UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
Coordenação de Pós-Graduação em Ciência da Computação

Arnaldo Gualberto de Andrade e Silva

Deep Multitask Learning for Automatic Evaluation of ICAO Requirements

Campina Grande, Paraíba, Brazil
Julho, 2023

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
Coordenação de Pós-Graduação em Ciência da Computação

Deep Multitask Learning for Automatic Evaluation of ICAO Requirements

Arnaldo Gualberto de Andrade e Silva

Tese submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Doutor em Ciência da Computação.

Área de concentração: Ciência da Computação
Linha de Pesquisa: Modelos Computacionais e Cognitivos

Prof. Dr. Herman Martins Gomes
(Orientador)

Prof. Dr. Leonardo Vidal Batista
(Co-orientador)

Campina Grande, Paraíba, Brazil
Julho, 2023

S586d Silva, Arnaldo Gualberto de Andrade.
Deep multitask learning for automatic evaluation of ICAO requirements / Arnaldo Gualberto de Andrade e Silva. – Campina Grande, 2023.
113 f. : il. color.

Tese (Doutorado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2023.
"Orientação: Prof. Dr. Herman Martins Gomes".
Referências.

1. Inteligência Artificial. 2. Modelos de Processos Cognitivos. 3. Representação do Conhecimento. 4. Sistemas de Aplicação da Inteligência Artificial. 5. Ciência da Computação. 6. Modelos Computacionais e Cognitivos. I. Gomes, Herman Martins. II. Título.

CDU 004.8(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA SEVERINA SUELI DA SILVA OLIVEIRA CRB-15/225

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
POS-GRADUACAO EM CIENCIA DA COMPUTACAO
Rua Aprígio Veloso, 882, Edifício Telmo Silva de Araújo, Bloco CG1, - Bairro Universitário, Campina Grande/PB, CEP 58429-900
Telefone: 2101-1122 - (83) 2101-1123 - (83) 2101-1124
Site: http://computacao.ufcg.edu.br - E-mail: secretaria-copin@computacao.ufcg.edu.br / copin@copin.ufcg.edu.br

FOLHA DE ASSINATURA PARA TESES E DISSERTAÇÕES

ARNALDO GUALBERTO DE ANDRADE E SILVA

DEEP MULTITASK LEARNING FOR AUTOMATIC EVALUATION OF ICAO REQUIREMENTS

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação como pré-requisito para obtenção do título de Doutor em Ciência da Computação.

Aprovada em: 20/07/2023

Prof. Dr. HERMAN MARTINS GOMES, UFCG, Orientador
Prof. Dr. LEONARDO VIDAL BATISTA, UFPB, Orientador
Prof. Dr. EANES TORRES PEREIRA, UFCG, Examinador Interno
Profa. Dra. JOSEANA MACÊDO FECHINE RÉGIS DE ARAÚJO, UFCG, Examinadora Interna
Prof. Dr. ANDERSON DA SILVA SOARES, UFG, Examinador Externo
Prof. Dr. ALUIZIO FAUSTO RIBEIRO ARAÚJO, UFPE, Examinador Externo

Documento assinado eletronicamente por **HERMAN MARTINS GOMES**, **PROFESSOR 3 GRAU**, em 21/07/2023, às 09:30, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Leonardo Vidal Batista**, **Usuário Externo**, em 21/07/2023, às 09:46, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **EANES TORRES PEREIRA**, **PROFESSOR 3 GRAU**, em 21/07/2023, às 10:17, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **JOSEANA MACEDO FECHINE**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 24/07/2023, às 09:00, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Anderson da Silva Soares**, **Usuário Externo**, em 25/07/2023, às 10:57, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

A autenticidade deste documento pode ser conferida no site https://sei.ufcg.edu.br/autenticidade, informando o código verificador **3581355** e o código CRC **34A6BCF8**.

O que você faria se não tivesse medo?

DEDICATÓRIA

A todos que duvidam de mim.

AGRADECIMENTOS

A meus orientadores. A *Herman Martins* por ter me aceitado no doutorado após vários anos tentando e sendo recusado por outros orientadores. Também por contribuir significantemente nas nossas reuniões sempre com suas ideias, questionamentos e incentivo. A *Leonardo Vidal* por me acompanhar desde o 2º período da graduação, passando pelo PET, 2 PIBICs, Mestrado e ter-me feito apaixonar pela área da Computação que mais amo até hoje: Visão Computacional. Considero-o como o meu pai acadêmico.

A minha família. A meu pai, *Arnaldo Gualberto da Silva*, por nunca medir esforços para me dar todas as oportunidades que tive na vida. Também por me ensinar sobre responsabilidade e me mostrar que nada resiste ao trabalho. À minha mãe, *Luzia de Andrade*, por construir os pontos mais importantes da minha personalidade e caráter. E por nunca ter me deixado desistir de nada na vida. A meu irmão, *Juan Gualberto*, pelo tudo que vivemos como irmãos e pelo orgulho que tem de mim - sentimento recíproco. Ao contrário da frase popular, costumo dizer que somos mais do que irmãos, somos amigos.

A Sabrina Figueiredo. Por me apoiar em todas as decisões e me permitir levá-las adiante. Se arrisquei, errei, ou acertei, é porque eu sabia que em todas as ocasiões você estaria lá. Sem você, eu não teria feito outro doutorado e essa tese não teria sido escrita. Você é meu porto seguro, GPS, e combustível. Também é minha força, motivação e inspiração. Eu me encontro em você pra ser o melhor que sou.

A Theo Gualberto. Por mudar a maneira como eu via o mundo, minhas prioridades e me fazer uma pessoa melhor em alguns aspectos. Também por ser minha Rede Neural Biológica. Espero que em algum momento você entenda, filho, que tudo que fiz (mesmo antes de você nascer) e ainda farei de importante na vida é para que, um dia, você tenha orgulho do seu pai.

RESUMO

A face é considerada o principal traço biométrico para documentos de viagem legíveis por máquina, como passaportes. Neste contexto, o padrão ISO/IEC 19794-5 define um conjunto de requisitos fotográficos para garantir a qualidade da imagem e simplificar o processo de reconhecimento facial. No entanto, devido ao grande número de requisitos definidos por esse padrão (quase 30), a verificação de conformidade de uma única imagem facial ainda é um desafio. Normalmente, problemas com várias tarefas, como os requisitos desse padrão, são divididos em subproblemas independentes que são resolvidos separadamente e, em seguida, recombinados. No entanto, isso ignora as informações comuns entre tarefas relacionadas e aumenta o risco de sobreajuste. O Aprendizado Multitarefa (do inglês, Multitasking Learning, MTL) tem se provado uma técnica importante para resolver várias tarefas relacionadas simultaneamente. Ele explora os aspectos comuns e distintos de tarefas do mesmo domínio para melhorar a generalização entre todas as tarefas. Além disso, o MTL concentra-se em aprender uma representação útil que possa gerar benefícios, especialmente em cenários em que um conjunto de dados rotulados para uma tarefa é limitado. Por fim, no caso das Redes Neurais Profundas, o MTL pode ajudar a reduzir o número de parâmetros e a velocidade de inferência. Esta pesquisa propõe o primeiro método de aprendizado profundo multitarefa projetado para avaliação automática dos requisitos do padrão ISO/IEC 19794-5, denominado ICAONet. Autoencoders subcompletos são estendidos para empregar uma abordagem de multi-aprendizagem colaborativa, onde a aprendizagem supervisionada e não-supervisionada são realizadas simultaneamente e de forma cooperativa. O método é treinado usando um banco de imagens especialmente construído para o problema descrito e avaliado por um sistema de *benchmark* oficial também utilizado por outras abordagens presentes na literatura. Os experimentos mostram que o método proposto alcança os melhores resultados em termos de Taxa de Erro Igual (do inglês, Equal Error Rate, EER) para 9 dos 23 requisitos fotográficos, o que não foi alcançado por nenhum outro método conforme a bibliografia consultada. Portanto, o método proposto pode ser considerado a melhor solução geral entre trabalhos acadêmicos publicados na literatura e SDKs privados analisados. No geral, a EER mediana (3,3%) também é competitiva. Em termos de tempo de execução, o método proposto se destaca entre os métodos mais rápidos para avaliar todos os 23 requisitos segundo o benchmark oficial. Por outro lado, há espaço para melhorias nos resultados da localização dos olhos e alguns requisitos específicos, que podem exigir investigação adicional. Por fim, por meio de técnicas de visualização de Redes Neurais, foram identificados padrões de representação relevantes aos requisitos do padrão ISO/IEC 19794-5.

Palavras-chave: Qualidade Facial, ICAO, ISO/IEC 19794-5, Aprendizado Multitarefa, Autoencoders, Aprendizado Profundo.

ABSTRACT

The face is considered the primary biometric trait for machine-readable travel documents such as passports. In this context, the ISO/IEC 19794-5 standard defines a set of photographic requirements to ensure the image quality and simplify the face recognition process. However, because the number of requirements defined by such a standard is high (almost 30), the compliance verification of a single face image is still a challenge. Usually, problems with multiple tasks, such as the ISO/IEC 19794-5 requirements, are broken into independent subproblems that are solved separately and then recombined. Nevertheless, it ignores the common information between related tasks and increases the risk of overfitting. Multitasking Learning (MTL) has proven to be an important technique for solving multiple related tasks simultaneously. It explores the common and distinct aspects of tasks from the same domain to improve the generalization among all tasks. In addition, MTL focuses on learning a useful representation that can yield benefits, particularly in scenarios where a labeled dataset for a task is limited. Finally, in the case of Deep Neural Networks, MTL can help reduce the number of parameters and inference speed. This research proposes the first deep Multitasking Learning method designed for automatic evaluation of both photographic and geometric requirements of the ISO/IEC 19794-5 standard, called ICAONet. Undercomplete Autoencoders are extended to employ a multi-and-collaborative learning approach, in which both supervised and unsupervised learning are performed concurrently and in a collaborative manner. The method is trained using an ad hoc image dataset and evaluated using an official benchmark system that is also used by other approaches presented in the literature. The experiments show the method proposed achieves the best results in terms of Equal Error Rate for 9 out of the 23 photographic requirements of ISO/IEC 19794-5, which was not achieved by any other individual method according to the consulted bibliography. Therefore, the proposed method can be considered the best overall solution among evaluated academic works published in the literature and private SDKs. Overall, the median Equal Error Rate (3.3%) is also competitive. In terms of running time, the proposed method stands out among the fastest methods to evaluate all 23 requirements according to the official benchmark. On the other hand, there are space for improvements on results of eye's landmark location and some specific requirements that may require additional investigation. Finally, through Neural Network visualization techniques, relevant patterns related to the requirements of the ISO/IEC 19794-5 standard could be observed.

Keywords: Face Quality, ICAO, ISO/IEC 19794-5, Multitasking Learning, Autoencoders, Deep Learning

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI Artificial Intelligence

CNN Convolutional Neural Network

DL Deep Learning

DNN Deep Neural Network

EER Equal Error Rate

FAR False Acceptance Rate

FICV Face Image ISO Compliance Verification

FMR False Match Rate

FNMR False Non-Match Rate

FN False Negative

FP False Positive

FRR False Rejection Rate

FVC Fingerprint Verification Competition

FICV Face Image ISO Compliance Verification

GAN Generative Adversarial Network

ICAO International Civil Aviation Organization

IEC International Electrotechnical Commission

ISO International Organization for Standardization

LDA Linear Discriminant Analysis

LIME Local Interpretable Model-agnostic Explanations

MAE Mean Absolute Error

MCC Matthews Correlation Coefficient

ML Machine Learning

MRTD Machine-Readable Travel Document

MSE Mean Squared Error

MTL Multitasking Learning

NLP Natural Language Processing

NPV Negative Predictive Value

PCA Principal Component Analysis

PPV Positive Predictive Value

ROC Receiver Operating Characteristic

SHAP SHapley Additive exPlanations

TN True Negative

TP True Positive

t-SNE t-Distributed Stochastic Neighbour Embedding

Contents

1 Introduction

The face has been traditionally used in identity documents for visual recognition of a person and therefore represents one of the most used physical traits for biometric recognition (Ferrara et al., 2012b). It also has an essential role in many other biometric-related applications such as video surveillance (la Torre et al., 2015) or facial expression recognition (Anil and Suresh, 2016). Compared to other physical traits, the face has some advantages. For instance, acquiring facial features via digital photography is non-intrusive, can be performed remotely, and does not require full cooperation from individuals or specialized hardware.

In the context of biometric recognition via identity documents, many efforts have been made over the years to allow machine-assisted verification of an individual's identity. One of the most important projects was developed by the International Civil Aviation Organization (ICAO). In 2002, the ICAO defined directives for automatic biometric recognition of people using machines (ICAO, 2003). The goal was to specify the ideal conditions of facial images to perform robust and highly accurate face verification/recognition by machines. These regulations are followed by many countries worldwide, including, for instance, all member states of the European Union (Ebinger et al., 2008).

The ISO/IEC 19794-5 (ISO, 2007) is an official standard that defines the requirements for facial photography used in electronic passports based on the ICAO guidelines. It describes a set of quality rules that include photographic properties (positioning, camera focus, etc.), scene constraints (lighting, pose, and expression), and digital attributes (image resolution, image size, etc.). For example, a given facial image suitable for passports must have a frontal pose with a neutral expression, open eyes, no objects covering the face (e.g., hair or veil), a uniform background, illumination, and focus.

Since the number of requirements defined by the ISO/ICAO standards is high (almost 30), the compliance verification of a single face image is still challenging. According to Ferrara et al. (2012a), this task used to be visually performed by human experts, sometimes with the support of an automated system. It prevents agility in critical scenarios, such as international airports, where this task is performed millions of times daily. Therefore, the complete automation of this task is still an ongoing request and may help avoid the need for a human expert and accelerate the document production process. In this context, the following research question arises:

> *Would it be possible to conceive a single Neural Network architecture that could efficiently (in terms of memory and running time) and precisely (with low error rates) assess all the requirements of ISO/IEC 19794-5 standard?*

Pattern recognition is the process of automatic discovery of regularities in data

through the use of computer algorithms (Bishop and Nasrabadi, 2006). The goal is to extract novel, useful, and hidden patterns from input data. These patterns can then be used to take action, such as classifying the data into different categories. According to Orriols-Puig et al. (2008), a competent algorithm must be able to: (i) identify hidden patterns between a set of features and the corresponding output; (ii) represent these patterns in a coherent structure; and (iii) generalize well to produce a compact representation.

In recent years, Deep Neural Networks (DNNs) have gained prominence in Computer Science due to their high propensity to recognize intricate patterns. One of the main advantages of this technique is the network's inherent ability to extract information from raw data, sometimes with little or no preprocessing. It allows a generic learning process that does not depend on attributes explicitly chosen or extracted from the data. In Computer Vision, this task is usually performed by Convolutional Neural Networks (CNNs), a particular type of architecture initially developed for images as input, reducing the number of parameters and improving the training process (Goodfellow et al., 2016).

In Machine Learning (ML), a model is commonly optimized regarding a single metric, typically a score on a specific benchmark. To achieve this, single models or an ensemble of models are often trained for the desired task and fine-tuned until performance plateaus. Although this approach can yield acceptable results in a single task, it neglects information that might potentially improve performance on the primary metric of interest. In particular, such information originates from the training signals of related tasks. The shared representation between related tasks can enable a model to generalize better for the original task (Ruder, 2017). This approach is referred to as Multitasking Learning (MTL) (Caruana, 1997).

Multitasking Learning is a specific Machine Learning technique that allows multiple tasks to be solved at the same time by exploring familiar and different aspects between them (Zhang and Yang, 2021). It expands the concept of feature sharing (Torralba et al., 2004) and goes against the traditional methodology in which one task is learned at a time. Usually, significant problems - such as the ICAO requirements - are broken into small, independent, and reasonably sized subproblems that are solved separately and then recombined. As stated in Caruana (1997), this approach is sometimes counterproductive, since a potentially rich source of information available in many real-world problems is ignored: the information contained in other tasks from the same domain. In the context of Deep Learning, the MTL can allow the network to share information on related tasks to improve generalization in all tasks. Moreover, the Multitasking Learning is best suited when there are limited training samples in multiple related tasks.

Compared to the case where each task of a multitasking problem is solved individually by a specific network, the multitask networks present several advantages. First, the number of parameters and memory used by the model is considerably reduced ow-

ing to the inherent layer sharing. Second, the inference speed increases because such networks avoid recomputing features in the shared layers. Finally, such networks can improve performance if related tasks share information or act as regularizers for one another (Vandenhende et al., 2021). In the context of the ISO/IEC 19794-5 standard, MTL has the potential to solve all requirements in parallel, reducing the processing time and increasing success rates.

In this thesis, we present a Deep Learning-based method for the automatic evaluation of the requirements of the ISO/IEC 19794-5 standard, called ICAONet. Our network was trained from scratch in a Multitask Learning approach with limited data (approximately 5700 images only). The architecture is based on Autoencoders, but the training is performed simultaneously in unsupervised (image reconstruction) and supervised (multi-label classification and landmark localization) manners. An experimental evaluation showed that our method outperformed the algorithms presented in the literature and SDK tools available as commercial solutions on most requirements. We also applied SHapley Additive exPlanations (SHAP) (Lundberg and Lee, 2017), Principal Component Analysis (PCA) (Wold et al., 1987), and t-SNE (van der Maaten and Hinton, 2008) to understand the representations and predictions related to each requirement presented in the ISO/IEC 19794-5 standard.

To summarize, the main contributions of this thesis are as follows:

1. To the best of our knowledge, this is the first work that employs a Multitasking Learning approach for assessment of the ISO/IEC 19794-5 requirements.

2. According to the consulted bibliography, we present the first Deep Learning-based work that evaluates all photographic requirements in a single method. All other published studies apply specific methods for each requirement assessment separately or evaluate only a subset of requirements.

3. Our method achieved state-of-the-art results in 9 out of the 23 photographic requirements of the ISO/IEC 19794-5 standard, being the method with the majority of best results.

4. We extended undercomplete Autoencoders to employ a multi-and-collaborative learning approach, where supervised and unsupervised learning are performed concurrently and collaboratively.

1.1 Objectives

The general objective of this thesis is to develop a state-of-the-art method for assessment of the requirements from ISO/IEC 19794-5 standard using Multitasking Learning. By state-of-the-art, we mean an algorithm that can assess all 23 requirements in a

single method with the best median EER among all the published works. The specific goals are:

- Build an ad-hoc labeled database for evaluation of ICAO requirements;
- Propose a method to perform preprocessing of face images such that the network can have accurate results without compromising run time speed;
- Investigate and develop a method for localization of eyes centers;
- Research and develop a single method based on Deep Learning for assessment of all photographic and pose-specific tests of ISO/IEC 19794-5 standard using Multitasking Learning (MTL) and compare it against all methods published in the literature; and
- Analyse the representations learned by the proposed method and its outputs using dimensionality reduction and explanation techniques.

1.2 Thesis Structure

The rest of this thesis is structured as follows. First, we detail the fundamental concepts used in this work in Chapter 2. In Chapter 3, a review of the principal Multitasking Learning architectures and works published for the ISO/IEC 19794-5 standard is presented. Next, in Chapter 4, we describe the proposed methodology. The results of the current thesis are presented in Chapter 5. Finally, the conclusions and possible future research are discussed.

2 Background

In this chapter, we explain the essential concepts used in this thesis. We start by introducing Deep Learning and its main aspects. Then, we present Convolutional Neural Networks and Autoencoders, two types of Deep Architectures used by the proposed method. In addition, an explanation of Multitasking Learning and its variants is presented. Later, we detail methods to explain Deep Networks, such as SHAP, and for dimensionality reduction, like Principal Component Analysis and t-SNE. Next, the metrics used to evaluate the proposed method are described in detail. Finally, the ISO/IEC 19794-5 standard is described, along with the FVC-onGoing competition and its protocol.

2.1 Deep Learning

Artificial Intelligence (AI) is a study field of Computer Science focused on the design and construction of intelligent agent that receives inputs from the environment and take actions that affect that environemnt (Russell and Norvig, 2020). It can include, for example, learning, reasoning, and self-correction. In the early days of AI, the first systems were heavily based on a set of rules previously provided by experts from certain subject areas. Usually, large rule-based problems are intellectually more difficult for humans than for computers; thus, machines can take advantage of them. Also, such systems have been developed for small and restricted environments. For example, we can cite Deep Blue (Hsu, 2002), a successful chess-playing system developed by IBM that defeated the world champion Garry Kasparov in 1997.

As the scale and amount of data have increased over the years, traditional AI methods have been replaced by a more data-driven approach called Machine Learning (ML). Instead of manually providing rules for the system, an algorithm automatically learns intrinsic patterns based on data. When the desired answers are also provided as inputs, we call it *Supervised Learning*; otherwise, it is called *Unsupervised Learning*. Many algorithms have been developed for both types of learning, for instance: Support Vector Machines (Boser et al., 1992), Decision Trees (Breiman et al., 1984), and Mean Shift (Fukunaga and Hostetler, 1975).

Although Machine Learning represents an important advance in the AI field, traditional algorithms are limited when processing raw data in their natural form (e.g., image pixels, text, or audio data). Typically, domain expertise is employed to carefully define how to extract useful features from raw data that can be used as inputs to the ML algorithm. This process is usually referred to as feature engineering. However, it may be challenging for many tasks (for instance, detecting people in images) to define "which" and "how" features should be extracted.

Representation learning allows ML-based systems to discover valuable representations from raw data automatically. According to Goodfellow et al. (2016), these learned representations often yield better performance in comparison to the hand-designed ones. Furthermore, representation learning algorithms enable AI systems to adapt faster to new tasks with minimal human intervention.

Deep Learning (DL) represents a particular sub-field of representation learning which allows computational models composed of multiple processing layers capable of learning data representations with many levels of abstraction (LeCun et al., 2015). Therefore, complex representations can be obtained from a composition of simpler ones. For example, we can cite the problem of detecting faces in images. Mathematically defining a function that maps a set of pixels (raw format) into the desired output (face location) can be challenging. However, Deep Learning can solve this problem by breaking the complex mapping into a series of simple nested mappings, each described by an individual model layer. For instance, the first layer may be responsible for detecting edges on raw pixels. Given these edge descriptions, the second layer may look for corners and contours since they are defined by a set of edges. Consecutively, the third layer can detect entire object parts (e.g., eyes, nose, and mouth) by identifying patterns in specific corners and contours. Finally, the parts of the object contained in the image can be evaluated to determine the presence (or absence) of the face in the image.

The most well-known examples of Deep Learning models are **feedforward neural networks**, also called deep feedforward networks or multilayer perceptrons. They were inspired by the biological model of neurons, their connections, and how the human brain processes information. The quintessential unit of a feedforward neural network is a node (neuron) that receives the inputs of other nodes and computes an output. Each input is associated with a learnable parameter w (synapse), also called **weight**, which assigns relative importance to the corresponding input. The weighted sum of the weights and inputs plus a **bias** compose the output of a neuron that is generally modified by an **activation function** responsible for introducing nonlinearity to the network. A stack of nodes at the same level forms a layer, and sequences of layers comprise the entire neural network. Each layer between the input and the output layer is called **hidden layer**. In a feedforward network, there are no feedback connections between the layers.

The set of weights and biases represents the trainable parameters of neural networks. Nevertheless, as with other ML algorithms, neural networks have other types of parameters called **hyperparameters**. They are used to control the learning process and are commonly divided into two groups:

- **Model hyperparameters**: define aspects mainly related to the neural network architecture. For instance, we can cite the number of layers, the number of neurons

in each layer, the activation function of each layer, momentum, dropout rate, and others.

- **Training hyperparameters**: related to the learning process itself. For example, the cost function, number of epochs, optimizer, batch size, etc.

It is crucial to note that no optimal set of hyperparams can work for all kinds of particular problems, and their tuning is commonly performed manually. Also, while some hyperparams influence the time and memory cost to perform a prediction with the trained network, others affect the quality of the model and its capacity to output correct results when the network is presented to new inputs.

The rest of this section focuses on some specific types of neural networks and layers adopted in this thesis. Moreover, we describe a special kind of learning employed when networks may learn multiple tasks simultaneously, called Multitasking Learning.

2.1.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs), also known as Convolutional Networks, are a specific kind of feedforward neural network specially developed to process grid-like data. For example, we can think of time-series data as a 1–D grid of time intervals with samples. Likewise, images can be considered as 2–D grid of pixels. In recent years, CNNs have achieved exceptional performance in several practical applications. It includes, but is not limited to, image classification (Li et al., 2014; Guo et al., 2017; Paoletti et al., 2018), object detection (Cai et al., 2016; Wu et al., 2017), and instance segmentation (Wang et al., 2020; Xu et al., 2020; Zhang et al., 2020).

The convolution is the basic operation of a CNN. In summary, it is a well-known mathematical operation that performs a linear calculation over two functions (Goodfellow et al., 2016). This is similar to the cross-correlation, but one of the functions is reversed and shifted. Given the functions x and w, the convolution operation over a period of measurement τ between x and w - denoted by $h(t)$ - can be defined according to Equation 1, for the continuous domain:

$$h(t) = \int_{-\infty}^{\infty} x(\tau)w(t-\tau)\ d\tau \tag{1}$$

Typically, the convolution operation is expressed by an asterisk operator (Equation 2):

$$h(t) = (x * w)(t) \tag{2}$$

where x represent the input, w is the kernel, and t is the point where the convolution is

computed. If w is a valid probability density function, the convolution can be considered a weighted average of the input at point t. Also, when working with discretized data (such as digital images), the Equation 1 can be rewritten as Equation 3:

$$h(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)w(t-\tau) \tag{3}$$

Usually, CNNs are primarily applied to process images as inputs. In these cases, the input x is represented by 4–D vectors, also called tensors, of $N \times H \times W \times C$ dimensions, where N denotes the number of images in the dataset (samples), H and W are the image dimensions, and C corresponds to the number of channels (e.g. colors) in each image. The kernel w is also a multidimensional array that represents the parameters to be learned by the algorithm.

As pointed out by Goodfellow et al. (2016), convolution has three essential characteristics that help improve the performance of ML-based systems:

- **sparse interactions**: unlike dense neural networks, the output units do not need to interact with each input unit. Instead, the kernel w is commonly smaller than the input. Therefore, fewer parameters must be stored, reducing the amount of memory and the number of operations the model requires, beyond improving its statistical efficiency.
- **parameter sharing**: refers to reusing the same kernel across the whole image. In traditional neural nets, each weight is strictly applied only once to compute the output of a layer. However, in CNNs, the weights presented in the kernel w are applied to every image pixel (except, in some cases, for the boundary pixels or when undersampling is employed - the stride hyperparameter). Therefore, convolutions are dramatically more effective than dense matrix multiplications.
- **equivariant representations**: refers to the fact that convolutions are invariant to translations. A function is called equivariant when the output changes correspond to changes in the input. In mathematical terms, this implies that a function $f(x)$ is equivariant to the function g if $f(g(x)) = g(f(x))$. In the case of convolutions, if g is a function that shifts the input image, then the convolution output is translated by the same amount. This property helps detect specific patterns across an image, such as edges or even more complex shapes like faces. However, convolutions are not equivariant to transformations like image scaling or rotation.

In Convolutional Neural Networks, beyond the convolutional layers, there are also other common types of layers employed to perform distinct operations. Some other layers used in this work are described in the following subsections.

2.1.1.1 Pooling Layers

Besides convolution, pooling layers are one of the essential components of Convolutional Neural Networks. Pooling is an operation that provides an approach to reduce the spatial size of feature maps, also called downsampling, and summarizes the features present in patches of the feature map. It helps to: (i) reduce the number of parameters; (ii) reduce the number of computations performed in the network; and (iii) make the model more robust to slight variations in the position of features in the input image, controlling overfitting. In general, pooling layers are applied after the activation function of the convolutional layers and operate independently on each feature map.

There are two common types of pooling methods: **max pooling** and **average pooling**. They compute the maximum and the average value for each patch of a feature map, respectively. An example of both operations can be seen in Figure 1.

(a) Max Pooling (b) Average Pooling

Figure 1: Example of Max Pooling and Average Pooling operations performed over a feature map of 4×4 with $pool\_size = 2 \times 2$, $stride = 2$, and no padding. Adapted from (Guissous, 2019).

The pooling layer has three input parameters: filter size (or pool size), stride, and whether to apply padding to the input image. Commonly, the filter size is 2×2, the stride is 2, and no padding is applied. Using this configuration, the feature map is reduced by half in each dimension, and 75% of the original activations are discarded. The number of channels in the feature map (depth) remains unchanged.

There is a particular type of pooling known as **global pooling**, introduced by Lin et al. (2013). In addition to the traditional method, the global version extends the pooling across the entire feature map. Therefore, if a given feature map has $H \times W \times C$ dimensions, it will be reduced to $1 \times 1 \times C$. Again, the most common functions used in global pooling are maximum and average. As stated by Zhou et al. (2016), such a difference makes the global pooling layers perform better in practice than the conventional approach. Moreover, it can substitute the flattened layers in the transition between convolutional layers and the fully connected network responsible for outputting a prediction.

Since pooling computes a fixed function of the input, no learning parameters are required by the pooling layers in Convolutional Networks. This is valid for all the pooling

approaches described above (max, average, and global pooling).

