



UNIVERSIDAD  
DE MÁLAGA

Escuela Técnica Superior de  
Ingeniería Informática

Programa de Doctorado en  
Tecnologías Informáticas

## TESIS DOCTORAL

# Gait Recognition from Multiple View-Points

***Francisco Manuel Castro Payán***

Directores:

***Nicolás Guil Mata***

***Manuel Jesús Marín Jiménez***


2018





UNIVERSIDAD  
DE MÁLAGA

AUTOR: Francisco Manuel Castro Payán

 <http://orcid.org/0000-0002-7340-4976>

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga



Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional:

<http://creativecommons.org/licenses/by-nc-nd/4.0/legalcode>

Cualquier parte de esta obra se puede reproducir sin autorización pero con el reconocimiento y atribución de los autores.

No se puede hacer uso comercial de la obra y no se puede alterar, transformar o hacer obras derivadas.

Esta Tesis Doctoral está depositada en el Repositorio Institucional de la Universidad de Málaga (RIUMA): [riuma.uma.es](http://riuma.uma.es)



UNIVERSIDAD  
DE MÁLAGA

Departamento de Arquitectura de Computadores

TESIS DOCTORAL

# Gait Recognition from Multiple View-Points

Francisco Manuel Castro Payán

Noviembre de 2018

Dirigida por:

Dr. Nicolás Guil Mata

Dr. Manuel Jesús Marín Jiménez



Dr. D. Nicolás Guil Mata.  
Catedrático del Departamento de Ar-  
quitectura de Computadores de la Uni-  
versidad de Málaga.

Dr. D. Manuel Jesús Marín Jiménez.  
Profesor Contratado Doctor del De-  
partamento de Informática y Análisis  
Numérico de la Universidad de Cór-  
doba.

### **CERTIFICAN:**

Que la memoria titulada “*Gait Recognition from Multiple View-Points*”, ha sido realizada por D. Francisco Manuel Castro Payán bajo nuestra dirección en el Departamento de Arquitectura de Computadores de la Universidad de Málaga y constituye la Tesis que presenta para optar al grado de Doctor en Tecnologías Informáticas.

Málaga, Noviembre de 2018

Dr. D. Nicolás Guil Mata.  
Codirector de la tesis.

Dr. D. Manuel Jesús Marín Jiménez.  
Codirector de la tesis.



A Marisa, mi familia, amigos y compañeros





# Agradecimientos

---

Tras estos cuatro años de duro trabajo, me gustaría dar las gracias a todas esas personas que, de una forma u otra, han estado apoyándome tanto en los buenos momentos como durante los más duros. Sin vosotros, todo hubiera sido mucho más complicado. Gracias.

En el ámbito académico, agradecer a mis directores Nicolás y Manuel Jesús todo su apoyo y trabajo durante estos años, habéis sido una fuente de inspiración y un modelo a seguir. A Nicolás, me gustaría agradecerle todo el tiempo y apoyo dedicado diariamente, especialmente en los interminables días antes de un deadline o tras una revisión. Gracias también por tus innumerables ideas y búsqueda de puntos en común con otros investigadores o empresas para colaborar. Me has demostrado que nuestro trabajo también puede ser útil desde un punto de vista práctico, no solamente teórico. A Manuel Jesús, agradecerle todo su trabajo e incansable ayuda durante estos más de seis años que llevamos trabajando juntos. Si hoy estoy escribiendo esta tesis, es gracias a ti por haberme enseñado lo que es la investigación y haber confiado en mí. Sin tu ayuda, probablemente estaría ajeno a este mundo picando código en una empresa. Gracias a los dos. Por supuesto, agradecer también a mi tutora de tesis M. Ángeles, que siempre ha estado ahí para resolver todas las dudas y ayudarme con los muchos trámites involucrados en la tesis.

I would also like to thank KartEEK his help, time and hard work. I learnt a lot during the internship and it was a pleasure and a great motivation to work with you and the other guys everyday. Thanks.

En el ámbito personal, agradecer a Marisa todo su apoyo e incansable ayuda. Siempre has estado ahí, soportando las interminables horas de trabajo, los viajes y estancias, problemas, mal humor, todo. Probablemente, todo lo que pueda escribir para darte las gracias es poco comparado con lo que tú me has dado. Gracias, parte de esta tesis lleva tu nombre porque sin tus ánimos se que no hubiera conseguido todo esto.

Por supuesto, dar las gracias a mis padres y mi hermano, quienes siempre me han apoyado y motivado a sacar lo mejor de mí mismo en todos los aspectos y a no rendirme nunca. Por animarme a no coger el camino fácil y rápido, sino a coger el complicado pensando en el futuro y en las recompensas que habrá al final. Tampoco podría olvidar a mi abuela Encarna y mis tíos Paco y Paqui por todos los detalles y cariño que han tenido conmigo durante todo este tiempo.

También me gustaría dar las gracias a Rafa, Esperanza, Fran y Maricarmen que siempre me han dado fuerzas y ayudado en lo que he necesitado. Aunque estéis lejos, os he sentido cerca en todo momento.

Dar las gracias a todos mis amigos malagueños que me han enseñado a querer esta hermosa ciudad. Hace cuatro años cuando llegué a Málaga, me sentía perdido y fuera de lugar, hoy me siento parte de ella. En especial me gustaría agradecer a Pedro y María todo su cariño y apoyo, me habéis hecho sentir como en casa. Nunca pensé que de un día para otro mi familia iba a crecer, pero hoy en día cuento con otro hermano y una hermana, gracias.

También quiero dar las gracias a mis compañeros de laboratorio que me han enseñado infinidad de cosas de un área nueva como es la Arquitectura y de la que hoy ya no tengo tanto “miedo”. Sois muchos pero me gustaría nombraros (y espero no olvidar a ninguno), gracias a Antonio, Jesús, Bernabé, Alejandro, Pedrero, Herruzo, Ricardo, Denisa, Vilches, Sergio, Carlos, Esteban, Óscar, Andrés, Iván y José Carlos. Por supuesto, dar las gracias también a los técnicos de laboratorio Juanjo y Paco, que siempre me han ayudado y aguantado cuando colgaba algún servidor o había que cambiar todas las GPUs del departamento. También dar las gracias a Carmen por ayudarme en todo el papeleo que he tenido que realizar estos años.

