four bounding box coordinates for each of those k anchors:

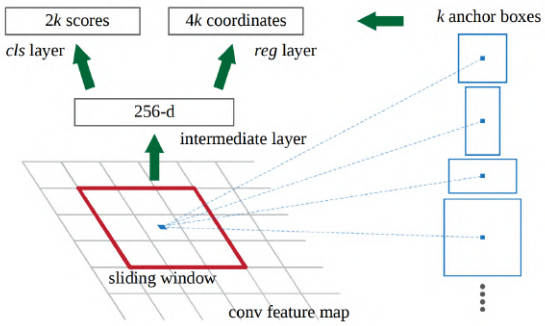

Figure 2.7: Visualisation of the proposed RPN [GS15]. By using multiple anchors with different aspect ratios and sliding them over the received feature maps, the network is able to extract several fixed-length vectors, which are later used for classification and regression.

With the region proposals and the image feature maps (the same used by the RPN module), the Fast R-CNN detector can find the slice of feature maps that correspond to each region proposal and pass it through the RoIPool layer. Then, just like before, a RoI feature vector is computed using a couple of FC layers and given to two sibling branches, which output a class and bounding box.

With the architecture assembled, all that is needed is a way to train it. To do so, the authors propose a 4-step training procedure:

1. The RPN is initialised with ImageNet weights and fine-tuned to perform the region proposal task.

2. With the generated proposals from step 1, the Fast R-CNN detector is also fine-tuned with ImageNet weights.

3. The convolutional layers that extract features from the original image are initialised with the Fast R-CNN weights (remember that these layers are shared across RPN and Fast R-CNN). The remaining unique layers of the RPN are fine-tuned.

4. Finally, keeping the shared convolutional layers fixed (from step 3), the unique layers of Fast R-CNN are fine-tuned, resulting in a trained and unified framework.

Faster R-CNN achieved state-of-the-art performance in popular datasets like Pascal Visual Object Classes (VOC) and Common Objects in COntext (COCO), remaining to this day as a very influential design.

2.2.2.4 Mask R-CNN

Specifically designed to produce segmentation masks, the Mask R-CNN framework [DG17] is an extension of the previously discussed Faster R-CNN. It adds a third branch to the already existing classification and bounding box regression ones. The mask branch is a small Fully Convolutional Network (FCN) applied to each RoI, resulting in the prediction of a binary mask:

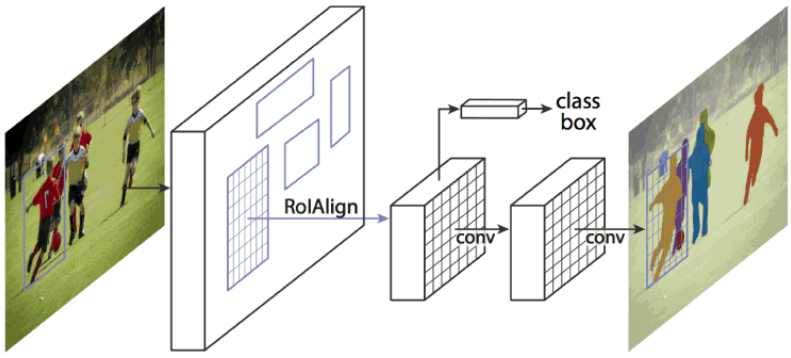

Figure 2.8: Mask R-CNN architecture [DG17]. Built on top of Faster R-CNN, this architecture favours segmentation and includes alignment techniques to ensure optimal mask quality.

By building on top of the Faster R-CNN architecture, Mask R-CNN adds very little extra overhead and keeps the speed that the former enjoyed. The Region(s) of Interest Align (RoIAlign) operation (depicted in Fig. 2.8), which fixes the RoI misalignment present in the previous R-CNN based methods, also helped the object detection performance (i.e., if the mask prediction branch is disabled, the resulting architecture behaves better than the baseline Faster R-CNN). The reason for the misalignment in the original frameworks has to do with the approximations that are made with RoIPool, resulting in the loss of some spatial information. RoIAlign, on the other hand, does not use the exact values in the feature maps, but rather, interpolations that take into account the surrounding values. By doing so, we are using some of the information that would have been discarded by a quantisation based approach like RoIPool. Only after doing this, can the max-pooling operation be used to obtain a fixed-size feature map (the maximum value still has some information about its surroundings).

Mask R-CNN achieved, at the time, state-of-the-art results on the COCO dataset and was even used to perform pose estimation, despite not being intentionally trained to do so.

2.2.3 Generative Adversarial Networks

As generative algorithms, GANs possess the ability to create new data, unlocking many interesting possibilities. Although seemingly complex, the motivation behind these frameworks is really intuitive: two networks with competing goals are put against each other to make for a better learning process (i.e., if one gets better, the other has to keep up).

Formally, a generator G tries to generate new data from random noise. Then, synthetic and real samples are given to a discriminator D, whose task is to distinguish their sources:

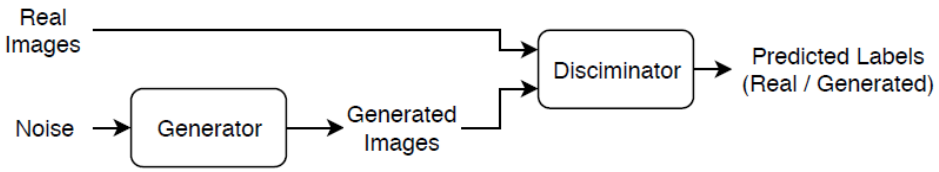

Figure 2.9: Typical GAN architecture [Mat20]. The generator takes random noise as input and outputs fake, but realistic images. By the end of training, the fake images should be hard to distinguish from the real ones.

Analysing Fig. 2.9, one can understand the training process that much better: G will learn to sample a random input vector and produce a synthetic image, so that it can fool D into thinking it came from the real distribution. Conversely, D will learn to distinguish fake samples from real ones. At first, G's images will look nothing like the real ones, but the feedback loop will drive the two adversaries to a converged state: G will be updated by how far it was from fooling D and D's update will be based on how it fared at discriminating between real and fake samples. After training, the discriminator is discarded and we are left with a generator that can produce realistic images.

Equations 2.1 and 2.2 present the formal loss functions that enable the training of both D and G. It should be noted that, in the original description, D will try to maximise the average of the log probability for real images, as well as, the log of the inverse probability for fake images. G, on the other hand, should seek to minimise the log of the inverse probability predicted by the discriminator for fake images:

$$\mathcal{L}_D = \frac{1}{m}\sum_{i=1}^{m}[\log(D(x_i)) + \log(1 - D(G(z_i)))] \quad \textbf{(2.1)}$$

$$\mathcal{L}_G = \frac{1}{m}\sum_{i=1}^{m}\log(1 - D(G(z_i))) \quad \textbf{(2.2)}$$

Since the original publication, the loss terms have been update, whether by making G maximise the log of the discriminator probabilities for fake images or rethinking core aspects of the training procedure (as in Wassertein GANs [CB17], where the discriminator is replaced with a critic that scores how realistic the given samples look). Due to the high interest shown by the research community, many different designs were proposed, including AttGAN [SC19], for image manipulation, and StyleGAN2 [LA20], for high quality image generation. Given its use in this work, the StyleGAN family is highlighted below.

2.2.3.1 StyleGAN

The StyleGAN team started experimenting with a concept called "progressive growing". In the paper where they introduce this idea [LL18], the authors postulate that the discriminator D and generator G could start working with low resolution images (e.g., 4x4 pixels) and then get progressively upgraded to deal with higher and higher resolutions. Both D and G would get new layers, at each resolution change, to deal with the higher dimensionality. To avoid the risk of sudden shocks or instability, the new layers are faded in smoothly and every layer remains trainable throughout the entire training process. This methodology allows both the D and G to first learn the large-scale aspects of the images and then, progressively, shift their attention to finer details, which usually come with higher resolutions. The resulting network, called ProgGAN, was able to generate high quality images at higher than usual resolutions (e.g., 1024x1024 pixels).

Following the work above, the same authors proposed an architecture called StyleGAN. The main differences between this architecture and the baseline ProgGAN are the use of an additional mapping network to map latent codes to a new latent space, the use of style vectors and the introduction of Adaptive Instance Normalisation (AdaIN).
The mapping network opposes itself to the traditional GAN formulation, where we map a point in the latent space to the image space. In this new configuration, we first map the latent points to an intermediate space, through an 8-layer MLP. This intermediate space W is, according to the authors, more disentangled than the former space Z, enabling better manipulation of attributes. Once we have the new latent codes, we apply an affine

transformation to create style vectors, which condition the image generation process. As for the AdaIN technique, it helps to incorporate the aforementioned styles with the feature maps that come from the successive convolution layers. There is also the use of noise vectors, which provide more variation in finer details (i.e., less smoothing):

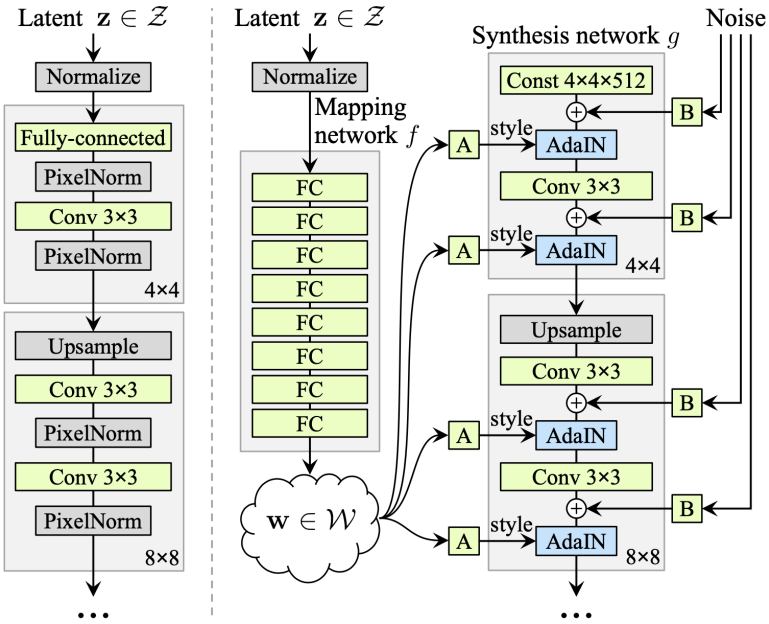

Figure 2.10: StyleGAN architecture [LA19]. On the left, we have the ProgGAN architecture, while on the right we have the proposed StyleGAN version with the mapping network and style injection.

After the release of the original StyleGAN, the same team corrected some of the visual artefacts that it tended to include in the generated images (i.e., "water droplets" and misaligned eyes or teeth). In fact, the "water droplet" effect results from the generator attempting to sneak some information past the normalisation step:

Figure 2.11: Water droplet effect generated by the original StyleGAN design [LA19].

The StyleGAN2 framework [LA20], as it ended up being called, redesigned the way style vectors are incorporated into the generation process, due to a belief that the AdaIN operation limited the generator's ability to use the maximum amount of information. Furthermore, the revised architecture replaced progressive growing with input/output skip connections (for the generator) and residual connections (for the discriminator):

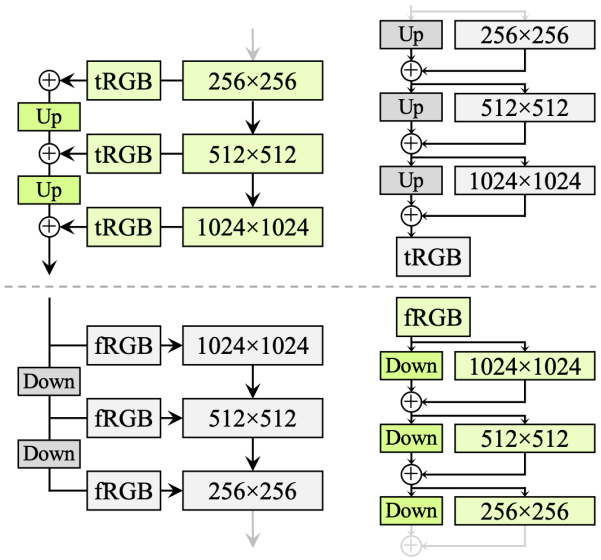

Figure 2.12: Proposed revisions of the networks' architectures [LA20]. On the left, we have input/output skip connections and, on the right, residual style connections. After some experiments, the authors settled on a skip generator (top-left) combined with a residual discriminator (bottom-right).

The improved architecture allowed both networks to have access to every resolution layer from the beginning, but only make use of them if it proved beneficial during training. This revision, along with several others, improved what was already a state-of-the-art design, elevating the quality of synthetic imagery even further.

2.2.4 Long Short Term Memory

As seen before, many DL architectures are particularly tuned for problems like image classification or synthesis. However, in other tasks, such as text generation, a model must be able to retain some of the information received previously. This notion of context is particularly addressed by a family of models known as RNNs. They comprise loops that feed the output back to the unit, allowing it to combine both new and previous information.

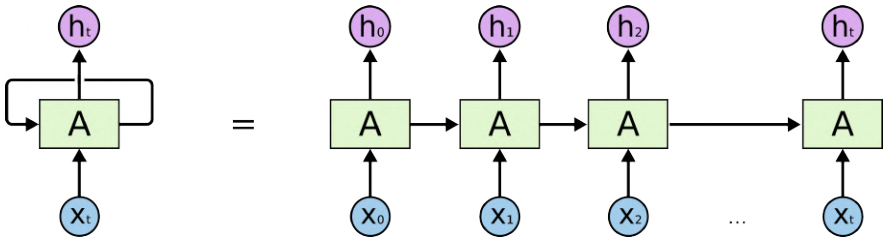

Figure 2.13: Visualisation of a single unit from a RNN [Ola15]. On the left, the unit is seen as theory describes it, while on the right, the loop is expanded indefinitely to form a chain (for explaining purposes).

An architecture similar to the above is perfectly suited for dealing with sequences and lists. Considering their applicability to text generation, these networks can infer the likelihood of outputting a word based on the immediately preceding words. For example, in the sentence "The Eiffel Tower is in ..." the most likely word to appear next is "Paris". The

gap between the relevant context and where the word needs to appear is quite small. But considering an example like "I have been in France for two years and I love speaking ...", we can realise how much more persistence is needed for the notion of "France" to influence the word that we expect to see at the end (i.e., "French"). To deal with longer-lasting dependencies, a more powerful version of RNNs was introduced in the form of LSTMs:

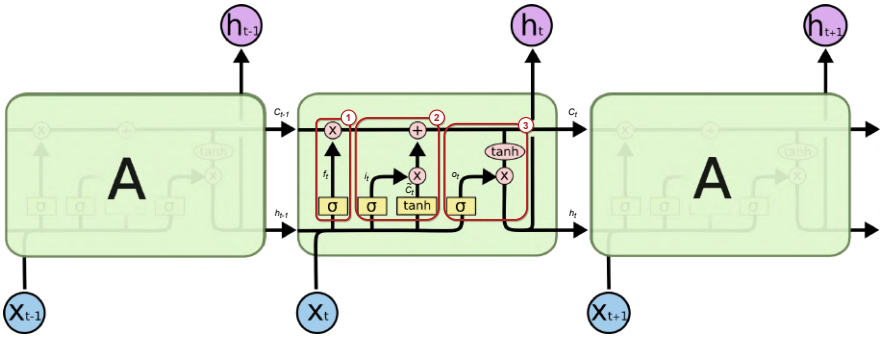

Figure 2.14: Representation of an LSTM's inner workings (edited for explaining purposes) [Ola15]. The top pathway represents the accumulated context (or cell state), which will be subjected to changes if it proves beneficial. In the lower portion of the schematics, two main steps are responsible for filtering and updating the cell state, based on the new information that the unit receives. Finally, the output is a combination of both the updated context and the new, short-term information.

More formally, the top pathway (commonly described as the cell state) represents the previously discussed notion of context (i.e., what we know so far). It is updated in steps 1 and 2, by filtering and adding information, respectively.

As an input x_t enters the unit (step 1), it is added to the output of the previous unit (h_{t-1}) and weighted by the learned weights. Then, the sigmoid function takes that summation and maps it to a value ranging from 0 to 1. It is this value f_t that, when multiplied by the current cell state C_{t-1}, determines if the context is to be forgotten or kept. These gates, as they are also called, allow the LSTM to have more control and sensibility.

Next (step 2), having a filtered context, we must determine how much new information should be added to that context. To do so, a weighted summation of h_{t-1} and x_t is once again given to a sigmoid function which creates a filter (just like before). Parallel to this computation, h_{t-1} and x_t are given to a tanh function which maps them to a value from -1 to 1, later multiplied by the filter. The resulting value is then added to the filtered context, thus creating a newly updated context (or cell state).

Finally, the output of the unit is comprised of the updated cell state conditioned on the new information h_{t-1} and x_t: the tanh function maps the updated context to a value from -1 to 1, later multiplied by the filter derived from the application of sigmoid to the weighted summation between h_{t-1} and x_t.

Upon closer inspection, it becomes clear how the LSTM unit performs as expected: steps 1 and 2 update the knowledge gathered so far, based on the new information received, and then step 3 focuses on outputting a value that is not only based on the short-term information (that is what h_{t-1} and x_t represent) but also on the new long-term context.

2.3 Machine Learning Explainability

As an example of what this topic is all about, let us imagine a binary classifier whose task would be to distinguish between wolves and huskies [Mol19]. After training, it ends up misclassifying a number of huskies as wolves. Explainability techniques could, perhaps, show a tendency for the classifier to use snow, in some of the images, as a feature for the class ”wolf”. Without these techniques, it would be decidedly harder to reach the same conclusion. Thus, the adoption of a higher degree of transparency can bridge the gap between humans and ML models, increasing the level of trust between the two parties.

In [Lip18], the author highlights the different degrees of explainability in ML models. We might link the word ”explainable” to the model's architecture, meaning that it can be easily understood, even if that simple nature leads to a deficiency in capacity. We might also consider the model's parameters as an explainable component. There seem to be many possible (and equally valid) forms of explainability. Therefore, authors in this field have debated on the most accurate taxonomy to describe the scope and depth of what it means to be an explainable system.

The proposed criteria [Mol19, Lip18] classifies ML Explainability in terms of depth, scope and model applicability:

- **Intrinsic or Post hoc**: are we reducing the model's complexity to make it more explainable? Or are we allowing the model to be complex and only explaining its output?

- **Local or Global**: are we explaining individual predictions? Or the behaviour of the entire model?

- **Model-specific or Model-agnostic**: are the techniques specific to a limited range of models (i.e., only work with them)? Or are they generic enough to be paired with practically any kind of ML model?

Following the above criteria, we argue that our approaches (chapter 3) are Post hoc, Local and Model-agnostic. In other words, they allow model complexity, explain single instances of data and can be paired with several types of ML and/or DL models (in specific, different types of classifiers, generative models and instance segmentation architectures).

As a natural extension to this introduction, the next subsections attempt to showcase the most common techniques found in the literature. These techniques have ranging levels of complexity and applicability to the problem of periocular recognition. Some of them rely on images, while other make use of plots. Nevertheless, they share a common goal: remove the mist that conceals the reasoning of complex ML models.

2.3.1 Partial Dependence Plot

A PDP is, as the name suggests, a plot relating one or two features with a target variable [Fri01]. Ideally, the features have to be independent in order to properly explain the behaviour of another, dependent variable (i.e., target). The main purpose of such technique is to determine the nature of feature-target relationships (e.g., linear or more complex). Additionally, PDP is a global explainability method, meaning that it can capture the behaviour of the entire dataset to produce a global relationship between features and targets. From this point forward (including subsection 2.3.2), the features we want to explain will be represented by $\mathbf{x}_S$ and the remaining ones by $\mathbf{x}_C$.