2.1.1.2 Batch Normalization Layers

Training Deep Neural Networks can be challenging due to many factors. For example, we can cite the random weights initialization, optimization algorithm, and chosen hyperparameters configuration. Furthermore, during the learning phase, each parameter is updated by the gradient under the assumption that the other layers do not change (Goodfellow et al., 2016). However, in practice, all weights are updated simultaneously.

Another significant factor is related to the expectations around layer distributions. During training, the distribution of each layer's input changes as previous layers' parameters also change. This is referred to as Internal Covariate Shift (Ioffe and Szegedy, 2015). It may cause unintended effects in the training of deep networks since small changes in shallower layers will be amplified during a forward pass to deeper hidden layers.

Batch Normalization is a regularization technique proposed by Ioffe and Szegedy (2015) to mitigate the effect of Internal Covariate Shift. It normalizes the inputs of hidden layers by using the first (mean) and second (variance) statistical moments of the current batch. This normalization step is usually applied before the activation function but can also be employed after the nonlinear function. The batch normalization makes the training of deep neural networks faster, more stable, and less likely to overfit. Also, batch normalization may substitute Dropout as a regularization technique.

Given a mini-batch B $\{x_i, \ldots, x_m\}$ of size m, the mean and variance of B are denoted by Equations 4 and 5, respectively:

$$\mu_B = \frac{1}{m} \sum_{i}^{m} x_i \tag{4}$$

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2 \tag{5}$$

Subsequently, the input x_i is normalized by Equation 6:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{6}$$

where x_i can be either the input or output of the activation function from the current layer and ϵ is an arbitrarily small constant added for numerical stability. The final output y_i of the current layers is given by Equation 7:

$$y_i = \gamma \hat{x}_i + \beta \tag{7}$$

where γ and β are new learnable parameters introduced by batch normalization.

One crucial aspect to highlight is that the network can automatically avoid batch normalization operations. That is, if the network during training converges to $\gamma = 1$ and $\beta = 0$, it means that the input x_i does not need to be normalized; thus, the input x_i remains unchanged.

During the training phase, the normalization steps are computed based on mini-batches to ensure reliable and efficient training. On the other hand, during inference, the network is generally asked to perform predictions for a given sample. For this reason, the normalization step is performed with the estimated population statistics. The population mean and variance are estimated during training and defined by Equations 8 and 9, respectively:

$$E[x] = E_B[\mu_B] \tag{8}$$

$$Var[x] = \frac{m}{m-1} E_B[\sigma_B^2] \tag{9}$$

Therefore, during inference, the input x is normalized by Equation 10:

$$\hat{x} = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} \tag{10}$$

And the output y will be given by Equation 11:

$$y = \frac{\gamma}{\sqrt{Var[x] + \epsilon}} \cdot \hat{x} + \left(\beta - \frac{\gamma E[x]}{\sqrt{Var[x] + e}} \right) \tag{11}$$

The implementation of batch normalization for Convolutional Neural Networks is slightly different compared to the fully connected networks. Since the output of CNN layers can have multiple channels, batch normalization is carried out for each channel. In other words, each channel has a single mean and standard deviation, as well as scale (γ) and shift (β) parameters. Again, these are scalar values learned during the optimization process. Similarly, the batch normalization procedure can be applied before or after the nonlinear activation function of the corresponding convolutional layer.

Besides reducing the internal covariate shift, batch normalization has other valuable advantages. Firstly, it can speed up training since we can use higher learning rates

without vanishing or exploding the gradients. Secondly, it can make the network more robust to different initialization schemes and learning rates. Finally, since batch normalization is a regularization technique, it helps the network improve in terms of generalization and diminishes overfitting. As pointed out earlier, batch normalization can replace other regularization methods like Dropout.

Although the effect of batch normalization is evident, the formal explanation of why it works remains an open question. For example, some authors have suggested that batch normalization does not reduce the internal covariance shift but actually smooths the objective function (Santurkar et al., 2018). Notwithstanding, batch normalization leads to harsh gradient explosion in deep networks at initialization, but such an effect is mitigated by skip connections as in residual networks (Yang et al., 2019). Other authors have suggested that the training of neural networks is faster with batch normalization due to the length-direction decoupling achieved by this technique (Kohler et al., 2019).

2.1.2 Autoencoders

Autoencoders are a specific type of neural network capable of discovering structures within data to create a compressed input representation. They are considered an unsupervised (or self-supervised) learning technique to leverage neural networks for representation learning. In the last years, Autoencoders have been successfully applied to many distinct tasks like dimensionality reduction (Petscharnig et al., 2017; Wang et al., 2015), information retrieval tasks (Pfeiffer et al., 2018), anomaly detection (Sakurada and Yairi, 2014), and image segmentation (Baur et al., 2018; Karimpouli and Tahmasebi, 2019).

The architecture of Autoencoders (Figure 2) is usually composed of two parts:

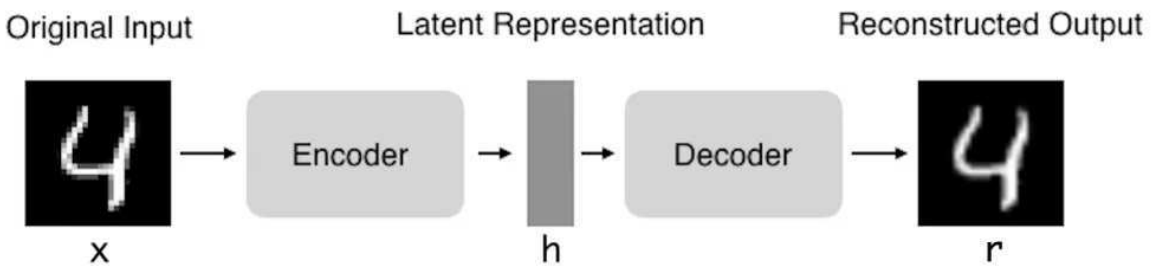


Figure 2: Architecture of an Autoencoder. Source: (Ryan, 2020)

1. **Encoder**: responsible for learning a useful representation of the input, also called as **embedding** or **latent representation** or even **code**. It can be formulated as an encoder function $h = f(x)$, where x is the input and h corresponds to the

codification of the input in the latent space ($h : \mathbb{R}^D \rightarrow \mathbb{R}^K$, where D and K represent the dimensionality of the input and latent space, respectively).

2. **Decoder**: maps the learned latent representation h back to the original input space ($\mathbb{R}^K \rightarrow \mathbb{R}^D$). It can be described as a reconstruction function $r = g(h)$, where r is the reconstructed input.

Since the exact copy of the input, $g(f(x)) = x$, is not especially useful, Autoencoders are intentionally designed to be unable to learn a perfect copy. Usually, Autoencoders are regularized in ways that allow them to learn an approximation of the identity function. By forcing the model to prioritize which input elements are relevant for reconstruction, it often learns valuable properties of the data.

A typical Autoencoder architecture restricts the embedding dimensions (h) to be smaller than input x. Such Autoencoder is called **undercomplete**. By constraining h, the Autoencoder is forced to learn an undercomplete representation and must prioritize the most salient features of the input data during training. Moreover, the reconstructed input r will not be a perfect copy of the input, and the undercomplete Autoencoders can be considered lossy compressors.

The learning process of Autoencoders involves the minimization of a loss function such as specified in Equation 12:

$$L(x, g(f(x))) \tag{12}$$

where L is a loss function that computes the similarity between the reconstructed input $g(f(x))$ and the original input x. For instance, L can be the Mean Squared Error (MSE) or the Mean Absolute Error (MAE)

If the decoder is linear and L is the MSE, the undercomplete Autoencoder will generate a subspace similar to PCA, described later in Section 2.5.1 (LeCun et al., 2015). On the other hand, Autoencoders with both nonlinear encoder and decoder can learn a more robust representation than PCA. However, if the encoder and decoder are too powerful, the Autoencoder can learn the identity function without extracting helpful information regarding the data distribution. For instance, each training sample $x^{(i)}$ can be expressed as the code i, and the decoder can learn to map these integer indices back to specific training examples. Nonetheless, this rarely occurs in practice (LeCun et al., 2015).

Although the training of Autoencoders is similar to regular Neural Networks, some hyperparameters require special attention, such as:

- **Embedding size**: the size of the latent representation represents a tradeoff be-

tween compression and accurate reconstruction. A smaller size results in more compression, but the Autoencoder may be forced to drop relevant features for reconstruction. On the other hand, the higher the embedding size, the more likely it is that the Autoencoder will memorize or overfit the training data. Usually, this effect can be diminished by adding a term to the loss function that discourages memorization/overfitting (e.g., a regularizer).

- **Number of Layers**: Vanilla Autoencoders are trained with a single hidden layer. However, training multilayer Autoencoders offers several advantages. First, depth can exponentially reduce the computational cost associated with representing a function. Furthermore, depth can also decrease the amount of training data required to train a useful Autoencoder (Goodfellow et al., 2016, p. 506), as the compact and expressive representations enable them to learn more robust and abstract features. Finally, experiments have shown that deep Autoencoders yield significantly better compression than equivalent shallow or linear Autoencoders (Hinton and Salakhutdinov, 2006).

- **Number of nodes per layer**: usually, Autoencoders have symmetric architecture. The number of nodes per layer in the encoder is the same as that in the decoder part, but inversely. In the case of undercomplete Autoencoders, as described earlier, the number of neurons per layer decreases at each layer and increases back in the decoder. In fact, symmetric architecture is not a mandatory rule but is commonly adopted in practice.

- **Loss Function**: as discussed before, generally, the MSE and MAE are used to measure the differences between the original input and the consequent reconstruction. Similarly, binary cross-entropy may also be used for cases where the input is in the range [0-1]. However, other types of Autoencoders can apply different loss functions. For example, Sparse Autoencoders add a sparsity constraint (e.g., L1 regularization) as a penalty term, so only a fraction of the nodes will become active (i.e., nonzero values).

As pointed out by LeCun et al. (2015), the lower-dimensional representations of Autoencoders have some benefits, such as (i) they can improve the performance of tasks such as classification; (ii) fewer parameters require less memory and runtime; and (iii) the hints provided by the lower-dimensional space aid generalization.

2.2 Representation Learning

The success of an ML system is closely tied to the choice and quality of data representation (commonly referred to as features) used for training. Although the criteria

for valuable features depend on the task, it is widely acknowledged that there are sets of features that are representative of a dataset and can be applied to many downstream classifiers or predictors. Focusing on learning representations can yield significant benefits, especially in scenarios where a labeled dataset for a task is limited and using a larger unlabeled dataset is desirable to enhance the performance of the learning system.

Representation Learning refers to the process of learning a parametric mapping from the raw input data domain into a feature vector or tensor that captures the underlying structure and patterns (Le-Khac et al., 2020). This enables the extraction of more abstract and useful concepts that can improve the performance on multiple downstream tasks. Commonly, the input data have a high-dimensional space (e.g., images, text, or sound), and the goal is to encode these data into a meaningful lower-dimensional representation. Representation Learning ensures that the learned mapping generalizes well to new data samples, making it a powerful technique for improving performance on various tasks.

Bengio et al. (2013) defines that a useful representation has the following properties: (i) local smoothness of input and representation; (ii) temporally and spatially coherent in a sequence of observations; (iii) multiple, hierarchically-organized explanatory factors which are shared across tasks; (iv) simple dependencies among factors; and (v) is sparsely activated for a specific input. In addition, especially for Deep Learning, the authors in (Le-Khac et al., 2020) also define other core principles for suitable representation, which are:

- **Distribution**: expressive representation can be a proxy for an exponential amount of configuration for their size.
- **Abstraction and Invariance**: valuable representations can capture more abstract concepts that are also invariant to local and small changes in the input data.
- **Disentangled representation**: a well-designed representation should effectively capture the relevant factors of the data while retaining as much information as possible. At the same time, it is desirable to disentangle these factors, as this allows the reuse of learned features in other learning systems and may be beneficial for explainability.

Natural Language Processing (NLP) is probably one of the areas that most benefited from Representation Learning. With the advent of Deep Learning, more useful representations have been obtained with embeddings computed from a large corpus of texts. In this case, the embeddings consist of a mapping from words represented as one-hot vectors into a distributed representation of real-valued numbers. Other commonly used

words embedding methods include bag-of-words, skip-grams, and global vectors (GloVe) (Pennington et al., 2014; Mikolov et al., 2013).

2.3 Multitask Learning

Multitasking Learning (MTL) is a recent Machine Learning technique in which multiple tasks are solved simultaneously, exploring common aspects and differences between them. This technique is inspired by the human learning process, in which the knowledge obtained in previous problems is applied to learn the pattern of a new problem (Zhang and Yang, 2021). Through MTL, the network can use relevant information in related tasks to improve generalization in all tasks. Besides, this technique can achieve comparable or even better performance than the corresponding single-task counterparts using much fewer training samples (Domhan and Hieber, 2017; Singla et al., 2018).

In a formal definition, given a set of m learning tasks $\{T_i\}_{i=1}^{m}$, where all tasks or a subset of them are related to each other - but not identical -, the goal of MTL is to help improve the learning process of a model for T_i by using the knowledge included in the m tasks. Moreover, there are two underlying factors for MTL based on this definition. The first is task relatedness, which refers to understanding how distinct tasks are related. Such information can be applied to the design of MTL models. Secondly, is the definition of the task itself. For example, there can be tasks of classification, regression, clustering, and many others. Consequently, different tasks result in particular MTL settings.

Regarding the relatedness of MTL tasks, three common issues must be addressed: what to share, when to share, and how to share. Starting with the "when to share" issue, the focus is choosing between single-task and multitasking problems. Generally, the decision is addressed by model selection techniques as cross-validation. The "what to share" defines the scheme of knowledge shared among all tasks. In this case, knowledge is usually shared in three forms: (i) feature-based, where joint features are learned among distinct tasks; (ii) instance-based, in which helpful data instances are identified and used for other tasks; and (iii) parameter-based, which uses model parameters (e.g., weights of Deep Learning models) in a task to help learn model parameters in other tasks. Lastly, the "how to share" defines specific ways to share knowledge between tasks. For instance, in feature-based MTL, the focus is on learning generic feature representations for multiple tasks that can be a subset or a transformation of source features. In parameter-based MTL, there are four main approaches: task clustering, low rank, decomposition, and task relations. More details regarding each approach can be found in (Zhang and Yang, 2021).

In terms of the nature of the tasks, Multitasking Learning can be divided into several categories (Zhang and Yang, 2018), including:

- **Multitask supervised learning**: in this case, each task is a classification or

regression problem. The goal is to predict labels for unseen data using a training dataset with labeled instances.

- **Multitask unsupervised learning**: applied mainly for clustering problems, the goal is to identify functional patterns in a dataset of data instances only.
- **Multitask semi-supervised learning**: similar to the supervised category, but each task in the training set contains labeled and unlabeled data.
- **Multitask active learning**: each task exploits unlabeled data to help learn from labeled data (likewise the semi-supervised learning). However, distinctly unlabeled instances are selected to query their labels actively.
- **Multitask reinforcement learning**: where the goal is to choose actions to maximize cumulative rewards for each task.
- **Multitask online learning**: each task handles sequential data.
- **Multitask multi-view learning**: aims to handle multi-view data with multiple sets of features to describe each instance.

From the Machine Learning perspective, Multitasking Learning can be seen as a form of inductive transfer. The goal of inductive transfer is to take advantage of additional information sources to increase the performance of the current task learning. It helps a model by introducing an **inductive bias**. Therefore, a model can prefer some hypotheses over others. A typical example of inductive bias is L1 regularization, which leads a model toward sparse solutions. In the context of MTL, the inductive bias is supplied by the auxiliary tasks. In this case, the model prefers the hypothesis that explains more than one task. According to Caruana (1997), the inductive transfer can decrease the learning process time and increase the generalization and intelligibility of the model.

The inductive bias obtained through MTL is intuitively compelling. In fact, the authors in (Ruder, 2017) cite a list of reasons why MTL works. In the examples below, we consider that there are two related tasks A and B which rely on a shared representation F, as follows:

- **Implicit Data Augmentation**: Multitasking Learning offers a powerful approach to increasing the sample size for training a model. Since all tasks present specific noise patterns, the goal is to learn a helpful representation that generalizes well and is resilient to data-dependent noise. Consequently, a model trained on multiple tasks simultaneously can learn a more general representation. In contrast, training on a single task alone increases the risk of overfitting that specific task. By learning

tasks A and B together, the model is able to obtain a more valuable representation F by averaging the noise patterns.

- **Attention Focusing**: When a task is characterized by a high degree of noise or the data is limited and high-dimensional, it can be challenging for a model to distinguish between relevant and irrelevant features. In this context, MTL can aid by enabling the model to focus on genuinely significant features since other tasks will provide the additional evidence for feature importance.

- **Eavesdropping**: some features G may be relatively easy to learn for some task B, while being hard to learn for another task A. This may occur especially when (i) A interacts with features in a complex way; or (ii) other features hamper the model's capability to learn feature G. Through the application of MTL techniques, the model can learn G through task B ("eavesdrop"). As shown in (Abu-Mostafa, 1990), it can be achieved by "hints", i.e., training the model directly to predict the most relevant features.

- **Representation Bias**: MTL biases the model in a way to build representations that are useful for multiple tasks. As stated by Baxter (2000), it aids a model to generalize better on new tasks in the future, as a hypothesis space that performs well for a significantly large number of training tasks will also perform well when learning new tasks as long as they are from the same environment.

- **Regularization**: as discussed before, Multitasking Learning serves as a regularization technique by introducing an inductive bias to the model. This approach mitigates the risk of overfitting and decreases the Rademacher complexity of the model (Shalev-Shwartz and Ben-David, 2014), i.e., the capacity to fit random noise.

Compared to Transfer Learning, the setting of MTL is similar but has some substantial differences. First, in MTL, different tasks are treated equally, and the goal is to improve performance among all tasks. In contrast, Transfer Learning is focused on improving the performance of a specific target task with the help of one or more source tasks. In this case, the target task receives more emphasis than the source tasks. In other words, in MTL, there is no distinction between different tasks, whereas, in Transfer Learning, the target tasks attract more attention. Second, from the perspective of knowledge flow, Transfer Learning transfers knowledge from the source task(s) to the target task. In contrast, Multitasking Learning knowledge sharing occurs between all pairs of tasks (Zhang and Yang, 2021).

We can also compare MTL with other methods to represent problems. For example, in Continual Learning (Chen et al., 2018b), the tasks are learned one by one sequentially, whereas, in MTL, all tasks are learned together. Additionally, in multi-label

problems, including multi-output regression, each data point is assigned to multiple categorical/numeric labels. In these cases, if each label is considered an individual task, they can be considered a particular case of Multitasking Learning, where different tasks always share the same data during the training and testing phases. Finally, in multi-view learning (Zhao et al., 2017), a newer paradigm in Machine Learning, each data point is represented with multiple views, each of which comes from a particular set of features. However, all the views are used together to learn the same task. Therefore, multi-view learning can be seen as single-task learning with multiple sets of features, which is, by definition, unlike MTL.

2.4 Network Explainability and Visualization

For some time, Deep Learning methods were considered black boxes because interpreting their outputs was difficult. To overcome this problem, some methods to visualize and explain the predictions of Neural Networks have been proposed in the specialized literature. Some of the most well-known methods are GradCam (Selvaraju et al., 2017), Local Interpretable Model-agnostic Explanations (LIME) (Ribeiro et al., 2016), DeepLIFT (Shrikumar et al., 2016, 2017), and SHAP (Lundberg and Lee, 2017). Furthermore, we cite other variations of SHAP and highlight its advantages.

2.4.1 SHAP

The SHapley Additive exPlanations (SHAP) is a method to explain the output of any machine learning model using a solid theoretical foundation of game theory. The classical Shapley values (Shapley, 1953) from game theory and their related extensions are combined to optimal credit allocation with local explanations. Nowadays, the method is considered state-of-the-art for ML explainability.

The Shapley values quantify each player's contribution to a game. In the context of Machine Learning, the "game" is considered the model's outcome, while the "players" are represented by one or more features included in the model. In the case of images, specifically, pixels can be grouped (superpixels) to explain predictions distributed among them. Therefore, the goal of SHAP is to measure the contribution of each feature concerning the model's prediction.

Given the original prediction model f to be explained and the explanation model g, the Shapley value explanation is represented as an additive feature attribution method (Equation 13), which is a linear function of binary values:

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \tag{13}$$

where $z' \in \{0,1\}^M$ is the coalition (subset) vector, M denotes the number of simplified input features (coalition size) and $\phi \in \mathbb{R}$ is the feature attribution for a feature j (Shapley values). Many methods havve explanation models that matches the Equation 13, including LIME (Ribeiro et al., 2016) and *DeepLIFT* (Shrikumar et al., 2016, 2017).

To compute the contribution of each feature (ϕ_i), the SHAP method requires the model to be retrained on all subsets of features $S \subseteq F$, where F is the complete set of features. A corresponding importance value will be assigned to each feature to represent the effect on the model prediction when such feature is included. To compute this effect, a model is trained with the presence of that feature ($f_{S\cup\{i\}}$), and another model is trained without that feature (f_S). Subsequently, the predictions of the two models are compared for the current input $f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S)$. In this case, x_S denotes the values of the input features in the set S. The preceding differences are computed for all possible subsets since the effect of withholding a feature also depends on other features in the model. The Shapley values are the weighted average of all possible differences (Equation 14):

$$\phi_i = \sum_{S\subseteq F\setminus\{i\}} \frac{|S|!(|F|-|S|-1)!}{|F|!} \left[f_{S\cup\{i\}}\left(x_{S\cup\{i\}}\right) - f_S(x_S)\right] \tag{14}$$

Since a model is trained for every subset of S, the exact computation of Shapley values is expensive. For $|F|$ features, there is a total of $2^{|F|}$ subsets of S. However, in the paper that presents SHAP (Lundberg and Lee, 2017), the authors propose a model-agnostic approach to approximate Shapley values, called Kernel SHAP, and four other model-type-specific approximation methods: Linear SHAP, Low-Order SHAP, Max SHAP, and Deep SHAP. The Deep SHAP is a specific version of SHAP, particularly designed for Deep Networks. The method is based on DeepLift (Shrikumar et al., 2016, 2017), which approximates SHAP values by considering the input features independent of each other and the deep model as linear. In Deep SHAP, the SHAP values of the entire network are a combination of the SHAP values computed for smaller network components. It generates effective linearization from the SHAP values computed for each component. Please refer to the original paper for further explanation of other methods.

The SHAP method has the four properties of classical Shapley values:

1. **Efficiency:** the sum of the Shapley values for all players is equal to the value of the total coalition.

2. **Symmetry**: all players have a fair chance to join the game.

3. **Dummy**: if a player i has no contribution to a coalition S (i.e., for each S, $f_{S\cup\{i\}} = f_S$), then its contributions is zero ($\phi_i = 0$).

4. **Additivity**: for any pair of games a and b, $\phi(a+b) = \phi(a) + \phi(b)$. This property enables simple arithmetic summation.

Some of the main advantages of SHAP are discussed next. First, it is model-agnostic, i.e., SHAP makes no prior assumption of the model and can work with any ML algorithm. Second, this method ensures consistency. It means that even if features are removed from the data, the others continue to contribute with the same importance (Shapley values). Lastly, SHAP can explain not only how each feature contributes to each sample (*local interpretability*) but also at a global level (*global interpretability*) by aggregating the local results.

2.5 Dimensionality Reduction

There are many applications for algorithms of dimensionality reduction. Usually, these methods are used when the cardinality of features is significantly larger than the number of samples (also called the "*curse of dimensionality*"). However, they may also be applied to project data into a lower-dimensional space (usually 2D/3D), aiming to simplify the visualization and search for patterns in data. The following subsections describe two methods of dimensionality reduction used in this work.

2.5.1 Principal Component Analysis

The Principal Component Analysis (PCA) (Wold et al., 1987) is an unsupervised Machine Learning algorithm for dimensionality reduction with minimal information loss. In PCA, the information is based on data variance. The idea is that features with a high variance contain more information because their data are more distributed than features with a low variance. Accordingly, PCA projects the data into a subspace where the axis with the highest variances, also called *principal components*, are preserved. More details of PCA are described below.

The foremost step of PCA is to compute the **eigenvectors** (principal components) of the input data. The eigenvectors explain the variance of the data along the new axes and determine the direction in the new feature space. In PCA, the eigenvectors are organized into a projection matrix and each eigenvector is associated with an **eigenvalue**. The eigenvalue can be interpreted as the magnitude of the corresponding eigenvector. If the eigenvalues have a similar magnitude, it can be considered a valuable indicator that the original data are already in a suitable subspace. Otherwise, if the magnitude of some eigenvalues is much higher than others, the corresponding eigenvectors may be chosen since they contain more information about our data distribution. Likewise, eigenvalues near zero are less informative and may be discarded when constructing the new subspace.

The classical approach of PCA (Wold et al., 1987) computes the covariance matrix, where each element represents the covariance between two features, as computed by Equation 15:

$$\sigma_{jk} = \frac{1}{n-1}\sum_{i=1}^{n}(x_{ij} - \bar{x_j})(x_{ik} - \bar{x}_k) \tag{15}$$

which can be rewritten in matrix form as Equation 16:

$$S = \frac{1}{n-1}((x - \bar{x})^T(x - \bar{x})) \tag{16}$$

where $\bar{x}$ is a D-dimensional vector, in which each value corresponds to the average of each feature, and n is the number of features per sample. In addition, x is the data matrix, where a row represents each sample and the features are columns. In practice, if we have a dataset with four features, for instance, the covariance matrix will have the following structure:

$$\begin{bmatrix} var(1) & cov(1,2) & cov(1,3) & cov(1,4) \\ cov(1,2) & var(2) & cov(2,3) & cov(2,4) \\ cov(1,3) & cov(2,3) & var(3) & cov(3,4) \\ cov(1,4) & cov(2,4) & cov(3,4) & var(4) \end{bmatrix}$$

where the main diagonal corresponds to the variance of each dimension and the remaining elements are the covariance between each dimension pair. An attractive property of the covariance matrix is that the sum of the main diagonal is always equal to the sum of eigenvalues.

The eigenvectors and eigenvalues of PCA can also be computed using the correlation matrix. Although the resulting matrices are different, they result in the same eigenvectors and eigenvalues since the correlation matrix is given by the standardization of the covariance matrix (Equation 17):

$$corr(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y} \tag{17}$$

Given the projection matrix W computed by PCA, the original data x can be projected onto the new subspace S by applying Equation 18:

$$S = (x - \bar{x}) \times W \tag{18}$$

Note that the original space x can be reconstructed by applying Equation 19:

$$x = (S \times W^{-1}) + \bar{x} \tag{19}$$

Finally, the Linear Discriminant Analysis (LDA) (Izenman, 2013) is a similar dimensionality reduction technique. In contrast to PCA, LDA is a supervised method that maximizes component axes for class separation. Although it appears that LDA performs better than PCA in reducing dimensionality for classification problems, this is not always true. Performance comparisons in terms of accuracy for image recognition problems have shown that PCA performs better than LDA if the number of samples per class is relatively low (Martinez and Kak, 2001).

2.5.2 t-SNE

The t-Distributed Stochastic Neighbour Embedding (t-SNE), developed by van der Maaten and Hinton (2008), is a nonlinear dimensionality reduction method well suited for visualizing high-dimensional datasets. It is based on Stochastic Neighbor Embedding (Hinton and Roweis, 2002), but the main difference is the utilization of the t-Student distribution to represent the data in low dimensions.

The t-SNE algorithm comprises two main stages. First, the symmetric probability of points in the original high-dimensional space is computed. Higher probabilities are assigned to similar points, whereas dissimilar points are assigned a lower probability. Such probabilities are computed using a t-Student kernel with a given degree of freedom. Secondly, a similar probability distribution is defined over the points mapped into the low-dimensional space. The Kullback-Leibler divergence (also called relative entropy) between these two distributions is minimized through a gradient descent technique. A simplified version of t-SNE is shown in Algorithm 1, and the details are explained below.

Given an object i, and each potential neighbor j in the high-dimensional space, the joint probability p_{ij} that computes the pairwise similarity between i and j is defined as in Equation 20:

$$p_{ij} = \frac{exp(-d_{ij}^2)}{\sum_{k \neq i} exp(-d_{ik}^2)} \tag{20}$$

In the original SNE, a conditional probability $p_{i|j}$ is used instead. Also, the t-SNE employs the scaled Euclidean Distance as the dissimilarity metric d_{ij}^2 between two high-dimensional points x_i and x_j (Equation 21). However, other distance metrics can also be used when appropriate. One important detail is that the pairwise similarity p_{ij} is not robust to outliers since $\|x_i - x_j\|^2$ will be large. In these cases, the values of p_{ij} will be minimal and the location of its low-dimensional map point will have minimal effect on the

Algorithm 1 Simple version of t-Distributed Stochastic Neighbor Embedding

Data: data set $X = \{x_1, x_2, ..., x_n\}$,
cost function parameter: perplexity ρ,
optimization parameters: number of iteration T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\gamma^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
compute pairwise affinities $p_{j|i}$ with perplexity ρ (using Equation 20)
set $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2n}$
sample initial solution $\gamma^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
for $i = 1$ **to** T **do**
    begin
    compute low-dimensional affinities q_{ij} (using Equation 24)
    compute gradient $\frac{\partial C}{\partial \gamma}$ (using Equation 27)
    set $\gamma^{(t)} = \gamma^{(t-1)} + \eta\frac{\partial C}{\partial \gamma} + \alpha(t)(\gamma^{(t-1)} - \gamma^{(t-2)})$
    end
end

cost function. In practice, such problem is prevented by computing $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2n}$ instead. This ensures that $\sum_j p_{ij} > \frac{1}{2n}$ for all the data points x_i. As a result, each data point x_i has a significantly higher contribution to the cost function.

$$d_{ij}^2 = \frac{\|x_i - x_j\|^2}{2\sigma_i^2} \tag{21}$$

where σ_i can be either defined by experimentation or via a binary search that makes the entropy of the distribution over neighbors equal to $log\ \rho$. In this case, ρ is an input parameter of the cost function called "perplexity." The perplexity may be interpreted as a measure of the effective number of neighbors and is defined by Equation 22:

$$\rho(P_i) = 2^{H(P_i)} \tag{22}$$

where $H(P_i)$ is the Shannon entropy of probability P_i measured in bits by Equation 23:

$$H(P_i) = -\sum_j p_{j|i}\ log_2\ p_{j|i} \tag{23}$$

In general, the performance of t-SNE is reasonably robust to changes in perplexity and typical values are between 5 and 50.