Por último, me gustaría nombrar a Vicky, Keiki e Ironman, que aunque no sean personas, forman parte de mi “rara familia” y siempre me han alegrado al volver a casa y encontrarme sus saludos y cariño.

# Abstract

---

Nowadays, people identification is gaining importance due to the security problems and terrorist attacks that have occurred in recent years. Traditionally, people identification has been performed using face, iris or fingerprint recognition approaches that automatically identify the subject without the collaboration of a human. Thus, this systems have helped to automatize the identification process providing reliable results faster than a group of watchmen identifying people through security cameras. However, those kind of systems require the collaboration of the subject for being identified what introduces a huge limitation because in some scenarios that collaboration is impossible. For example, people wearing special security suits (*e.g.* NBC suits) should remove a part of the suit to be identified what is a nonsense situation. Moreover, terrorists or thieves are not going to collaborate with a system that will help the police to capture them.

In this world where safety is a priority, gait recognition has emerged as an effective alternative to traditional recognition methods. In contrast to face, iris or fingerprint recognition, gait recognition does not require the collaboration of the subject and it is robust against clothes that cover the entire body. Therefore, gait recognition can be implemented in situations where any other recognition system could work due to the limitations of the environment. Thus, gait recognition has softer requirements both for the hardware and for the subject to be identified. It allows to use cheap security cameras that can be located in high places covering a large visible space. Moreover, the subjects do not need to collaborate or even know that they are being identified by their way of walking.

Gait recognition is being studied more in detail in recent years and everyday is more often to find published paper about this topic while some time ago very few researchers worked on it. However, not only theoretical applications are being proposed using gait recognition. Real applications are being used in Japan where gait recognition is considered as a criminal probe to identify the causer. Moreover, it is sounding like possible security control for airports in the near future.

In this thesis we tackle the gait recognition problem from different perspectives which are useful in different situations. Firstly, we propose a solution based on hand-crafted features which is specially useful when few training data is available. This solution uses a pipeline to obtain high-level features from the input data and, finally, it produces a probability distribution to identify the subject that appears in the input. Secondly, we propose a deep learning approach which is able to obtain the high-level features and the probability distribution of the subject identity in a joint manner. This approach is specially suitable when the video resolution is small and the training data is enough to avoid overfitting. Additionally, if multiple sources of information are available at the same time, we have designed a set of fusion schemes to combine the information to improve the performance of the system. This way, the information can be combined at different levels in the pipeline of the approaches producing different grades of precision. When instead of multiple sources of information, there are multiple kind of labels per sample, we have designed a multi-task approach which learns to produce all the output information at the same time. Thus, for a given sample, the proposed approach produces multiple labels using the gait information. Due to the current importance of the deep learning approaches, we have studied the power consumption of those kind of methods during training to propose improvements to save energy and money. Lastly, we have proposed an incremental learning approach which learns new classes while retains the previously learned ones. This way, this approach allows to perform smaller training processes using less information.

# Contents

<b>Agradecimientos</b>	<b>I</b>
<b>Abstract</b>	<b>III</b>
<b>Contents</b>	<b>IX</b>
<b>List of Figures</b>	<b>XII</b>
<b>List of Tables</b>	<b>XIV</b>
<b>1.- Introduction</b>	<b>1</b>
1.1. Motivation . . . . .	3
1.2. Objectives and phases . . . . .	4
1.3. Thesis contributions . . . . .	6
1.4. Thesis organization . . . . .	8
<b>2.- Background</b>	<b>9</b>
2.1. Machine learning based on hand-crafted features . . . . .	9
2.1.1. Data pre-processing . . . . .	10
2.1.2. Low-level descriptors . . . . .	11
2.1.2.1. Dense trajectories and DCS . . . . .	13
2.1.3. High-level descriptors . . . . .	14

2.1.3.1.	Fisher Vectors . . . . .	15
2.1.4.	Classification . . . . .	17
2.1.4.1.	SVM . . . . .	18
2.2.	Deep learning . . . . .	19
2.2.1.	Layers . . . . .	20
2.2.1.1.	Convolutional layer . . . . .	21
2.2.1.2.	Pooling layer . . . . .	23
2.2.1.3.	Local Response Normalization layer . . . . .	24
2.2.1.4.	Fully-connected layer . . . . .	25
2.2.1.5.	Dropout layer . . . . .	25
2.2.1.6.	Batch-normalization layer . . . . .	26
2.2.2.	Loss function . . . . .	26
2.2.2.1.	Cross-Entropy loss function . . . . .	27
2.2.2.2.	Tukey's biweight loss function . . . . .	27
2.2.3.	Optimizers . . . . .	28
2.2.3.1.	Mini-batch Gradient Descent . . . . .	28
2.2.3.2.	Momentum . . . . .	29
2.2.3.3.	Nesterov accelerated gradient . . . . .	29
2.2.3.4.	Adagrad . . . . .	30
2.2.3.5.	Adadelata . . . . .	30
2.2.3.6.	Adam . . . . .	31
2.2.4.	Hyper-parameters . . . . .	32
2.3.	Information fusion . . . . .	33
2.3.1.	Early fusion . . . . .	33
2.3.1.1.	Feature vector concatenation . . . . .	34
2.3.1.2.	Bi-modal codewords . . . . .	34
2.3.1.3.	Multiple Kernel Learning . . . . .	34
2.3.2.	Late fusion . . . . .	35