As an example, let us consider a simple, linear regression model. In this setting, the Partial Dependence Function, that would allow us to create a plot, has the form:

$$\hat{f}_{\mathbf{x}_S}(\mathbf{x}_S) = \int_{\mathbf{x}_C} f(\mathbf{x}_S, \mathbf{x}_C)\, \mathbb{P}(\mathbf{x}_C) d\mathbf{x}_C \quad (2.3)$$

The formulation above shows that the PDP values for the features in $\mathbf{x}_S$ are the result of marginalising the model f over the distribution of the features present in $\mathbf{x}_C$. By doing so, a new model $\hat{f}$ is obtained, depending exclusively on the features in $\mathbf{x}_S$. To approximate the indefinite integral, one can take a Monte Carlo approach and average the prediction of instances in the training set (with length m), while fixating the values of $\mathbf{x}_S$:

$$\hat{f}_{\mathbf{x}_S}(\mathbf{x}_S) = \frac{1}{m} \sum_{i=1}^{m} f(\mathbf{x}_S, \mathbf{x}_C^{(i)}) \quad (2.4)$$

It becomes evident, then, that the basic principle of PDP is the lack of correlation between the features in $\mathbf{x}_S$ and the ones in $\mathbf{x}_C$, which is usually not the case.
Suppose we only have weight ($\mathbf{x}_S$) and height ($\mathbf{x}_C$) as features. If a given value of $\mathbf{x}_S$ is fixated, all the other possible values for $\mathbf{x}_C$, observed in the training data, will be paired with that specific value of $\mathbf{x}_S$. In many cases, unlikely combinations will appear, like an instance where a really small weight will be paired with a really high height. It simply comes down to the fact that these two features are heavily dependent on each other. Unfortunately, such limitation limits PDP's applicability in a plethora of cases.

In spite of this, in situations where the chosen features are independent, this technique can even be used for classification problems, in which case the ML model outputs class probabilities. Consequently, a PDP would display the class, as the dependent variable, and one or two features as the independent ones. To get a broader perspective, one could simply create a PDP per class.

To better understand and visualise this concept, Fig. 2.15 shows three different variables (features) being individually plotted against the target variable (in this case, the number of bike rentals on any given day). Note that here we are using the model $\hat{f}$, which only requires one input variable at a time, given that the remaining variables were marginalised:

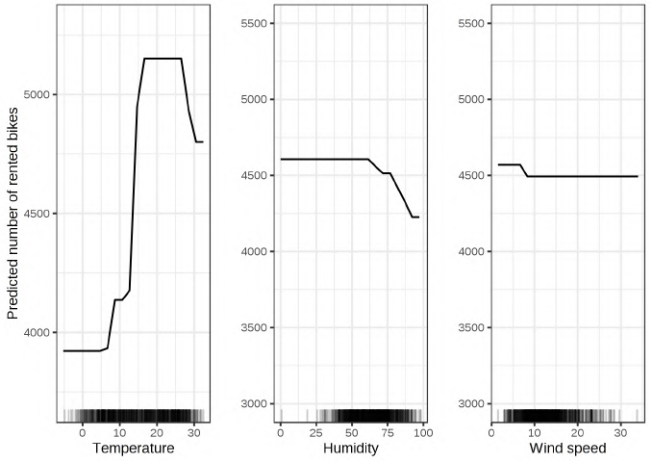

Figure 2.15: Three PDPs from a regression model with three independent variables and one, dependent variable [Mol19]. The predicted values come from a marginalised model ($\hat{f}$) that only relies on one feature.

As an explainable technique, PDP works well if its conditions are met (variable independence and need for only one or two features per plot). However, when features become too entangled, PDP can produce unreliable results [Mol19].

Considering the purposes of this work, this method will not suffice for two main reasons. Firstly, our attributes are, intuitively, entangled: blue-eyed people tend to have lighter skin tones, as opposed to darker skin tones, which often accompany darker iris shades. Feature independence is much more theoretical and usually present in toy problems. Secondly, it is simply not enough to display just one or two features at a time.

2.3.2 Accumulated Local Effects

Another technique that produces plots is ALE, commonly cited as an alternative to PDP, particularly due to the latter's unreliability when the features are correlated [Apl16]. Before getting into the formulas, it is important to visualise ALE's reasoning:

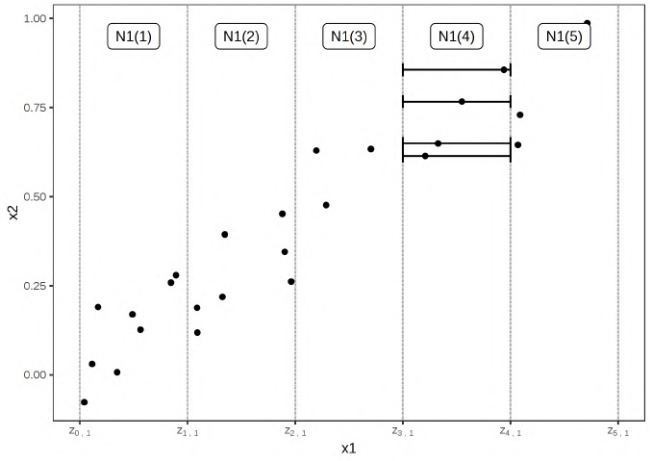

Figure 2.16: Calculation of ALE for feature x_1, strongly correlated with feature x_2 [Mol19]. The distribution in divided into intervals and, for each one, we determine the difference in predictions when feature x_1 takes on the values of the lower and upper limits. These results are later accumulated (i.e., summed) and centred.

From the figure above, we get the idea that ALE evaluates differences in predictions at a smaller scale (i.e., the intervals) and then sums everything to get a broader perspective on how the model f behaves. Indeed, that is what the next formula describes:

$$\hat{f}_{\mathbf{x}_S}(\mathbf{x}_S) = \int_{\mathbf{z}_{0,1}}^{\mathbf{x}_S} \int_{\mathbf{x}_C} \frac{\partial f(\mathbf{z}_S, \mathbf{x}_C)}{\partial \mathbf{z}_S} \mathbb{P}(\mathbf{x}_C|\mathbf{z}_S) d\mathbf{x}_C d\mathbf{z}_C - c \quad (2.5)$$

In equation 2.5, ALE also tries to produce a new model that only requires the features in $\mathbf{x}_S$. The difference with regards to PDP comes from the fact that we do not use all the possible values for the features in $\mathbf{x}_C$, but rather, the values that are possible in the range that our specific feature values belong to ($\mathbb{P}(\mathbf{x}_C|\mathbf{z}_S)$). By using a conditional distribution, we eliminate the unlikely combinations that PDP would have considered. Then, the leftmost integral applies the same reasoning to the entire range of possible values for the features that $\mathbf{x}_S$ contains. Finally, a constant c is subtracted to ensure that, in the ALE plot, the average effect is zero.

As an example, the next couple of formulas are going to describe how the ALE values are calculated when we want to use just one feature (this method is broad enough to work for two features at a time and even for categorical features).
Just like with PDP, in practice we approximate the theoretical definition, and, in this case, that process is done with the following formula:

$$\hat{f}_j(x) = \sum_{k=1}^{k_j(x)} \frac{1}{|\mathbf{n}_j(k)|} \sum_{i:\mathbf{x}_j^{(i)} \in \mathbf{n}_j(k)} [f(\mathbf{z}_{k,j}, \mathbf{x}_{\setminus j}^{(i)}) - f(\mathbf{z}_{k-1,j}, \mathbf{x}_{\setminus j}^{(i)})] \quad (2.6)$$

Starting from the right, in $[f(\mathbf{z}_{k,j}, \mathbf{x}_{\setminus j}^{(i)}) - f(\mathbf{z}_{k-1,j}, \mathbf{x}_{\setminus j}^{(i)})]$, we are interested in determining the difference in predictions when we replace the value of the feature j we want to explain with the values from the upper ($\mathbf{z}_{k,j}$) and lower ($\mathbf{z}_{k-1,j}$) limits of a given interval. The summation $\sum_{i:\mathbf{x}_j^{(i)} \in \mathbf{n}_j(k)}$ allows us to repeat the aforementioned process for every instance whose feature j belongs to a neighbourhood given by $\mathbf{n}_j(k)$. If we recall Fig. 2.16, these neighbourhoods are the vertical slices that divide feature j's distribution. Then, to average the differences we divide the summation by the number of instances in the given interval ($|\mathbf{n}_j(k)|$). This part of the formula computes the local effect of feature j.
Following the above steps, the leftmost summation is responsible for computing and, more importantly, accumulating the local effects of every interval (thus giving meaning to the acronym ALE). Finally, we subtract the average to the resulting values, ensuring that they stay centred around zero:

$$\hat{f}_j(\mathbf{x}_j) = \hat{f}_j(\mathbf{x}_j) - \frac{1}{m} \sum_{i=1}^{m} \hat{f}_j(\mathbf{x}_j^{(i)}) \quad (2.7)$$

Fig. 2.17 depicts an ALE plot of the problem seen in subsection 2.3.1. This example shows that the plots are indeed centred around zero, with positive values indicating an increase in the model's prediction and negative values indicating a decrease, when compared to the

average prediction. Additionally, it becomes clear how shaky ALE plots can be, especially when compared with the PDP version (as a side note, the number of intervals contributes to this effect):

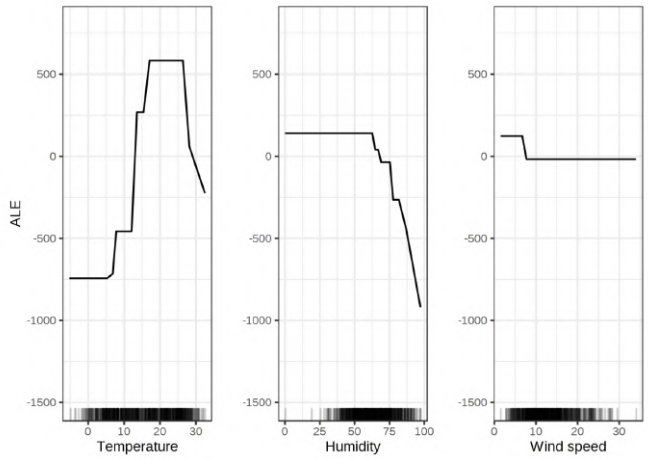

Figure 2.17: Three ALE plots from a regression model with three independent variables and one, dependent variable [Mol19]. As per the trained model, the temperature and humidity's influence outweighs that of the wind speed.

Overall, ALE plots are considered a superior technique to PDPs, despite having some drawbacks too: as mentioned before, the shaky lines can become undesirable and the implementation of this method is more complex. In spite of this, it is generally accepted that, when we are working with independent variables and require fast computing times, PDP is the preferred option. For virtually any other scenario, ALE is regarded as the superior solution.

Just like PDP, and despite being better for the specific use case described in the present document, the visualisation of only a couple features limits ALE's applicability. The following subsections will describe other methods that come closer to what is expected from a framework that can, visually, explain its decisions.

2.3.3 Occlusion Map

Being one of the most straightforward techniques, Occlusion Maps try to make perturbations on certain areas of an input image and register how the model (usually a classifier, like a CNN) reacts to those changes.

In practice, this method involves taking a square, of fixed size and colour, and sliding it over the image, iteratively. Then, the perturbed images are fed to the classifier and its scores are noted. Finally, the scores are translated into a heat map, indicating the portions of the image where an occlusion had the most impact (i.e., the areas most used by the model to predict the correct class).

Fig. 2.18 shows a clear example of what kind of results can be expected with this tech-

nique. Another example can be found in [ZF14] and Fig. 2.19, where the authors went even further by visualising feature maps and noticing how the classifier would attribute different (wrong) classes when certain parts of the image get occluded:

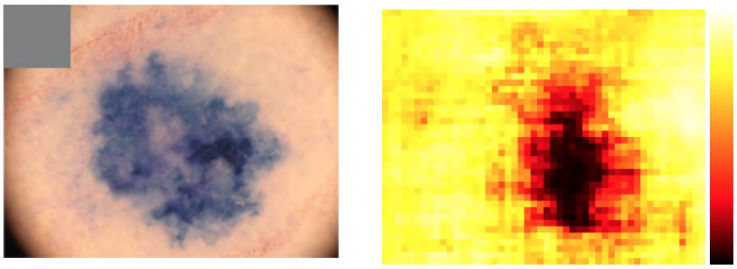

Figure 2.18: Example of a heat map generated with the Occlusion Map technique [She18]. Here, the model clearly identified the melanoma as a crucial element.

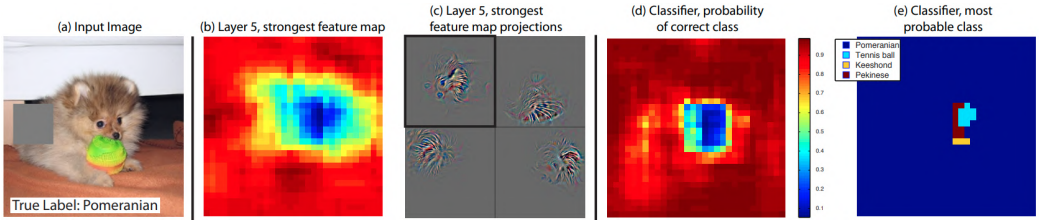

Figure 2.19: Another example of Occlusion Maps being used to interpret a model's decisions [ZF14]. The rightmost figure shows how the predicted class varies with respect to occlusions.

As seen above, Occlusion Maps are able to locate areas of an image that contribute the most to a classifier's decision. As an additional benefit, they are also easy to implement and require almost no redesign or retraining of the classifier.

The applicability of this technique, however, is somewhat limited, given the need for a higher level explanation (i.e., the ability to highlight well defined areas of the periocular region, like the iris or the eyebrow).

2.3.4 Saliency Map

Described in [VZ14], this technique also attempts to highlight certain parts of the input image with brighter or darker shades, based on their importance to the model's score. The authors propose a simple example to start with, similar to the following:

$$S_c(\mathbf{I}) = \mathbf{w}_c^T \mathbf{I} + b_c \quad (2.8)$$

Equation 2.8 comes from a linear score model for class c and, as standard, there is a weight vector $\mathbf{w}_c$, which is later transposed, and a bias b_c. In this setup, each pixel of image $\mathbf{I}$ is weighted according to its importance. Translating such formulation to a CNN becomes more complicated. In spite of this, one can approximate that value by computing the first-order Taylor expansion:

$$S_c(\mathbf{I}) \approx \mathbf{w}^T \mathbf{I} + b \quad (2.9)$$

In this case, w is the derivative of S_c with respect to a specific image $\mathbf{I}_0$:

$$\mathbf{w} = \left.\frac{\partial S_c}{\partial \mathbf{I}}\right|_{\mathbf{I}_0} \quad (2.10)$$

One can interpret equation 2.10 as obtaining an image-specific class Saliency Map where the magnitude of the derivative indicates the pixels that need to be changed the least to change the final score the most.

Having this foundation, the authors continued the Saliency Map extraction process. Remembering that any image (i.e., $\mathbf{I}_0$) has m rows and n columns, a class Saliency Map belongs to R^{m*n}. The first step is to find $\mathbf{w}$, as per equation 2.10 through back-propagation. Then, if $\mathbf{I}_0$ is a greyscale image, $\mathbf{w}$ has exactly one element for each pixel, meaning the Saliency Map could be calculated as $\mathbf{M}_{ij} = |\mathbf{w}_{h(i,j)}|$, where $h(i,j)$ is the index of the element in $\mathbf{w}$ directly corresponding to $\mathbf{I}_0$'s pixel in row i and column j. As for Red Green Blue (RGB) images, with multiple depth channels, an index takes the form $h(i,j,c)$ and only the maximum magnitude across all channels is kept: $\mathbf{M}_{ij} = \max_c |\mathbf{w}_{h(i,j,c)}|$.

Fig. 2.20 has some examples extracted directly from the original source. It should be note that the three Saliency Maps shown are from the highest scoring class on pseudo-randomly selected ILSVRC-2013 test images:

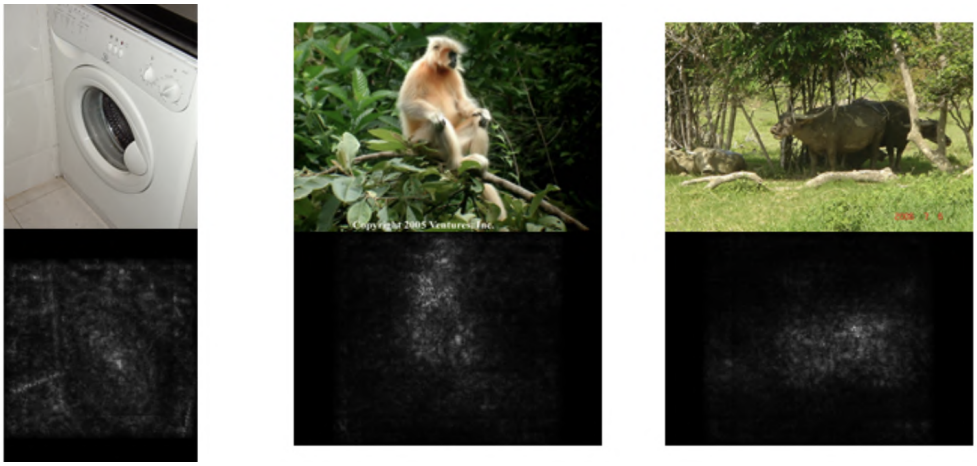

Figure 2.20: Three Saliency Maps (below) extracted from each of the input images (above) [VZ14]. The pixels in whiter tones are the most significant to the classes that were predicted, meaning that changes to them could impact the output class.

One advantage of Saliency Maps is that they are not expensive to compute, only requiring a single back-propagation pass, and do not assume the existence of any additional annotations (apart from the labels used when training the original model).

The ability to locate the most relevant components of an image fits perfectly into the desirable output of a periocular recognition system (not to mention the higher finesse, when compared with Occlusion Maps). Therefore, subsection 4.2.1 presents some results that make use of Saliency Maps to achieve satisfactory levels of explainability.

2.3.5 Local Interpretable Model-Agnostic Explanations

One of the most widely adopted techniques is LIME, originally proposed in [SG16]. This method relies on surrogate, auxiliary models to, locally, explain the behaviour of a much more complex, black-box model.
In order to create a basic understanding of how LIME works, the authors propose we forget about the training data and assume we only have access to the trained black-box model itself, to which samples can be fed and predictions can be withdrawn. LIME's approach starts by taking a sample and creating variations from it. With this process, a new dataset is created and these samples can be fed to the original, black-box model. Having the dataset and the corresponding predictions, LIME moves on to the next phase, where a much simpler model (often linear) is trained with the aforementioned data. The linear model is, however, weighted, meaning that it gives more importance to samples that are closer to the one being explained (thus forcing locality). The learned model should be a good approximation of the original one, albeit locally (the so called local fidelity [Mol19]).

In broader terms, an explanation can be defined as follows:

$$\text{explanation}(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g) \quad \textbf{(2.11)}$$

Equation 2.11 shows how an explanation for an instance x is the summation of a loss component L (e.g., Mean Squared Error (MSE)) and a complexity component Ω. L measures how the predictions from the surrogate model g and the original model f compare, considering the neighbourhood given by π_x. Ω controls the degree of complexity of the surrogate model (preferably low). It should be noted that, in practice, LIME optimises the former term, with the latter being left to the user's responsibility.

One interesting characteristic of this method is that it works for tabular, text or image data. For each of them, LIME tries to create the previously mentioned variations by changing certain aspects of the raw data. With tabular data, it varies each feature individually, helped by a normal distribution with mean and standard deviation in accordance to that feature. As for text and images, LIME turns words or super-pixels (contiguous patches of neighbouring pixels) on or off.
Fig. 2.21 presents an example on how LIME obtains visual explanations:

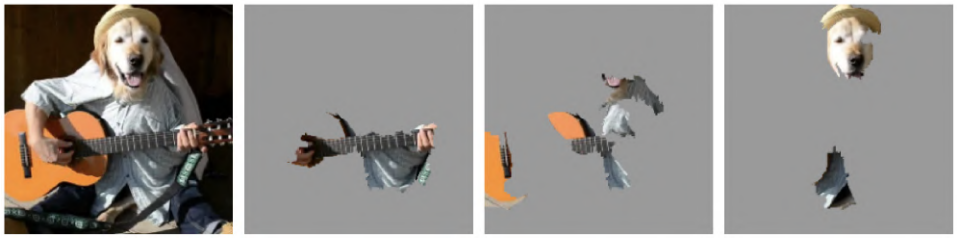

Figure 2.21: Inception's top-3 predictions ("electric guitar", "acoustic guitar" and "labrador", respectively) explained using LIME [SG16]. The highest contributing super-pixels were kept, while the remaining ones were greyed out.