The similarity in low-dimensional space is given by Equation 24:

$$q_{ij} = \frac{exp(-\|y_i - y_j\|^2)}{\sum_{k\neq i} exp(-\|y_i - y_k\|^2)} \tag{24}$$

where y_i and y_j represent the low-dimensional counterparts of the high-dimensional data points x_i and x_j. Usually, for tasks in which the dimensionality of the data is reduced to 2D or 3D, the t-Student distribution is applied with one degree of freedom, and Equation 24 can be rewritten as Equation 25

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}} \tag{25}$$

Finally, t-SNE attempts to minimize a single Kullback-Leibler divergence (Equation 26) between a joint probability distribution P in the high-dimensional space and a joint probability distribution Q in the low-dimensional space:

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}} \tag{26}$$

In contrast to the SNE implementation, the t-SNE is symmetric since $p_{ij} = p_{ji}$ and $q_{ij} = q_{ji}\ \forall i, j$. The main advantage of such symmetry is the simpler gradient expression, which is faster to compute. The gradient of cost function C with respect to point y_i is given by Equation 27:

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j) \tag{27}$$

And the set $\gamma^{(t)} = \{y_1, y_2, ..., y_N\}$ is updated by the momentum term given by Equation 28:

$$\gamma^{(t)} = \gamma^{(t-1)} + \eta \frac{\partial C}{\partial \gamma} + \alpha(t)(\gamma^{(t-1)} - \gamma^{(t-2)}) \tag{28}$$

where $\gamma^{(t)}$ indicates the solution at iteration t, η denotes the learning rate, and $\alpha(t)$ represents the momentum at iteration t. For the derivation of t-SNE gradient, please refer to Appendix A of (van der Maaten and Hinton, 2008).

Compared with PCA, the t-SNE does not conserve the original space's distances or densities. Only the neighborhood is preserved, but up to a certain degree. Furthermore, due to the non-deterministic behaviour of t-SNE, there is no guarantee of convergence to the global optimum of the cost function and the performance is not clearly defined for general dimensionality reduction tasks.

2.6 Performance Measures

There are a variety of performance measures to evaluate algorithms in binary classification problems. In common, most of these metrics take into account some of/all the four possible results present in the binary confusion matrix: True Positives (TPs), True Negatives (TNs), False Positives (FPs), and False Negatives (FNs). In this subsection, the evaluation metrics used in this study are described. Comments on the advantages and disadvantages of each are also made when appropriate.

2.6.1 Accuracy

Accuracy is one of the most commonly used metrics to evaluate binary classifiers. In simple terms, it measures how many predictions were correct among all samples in a given dataset. The output of this metric is a score $\in [0, 1]$, which is interpreted as the proportion of correct predictions (i.e., true positives and true negatives) among the total number of samples analyzed. Accuracy is described by Equation 29:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{29}$$

Although accuracy is one of the most used metrics to evaluate binary classifiers, it is not well suited for unbalanced datasets. Suppose a dataset in which 80% of the samples are from the positive class, and the remaining are from the negative class. If some classifier outputs only the positive class, the overall accuracy will be 80%. Therefore, accuracy must only be considered valid when the class distributions are considerably uniform.

2.6.2 Precision & Recall

Precision and Recall are two metrics highly adopted in the evaluation of binary classifiers. While Precision measures how much we can trust a model when it predicts the positive class, Recall measures how accurate a model is regarding the positive class. In other words, the Precision metric answers the question "*given the positive predictions, how many were correct?*", while Recall answers "*given the positive samples, how many predictions were correct?*". Precision and Recall are also known as Positive Predictive Value (PPV) and Sensitivity, respectively, and are defined by Equations 30 and 31:

$$Precision = \frac{TP}{TP + FP} \tag{30}$$

$$Recall = \frac{TP}{TP + FN} \tag{31}$$

Even though the Precision and Recall metrics are considered more robust than accuracy for evaluating binary classifiers, both metrics present some drawbacks. First, they are purpose-specific. For example, Precision is a better choice when we want to trust the prediction of the positive class. However, if a model is optimized to maximize Precision only, it can bias the classifier to avoid uncertain positive samples, thereby increasing the number of false negatives. On the other hand, Recall is more pertinent when we need a classifier that may correctly identify positive samples, although it can generate more false positives. Hence, in the case of a problem where Precision and Recall are equally important, the f-measure can be computed to generate a balanced score between these two metrics. The f-measure is better described in the following subsection.

Secondly, by taking a closer look at the Equations 30 and 31, we can notice that they do not consider the True Negatives. Therefore, both metrics are more appropriate when the positive class is more important than the negative class. This is the case for detection problems, for example, where identifying the negative class (usually background) is less important than the positive class (object of interest). Nevertheless, this is not always true for some classification problems. For instance, in fraud-detection problems, the correct prediction of the True Negatives has a higher priority over the True Positives. For these cases, we can compute the Negative Predictive Value and Specificity metrics, which will be described later.

2.6.3 F-measure

The f-measure, also called f-score or f1-score, is a metric that balances the values of Precision and Recall. The importance of f-measure is better understood by an example. Suppose there is a face detection problem where two different models, named A and B, are applied to an image with ten faces. Model A returned five detections, where 3 were real faces (TP), and 2 were incorrect. In this case, model A has a 30% Recall (3 out of 10 detected faces) and 60% Precision (3 out of 5 correct predictions). Otherwise, model B returned two detections, which were all real faces. In this case, the Recall is 20% (2 out of 10 detected faces), and the Precision is 100% (all detections were real faces). Therefore, if Precision and Recall are equally crucial for this problem, which model is better: model A with 30% of Recall and 60% of Precision, or model B with 20% of Recall and 100% of Precision? To answer this problem, we can compute the f-measure described by Equation 32:

$$F1 = 2\frac{Precision \cdot Recall}{Precision + Recall} \tag{32}$$

In the example cited above, the f-measure of model A is 40%, while the model B is 33.3%. Therefore, model A can be considered a better model for this case because it has

a higher f-measure and, consequently, a better equilibrium between Precision and Recall.

By definition, the f-measure is considered the harmonic mean between Precision and Recall. This is harmonic because, in contrast to the arithmetic mean, the metric cannot be made arbitrarily large by changing only the Precision or Recall to a larger value (while keeping the other metrics unchanged). Therefore, to increase f-measure considerably, both Precision and Recall must be higher simultaneously.

2.6.4 F-Beta

F-Beta is a generalization of F-measure. It adds a β term to weigh how much Recall is more important than Precision. The F-Beta is defined by Equation 33:

$$F_\beta = \frac{(1+\beta^2) \cdot P \cdot R}{\beta^2 \cdot P + R} \tag{33}$$

As can be noticed, when $\beta = 1$, the original formula of f1-score is obtained. Another typical value is $\beta = 2$, generating the f2-score. In this case, the Recall has a higher weight than Precision. In other words, better Recall is more critical for the classifier than Precision. This is useful to ensure that the classifier is better at identifying the positive class, even though it might generate more false positives. In the detection example cited before, the detector would try to detect as many real faces as possible but would return more false positives (detections that are not real faces).

2.6.5 Negative Predictive Value & Specificity

The metrics Specificity and Negative Predictive Value (NPV) are equivalent to Recall and Precision, respectively, but for the negative class. In analogy to Precision and Recall, the Specificity answers the question "*given the negative samples, how many predictions were correct?*", while the NPV answers "*given the negative predictions, how many were correct?*". These metrics are defined by Equations 34 and 35:

$$Specificity = \frac{TN}{TN + FP} \tag{34}$$

$$NPV = \frac{TN}{TN + FN} \tag{35}$$

Since both NPV and Specificity are similar to Precision and Recall, they share the same problems of these metrics described in Section 2.6.2. Hence, NPV and Specificity are purpose-specific (but for the negative class) and, analogously, do not consider the positive class.

2.6.6 Matthews Correlation Coefficient

The Matthews Correlation Coefficient (MCC), also known as the phi (ϕ) coefficient, is a binary classification metric that considers all four categories: True Positives, True Negatives, False Positives, and False Negatives. It was introduced by biochemist Brian W. Matthews in 1975 (Matthews, 1975) and is defined by the Equation 36

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{36}$$

The MCC is considered a balanced metric and can be used even in highly unbalanced datasets. The output of MCC is a normalized coefficient between -1 and $+1$. A coefficient of $+1$ indicates a perfect prediction, whereas -1 represents the total disagreement between the prediction and ground truth. In addition, a coefficient of 0 indicates that the predictions are not better than random guesses.

Because MCC considers the four possible results of binary classification problems, it presents some advantages compared to other classification metrics. For example, suppose a trained classifier with a confusion matrix of TP = 90, FP = 5, TN = 1, and FN = 4. In this case, the accuracy is 91% and the f-measure is 95.24%. On the other hand, the MCC gives a score of 0.14, which means the algorithm is mainly predicting the most frequent class and performing similarly to random guessing. As suggested by Chicco (2017), the use of MCC is highly encouraged to evaluate performance in the test set of any binary classification problem.

2.6.7 Equal Error Rate

The performance of biometric systems is usually measured using metrics such as False Acceptance Rate (FAR), False Rejection Rate (FRR), and Equal Error Rate (EER). The False Acceptance Rate, also known as False Match Rate (FMR), is the probability of an incorrect match between a given input pattern and a non-matching template. In other words, it computes the proportion of invalid inputs that are incorrectly accepted. On the other hand, the False Rejection Rate, also known as False Non-Match Rate (FNMR), represents the likelihood of a system falsely rejecting a match between the input pattern and a genuine matching template. Alternatively, it computes the proportion of valid matches incorrectly rejected. According to Ross et al. (2006), FAR and FRR can be formally defined in the continuous case by the Equations 37 and 38, respectively:

$$FAR(\tau) = \int_{\tau}^{\infty} p(s|impostor)\ ds \tag{37}$$

$$FRR(\tau) = \int_{\tau}^{\infty} p(s|genuine)\ ds \tag{38}$$

where $p(s|impostor)$ and $p(s|genuine)$ correspond to the probability distributions of a score s under genuine and impostor conditions, appropriately, and τ is a threshold for defining whether a given individual is genuine or an impostor.

Finally, the EER is designated by the point at which the FAR and FRR curves intercept each other (see Figure 3). Therefore, EER represents the rate at which both acceptance and rejection errors are equal, i.e., $FAR = FRR$ or $p(s|impostor) = p(s|genuine)$. All the mentioned metrics (FAR, FRR, and EER) are inversely proportional to the performance of the biometric system (the lower, the better).

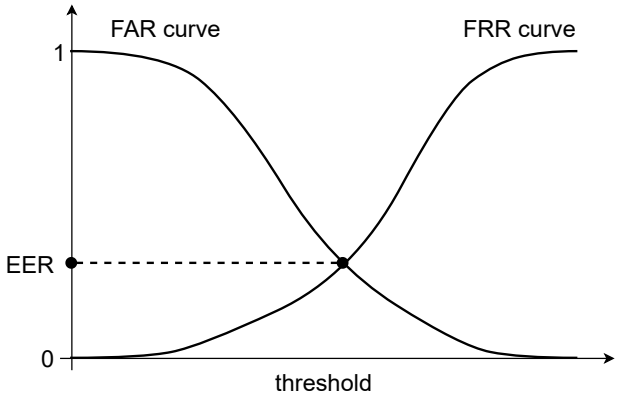


Figure 3: The typical curves of the FAR and FRR error rates, plotted side by side, in relation to the threshold τ configured for the system. The EER is represented by the intersection point of the curves. Source: own elaboration.

Since FAR and FRR are computed for a given threshold τ, biometric systems can be calibrated by manually choosing the threshold that best fits the desired needs. However, this represents a tradeoff between convenience and security. For example, when security issues are critical (e.g., banking systems), threshold τ may be adjusted in a way that the FAR is minimal. However, genuine users may have blocked access until the system is entirely sure about their profile. On the other side, if τ is fine-tuned to improve user convenience, the system can allow access to non-legitimate users.

2.6.8 Wing Loss

Wing loss is a recently proposed function specially designed for facial landmark localization. It was introduced as an alternative to the commonly used MSE and MAE losses. The function is named after its shape, which resembles a wing (see Figure 4). It was created by (Feng et al., 2018) and can be defined by Equation 39:

$$wing(x) = \begin{cases} w \ln(1 + |x| / \epsilon) & if \ \ |x| < w \\ |x| - C & otherwise \end{cases} \tag{39}$$

where w is a non-negative constant that defines the range of the nonlinear part to $(-w, w)$, ϵ limits the curvature of the nonlinear region, and $C = w - w \ln(1 + w/\epsilon)$ is another constant that smoothly links the piecewise-defined linear and nonlinear parts. A critical detail regarding Wing Loss is that ϵ may not be set to a very small value because it makes the training unstable and causes an explosion of gradients.

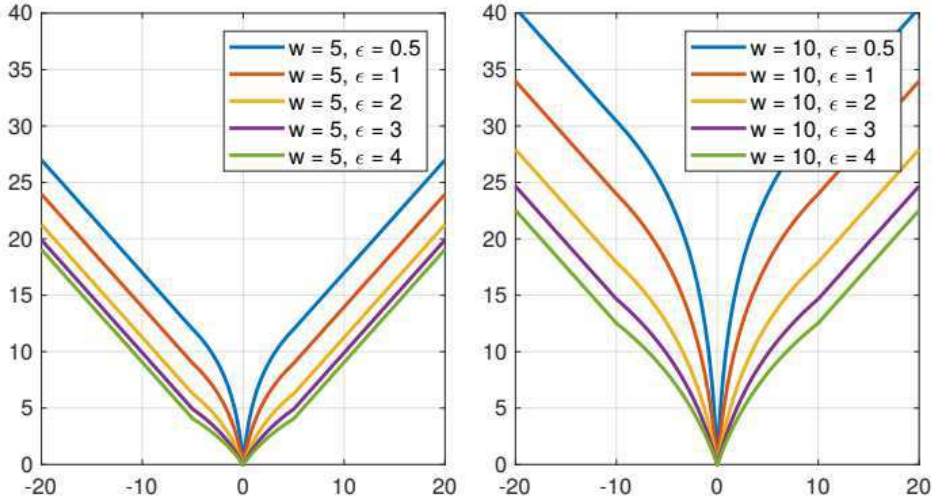


Figure 4: Wing loss function (Equation 39) plotted for different settings of w and ϵ. Source: (Feng et al., 2018).

The main characteristic of Wing loss is that it amplifies the impact of samples with small and medium-range errors during the training step. The influence of small errors is enhanced by the ln component in the formula. In addition, in the case of large errors, Wing loss promotes a fast recovery behaving as L1 and being robust to outliers.

2.7 The ISO/IEC 19794-5 Standard

In 1980, the International Civil Aviation Organization (ICAO) began a project focused on the standardization of automatic biometric identification of people using machines (ICAO, 2003). A specific working group was established to determine the most suitable way of "uniquely encoding a particular physical characteristic of a person into a biometric-identifier that can be machine-verified to confirm the presenter's identity" (ICAO, 2003). Initially, three physical attributes were chosen for possible applications: face, fingerprint, and iris. Later, in the "Berlin resolution" (2002), the ICAO has chosen the face as the primary globally interoperable biometric trait for machine-assisted identity confirmation in Machine-Readable Travel Document (MRTD) (Ferrara et al.,

2012b). However, the decision also states the possibility of identity confirmation through a fingerprint or iris to support machine-assisted decisions.

Following the ICAO guidelines, the International Organization for Standardization (ISO), together with the International Electrotechnical Commission (IEC), proposed a standard for facial photography to be used in electronic passports. The ISO/IEC 19794-5 (ISO, 2007) standard specifies rules and a record format for encoding, recording, and transmitting facial image information. It also defines a set of environmental conditions, photographic properties, shooting features, and digital attributes of facial images. For example, a facial image may have a uniform background with the absence of shadows in any region of the image to be included in an electronic passport. The complete list of requirements can be seen in Table 1, and examples of non-compliant images for each requirement are given in Figure 5. The ISO/IEC 19794-5 standard is also referenced in the ANSI/NIST-ITL 1–2011 standard (Wing, 2011) as one of the standard profiles for face acquisition. In 2019, a new version of the ISO/IEC 19794-5 standard was released. The ISO/IEC 39794-5 provides more detailed information about each requirement evaluation.

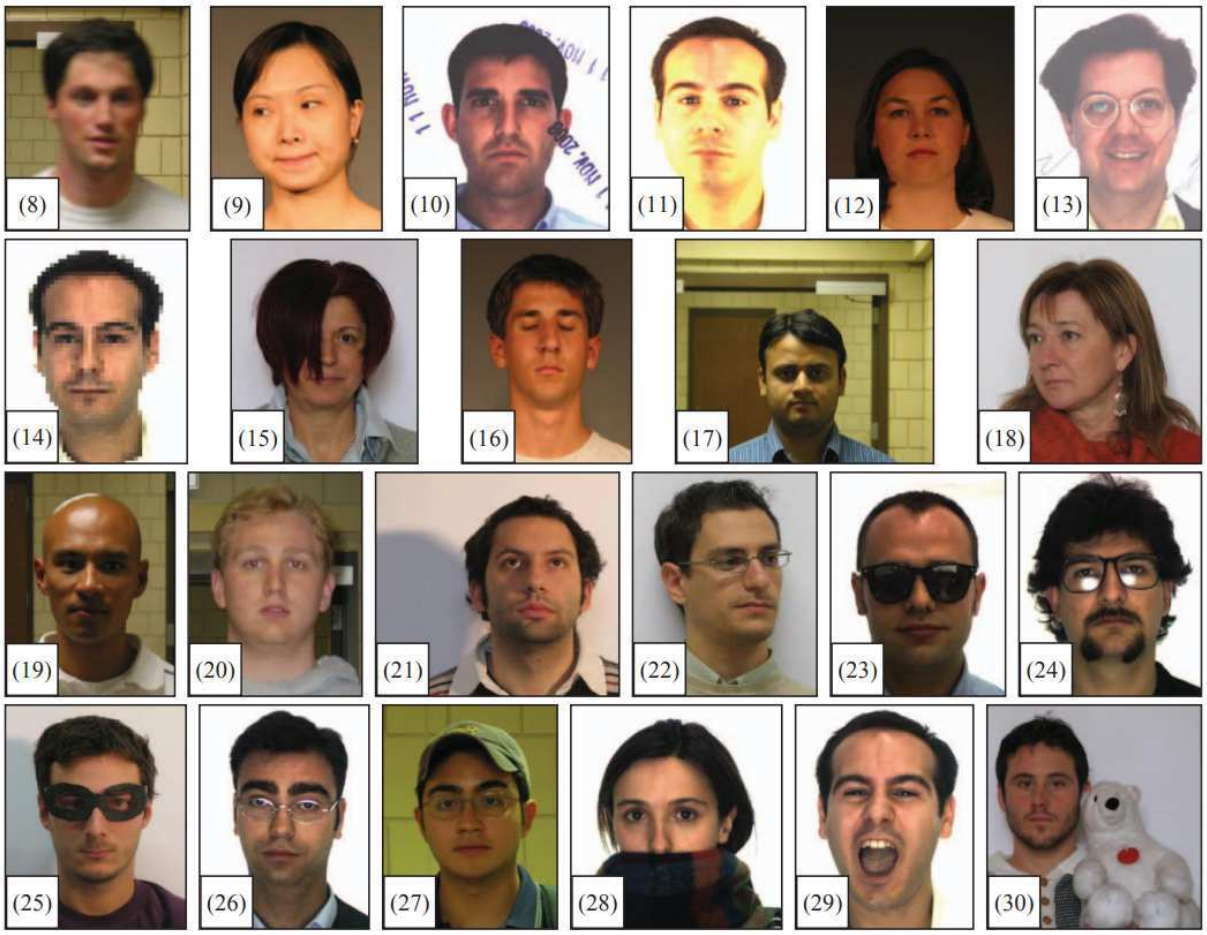


Figure 5: Examples of non-compliant images for the requirements 8-30 listed in Table 1. Source: Maltoni et al. (2009).

Table 1: Description of facial image quality tests performed by BioLab, according to ISO/IEC 19794-5 standard.

Req. #	Test description
Facial feature extraction tests	
1	Eye center location accuracy
2	Face location accuracy
Geometric tests	
3	Eyes distance (min. 90 pixels)
4	Relative vertical position
5	Relative horizontal position
6	Ratio of head width
7	Ratio of head height
Photographic and pose-specific tests	
8	Blurred
9	Looking away
10	Ink marked/creased
11	Unnatural skin tone
12	Too dark/light
13	Washed out
14	Pixelation
15	Hair across eyes
16	Eyes closed
17	Varied Background
18	Roll/pitch/yaw rotations greater than a predefined thresholds
19	Flash reflection on skin
20	Red eyes
21	Shadows behind head
22	Shadows across face
23	Dark tinted lenses
24	Flash reflection on lenses
25	Frames too heavy
26	Frame covering eyes
27	Hat/cap
28	Veil over face
29	Mouth open
30	Presence of other faces or toys too close to face

2.8 FVC-onGoing

The Fingerprint Verification Competition (FVC) is an event organized by four leading institutions: *Biometric System Laboratory* (University of Bologna), *Pattern Recognition and Image Processing Laboratory* (Michigan State University), *Biometric Test Center* (San Jose State University), and *Biometric Recognition Group - ATVS* (Universidad Autonoma de Madrid). The main goal of the FVC was to benchmark systems that evaluate fingerprint images. Initially, the competitions used to happen biannually (2000, 2002, 2004, and 2006), and such events drew attention from the academic community and the industry related to biometrics. Over the years, the FVC has become a reference for evaluating fingerprint systems, allowing research groups, private companies, and even individual developers to compare their methods and follow state-of-the-art fingerprint research.

After 2006, the organizers of FVC decided to create the FVC-onGoing. It turned the original biannual FVC into an ongoing competition, i.e., every participant could register and submit new algorithms continuously. Moreover, new competitions regarding fingerprints were added to FVC-onGoing, like fingerprint orientation extraction, fingerprint indexing, and minutiae matching.

In 2012, a new benchmark category related to the ISO/IEC 19794-5 standard was added to FVC-onGoing (Ferrara et al., 2012b), called Face Image ISO Compliance Verification (FICV). The main goal of this benchmark is to evaluate algorithms that assess the compliance of face images with the ISO standard. In total, 24 out of 30 requirements are evaluated by the FICV: the *Eye Center Location* (01) and all photographic and pose-specific tests (8–30) - see Table 1. Moreover, the FICV evaluates whether the image can be converted to a Token Format (see Section 9.2.3 in (ISO, 2007)) based on the eye positions predicted by a submitted algorithm. In short, an image is considered *tokenizable* (without padding) if:

- the distance (E_{Dist}) between eyes is at least 60 pixels;
- the rectangular region of size $W \times H$ (with $W = 4 \cdot E_{Dist}$ and $H = W \cdot 4/3$), determined so that the eyes are horizontally aligned and their center is in position $C_E = (W \cdot 1/2, W \cdot 3/5)$, is totally enclosed in the original image (see Figure 6).

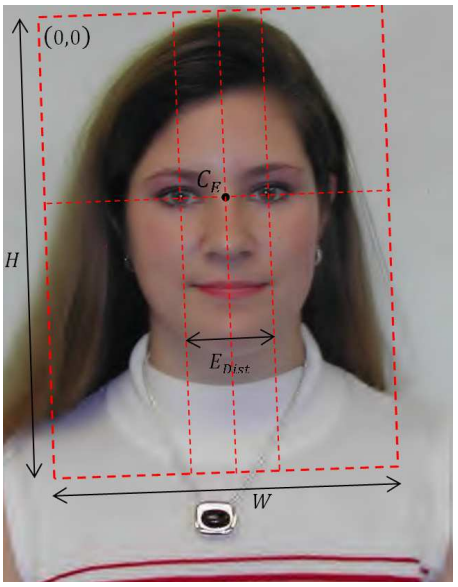


Figure 6: Geometric characteristics of the token image format. Source: (FVC-onGoing, 2006).

Two datasets are used to benchmark the algorithms submitted to the FICV competition: *FICV-TEST* and *FICV-1.0*. The *FICV-TEST* is a small but representative dataset (720 images) that is only used to test the compliance of the submitted algorithm

with the testing protocol (described below). The results computed on this dataset are only visible to the participant and are not considered as official results of the competition. On the other hand, the *FICV-1.0* contains 4868 images and is private to all participants. Such dataset is detailed in Ferrara et al. (2012b). All results presented in the literature and this thesis were computed using the *FICV-1.0* dataset.

All the participants who wish to submit an algorithm for evaluation in the FICV competition must follow a well-defined protocol. The algorithm must be submitted as a Windows 32 bits (Win32) console application as an executable file of up to 50MB compressed using the zip format. Such executable receives a face image as input and must output the coordinates (x, y) of the left and right eye centers and a score in the range of $[0, 100]$ to indicate the compliance degree of the input image for each photographic requirement of Table 1.

Additionally to the submission protocol, the algorithm must still comply with certain constraints. For example, the maximum time to process each image is 10 seconds. In the case of time violations, the current image evaluation is considered a failure. Also, there is a minimum break of 12 hours between two consecutive submissions by the same participant in the *FICV-TEST* dataset. For the *FICV-1.0*, this interval is 15 days. However, there are no constraints on the memory allocation limit. Also, the memory consumption reported refers to the maximum amount of MB allocated by the method to perform predictions during evaluation.

Finally, since the requirements #1 and #8–30 (see Table 1) represent distinct types of problems, they are evaluated differently. The *Eye Center Location* (01) performance is measured by the relative error according to the distance between the expected and estimated eye positions (d_{eye}). This is the same metric introduced by Jesorsky et al. (2001) and is calculated as follows (Equation 40):

$$d_{eye} = \frac{max(\left\|C_l - \hat{C}_l\right\|, \left\|C_r - \hat{C}_r|\right\|)}{\|C_l - C_r\|} \tag{40}$$

where $C_{l/r}$ and $\hat{C_{l/r}}$ represent the ground truth and positions returned by the algorithm for the left and right eyes, respectively. This measure is scale-independent and allows for comparing datasets with different image resolutions. In the FICV competition, the d_{eye} is evaluated using four particular intervals: $d_{eye} \in [0; 0.1[$, $d_{eye} \in [0.1; 0.2[$, $d_{eye} \in [0.2; 0.3[$, and $d_{eye} \geq 0.3$.

Furthermore, to evaluate the performance of the photographic and pose-specific requirements, the benchmark datasets are divided into 23 subsets, each related to a specific requirement. Also, each subset contains the same number of compliant and non-compliant images. Then, for each requirement, the following performance metrics are computed and

reported:

- EER: the Equal Error Rate;
- FAR_{100}: the lowest FRR for FAR $\leq 1\%$;
- $Zero_{FAR}$: the lowest FRR for FAR=0%;
- $Zero_{FRR}$: the lowest FAR for FRR=0%;
- *RejectionRate*: percentage of images where the algorithm did not evaluate the requirement;
- *Impostor* and *Genuine* score distributions;
- $FAR(\tau)/FRR(\tau)$ curves, where τ represents the acceptance threshold; and
- $DET(\tau)$ curve: the Detection Error Tradeoff, which plot the False Rejection Rate vs. the False Acceptance Rate.

According to FICV, rejections are implicitly included in the metrics computation to discourage the algorithm from rejecting the most uncertain cases and, consequently, improve the performance of the processed images. In these cases, the compliance score is set to zero for the corresponding requirement of the given image. This definition follows the best practices of evaluation systems and is also performed on other benchmarks of FVC-onGoing.

The next chapter presents a literature analysis related to this thesis. It focuses on the most relevant published methods for Multitask Learning and the ISO/IEC 19794-5 standard, which are core concepts of the proposed method.

3 Literature Review

This chapter reviews Deep Multitasking Learning techniques applied to computer vision problems. Emphasis is placed on different types of Deep Multitask architectures and the relevant researches published for each category. Additionally, this chapter includes a historical review of the methods that address the ISO/IEC 19794-5 standard, including the methods published on the FVC-onGoing website (FVC-onGoing, 2006). Finally, this chapter is concluded with a discussion around relevant findings of the literature review on both topics.