2.3.2.1. Weighted Scores . . . . .	35
2.3.2.2. Classifier over the scores . . . . .	36
2.3.2.3. Rank Minimization . . . . .	36
2.4. Incremental learning . . . . .	37
2.4.1. Deep incremental learning . . . . .	38
2.5. Energy consumption of deep learning . . . . .	39
2.5.1. Energy consumption on GPUs . . . . .	40
<b>3.- Related work</b>	<b>43</b>
3.1. Hand-crafted approaches . . . . .	43
3.2. Deep learning approaches . . . . .	44
3.3. Fusion approaches . . . . .	46
3.4. Multi-task approaches . . . . .	47
3.5. Incremental learning approaches . . . . .	47
3.6. Energy consumption studies on deep learning . . . . .	49
<b>4.- Published Work</b>	<b>51</b>
4.1. List of Published Papers . . . . .	51
4.2. Summary of the papers that support this thesis . . . . .	52
4.2.1. Reference [89] ‘On how to improve tracklet-based gait recognition systems’ . . . . .	53
4.2.2. Reference [18] ‘Multimodal features fusion for gait, gender and shoes recognition’ . . . . .	53
4.2.3. Reference [21] ‘Fisher motion descriptor for multiview gait recognition’ . . . . .	54
4.2.4. Reference [17] ‘Energy-based Tuning of Convolutional Neural Networks on Multi-GPUs’ . . . . .	54
4.3. Additional papers . . . . .	55
4.3.1. Reference [88] ‘Deep Multi-Task Learning for Gait-based Biometrics’ . . . . .	55

4.3.2. Reference [20] ‘End-to-End Incremental Learning’ . . . . .	56
<b>5.- Unpublished Work</b>	<b>57</b>
5.1. Multimodal gait recognition with CNNs . . . . .	57
5.1.1. Proposed approach . . . . .	58
5.1.1.1. Input data . . . . .	59
5.1.1.2. CNN architectures for gait signature extraction . .	61
5.1.1.3. Single modality . . . . .	65
5.1.1.4. Multiple modalities . . . . .	65
5.1.2. Implementation details . . . . .	67
5.1.3. Performance evaluation . . . . .	68
5.1.4. Experimental results on TUM-GAID . . . . .	69
5.1.4.1. Architecture and feature evaluation . . . . .	70
5.1.4.2. Feature fusion . . . . .	73
5.1.4.3. State-of-the-art on TUM-GAID . . . . .	74
5.1.5. Experimental results on CASIA-B . . . . .	76
5.1.5.1. Architecture and feature evaluation . . . . .	78
5.1.5.2. Feature fusion . . . . .	79
5.1.5.3. State-of-the-art on CASIA-B . . . . .	80
5.1.6. Conclusions . . . . .	81
5.2. Incremental learning for gait recognition . . . . .	83
5.2.1. Our model . . . . .	83
5.2.1.1. Representative memory . . . . .	84
5.2.1.2. Deep network . . . . .	85
5.2.2. Incremental learning . . . . .	87
5.2.3. Experiments . . . . .	89
5.2.4. Results . . . . .	90
5.2.5. Conclusions . . . . .	91



<b>6.- Conclusions</b>	<b>93</b>
6.1. Conclusions . . . . .	93
6.2. Future Work . . . . .	95
 <b>Appendices</b>	 <b>97</b>
<b>A.- Resumen en español</b>	<b>97</b>
A.1. Publicaciones . . . . .	103
A.1.1. Referencia [89] ‘On how to improve tracklet-based gait re- cognition systems’ . . . . .	105
A.1.2. Referencia [18] ‘Multimodal features fusion for gait, gender and shoes recognition’ . . . . .	105
A.1.3. Referencia [21] ‘Fisher motion descriptor for multiview gait recognition’ . . . . .	106
A.1.4. Referencia [17] ‘Energy-based Tuning of Convolutional Neu- ral Networks on Multi-GPUs’ . . . . .	107
A.1.5. Referencia [88] ‘Deep multi-task learning for gait-based bio- metrics’ . . . . .	108
A.1.6. Referencia [20] ‘End-to-End Incremental Learning’ . . . . .	108
A.2. Conclusiones . . . . .	109
A.3. Trabajo Futuro . . . . .	110
 <b>Bibliography</b>	 <b>113</b>



# List of Figures

1.1. Same scenario recorded with different kinds of sensors. . . . .	3
2.1. Hand-crafted features pipeline . . . . .	11
2.2. Different pre-processing operations applied to the same image . . .	12
2.3. Different low-level features . . . . .	13
2.4. Bag of words . . . . .	16
2.5. Fisher Vectors . . . . .	17
2.6. Example of a generic CNN architecture, where the input is a set of images and the output is a probability distribution with as many elements as classes included in the system . . . . .	21
2.7. Activation functions . . . . .	23
2.8. Mach bands . . . . .	24
2.9. Incremental model. Given an input image, the feature extractor produces a set of features which are used by the <i>classification layers</i> (CLi blocks) to generate a set of <i>logits</i> . Grey <i>classification layers</i> contain old classes and their <i>logits</i> are used for distillation and classification. The green <i>classification layer</i> (CLN block) contains new classes and its <i>logits</i> are involved only in classification. . . . .	39
2.10. Wires, slots, cables and connectors for measuring energy on GPUs.	41
5.1. <b>Pipeline for gait recognition.</b> a) The input is a sequence of RGB-D video frames. b) Low-level features are extracted along the sequence and stacked building volumes. c) Volumes are passed through the CNN to obtain gait signatures. d) CNN outputs are combined. e) A final decision is taken to output an identity. . . . .	58