LIME's disadvantages include the inherent limitations of linear models, which can be incapable of modelling a complex decision boundary, even if it is local and at a smaller scale. Despite the drawbacks, this technique is certainly capable of providing a hint of explainability to otherwise opaque models.

Considering LIME is popular and effective, subsection 4.2.1 includes a perspective on how it could be applied to the present work.

2.3.6 Anchors

Following their work with LIME, the authors proposed another approach in [SG18], whose goal is to create decision rules in the form IF-THEN, such that a prediction is sufficiently anchored by some features (i.e., if changes in other feature values do not change the prediction). The techniques employed to that end include reinforcement learning and graph search.

Just like LIME, this method generates perturbations on real instances to create local explanations. Simple IF-THEN rules are used to explain local behaviours, instead of surrogate models. Additionally, the notion of coverage is introduced to specify the amount of instances a rule applies to (i.e., they are expected to be reusable). Fig. 2.22 provides a visualisation of both anchors and LIME:

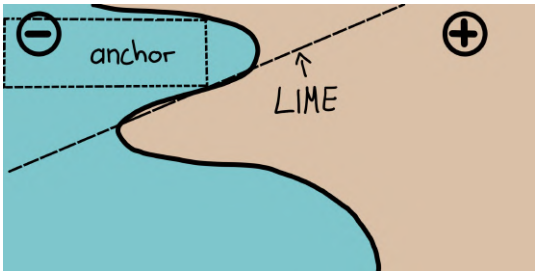

Figure 2.22: A comparison between LIME and anchors [SG18]. One can see that, while LIME tries its best to approximate the decision boundary's behaviour, an anchor provides a more realistic result.

The form of an anchor is usually:

IF (feature$_1$ == value$_1$ AND/OR feature$_2$ == value$_2$ AND/OR ...) THEN
PREDICT target = value$_3$
WITH PRECISION ...% AND COVERAGE ...%

The example above shows the readability provided by an anchor, which specifies the features that contributed the most to a prediction, the original model's prediction, the level of precision (i.e., accuracy) shown by the anchor and how applicable it is to the perturbation space's instances.
In more formal terms, an anchor A must satisfy a given level of precision (τ) such that

$\text{precision}(A) \geqslant \tau$. To determine an anchor's precision, we can use the following equation:

$$\text{precision}(A) = \mathbb{E}_{D(\mathbf{z}|A)}[\mathbb{1}_{f(x)=f(\mathbf{z})}] \quad (2.12)$$

Analysing equation 2.12, $\mathbf{z}$ stands for the perturbed neighbours of x to which A is applicable, D is the distribution of perturbed instances and $\mathbb{1}_{f(x)=f(\mathbf{z})}$ denotes the black-box model's predictions with respect to x and $\mathbf{z}$ (expectably, the same). In practice, it is intractable to determine adequate anchors using equation 2.12. To solve such issue, the authors propose the introduction of a new parameter (referred to as δ) such that $0 \leqslant \delta \leqslant 1$, effectively creating a probabilistic definition:

$$P(\text{precision}(A) \geqslant \tau) \geqslant 1 - \delta \quad (2.13)$$

Furthermore, the notion of coverage, intuitively explained as the need for rules that are applicable to a large portion of D, can also be described with an equation:

$$\text{coverage}(A) = \mathbb{E}_{D_{(\mathbf{z})}}[A(\mathbf{z})] \quad (2.14)$$

Maximising coverage is desirable, given that the generated anchors should be reusable on a decently sized portion of the perturbation space:

$$\max_{\text{A s.t.} P(\text{precision}(A) \geqslant \tau) \geqslant 1-\delta} \text{coverage}(A) \quad (2.15)$$

From all the equations shown, it becomes clear that this process tries to find anchors with the highest coverage, assuming they satisfy the precision constraint (2.13). One interesting aspect is that, rules with more predicates (i.e., conditions in the IF branch) have a tendency for higher precision. However, such characteristic is not to be pushed to extreme cases. A rule that has too many predicates may be overly tuned to predict the instance given (x) and none other (or a really small amount of similar instances). There is, then, a trade-off between precision and coverage.

In addition to the formulations described above, this technique relies on the following steps:

1. Candidate anchors are generated in rounds. The first round is responsible for creating one candidate per feature of x.
2. The best performing candidates move to the next round, where they are expanded to explain yet another feature of x.
3. This iterative process eventually stops when another routine determines we have an anchor good enough to satisfy equation 2.13.

When an anchor is said to perform better than the remaining ones in any given round, a step was used involving the Multi-Armed-Bandit formulation, where each arm is a candidate rule and a classical problem of exploration and exploitation arises. This type of

setting helps speed up the search for an optimal rule, at any given moment.

In terms of examples, the original paper describes the process of obtaining anchors for text and image classification purposes (among other tasks), showing the versatility of this method. Text data is perturbed by omitting certain words and replacing them with pseudo-random ones (following some rules that guarantee coherent replacement words). As for images, instead of turning super-pixels on or off (as was the case with LIME), the active ones are kept and superimposed over an unrelated image, to determine how the black-box model handles them:

Figure 2.23: Anchor explanations for the class "beagle" [SG18]. In the rightmost figures, we can visualise how this technique superimposes the active super-pixels over unrelated samples to mislead a CNN.

Anchors are intuitive and easy to understand, delivering a performance level close to LIME's. Subsection 4.2.1 includes some experimental results with LIME, instead of anchors, mainly due to the fact that there is an official implementation of LIME for image data, but not for anchors.

2.3.7 SHapley Additive exPlanations

SHAP, first introduced in [LL17], has its foundation laid upon Shapley values, which in turn originated from cooperative game theory. Here, features are seen as players inside a potentially collaborative environment, where they can choose to form coalitions (i.e., cooperative parties) to maximise future gains.
As seen in the original source, to calculate the Shapley value for feature/player i, one can use the following formula:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f_{S \cup \{i\}}(\mathbf{x}_{S \cup \{i\}}) - f_S(\mathbf{x}_S)] \quad (2.16)$$

Breaking down equation 2.16 step by step, we consider F to be the set containing every feature in our problem and S is some subset of F that does not contain i. Additionally, $\mathbf{x}$ is the instance we seek to explain and f is a version of the black-box model that considers n features (for example, $f_{S \cup \{i\}}$ is a model that was trained to used all features in S and i, but f_S was trained exclusively with S). Hence, this method requires retraining f for every possible coalition of the features present in F.

Starting from the right, with $[f_{S\cup\{i\}}(\mathbf{x}_{S\cup\{i\}}) - f_S(\mathbf{x}_S)]$ we are computing the marginal value of adding player i to subset S. In other words, we have a coalition where i was not originally present (S), allowing us to perform a prediction using f_S on $\mathbf{x}_S$. The obtained prediction is our base value (i.e., one where i has no influence whatsoever). Then, if we train a version of the black-box model f that uses i ($f_{S\cup\{i\}}$) and give it an instance $\mathbf{x}_{S\cup\{i\}}$, we will get a prediction where feature i was considered. Therefore, by subtracting this latter value by the former base value, we determine the marginal influence of feature i in the game and feature subset S.

Moving to the left, the fraction $\frac{|S|!(|F|-|S|-1)!}{|F|!}$ is responsible for averaging out the effect of every other feature in S that is not i. Imagining that $|S| = 3$ and $|F| = 5$, we would have $\frac{3!(5-3-1)!}{5!} = \frac{1}{20}$. Hence, every possible subset S with three elements, derived from a set F with five elements, contributes by $\frac{1}{20}$. This factor will then rescale the influence of i in a specific subset S. By doing so, we are marginalising the actual composition of S, thus isolating the effect of feature i.

Finally, the summation repeats the same process for every possible subset S, essentially approximating the real Shapley value of feature i.

With the theoretical foundation provided above, one can understand SHAP that much better. As the authors mention, SHAP borrows from LIME and Shapley values to build a better technique. Similarly to LIME, SHAP uses an auxiliary (linear) model to aid in the search for explanations. One interesting aspect of SHAP, are the several branches that build upon it, like KernelSHAP [Mol19], whose details will be described below.

Being based on a linear model that requires a training stage, the first step to understanding KernelSHAP is to analyse its loss function:

$$L(f, g, \pi_x) = \sum_{\mathbf{z}' \in Z} [f(h_x(\mathbf{z}')) - g(\mathbf{z}')]^2 \pi_x(\mathbf{z}') \quad \textbf{(2.17)}$$

The main aspects of equation 2.17 are f (the original black-box model), g (the surrogate model) and π_x (the SHAP kernel). The model f can, basically, be any ML model (i.e., we are only interested in what comes out of it). The same cannot be said about g, which is much more important for this technique and can be defined as:

$$g(\mathbf{z}') = \phi_0 + \sum_{i=1}^{m} \phi_i \mathbf{z}'_i \quad \textbf{(2.18)}$$

In the equation above, g is, as we know, the explanation (surrogate) model, $\mathbf{z}'$ is a simplified vector of ones and zeros to enable or disable certain features (also known as the coalition vector), M is the maximum coalition size and ϕ_i is the coefficient (i.e., Shapley value) for feature i. For tabular data, the coalition vector would turn on or off certain features and for images it would do the same process but for super-pixels. According to [LL17], formulation 2.18 has certain properties, amongst which, local accuracy (i.e., the explanation model should match the original model in terms of predictions).

The final piece of the puzzle is the SHAP kernel, responsible for attributing more weight to small or large coalitions, as opposed to coalitions that only aggregate half the features (or close to it). We happen to learn more about individual features if we can analyse them in isolation (small coalitions) or if we have almost every feature but one (large coalitions):

$$\pi_x(\mathbf{z}') = \frac{(m-1)}{\binom{m}{|\mathbf{z}'|}|\mathbf{z}'|(m-|\mathbf{z}'|)} \quad (2.19)$$

Finally, the five steps that bring everything together are as follows:

1. Sample k coalitions: $\mathbf{z}'_i \in \{0,1\}^m, 0 \leq i < k$.
2. Obtain predictions for each $\mathbf{z}'_i$ by using $f(h_x(\mathbf{z}'_i))$. Note that h is a function that enables or disables features to form coalitions. For tabular data, it keeps the features we wish to form a coalition with and, for the deactivated features, it samples values from other instances. As for images, it keeps super-pixels (if they are paired with a 1 in the coalition vector) or replaces them with a solid colour (if paired with a 0).
3. Compute the weight for each $\mathbf{z}'_i$ with equation 2.19.
4. Fit the weighted linear model using equation 2.17.
5. Return the coefficients from the linear model, namely, the Shapley values ϕ.

After training, the computed coefficients (ϕ) are the Shapley values that explain each feature's effect over a sample's prediction. In order to illustrate the way KernelSHAP works, the following figure contains a visual explanation of a CNN's prediction:

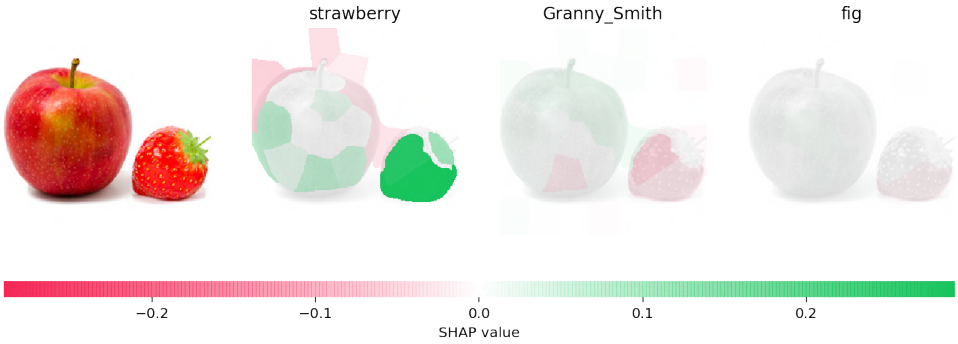

Figure 2.24: KernelSHAP's explanations of a CNN's predictions when given an image from the ImageNet dataset [LL17]. Green super-pixels have higher Shapley values and thus contribute more to the predicted class (unlike super-pixels marked with red tones - this behaviour is inverted in subsection 4.2.1).

SHAP is a solid technique with comparable results to LIME's, exceeding it in some cases. Additionally, it relies on grounded and proven concepts, like game theory, Shapley values and LIME's reasoning. However, KernelSHAP suffers much the same problems as other permutation based methods: by replacing omitted features with random ones, unlikely data points may be generated, which can lead to unrealistic explanations [Mol19].

Nonetheless, a possible application of SHAP to the problem of periocular recognition can be found in subsection 4.2.1.

2.4 Conclusion

The present chapter focused on describing DL and some of its main architectures, which lay the foundation for the proposed methods. Such description was followed by a compilation of the most relevant techniques in the field of ML Explainability. Some of them are perfectly valid and interesting, but their applicability is let down by the lack of visual explanations or overly simple nature, while others are better suited to what this work is trying to accomplish. As an attempt to empirically validate this belief, the following chapter contemplates a section solely dedicated to discussing what results can be expected by only using existing techniques, in the search for an adequate level of explainability.

Chapter 3

Proposed Methods

The present chapter provides an extensive overview of two methods that try to explain whether two images from periocular regions come from the same person or not. To that end, the remainder of this chapter is organised as follows: sections 3.1 and 3.2 contain an overview of the methods that were developed, while section 3.3 summarises the main takeaways from the present chapter.

3.1 Deep Adversarial Framework for Visually Explainable Periocular Recognition

The first framework, that set out to tackle the problem of periocular recognition, is based on widely used DL architectures: CNNs and GANs. Such models were used as the basis of both parts that make up the final answer: the CNN for the traditional binary part and the GAN for the visual counterpart. To go over the details of said method, subsection 3.1.1 covers the pre-processing steps that enabled the subsequent stages, which are later described in subsection 3.1.2.

3.1.1 Data Pre-processing

The data pre-processing stage is quite common in pipelines that involve some kind of DL model. These models often demand fixed size inputs and perform optimally with balanced and rich datasets. Naturally, the present method also required some preliminary steps:

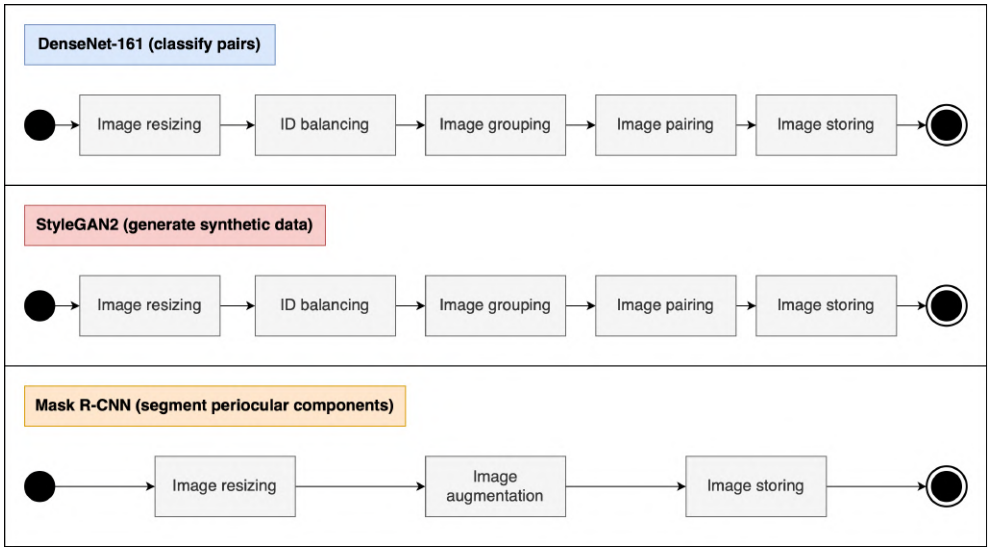

Figure 3.1: Diagram of the data pre-processing pipeline. The images are resized, processed and stored in proper folders.

Fig. 3.1 depicts three major sets (one for each of the main components):

- **Image resizing**: the images are resized to a fixed size (e.g., 256x256 pixels).
- **ID balancing**: considering that, in the unprocessed dataset, some IDs possess more images than others, each ID gets it images either sampled or augmented. On one hand, if the ID has less images than a pre-defined target, the existing ones are augmented to reach said target (with techniques like horizontal flips, rotations, contrast changes or noise additions). On the other hand, if the image count is too great, a simple sampling routine chooses as many images as the target value enforces.
- **Image grouping**: the images are grouped by ID and some of them are reserved just for the test phase (according to a pre-established set).
- **Image pairing**: the images are paired with each other, to form either ”genuine” or ”impostor” pairs.
- **Image storing**: the images are stored in folders targeted towards training, validation and testing (in the case of DenseNet). As obviously required for the CNN and GAN, within each of these main folders there are class sub-folders (”0” for ”impostor” pairs and ”1” for ”genuine” pairs). The Mask R-CNN model just required the images and corresponding masks to be stored in standard folders.

3.1.2 Method Description

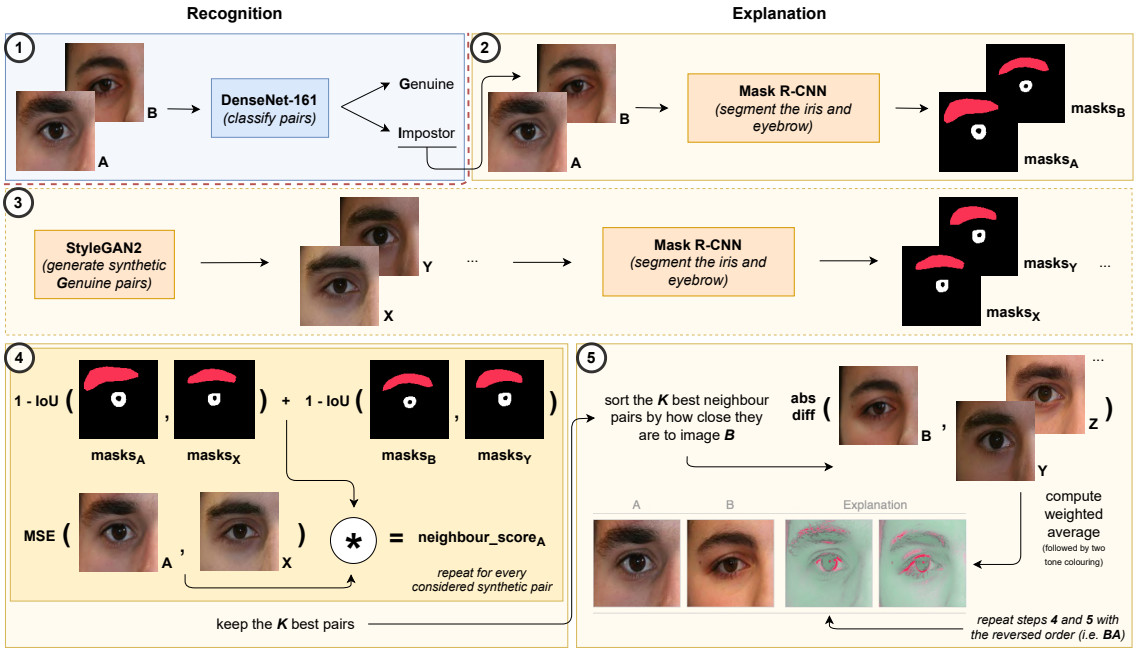

Figure 3.2: Cohesive perspective of the main pipeline of the proposed solution. The recognition step encompasses a CNN that distinguishes between ”genuine” and ”impostor” pairs. Then, upon an ”impostor” decision, steps two to five (explanation) find the k ”genuine” synthetic pairs amongst a large set that most closely resemble the query pair. Assuming the alignment between the query and the retrieved pairs, the element-wise differences between the query and a weighted average of the retrieved elements provides a visual explanation of the features in the query that would have to be different to turn it into a ”genuine” pair.