3.1 Representation Learning

According to the literature, Representation Learning approaches can be divided into two main categories: generative and discriminative modeling. Both approaches consider that a helpful representation can explain the variation in data by capturing underlying factors, such as hierarchical relationships, data dependencies, and domain-specific factors. However, there are essential differences in the modeling process, as detailed next.

In generative approaches, the representation is learned by modeling the data distribution $p(x)$. In the case of images, it can be the pixels. It presumes that if a helpful model $p(x)$ can create realistic data samples, it must also be able to capture the underlying structure associated with the dependent variable y. The conditional distribution $p(y|x)$ can be determined through Bayes' rule and allows the evaluation of discriminative tasks on y (Le-Khac et al., 2020). Generally, generative approaches are categorized as unsupervised learning and can be seen, for instance, in the works of Goodfellow et al. (2014); Kingma and Welling (2013).

On the other hand, discriminative approaches learn a representation by directly modeling the conditional distribution $p(y|x)$ through a parametric model. This model takes the data sample x as the input and outputs the label variable y. Latent variables $p(v|x)$ are inferred, and downstream decisions $p(y|v)$ are made from these variables. The most applied discriminative approaches are a type of supervised learning (sometimes called "self-supervised"). Examples of this approach are given in (Dosovitskiy et al., 2014; Zhang et al., 2017a).

Compared to the generative approach, discriminative models present some advantages. Primarily, modeling the distribution of x is computationally intensive and is not essential for extracting representations. Furthermore, the generation process of a generative model can be deemed inefficient if the objective is to obtain a lower-dimensional representation. Lastly, generative models usually employ a more expensive objective function specially designed for the input space.

Representation Learning plays an essential role in several fields and applications. First, we highlight Natural Language Processing. It started with the idea of distributed representations for symbolic data introduced by Hinton et al. (1986), which was later implemented by Bengio et al. (2000) in the context of statistical language modeling. In (Collobert et al., 2011), the authors developed SENNA, a convolutional architecture for sharing representations across different NLP tasks, including: part-of-speech tagging, named entity recognition, chunking, semantic role labeling, language modeling, and syntactic parsing. Methods for learning word representations can also be found in the works of Mnih and Kavukcuoglu (2013); Pennington et al. (2014). Usually, they are based on unsupervised objectives for predicting words or word frequencies from raw text. These methods have demonstrated remarkable achievements when used for transfer learning, overcoming the limitations of supervised models across multiple downstream tasks.

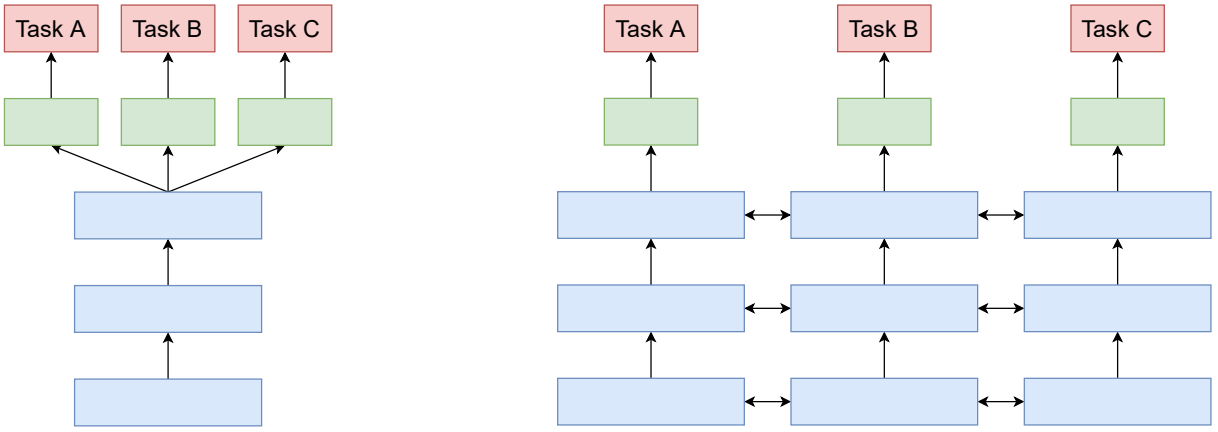
In Computer Vision, the most popular methods for image-feature learning are based on Convolutional Neural Networks. Traditional architectures, such as AlexNet (Krizhevsky et al., 2012), VGG (Simonyan and Zisserman, 2014), and ResNet (He et al., 2016), are examples of models used for this purpose. They extract hierarchical features learned from input images, which can be later applied to other classifiers such as Neural Networks or SVMs. Additionally, encoder-decoder architectures, like Autoencoders, are well-known learning methods for representation learning. Examples of other Computer Vision models and architectures are presented in the next section.

3.2 Multitask Learning

As discussed in the previous chapter, Multitasking Learning can be defined as a technique to learn multiple tasks *jointly* instead of learning each task independently. It has been studied since the '90s, initially focused on Neural Network models (Thrun and Pratt, 1998; Caruana, 1997; Baxter, 2000). More recent approaches have been based on structured sparsity and convex optimization (Argyriou et al., 2006) and kernel methods (Evgeniou et al., 2005).

Historically, MTL methods have been classified as hard or soft parameter-sharing techniques (Vandenhende et al., 2021). In hard parameter sharing, initially proposed by Caruana (1997), network parameters are divided into shared and task-specific parameters. Typically, MTL models that use hard parameter sharing contain a shared encoder with task-specific branches (Kendall et al., 2018; Chen et al., 2018a; Sener and Koltun, 2018). As shown by Baxter (1997), the risk of overfitting in hard parameter sharing is order N smaller than overfitting the task-specific branches, where N is the number of tasks. Intuitively, since it is harder for a model to find a shared representation for all tasks, the probability of overfitting is less likely. On the other hand, in soft parameter sharing, the parameters of each task are handled by a feature-sharing mechanism (Ruder et al., 2019;

Gao et al., 2019; Liu et al., 2019). In this case, each task has a specific model, and the parameters of each model were regularized to be similar. Figure 7 visually explains the differences between hard and soft parameter sharing.



(a) Hard Parameter Sharing (b) Soft Parameter Sharing

Figure 7: Types of parameter sharing in Multitasking Learning. The network backbone is represented in blue, whereas the task-specific heads are indicated in green. Source: own elaboration.

Various techniques and architectures for MTL have been proposed in the literature on Deep Learning. Usually, Deep Multitask Architectures can be divided into encoder-focused and decoder-focused architectures (Vandenhende et al., 2021). The main characteristic of encoder-focused architectures (Kendall et al., 2018; Chen et al., 2018a; Sener and Koltun, 2018) is the presence of an off-the-shelf backbone network, usually called an encoder. The goal of the encoder is to learn a generic representation shared by a set of independent task-specific heads. Differently, the decoder-focused architectures also exchange information during the decoding stage (Xu et al., 2018; Zhang et al., 2018; Vandenhende et al., 2020). The following subsections discuss the most relevant architectures of Deep Multitasking Learning, according to the above classification.

3.2.1 Encoder-focused Architectures

The **Cross-stitch networks** (Misra et al., 2016) combine two given activation maps x_A and x_B, which belong to tasks A and B respectively, in a learnable linear manner. The transformation can be expressed by learnable weights α, as shown in Equation 41.

$$\begin{bmatrix} \bar{x}_A \\ \bar{x}_B \end{bmatrix} = \begin{bmatrix} \alpha_{AA} & \alpha_{AB} \\ \alpha_{BA} & \alpha_{BB} \end{bmatrix} \begin{bmatrix} x_A \\ x_B \end{bmatrix} \tag{41}$$

Each task's α values were initialized in the $[0, 1]$ range and obtained during training through a convex combination. Using this equation, the cross-stitch networks can deter-

mine the degree to which the features are shared between different tasks. However, such networks must be pre-trained before stitching them together to maximize performance. Additionally, the size of the Cross-stitch network increases linearly with the number of tasks.

Neural Discriminative Dimensionality Reduction CNNs (NDDR-CNNs) (Gao et al., 2019) presents a similar architecture with Cross-stitch networks. Nonetheless, a dimensionality reduction component is employed instead of a linear combination to merge all single-task network activations. However, besides the NDDR-CNNs being susceptible to the same problems as the Cross-stitch networks, they also require additional design choices (e.g., where to include the NDDR layers). Nevertheless, NDDR-CNNs and Cross-stitch are limited to local information when the activations from different single-task networks are fused.

The **Multitask Attention Networks** (MTAN) (Liu et al., 2019) are an encoder-focused design that combines an encoder with task-specific attention modules in the backbone network. While the encoder is responsible for computing a general pool of features, the task-specific attention module selects features from the public pool by applying a soft attention mask. Regular convolutional layers with sigmoids are used to implement the attention mechanism. Compared to Cross-stitch networks and NDDR-CNNs, the MTAN model is less prone to scalability issues but is also limited to local information to produce the attention mask.

3.2.2 Decoder-focused Architectures

One of the first decoder-focused architectures published in literature was **PAD-Net** (Xu et al., 2018). Although the input image is still processed by an off-the-shelf backbone network, the backbone features are further processed by task-specific heads that produce the initial predictions for each task. The task-specific heads contain a per-task feature representation of the input image and are recombined by a multi-modal distillation. The main goal of the distillation unit is to extract the cross-task information using a spatial attention mechanism. The output features F_k^o for a given task k and training sample i are computed according to the Equation 42:

$$F_k^o = F_k^i + \sum \sigma(W_{k,l} F_l^i) \odot F_l^i \tag{42}$$

where l is the feature map index, $W_{k,l}$ is the convolution parameter, σ represents a sigmoid function, and $\odot$ denotes element-wise multiplication. However, Equation 42 presumes that task interactions are location-independent. Therefore, there must be no relationship between tasks across the entire image.

Similar to PAD-Net, the **Pattern-Affinitive Propagation Networks** (PAP-Net) (Zhang et al., 2019) improved multi-model distillation. By statistically observing that pixel affinities contribute to a better alignment with common local structures on the task label space, they proposed leveraging pixel affinities to perform multi-modal distillation. A pixel affinity matrix M_{T_j} is computed by estimating pixel-wise correlations for each task-specific head's features. Then, a cross-task information matrix $\hat{M}_{T_j}$ for each task T_j is learned by an adaptive combination of the affinity matrices M_{T_j} for tasks T_i with learnable weights $\alpha_i^{T_j}$, as defined in Equation 43:

$$\hat{M}_{T_j} = \sum_{T_i} \alpha_i^{T_j} \cdot M_{T_i} \tag{43}$$

where i represents the index of other tasks. The task features of a task j are refined by the cross-task information matrix M_{T_j}. In fact, M_{T_j} is dissipated across the task feature space to spread the pixel correlation for task T_j based on the pixel similarities from the other tasks T_i. The learnable weights $\alpha_i^{T_j}$ are obtained by affinity learning layers during the decoding process with different input scales. Unlike the other decoder-focused architectures mentioned, PAP-Net models non-local relationships through pixel similarities computed across the entire image.

The **Joint Task-Recursive Learning** (JTRL), proposed by Zhang et al. (2018), recursively predicts two tasks by increasing higher scales to refine the results of past states gradually. Compared to PAD-Net and PAP-Net, there is also a multi-modal mechanism that combines information from earlier task predictions, which are used to refine the later ones. However, the JTRL model can only predict two tasks sequentially and in an intertwined approach. Moreover, the main drawback of the JTRL model is that it is not simple, or even possible, to extend the architecture to more than two tasks because of the intertwined approach to refine predictions.

In this thesis, an encoder-focused architecture is applied because there are requirements in the ISO/IEC 19794-5 standard that share similar characteristics (e.g., requirements related to the eyes or background). Therefore, the generic representation learned by the encoder may be useful when shared with task-specific heads. Compared with the encoder-focused architectures cited in this chapter, the proposed method does not require pre-training, such as Cross-stitch networks or advanced design choices as in NDDR-CNNs. In addition, the proposed method is easy to scale like MTAN networks. Further details are discussed in Chapter 4.

3.3 Methods for the ISO/IEC 19794-5 Standard

One of the first studies to address ICAO requirements was proposed by Sang et al. (2009). It presents methods to evaluate the requirements related to illumination conditions and facial poses based on Gabor wavelet features. Furthermore, a method for evaluating image blur is proposed using the Discrete Cosine Transform (DCT). The authors assess their methods using images from the CMU-PIE (Sim et al., 2002) and FERET (Phillips et al., 1998) datasets. However, only analytical results are presented.

The popularization of methods for ISO/IEC 19794-5 standard can be credited to the Biolab group from the University of Bologna. In 2009, they presented the Biolab-ICAO framework (Maltoni et al., 2009), a benchmark tool for systems assessing face image compliance to ICAO requirements. In 2012, the benchmark was refined, and the official ground-truth face database (4868 images) and testing protocol were presented (Ferrara et al., 2012b). In summary, the dataset has 5588 images that were either collected from different sources or artificially generated. A subset of 720 images is publicly provided to the participants, while the 4468 remaining images are private and used to evaluate submitted algorithms. A full description of this dataset is provided in Section 4.1.1. Moreover, the authors proposed BioLabSDK, the first known method published in the literature to evaluate all 23 face-and-pose requirements (8–30 in Table 1). BiolabSDK uses different color spaces, a face detector, and points that define the face and its elements to generate a score for each requirement. The paper also compares BioLabSDK against two anonymous SDKs using the EER and Rejection Rates. The results can be seen in the first three columns of Table 2.

Table 2: Comparison of methods for the pose and photography requirements (8–30) of the ISO/IEC 19794-5 standard. The EER and Rejection Rate are presented for each method and were evaluated by the BioLab-ICAO framework in the FICV competition. The "-" indicates the method does not evaluate that requirement or that results were not informed by the authors.

	SDK1		SDK2		BioLab		BioTest		BioPass Face		id3		ICAO SDK		FerraraSeg		Borges et al.		Andrezza et al.		Parente et al.		HMAX	
Req. #	EER	Rej.	EER	Rej.	EER	Rej.	EER	Rej.	EER	Rej.	EER	Rej.	EER	Rej.	EER	Rej.	EER	Rej.	EER	Rej.	EER	Rej.	EER	Rej.
8	26.00	8.90	48.10	0.60	5.20	0.00	30.50	36.00	1.60	3.30	1.70	0.20	48.80	0.00	-	-	-	-	-	-	-	-	-	-
9	27.50	7.10	-	-	20.60	0.00	24.20	3.10	13.30	3.30	15.30	15.80	47.50	2.50	-	-	16.90	1.20	-	-	-	-	10.00	0.16
10	-	-	-	-	3.40	1.20	3.60	1.40	4.80	0.50	-	-	-	-	-	-	-	-	-	-	-	-	-	-
11	18.70	4.80	50.00	0.80	4.00	0.20	5.10	1.70	1.90	0.00	2.10	0.20	50.00	88.8	-	-	-	-	3.70	0.00	-	-	14.29	0.30
12	-	-	3.10	0.00	4.20	0.00	4.60	0.20	3.10	0.20	2.90	0.00	27.70	0.00	-	-	-	-	-	-	-	-	-	-
13	-	-	40.80	0.20	9.60	0.00	9.20	0.00	0.00	0.00	0.20	0.00	-	-	-	-	-	-	-	-	-	-	-	-
14	-	-	0.00	0.00	1.30	0.00	32.40	0.60	1.30	0.00	0.20	0.40	-	-	-	-	-	-	-	-	1.70	0.00	-	-
15	50.00	81.90	-	-	12.80	0.00	12.40	4.60	13.00	6.30	-	-	-	-	13.87	0.00	-	-	-	-	11.90	3.40	25.00	0.00
16	2.90	3.10	-	-	4.60	0.00	6.70	7.10	4.60	4.00	0.20	1.00	-	-	-	-	3.80	5.00	-	-	-	-	-	-
17	7.50	3.30	17.90	1.40	5.20	0.00	3.70	7.90	5.20	0.40	-	-	18.70	1.70	6.35	0.40	-	-	-	-	-	-	-	-
18	-	-	26.00	2.90	12.70	0.20	12.60	3.80	10.70	1.20	9.10	6.90	-	-	-	-	-	-	-	-	-	-	40.82	0.00
19	5.00	2.70	50.00	7.50	0.60	0.00	1.20	0.40	1.40	2.50	1.70	0.60	-	-	0.77	0.00	-	-	1.30	0.00	-	-	-	-
20	5.20	4.50	34.20	0.00	7.40	0.00	10.30	8.40	1.70	0.00	1.00	2.00	48.30	1.70	-	-	4.00	3.70	-	-	-	-	12.50	0.20
21	-	-	-	-	2.30	0.20	2.40	7.90	5.40	8.40	-	-	30.00	0.00	-	-	-	-	-	-	-	-	-	-
22	36.40	8.10	-	-	13.10	0.40	15.90	19.80	9.90	0.60	10.50	1.20	-	-	-	-	-	-	7.70	2.50	-	-	13.64	0.40
23	-	-	-	-	1.90	0.20	2.10	1.20	1.80	1.20	2.70	20.40	50.00	0.00	-	-	-	-	-	-	-	-	-	-
24	-	-	-	-	2.10	0.00	2.30	0.00	2.70	0.80	-	-	-	-	-	-	-	-	-	-	-	-	-	-
25	-	-	-	-	5.80	0.00	3.30	8.40	2.10	12.60	1.40	15.80	-	-	-	-	-	-	-	-	-	-	0.00	0.00
26	50.00	62.30	-	-	6.30	0.00	4.00	31.90	10.70	13.80	6.60	2.30	-	-	-	-	-	-	-	-	-	-	0.00	0.10
27	-	-	-	-	14.00	0.00	16.50	21.60	9.80	0.40	6.80	0.80	-	-	-	-	-	-	-	-	-	-	-	-
28	-	-	-	-	2.50	0.00	3.70	0.00	1.40	4.80	-	-	50.00	0.00	-	-	-	-	-	-	1.20	0.50	-	-
29	3.30	52.10	-	-	6.20	0.00	5.00	2.70	3.80	0.00	0.60	0.40	-	-	-	-	-	-	-	-	4.20	0.20	10.71	0.40
30	-	-	-	-	21.60	0.00	15.40	14.20	1.20	2.70	-	-	-	-	-	-	-	-	-	-	-	-	-	-
mean (%)	21.10	21.70	30.00	8.10	7.30	0.10	9.90	8.00	4.80	2.90	3.90	4.30	41.20	10.50	7.00	0.10	8.20	3.30	4.20	0.80	4.80	1.00	14.10	0.20
median (%)	18.70	7.10	34.20	0.80	5.20	0.00	5.10	3.80	3.10	1.20	1.90	0.90	48.30	0.00	6.40	0.00	4.00	3.70	3.70	0.00	3.00	0.40	12.50	0.20
avg. time (s)	-	-	-	-	-	-	8.2		2.2		1.2		4.0		-	-	-	-	-	-	-	-	-	-
max. time (s)	-	-	-	-	-	-	10.0		3.3		5.1		6.5		-	-	-	-	-	-	-	-	-	-
memory (MB)	-	-	-	-	-	-	268		126		334		373		-	-	-	-	-	-	-	-	-	-

Currently, the Biolab-ICAO framework is used to evaluate algorithms via an online public competition called Face Image ISO Compliance Verification (FICV), hosted on the FVC-onGoing website (FVC-onGoing, 2006). The FICV is considered the official evaluation tool for ISO/IEC 19794-5 standard and is used by all relevant works presented in the literature or commercial products. The photographic and pose requirements are evaluated individually in terms of the EER. All the results presented in Table 2 and accomplished by the proposed method in this thesis were evaluated using the FICV.

To date, four algorithms have been published on the FVC-onGoing platform: BioTest (BioTest, 2014), BioPass Face (BioPassFace, 2017), id3 (ICAO_compliance, 2016), and ICAO SDK (ICAO SDK, 2021) (see Table 2). Compared with BioLabSDK, such algorithms achieve comparable or even better performance rates for particular requirements. However, all of these algorithms are commercial (Biometrika, 2014; id3, 2016; Seamfix, 2021; Vsoft, 2017). Thus, there is no detailed explanation of their methodology in the scientific literature.

Ferrara et al. (2012a) present a segmentation method for passport images based on a multi-classifier approach. Using the position, color, texture, and histogram classifiers, the algorithm proposed by the authors classifies and post-processes regions in the face image to segment them into four distinct classes: face, hair, clothes, and background. However, the authors applied the segmentation results to analyze only three ICAO requirements (*Hair Across Eyes* (15), *Varied Background* (17), and *Flash Reflection on Skin* (19)), obtaining EERs of 13.87%, 6.35%, and 0.77%, respectively. This method is named as "FerraraSeg" in Table 2. Two other face segmentation methods for passport images have been proposed in the literature: Hirzer et al. (2009) and Subasic et al. (2009). Nonetheless, they did not analyze their results regarding the ISO/IEC 19794-5 standard.

The work of Nguyen et al. (2013) proposed a set of normalized metrics for the quantitative conformance testing of ICAO requirements. Their method comprises three main steps: foreground and background segmentation, face detection, and facial feature extraction. Each step takes advantage of the color, intensity, and edge information to compute the scores for a subset of requirements. These metrics were evaluated over a subset of the FERET (Phillips et al., 1998), GTAV (Tarrés, 2012), and FIePI databases. However, results regarding EER are not presented, which is a common practice of algorithms that assess the ISO/IEC 19794-5 requirements in the literature.

In Parente et al. (2016), methods for individual evaluation of four requirements were proposed. For the *Pixelation* (14) requirement, the Canny edge detection method was applied in the eye region, and the Hough Transform was used to detect horizontal and vertical lines. In the case of *Hair Across Eyes* (15), both eye regions were preprocessed using classic techniques and compared using an XOR operator. For *Veil Over Face* (28), the authors computed a score based on the proportion of skin pixels presented in the lower

region of the face, using the YCrCb color space. Finally, to assess the *Mouth Open* (29) requirement, the method analyzes the teeth and lips based on a color-search approach inside the detected mouth. The results for each requirement can be seen in Table 2.

The requirements *Unnatural Skin Tone* (11), *Shadows Across Face* (22), and *Flash Reflection on Skin* (19) were evaluated in the work of Andrezza et al. (2016). As these requirements are directly related to the skin, the authors developed a custom segmentation method. Each pixel in the face image that falls into a predefined range of the YCrCb color space is marked as skin. To evaluate the *Unnatural Skin Tone* (11), a score was computed based on the proportion of pixels with a natural tone according to the histogram analysis. For *Flash Reflection on Skin* (19), a metric is defined based on the binarized image of the Y channel of YCrCB. Similarly, the Z channel of the XYZ color space was analyzed to evaluate the shadows in the face. In Table 2, the results are shown in the column named "Andrezza et al.".

The work of Borges et al. (2016) analyzed some of the requirements related to the eyes: *Eyes Closed* (16), *Red Eyes* (20), and *Looking Away* (09). First, an appearance-based method was applied to find the eye corners and estimate the iris center based on the Canny edge detector and Hough Circle Transform. Such information is used in the remaining methods. To detect whether the eyes were closed or open, the authors computed a metric based on eye dimensions and the presence of the sclera. For the evaluation of *Red Eyes* (20), custom computations are performed on RGB, HSV, and YCrCB to find the "red" pixels, and three corresponding binarized images are generated. A score was then computed based on the logical operations combining these images. Finally, to assess the *Looking Away* (09) requirement, the authors assumed that the eyes were symmetric and inspected each eye's left and right sides. An OR operation was applied between both sides, and a score was computed based on the proportion of the minimum and maximum sums of white pixels. This method is identified by "Borges et al." in Table 2.

One of the first studies that employed a Deep Learning-based method for ISO/IEC 19794-5 was presented by Ahmadvand and Moin (2018). The authors applied the fine-tuning technique to the VGGFace model (Simonyan and Zisserman, 2014) to train a new model that assesses the *Roll/Pitch/Yaw* (18) requirement. Six different datasets were employed, with a total of 320,000 images, of which only 12,000 were compliant with the ICAO standard. Cross-entropy was used to optimize the model, and the accuracy was chosen to evaluate the final results. The authors reported 95.5% and 97.8% accuracy in the PIE (Sim et al., 2002) and CSIE Robotic (Kung, 2006) databases, respectively. The evaluation results according to the FICV competition are not mentioned in the paper.

The Biolab-ICAO framework is used by Hernandez-Ortega et al. (2019) to train a method based on Convolutional Neural Networks, called FaceQnet. In this case, the framework is applied to label the VGGFace2 dataset (Cao et al., 2018). ResNet-50 (He

et al., 2016) was fine-tuned to return a score representing a numerical quality measure for each input image. The authors analyzed if the score could determine whether an image was suitable for face recognition. However, the authors did not provide results regarding the FICV competition and, thus, they are not included in Table 2.

A similar work to FaceQnet was presented in (Hernandez-Ortega et al., 2022), called FaceQvec. In this paper, the authors developed a set of methods to give a quality score for each requirement of ISO/IEC 19794-5 standard plus white noise and expression. Deep Learning methods were applied for *Mouth Open* (29), *Eyes Closed* (16), *Hair Across Eyes* (15), and *Roll/Pitch/Yaw* (18) requirements. The authors evaluated their method using Receiver Operating Characteristic (ROC) curve and Accuracy over two *ad hoc* datasets labeled by experts with approximately 3500 images. However, as in FaceQnet, the authors do not report results using the FICV benchmark.

Finally, another recent method for evaluating ICAO requirements was published in 2019 (Nourbakhsh et al., 2020). The authors proposed a method based on the Hierarchical Max-pooling (HMAX) model, which consists of a CNN with multiresolution spatial pooling. First, face components were extracted from image patches using the Viola-Jones algorithm (Viola and Jones, 2001). Then, the HMAX model was applied to acquire discriminative signatures. The AR (Martinez, 1998) and PUT (Kasinski et al., 2008) databases were used to train the model for 9 requirements. In Table 2, their results are represented by the "HMAX" column.

3.4 Conclusions

From the analysis of the literature on Multitasking Learning, some relevant aspects can be highlighted. First, MTL is a relatively new field of study when applied to Deep Learning. Thus, most research on this topic is still beginning, and some gaps can be filled (e.g., the high EER of some requirements or the lack of representative datasets). Second, encoder/decoder-focused architectures present advantages and drawbacks. For example, in encoder-focused architectures, the generic representation learned by the encoder may be valuable when shared with task-specific heads. However, they may fail to capture familiar and different aspects among tasks. On the other hand, decoder-focused architectures also share or exchange information during the decoding stage. It can help improve performance; however, such networks usually assume independent tasks or are limited to the number of tasks they can solve.

In this thesis, an encoder-focused architecture is applied since there are requirements in the ISO/IEC 19794-5 standard that share similar characteristics (e.g., requirements related to the eyes or background). Therefore, the features learned by the encoder for specific requirements can be leveraged for task-specific heads. Unlike the encoder-

focused architectures mentioned in this chapter, the proposed method eliminates the need for pre-training (as in Cross-stitch networks) or complex design choices like in NDDR-CNNs. In addition, the proposed ICAONet is easy to scale as MTAN networks. More details will be discussed in Chapter 4.

Regarding Representation Learning, the proposed method employs both generative and discriminative approaches. The generative approach is present in the unsupervised component responsible for creating realistic data samples, whereas it learns a useful representation (called embeddings) by modeling the input distribution. On the other hand, the same representation is also applied to predict different outputs (i.e., requirements and landmark localization). It allows the creation of a proper shared representation built from errors backpropagated from related tasks, as stated in (Zhang et al., 2014). Also, a more robust requirement assessment can be achieved through joint learning with heterogeneous but subtly correlated tasks. Again, a detailed explanation of the proposed architecture is provided in Chapter 4.

Concerning the literature on the ISO/IEC 19794-5 standard, we can conclude that although many studies have addressed this problem for over a decade, it is still an open challenge. For instance, there are still requirements with EERs greater than 10% as the best result among all published studies (e.g., *Looking Away* (09) and *Hair Across Eyes* (15)). Another point is that most of the best results for each requirement presented in Table 2 are dominated by private companies such as Biometrika (2014), id3 (2016), Seamfix (2021), and Vsoft (2017). Therefore, there is no detailed explanation of their methods, and there is a lack of state-of-the-art methods published as open research. In fact, 12 of the best results for all 23 requirements are owned by private companies. Another gap is the low number of single methods that evaluate all requirements. Only three methods (BioLab, BioTest, and BioPass Face) are designed to evaluate all of them. We believe this is due to the absence of public datasets specialized for the ICAO problem. Finally, we can also conclude that Deep Learning can be a helpful approach for improving the current results of methods that address ICAO requirements. One example is the HMAX work, which achieved 0.0% EER in two requirements - *Frame Too Heavy* (25) and *Frame Covering Eyes* (26)- with very low rejection rates.

4 Materials and Methods

This chapter details the materials used in this work and the method proposed in this thesis, called ICAONet. We begin by describing how the dataset have been gathered to train the neural network model and present an analysis of the final database. Later, the proposed method is thoroughly explained, including the main idea behind the architectural design, its components, and the implementation details. The results and other analyses of the proposed work are reserved for Chapter 5.

4.1 Datasets

In this section, the databases used in this study are described. First, the official databases used by FICV to benchmark algorithms that assess the ISO/IEC 19794-5 standard is presented. Subsequently, an expansion of this database used to train the proposed method. Finally, statistics about the database are detailed and discussed.