- 5.2. **CNN input data.** Sample frames extracted from a subsequence of 25 frames. **(top rows)** Optical flow in  $x$ -axis and  $y$ -axis. where positive flows are displayed in pink and negative flows in blue (best viewed in color). **(bottom rows)** Gray pixels and depth maps of the same sequence. . . . . 59
- 5.3. **Proposed CNN architectures for gait signature extraction.**  
**a) 2D-CNN:** linear CNN with four 2D convolutions, two fully connected layers and a softmax classifier. **b) 3D-CNN:** 3D CNN four 3D convolutions, two fully connected layers and a softmax classifier. **c) ResNet-A:** residual CNN with a 2D convolution, three residual blocks (red boxes), an average pooling layer and a final softmax classifier. **d) ResNet-B:** residual CNN with a 2D convolution, four residual blocks (red boxes), an average pooling layer and a final softmax classifier. More details in the main text. 61
- 5.4. **Proposed set of layers for early fusion.** A concatenation layer and three fully-connected layers are followed by a softmax classifier used to directly derive an identity. More details in the main text. . 67
- 5.5. **Datasets for gait recognition.** **(left) TUM-GAID.** People walking indoors under four walking conditions: normal walking, wearing coats, carrying a bag and wearing coating shoes. Top and bottom rows show the same set of subjects but in different months of the same year. **(right) CASIA-B.** People walking indoors recorded from eleven camera viewpoints and under three walking conditions: normal walking, wearing coats and carrying a bag. . . 70
- 5.6. **Best average accuracy.** **a)** non-temporal scenarios (N, B, S) **b)** temporal scenarios (TN, TB, TS) . . . . . 71
- 5.7. **Our incremental model.** Given an input image, the feature extractor produces a set of features which are used by the *classification layers* (CLi blocks) to generate a set of *logits*. Grey *classification layers* contain old classes and their *logits* are used for distillation and classification. The green *classification layer* (CLN block) contains new classes and its *logits* are involved only in classification. (Best viewed in color.) . . . . . 84
- 5.8. **Incremental training.** Grey dots correspond to samples stored in the representative memory. Green dots correspond to samples from the new classes. Dots with red border correspond to the selected samples to be stored in the memory. (Best viewed in color.) 87

# List of Tables

- 5.1. **Feature selection on TUM-GAID *Gray*-modality.** Percentage of correct recognition by using *rank-1* (R1) and *rank-5* (R5) metrics. Each row corresponds to a different classifier and modality. Each column corresponds to a different scenario. Best average results are marked in bold. . . . . 71
  
- 5.2. **Feature selection on TUM-GAID *OF*-modality.** Percentage of correct recognition by using *rank-1* (R1) and *rank-5* (R5) metrics. Each row corresponds to a different classifier and modality. Each column corresponds to a different scenario. Best average results are marked in bold. . . . . 72
  
- 5.3. **Feature selection on TUM-GAID *Depth*-modality.** Percentage of correct recognition by using *rank-1* (R1) and *rank-5* (R5) metrics. Each row corresponds to a different classifier and modality. Each column corresponds to a different scenario. Best average results are marked in bold. . . . . 72
  
- 5.4. **Fusion strategies in TUM-GAID with 2D-CNN.** Percentage of correct recognition for different modalities and fusion methods. Each row corresponds to a different fusion strategy. Best results are marked in bold. . . . . 74
  
- 5.5. **Fusion strategies in TUM-GAID with 3D-CNN.** Percentage of correct recognition for different modalities and fusion methods. Each row corresponds to a different fusion strategy. Best results are marked in bold. . . . . 75



5.6. <b>Fusion strategies in TUM-GAID with ResNet.</b> Percentage of correct recognition for different modalities and fusion methods. Each row corresponds to a different fusion strategy. Best average results are marked in bold. . . . .	76
5.7. <b>State-of-the-art on TUM GAID.</b> Percentage of correct recognition on TUM-GAID for diverse methods published in the literature. Bottom rows of each modality correspond to our proposal, where instead of using video frames at $640 \times 480$ , a resolution of $80 \times 60$ is used. Each column corresponds to a different scenario. Best results are marked in bold. (See main text for further details). . . . .	77
5.8. <b>Feature selection on CASIA-B 90°: <i>Gray</i> and <i>OF</i> modalities.</b> Percentage of correct recognition by using <i>rank-1</i> (R1) and <i>rank-5</i> (R5) metrics. Each row corresponds to a different classifier and modality, grouped by architecture. Each column corresponds to a different scenario. Best average results are marked in bold. . . . .	78
5.9. <b>Fusion strategies in CASIA-B 90°.</b> Percentage of correct recognition with different fusion methods. Each row corresponds to a different fusion method, but the two top rows that correspond to the baseline cases. Best average results are marked in bold. . . . .	80
5.10. <b>State-of-the-art on CASIA-B, camera 90°.</b> Percentage of correct recognition for several methods on camera 90°. Bottom rows of each modality correspond to our proposal, where instead of using video frames at $640 \times 480$ , a resolution of $80 \times 60$ is used. Acronyms: ‘#subjs’ number of subjects used for test; ‘#train’ number of sequences per person used for training; ‘#test’ number of sequences per person used for test. Best results are marked in bold. . . . .	82
5.11. <b>Incremental learning results for experiment 1.</b> Percentage of correct recognition with different step size. Each row represents a different model. Each column represents a different step size. More details in the main text. . . . .	91
5.12. <b>Incremental learning results for experiment 2.</b> Percentage of correct recognition with different step size. Each row represents a different model. Each column represents a different step size. More details in the main text. . . . .	91

# 1 Introduction

---

The word *gait* refers to the way a person walks. The gait has been studied since a long time ago, specially, applied to medical and biomechanical analysis. In the 1890s, Christian Wilhelm Braune and Otto Fischer performed an analysis of the human gait under loaded and unloaded conditions [37]. With this analysis, they optimized the equipment of the German infantry. Later, in the 1970s, Morrison [102] analyzed the gait focusing on physical parameters such as joint angles, joint movements, etc., and how the different parts affect the walking process, parameterizing the movement of the limbs involved in the gait. In the last years, gait analysis is attracting attention in medical environments because it is a non-invasive method useful to detect diseases affecting the locomotion of a subject. Some examples are Parkinson's disease [101], scoliosis [69] or osteoarthritis [99]. Very related to the medical environment, we found the performance analysis in sports, which aims to provide better equipment (*e.g.* shoes, bikes, etc.) to elite athletes. In this field, gait analysis helps to develop better racing shoes [54] or better running techniques [34] to maximize the performance of the athletes.

Recently, the gait is being used as a biometric pattern to identify people. Actually, humans are good recognizing people at certain distance, what provides a non-invasive approach to identify people without their collaboration. Most of the biometric approaches used to identify people require the collaboration of the subject, for example, putting a finger in a sensor (fingerprint recognition [86]), looking at a camera (iris [29] or face recognition [156]). However, gait recognition does not require this collaboration and could be applied in the context of video surveillance, ranging from control access in restricted areas to early detection of persons of interests as, for example, v.i.p. customers. Even, gait recognition has been used in criminal investigation [85] to identify a perpetrator using the gait

related probes collected in a crime scene.