3.1.2.1 Learning Phase

The main components of the proposed method comprise three well known models: the DenseNet-161, Mask R-CNN and StyleGAN2. The first one (DenseNet-161) is trained to solve an identity verification problem, while the segmentation model (Mask R-CNN) is fine-tuned to produce high-quality masks for the iris and eyebrow. Finally, the GAN model (StyleGAN2) learns how to create synthetic data that, while closely resembling the distributions in the training set, is diverse enough to approximate unseen subjects. Additionally, a fourth, auxiliary model (ResNet-18) is fitted to discriminate between images from the left and right sides of the face. Although trained separately, all the models learn from the same training split, which excludes a set of disjoint IDs that are reserved for performance evaluation purposes.

Regarding the model used in the verification task (DenseNet-161), it should be stated that it has much more parameters than the network used by Zhao and Kumar [ZK17] in their solution. This might be the fact that sustained slightly better recognition performance of our model with respect to the baseline (subsection 4.2.2), but also at the expense of a substantial higher computational cost of classification than the baseline, which might be impracticable in some cases.

3.1.2.2 Inference Phase

Once trained, our method is conceptually divided into five major steps, as depicted in Fig. 3.2. Firstly, the DenseNet-161 model is used to verify the claimed identity: upon receiving a pair of images, the model discriminates between ”genuine”/”impostor” pairs. If the pair is deemed to be ”impostor”, the remaining steps create a visually accurate explanation of that decision.

The second step takes the query pair and, using Mask R-CNN, segments the irises and eyebrows regions. Next, step three uses the StyleGAN2 generator to create a large, synthetic set of exclusively ”genuine” pairs (i.e., where both images belong to the same person). For each of these synthetic pairs, the ResNet-18 model determines its side configuration (i.e., whether images regard the left or right side of the face) and, as before, masks are obtained by the segmentation model. Uncurated synthetic samples are shown in Fig. 3.3.

After obtaining the synthetic data and their corresponding masks, the synthetic dataset is indexed based on the coordinates of the center of the iris, which will enable faster search in the retrieval step. To that end, the clustering algorithm K-Means is trained on a subset of the iris segmentation masks to obtain three centroids, one for each major iris gaze family (i.e., left, centre and right). This way, we index the available pairs based on their combination of iris positions (e.g., left-left, right-centre, ...). By doing so, when searching, we can just rely on the synthetic pairs that share the same combination as the test pair, saving time and useless calculations.

Figure 3.3: Examples of the synthetic image pairs in our dataset, generated according to a GAN model. These elements are drawn exclusively from the ”genuine” distribution. Upon a query, the most similar synthetic pairs with respect to the query are found, which will provide the features/regions that would transform the query into a ”genuine” comparison.

Upon settling for a portion of the synthetic dataset that closely meets the iris position constraint, the segmentation masks are further used to determine which synthetic pairs have the iris and eyebrow approximately overlapped to the query. This is an important requirement to obtain visually pleasant explanations, given that pixel-wise differences are extremely sensitive to differences in phase (i.e., component misalignment). Accordingly, we obtain a similarity score s_X between each synthetic neighbour and the query, given by:

$$s_X = \omega_{\text{masks}} * ||\text{query}_\text{A} - \text{neighbour}_\text{X}||_2, \quad (3.1)$$

being $||.||_2$ the $\ell - 2$ norm and $\omega_.$ a weight that considers component misalignment. This way, we obtain a weighted distance between each synthetic neighbour and the first image of the query pair. ω_{masks} values serve to favour pairs that have good alignment, considering $1 - \text{IoU}(.,.)$, i.e., the complement of the Intersection over Union (IoU) of the synthetic/query segmentation masks.

In practice, we search amongst the (large) thousands of synthetic pairs for the closest to the query pair in terms of the first image. Therefore, given that the second image of the query pair is from a different subject, it will most likely have features that are different to the synthetic neighbours, which are exactly the kind of dissimilarities that make up the final explanations. This way, the K closest neighbours are sorted according to their element-wise distance to image B, using (4.1).

Finally, to produce the visual explanation, the K best neighbours are used to obtain the pixel-wise differences against the query pair image B. In practice, a neighbour distance is subtracted from the total sum of distances, creating an inverted distance. This assures that the contribution of the closest synthetic neighbours to the final result is more important than of those with bigger distances.

3.1.2.3 Implementation Details

The DenseNet-161 model was trained for 15 epochs with a learning rate of 0.0002 and a batch size of 64 image pairs. The Adam algorithm was used for the weight optimisation process (with default β_1 and β_2 values). A similar training setup was used to train the ResNet-18 model, albeit for a smaller number of epochs (i.e., 5).

For the Mask R-CNN's training process, we kept its default values, using a learning rate of 0.001, a batch size of 1 and 30 epochs worth of training (in this case, fine-tuning from the COCO pre-trained weights).

Regarding the StyleGAN2 architecture, the training step comprised a total of 80000 iterations and a batch size of 8. After converging, the generator is capable of synthesising realistic looking images, such as the roughly 400000 pairs that make up the artificial dataset. Finally, for the number K, that determines how many synthetic neighbours should be kept, we used a default value of 15.

3.2 Automatic Generation of Image Captions

For the second method considered in the scope of this dissertation, the feature extraction prowess of a CNN and the text generation of an LSTM were combined to allow for automatic image captioning. The basic premise is that this solution takes as input an image pair and produces text descriptions in which, hopefully, the different periocular components are highlighted. As before, subsection 3.2.1 describes the data preparation steps, while subsection 3.2.2 describes the main architecture.

3.2.1 Data Pre-processing

This admittedly simpler method also required some preliminary steps, to ensure proper sizing and storing of the image pairs:

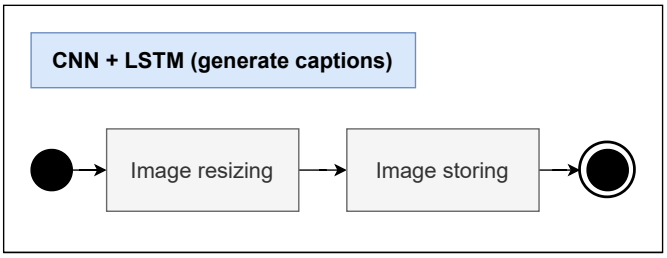

Figure 3.4: Diagram of the second method's pre-processing pipeline. The images are resized and stored in proper folders.

As Fig. 3.4 depicts it, the image pairs are resized to the pre-established size required by the ResNet model (i.e., 224x224 pixels) and stored in a simple folder. Then, once in the training stage, the images and captions are adequately paired.

3.2.2 Method Description

3.2.2.1 Learning Phase

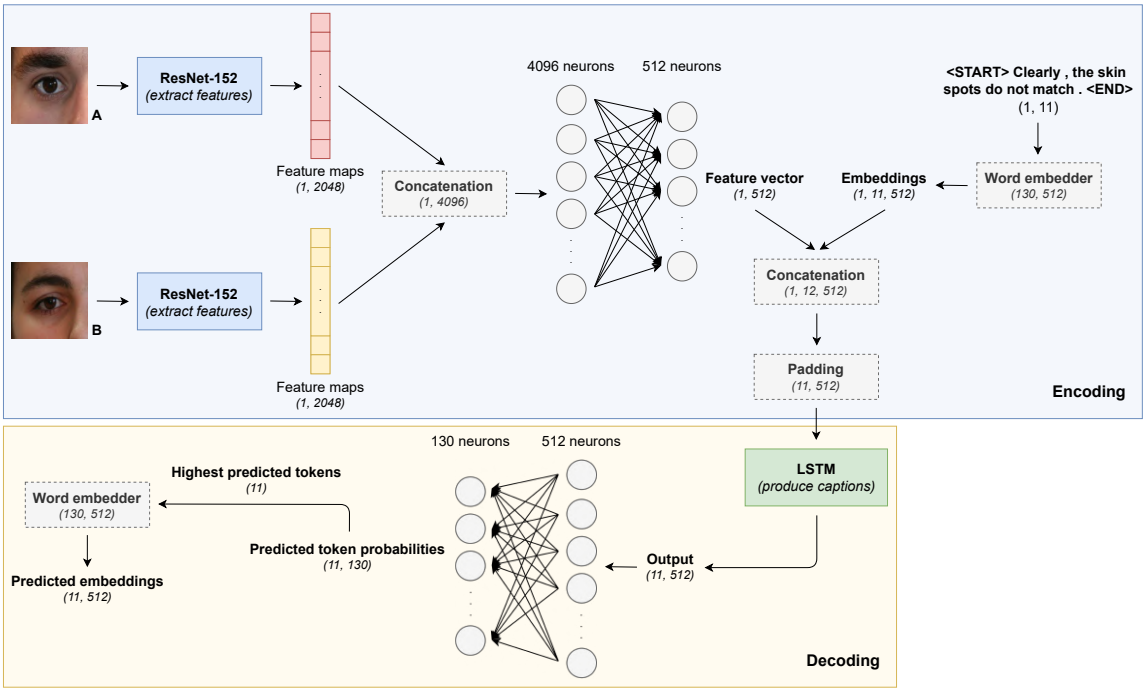

Figure 3.5: Overview of the learning stage of the captioning solution. A given image pair is given to the same CNN, which extracts a feature map for each image. Then, the two feature maps are concatenated lengthwise and given to a couple of linear layers, culminating in a 512-dimensional feature vector. At the same time, the ground-truth caption is embedded and concatenated with the feature vector, leading the resulting tensor through a padding operation (to ensure consistency). Next, the LSTM receives the padded tensor and tries to output the most likely tokens (that make up a caption). Finally, the predicted tokens are encoded so as to retrieve the corresponding embedding, which can naturally be compared to the ground-truth embedding for learning purposes.

In this architecture, two major components can be considered: an encoder (i.e., the CNN) and a decoder (i.e., the LSTM). The former tries to produce a compact representation of the images received, while the latter attempts to generate a plausible caption from that representation. More formally, the encoding is done using the ResNet's feature extraction abilities. Firstly, 2048-dimensional feature maps for both images A and B and concatenated lengthwise to form a single vector with 4096 values (further compressed into a 512-dimensional representation with a couple of linear layers). Then, this vector is concatenated with an embedding of the ground-truth caption and sent through a padding operation that rearranges its input to achieve the desired shape.

After these steps, the resulting tensor is fed to the LSTM, whose task is to predict the tokens that would make up a realistic caption (after a couple of linear layers). To allow for proper comparison against the ground-truth, the predicted sequence is embedded in a similar fashion as before.

3.2.2.2 Inference Phase

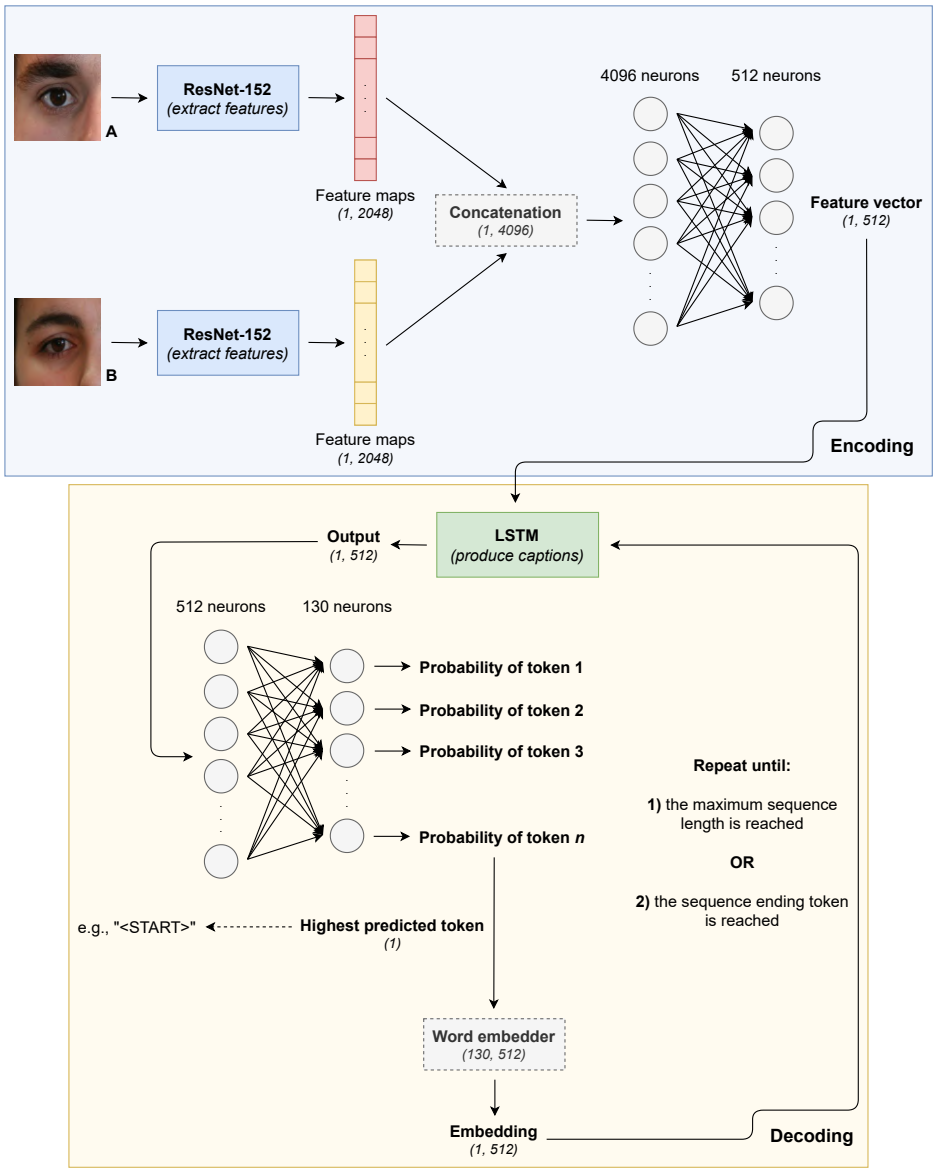

Figure 3.6: In inference mode, the encoding stage is kept: two feature maps are derived, concatenated and compressed further into a 512-dimensional feature vector. Unlike before, the decoding starts with the LSTM receiving the feature vector as is and, in conjunction with two linear layers, outputting the most probable token to start a sentence (ideally, "<START>"). The predicted token is fed to the embedder and the resulting representation is fed back to the LSTM, completing a loop. To exit said loop, one of two conditions must be met: either the maximum sequence length is reached or the "<END>" token is output.

Once finished training, the encoding stage remains virtually the same: the two images go through the same CNN, the resulting feature maps are concatenated and a couple of linear layers predict a feature vector. It is in the decoding stage that differences start to appear. Firstly, the LSTM receives just the feature vector and, with the aid of two linear layers, predicts the first token (ideally, "<START>"). An embedding is derived (as in training) and fed to the LSTM so that it contiues the token generation process until we reach one the following stop criteria: **1**) the predicted sequence reaches a length bigger than the pre-defined maximum or **2**) the "<END>" token is reached.

3.2.3 Implementation Details

The method described before was trained for a total of 20 epochs with a learning rate of 0.001 and a batch size of 64 samples. The total number of training samples was roughly 1000 unique pairs, with each having 2 available captions (thus equating to approximately 2000 "pair-caption" combinations). Once again, the Adam optimiser was used to update the weights (with default β_1 and β_2 values). Unlike before, the CNN's parameters were not update and instead the ImageNet weights were used. Furthermore, the embedding length was set to 512, as was the LSTM's hidden size (i.e., the number of units in the unrolled chain).

3.3 Conclusion

This chapter described the two methods developed to fulfil the proposed goal: being able to explain *why* two images appear to be from different subjects. While the first solution operates in the visual domain (i.e., images), the second method attempts to convey a similar amount of information in written form. Naturally, the next chapter will evaluate these approaches, with a range of samples from each.

Chapter 4

Results and Discussion

The present chapter focuses on the experimental and practical side of this dissertation. In it, one can expect to find results for both state-of-the-art and proposed methods, as well as, the conclusions that each entails. To that end, section 4.1 describes the datasets used; subsections 4.2 and 4.3 assess the performance of both proposed methods; section 4.4 presents some final remarks.

4.1 Datasets and Working Scenario

As mentioned above, the proposed framework is composed of two modules: 1) one for recognition and 2) the other for explanation purposes. Regarding the former, the chosen CNN is solely trained on the UBIPr dataset [PP12a], which provides the ID annotations used in the identity verification problem. Regarding the explanation step, it mainly relies on a combination of UBIPr and FFHQ [LA19]. Despite not being directly applicable to the context of this work (i.e., it contains full face images, thus requiring extra steps to extract the periocular region), the FFHQ dataset contains a large variety in terms of periocular attributes, some of which are scarcer in the UBIPr dataset. In practice, a small, but curated, portion of the FFHQ samples was used to create a data super set (Fig. 4.1). Regardless of their source, all images were resized to a common shape, depending on the task (i.e., 512x512x3 for Mask R-CNN, 256x256x3 for StyleGAN2 and 128x128x3 for the CNNs).

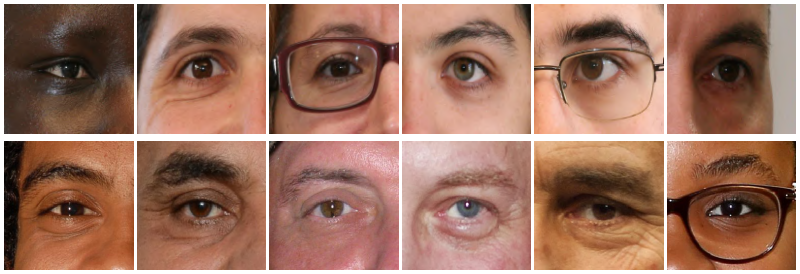

Figure 4.1: Samples from the two datasets used. The top row represents images of the UBIPr dataset, whereas the bottom row illustrates cropped samples of the FFHQ dataset.

As it is usual in the biometric recognition context, it is important to define proper working modes and world settings, for which the system is built. With respect to the working mode, our model runs in verification mode (also referred to as *one-to-one*), where the system validates a claimed identity [RP04]. As for the world setting, we assume an open-world setting, in which unseen subjects can be faithfully handled in the inference step.

4.2 Deep Adversarial Framework for Visually Explainable Recognition

4.2.1 Explainability Evaluation

To justify the pursuit of new explainable techniques, one must start by exploring existing methods. Thus, model-specific (HL and Saliency Maps) and model-agnostic techniques (LIME and SHAP) were employed, so as to give a general overview of what is achievable with currently available approaches. Apart from HL (which used a ResNet-101 model), the techniques were paired with the same DenseNet-121 network. Regardless of the architecture, the task was to output the class of an image pair: "genuine" or "impostor".
The DenseNet-121 model was trained for 15 epochs with a learning rate of 0.0002 and 32 samples per batch. Regarding the ResNet training procedure, it comprised 5 epochs, a batch size of 32 and a learning rate of 0.001. Excluding Saliency Maps, the remaining techniques rely on pre-defined hyper-parameters. As such, LIME was set to show the top 100 super-pixels and was allowed 20000 perturbed samples; KernelSHAP used 100 segments/super-pixels and 10000 samples; HL was trained with four discoverable parts. Finally, note that for LIME, SHAP and HL, the official implementations were used [1][2][3]

Saliency Maps are, essentially, greyscale images where whiter tones highlight crucial areas used by the CNN to predict a class. Accordingly, in Fig. 4.2, the first pair is explained by highlighting subject B's glasses while, in the second pair, subject A's eyebrow definitely justifies a non-match decision (just like a skin spot in subject B's sample, which this technique fails to identify). Moving to the bottom row, pair number three is clearly explained by accentuating one of the irises and the fourth pair is explained through the differences in the eyebrow regions (albeit, not as detrimental as the skins could have been).
Overall, Saliency Maps provide relatively easy explanations to otherwise opaque models, and manage to outline big components (like the eyebrow or iris), while at the same time, leaving behind skin spots and equally small, but relevant, features.