4.1.1 FICV Dataset

As mentioned in the literature review (see Chapter 3), one of the challenges behind the ISO/IEC 19794-5 standard is the lack of fully labeled datasets for all 23 requirements. According to our previous investigation, the only dataset publicly available for research is the one provided by the FICV competition for its participants. The full dataset comprises 5588 images from 601 subjects gathered from different sources. It was built *ad hoc* using images from public databases, and additional images were manually acquired to cover some missing requirements. The image distribution of the FICV dataset is as follows:

- 1741 images from the AR database (Martinez, 1998) of size 768×576 pixels;
- 1935 images from the FRGC database (Phillips et al., 2005) of sizes 1704×2272 or 1200×1600 pixels;
- 291 images from the PUT database (Kasinski et al., 2008) of size 2048×1536 pixels;
- 804 images artificially generated by applying ink-marked/creased, pixelation, and washed out effects to compliant images from the AR database; and
- 817 newly acquired images of size 1600×1200 pixels.

Moreover, the following information is given for each image:

- the **coordinates of the eye corners** expressed by four pairs of (x, y) coordinates (two for each eye);

- the **compliance to the photographic and pose requirements** expressed as one of three possible values:

  i **compliant**: when a specific requirement is declared compliant for an image, it means that such image is acceptable for that characteristic. In theory, only fully compliant images should be considered for a later face-recognition process. In the ground-truth, compliance is represented by integer 1.

  ii **non-compliant**: represented by the label 0 in the ground-truth, means the opposite of compliant, i.e., for a given image, the specific requirement is not-acceptable; and

  iii **dummy**: used for uncertainty cases. For example, when a person wears glasses with dark-tinted lenses, it is difficult to evaluate whether the eyes are open, even for human experts. It is represented by an integer -1 in the ground-truth.

In total, 310 images are fully compliant (i.e., compliant with all the requirements), and 5278 images have one or more requirements that are non-compliant. A representative subset of 720 images, called *FICV-TEST*, is publicly available to the participants of the FICV competition, and can be used for parameter setup and training. It contains 50 fully compliant images and 670 images not compliant with one or more characteristics. The remaining images are used as the private image set reserved for the benchmark of the submitted algorithms. The private database is referred to as *FICV-1.0* by the FICV. In this thesis, we refer to this as the official dataset of the FICV or FVC-onGoing.

Since some of the images in *FICV-TEST* belong to public datasets and cannot be directly distributed to third parties, only the ground-truth data of each image is provided for the participants. Nevertheless, FVC-onGoing provides a utility tool with instructions to generate the training set. Therefore, participants must first download the images of the public datasets (i.e., AR, FRGC, and PUT) from their respective websites and then use the utility tool to produce the training set. However, using this tool, we obtained only 571 annotated images out of the 720 images. We contacted the person responsible for the FICV competition, but it was informed that the remaining images belong to the set of images acquired internally, and these could not be shared.

Table 3 shows the distribution of images for each requirement in the *FICV-TEST* database. It is possible to see that some requirements do not have non-compliant or dummy images (e.g., *Ink Marked* (10) or *Frame Too Heavy* (25)). Therefore, we increased the *FICV-TEST* dataset using an *ad hoc* approach. Further details are provided in the following subsection.

As indicated on the FICV competition webpage, there are dependencies between specific requirements. For instance, when a person is wearing a veil (i.e., *Veil Over Face*

Table 3: Distribution of compliant (C), non-compliant (NC), and dummy (D) images for each requirement in the *FICV-TEST* dataset.

Req. #	Requirement description	C	NC	D
08	Blurred	515	31	25
09	Looking away	433	32	106
10	Ink marked/creased	571	0	0
11	Unnatural skin tone	407	37	127
12	Too dark/light	503	38	30
13	Washed out	537	33	1
14	Pixelation	541	30	0
15	Hair across eyes	523	12	36
16	Eyes closed	474	31	66
17	Varied Background	335	155	81
18	Roll/pitch/yaw	476	4	91
19	Flash reflection on skin	435	49	87
20	Red eyes	405	26	140
21	Shadows behind head	452	2	117
22	Shadows across face	341	102	128
23	Dark tinted lenses	538	31	2
24	Flash reflection on lenses	478	84	9
25	Frames too heavy	571	0	0
26	Frame covering eyes	477	29	65
27	Hat/cap	538	32	1
28	Veil over face	498	73	0
29	Mouth open	340	115	116
30	Presence of other faces	561	0	10

(28) is non-compliant), evaluating the requirement *Mouth Open* (29) may be impossible. In this case, this requirement would be considered as a dummy. Therefore, we analyzed the *FICV-TEST* dataset to determine the dependencies between non-compliant and dummy requirements. A diagram of these dependencies is shown in Figure 8.

4.1.2 *Ad hoc* Dataset

Usually, Deep Learning-based methods require a large dataset for learning, primarily if training is performed from scratch. Techniques like Transfer Learning or Data Augmentation can be employed for low-size datasets. However, they presume some assumptions. For example, Transfer Learning works well when the original network was trained in a similar domain to the new problem. Similarly, we need to have significant intraclass variance in the samples to improve the results with data augmentation. Moreover, traditional Data Augmentation does not change the distribution of labels in a dataset because random transformations are uniformly applied.

These problems can be found in the *FICV-TEST* dataset. First, the ISO/IEC

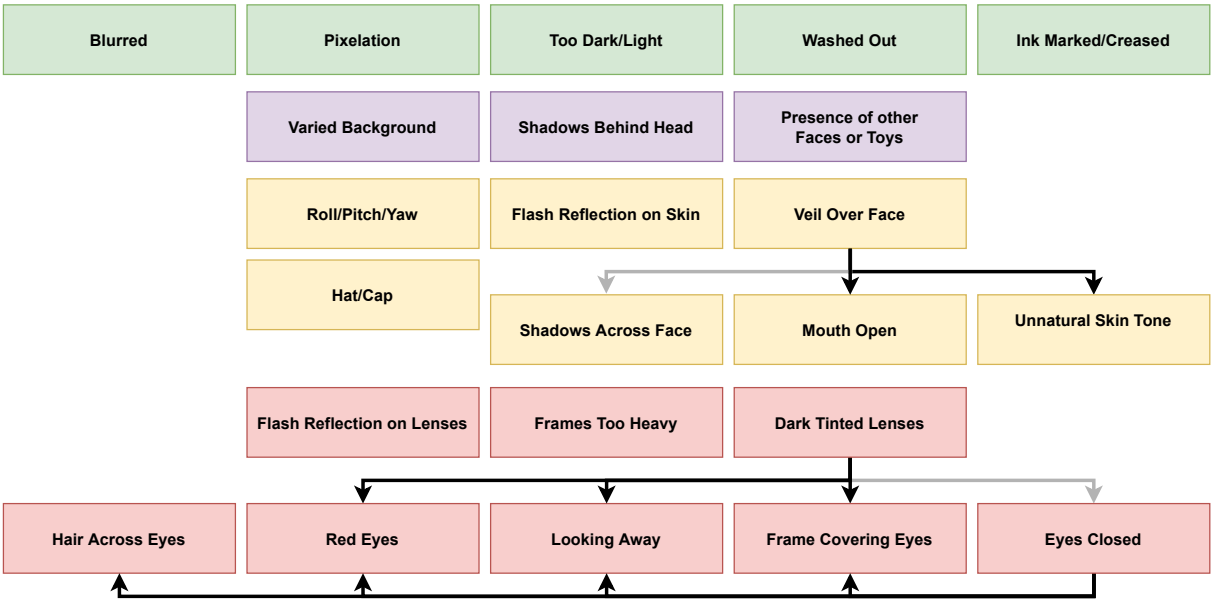


Figure 8: Diagram of dependencies between non-compliant and dummy requirements. An arrow indicates that when the parent requirement is labeled non-compliant, the children's requirements are considered dummy. A light gray arrow indicates that such a relationship is not always true. Source: own elaboration.

19794-5 standard represents a unique type of problem, and thus, Transfer Learning may not be a valid option. Although some face datasets are available for research, Neural Networks are typically trained for face recognition in these datasets. Hence, although some features learned by these networks might be helpful for ICAO, some requirements are not present in these databases (e.g., *Ink Marked* (10)) or may be ignored by the network (e.g., *Varied Background* (17)). Finally, Data Augmentation might not help with the *FICV-TEST* database due to the high imbalance in specific requirements. Also, some transformations applied by Data Augmentation procedures can affect some requirements. For example, random brightness may affect the *Too Dark/Light* (12) requirement, and random rotations can change the angles of the face and disturb *Roll/Pitch/Yaw* (18).

With only 571 images available in the *FICV-TEST* database for all 23 requirements, we decided to increase the dataset by following a procedure similar to that adopted by FICV. Therefore, we manually gathered additional images from the AR, FRGC, and PUT databases. We also included images from the AFW database (Zhu and Ramanan, 2012) and acquired new images according to the ISO/IEC 19794-5 requirements. In this case, we created a dataset of pictures by instructing hundreds of volunteers to reproduce the characteristics of non-compliant photographic requirements (#8–#30 of Table 1). The images were later annotated by three people from our research group, specially trained for this task. As in (Nowak and Ruger, 2010), we provided an annotation tool in which the task of annotators was to select the compliance level of each requirement: compliant

(C), non-compliant (NC) or Dummy (D). It was developed with Pybossa[1] and provided as a web application tool. To produce the final labels, we followed an approach similar to that used by (Chang et al., 2017). Thus, expert annotators compared and discussed conflicting labels in an iterative and collaborative workflow. In total, we ended up with a training set of 5763 images. The image distribution per dataset is defined as follows:

- 22 images from AFW database of sizes from 362×362 to 1984×1984 pixels;
- 1368 images from AR database;
- 50 images from PUT database;
- 1772 images from FRGC database; and
- 2551 newly acquired images of sizes from 976×1301 to 4608×3456 pixels

Samples from each database can be seen in Figure 9. Our dataset has 177 fully compliant images and 5586 images with one or more non-compliant requirements. One crucial detail to notice is that, although the *ad hoc* dataset has the label dummy in the ground-truth, we decided to merge dummy and non-compliant labels. This was done for two reasons. First, according to the FICV protocol, only compliant and non-compliant images are considered for the benchmark of each requirement (see Section 2.8 of Chapter 2). Furthermore, the dataset imbalance is diminished by combining these two labels, and the intra-class variance of non-compliant images is increased. Therefore, this can help the network define a better decision boundary between compliant and non-compliant requirements and improve performance. Our results corroborated this hypothesis. However, this decision aims at the competition of the FICV. In other contexts, the dummy information may be relevant (e.g., eye detection in case of dark glasses may be avoidable and lead to wrong analysis). Finally, the distribution of labels per requirement can be seen in Table 4.

Since the FICV competition also evaluates the capacity of the algorithm to locate the eye's centers, we also had to label them in our *ad hoc* dataset. It was accomplished using the Computer Vision library called Dlib (King, 2009), which contains a built-in shape predictor for 68 face landmarks. When submitted to FICV competition, the detection of eye landmarks by Dlib achieved $d_{eye} \in [0; 0.1[= 94.13\%$ in the *FICV-TEST* dataset (see Equation 40). Thus, it has a high level of effectiveness and defines an upper bound for our landmark localization predictor.

To better understand the *ad hoc* dataset, we analyzed the images and labels. Such analyses are better described in the following subsection.

[1]https://pybossa.com

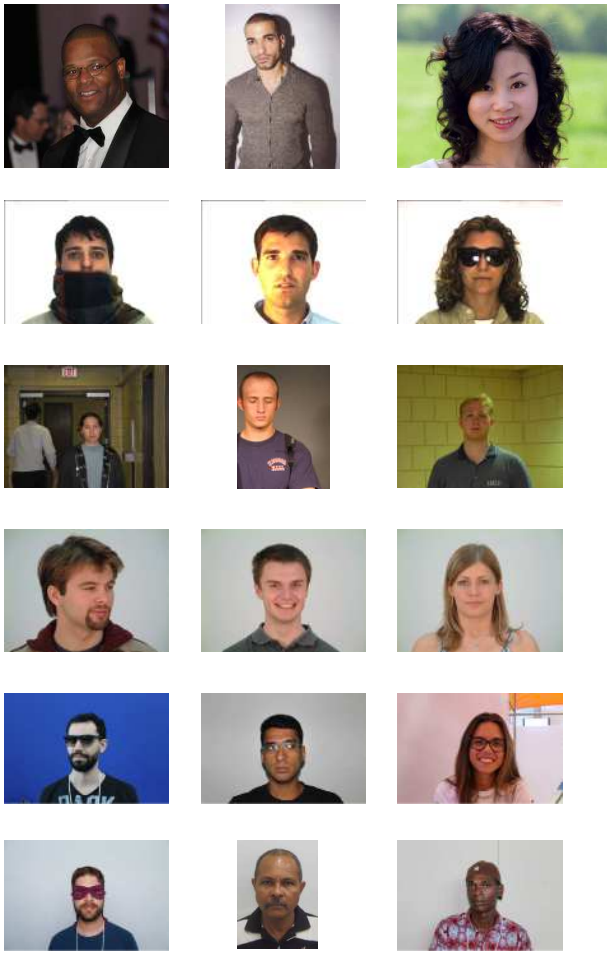


Figure 9: Samples of images in our dataset. The first four rows contain samples of the AFW (Zhu and Ramanan, 2012), AR (Martinez, 1998), FRGC (Phillips et al., 2005), and PUT (Kasinski et al., 2008) databases, respectively. The last two rows are samples of images we acquired for this work.

4.1.3 *Ad hoc* Dataset Analysis

Figure 10 shows the number of images with a certain number of compliant/non-compliant labels in the *ad hoc* dataset. As mentioned, there are 177 fully compliant images (i.e., the number of non-compliant labels equals zero). In addition, most images have 2 to 6 non-compliant requirements, but there are images with up to 14 non-compliant requirements.

Figure 11 contains the label distribution by requirement presented earlier in Table 4. As can be seen, the requirements *Ink Marked* (10), *Washed Out* (13), *Frame Too Heavy* (25), and *Presence of Other Faces or Toys too Close to Face* (30) are the most unbalanced. Also, the *Unnatural Skin Tone* (11) is the only requirement in which the number of non-compliant samples exceeds the compliant.

Figure 12 shows the co-occurrence between non-compliant requirements. In other words, it measures the number of images with two different non-compliant requirements

Table 4: Distribution of compliant (C) and non-compliant (NC) images for each requirement in the dataset. The last column indicates the proportion of NC images for the corresponding requirement.

Req. #	Requirement description	C	NC	NC (%)
08	Blurred	4858	905	15.7%
09	Looking away	3946	1817	31.5%
10	Ink marked/creased	5742	21	0.3%
11	Unnatural skin tone	2540	3223	55.9%
12	Too dark/light	5307	456	7.9%
13	Washed out	5690	73	1.3%
14	Pixelation	5366	397	6.9%
15	Hair across eyes	4252	1511	26.2%
16	Eyes closed	4440	1323	22.9%
17	Varied Background	3248	2515	43.6%
18	Roll/pitch/yaw	4347	1416	24.6%
19	Flash reflection on skin	3143	2620	45.5%
20	Red eyes	4531	1232	21.4%
21	Shadows behind head	3866	1897	32.9%
22	Shadows across face	4621	1142	19.8%
23	Dark tinted lenses	5121	642	11.1%
24	Flash reflection on lenses	4584	1179	20.5%
25	Frames too heavy	5746	17	0.3%
26	Frame covering eyes	4084	1679	29.1%
27	Hat/cap	4914	849	14.7%
28	Veil over face	5399	364	6.3%
29	Mouth open	4231	1532	26.6%
30	Presence of other faces	5693	70	1.2%

occurring together. As expected, there is a strong co-occurence between eyes-related characteristics, such as *Looking Away* (09), *Hair Across Eyes* (15), *Eyes Closed* (16), *Red Eyes* (20), *Dark Tinted Lenses* (23), and *Frame Covering Eyes* (26). This is mainly caused by the dummy requirements that were converted to non-compliant. Moreover, considerable co-occurrences can be observed between the requirements associated with skin, for instance, *Unnatural Skin Tone* (11) and *Flash Reflection on Skin* (19).

Based on our analysis, we arrived at the following conclusions. Firstly, since the dimensions of the images are different across the dataset, an approach to normalize images is required. It must avoid undesired normalization effects (like blur and pixelation) whenever possible and consider the trade-off between the input image quality and processing speed by the network. Regarding labels, the proposed method must consider dataset unbalancing, and the most unbalanced requirements may require special attention. Furthermore, the correlation between certain requirements can be leveraged by a mechanism that shares features, like Autoencoders.

In the next section, we describe in detail the proposed method, called ICAONet.

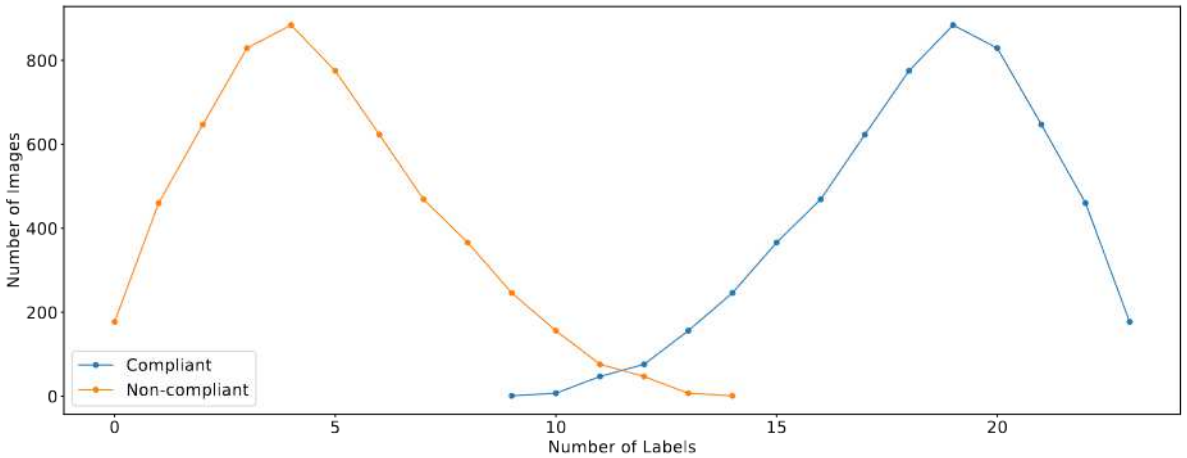


Figure 10: Number of images according to the count of compliant/non-compliant labels. Source: own elaboration.

This includes the architecture, training process, and implementation details.

4.2 ICAONet

ICAONet is a Deep Neural Network developed to address the ISO/IEC 19794-5 standard. The architecture of ICAONet is mainly based on Autoencoders. However, it has also been extended to apply a multi-and-collaborative learning approach. Further details regarding the proposed method are described in the remainder of this section.

4.2.1 Preprocessing

Since the *ad hoc* dataset images have an extensive range of size dimensions (from 362×362 to 4608×3456 pixels), a preprocessing step was required to standardize the input image to ICAONet. Figure 13 summarizes the preprocessing method applied in this work. A detailed explanation is provided below.

The first step in our preprocessing method is face detection. We use a single-shot multi-box detector based on MobileNet (Yeephycho, 2017) and trained on the WIDER FACE dataset (Yang et al., 2016). This face detector was chosen because it balances speed and accuracy fairly. According to our benchmarks, 98.99 % of all faces were detected, with an average processing time of 1.6s per image in the CPU. Moreover, this detector is compatible with TensorFlow 1.X versions used in the proposed method.

Since the bounding box of the detected face is limited to the face region (from forehead to chin), we crop a squared region 1.5× larger than the detected face to include background and other relevant information to assess the requirements. The padded regions are filled with zeros because they generate fewer undesired artifacts than methods

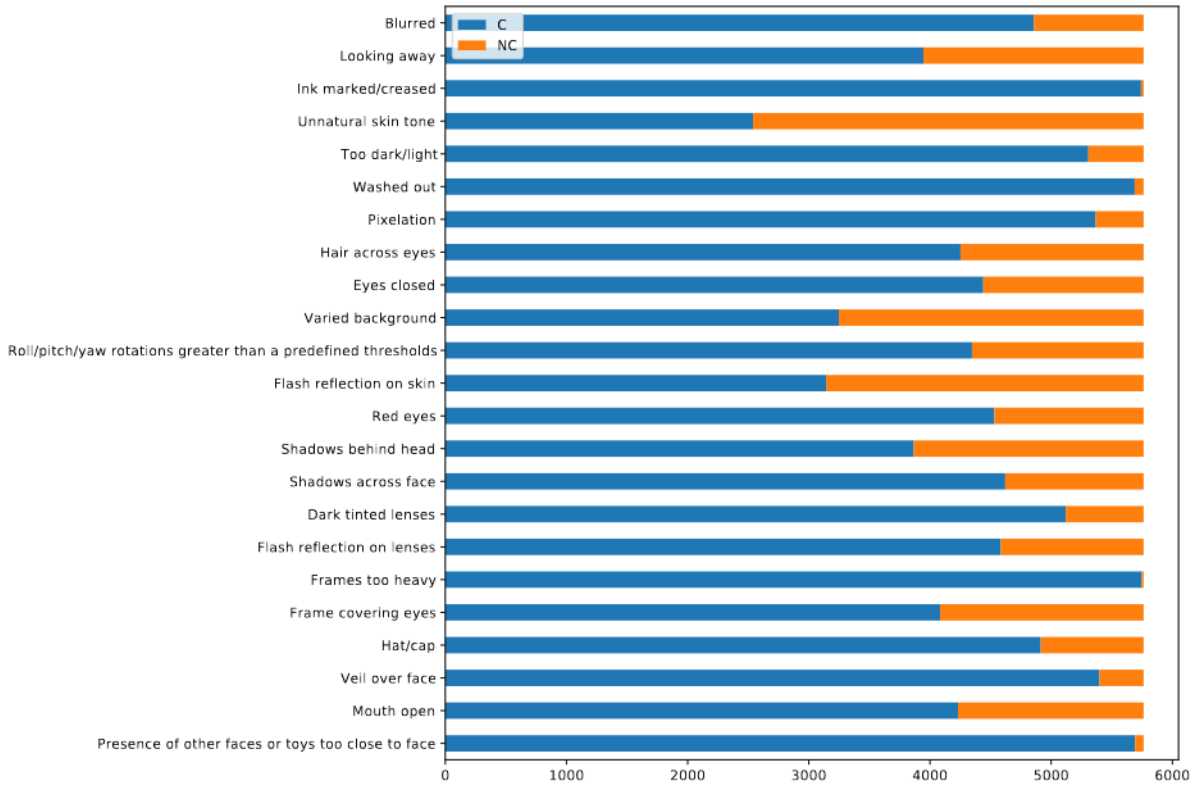


Figure 11: Labels distribution per requirement. Source: own elaboration.

such as border reflection, replication, or wrapping. We do not apply any image normalization like illumination correction or rotation, as it can affect requirements evaluation.

Finally, the cropped image is resized to 160×160 pixels using the `INTER_AREA` method of OpenCV[2], since it is the recommended method for image decimation. Then, all pixel intensities are normalized to the real-valued [0...1] range before being fed to ICAONet. Again, the size of 160×160 pixels was chosen, considering the trade-off between computational costs and results. More details are provided in Chapter 5.

4.2.2 Architecture

The overall architecture of ICAONet can be seen in Figure 14. The architecture is composed of an Autoencoder combined with multiple dense network branches. While the Autoencoder is employed for unsupervised learning of a highly discriminative embedding space, the dense layers perform multi-label classification and landmark localization.

The main idea behind the proposed architecture is to employ a multi and collaborative learning approach for ISO/IEC 19794-5 requirements. This is multi-learning (also called multitasking) because the network solves both regression (image reconstruction and landmark prediction), multi-label classification (compliance prediction for each

[2] `https://docs.opencv.org/3.4/da/d54/group__imgproc__transform.html`

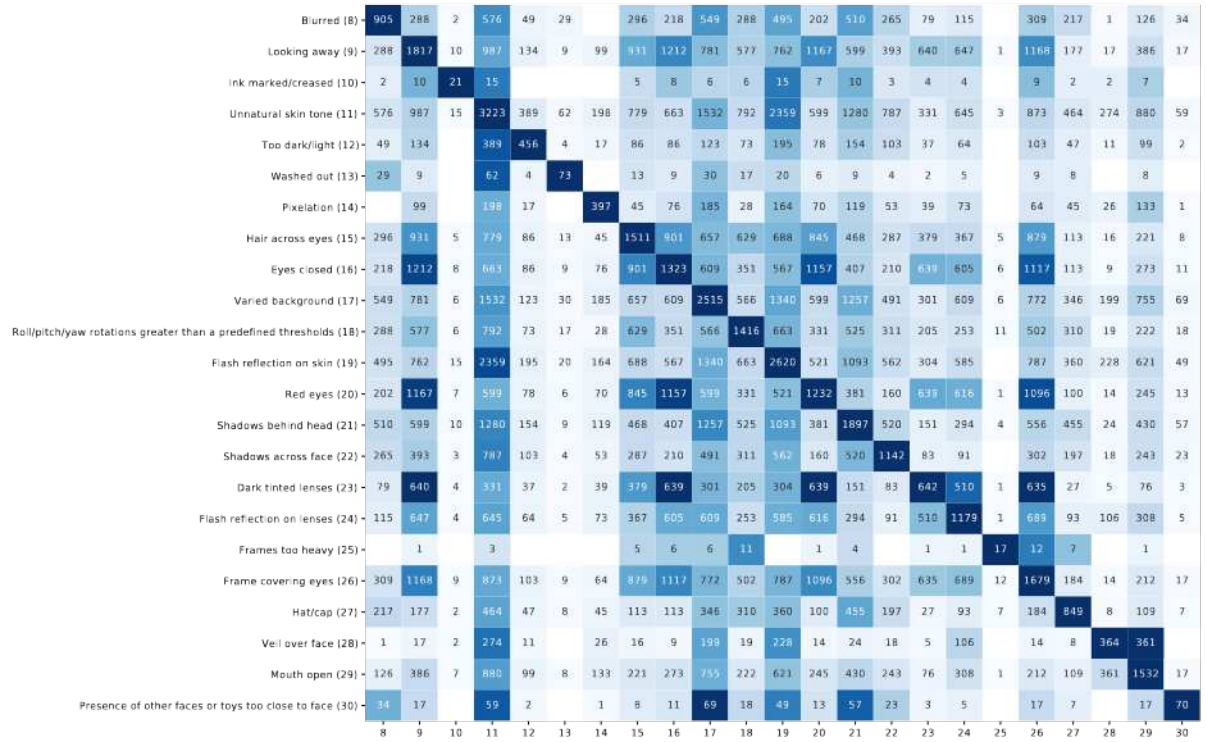


Figure 12: Co-occurrence between non-compliant requirements. The value indicates the number of images with both requirements non-compliant. The blue scale is normalized by row (the higher the darker). Source: own elaboration.

requirement), and binary classification (pixelation assessment) tasks simultaneously. In contrast to Generative Adversarial Networks (GANs), the learning process must also be collaborative since all tasks are optimized together in training time. Hence, all branches must collaborate to learn an appropriate representation for solving all tasks. Additionally, it can be possible to assign different weights for each task to determine their importance during the optimization step (more details are provided later).

The ICAONet has three main components: (i) a **Shared Network** to compute shared embeddings, (ii) an **Unsupervised Branch** to perform image reconstruction, and (iii) **Supervised Branches** for requirement assessment and landmark localization. Such components are detailed as follows.

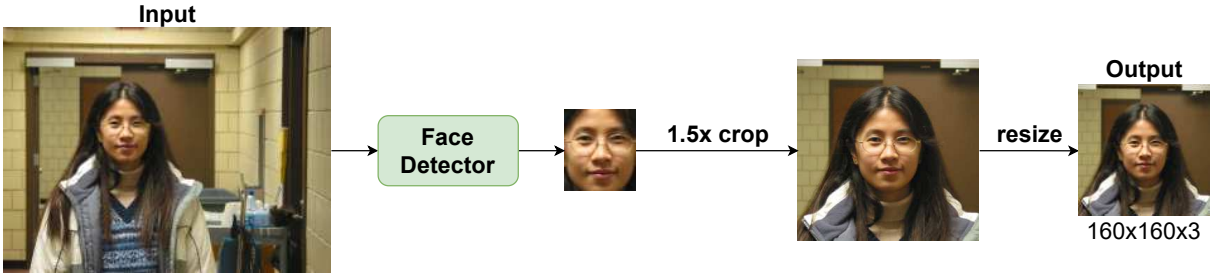


Figure 13: Preprocessing step of ICAONet. Source: own elaboration.