In all aforementioned scenarios, the general process of study consists of three main steps: *data acquisition*, *data analysis* and *decision making* according to the acquired data.

The data acquisition step has evolved with the advent of the new technologies. At the beginning of the gait studies, the data acquisition was handmade, that is, the researcher took notes from the raw data observed with his or her eyes. This methodology limited the quantity and the quality of the acquired data, specially in those experiments where the observed subject was in movement, as it was very dependent of the visual and writing capabilities of the researcher. Moreover, the short-term memory or active memory of the researcher played an important role as the more capacity of the active memory, the more data could be annotated. By the end of the 19th century, the first rudimentary sensors appeared and they made the data acquisition easier. These first sensors were mechanical and inaccurate but they allowed to discretize the movement. Then, at the end of the 20th century, the computer appears and many different sensors were developed to measure different physical modalities (*e.g.* infrared, depth, visual, audio). An example of different outputs obtained from different sensors can be seen in Figure 1.1, where the same scene is recorded with three different sensors (normal camera, gray-scale camera and depth sensor). Combining a computer with those sensors, researchers could record the information of the movement with higher sampling frequency, what allowed more detailed analysis. Moreover, the different kind of sensors gave new information that was impossible to perceive by the human eye. Thus, all those new devices improved the data acquisition.

The data analysis, like the data acquisition, was handmade until the computers appeared. As computing capabilities of computers grew, more complex analyses could be performed. Thus, basic analyses such as statistical models obtained from tens or hundreds of samples, evolved to more complex methods that were trained on thousands or tens of thousands of samples. Researchers realized that the more data and the more complex models were used, the better results were obtained. This led to a race to get more data to build larger models that in turn needed more and more data. Today, huge models trained with thousands of millions of samples produce outputs that were unavailable some years ago. So, what will be the limit of this race without control? Probably, the size of the databases will reach a limit due to the costs of creating and maintaining them. The same will happen with the size of the models due to the hardware resources necessary to train them in reasonable times. The only exceptions will be large companies with millions of users using their services (*e.g.* Facebook or Google). This makes available an infinity of data and, thanks to their great benefits, they



can afford the computational resources to manage them.

Related to the analysis step, we have the decision making step that, unlike the other two previous ones, has not changed substantially. Although there is more available data and more complex analyses can be performed, the main objectives that led to develop those analysis are essentially the same. Roughly, there are three main kinds of outputs or decisions: classification, regression and detection; and those are the same for problems from the 19th century and for problems for the 21th century. However, it is true that today, we are facing more ambitious problems thanks to computers, new sensors and analysis, and what used to be image segmentation or classification, today is action recognition or object detection and tracking in a video.

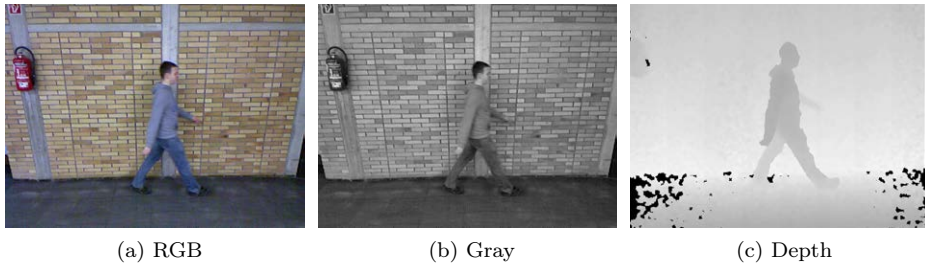


Figure 1.1: Same scenario recorded with different kinds of sensors.

## 1.1. Motivation

Access control and security in general are gaining importance during the last years. Most of the solutions developed to solve those problems are based on biometric identification using a wide range of physical characteristics like iris, fingerprint, face, etc., but all these approaches have weaknesses. On the one hand, some of them require the collaboration of the subject to be identified (*e.g.* iris or fingerprint recognition) or, on the other hand, they are very sensitive to changes in the shape (*e.g.* face recognition). However, gait recognition is a non-invasive way to implement security controls which does not require collaboration of the subject.

Focusing on the gait recognition problem, most of the works available in the literature are based on silhouette representations [47]. This kind of input is very weak against changes in the shape of the subject, specially to changes in

the clothes or the view-point. Therefore, the great interest on gait recognition combined with the limitations of the previous approaches in the literature, created the following motivations for our studies:

- To develop a more robust method against changes in the shape of the subjects using a different kind of input.
- Be able to identify people in a realistic scenario with multiple cameras at different points of view. In this regard, our approach must be able to identify people without knowing the view-point beforehand.
- Due to the advent of deep learning, we decided to create one or more approaches based on deep learning techniques to compare them with hand-crafted approaches.
- Take advantage of the different data sources and information about the subjects available today in many datasets and real life.

## 1.2. Objectives and phases

The main purpose of this thesis is to develop a new approach for multi-view gait recognition that uses optical flow as main input instead of silhouettes. This input provides a richer information about the movement of the subject while he or she walks in a video and it allows to build better representations and models. In addition, as the representation focus on the movement, it is more robust against changes in the shape of the subjects. As part of this main objective, we have defined a set of specific objectives enclosed to it:

1. To develop an approach based on hand-crafted features, specifically, using optical flow as input to build a low-level representation and, finally, a high-level representation which is used to train a traditional classifier. If there were available multiple kinds of input data, all that input information will be used by the model.
2. To develop a second approach based on deep learning building architectures for the gait recognition problem. In this case, we plan to explore the best kinds of architectures available in the literature for other tasks. Like in the previous objective, if there were available multiple kinds of input data, all that input information will be used by the model.

3. To develop an incremental learning methodology to add new classes to previously trained models keeping only a small subset of samples from the old classes. This methodology will be useful for fast addition of a new class in a real-life scenario. In that situation, each incremental training step of new classes will require a small subset of training data and much less training time.
4. To perform a power consumption study in deep learning models to characterize them in an unexplored field which is very important from a monetary point of view.