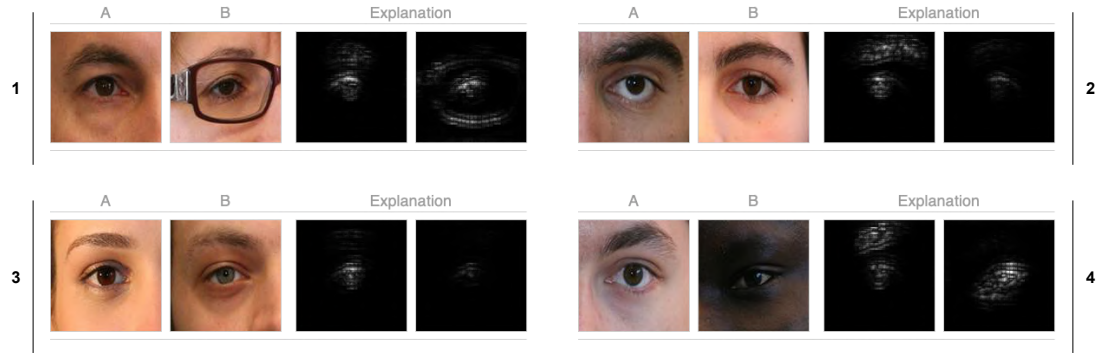

Figure 4.2: Impostor pairs explained using Saliency Maps. Whiter tones highlight areas that justify the predicted class.

[1]https://github.com/marcotcr/lime
[2]https://github.com/slundberg/shap
[3]https://github.com/zxhuang1698/interpretability-by-parts

LIME, as previously mentioned, divides images into super-pixels that are either kept or disabled. With effect, in Fig. 4.3, the first pair's explanation includes a portion of person B's glasses and, in the second pair, a disparity in terms of eyebrows is somewhat manifested. As for pairs three and four, the former is explained by keeping super-pixels that comprise subject A's eyebrow (but missing those that include the irises), while the latter's explanation preserves some skin super-pixels.
Similarly to Saliency Maps, LIME generates passable explanations, meaning that it highlights at least one major feature, but failing to be consistently incisive, in our experiments.

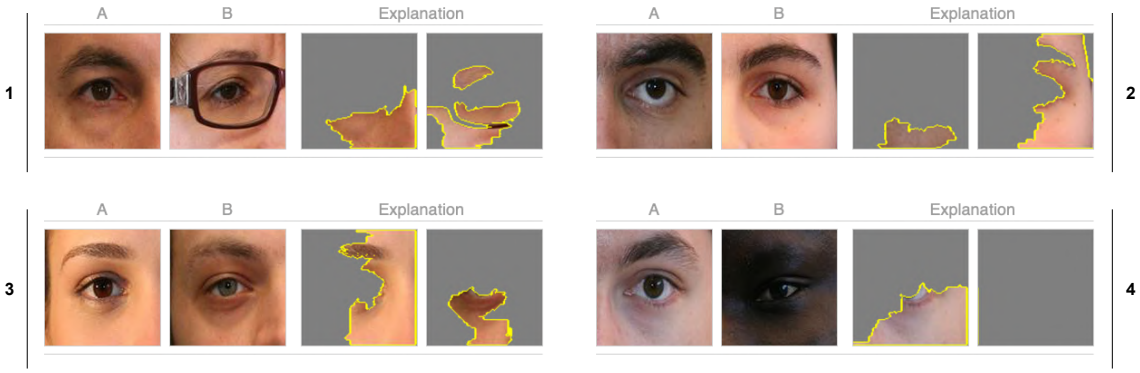

Figure 4.3: Impostor pairs explained with LIME. As mentioned earlier, LIME keeps the most favourable super-pixels and replaces the remaining ones with a solid colour.

KernelSHAP, on the other hand, produces results with various shades of green and red, depending on how favourable (red) or not (green) the highlighted super-pixels are to the predicted "impostor" class. As before, the first pair is explained with the presence or absence of glasses and the second with the eyebrows and a portion of the skin. As for pair number three, both irises are shaded with a slight red tone, just like the eyebrows. Lastly, regarding the fourth pair, a major skin area, belonging to person B, is accurately painted in red (even though some of the skin portrays either green or very slight red tones).
In general, this implementation of SHAP (i.e., KernelSHAP) is able to colour specific features in an image based on how likely they are to change the "impostor" class. Despite missing some features (e.g., some skin areas), the overall results are satisfactory, with some room for improvement.

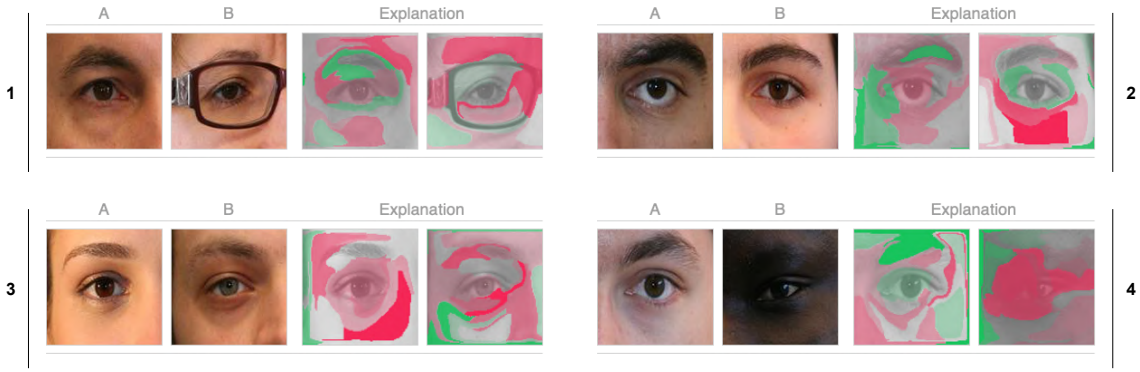

Figure 4.4: Impostor pairs explained with SHAP. SHAP diverges from LIME by highlighting certain areas with red or green tones, depending on whether they increase or decrease the probability of the output class.

Finally, as described in [HL20], HL produces intuitive heat maps as a form of explainability. In Fig. 4.5, pair one is rightly explained by colouring one of the eyebrows and the glasses with red tones. Pair number two remains accurate by strongly identifying both eyebrow as being different, in addition to the eyelashes. As pleasing as the top two results are, the third explanation unfortunately fails to capture obvious differences in iris colour, amongst other possible explanations. Pair number four concludes this method's results with an acceptable explanation, in the sense that the skins are largely ignored in favour of the eyebrows.

HL delivers readable results, mostly aided by a clear indication of which areas are regarded as important. Unfortunately, some features are not displayed in a sufficiently prominent manner (e.g., the skin in the fourth sample), leading to partial explanations.

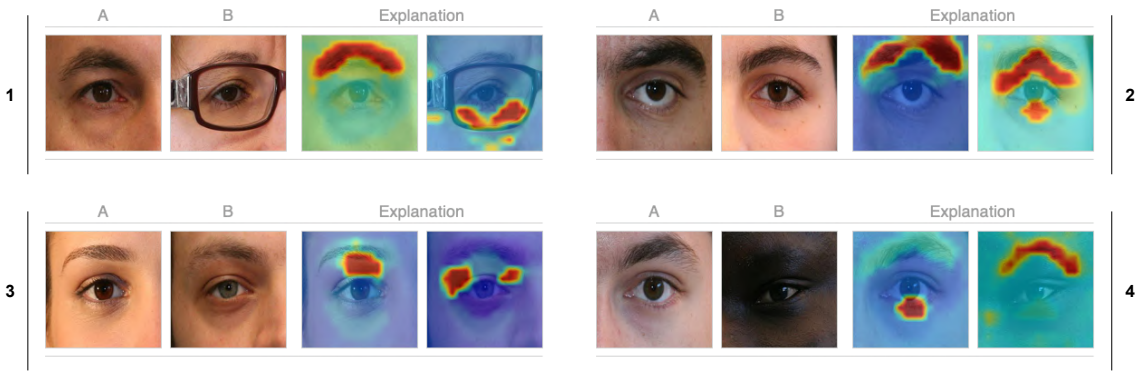

Figure 4.5: Impostor pairs explained with the method by Huang and Li. Such method produces heat maps in which red areas are the most significant.

These results are already decent and certainly keep explainability at the forefront. However, there are cases where these techniques miss obvious components, and we argue improvements can be made with approaches that are specifically designed to address this type of problem.

Considering the points established above, our deep framework attempts to produce both readable and effective results. As seen in Fig. 4.6, our explanations follow the same colour coding as KernelSHAP (i.e., green for irrelevant features and red for relevant ones). We produce immediately discernible explanations, highlighting, where applicable, eyebrows, irises, skins, glasses and skin spots. More specifically, to explain the first pair our approach gave emphasis to the glasses and, to a lesser scale, person A's skin texture. For pair number four, the explanation shows how decisive the skins were to an "impostor" decision, while in pair five one of the eyelids and both eyebrows are shown to be different. Furthermore, an obvious disparity in eyebrow thicknesses is clearly portrayed in pair six, while pair eight is perhaps the pinnacle of what our method can highlight: contrasting iris colours and the totally different eyebrows. Finally, the last row contains samples with generally accurate explanations, thus proving our solution's effectiveness.

In broad terms, we argue that our approach delivers explanations that are easy to understand, categorically stating *why* a decision was taken (in this case, "impostor").

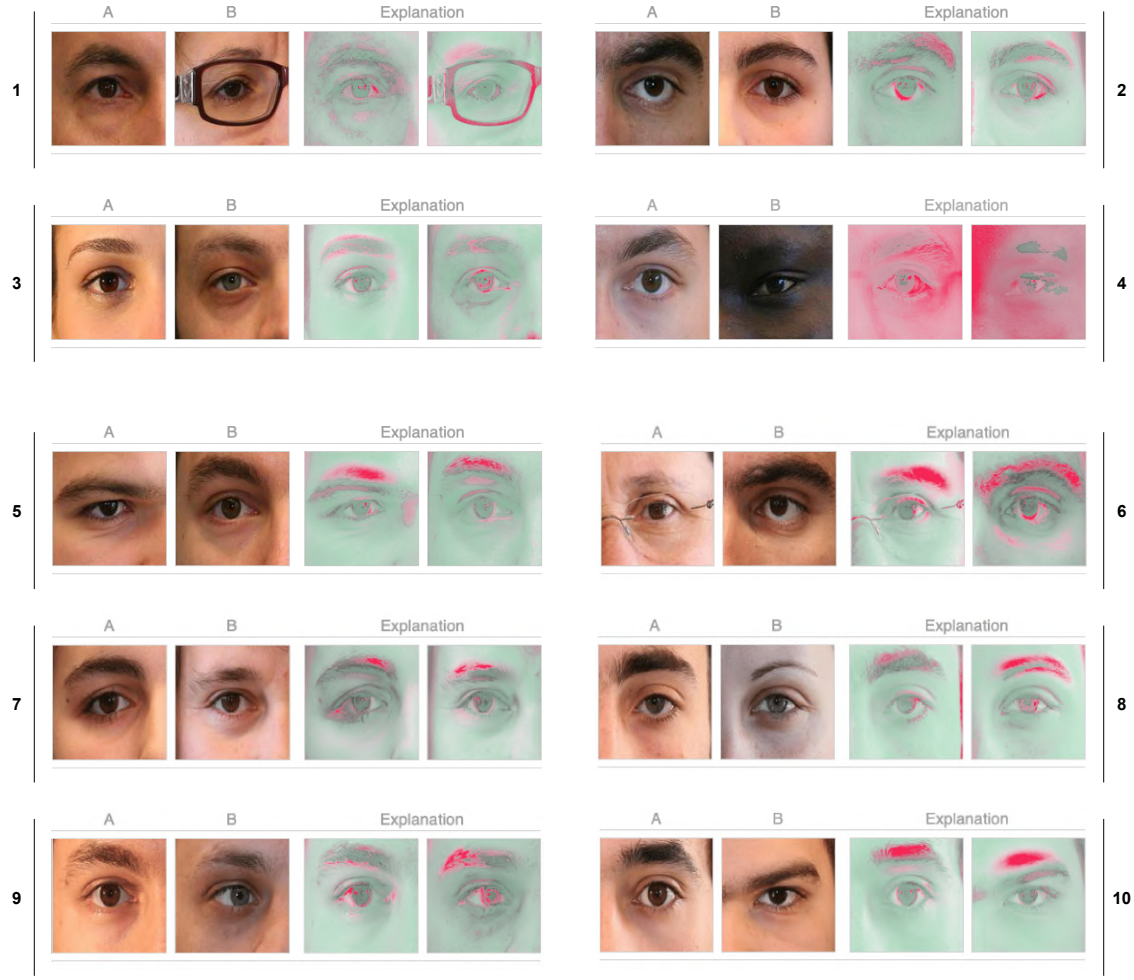

Figure 4.6: Results obtained with the first method. The top four pairs can be directly compared with the results seen in Figs. 4.2, 4.3, 4.4 and 4.5, whereas the bottom six pairs are exclusive to our method, so as to test it against a broader set.

When objectively measuring the differences between the explanations provided by the proposed method and the baselines (LIME, SHAP, Huang and Li (HL) and Saliency Maps (SM)), we used a set of 100 heterogeneous test queries and measured the pixel-wise explanation coefficients returned by each technique, which correspond to the importance (weight) given by each method to a particular image position for a decision. Next, considering that any meaningful correlations between the responses of two methods would have to be linear, we measured the Pearson's linear correlation between pairs of techniques:

$$r_{xy} = \frac{\sum_i (x_i - \hat{x})(y_i - \hat{y})}{\sqrt{\sum_i (x_i - \hat{x})^2 \sum_i (y_i - \hat{y})^2}}, \quad (4.1)$$

where x_i / y_i denote the i^{th} scores provided by each technique and the $\hat{}$ symbol denotes the mean value. This way, r_{xy} measures how similar are the explanations provided by the x and y techniques: values close to 0 will correspond to more independent explanations, while values towards 1 will point for semantic similarities between the explanations provided by both techniques.

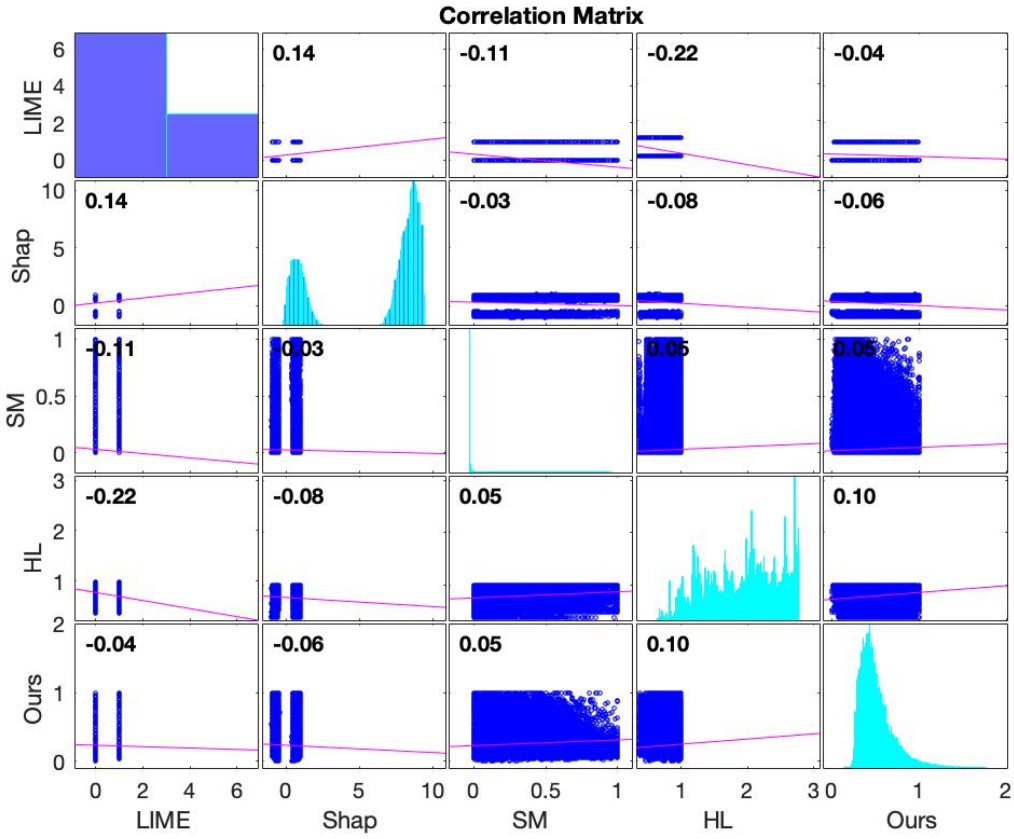

Figure 4.7: Pearson correlation values between the pixel-wise responses provided by the method proposed in this document (Ours) and four baselines techniques (LIME, SHAP, Huang and Li (HL) and Saliency Maps (SM)).

The results are conveyed through the confusion matrices shown in Fig. 4.7, where the main diagonal provides the distributions of the scores generated by each technique and the remaining cells provide the scatter plots between pairs of techniques with the Pearson's correlation value r_{xy} given at the top-left corner of each cell ("SM" stands for Saliency Maps and "HL" denotes the Huang and Li solution)). All these techniques report a local numeric value that corresponds to the role/importance of each region in the final decision. The exception is LIME, where the pixels are discriminated in a binary manner (i.e., "visible" or "occluded"). In this case, we consider "visible" to be equal to 1 and "occluded" to 0.

Overall, we observed that the techniques provide relatively independent responses for the importance given to each pixel in the final decision. Interestingly, in some cases, there are even negative correlation values between two methods (e.g., HL and LIME or SM and LIME). There are other pairs of solutions that achieved almost full independence between their responses (the Shapley/Ours methods), which points for completely different strategies being used to define the explaining regions/features. Regarding our method, its levels of correlation were kept relatively low with respect to the remaining methodologies, achieving values of 0.24 with respect to the method of Huang and Li (the most

correlated), and 0.1 for Saliency maps. Still, we conclude that the proposed solution is extracting semantic information (e.g., features and regions) of the vicinity of the human eye that is evidently different of the kind of information emphasised by any of the remaining methods, which supports the usefulness of the solution described in this document.

4.2.2 Recognition Accuracy Evaluation

At first, note that we do not aim at providing a better recognition framework than the state-of-the-art, in terms of the recognition rates. Even though, our main purpose in this section was to perceive if the proposed recognition/explanation network is able to achieve competitive recognition performance with respect to the state-of-the-art.

We compare the recognition effectiveness of the proposed method with respect to a well known periocular recognition model (due to Zhao and Kumar [ZK17], considered to represent the state-of-the-art). In order to assess the performance of both methods, the EER and AUC metrics were used.
On one hand, EER measures the threshold at which we have an equality between a system's *false acceptance rate* (i.e., the percentage of samples in which the system output a 1 when it should have been a 0) and *false rejection rate* (i.e., the percentage of samples that a system classifies as 0 when they are, in fact, of class 1). Ideally, the EER should be as small as possible, therefore indicating that a system is highly accurate.
On the other hand, AUC is typically applied on binary systems (like ours) and it quantifies the system's ability to distinguish between classes. To compute such metric, we must start by obtaining a ROC curve (similar to the one shown in Fig. 4.8). This graph plots the relation between True Positive Rate (TPR) and False Positive Rate (FPR) values at many different decision thresholds (i.e, the values beneath which our systems considers a sample to be of class 0 and above which the sample belongs to class 1). AUC, as the name suggests, stands for the area directly beneath the ROC curve. In a perfect situation, the curve would touch the $(0, 1)$ point and the area would obviously be 1. Conversely, a random classifier would be closer to the red dashed line in Fig. 4.8, meaning that no threshold would save it from being a bad model.