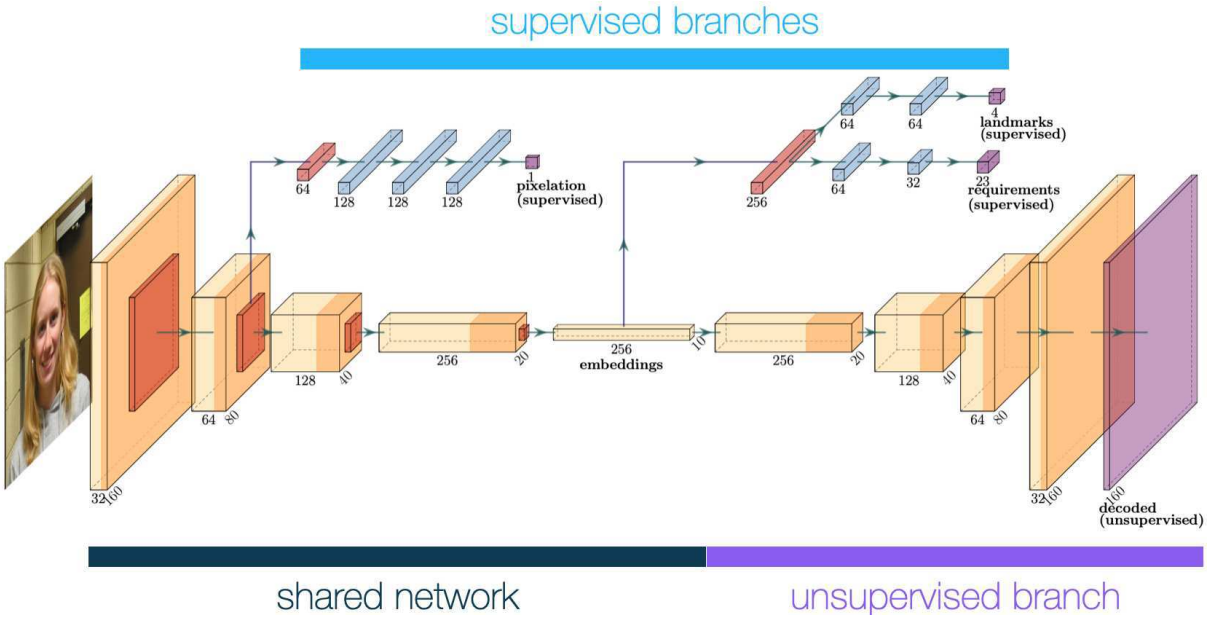


Figure 14: Architecture of ICAONet. Source: own elaboration.

4.2.2.1 Shared Network

The initial part of the network is responsible for learning the embeddings shared by unsupervised and supervised branches. These embeddings can be interpreted as a proper input representation and can be defined as an encoder function $h = f(x)$. Besides being used for dimensionality reduction, the shared embeddings are also used for feature learning in our network.

The architecture to compute the shared embeddings is based on the encoder component of Undercomplete Convolutional Autoencoder networks (Goodfellow et al., 2016, p. 500). The shared network receives a preprocessed input image of 160x160 pixels in the BGR color space. Details of the preprocessing are given in Section 4.2.1. The first four convolutional layers are composed of 3x3 filters with batch normalization and ReLU activation. A 2D max-pooling layer is applied for dimensionality reduction after each layer. The fifth layer is also composed of 3x3 convolutions with batch normalization. However, *tanh* is used as a non-linear activation function instead of ReLU to normalize the embedding values between -1 and 1. In our experiments, *tanh* performed slightly better than ReLU for encoded representations. The output of this layer stores the embeddings shared by other branches. The embeddings are a 256-dimensional vector.

4.2.2.2 Unsupervised Branch

This branch represents the decoder of an Autoencoder network. It is responsible for decoding the shared embeddings back to a lossy representation of the original input. Mathematically, this branch is responsible for learning the decoder function $\hat{x} = g(h)$,

where $\hat{x}$ represents the reconstructed image. In backpropagation step, it also assists the creation of useful embeddings.

The architecture reflects the first four convolutional layers of the shared embeddings network. However, the Max-pooling layers are replaced with 2D Transposed Convolution layers. Furthermore, the last layer activation function is sigmoid since the input image pixels are normalized into the [0...1] range. The shared embedding network and the unsupervised branch produce a fully Convolutional Autoencoder Network (Goodfellow et al., 2016).

The unsupervised branch uses the Mean Squared Error (MSE) to measure the image reconstruction task ($\mathcal{L}_1$), as defined by Equation 44:

$$\mathcal{L}_1 = \frac{1}{N} \sum_{h}^{H} \sum_{w}^{W} (I_{h,w} - \hat{I}_{h,w})^2 \tag{44}$$

where H and W are the image dimensions; $N = H \times W$ represents the number of pixels; and I and $\hat{I}$ are the input and the reconstructed images, respectively. Since the same input image I is used in the reconstruction task as the ground-truth, this task is considered unsupervised (sometimes called semi-supervised). During training, the unsupervised branch tries to minimize the squared difference between the input image I and the reconstructed image $\hat{I}$.

4.2.2.3 Supervised Branches

There are three supervised branches in ICAONet: (i) **requirements**, (ii) **eye-landmark localization**, and (iii) **pixelation**. The requirements and landmark localization branches take the shared embeddings as input and apply a fully connected network to perform multi-label classification and regression, respectively. On the other hand, the pixelation branch leverages information from shallow layers to use low-level features in the evaluation of *Pixelation* (14) requirement.

The first layer of the requirement branch performs GlobalAveragePooling on the shared embeddings. It transforms the 4-D dimensional vector of the embeddings into the 2-D dimensions required by dense networks. We chose GlobalAveragePooling layers instead of MaxPooling layers because they are known to perform better in practice, as stated in Zhou et al. (2016). Then, there are two consecutive dense layers with 64 and 32 neurons, respectively. Dropout layers are included between each layer to prevent overfitting. Finally, the output layer contains 23 neurons with sigmoid activation and the Multi-label Cross-Entropy is used as the loss function for the requirements branch, defined by Equation 45:

$$\mathcal{L}_2 = -\frac{1}{M}\sum_{i}^{M} y_i \cdot log(\hat{y}_i) \tag{45}$$

where M represents the number of requirements; y_i is the ground-truth for each requirement (0: non-compliant, 1: compliant); and $\hat{y}_i$ is the predicted score for each requirement. Therefore, each neuron of the final layer outputs a normalized score between 0 and 1 for each corresponding requirement. In practice, the output of requirements branch can be considered as the likelihood that a given input image is compliant with each requirement of the ISO/IEC 19794-5 standard.

The landmark localization branch is almost identical to the requirement branch. It also shares the GlobalAveragePooling, but the final layer has 4 neurons representing each eye's center (x, y). The activation function of the output layer is also sigmoid, since the eye positions are normalized to the input size. However, in this case, we are performing a regression task and the Wing loss ($\mathcal{L}_3$) is applied as the loss function (see Section 2.6.8).

The pixelation branch also starts with GlobalAveragePooling applied after the second Max-Pooling layer of the Shared Network. Then, there are 3 consecutive layers of Dropout and fully connected layers with 64, 128, and 128 neurons, respectively. Finally, the last layer has a single neuron with sigmoid activation. It outputs the likelihood of the input image being compliant/non-compliant regarding the *Pixelation* (14) requirement using the regular Cross-Entropy loss ($\mathcal{L}_4$), defined by Equation 46:

$$\mathcal{L}_4 = -\frac{1}{M}\sum_{i}^{M} y_i \cdot log(\hat{y}_i) + (1 - y_i) * log(1 - \hat{y}_i) \tag{46}$$

The pixelation branch deserves special considerations. First, unlike the other branches, it is the only branch that does not use the embeddings learned by the Shared Network as inputs. This branch begins after the second Max-Pooling layer of the Shared Network. Therefore, the low-level features present in these shallow layers are used to assess this specific requirement. Moreover, this is a particular case of Multitasking Learning since there is a task-specific head decoupled from the joint representation. Finally, the pixelation branch is trained with face patches of the original image, i.e., before preprocessing. In this way, the image resolution is preserved, and it avoids undesired pixelation effects artificially generated by the resize operation of the preprocessing step.

Since the network has supervised and unsupervised branches, and they are optimized using different loss functions, the final loss function $\mathcal{L}(I, \hat{I}, y, \hat{y})$ is defined by Equation 47:

$$\mathcal{L}(I, \hat{I}, y, \hat{y}) = \lambda_1\mathcal{L}_1 + \lambda_2\mathcal{L}_2 + \lambda_3\mathcal{L}_3 + \lambda_4\mathcal{L}_4 \tag{47}$$

where $\lambda_{1...4}$ are hyperparameters to control the trade-off between each loss function. Table 5 summarizes the loss function used by all ICAONet branches.

Table 5: Summary of task type performed for each branch of ICAONet and their corresponding output and loss function.

Branch	Task Type	Output Dim.	Output Interpretation	Loss Function
Unsupervised	Image Reconstruction	160x160x3	Decoded Image	Equation 44
Requirements	Multilabel Classification	23	Compliance Likelihood for each requirement	Equation 45
Landmarks	Regression	4	(x, y) coordinates for each eye's center	Equation 39
Pixelation	Binary Classification	1	Compliance Likelihood for Pixelation requirement	Equation 46

There are three important points to be mentioned about Equation 47. First, each loss function can output values within different ranges, which can significantly affect the training process of the neural network. Although it can be controlled by $\lambda_{1...4}$ parameters, a more advanced normalization process would be required for arbitrary ranges. Second, each individual loss ($\mathcal{L}_{1...4}$) is known to be continuous and differentiable. In the case of Equation 47, this is still true due to the sum rule of differentiable functions (Strang, 2020, p. 250) and because all $\mathcal{L}_{1...4}$ functions have the same domain in our network (normalized input image). However, if the domain changes in the future (e.g., because of architectural changes or use of other loss functions with distinct domains), it may not be valid anymore. Finally, all branches are individually optimized up to the point where they merge during the backpropagation step. When this occurs, the gradients of each loss function ($\mathcal{L}_{1...4}$) are summed up. Therefore, all of these aspects may pose challenges in balancing the optimization process, potentially resulting in slower convergence, vanishing or exploding gradients, and difficulties in finding an optimal solution. Careful consideration and experimentation are crucial to mitigate such issues and enhance the overall effectiveness of ICAONet.

4.2.3 Training

To train ICAONet, the *ad hoc* dataset was split into training and validation sets only. We consider the private dataset of the FICV competition (*FICV-1.0*) as the test set, and the benchmark results are reported as our final results. Both the training and validation sets were randomly divided using a stratified multi-label approach. Thus, the compliant and non-compliant proportions listed in Table 4 are preserved. Approximately

10% of the dataset (580 images) was used as the validation set, and the remaining images were selected for network training.

The supervised branches of the ICAONet were trained using distinct settings and timestamps. First, the unsupervised and requirement branches were trained together. Then, before training the pixelation and landmarks branches, the network was frozen to avoid changes in the requirement results when submitted to FICV. Later, the landmark localization branch was trained on a subset of the *ad hoc* dataset without the *Dark Tinted Lenses* (23) images. According to our experiments, removing these images could improve our predictions' performance because the landmarks labeled by the shape predictor of Dlib were incorrect in this case (see Section 4.1.2). Finally, the pixelation branch was trained after the analysis to improve the results of this specific requirement. Such a branch was trained on approximately 9000 patches extracted from the original images (before preprocessing). Therefore, we avoided the resizing operation performed by the preprocessing step, which can create artificial pixelation artifacts.

Regarding the architecture, since most of the ICAONet structure is based on undercomplete Autoencoders, there was no need to employ many regularization techniques during training. However, batch normalization was still applied before activation functions in the Convolutional layers and Dropouts in some dense layers of the supervised branches. The Early Stopping technique was also employed to prevent overfitting. All metrics described in Section 2.6 of Chapter 2 are evaluated at the end of each epoch and will be reported in Chapter 5.

The network is written in Python using Keras[3] framework (v2.3.1) with TensorFlow (v1.13.1) backend. The Mlflow[4] library (v1.7.0) is used for experiments tracking, comparison and logging of hyperparameters, metrics, and artifacts. The source code and the trained network can be found in Github[5]. Furthermore, the experiments were conducted on a Windows 10 machine with Intel® Core™ i5-8300H of 8th generation, 16 GB of DDR4 2666 MHz RAM, SSD of 512 GB, and NVIDIA® GeForce® GTX 1050 with 4GB of RAM.

One last important detail about ICAONet training is that special care was taken during experimentation. For example, the random seeds of Python, its `random` module[6], and Numpy[7] and TensorFlow libraries have been set to be the same for all experiments. It assures reproducibility and ensures that the best results of the current experiments are not achieved by chance.

[3] `https://keras.io`
[4] `https://www.mlflow.org`
[5] `https://github.com/arnaldog12/icaonet`
[6] `https://docs.python.org/3/library/random.html`
[7] `https://numpy.org`

4.2.4 Parameters and Hyperparameters

The base architecture of ICAONet– composed of the three main components described earlier (shared network, supervised branches, and unsupervised branch) – contains 2,043,231 parameters. Of these parameters, 2,040,799 are trainable, and the remaining 2,432 are non-trainable parameters related to the mean and variance computed by batch norm layers. Regarding size, ICAONet occupies 7.92 MB in the disk when stored as a `hdf5`[8] file.

There are two important details to mention about the proposed architecture. The first refers to the unsupervised branch, which is used only during training. Once the network is trained, the unsupervised branch is detached from the model because the reconstruction task is not valuable for the ICAO assessment and can be ignored. In this case, the number of parameters is reduced to 1,063,068. Furthermore, after model *freezing*[9], the model size is reduced to only 4.06 MB on the disk. It helps to speed up the running time considerably. Secondly, the score for *Pixelation* (14) is outputted from both the pixelation and requirement branches during training. In fact, the pixelation output is kept in the requirements branch only for convenience. For the FICV competition, only the score from the pixelation branch is considered.

The ICAONet was trained using a batch size of 32. The pixelation branch was trained for 500 epochs, whereas the others were trained for 100. All layers were randomly initialized by Xavier initialization (Glorot and Bengio, 2010). To prevent overfitting and improve generalization, we used Early Stopping with 30 epochs of patience for the requirement and landmark branches and 50 epochs for the pixelation branch. Also, the F-beta, MCC, and Wing loss are used as monitoring metrics of the requirement, pixelation, and landmark localization branches, respectively. In layers where Dropout is applied, we keep 50% of neurons. The Adaptive Momentum Estimation (Adam) is applied as the optimizer with learning rates $\alpha = 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-7}$. For the final loss function (Equation 47), we used $\lambda_1 = 2.0$ (unsupervised), $\lambda_2 = 1.0$ (requirements), $\lambda_3 = 1.0$ (landmarks), and $\lambda_4 = 1.0$ (pixelation). Regarding eye location accuracy, the Wing loss (Equation 39) is applied with $w = 10$ and $\epsilon = 2$. All these parameters were chosen after a systematic search. However, it is conceivable that searching for better hyperparameters may further improve the performance of ICAONet.

[8] `https://www.hdfgroup.org/solutions/hdf5/`

[9] Freezing is a typical operation in Keras/TensorFlow models. It removes unnecessary data for prediction in the model file, for example, the optimizer, metrics, metadata, and gradients. It may not be confused with layer freezing.

5 Results and Discussions

This chapter presents and discusses the results achieved by the proposed method. First, the ICAONet results in the training stage are presented, including the computed loss and metrics. Secondly, the performance of the proposed method is discussed in the context of the FICV competition. Then, ICAONet is compared against the best results achieved by the methods shown in Chapter 3. Finally, some visualizations performed over the proposed method are analyzed to help understand its predictions.

5.1 Training Performance

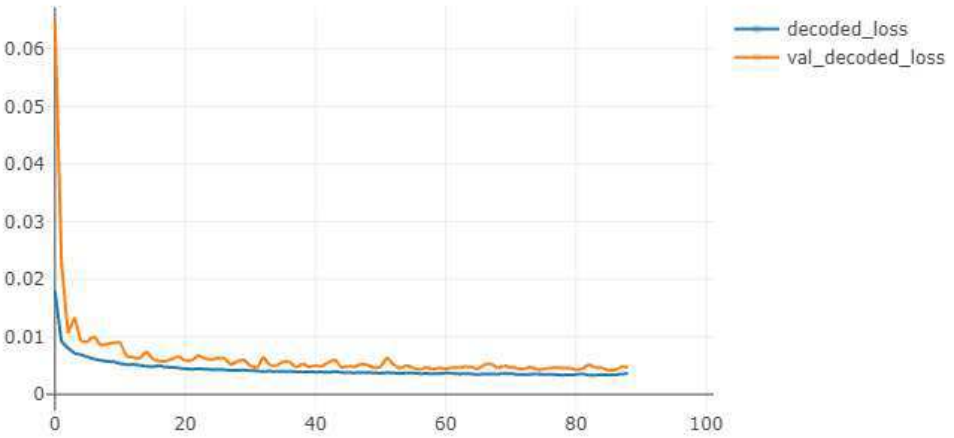
In this subsection, we discuss the training performance regarding the ISO/IEC 19794-5 requirements followed by the results of eye landmarks detection. We focus on analyzing the loss charts and values of metrics described in Chapter 2.6.

5.1.1 Requirements

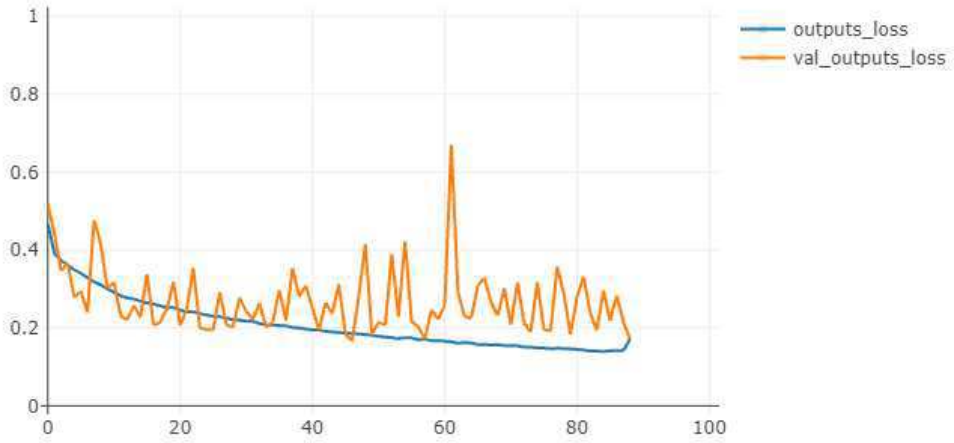
We begin by presenting the results of ICAONet during the training phase. The Figure 15 shows the loss in training and validation sets for unsupervised and requirement branches individually. As can be seen, the loss of the unsupervised branch was lower and smoother than the requirements' during the entire training. This behavior was expected since (i) the reconstruction task performed by the unsupervised branch is theoretically more straightforward than the multi-label classification carried out by the requirement branch, and (ii) the unsupervised loss has a higher weight during training (see section 4.2.4 and Equation 47).

A more detailed analysis of our loss curves shows that ICAONet presented an appropriate balance between bias and variance. In the unsupervised branch, the training and validation losses are approximately zero and similar. Therefore, the network achieved notable performance in reconstructing the input image from the shared embeddings. Similarly, even though the losses from the requirements branch are more significant than those from the unsupervised branch, they are still close to zero. However, we can notice a noisier loss curve in the validation set. Although such loss curves present a higher variance, they keep near the training curve through the epochs. Furthermore, overfitting was prevented by the regularization techniques employed during training, i.e., batch normalization, dropout layers, Early Stopping, and the architecture itself (as detailed in Section 4.2.4).

The final metrics of ICAONet are given in Table 6. These metrics are obtained in the validation set at the 57^{th} epoch, the best epoch before the Early Stop. Some meaningful insights about the proposed method can be observed. First, it is better to predict



(a) unsupervised



(b) supervised

Figure 15: Loss of training and validation sets for (a) unsupervised and (b) supervised branches. Source: own elaboration.

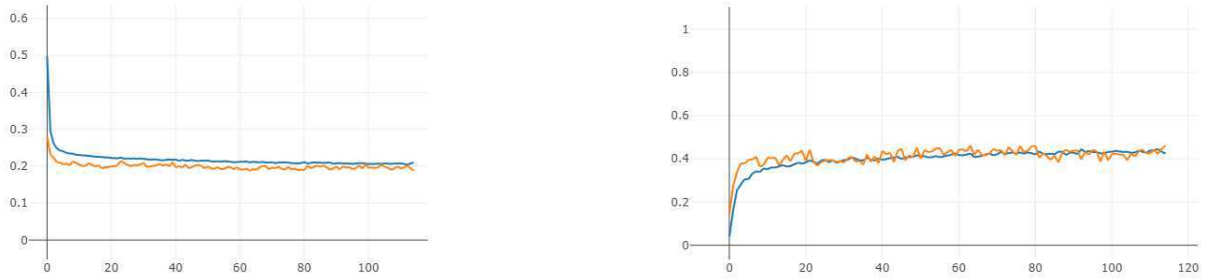
the compliant requirements (positive class) than the non-compliant since the Precision and Recall are higher than the NPV and Specificity. Probably, this is influenced by the unbalanced dataset. Moreover, the False Positive predictions (type-I error) are a more critical problem of ICAONet because both Recall and NPV are greater than Precision and Specificity, respectively. In particular, the Specificity indicates that a reasonable number of non-compliant requirements are assigned as compliant. On the other hand, the proposed method achieved considerably high F-measure and F-beta values, showing a fair balance between Precision and Recall. Finally, the notable MCC score (82.78) indicates that the ICAONet was able to learn valuable patterns for both compliant/non-compliant requirements even with the unbalancing present in the *ad hoc* database.

Table 6: Global metrics of ICAONet achieved in the best training epoch.

Metric	Value (%)
Accuracy	94.27
Precision	94.53
Recall	97.89
F-measure	96.15
F-beta	97.14
NPV	91.67
Specificity	81.69
MCC	82.78

5.1.2 Eye Location Accuracy

In contrast to intuition, the detection of eye landmarks revealed to be a harder task than the assessment of ISO/IEC 19794-5 requirements. Figure 16 presents the Wing loss and eye location accuracy ($d_{eye} \in [0; 0.1[$) in the training and validation sets. In contrast to the supervised branch for requirements, we noticed the presence of bias in the loss graph during training (the loss is far from zero), indicating that the network could not learn useful patterns and detect eye landmarks accurately. Such behavior also occurred in the validation set. These results are also reflected in the eye location accuracy metric, which reached a maximum value of 46.18% in the *ad hoc* dataset.



(a) Wing Loss

(b) Eye Localization Accuracy

Figure 16: Results of eye localization for training (blue) and validation (orange) sets: (a) wing loss and (b) $d_{eye} \in [0; 0.1[$. Source: own elaboration.

A sample of images with landmarks predicted by ICAONet is shown in Figure 17. The first two rows contain arbitrary examples of the most precise detections (i.e., $d_{eye} \in [0; 0.1[$). We can observe that ICAONet can perform accurate detections for frontal face images even with the presence of requirements that could potentially harm the accurate localization of the landmarks. For example, we can see the presence of images with *Frame Covering Eyes* (26) and *Too Dark/Light* (12). On the other hand, in the last two rows, there are samples of the worst detections, i.e., $d_{eye} \geq 0.3$. In this case, a pattern of highly rotated facial images (*Roll/Pitch/Yaw* (18)) is perceptible. Furthermore, other

requirements can be noticed (e.g., *Blurred* (08) and *Frame Too Heavy* (25)).

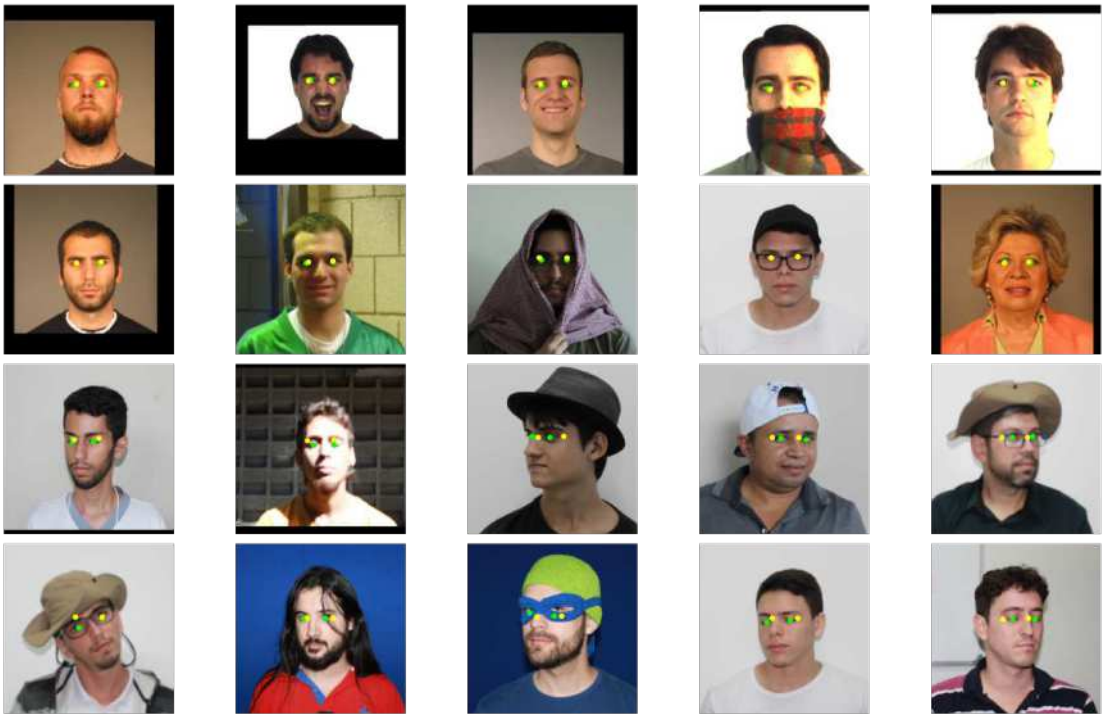


Figure 17: Results of eyes landmarks detection by ICAONet. The first two rows contain images with $d_{eye} \in [0; 0.1[$, and the last rows are for $d_{eye} \geq 0.3$. The ground-truth annotations are in green, while network predictions are in yellow.

We analyzed the network predictions for eye landmarks to understand the bias observed during training. In Figure 18, we can see a heatmap of landmarks in the validation set. As can be noticed, there are two noticeable clusters for each eye, indicating that the networks essentially predict landmarks over the same regions regardless of the input image. We suppose this behavior is caused mainly by the preprocessing step, which centers the face in the input image.

We presume that the low performance of eye landmark localization can be explained by one or more of the following reasons:

i **Preprocessing**: our preprocessing method centralizes the face by creating a region around it (see Section 4.2.1). Therefore, the network can be induced to predict the mean of the landmark positions to minimize the loss function. However, we attempted to apply different levels of image augmentation in other experiments. Although the predictions were more distributed over the image, there were no significant changes in loss and location accuracy.

ii **Dataset**: our *ad hoc* image dataset mainly comprises images with frontal faces with slight variation in face poses and alignment. In addition, the number of images (ap-

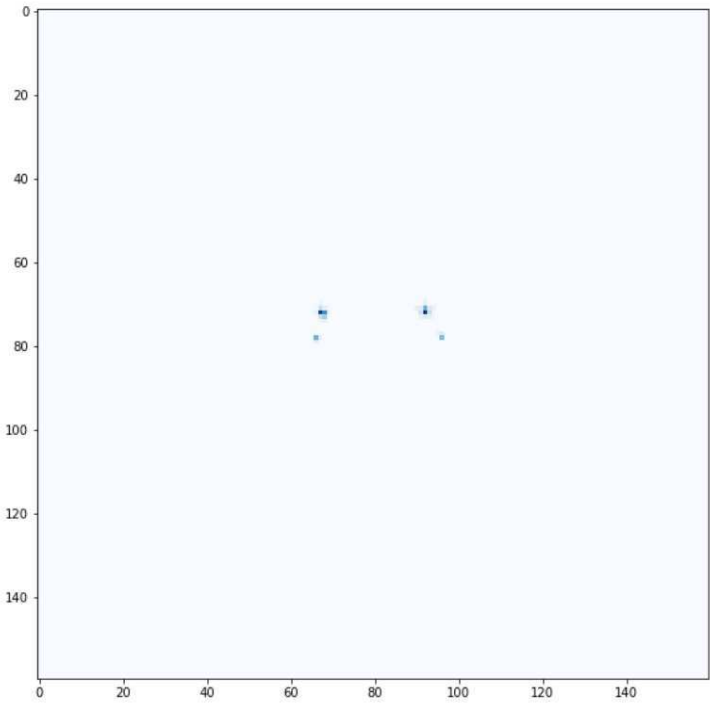


Figure 18: Heatmap of detected eye landmarks in the *ad hoc* dataset.

proximately five thousand) is noticeably lower than other landmark datasets. For example, there are more than 200 thousand images with labeled landmarks in the 300-VW (Tzimiropoulos, 2015) and CelebFaces (Yang et al., 2015) datasets. In fact, we ran some experiments with a subset of the CelebFaces dataset as our training set (and left the entire *ad hoc* dataset for validation). In all of them, overfitting was achieved (i.e., high performance in training but low performance in validation). We believe it comes from the fact that the patterns found in the CelebFaces dataset are even easier than in ours. All faces are centered and with corrected orientation (the last is not carried out in our preprocessing step). Again, as mentioned before, we attempted data augmentation, but it did not improve the overall performance.

iii **Embeddings**: when training the branch to detect eye landmarks, the remaining parts of ICAONet architecture were frozen, including the encoder and corresponding embeddings. We could argue that (i) the embeddings do not contain helpful representations of the eyes that can be relevant for landmark detection, and (ii) they can be harmful to the landmark branch because they do not have the chance to update the embeddings. Despite this, we already had empirical evidence that the embeddings contain helpful information about the eyes before training the landmark branch. In the next chapter, it will be seen that regions closer to the eyes are important for requirements assessment (see Section 5.4).

iv **Training Components**: we ran tests with MSE, Wing Loss, and d_{eye} as both loss functions and metrics for early stopping. After all, ICAONet was unable to predict the

landmarks accurately. This goes against other studies found in the literature that also apply MTL for facial landmark prediction (for example, (Zhang et al., 2014; Ranjan et al., 2017; Zhang et al., 2015). On the other hand, we could have leveraged more advanced losses and models. It will be left for future research.