To achieve these objectives, the following phases have been carried out:

1. We reviewed the state-of-the-art to select the best approaches and the datasets used to evaluate our method and compare it with other state-of-the-art approaches. According to the published data, size of the datasets and objectives proposed, we decided to use AVAMVG [30], MoBo [45], CASIA-B and C [151] and TUM-GAID [52] datasets.
2. To develop the hand-crafted approach, we selected the low-level and high-level representations to encode the movement and information about the gait. Finally, the video representation is fed into a classifier to obtain the identity of a subject. As there were available multiple types of inputs in the used datasets, we explored different fusion schemes to combine the information and to improve the results of our baseline approach. We divided the fusion into two different schemes: late fusion which is performed with the probabilities of the classifiers, and early fusion which is performed at feature level before the classifier. For each one of the schemes, we performed three different fusion approaches.
3. To develop the deep learning approach, we selected a baseline architecture based on AlexNet [71] and we adapted it to the gait recognition problem. We performed a comparison with other two kinds of architectures: ResNet [49] and a network based on 3D convolutions [130]. The same experiments are executed for all networks to find the best one for the problem of gait recognition. As in the previous point, we evaluated different fusion schemes for the deep learning approach. In this case, we used three late fusion approaches and one early fusion approach. Note that, as the deep learning method builds its own representation, the early fusion can only be done in the definition of the architecture. In our case, we concatenated the features coming from different kinds of inputs at different levels in the

deep architecture. In this step, we also decided to explore the benefits of training a deep learning approach with a combined loss function to perform multiple tasks at the same time.

4. We studied the gait recognition problem from an incremental learning point of view. In this case, we wanted to add more classes to a trained model without re-training from scratch. Thus, we combined two loss functions, one to retain the previous knowledge and the other one to learn new information from the new classes.
5. Finally, we analyzed the energy consumption during the training process of different kinds of CNN architectures and the impact of the hyper-parameters in the power consumption and accuracy of the models. The energy consumption has been measured physically with an external device connected to the GPUs used for running the training code.

In this list of phases, phase 1 is necessary for all objectives since the dataset are used in all of them. Phase 2 and 3 accomplished objectives 1 and 2, respectively. Finally, phases 4 and 5 carried out objectives 3 and 4.

### 1.3. Thesis contributions

We have developed different approaches using different techniques to find the best possible solution. Therefore, we have tried to solve the problem starting from different points of view taking advantage of the last techniques developed in other fields and the information included in the datasets. Thus, we have developed approaches based on hand-crafted features and deep learning. Moreover, we have proposed different fusion schemes to take advantage of multiple sources of information. Finally, to make more attractive our approaches to the industry, we have developed an incremental learning approach which is able to learn new classes incrementally, and a power consumption study of our deep learning approach.

Therefore, the contributions of this thesis according to the proposed objectives are:

- For objective 1, hand-crafted approach, the contributions are:
  1. A new gait descriptor (PFM) that combines the potential of HAR descriptors with the rich representation provided by FV encoding. This

new descriptor uses optical flow as input instead of the common silhouettes used by most of the published works.

2. The proposed descriptor is able to deal with multiple viewpoints at test time, even if the approach is trained with different viewpoints to the tested ones.
  3. An unified approach for using audio, depth and visual information for the problem of gait recognition.
  4. A multimodal-based approach for gender and shoes recognition.
  5. State-of-the-art results on AVAMVG, MoBo, CASIA-B, CASIA-C and TUM-GAID datasets.
- For objective 2, deep learning approach, the contributions are:
    1. A preprocessing stage to extract, organize and normalize low-level motion features for defining the input data.
    2. A set of CNN architectures that can be used directly as classifiers or to extract discriminative gait signatures from low-level motion features (*i.e.* optical flow).
    3. The proposed approach is able to deal with multiple viewpoints at test time, even if the approach is trained with different viewpoints to the tested ones.
    4. A set of CNN architectures for data fusion.
    5. A multi-task approach that is able to produce multiple kind of outputs (*i.e.* identification, gender and age) from a single input.
    6. State-of-the-art results on TUM-GAID and CASIA-B datasets.
  - For objective 3, incremental learning approach, the contributions are:
    1. An end-to-end approach designed specifically for incremental learning. This approach can be realized with any deep learning architecture.
    2. State-of-the-art results on CIFAR-100 [70] and ImageNet [71].
  - For objective 4, energy consumption study, the contributions are:
    1. A combined energy and performance analysis on a multi-GPU setup using the two most popular types of CNNs, and particularized for the forward, backward and weight update stages of a deep learning algorithm.

2. Accuracy statistics to find out the best algorithm parametrization depending on three different metrics: time, energy consumption and energy-delay product.
  3. Comparison between Maxwell and Pascal architectures for all those features above.
- A public MATLAB's implementation of all approaches <sup>1</sup>.

## 1.4. Thesis organization

This thesis is organized as follows. Chapter 2 describes the principles of the different algorithms used in the thesis. Chapter 3 reviews the state-of-the-art of gait recognition. Chapter 4 contains the published works that support this thesis with a brief summary of each one of them. Chapter 5 contains the work that is not published yet. Finally, the conclusions of this thesis and the future work are presented in Chapter 6.

---

<sup>1</sup><https://github.com/fmcp/>

# 2 Background

---

In this chapter we present some background on machine learning based on hand-crafted features, deep learning, information fusion schemes, incremental learning and energy consumption in deep learning approaches. Although there exist many different approaches for each developed topic, this chapter focuses on those works directly related to our contributions.

Section 2.1 and Section 2.2 provide background on hand-crafted approaches and deep learning approaches, discussing the different ways to solve the problem depending on the kind of input of the system. Section 2.3 and Section 2.4 describe the bases of information fusion and incremental learning using hand-crafted or deep learning approaches. Finally, Section 2.5 talks about optimization of deep learning approaches from an energy consumption point of view.