Using the UBIRIs.v2 set [OA10] and the learning/evaluation protocols described in [ZK17], we obtained the results summarised in Table 4.1. Also, we provide ROC values of the proposed strategy, that can be fairly combined with the similar ROC plot provided by the original authors of the baseline in [ZK18].
A bootstrapping-like strategy was used, by sampling 90% of the available data in UBIRIS.v2 and dividing the resulting samples between two disjoint sets: 80% for training and the remaining 20% for test. The models were trained separately in each sample and the performance evaluated in the corresponding test set, from where the EER and AUC scores were obtained. This process was repeated 10 times, to perceive the mean $\pm$ standard deviation values for both metrics. Overall, results were satisfactory, particularly considering that - due to our modular design - the recognition module of the proposed framework can be

easily replaced by any other, while keeping its explainability abilities.

Method	EER	AUC
Ours (open-world)	$0.108 \pm 3e{-2}$	$0.813 \pm 5e{-2}$
Ours (closed-world)	$\mathbf{0.087 \pm 2e{-2}}$	$\mathbf{0.910 \pm 2e{-2}}$
Zhao and Kumar [ZK17]	$0.109 \pm 2e{-3}$	–

Table 4.1: Comparison between the recognition rates attained by the proposed method (in both world settings) and a state-of-the-art method (strictly operating in an open-world setting). Results are given for the same learning/test sets of the UBIRIS.v2 dataset.

For reference purposes, Fig. 4.8 provides the ROC curve of our solution. When drawing comparisons with the corresponding results reported by authors in [ZK17] in the same set, it can be seen a close recognition summary performance between both methods (summarised in Table 4.1). Overall, we observed a similar performance between these techniques in this dataset, which supports the idea that the proposed solution is able to approach state-of-the-art recognition rates.

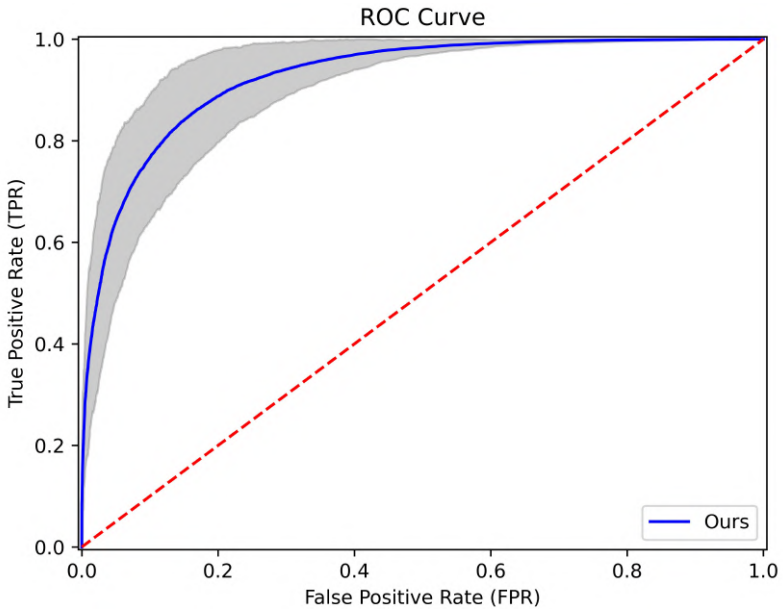

Figure 4.8: ROC curve obtained for the proposed method in data of the UBIRIS.v2 set, according to the empirical protocol designed by Zhao and Kumar [ZK17]. The ROC curve corresponds to EER and AUC values of about 0.108 and 0.813, respectively.

4.2.3 Inference Time Evaluation

Besides explainability and accuracy metrics, one can assess the time span that any given technique needs to produce an explanation. As such, Table 4.2 presents the inference times attained by each method in 10 randomly sampled test pairs (so as to obtain mean $\pm$ standard deviation values):

Method	Inference Time (m)
Saliency Maps (SM) [VZ14]	15.71 ± 0.24
LIME [SG16]	3.64 ± 0.05
SHAP [LL17]	3.14 ± 0.04
Huang and LI (HL) [HL20]	$\mathbf{0.08 \pm 0.01}$
Ours	3.23 ± 1.75

Table 4.2: Comparison between the mean inference times (in minutes) attained by our approach and four baseline techniques (LIME, SHAP, Huang and Li (HL) and Saliency Maps (SM)). HL stands out from the rest, mainly due to the fact that it is, essentially, a CNN with some extra steps for explainability, thus leveraging the swift inference times that CNNs usually have.

The results shown above indicate that our approach is competitive against all but one technique (HL, which fairs much better than its competitors). In terms of inference times alone, it becomes clear how the permutation and search based techniques end up consuming more time when either feeding a black-box model with artificial permutations or searching amongst a synthetic dataset. Nonetheless, these results should not be analysed in isolation, since they are entangled with the explanations seen in subsection 4.2.1.

4.2.4 Ablation Studies

For our ablation experiments, we identified two hyper-parameters of our method that might play the most significant roles in the final effectiveness of the whole solution: **1**) the number of neighbours retrieved (K) from the synthetic set for every query and **2**) the length of the synthetic set itself. This section discusses how changes in these values affect the quality of the generated explanations in a less than optimal way (as seen in Fig. 4.9).

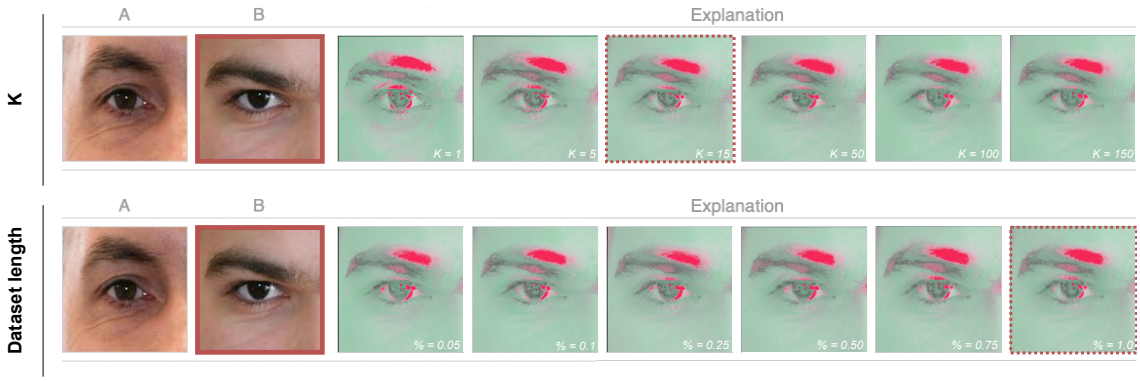

Figure 4.9: Typical changes in the results when the two most important parameters of the proposed method are varied. The red square indicates which image is being explained (i.e., B), while the red dashed squares provide the default values used in our experiments. In general, increasing K up to 15 allows for smoother explanations, as does keeping a large dataset. Reducing the latter tends to produce less sensitive results, substantially decreasing the plausibility of the visual explanations.

4.2.4.1 Number of Neighbours

The value K determines how many synthetic pairs are considered with respect to a query. Overall, we observed that smaller values lead to more sensitive and jagged results. Up to a certain point (e.g., 15), increasing K typically enables to obtain smoother explanations, due to the larger number of samples taken into account when averaging the closest neighbours. This trend, however, starts returning incremental improvements (notice in Fig. 4.9, where $K >= 50$ progressively stops presenting a prominent tone on the eyelid).

4.2.4.2 Length of the Synthetic Dataset

This is the most sensitive parameter of our solution. Considering that it is important to find ”genuine” pairs that closely resemble a query, it is particularly sensitive to assure that all typical periocular data variations are faithfully represented in the synthetic set, assuring that the retrieved elements (i.e., the most similar) will have its major components (iris, eyebrows and eyelids) aligned to the query itself. If this condition is not satisfied, the explanations lose their biological plausibility and effectiveness. Fig. 4.9 illustrates how smaller synthetic sets lead to less evident explanations, especially around the eyelid and the eyebrow.

4.3 Automatic Generation of Image Captions

As the first method, the proposed solution for image captioning attempts to produce explanations that are as easy to understand as possible. With effect, the generated sentences are syntactically correct, following the training distribution, which included many different ways of opening or closing a sentence and conveying the relevant information (i.e., which periocular components sustain an ”impostor” decision).

Practically speaking, the first pair has evidently different eyelid shapes and, in the second, the people have obvious differences in the eyebrows (both explanations were included in the generated captions). Next, in the third pair the eyebrow distributions are not different enough so as to justify their inclusion in the explanation (there is a skin spot that could better justify the ”impostor” decision). Then, in the fourth pair the explanation returns to a decent accuracy level, by specifying the skins, while in the fifth sample the iris colours are left behind in favour of the skin spots (once again, this is not totally wrong but the spots are perhaps not the most striking visual difference). Finally, the sixth pair is correctly explained by highlighting the skins and eyebrows.

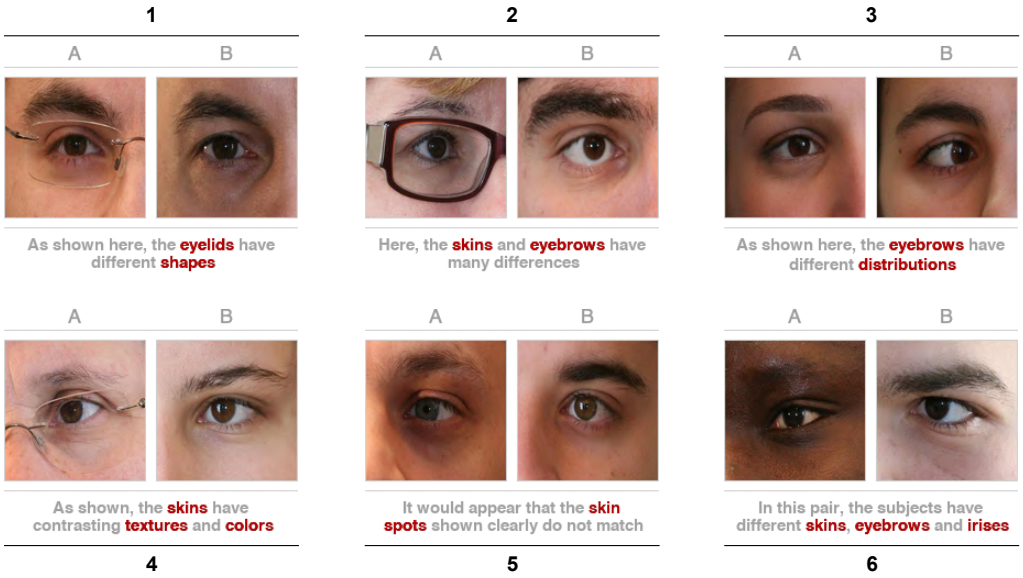

Figure 4.10: Samples generated by the proposed solution for automatic captioning. In general, the periocular components (in red) that are included in each explanation are fairly accurate, with some exceptions (e.g., fifth sample).

4.4 Conclusion

The present chapter attempted to validate whether explainability could be achieved in practical scenarios (and in several forms - visual and written, in this case). As seen through the experimental results, both solutions appear to produce satisfactory results (even when the most obvious components are not highlighted, the ones that end up being chosen are not technically wrong). Finally, an interesting possibility would be to combine the two approaches into a single, unified framework that produces more thorough explanations.

Chapter 5

Conclusions and Further Work

The present chapter is a general, high-level overview of the work that was developed in the scope of this dissertation. Therefore, the next paragraphs attempt to validate whether the main goal was achieved, what intuitions were developed and, if possible, how the work could be further improved.

Regarding the primary goal of this work (i.e., the adoption of *explainable* principles into a periocular recognition system), the two methods present different, yet equally substantial levels of readability. On one hand, the first method generates images with various shades of green or red, according to each pixel's contribution to an "impostor" decision. On the other hand, the second approach conveys its reasoning through text captions that highlight the most different periocular components.
As seen in the results chapter, we managed to deliver images and captions that, more often than not, faithfully explain the differences between two subjects. The first method was able to highlight relevant periocular components such as the irises, eyebrows, skins and even eyelashes. Compared to other implementations, our explanations are definitely easy to read and convey more useful information. Even in terms of the recognition task (i.e., excluding the explainable components), our method attained competitive results to those possible with an existing solution, despite that not being the main focus. As an additional benefit, the recognition stage could be replaced with other, possibly more performant approach, with virtually no cost for the explanations.
The second method that was proposed focused on the text domain by generating plausible captions for a given "impostor" pair. These text descriptions give emphasis to periocular components that stand out as being too different. In many cases, the captions included the most visually different components, while in others, less obvious (but still valid) components were preferred.
Based on both methods' results, it is expected that a combination of the two would produce even more complete results, with visual and text based explanations. Such system would take the premise of this dissertation even further.

Finally, further development stages of this work could focus on **1**) reducing the time that it takes to generate "impostor" explanations and **2**) allowing "genuine" pairs to be explained as well. With regard to the first point, our method is relatively consistent with LIME and SHAP (while being much better than Saliency Maps), but falls quite behind Huang and Li's method. Consequently, as a way to close this gap, a potential direction in the future could improve the way in which our synthetic dataset is structured. Currently, we mainly divide the images in terms of the position of the irises. This ensures that only a suitable portion

of the images ends up being used, avoiding useless calculations. In addition, we could also store the images based on their attributes (i.e., iris colour, eyebrow density, amongst others). By doing so, in inference time, we could have additional processing to determine the attributes of both test images, that make up a pair, and only use the synthetic samples that meet the iris position and attribute constraints, hopefully saving time altogether.
As for explaining ”genuine” pairs, a different approach would have to be added, so as to fully cover the possible outputs of our recognition system. Solutions like keypoint matching or image registration could help us understand how little an image A needs to be changed so that it closely matches an image B (which should, more often than not, be the case with ”genuine” pairs).
The above improvements (and others that are not described here) could make our work even more valuable, proving that we can take an existing task and make it more transparent for the end user.

A Deep Adversarial Framework for Visually Explainable Periocular Recognition

João Brito and Hugo Proença
IT: Instituto de Telecomunicações
University of Beira Interior
6200-001, Covilhã, Portugal
joao.pedro.brito@ubi.pt, hugomcp@di.ubi.pt

Abstract

*In the biometrics context, the ability to provide the reasoning behind a decision has been at the core of major research efforts. These explanations have powerful benefits, such as increasing trust amongst the users of a system and augmenting the system's overall accountability and transparency. In this work, we describe a periocular recognition framework that not only performs biometric recognition but also provides visual representations of the features/regions that supported a decision. Being particularly designed to explain non-match ("*impostors*") decisions, our solution uses adversarial generative techniques to synthesise a large set of "*genuine*" image pairs, from where the most similar elements with respect to a query are retrieved. Then, assuming the alignment between the query/retrieved pairs, the element-wise differences between the query and a weighted average of the retrieved elements yields a visual explanation of the regions in the query pair that* would have to be different *to transform it into a "*genuine*" pair. Our quantitative and qualitative experiments validate the proposed solution, yielding recognition rates that are similar to the state-of-the-art, but - most importantly - also providing the visual explanations for every decision.*

1. Introduction

This work describes an integrated framework for periocular biometric recognition which - apart performing the recognition task - also provides a visual explanation that sustains every decision. Considering the biometric recognition ubiquity and dependability [21], our main goal in this paper is not to propose a *better* recognition framework in terms of the error rates, but to particularly diverge of the black-box paradigm and follow a *visually explainable* paradigm, as illustrated in Fig. 1.

[0]The code is publicly available at https://github.com/ojoaobrito/ExplainablePR.git

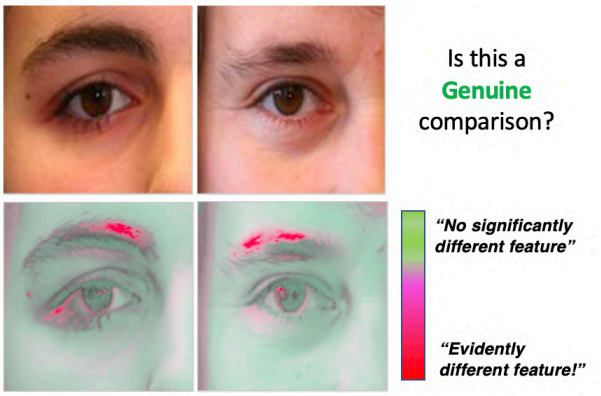

Figure 1. Key insight of the proposed visual explainable framework: given a pair of images, the system **not only reports a binary decision** ("*genuine*"/"*impostor*" classes), but also **highlights the regions in each sample that contributed the most in case of a *non-match* decision**. In this example, yet the iris and skin colour are similar between samples, the eyebrows and eyelashes shapes are evidently different, along with a skin spot in the sample illustrated at the left side. These are exactly the regions highlighted in the visual explanations.

Typically, a recognition problem involves a set of unique and non-transferable features that can unmistakably identify a subject. Biometric traits, as they are designated in the field, serve such purposes, as long as they are universal, distinguishable, resilient to changes and easy to collect [16]. Upon proving their compliance with these requirements, biometric traits can be divided into two major categories:

1. *Physiological* features (e.g., the iris, fingerprint and retina) that are naturally possessed by a given subject;

2. *Behavioural* biometrics, that yield from the interaction between a subject and the surrounding environment (e.g., the gait and handwritten signature) [2].

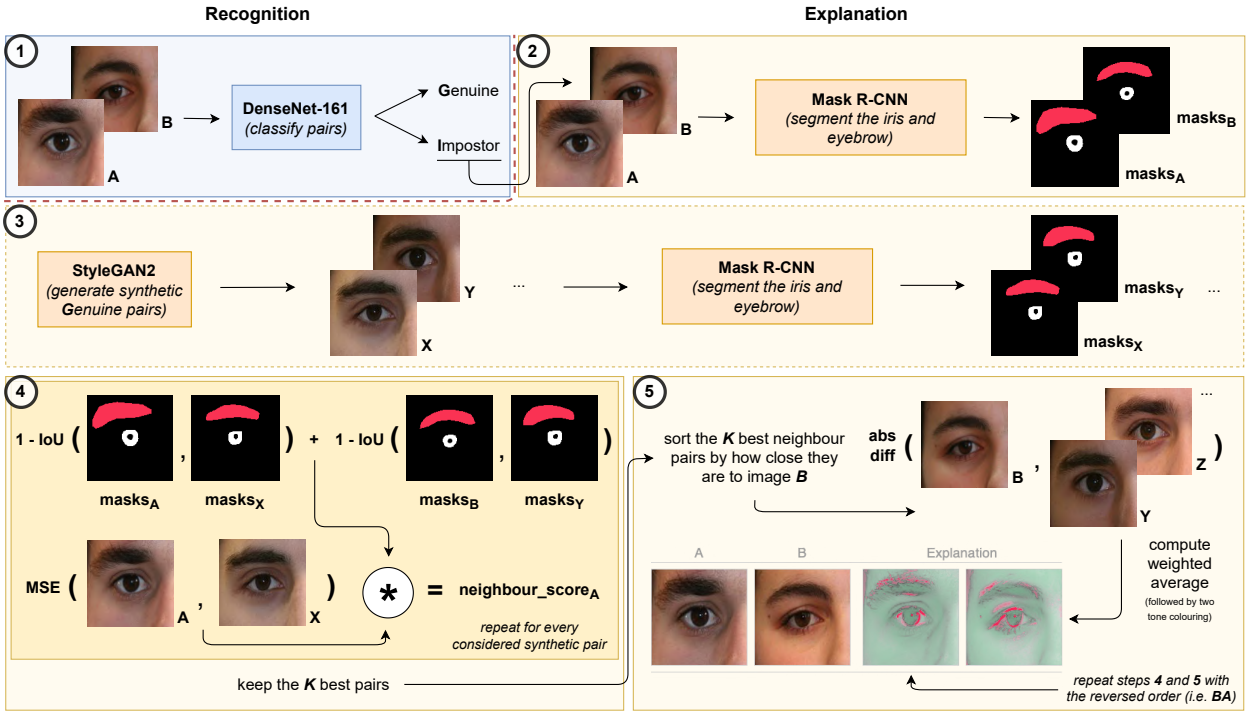

Figure 2. Cohesive perspective of the main pipeline of the proposed solution. The first step (recognition) encompasses a CNN that distinguishes between "*genuine*" and "*impostor*" pairs. Then, upon an "*impostor*" decision, steps two to five (explanation) find the k "*genuine*" synthetic pairs among a large set that most closely resemble the query pair. Assuming the alignment between the query and the retrieved pairs, the element-wise differences between the query and a weighted average of the retrieved elements provides a visual explanation of the regions/features in the query that would have to be different to turn the query into a "*genuine*" pair.