In conclusion, considering the scope of the possible reasons mentioned above, we suspect the dataset is the main problem for the low performance of eye localization accuracy. Therefore, the results may be improved by revamping the data with more variations in the face pose, location, and orientation. Nevertheless, further investigations must be conducted to verify this hypothesis.

5.2 Results in the FICV Competition

In Table 7, the Equal Error Rate and Rejection Rate for each FICV dataset are presented. In *FICV-TEST*, the proposed method achieved perfect EER in eight requirements (08, 11, 12, 16, 23, 24, 28, and 29). In the *FICV-1.0* dataset, it occurred only in the *Veil Over Face* (28), even though most of the other results were considerably low.

Three requirements had high values of EER (10, 14, and 30) in both datasets. In common, they all have a high level of imbalance (as presented in Table 4). However, other requirements with similar or even worse unbalancing achieved better performance (e.g., 25, 13, or 28). In the case of *Pixelation* (14), we credit this poor result to the preprocessing step because some high-resolution images are pixelated after the resizing step. Moreover, to quickly improve the results of the *Presence of Other Faces or Toys too Close to Face* (30), we could automatically decrease the score of images with two or more faces detected by our detector in the original image. However, the development of post-processing methods is not the primary objective of this work, but it could be considered a future work or be released in the subsequent versions of ICAONet.

Regarding Rejection Rates, eight requirements had images rejected during evaluation. According to our implementation, we only reject images for evaluation when a face is not detected. Therefore, such rejections represent false negatives from the face detector used to preprocess the input images (see section 4.2.1 of Chapter 4). Furthermore, by analyzing these requirements, all of them may hamper face detection in extreme cases.

We highlight the substantial differences in EER between both datasets in Table 7 for requirements 13, 15, and 17. In the case of *Washed Out* (13), we believe it is caused because all the non-compliant images of this requirement in the *FICV-TEST* belong to the AR database (see Figure 19). Therefore, the pattern learned by the network may not have been generalized to the official database. For the *Varied Background* (17) requirement, it may be affected by the cropping applied during the preprocessing step of our method

Table 7: Results of ICAONet according to the benchmark of the FICV competition. The EER and Rejection Rate are shown in percentage.

Req. #	Requirement description	FICV-TEST		FICV-1.0	
		EER	Rej.	EER	Rej.
08	Blurred	0.00	0.00	2.10	0.60
09	Looking away	5.00	0.00	5.40	0.00
10	Ink marked/creased	46.70	0.00	49.00	0.00
11	Unnatural skin tone	0.00	0.00	1.70	0.00
12	Too dark/light	0.00	0.00	1.20	0.00
13	Washed out	1.50	0.00	7.30	0.00
14	Pixelation	26.70	0.00	29.00	0.00
15	Hair across eyes	4.50	0.00	13.70	0.40
16	Eyes closed	0.00	0.00	0.80	0.00
17	Varied Background	1.00	1.00	8.40	1.30
18	Roll/pitch/yaw	2.00	0.00	4.60	0.20
19	Flash reflection on skin	2.10	2.00	1.00	0.00
20	Red eyes	6.90	1.70	8.20	1.50
21	Shadows behind head	2.90	0.00	3.30	0.00
22	Shadows across face	2.00	0.00	3.30	0.20
23	Dark tinted lenses	0.00	0.00	0.40	0.00
24	Flash reflection on lenses	0.00	1.00	0.80	0.00
25	Frames too heavy	9.10	0.00	9.50	0.00
26	Frame covering eyes	1.50	0.00	2.30	0.60
27	Hat/cap	3.10	0.00	5.70	0.20
28	Veil over face	0.00	1.00	0.00	0.00
29	Mouth open	0.00	0.00	2.30	0.00
30	Presence of other faces	41.00	0.00	41.40	0.00

(see Figure 13). The cropping can either (i) generate black borders in the input image or (ii) exclude artifacts that introduce variability to the background. According to our analysis, to understand our network's output, we could observe that black borders do not substantially influence the predictions (more details are provided in Section 5.4). Thus, as can be seen in Figure 20, the artifacts excluded by crop can have a noticeable effect on network learning. Lastly, one possible reason for the *Hair Across Eyes* (15) requirement can be the resizing operation performed by the preprocessing step (see Figure 21). Since the *ad hoc* dataset images are primarily of high resolution and are reduced to 160×160 pixels, it can affect the images where thin locks of hair cross the eye region. These cases are not rare to occur in the dataset, and even using the recommended method for image decimation (see Section 4.2.1), the resize may be contributing negatively to the patterns of this requirement.

The Figure 22 shows the EER in both FICV datasets (as in Table 7), but requirements are ordered by the proportion of non-compliant images in the ad-hoc dataset.

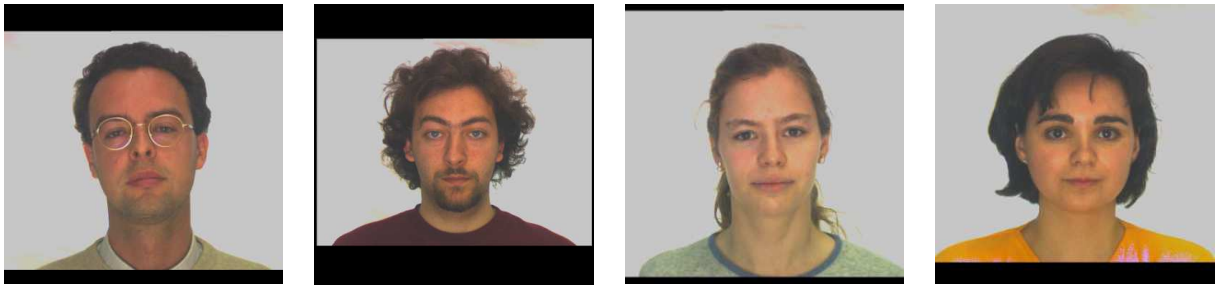
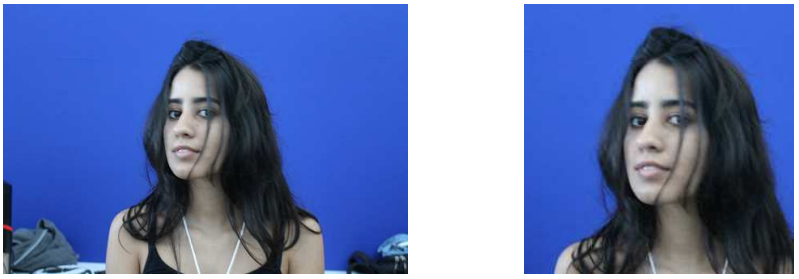


Figure 19: Example of preprocessed non-compliant images from the *Washed Out* (13) requirement. Source: own elaboration.

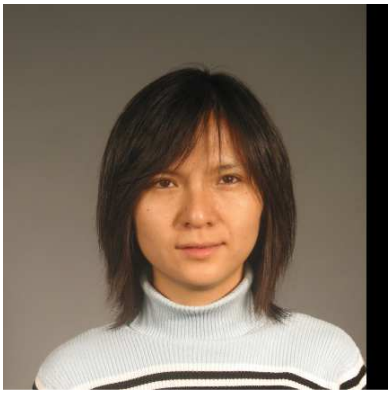


(a) original image (b) preprocessed image

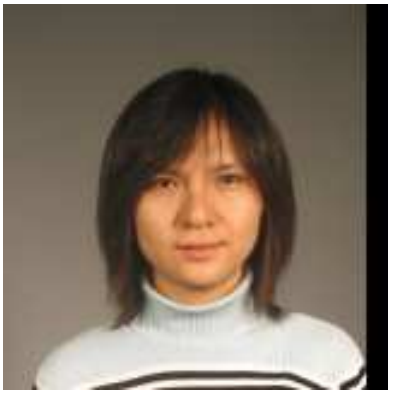
Figure 20: Example of a non-compliant image from the *Varied Background* (17) requirement before and after the preprocessing step. Source: own elaboration.

Through the analysis of this graph, we can draw several conclusions. First, as pointed out earlier, the performances on both datasets are similar. It reinforces the premise that *FICV-TEST* is a representative set of the *FICV-1.0* dataset (see section 2.8). Secondly, there is a moderate correlation between the EER and the degree of imbalance in our dataset. According to Pearson's correlation, these coefficients are -0.48 and -0.46 for the *FICV-TEST* and *FICV-1.0* datasets, respectively. Although these values are indicative of a moderate correlation, it may show that if the proportion of non-compliant images increases, the EER tends to decrease. However, it is important to note that the fifth most unbalanced requirement (*Veil Over Face* (28)), with only 364 non-compliant images (or 6.31%), achieved 0.0% of EER in both datasets. Also, these correlations become very weak and positive ($< +0.1$) when computed starting from the seventh most unbalanced requirement (*Too Dark/Light* (12), with 456 non-compliant images or 7.91%).

Regarding eye location accuracy, the results in FICV can be seen in Table 8. In both datasets of the competition, the results were close to those observed locally during training ($d_{eye} \in [0; 0.1[= 46.18\%$, see Section 5.1.2). Even though $d_{eye} \in [0; 0.1[$ is not as high as expected, we notice that $d_{eye} \leq 0.2$ of ICAONet is higher than 90% in both official datasets. Hence, if we can improve landmark detection of our method, acceptable levels of eye location accuracy can be accomplished. Despite this, significant tokenizable



(a) before resize (1715x1715)



(b) after resize (160x160)

Figure 21: Example of a non-compliant image from the *Hair Across Eyes* (15) requirement before and after the resizing operation performed by the preprocessing step. Source: own elaboration.

results have been achieved.

Table 8: Results of eye localization accuracy for ICAONet in the official datasets of FICV competition.

	$d_{eye} \in [0; 0.1[$	$d_{eye} \in [0.1; 0.2[$	$d_{eye} \in [0.2; 0.3[$	$d_{eye} \geq 0.3$	**Rejected**	**Tokenizable**
FICV-TEST	43.59%	46.44%	4.63%	4.63%	0.71%	88.43%
FICV-1.0	42.68%	48.24%	5.38%	3.41%	0.30%	89.05%

5.2.1 Analysis of the Worst Requirements

The ICAONet achieved non-competitive values of EER for three requirements: *Ink Marked* (10), *Pixelation* (14), and *Presence of Other Faces or Toys too Close to Face* (30). We performed analyses to understand what could be helpful for these requirements. Additionally, experiments were conducted to try to improve the results. The deliberations regarding each requirement are as follows.

As discussed in Section 5.2, the *Pixelation* (14) requirement is primarily affected by the preprocessing step. Because the high-resolution images presented in the dataset are resized to 160×160 pixels, they can generate artificial pixelation effects that may harm the network. In fact, the pixelation branch of ICAONet was placed in shallow layers for this purpose. Thus, the network can use more low-level features from the input image. According to our experiments, without the pixelation branch, the best result for this requirement was 42.7% in terms of EER. However, with the pixelation branch, we could improve EER to 29%. Although this result is still not competitive, it helped decrease our mean EER from 9.3% to 8.8%.

Regarding the *Ink Marked* (10) requirement, the high level of imbalance is the

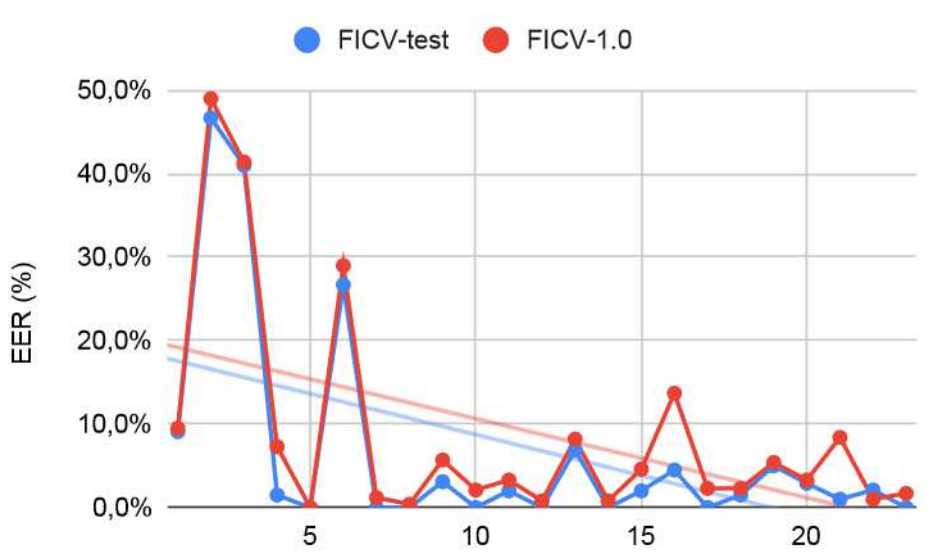


Figure 22: EER by the proportion of non-compliant images for each requirement in ascending order. Source: own elaboration.

main factor for the poor result. Hence, the dataset was improved using data augmentation techniques. We manually collected 32 stamp images with a transparent background across the internet. Then, an arbitrary stamp with random affine transformations (i.e., rotation, zoom, and horizontal/vertical flip) was added to the input image via alpha blending for approximately half of the dataset. Figure 23 shows some examples of augmented images. However, in our experiments, there were no significant improvements in this requirement. Thus, in this case, our central hypothesis is that images from the FICV competition have a different distribution or pattern compared to the generated images.

Figure 23: Example of images augmented with random stamps for *Ink Marked* (10) requirement. Source: own elaboration.

Lastly, the *Presence of Other Faces or Toys too Close to Face* (30) requirement is also influenced by the preprocessing step (see Figure 24). As a reminder, the input image is

cropped around the facial region. Therefore, if other faces are too far from the foreground face, they may not be available in the input image of the network. Moreover, there was one case in which the face detected with the highest score was not in the foreground. In both cases, such errors were caused by the face detector instead of the proposed network. As mentioned before, we can use the number of detected faces to improve the results of such a requirement, but this will be left for future work.

One alternative that could work for both *Pixelation* (14) and *Presence of Other Faces or Toys too Close to Face* (30) requirements is to divide the original high-resolution image into grids. It can be achieved by applying convolutional layers with strides equal to the filter size in the first layer. In this case, the problems discussed regarding the preprocessing step are diminished, and the results can be improved for these requirements. However, for training, this approach consumes considerably more memory and takes longer due to the bigger input image. On the other hand, these differences may not be significant for inference. Again, this hypothesis needs to be verified in future studies.

(a)

(b)

Figure 24: Examples of images affected by preprocessing step for *Presence of Other Faces or Toys too Close to Face* (30) requirement. The original image is on the left side, while the preprocessed image is on the right. (a) The second face cut off by preprocessing step. (b) Wrongly detected faces in the background. Source: own elaboration.

5.3 Comparison Against Other Methods

We begin by comparing the results of ICAONet with well-known architectures fine-tuned for ISO/IEC 19794-5 requirements. Because of the FICV constraint for submission files (up to 50MB, see Section 2.8), only three architectures could be evaluated in the competition: MobileNet v1/v2 (Howard et al., 2017; Sandler et al., 2018) and NasNet-Mobile (Zoph et al., 2018). All networks were previously trained for the general-purpose ImageNet dataset (Deng et al., 2009) and fine-tuned using the same hyperparameters as those used for ICAONet training (see Section 4.2.4).

As seen in Table 9, ICAONet outperformed the other architectures in almost all requirements (18 out of 23). There is a draw for *Veil Over Face* (28), in which MobileNet v1/v2 and our method achieved a perfect EER. Also, MobileNet v1 achieved the best results in three other requirements (*Pixelation* (14), *Hat/Cap* (27), and *Presence of Other Faces or Toys too Close to Face* (30)), whereas NasNet had the worst results among all methods compared. In addition, ICAONet obtained the lowest results regarding the mean/median EER, running time, and memory consumption.

A detailed analysis of the results in Table 9 reveals some interesting patterns. First, we can cite eye-related requirements. For *Looking Away* (09), *Eyes Closed* (16), *Red Eyes* (20), *Dark Tinted Lenses* (23), *Flash Reflection on Lenses* (24), *Frame Too Heavy* (25), and *Frame Covering Eyes* (26), our method achieved considerable improvements in terms of EER (more than 50% lower in comparison to any other method). The only requirement related to eyes that our method accomplished similar performance to the other architectures was *Hair Across Eyes* (15). As mentioned previously (see Section 5.2), these results were probably influenced by our preprocessing step. We can also observe noticeable improvements in illumination-related requirements, such as *Too Dark/Light* (12), *Shadows Behind Head* (21), and *Shadows Across Face* (22). Finally, when considering the highest unbalanced requirements, all methods had high EER values for *Ink Marked* (10) and *Presence of Other Faces or Toys too Close to Face* (30), but the proposed method performed substantially better for *Washed Out* (13) and *Frame Too Heavy* (25).

Table 10 summarizes the best results by requirement among all methods listed in Table 2. Also, it includes the results of ICAONet for ease of comparison. All methods were evaluated using the FICV competition benchmark tool and official dataset (*FICV-1.0*). As can be seen, the proposed method has the best results in 9 out of the 23 requirements of the ISO/IEC 19794-5 standard. Thus, the proposed method has the highest number of best results in terms of the requirements. We can also observe that ICAONet has low rejection rates, rejecting at most 1.3% of the evaluated images. Such rejections represent false negatives from the face detector used to preprocess the input images.

Four methods with public results have been published in FVC-onGoing: BioPass

Table 9: Comparison of the ICAONet against fine-tuned versions of well-known architectures in *FICV-1.0* dataset.

Req. #	Requirement description	MobileNet v1	MobileNet v2	NasNet	ICAONet
08	Blurred	2.1	**1.9**	4.8	2.1
09	Looking away	17.3	26.3	23.1	**5.4**
10	Ink marked/creased	50	49.3	51.0	**49.0**
11	Unnatural skin tone	18.5	19.0	24.0	**1.7**
12	Too dark/light	7.7	7.1	6.7	**1.2**
13	Washed out	15.6	12.3	23.7	**7.3**
14	Pixelation	**24.2**	30.7	27.3	29.0
15	Hair across eyes	15.2	14.8	21.5	**13.7**
16	Eyes closed	9.2	14.8	24.2	**0.8**
17	Varied Background	9.4	10.4	13.7	**8.4**
18	Roll/pitch/yaw	10.6	9.6	24.5	**4.6**
19	Flash reflection on skin	8.3	6.7	12.9	**1.0**
20	Red eyes	28.7	21.5	14.9	**8.2**
21	Shadows behind head	13.3	11.4	15.8	**3.3**
22	Shadows across face	12.1	12.1	12.7	**3.3**
23	Dark tinted lenses	1.0	1.5	1.2	**0.4**
24	Flash reflection on lenses	1.7	2.5	3.3	**0.8**
25	Frames too heavy	50.0	45.3	35.1	**9.5**
26	Frame covering eyes	19.6	19.6	26.2	**2.3**
27	Hat/cap	**0.6**	1.9	1.0	5.7
28	Veil over face	**0.0**	**0.0**	0.2	**0.0**
29	Mouth open	11.7	11.7	17.7	**2.3**
30	Presence of other faces	**36.7**	45.5	48.4	41.4
	mean (%)	15.8	16.4	18.9	**8.8**
	median (%)	12.1	12.1	17.7	**3.3**
	avg. time (s)	2.9	3.1	4.2	**2.7**
	max. time(s)	**3.4**	4.2	5.2	**3.4**
	memory (MB)	332.2	322.5	377.5	**306.1**

Face (BioPassFace, 2017), BioTest (BioTest, 2014), id3 (ICAO_compliance, 2016), and ICAOSDK (ICAO SDK, 2021). Compared to them, we obtained the best results in 11 out of all 23 requirements. Therefore, ICAONet is also the method with the highest number of best results in terms of requirements in FICV competition. Additionally, the proposed method had the second-best Median EER (3.3%).

In terms of performance, the ICAONet takes an average of 2.7 seconds per image, according to the official benchmark results on the FICV competition of the FVC-onGoing website. Compared with methods that evaluate all requirements, the proposed method is among the fastest ones. However, according to our benchmarks, most of the running time is dominated by the face detector (1.6s on average), which is a preprocessing step. On the other hand, the architecture of the proposed network takes only 0.15s of the total time. Furthermore, since the FICV competition runs the benchmarks in CPU-only computers, our network could be even faster using a GPU. In the future, we intend to change our

Table 10: Comparison of the ICAONet against the best results reported in the literature and by private SDK tools (see Table 2). All methods were evaluated by the benchmark tool of FICV using the *FICV-1.0* dataset.

	Best of Literature/ Commercial SDK			ICAONet	
Req. #	Method	EER	Rej.	EER	Rej.
08	BioPass Face	1.60	3.30	2.10	0.60
09	HMAX	10.00	0.16	**5.40**	0.00
10	BioLab	3.40	1.20	49.00	0.00
11	BioPass Face	1.90	0.00	**1.70**	0.00
12	id3	2.90	0.00	**1.20**	0.00
13	BioPass Face	0.00	0.00	7.30	0.00
14	SDK 2	0.00	0.00	29.00	0.00
15	Parente et al.	11.90	3.40	13.70	0.40
16	id3	0.20	1.00	0.80	0.00
17	BioTest	3.70	7.90	8.40	1.30
18	id3	9.10	6.90	**4.60**	0.20
19	BioLab	0.60	0.00	1.00	0.00
20	id3	1.00	2.00	8.20	1.50
21	BioLab	2.30	0.20	3.30	0.00
22	Andrezza et al.	7.70	2.50	**3.30**	0.20
23	BioPass Face	1.80	1.20	**0.40**	0.00
24	BioLab	2.10	0.00	**0.80**	0.00
25	HMAX	0.00	0.00	9.50	0.00
26	HMAX	0.00	0.10	2.30	0.60
27	id3	6.80	0.80	**5.70**	0.20
28	Parente et al.	1.20	0.50	**0.00**	0.00
29	id3	0.60	0.40	2.30	0.00
30	BioPass Face	1.20	2.70	41.40	0.00
		mean (%)		8.80	0.22
		median (%)		3.30	0.00
		avg. time (s)		2.7	
		max. time (s)		3.4	
		memory (MB)		306.1	

face detector to a faster alternative so that the total CPU time will be further reduced.

To improve these results, two distinct approaches can be followed. First, our dataset can be enriched by increasing (i) the number of images and (ii) the variability of patterns of specific requirements (such as hat/cap). Thereby, the network can learn more effective descriptors and decrease the EER for these requirements. Secondly, we may change the network or attempt other loss functions. For instance, we can test loss functions designed for multi-label classification problems, like the Contrastive Loss (Khosla et al., 2020).

A side-by-side comparison between the proposed method and other methods is

Table 11: Side-by-side comparison between ICAONet and all methods presented in Table 2. Each method is summarized by the number of requirements (#Reqs) evaluated. In comparison to ICAONet, the table shows the number of requirements for which the compared method has a greater EER (loses) and by how much the mean/median EER is greater (> 1) or lower (< 1).

		vs ICAONet		
Method	**#Reqs**	**Loses**	**Mean EER**	**Median EER**
SDK1	11	9	2.41×	5.67×
SDK2	9	8	3.43×	10.36×
BioLab	23	14	0.83×	1.58×
BioTest	23	17	1.13×	1.55×
BioPass Face	23	14	0.55×	0.94×
id3	16	9	0.45×	0.58×
ICAO SDK	9	9	4.71×	14.64×
FerraraSeg	3	1	0.80×	1.92×
Borges et al	3	2	0.94×	1.21×
Andrezza et al	3	3	0.48×	1.12×
Parente et al	4	2	0.54×	0.89×
HMAX	9	7	1.61×	3.79×

presented in Table 11. In terms of requirements, ICAONet has the majority of best results against all individual methods, except for "FerraraSeg" (Ferrara et al., 2012a), which has a lower EER in two of the three evaluated requirements. In the case of (Parente et al., 2016), there was a draw (two best results for each method). Moreover, seven methods have a mean EER lower than ICAONet ($< 1\times$). However, four of these seven studies evaluates fewer than four requirements. On the other hand, only three methods presented a median EER better than the proposed method: "BioPass Face" (BioPassFace, 2017), "id3" (id3, 2016), and (Parente et al., 2016). As before, one of these methods evaluates only four requirements.

With respect to eye location accuracy, Table 12 compares ICAONet with the other methods published in the FICV competition. Despite having the worst performance for $d_{eye} \in [0; 0.1[$ (42.68%), as previously discussed, $d_{eye} \leq 0.2$ accuracy of ICAONet is greater than 90%. Thus, if we improve our landmark prediction, we can achieve performance results comparable to BioPass Face and id3 methods. Finally, the landmarks detected by our method can already allow tokenization of almost 90% of the input images, which is better than the BioTest algorithm.

Table 12: Results of eye localization accuracy for the methods with published results in the FICV competition.

	$d_{eye} \in [0; 0.1[$	$d_{eye} \in [0.1; 0.2[$	$d_{eye} \in [0.2; 0.3[$	$d_{eye} \geq 0.3$	**Rejected**	**Tokenizable**
BioPass Face	94.46%	2.00%	1.49%	0.65%	1.41%	92.86%
id3	95.46%	2.11%	0.73%	0.38%	1.32%	95.03%
BioTest	77.08%	5.08%	0.89%	2.73%	14.22%	78.22%
ICAONet	42.68%	48.24%	5.38%	3.41%	0.30%	89.05%

We believe improving our dataset is the most effective way to enhance our landmark predictions. First, the number of samples must be increased to follow other landmark datasets with hundreds of thousands of images. Furthermore, variations in landmarks are an essential factor. For example, it includes more face positions with a higher range of head rotations in all axes (which cannot be achieved using classic image augmentation techniques). Lastly, we could try other custom loss functions specially designed for landmark localization.

5.4 Network Visualization

In addition to the presented performance results, we applied different techniques to understand the network outputs. First, we analyzed the embeddings learned by the network using algorithms for dimensionality reduction. Secondly, we applied network visualization techniques to understand which image regions are the most relevant to each requirement. More details can be found in the following paragraphs.

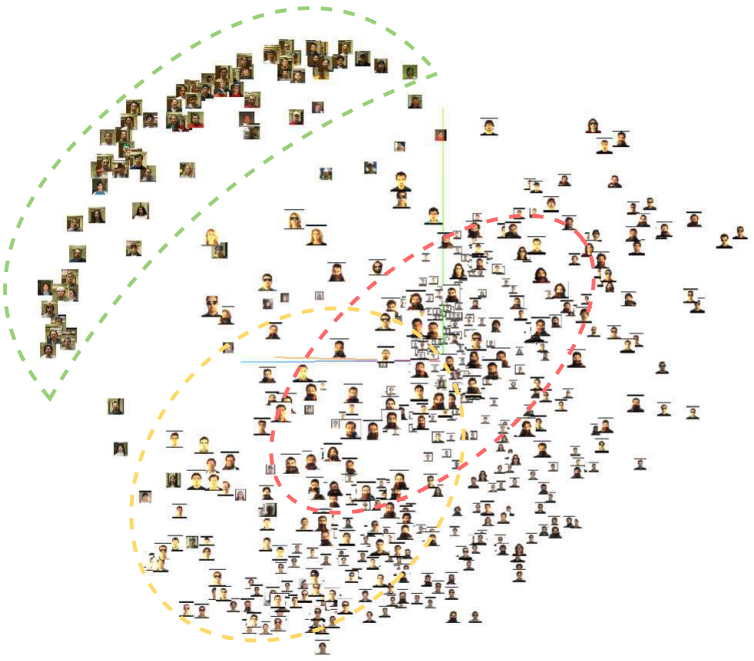
A visualization of the embeddings learned by ICAONet is shown in the 3D plots in Figure 25. The embedding was reduced to 3 dimensions and visualized via the PCA and t-SNE methods using TensorBoard[10]. Each point in the plot is represented by a facial image from the dataset. Although the figure shows only three dimensions, it is possible to observe that some dimensions are related to particular ICAO requirements. For example, in the figure related to PCA, we can observe that the images with *Varied Background* (17), *Unnatural Skin Tone* (11), and *Veil Over Face* (28) are closer to each other to a certain degree. On the other hand, in t-SNE, we notice that the clusters related to such requirements are more well-defined, especially for *Varied Background* (17) and *Veil Over Face* (28) requirements. Moreover, in both figures, we can see the intersection of some regions. For instance, images with unnatural skin tones and veils over the face tend to belong to both clusters. Such information is relevant to the multitask classification branch of the ICAONet architecture.