## 2.1. Machine learning based on hand-crafted features

Hand-crafted features are those features which are obtained from manually designed operations. These operations can be a convolution, a morphological operation or any other operation which takes an image or video as input and produces a set of new information. Although there exists a wide range of operations, all of them share a common characteristic, they are manually designed for a specific problem. This is a huge difference compared to deep learning approaches, described in Section 2.2, where the operations are self-learned.

In these kinds of approaches, the input data is passed through a pipeline to

obtain the final output which, depending on the problem, can be a probability distribution among classes (classification problem), a number or set of numbers (regression problem), etc. Although, there are many different problems, in general, this pipeline is composed of a set of common operations to any computer vision problem. Figure 2.1 shows a general sketch with the basic operations composing this pipeline which will be described below. Those operations can be summarized in a *pre-processing step* to prepare the input data for further operations, a *low-level features computation* to describe basic information of the input, a *high-level features computation* to describe the global input, and a final step to *solve the problem* (classification, regression, etc.).

In this thesis we designed a new approach based on dense trajectories [62] and Fisher vectors [110] to identify people using their way of walking. To the best of our knowledge, our work is pioneer using this setup for gait recognition.

In the next sections those steps will be described in detail, paying special attention to specific approaches used in the developed methods of this thesis. Note that in the following, the word features or descriptors are going to be used indistinctly as they have the same meaning.

### 2.1.1. Data pre-processing

This is usually the first step done in any computer vision problem. The main objective of this step is to prepare the data to obtain better results. Usually, the information recorded by any sensor contains noise or artifacts due to the physical conditions of the scene or limitations of the sensor itself. Therefore, in an ideal world, this noise should not exist and this step would be unnecessary. However, the noise is present in all signals and the approaches must deal with it trying to remove it or, at least, mitigate it as much as possible.

The pre-processing not only has to deal with noise, it also can be used to prepare the data used by the algorithm. Common operations are: balancing of the training set so all classes have a similar number of samples, removing duplicate data, mean subtraction, data compression, data augmentation, etc. Therefore, after these operations, the information will be better used by the algorithm.

The operations executed may differ depending on the kind of input data and the kind of problem. Thus, for images, RGB pictures are usually transformed into gray-scale ones to compress the information and save time and computational resources. For video, it is usual to resize the video into a smaller scale to save resources and time. To remove noise in the images, the most common operation is gaussian blurring which decreases the noise but also reduces the details of the



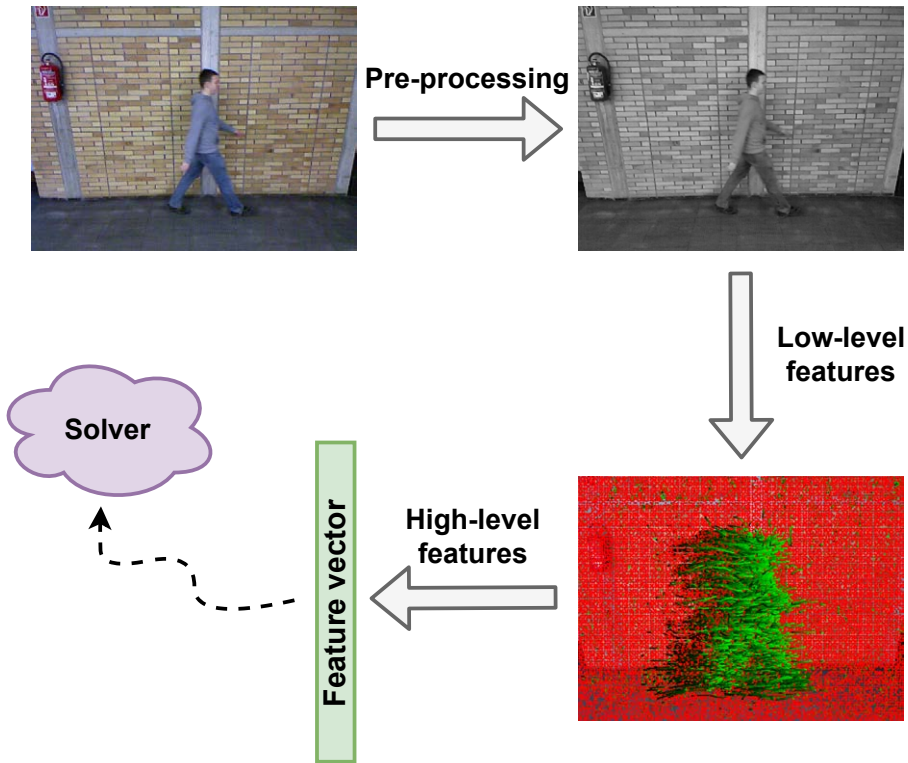


Figure 2.1: Hand-crafted features pipeline

image.

Figure 2.2 shows the output of three different operations applied to the same frame. An example of applying a gaussian blur can be observed in Figure 2.2a. Figure 2.2b contains the image in gray-scale. Finally, Figure 2.2c shows the mirror image as an example of data augmentation.

### 2.1.2. Low-level descriptors

This step takes the input coming from the pre-processing step (Section 2.1.1) to produce a set of descriptors, that is, information which describes the scene observed in an image or video. By this way, the visual information stored in the input data is summarized and structured to make easier the search for the solution in the explored problem.

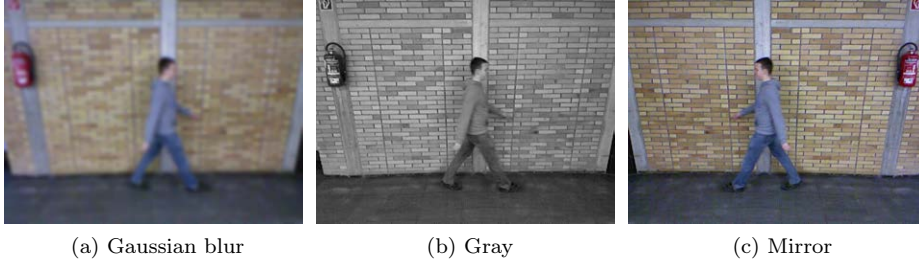


Figure 2.2: Different pre-processing operations applied to the same image

To build low-level descriptors, two sub-steps must be performed. The first step involves the detection of the most important points or regions in the input. Thus, the operations to compute the descriptors will focus on those regions producing better features which describe the important visual information, ignoring the superfluous and unnecessary data. Sometimes this process requires a long time or it is very difficult to perform due to specific characteristics of the input data, so, there exist approaches where the detection process is replaced by a dense grid of points equally distributed along the input. As all image information is taken into account with this approach, the execution time is negligibly but the amount of information is huge and the features obtained can be worse than when a detection step is performed. Assuming this disadvantage can be mitigated during the construction of high-level descriptors (Section 2.1.2), a good policy is implemented to filter out the unnecessary information.