Concentrating growing interests in the biometrics domain, periocular recognition uses the information in the vicinity of the eye to perform recognition, in which the iris, sclera, eyebrow, eyelid and skin stand out.

Regarding the concept of *explainability* and its application to recognition problems, it should be noted that Deep Learning solutions rely on model complexity and abstraction prowess to become truly accurate. Although seemingly innocuous, there could be seriously negative outcomes if such *black-boxes* gamble on the clearance of unauthorised people into sensible areas. Hence, it is particularly important to provide human understandable explanations of the decisions, which will augment the overall system accountability and transparency, enabling a broader range of applications (i.e., forensics). Recently, the EU, through the GDPR [3], introduced the notion of "*right to an explanation*". Even though the definition and scope of such explanations are still subject to debate [10], these are definite strides towards a formal regulation regarding the importance given to the concept of explainability.

According to the above points, this paper describes a framework that receives a pair of images and returns a two-fold output: 1) a binary match/non-match decision, that discriminates between the "*genuine*"/"*impostor*" pairs; and 2) a *visual explanation* that highlights the features/regions of the input data that sustained a particular decision. This is considered the main contribution of our work, in the sense that - to the best of our knowledge - it is the first that creates an accurate and explainable representation of the reasons behind certain decisions of the recognition system. Other contributions include the use of Generative Adversarial Networks (GANs), to synthesise visually pleasant images pair that faithfully resemble the distribution of the "*genuine*" pairs, which augments the variety and flexibility of the learning set and can be seen as an alternate form of data augmentation.

Fig. 2 provides a cohesive overview of the framework that performs the recognition task and provides the corresponding explanations: at first, a CNN (of a well known architecture) is trained to discriminate between match/non-match decisions. If the pair is deemed to belong to the "*impostors*" distribution, we find its most similar "*genuine*" pairs in a large set of synthetic data. The insight here is that, even if the query pair has significant differences between its elements that led to an "*impostor*"

decision, the closest synthetic pairs most likely do not (as they were drawn from the "*genuine*" distribution). Then, assuming that the most likely synthetic pairs and the query are sufficiently aligned, obtaining the pixel-wise weighted differences between them will elevate visual disparities.

The remainder of this paper is organised as follows: Section 2 summarises the most relevant research in the fields of periocular recognition and Machine Learning Explainability. Section 3 describes our method and Section 4 analyses the results obtained. Section 5 concludes this paper, while also providing some final remarks.

2. Related Work

2.1. Periocular Recognition

The seminal breakthroughs in the periocular recognition problem can be traced to a set of methods termed *feature descriptors*. Methods such as HoG, LBP and SIFT were able to produce simplified data representations by relying on edges, textures and keypoints, respectively. In [17], the results from each feature descriptor were fused to faithfully discriminate between the "*genuine*"/"*impostor*" pairs. This work served as basis for subsequent fusion-based approaches, as in [14]. In [6] a Restricted Boltzmann Machine was used to learn a probabilistic distribution over the input data, further discriminated with metric learning and SVMs.

With the effective application of Deep Learning solutions, researchers turned to popular architectures (in particular Convolutional Neural Networks), to pursue ever increasing recognition accuracy. Accordingly, in [23] the main concept involves the use of multiple CNNs that are specialised in classifying a particular kind of semantic information (e.g., gender or age). Then, a score fusion process yields the final response. In [15], authors enforce a CNN to ignore the ocular region (due to its likelihood to contain specular reflections) and rely in the eye's surrounding area (eyebrow, eyelid and skin). [18] created independent representations of the iris and periocular regions, that feed classification modules, whose scores are finally fused to reach the decision. Using a multi-glance mechanism, where part of the intermediate components are configured to incorporate emphasis on the most important semantical regions (i.e., eyebrow and eye), Zhao and Kumar [24] developed a recognition model that particularly focus these regions, enabling the deep Convolutional Neural Network (CNN) to learn additional discriminative features that improve the recognition capability of the whole model. Recently, [19] attempted to bridge the gap between biometric recognition and interpretability, by learning feature specific filters that respond to a range of preferred spatial locations. [5] propose an integrated solution that leverages the discovery of parts as a form of attention.

2.2. Machine Learning Explainability

In the literature, the existing explainable techniques are commonly divided in terms of their depth, scope and model applicability [8], [11]. Depth is related to the length to which we explain a given model, i.e., whether the technique limits the model's complexity to make it more transparent (*intrinsic* explainability) or allows complexity and focuses on explaining exclusively the system outputs (*post hoc* explainability). Scope indicates the range that a technique possesses, i.e., if it explains individual predictions (*local*) or the model's entire behaviour (*global*). Finally, applicability divides the techniques based on their model affinity, i.e., whether they are only compatible with a specific family of models (*model-specific*) or any kind of model (*model-agnostic*). The most commonly cited techniques include LIME [20] and Shapley codes (SHAP) [9]. The former uses a surrogate linear model, trained on perturbed data (e.g., disabled clusters of adjacent pixels), to locally approximate the behaviour of a complex black-box model. The latter uses game theory and Shapley values, which are assigned to the features based on how important they are to a given prediction. Additionally, Saliency Maps [22] use the derivative of a highly complex function (essentially, a CNN) with respect to a given input image, to determine which pixels need to be changed the least, while also changing the output class the most. Finally, for visualisation purposes and, therefore, outside the scope of this work, PDP [4] and ALE [1] techniques are able to produce plots that correlate the independent variables to a target variable, exploiting the notions of marginal and conditional distributions, respectively.

3. Proposed Method

3.1. Learning Phase

The main components of the proposed method comprise three well known models: the DenseNet-161, Mask R-CNN and StyleGAN2. The first one (DenseNet-161) is trained to solve an identity verification problem, while the segmentation model (Mask R-CNN) is fine-tuned to produce high-quality masks for the iris and eyebrow. Finally, the GAN model (StyleGAN2) learns how to create synthetic data that, while closely resembling the distributions in the training set, is diverse enough to approximate unseen subjects. Additionally, a fourth, auxiliary model (ResNet-18) is fitted to discriminate between images from the left and right sides of the face. Although trained separately, all the models learn from the same training split, which excludes a set of disjoint IDs that are reserved for performance evaluation purposes.

Regarding the model used in the verification task (DenseNet-161), it should be stated that it has much more parameters than the network used by Zhao and Kumar [23] in their solution. This might be the fact that sustained slightly better recognition performance of our model with

respect to the baseline (Sec. 4.3), but also at the expense of a substantial higher computational cost of classification than the baseline, which might be impracticable in some cases.

3.2. Inference Phase

Once trained, our method is conceptually divided into five major steps, as depicted in Fig. 2. Firstly, the DenseNet-161 model is used to verify the claimed identity: upon receiving a pair of images, the model discriminates between "*genuine*"/"*impostor*" pairs. If the pair is deemed to be "*impostor*", the remaining steps create a visually interpretable explanation of that decision.

The second step takes the query pair and, using Mask R-CNN, segments the irises and eyebrows regions. Next, step three uses the StyleGAN2 generator to create a large, synthetic set of exclusively "*genuine*" pairs (i.e., where both images belong to the same person). For each of these synthetic pairs, the ResNet-18 model determines its side configuration (i.e., whether images regard the left or right side of the face) and, as before, masks are obtained by the segmentation model.

After obtaining the synthetic data and their corresponding masks, the synthetic dataset is indexed based on the coordinates of the center of the iris, which will enable faster search in the retrieval step. To that end, the clustering algorithm K-Means is trained on a subset of the iris segmentation masks to obtain three centroids, one for each major iris gaze family (i.e., left, centre and right). This way, we index the available pairs based on their combination of iris positions (e.g., left-left, right-centre ...). By doing so, when searching, we can just rely on the synthetic pairs that share the same combination as the test pair, saving time and useless calculations.

Upon settling for a portion of the synthetic dataset that closely meets the iris position constraint, the segmentation masks are further used to determine which synthetic pairs have the iris and eyebrow approximately overlapped to the query. This is an important requirement to obtain visually pleasant explanations, given that pixel-wise differences are extremely sensitive to differences in phase (i.e., component misalignment). Accordingly, we obtain a similarity score s_X between each synthetic neighbour and the query using:

$$s_X = \omega_{\text{masks}} * ||\text{query}_\text{A} - \text{neighbour}_\text{X}||_2, \qquad (1)$$

being $||.||_2$ the $\ell - 2$ norm and $\omega_.$ a weight that considers component misalignment. This way, we obtain a weighted distance between each synthetic neighbour and the first image of the query pair. ω_{masks} values serve to favour pairs that have good alignment, considering $1 - \text{IoU}(.,.)$, i.e., the complement of the intersection-over-union of the synthetic/query segmentation masks. In practice, we search amongst the (large) thousands of synthetic pairs, the closest to the query pair in terms of the first image. Therefore, given that the second image of the query pair is from a different subject, it will most likely have features that are different to the synthetic neighbours, which are exactly the kind of dissimilarities that make up the final explanations.

This way, the k closest neighbours are sorted according to their element-wise distance to image B, using (2). Finally, to produce the final explanation, the k best neighbours are used to obtain the pixel-wise differences against the query pair image B. In practice, a neighbour distance is subtracted from the total sum of distances, creating an inverted distance. This assures that the contribution of the closest synthetic neighbours to the final result is more important than of those with bigger distances.

3.3. Implementation Details

The DenseNet-161 model was trained for 15 epochs with a learning rate of 0.0002 and a batch size of 64 image pairs. The Adam algorithm was used for the weight optimisation process (with default β_1 and β_2 values). A similar training setup was used to train the ResNet-18 model, albeit for a smaller number of epochs (i.e., 5). For the Mask R-CNN's training process, we kept its default values, using a learning rate of 0.001, a batch size of 1 and 30 epochs worth of training (in this case, fine-tuning from the COCO pre-trained weights). Regarding the StyleGAN2 architecture, the used training step comprised a total of 80000 iterations and a batch size of 8. After converging, the generator is capable of synthesising realistic looking images, such as the roughly 400000 pairs that make up the artificial dataset. Finally, for the number k, that determines how many synthetic pairs should be kept, we used a default value of 15.

4. Experiments and Discussion

4.1. Datasets and Working Scenario

As mentioned above, the proposed framework is composed of two modules: 1) one for recognition; and 2) the other for explanation purposes. Regarding the former, the chosen CNN is solely trained on the UBIPr dataset [13], which provides the ID annotations used in the identity verification problem. Regarding the explanation step, it mainly relies on a combination of UBIPr and FFHQ [7]. Despite not being directly applicable to the context of this work (i.e., it contains full face images, thus requiring extra steps to extract the periocular region), the FFHQ dataset contains a large variety in terms of periocular attributes, some of which are scarcer in the UBIPr dataset. In practice, a small, but curated, portion of the FFHQ samples was used to create a data super set. Regardless of their source, all images were resized to a common shape, depending on the task (i.e., 512x512x3 for Mask R-CNN, 256x256x3 for StyleGAN2 and 128x128x3 for the CNNs).

As it is usual in the biometric recognition context, it is important to define proper working modes and world settings, for which the system is built. With respect to the working mode, our model runs in verification mode (also referred to as *one-to-one*), where the system validates a claimed identity [16]. As for the world setting, we assume an open-world setting, meaning that unseen subjects can be faithfully handled in the inference step.

4.2. Explainability Evaluation

Our explainability chain starts by the train of a DenseNet-121 model to perform the verification task. This model can be further paired to either LIME, SHAP or Saliency Maps to create comprehensive comparison schemes, to which we add the method described in [5]. Fig. 4 provides several examples of the synthetic "*genuine*" images pairs generated from the GAN model. Apart their obvious visual realism, it is important that this set contains samples with the most likely known data covariates for the periocular region: varying gazes, wide-opened/closed eyes, varying poses, partial occlusions, and even varying facial expressions. Failing in incorporating such diversity will determine that the closets synthetic pairs of a query will still be notoriously different from it, and that the visual representations obtained will have poor realism.

Fig. 3 displays the expected results from a visually explainable system. In practice, LIME tries to keep the most important super-pixels, SHAP highlights those it deems important in red tones and Saliency Maps produce greyscale explanations. As for the method by Huang and Li, it generates a heat-map in which red tones elevate important areas. Focusing on the common pairs between all methods, the left sample is essentially different with regards to eyebrow thickness and presence/absence of a noticeable skin spot. As for the right one, the most obvious disparities have to do with the eyebrow areas. Overall, our results are the most informative, when compared with the remaining four solutions. While LIME and SHAP do a decent job, Saliency Maps provide a faint explanation. It is Huang and Li's method that comes closer to our level of visual appeal, by clearly highlighting portions of the eyebrow and a portion of subject A's skin spot, in the left pair. Moreover, when given the right sample, it generates a solid red area comprising subject B's eyebrow. However, upon closer inspection, our results show more appealing visual cues: in the left sample, distinct red tones on top of A's skin spot and eyelashes, as well as, reiterated eyebrow differences in the right sample with highlights in both eyebrows, rather than just one. As for the remaining samples, the third (just below the first) is clearly explained by highlighting the entirety of both skin areas, which are obviously different between images A and B. Finally, in the fourth pair it is also shown how the eyelids differ, by colouring that periocular component on subject B's image, and, in the fifth sample, subjects B's eyebrow and iris are accurately shown in red.

When objectively measuring the differences between the explanations provided by the proposed method and the baselines (LIME, SHAP, Huang and Li (HL) and Saliency Maps (SM)), we used a set of 10 heterogeneous test queries and measured the pixel-wise explanation coefficients returned by each technique, which correspond to the importance (weight) given by each method to a particular image position for a decision. Next, considering that any meaningful correlations between the responses of two methods would have to be linear, we measured the Pearson's linear correlation between pairs of techniques:

$$r_{xy} = \frac{\sum_i (x_i - \hat{x})(y_i - \hat{y})}{\sqrt{\sum_i (x_i - \hat{x})^2 \sum_i (y_i - \hat{y})^2}}, \tag{2}$$

where x_i/y_i denote the ith scores provided by each technique and the $\hat{}$ symbol denotes the mean value. This way, r_{xy} measures the similarity between explanations provided by the x and y techniques: values close to 0 will correspond to more independent explanations, while values towards 1 will hint at semantic similarities between such explanations.

The results are provided in the confusion matrices shown at Fig. 5, where the main diagonal provides the distributions of the scores generated by each technique and the remaining cells provide the scatter plots between pairs of techniques with the Pearson's correlation value r_{xy} given at the top left corner of each cell ('SM' stand for Saliency Maps and 'HL' denotes the Huang and Li solution)). All these techniques report a local numeric value that corresponds to the role/importance of each region in the final decision. The exception is LIME, where the pixels are binary discriminated into "visible"/"occluded'. In this case, we considered that "visible" will be equal to 1, while 'occluded" will be equal to 0. Overall, we observed that the techniques provide relatively independent responses for the importance given to each pixel in the final decision. Interestingly, in some cases, there are even negative correlation values between two methods (e.g., HL and LIME or SM and LIME). There are other pairs of solutions that achieved almost full independence between their responses (the Shapley/Ours methods), which points for completely different strategies being used to define the explaining regions/features. Still considering our method, its levels of correlation were kept relatively low with respect to the remaining methodologies, achieving values of 0.24 with respect to the method of Huang and Li (the most correlated), and 0.1 for Saliency maps. Still, we concluded that the proposed solution is extracting semantic information (e.g., features and regions) of the vicinity of the human eye that is evidently different of the kind of information emphasised by any of the remaining methods, which supports the usefulness of the solution described in this paper.

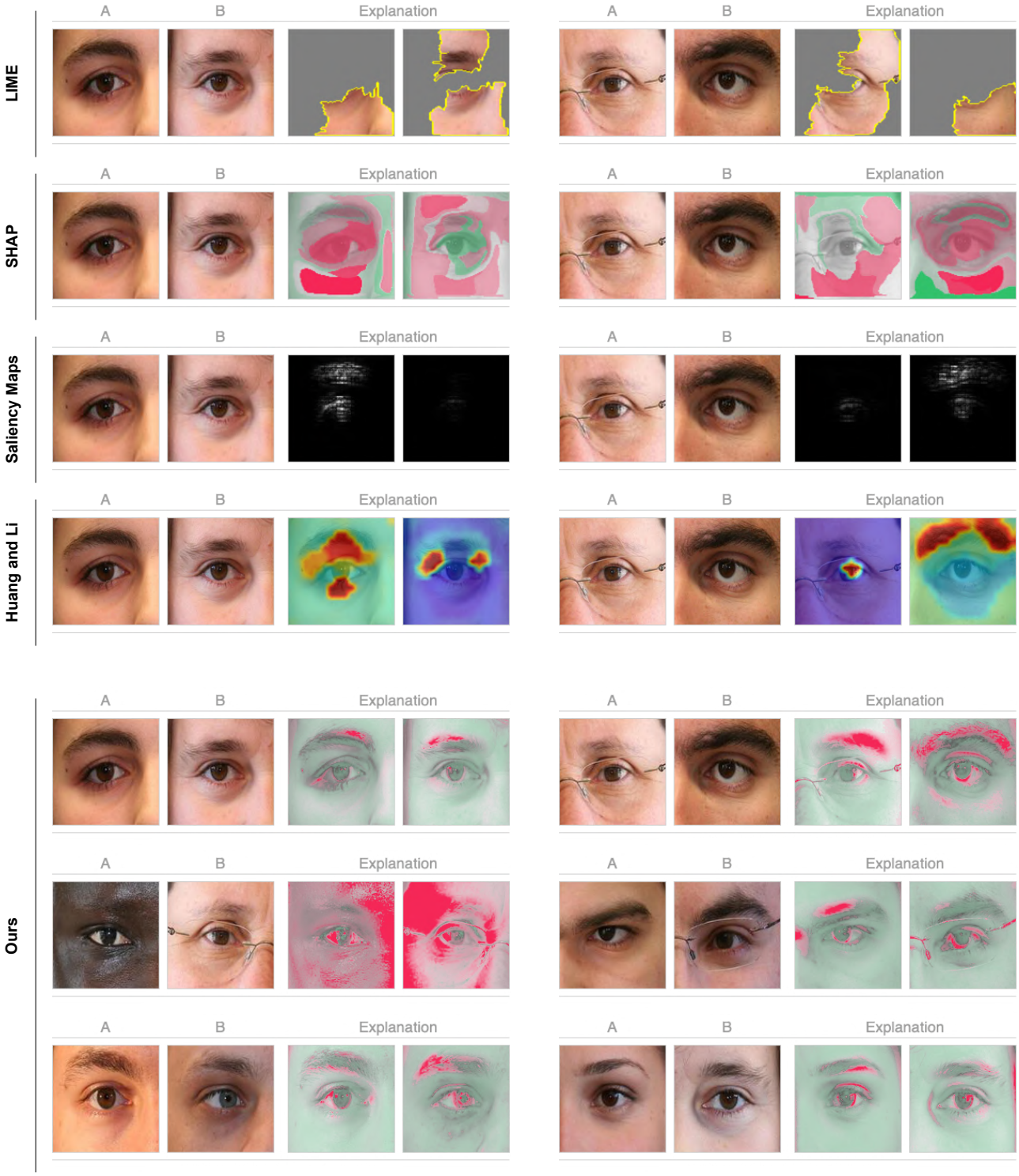

Figure 3. Examples of the results attained by three standard interpretability techniques (LIME, SHAP and Saliency Maps), a state-of-the-art interpretable deep model for fine-grained visual recognition (i.e., [5]) and our method. Notice how our results are clearer in highlighting the components that justify every non-match decision (e.g., skin texture and color, eyebrows/eyelashes size and distribution, irises color and even skin spots).

Figure 4. Examples of the synthetic image pairs in our dataset, generated according to a GAN model. These elements are drawn exclusively from the "genuine" distribution. Upon a query, the most similar synthetic pairs with respect to the query are found, which will provide the features/regions that would transform the query into a "*genuine*" comparison.