Figure 26 shows a visual representation of the input images with local region contributions associated with each pose and photograph requirements. The SHAP (Lundberg

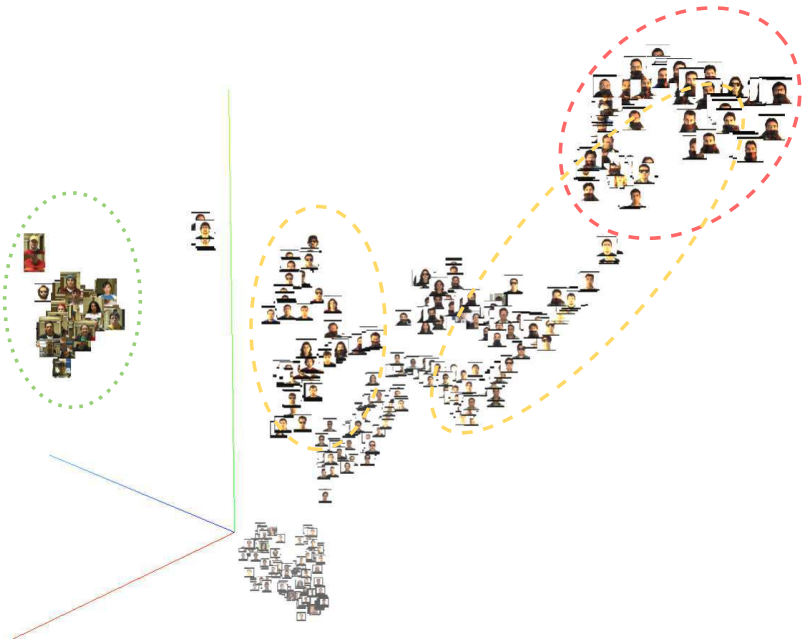
[10]https://www.tensorflow.org/tensorboard

and Lee, 2017) method was used to create the visualization. The figure provides evidence that the network learned useful representations for most requirements. In fully compliant images (first three rows), the classification output is usually increased by the image regions related to that requirement. For example, in the eye-region dependent requirements (09, 15, 16, 20, 23, 24, and 26), the output is mainly influenced by regions closer to the eyes. Similar behaviors can be observed in requirements related to the mouth (28 and 29), skin (11, 19, and 22), and image aspects (08, 12, and 13).

However, we can notice the network could not learn relevant patterns in some requirements like *Ink Marked* (10), *Pixelation* (14), *Frame Too Heavy* (25), *Hat/Cap* (27), and *Presence of Other Faces or Toys too Close to Face* (30). For requirements 10, 25, and 30, we believe that the low number of non-compliant images was the most crucial factor contributing to the worst results of the proposed architecture, as shown in Table 4. In the case of *Pixelation* (14), a possible cause for the high EER (42.7%) could be the image resizing step applied to the input images. Finally, the random patterns in *Hat/Cap* (27) may show that the variability of head props in our dataset must be insufficient to distinguish them from other patterns.



(a) PCA



(b) t-SNE

Figure 25: Visualization of the embeddings learned by ICAONet. The original embedding dimensions were reduced to 3D using (a) PCA and (b) t-SNE. In both visualizations, we highlight the regions of *Varied Background* (17) (green), *Unnatural Skin Tone* (11) (yellow), and *Veil Over Face* (28) (red) requirements. Source: own elaboration.

Figure 26: Visual explanation of ICAONet's output using SHAP. The rows represent the input images, and the columns denote the requirements from the ISO/IEC 19794-5 standard. The first three images are fully compliant, whereas the remaining images have at least one non-compliant requirement. According to the SHAP values, each pixel contributes negatively (blue), positively (red), or has a low contribution (white) to the network output. Some requirements related to the eyes, skin, and mouth were highlighted with vertical coloured rectangles for convenience. Source: own elaboration.

6 Concluding Remarks

This thesis presented a deep learning-based method developed to evaluate photographic requirements and eye location accuracy of ISO/IEC 19794-5 standard, called ICAONet. Our method extends undercomplete Convolutional Autoencoders with supervised branches that performs multi-label classification and landmark regression in a collaborative fashion with unsupervised learning. The architecture has three main components: (i) an encoder to encode the input image into a proper 256-D representation shared by (ii) an unsupervised branch to reconstruct the input image; and (iii) supervised branches to assess the requirements as a multi-label problem, determine the eye-center positions as a regression problem, and classify the specific *Pixelation* (14) requirement as a binary classification task.

We can consider that ICAONet presented valuable advances in its research field. First, compared to other encoder-focused architectures available in the literature, the proposed method does not require pre-training and is easy to scale. Secondly, Multitasking Learning was leveraged with different learning techniques (supervised and unsupervised) and tasks (regression and binary/multi-label classification). Regarding Representation Learning, the proposed architecture also employs both generative and discriminative approaches. Therefore, the method learned a functional representation built from related tasks to predict different outputs. Finally, ICAONet is the first and only open-source research that evaluates all 23 photographic requirements with considerably low memory consumption and running time.

We evaluated our method using a small amount of unbalanced but stratified data. It comprises a subset of the *FICV-TEST* dataset in conjunction with an *ad hoc* dataset built especially for ICAO requirements. The network was trained from scratch, and a custom loss function was used in the network optimization process. This function balances the tasks solved by the method, i.e., image reconstruction, landmark localization, and requirements assessment. Additionally, the training was monitored to preserve the model with the highest F-Beta score.

Individually, the ICAONet was able to achieve significant results. In training, most of the metrics evaluated in the validation set had a score greater than 90%. Through the analysis of these metrics, we were able to notice some patterns in the method predictions. For example, it is better to predict the positive class, but the False Positives are more troublesome than False Negatives. Additionally, in the FICV competition, the method presented a substantial performance in most requirements. Nevertheless, unacceptable performance in terms of EER ($> 40\%$) were obtained for two requirements and the eye location accuracy was below expectations. Both results will require further work for improvement.

Compared to other methods evaluated by the FICV benchmark, the ICAONet was able to achieve state-of-art results in 9 out of 23 requirements and a global median EER of 3.3%. Therefore, the proposed work has the highest amount of best results in a single method compared to all the works presented in the literature or private SDK tools. In terms of EER, the ICAONet has the second-best median EER compared to methods that evaluate all requirements. Furthermore, the proposed method is also among the fastest methods to evaluate all requirements on the CPU, taking only 2.7s to evaluate an input image in average. However, our running time still has a spot for improvement since it is highly influenced by the face detector used for preprocessing. The architecture proposed by itself takes only 0.15s to run in the CPU.

We believe that our method stood out from the others due to the following reasons. First, by using MTL, the model can exploit similarities and dependencies between tasks and learn a useful set of features shared among tasks. The features learned from one task can be beneficial for related tasks, particularly when the tasks have overlapping characteristics. This is the case with the ISO/IEC 19794-5 standard. Furthermore, MTL is known to handle imbalanced and limited data (like the dataset used in this thesis). Thus, we avoided the creation of a model biased to the majority class, which was also enforced by the metrics and regularization techniques applied during training.

Regarding our research question presented in Chapter 1, we can conclude that it has been answered. In terms of efficiency, as cited previously, the proposed method is among the fastest, even without post-optimization. In addition, according to the official FICV benchmark, the method was lightweight, consuming approximately 300 MB of memory. On the other hand, although ICAONet was not able to achieve state-of-the-art results for all 23 requirements, we noticed that Deep Multitasking Learning can be an alternative capable of achieving low error rates. Furthermore, by applying the recommendations for future work cited below, the proposed method will likely improve the results.

As future works, we intend to concentrate efforts on three different aspects to improve the results:

- **Dataset**: the dataset quality can be improved by increasing (i) the number of images and (ii) the variability of patterns of some requirements (like hat/cap). Thereby, the network can learn more effective descriptors and decrease the EER in these requirements. The most unbalanced requirements may require special attention, and probably more images must be gathered. Furthermore, the dataset labels may be revised to fix possible labeling errors.

- **Preprocessing**: as discussed in Chapter 5, the preprocessing step may have been responsible for some of the errors. First, we can replace the current face detector with a faster and more reliable approach like Zhang et al. (2017b). It can help

to decrease the detection time (approximately 90% of total running time) and the Rejection Rates (0.4% max). Moreover, we must improve the input image provided to the network. For some images, the cropping and resizing steps remove or generate artifacts that can harm network learning. See Figures 20 and 21 of Section 5.2 for further details. Thus, we need to find a better way to preprocess the input image as a whole without injuring the trade-off between speed and accurate results.

- **Method**: some elements of the network can also be considered. It includes, but is not limited to, the architecture and the loss function. For example, the Capsule Neural Networks (CapsNets), proposed by Sabour et al. (2017), may help the method create hierarchical representation and capture the spatial relationships between different parts of the input image. Also, the Vision Transformers (Dosovitskiy et al., 2020) can be used to capture global context information, leading to a better understanding of the image as a whole and avoiding image resizing. Recent techniques like Self-Supervised Learning (Doersch and Zisserman, 2017) may also be considered, specially for their ability to reduce annotation costs and handling noisy data. Finally, we intend to test other losses functions specially designed for the multi-label classification task. For instance, the Contrastive Loss (Khosla et al., 2020), designed for few-shot learning scenarios, encourages the model to learn to differentiate between similar and dissimilar pairs of data.

Finally, the present work won an award and was published in a journal, as follows:

- **AI Awards (2^{nd} place)** (DataH, 2019): The AI Awards is a national award for Artificial Intelligence in Brazil. It is considered the highest award of the Brazilian academy for innovative postgraduate projects involving Artificial Intelligence.

- de Andrade e Silva, A. G., Gomes, H. M., and Batista, L. V. (2022). A collaborative deep multitask learning network for face image compliance to ISO/IEC 19794-5 standard. *Expert Systems with Applications*, 198:116756

REFERENCES

Abu-Mostafa, Y. S. (1990). Learning from hints in neural networks. *Journal of complexity*, 6(2):192–198.

Ahmadvand, A. and Moin, M.-S. (2018). Estimating Conformity of Head Yaw to the ICAO Standard using a Convolutional Neural Network. In *2018 9th International Symposium on Telecommunications (IST)*, pages 69–73.

Andrezza, I. L. P., Borges, E. V. C. L., Mota, R. A. T., and Primo, J. J. B. (2016). Facial Compliance for Travel Documents. In *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 166–172. IEEE.

Anil, J. and Suresh, L. P. (2016). Literature survey on face and face expression recognition. In *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pages 1–6. IEEE.

Argyriou, A., Evgeniou, T., and Pontil, M. (2006). Multi-task feature learning. *Advances in Neural Information Processing Systems*, 19.

Baur, C., Wiestler, B., Albarqouni, S., and Navab, N. (2018). Deep autoencoding models for unsupervised anomaly segmentation in brain MR images. In *International MICCAI Brainlesion Workshop*, pages 161–169. Springer.

Baxter, J. (1997). A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine learning*, 28(1):7–39.

Baxter, J. (2000). A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Bengio, Y., Ducharme, R., and Vincent, P. (2000). A neural probabilistic language model. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press.

Biometrika (2014). Biometrika - Identificazione di impronte digitali per sistemi di sicurezza. `http://www.biometrika.it`. [Online; accessed May-2023].

BioPassFace (2017). Result of algorithm BioPass Face 5.6 on FICV-1.0. `https://biolab.csr.unibo.it/FvcOnGoing/UI/Form/AlgResult.aspx?algId=6336`. [Online; accessed December-2020].

BioTest (2014). Result of algorithm BioTest 1.3.8 on FICV-1.0. `https://biolab.csr.unibo.it/FvcOnGoing/UI/Form/AlgResult.aspx?algId=2787`. [Online; accessed December-2020].

Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer.

Borges, E. V. C. L., Andrezza, I. L. P., Marques, J. R. T., Mota, R. A. T., and Primo, J. J. B. (2016). Analysis of the Eyes on Face Images for Compliance with ISO/ICAO Requirements. In *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 173–179. IEEE.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*, pages 144–152. ACM Press.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). Classification and regression trees. *International Group*, 432:151–166.

Cai, Z., Fan, Q., Feris, R. S., and Vasconcelos, N. (2016). A Unified Multi-scale Deep Convolutional Neural Network for Fast Object Detection. In *Computer Vision – ECCV 2016*, pages 354–370. Springer International Publishing.

Cao, Q., Shen, L., Xie, W., Parkhi, O. M., and Zisserman, A. (2018). VGGFace2: A Dataset for Recognising Faces across Pose and Age. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 67–74. IEEE, IEEE.

Caruana, R. (1997). Multitask Learning. *Machine Learning*, 28(1):41–75.

Chang, J. C., Amershi, S., and Kamar, E. (2017). Revolt: Collaborative Crowdsourcing for Labeling Machine Learning Datasets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 2334–2346. ACM.

Chen, Z., Badrinarayanan, V., Lee, C.-Y., and Rabinovich, A. (2018a). Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pages 794–803. PMLR.

Chen, Z., Liu, B., Brachman, R., Stone, P., and Rossi, F. (2018b). *Lifelong Machine Learning: Second Edition*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Chicco, D. (2017). Ten quick tips for machine learning in computational biology. *BioData Mining*, 10(1):1–17.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537.

DataH (2019). AI Awards. `https://www.datah.ai/a-i-awards`. [Online; accessed 26-March-2023].

de Andrade e Silva, A. G., Gomes, H. M., and Batista, L. V. (2022). A collaborative deep multitask learning network for face image compliance to ISO/IEC 19794-5 standard. *Expert Systems with Applications*, 198:116756.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Doersch, C. and Zisserman, A. (2017). Multi-task self-supervised visual learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2051–2060.

Domhan, T. and Hieber, F. (2017). Using target-side monolingual data for neural machine translation through multi-task learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1500–1505.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Dosovitskiy, A., Springenberg, J. T., Riedmiller, M., and Brox, T. (2014). Discriminative unsupervised feature learning with convolutional neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Ebinger, P., Castro Neves, M., Salamon, R., and Seibert, H. (2008). International Database of Facial Images for Performance and ISO/IEC 19794-5 Conformance Tests. *BIOSIG 2008: Biometrics and Electronic Signatures*.

Evgeniou, T., Micchelli, C. A., Pontil, M., and Shawe-Taylor, J. (2005). Learning multiple tasks with kernel methods. *Journal of machine learning research*, 6(4).

Feng, Z., Kittler, J., Awais, M., Huber, P., and Wu, X.-J. (2018). Wing loss for robust facial landmark localisation with convolutional neural networks. In *Proceedings - 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2018*, IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2235–2245, United States. IEEE Computer Society. © 2018, The Author(s). ; 31st Meeting of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2018 ; Conference date: 18-06-2018 Through 22-06-2018.

Ferrara, M., Franco, A., and Maio, D. (2012a). A multi-classifier approach to face image segmentation for travel documents. *Expert Systems with Applications*, 39(9):8452–8466.

Ferrara, M., Franco, A., Maio, D., and Maltoni, D. (2012b). Face image conformance to ISO/ICAO standards in machine readable travel documents. *IEEE Transactions on Information Forensics and Security*, 7(4):1204–1213.

Fukunaga, K. and Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1):32–40.

FVC-onGoing (2006). Face Image ISO Compliance Verification. [Online; accessed December-2020].

Gao, Y., Ma, J., Zhao, M., Liu, W., and Yuille, A. L. (2019). NDDR-CNN: Layerwise Feature Fusing in Multi-task CNNs by Neural Discriminative Dimensionality Reduction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3205–3214.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Guissous, A. E. (2019). Skin Lesion Classification Using Deep Neural Network. Master's thesis, UNIVERSITY OF MOHAMED El-BACHIR El-IBRAHIMI.

Guo, T., Dong, J., Li, H., and Gao, Y. (2017). Simple convolutional neural network on image classification. In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, pages 721–724. IEEE.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hernandez-Ortega, J., Fierrez, J., Gomez, L. F., Morales, A., Gonzalez-de Suso, J. L., and Zamora-Martinez, F. (2022). Faceqvec: Vector quality assessment for face biometrics based on iso compliance. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 84–92.

Hernandez-Ortega, J., Galbally, J., Fierrez, J., Haraksim, R., and Beslay, L. (2019). FaceQnet: Quality assessment for face recognition based on deep learning. In *2019 International Conference on Biometrics (ICB)*, pages 1–8.

Hinton, G. and Roweis, S. T. (2002). Stochastic neighbor embedding. In *NIPS*, volume 15, pages 833–840. Citeseer.

Hinton, G. E. et al. (1986). Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Hirzer, M., Urschler, M., Bischof, H., Birchbauer, J. A., and Center, S. B. (2009). An automatic hybrid segmentation approach for aligned face portrait images. In *Proceedings of the Workshop of the Austrian Association for Pattern Recognition*, pages 49–60.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Hsu, F.-H. (2002). *Behind Deep Blue: Building the computer that defeated the world chess champion*. Princeton University Press.

ICAO (2003). Biometrics deployment of machine readable travel documents. Technical Report ICAO TAG MRTD/NTWG Version 1.9.

ICAO SDK (2021). Result of algorithm ICAO SDK 1.0.0 on FICV-1.0. `https://biolab.csr.unibo.it/FvcOnGoing/UI/Form/AlgResult.aspx?algId=8134`. [Online; accessed June-2021].

ICAO_compliance (2016). Result of algorithm ICAO_compliance 1.1.4 on FICV-1.0. `https://biolab.csr.unibo.it/FvcOnGoing/UI/Form/AlgResult.aspx?algId=5343`. [Online; accessed December-2020].

id3 (2016). id3 Technologies - Biometrics | ID Solutions | Eletronic design. `https://id3.eu`. [Online; accessed May-2023].

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.

ISO (2007). Information Technology - Biometric Data Interchange Formats - Part 5: Face Image Data/Amendment 1: Conditions for Taking Photographs or Face Image Data. Technical Report ISO/IEC 19794-5:2005.

Izenman, A. J. (2013). Linear discriminant analysis. In *Modern multivariate statistical techniques*, pages 237–280. Springer.

Jesorsky, O., Kirchberg, K. J., and Frischholz, R. W. (2001). Robust Face Detection Using the Hausdorff Distance. In *Lecture Notes in Computer Science*, pages 90–95. Springer Berlin Heidelberg.

Karimpouli, S. and Tahmasebi, P. (2019). Segmentation of digital rock images using deep convolutional autoencoder networks. *Computers & Geosciences*, 126:142–150.

Kasinski, A., Florek, A., and Schmidt, A. (2008). The PUT face database. *Image Processing and Communications*, 13(3-4):59–64.

Kendall, A., Gal, Y., and Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. (2020). Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*.

King, D. E. (2009). Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes.

Kohler, J., Daneshmand, H., Lucchi, A., Hofmann, T., Zhou, M., and Neymeyr, K. (2019). Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 806–815. PMLR.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Kung, T. N. C. (2006). NCKU CSIE Robotics Lab face. `http://robotics.csie.ncku.edu.tw/database.html`. [Online; accessed June-2021].

la Torre, M. D., Granger, E., Radtke, P. V. W., Sabourin, R., and Gorodnichy, D. O. (2015). Partially-supervised learning from facial trajectories for face recognition in video surveillance. *Information Fusion*, 24:31–53.

Le-Khac, P. H., Healy, G., and Smeaton, A. F. (2020). Contrastive representation learning: A framework and review. *Ieee Access*, 8:193907–193934.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Li, Q., Cai, W., Wang, X., Zhou, Y., Feng, D. D., and Chen, M. (2014). Medical image classification with convolutional neural network. In *2014 13th international conference on control automation robotics & vision (ICARCV)*, pages 844–848. IEEE.

Lin, M., Chen, Q., and Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Liu, S., Johns, E., and Davison, A. J. (2019). End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1871–1880.

Lundberg, S. and Lee, S. (2017). A unified approach to interpreting model predictions. *CoRR*, abs/1705.07874.

Maltoni, D., Franco, A., Ferrara, M., Maio, D., and Nardelli, A. (2009). Biolab-ICAO: a new benchmark to evaluate applications assessing face image compliance to ISO/IEC 19794-5 standard. In *16th IEEE International Conference on Image Processing (ICIP)*, pages 41–44.

Martinez, A. M. (1998). The AR face database. Computer Vision Center (CVC) Technical Report.

Martinez, A. M. and Kak, A. C. (2001). PCA versus LDA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):228–233.

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. (2016). Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003.

Mnih, A. and Kavukcuoglu, K. (2013). Learning word embeddings efficiently with noise-contrastive estimation. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Nguyen, T. H. B., Nguyen, V. H., and Kim, H. (2013). Automated conformance testing for ISO/IEC 19794-5 Standard on facial photo specifications. *International Journal of Biometrics*, 5(1):73.

Nourbakhsh, A., Moin, M.-S., and Sharifi, A. (2020). Facial Images Quality Assessment based on ISO/ICAO Standard Compliance Estimation by HMAX Model. *Journal of Information Systems and Telecommunication (JIST)*, 7(3):225–237.

Nowak, S. and Ruger, S. (2010). How reliable are annotations via crowdsourcing. In *Proceedings of the international conference on Multimedia information retrieval - MIR '10*, MIR '10, pages 557–566. ACM Press.

Orriols-Puig, A., Casillas, J., and Bernadó-Mansilla, E. (2008). Genetic-based machine learning systems are competitive for pattern recognition. *Evolutionary Intelligence*, 1:209–232.

Paoletti, M. E., Haut, J. M., Plaza, J., and Plaza, A. (2018). A new deep convolutional neural network for fast hyperspectral image classification. *ISPRS journal of photogrammetry and remote sensing*, 145:120–147.

Parente, R. L., Batista, L. V., Andrezza, I. L. P., Borges, E. V. C. L., and Mota, R. A. T. (2016). Assessing Facial Image Accordance to ISO/ICAO Requirements. In *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 180–187. IEEE.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Petscharnig, S., Lux, M., and Chatzichristofis, S. (2017). Dimensionality reduction for image features using deep learning and autoencoders. In *Proceedings of the 15th international workshop on content-based multimedia indexing*, pages 1–6.

Pfeiffer, J., Broscheit, S., Gemulla, R., and Göschl, M. (2018). A neural autoencoder approach for document ranking and query refinement in pharmacogenomic information retrieval. In *Proceedings of the BioNLP 2018 workshop*, pages 87–97.

Phillips, P. J., Flynn, P. J., Scruggs, T., Bowyer, K. W., Chang, J., Hoffman, K., Marques, J., Min, J., and Worek, W. (2005). Overview of the face recognition grand challenge. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 947–954.

Phillips, P. J., Wechsler, H., Huang, J., and Rauss, P. J. (1998). The FERET database and evaluation procedure for face-recognition algorithms. *Image and vision computing*, 16(5):295–306.

Ranjan, R., Patel, V. M., and Chellappa, R. (2017). Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *IEEE transactions on pattern analysis and machine intelligence*, 41(1):121–135.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "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*, pages 1135–1144.

Ross, A. A., Nandakumar, K., and Jain, A. K. (2006). *Handbook of multibiometrics*, volume 6. Springer Science & Business Media.

Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Ruder, S., Bingel, J., Augenstein, I., and Søgaard, A. (2019). Latent multi-task architecture learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4822–4829.

Russell, S. and Norvig, P. (2020). *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson.

Ryan (2020). Day 88: What is Autoencoders? `https://ryanong.co.uk/2020/03/28/day-88-what-is-autoencoders/`. [Online; accessed June-2021].

Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. *arXiv preprint arXiv:1710.09829*.

Sakurada, M. and Yairi, T. (2014). Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, pages 4–11.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

Sang, J., Lei, Z., and Li, S. Z. (2009). Face Image Quality Evaluation for ISO/IEC Standards 19794-5 and 29794-5. In *Advances in Biometrics*, pages 229–238. Springer Berlin Heidelberg.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. (2018). How does batch normalization help optimization? *arXiv preprint arXiv:1805.11604*.

Seamfix (2021). Seamfix - Powering seamless digital transformation. `https://seamfix.com`. [Online; accessed May-2023].

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626. IEEE.

Sener, O. and Koltun, V. (2018). Multi-task learning as multi-objective optimization. *arXiv preprint arXiv:1810.04650*.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning : from theory to algorithms*. Cambridge university press.

Shapley, L. S. (1953). A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317.

Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153. PMLR.

Shrikumar, A., Greenside, P., Shcherbina, A., and Kundaje, A. (2016). Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*.

Sim, T., Baker, S., and Bsat, M. (2002). The CMU pose, illumination, and expression (PIE) database. In *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, pages 53–58. IEEE.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Singla, K., Can, D., and Narayanan, S. (2018). A multi-task approach to learning multilingual representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 214–220.

Strang, G. (2020). *Calculus v1*. OpenStax.

Subasic, M., Loncaric, S., and Birchbauer, J. (2009). Expert system segmentation of face images. *Expert Systems with Applications*, 36(3):4497–4507.

Tarrés, F. (2012). GTAV face database. `http://gps-tsc.upc.es/GTAV/ResearchAreas/UPCFaceDatabase/GTAVFaceDatabase.html`.

Thrun, S. and Pratt, L. (1998). Learning to learn: Introduction and overview. *Learning to learn*, pages 3–17.

Torralba, A., Murphy, K. P., and Freeman, W. T. (2004). Sharing features: efficient boosting procedures for multiclass object detection. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II. IEEE.

Tzimiropoulos, G. (2015). Project-out cascaded regression with an application to face alignment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3659–3667.

van der Maaten, L. and Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605.

Vandenhende, S., Georgoulis, S., Gansbeke, W. V., Proesmans, M., Dai, D., and Gool, L. V. (2021). Multi-Task Learning for Dense Prediction Tasks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1.

Vandenhende, S., Georgoulis, S., and Van Gool, L. (2020). MTI-Net: Multi-Scale Task Interaction Networks for Multi-Task Learning. In *European Conference on Computer Vision*, pages 527–543. Springer.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. IEEE.

Vsoft (2017). BioPass ID | Vsoft. `https://www.vsoft.com.br/produto/biopassid`. [Online; accessed May-2023].

Wang, X., Zhang, R., Kong, T., Li, L., and Shen, C. (2020). SOLOv2: Dynamic and fast instance segmentation. *Advances in Neural Information Processing Systems*.

Wang, Y., Yao, H., Zhao, S., and Zheng, Y. (2015). Dimensionality reduction strategy based on auto-encoder. In *Proceedings of the 7th International Conference on Internet Multimedia Computing and Service*, pages 1–4.

Wing, B. J. (2011). Data Format for the Interchange of Fingerprint, Facial and Other Biometric Information ANSI/NIST-ITL 1-2011 NIST Special Publication 500-290 Edition 2. Technical report, National Institute of Standards and Technology.

Wold, S., Esbensen, K., and Geladi, P. (1987). Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1-3):37–52.

Wu, B., Wan, A., Iandola, F., Jin, P. H., and Keutzer, K. (2017). SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 129–137. IEEE.

Xu, D., Ouyang, W., Wang, X., and Sebe, N. (2018). Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 675–684.

Xu, Y., Arai, S., Tokuda, F., and Kosuge, K. (2020). A convolutional neural network for point cloud instance segmentation in cluttered scene trained by synthetic data without color. *IEEE Access*, 8:70262–70269.

Yang, G., Pennington, J., Rao, V., Sohl-Dickstein, J., and Schoenholz, S. S. (2019). A Mean Field Theory of Batch Normalization. In *International Conference on Learning Representations*.

Yang, S., Luo, P., Loy, C.-C., and Tang, X. (2015). From facial parts responses to face detection: A deep learning approach. In *Proceedings of the IEEE international conference on computer vision*, pages 3676–3684.

Yang, S., Luo, P., Loy, C.-C., and Tang, X. (2016). Wider face: A face detection benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5525–5533.

Yeephycho (2017). Tensorflow Face Detector. `https://github.com/yeephycho/tensorflow-face-detection`.

Zhang, R., Isola, P., and Efros, A. A. (2017a). Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhang, S., Zhu, X., Lei, Z., Shi, H., Wang, X., and Li, S. Z. (2017b). FaceBoxes: A CPU real-time face detector with high accuracy. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, volume abs/1708.05234. IEEE.

Zhang, Y., Chu, J., Leng, L., and Miao, J. (2020). Mask-refined R-CNN: A network for refining object details in instance segmentation. *Sensors*, 20(4):1010.

Zhang, Y. and Yang, Q. (2018). An overview of multi-task learning. *National Science Review*, 5(1):30–43.

Zhang, Y. and Yang, Q. (2021). A Survey on Multi-Task Learning. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1.

Zhang, Z., Cui, Z., Xu, C., Jie, Z., Li, X., and Yang, J. (2018). Joint task-recursive learning for semantic segmentation and depth estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 235–251.

Zhang, Z., Cui, Z., Xu, C., Yan, Y., Sebe, N., and Yang, J. (2019). Pattern-affinitive propagation across depth, surface normal and semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4106–4115.

Zhang, Z., Luo, P., Loy, C. C., and Tang, X. (2014). Facial landmark detection by deep multi-task learning. In *European conference on computer vision*, pages 94–108. Springer.

Zhang, Z., Luo, P., Loy, C. C., and Tang, X. (2015). Learning deep representation for face alignment with auxiliary attributes. *IEEE transactions on pattern analysis and machine intelligence*, 38(5):918–930.

Zhao, J., Xie, X., Xu, X., and Sun, S. (2017). Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929.

Zhu, X. and Ramanan, D. (2012). Face detection, pose estimation, and landmark localization in the wild. In *2012 IEEE conference on computer vision and pattern recognition*, pages 2879–2886. IEEE.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710.