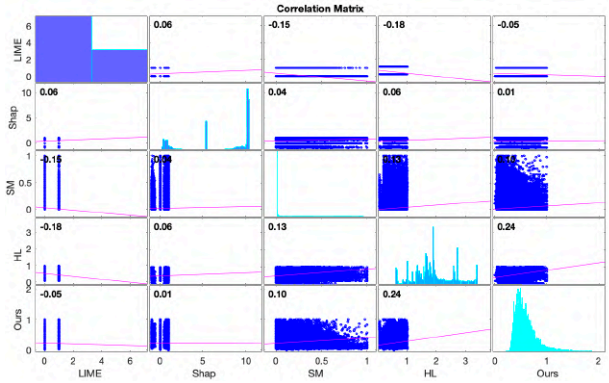

Figure 5. Pearson correlation values between the pixel-wise responses provided by the method proposed in this paper (Ours) and four baselines techniques (LIME, SHAP, Huang and Li (HL) and Saliency Maps (SM)).

4.3. Recognition Accuracy Evaluation

At first, note that we do not aim at providing a *better* recognition framework than the state-of-the-art, in terms of the recognition rates. Even though, our main purpose in this section was to perceive if the proposed recognition/explanation network is able to achieve competitive recognition performance with respect to state-of-the-art implementations.

We compare the recognition effectiveness of the proposed method with respect to a well known periocular recognition model (due to Zhao and Kumar [23], considered to represent the state-of-the-art). Using the UBIRIs.v2 set [12] and the learning/evaluation protocols described in [23], we obtained the results summarised in Table 1. Also, we provide ROC values of the proposed strategy, that can be fairly combined with the similar ROC plot provided by the original authors of the baseline in [24].

A bootstrapping-like strategy was used, by sampling 90% of the available data in UBIRIS.v2 and dividing the resulting samples between two disjoint sets: 80% for training and the remaining 20% for test. The models were trained separately in each sample and the performance evaluated in the corresponding test set, from where the EER and AUC scores were obtained. This process was repeated 10 times, to perceive the mean $\pm$ standard deviation values for both metrics. Overall, results were satisfactory, particularly considering that - due to our modular design - the recognition module of the proposed framework can be easily replaced by any other, while keeping its explainability abilities.

Method	EER	AUC
Ours (open-world)	$0.108 \pm 3e{-2}$	$0.813 \pm 5e{-2}$
Ours (closed-world)	$\mathbf{0.087 \pm 2e{-2}}$	$\mathbf{0.910 \pm 2e{-2}}$
Zhao and Kumar [23]	$0.109 \pm 2e{-3}$	–

Table 1. Comparison between the recognition rates attained by the proposed method (in both world settings) and a state-of-the-art method (strictly operating in an open-world setting). Results are given for the same learning/test sets of the UBIRIS.v2 dataset.

For reference purposes, Fig. 6 provides the Receiver Operating Characteristic (ROC) curve for our solution. When comparing to the corresponding results reported by authors in [23] in the same set, a close recognition summary performance between both methods can be derived (as seen in Table 1). Overall, we observed a similar performance between these techniques in this dataset, supporting the idea that our solution is able to approach state-of-the-art recognition rates.

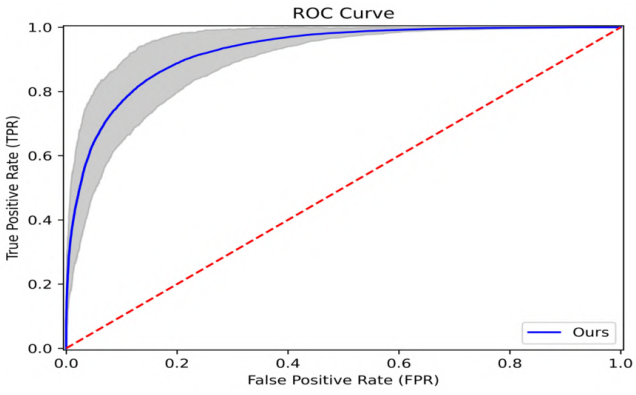

Figure 6. Receiver Operating Characteristic (ROC) curve obtained for the proposed method, using the UBIRIS.v2 set and a similar empirical protocol as Zhao and Kumar [23]. The ROC curve equates to EER and AUC values of 0.108 and 0.813, respectively.

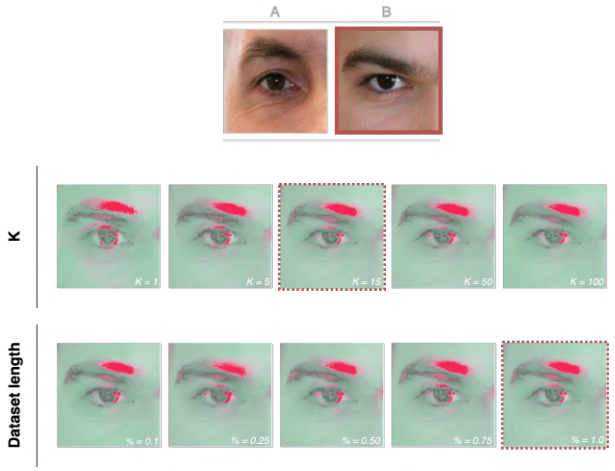

Figure 7. Typical changes in the results when two key parameters of the proposed method are varied. The red square indicates which image is being *explained* (i.e., B), while the red dashed squares provide the default values used. In general, increasing k up to 15 allows for smoother explanations, as does keeping a large dataset. Reducing the latter tends to produce less sensitive results, substantially decreasing the plausibility of the explanations generated.

4.4. Ablation Studies

For our ablation experiments, we identified two hyper-parameters of our method that might play the most significant roles in the final effectiveness of the whole solution: 1) the number of neighbours retrieved (k) from the synthetic set for every query and 2) the length of the synthetic set itself. This section discusses how changes in these values affect the quality of the generated explanations in a less than optimal way (as seen in Fig. 7).

4.4.1 Number of Neighbours

The value k determines how many synthetic pairs are considered with respect to a query. Overall, we observed that smaller values lead to more sensitive and jagged results. Up to a certain point (e.g., 15), increasing k typically enables to obtain *smoother* explanations, due to the larger number of samples taken into account when averaging the closest neighbours. This trend, however, starts returning incremental improvements (notice in Fig. 7, where $k \geq 50$ progressively stops presenting a prominent tone on the eyelid).

4.4.2 Length of the Synthetic Dataset

This is the most sensitive parameter of our solution. Considering that it is important to find "*genuine*" pairs that closely resemble a query, it is particularly sensitive to assure that all typical periocular data variations are faithfully represented in the synthetic set, assuring that the retrieved elements (i.e., the most similar) will have its major components (iris, eyebrows and eyelids) aligned to the query itself. If this condition is not satisfied, the explanations loose their biological plausibility and effectiveness. Fig. 7 illustrates how smaller synthetic sets lead to less evident explanations, especially around the eyelid and the eyebrow.

5. Conclusions and Further Work

This paper described an integrated framework, based in well known deep-learning architectures, to simultaneously perform periocular recognition and - most importantly - to provide visual explanations of the regions/features that sustained every *non-match* decision, which we consider to be the cases where explanations are the most required. According to the powerful generative ability of GANs, we create a very large set of synthetic pairs that follow the "*genuine* distribution". At inference time, for every "*impostor*" comparison we are able to perceive the regions and features that *failed the most* (i.e., those that most evidently were different from a subset of the "*genuine*" synthetic pairs). This enables to generate pleasant explanations, where each component of the target region appears with a different colour depending on how it influenced the final decision. Importantly, the modular nature of our method ensures that the periocular region can be replaced by other biometric traits (e.g., the face) without compromising the explanations.

As future work, we are developing a strategy for also providing intuitive explanations of the "*genuine*" observations, where the strategy has to be very different from the idea behind the "*impostors*" insight used in this paper.

Acknowledgements

This work is funded by FCT/MEC through national funds and co-funded by FEDER - PT2020 partnership agreement under the project UIDB//50008/2020. Also, it was supported by operation Centro-01-0145-FEDER-000019 - C4 - Centro de Competências em Cloud Computing, co-funded by the European Regional Development Fund (ERDF) through the Programa Operacional Regional do Centro (Centro 2020), in the scope of the Sistema de Apoio a Investigacão Cientifica e Tecnologica - Programas Integrados de IC&DT.

References

[1] D. W. Apley and J. Zhu. Visualizing the effects of predictor variables in black box supervised learning models. arXiv:1612.08468 [stat], Ago. 2019. Accessed on: Feb. 27, 2021. [Online]. Available: http://arxiv.org/abs/1612.08468.

[2] S. Minaee, A. Abdolrashidi, H. Su, M. Bennamoun and D. Zhang. Biometrics recognition using deep learning: A survey. arXiv:1912.00271 [cs], Fev. 2021. Accessed on: Feb. 27, 2021. [Online]. Available: http://arxiv.org/abs/1912.00271.

[3] European Commission. General data protection regulation. 2018. Accessed on: Feb. 27, 2021. [Online]. Available: https://gdpr-info.eu.

[4] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[5] Z. Huang and Y. Li. Interpretable and accurate fine-grained recognition via region grouping. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8659–8669, 2020.

[6] L. Nie, A. Kumar and S. Zhan. Periocular recognition using unsupervised convolutional rbm feature learning. *2014 22nd International Conference on Pattern Recognition*, pages 399–404, 2014.

[7] T. Karras, S. Laine and T. Aila. A style-based generator architecture for generative adversarial networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4396–4405, 2019. Long Beach, CA, USA.

[8] Z. C. Lipton. The mythos of model interpretability. arXiv:1606.03490 [cs, stat], Mar. 2017. Accessed on: Feb. 27, 2021. [Online]. Available: http://arxiv.org/abs/1606.03490.

[9] S. Lundberg and S. Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 4768–4777, Red Hook, NY, USA, 2017. Curran Associates Inc. Long Beach, California, USA.

[10] S. Wachter, B. Mittelstadt and L. Floridi. Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law*, 7(2):76–99, 2017.

[11] C. Molnar. Interpretable machine learning. A guide for making black box models explainable. 2019. Accessed on: Feb. 27, 2021. [Online]. Available: https://christophm.github.io/interpretable-ml-book/.

[12] H. Proença, S. Filipe, R. Santos, J. Oliveira and L. A. Alexandre. The ubiris.v2: A database of visible wavelength iris images captured on- the-move and at-a-distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1529–1535, 2010.

[13] C. Padole and H. Proença. Periocular recognition: Analysis of performance degradation factors. *Proceedings of the Fifth IAPR/IEEE International Conference on Biometrics – ICB 2012*, 2012. New Delhi, India.

[14] A. Ross, R. Jillela, J. M. Smereka, V. N. Boddeti, B. V. K. V. Kumar, R. Barnard, X. Hu, P. Pauca and R. Plemmons. Matching highly non-ideal ocular images: An information fusion approach. *2012 5th IAPR International Conference on Biometrics (ICB)*, pages 446–453, 2012.

[15] H. Proença and J. C. Neves. Deep-prwis: Periocular recognition without the iris and sclera using deep learning frameworks. *IEEE Transactions on Information Forensics and Security*, 13(4):888–896, 2018.

[16] A. K. Jain, A. Ross and S. Prabhakar. An introduction to biometric recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1):4–20, 2004.

[17] U. Park, A. Ross and A. K. Jain. Periocular biometrics in the visible spectrum: A feasibility study. *2009 IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems*, pages 1–6, 2009.

[18] S. Umer, A. Sardar, B. C. Dhara, R. K. Rout and H. M. Pandey. Person identification using fusion of iris and periocular deep features. *Neural Networks*, 122:407–419, 2020.

[19] B. Yin, L. Tran, H. Li, X. Shen and X. Liu. Towards interpretable face recognition. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9347–9356, 2019.

[20] M. T. Ribeiro, S. Singh and C. Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[21] A. Preece, D. Harborne, D. Braines, R. Tomsett and S. Chakraborty. Stakeholders in explainable ai. arXiv:1810.00184 [cs], Set. 2018. Accessed on: Feb. 27, 2021. [Online]. Available: http://arxiv.org/abs/1810.00184.

[22] K. Simonyan, A. Vedaldi and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv:1312.6034 [cs], Abr. 2014. Accessed on: Feb. 27, 2021. [Online]. Available: http://arxiv.org/abs/1312.6034.

[23] Z. Zhao and A. Kumar. Accurate periocular recognition under less constrained environment using semantics-assisted convolutional neural network. *IEEE Transactions on Information Forensics and Security*, 12(5):1017–1030, 2017.

[24] Z. Zhao and A. Kumar. Improving periocular recognition by explicit attention to critical regions in deep neural network. *IEEE Transactions on Information Forensics and Security*, 13(12):2937–2952, 2018.

Bibliography

[Apl16] D. Apley. Visualizing the effects of predictor variables in black box supervised learning models. *arXiv: Methodology*, 2016. 20

[BC19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 6

[BH98] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner. Gradient-based learning applied to document recognition. 1998. 6

[BZ19] S. Minaee, A. Abdolrashidi, H. Su, M. Bennamoun and D. Zhang. Biometric recognition using deep learning: A survey. *ArXiv*, abs/1912.00271, 2019. 4

[CB14] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio. Generative adversarial networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2672–2680, Cambridge, MA, USA, 2014. MIT Press. 6

[CB17] M. Arjovsky, S. Chintala and L. Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017. 14

[Com18] European Commission. General data protection regulation. 2018. Accessed on: Feb. 27, 2021. [Online]. Available: https://gdpr-info.eu. v

[DG17] K. He, G. Gkioxari, P. Dollár and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. ix, 12

[DM14] R. Girshick, J. Donahue, T. Darrell and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014. ix, 9, 10

[Fri01] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001. 19

[Fuk04] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 2004. 6

[Gir15] R. Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. ix, 10

[GL17] Y. Guo, Y. Liu, T. Georgiou and M. S. Lew. A review of semantic segmentation using deep neural networks. *International Journal of Multimedia Information Retrieval*, 7:87–93, 2017. 6

[GS13] J. R. R. Uijlings, K. E. A. Sande, T. Gevers and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104:154–171, 09 2013. ix, 9

[GS15] S. Ren, K. He, R. Girshick and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. ix, 11

[Gue16] S. Singh, M. T. Ribeiro, C. Guestrin. Programs as black-box explanations. 11 2016. v

[HL20] Z. Huang and Y. Li. Interpretable and accurate fine-grained recognition via region grouping. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8659–8669, 2020. 5, 44, 49

[KZ14] L. Nie, A. Kumar and S. Zhan. Periocular recognition using unsupervised convolutional rbm feature learning. *2014 22nd International Conference on Pattern Recognition*, pages 399–404, 2014. 4

[LA19] T. Karras, S. Laine and T. Aila. A style-based generator architecture for generative adversarial networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4396–4405, 2019. Long Beach, CA, USA. x, 15, 41

[LA20] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen and T. Aila. Analyzing and improving the image quality of stylegan. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8107–8116, 2020. x, 14, 15, 16

[Lip18] Z. C. Lipton. The mythos of model interpretability. *Queue*, 16(3):31–57, June 2018. 18

[LL17] S. Lundberg and S. Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 4768–4777, Red Hook, NY, USA, 2017. Curran Associates Inc. Long Beach, California, USA. xi, 28, 29, 30, 49

[LL18] T. Karras, T. Aila, S. Laine and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018. 14

[Mat20] MathWorks. Train generative adversarial network (gan), 2020. [Online]. Available from: https://www.mathworks.com/help/deeplearning/ug/train-generative-adversarial-network.html. ix, 13

[MF17] S. Wachter, B. Mittelstadt and L. Floridi. Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law*, 7(2):76–99, 2017. v

[Mol19] C. Molnar. *Interpretable Machine Learning*. 2019. [Online]. Available from: https://christophm.github.io/interpretable-ml-book/. x, 18, 20, 22, 25, 29, 30

[MW17] G. Huang, Z. Liu, L. Maaten and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017. ix, 8

[OA10] H. Proença, S. Filipe, R. Santos, J. Oliveira and L. A. Alexandre. The ubiris.v2: A database of visible wavelength iris images captured on- the-move and at-a-distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1529–1535, 2010. 47

[Ola15] Christopher Olah. Understanding lstm networks. [Online]. Available from: https://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015. x, 16, 17

[PN18] H. Proença and J. C. Neves. Deep-prwis: Periocular recognition without the iris and sclera using deep learning frameworks. *IEEE Transactions on Information Forensics and Security*, 13(4):888–896, 2018. 4

[PP12a] C. Padole and H. Proença. Periocular recognition: Analysis of performance degradation factors. *Proceedings of the Fifth IAPR/IEEE International Conference on Biometrics – ICB 2012*, 2012. New Delhi, India. 41

[PP12b] A. Ross, R. Jillela, J. M. Smereka, V. N. Boddeti, B. V. K. V. Kumar, R. Barnard, X. Hu, P. Pauca and R. Plemmons. Matching highly non-ideal ocular images: An information fusion approach. *2012 5th IAPR International Conference on Biometrics (ICB)*, pages 446–453, 2012. 4

[Pra18] Prabhu. Understanding of cnn. [Online]. Available from: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148, 2018. ix, 7

[RJ09] U. Park, A. Ross and A. K. Jain. Periocular biometrics in the visible spectrum: A feasibility study. *2009 IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems*, pages 1–6, 2009. 4

[Ros57] F. Rosenblatt. *The perceptron - a perceiving and recognizing automaton*. Report: Cornell Aeronautical Laboratory. Cornell Aeronautical Laboratory, 1957. 5

[RP04] A. K. Jain, A. Ross and S. Prabhakar. An introduction to biometric recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1):4–20, 2004. 3, 41

[RP20] S. Umer, A. Sardar, B. C. Dhara, R. K. Rout and H. M. Pandey. Person identification using fusion of iris and periocular deep features. *Neural Networks*, 122:407–419, 2020. 4

[RS16] K. He, X. Zhang, S. Ren and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 8

[SC19] Z. He, W. Zuo, M. Kan, S. Shan and X. Chen. Attgan: Facial attribute editing by only changing what you want. *IEEE Transactions on Image Processing*, 28(11):5464–5478, 2019. 14

[SG16] M. T. Ribeiro, S. Singh and C. Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016. x, 25, 49

[SG18] M. T. Ribeiro, S. Singh and C. Guestrin. Anchors: High-precision model-agnostic explanations, 2018. xi, 26, 28

[SH12] A. Krizhevsky, I. Sutskever and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105. Curran Associates, Inc., 2012. 6

[She18] S. Sheldrick. What does AI know: Model interpretability and occlusion. [Online]. Available from: https://www.linkedin.com/pulse/what-does-ai-know-model-interpretability-occlusion-susan-sheldrick/, 2018. x, 23

[SL19] B. Yin, L. Tran, H. Li, X. Shen and X. Liu. Towards interpretable face recognition. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9347–9356, 2019. 5

[SW16] G. Huang, Y. Sun, Z. Liu, D. Sedra and K. Weinberger. Deep networks with stochastic depth. In *Computer Vision – ECCV 2016*, pages 646–661, Cham, 2016. Springer International Publishing. 8

[SZ15] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 8

[VR15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. 8

[VZ14] K. Simonyan, A. Vedaldi and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2014. x, 23, 24, 49

[YZ15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu and X. Zheng. Tensorflow: Large-scale machine learning on heterogeneous systems. [Online]. Available from: https://www.tensorflow.org/, 2015. 6

[ZF14] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer Vision – ECCV 2014*, pages 818–833, Cham, 2014. Springer International Publishing. x, 23

[ZK17] Z. Zhao and A. Kumar. Accurate periocular recognition under less constrained environment using semantics-assisted convolutional neural network. *IEEE Transactions on Information Forensics and Security*, 12(5):1017–1030, 2017. xii, 4, 35, 47, 48

[ZK18] Z. Zhao and A. Kumar. Improving periocular recognition by explicit attention to critical regions in deep neural network. *IEEE Transactions on Information Forensics and Security*, 13(12):2937–2952, 2018. 5, 47