Figure 2.3 shows two different approaches for detecting important regions on images or videos. Figure 2.3a shows the output of the minimum eigenvalue algorithm for corners detection. Green points represent the important regions, and, as the algorithm detects corners, they are located on the bricks and borders of the person. A dense grid can be observed in Figure 2.3b. Red dots represent the points of the grid and the green lines show the movement along time of the grid points. Note that this detection has been applied to a video to track the movement of the subjects appearing in the scene. Comparing both images, it is clear the huge difference of detecting the important parts of the image or using a dense grid. While in the first image the amount of points is relatively small, in the second image almost the whole image is covered by points of the grid, what produces larger descriptors.

Once the important parts are located, they have to be described in an understandable way by the computer. The main objective is to represent the visual

information of each important region and its neighbourhood in a way that should be invariant to rotations, scale, etc. That is the most important point when researchers develop new descriptors. For example, when people describe an image, they do not care about the scale or the rotation of the image with respect to their point of view. Therefore, the description will be the same for an image of 10x15cm and for an image of 100x150cm. Although this process is easy for a human, for a computer it is really hard because for bigger images, more information is taken into account when the descriptor is built. Therefore, the descriptors for images with different sizes will be different although the information is essentially the same. However, there exist descriptors like SIFT [84] which are invariant to some image transformations such as rotations or scale. Then, it is important to develop or use invariant descriptors during the design of the pipeline of the approach. Some examples of common descriptors used in the literature are SIFT [84], HOG [28], HOF [111] or DCS [62], being the first one applied to images and the rest for videos.

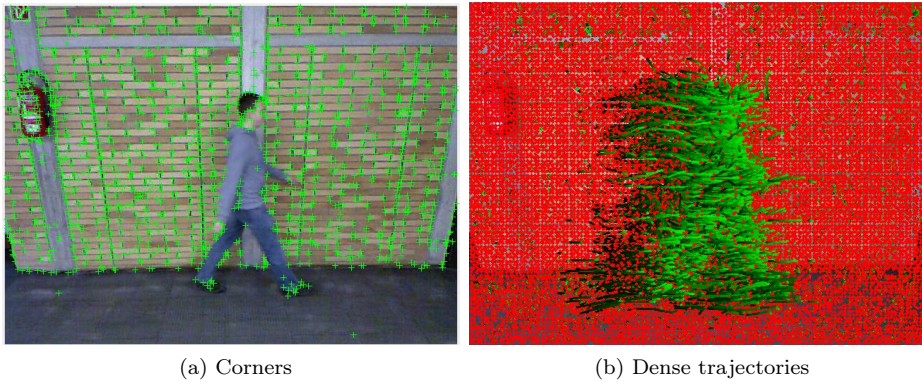


Figure 2.3: Different low-level features

### 2.1.2.1. Dense trajectories and DCS

In this section, the use of dense trajectories and DCS as low-level descriptors is explained in detail since they are used in an approach developed in this thesis (Section 4.1).

The first step is to compute densely sampled trajectories. Those trajectories are computed by following the approach of Wang et al. [136]. Firstly, dense optical flow  $F = (u_t, v_t)$  is computed [36] on a dense grid (*i.e.* step size of 5

pixels and over 8 scales). Then, each point  $p_t = (x_t, y_t)$  at frame  $t$  is tracked to the next frame by median filtering as follows:

$$p_{t+1} = (x_{t+1}, y_{t+1}) = (x_t, y_t) + (M * F)|_{(\bar{x}_t, \bar{y}_t)} \quad (2.1)$$

where  $M$  is the kernel of median filtering and  $(\bar{x}_t, \bar{y}_t)$  is the rounded position of  $p_t$ . To minimize drifting effect, the tracking is limited to  $L$  frames. As a post-processing step, noisy and uninformative trajectories (*e.g.* excessively short or showing sudden large displacements) are removed. These short-term trajectories (or tracklets) are represented in Figure 2.3b by green lines for each considered point (in red).

Once the local trajectories are computed, they are described with the Divergence-Curl-Shear (DCS) descriptor proposed by Jain et al. [62], which is computed as follows:

$$\left\{ \begin{array}{l} \text{div}(p_t) = \frac{\partial u(p_t)}{\partial x} + \frac{\partial v(p_t)}{\partial y} \\ \text{curl}(p_t) = \frac{-\partial u(p_t)}{\partial y} + \frac{\partial v(p_t)}{\partial x} \\ \text{hyp}_1(p_t) = \frac{\partial u(p_t)}{\partial x} - \frac{\partial v(p_t)}{\partial y} \\ \text{hyp}_2(p_t) = \frac{\partial u(p_t)}{\partial y} + \frac{\partial v(p_t)}{\partial x} \end{array} \right. \quad (2.2)$$

As described in [62], the divergence is related to axial motion, expansion and scaling effects, whereas the curl is related to rotation in the image plane. From the hyperbolic terms ( $\text{hyp}_1, \text{hyp}_2$ ), we can compute the magnitude of the shear as:

$$\text{shear}(p_t) = \sqrt{\text{hyp}_1^2(p_t) + \text{hyp}_2^2(p_t)} \quad (2.3)$$

Then, those kinematic features are combined in pairs as in [62] to get the final motion descriptors.

### 2.1.3. High-level descriptors

While low-level descriptors focus on features like corners or edges, high-level descriptors focus on describing the image as a whole, like humans do. In this step, the low-level features produced in the previous step are used as input to

produce a new set of features to describe the global input. Although this step seems repetitive with respect to the previous one, it is necessary to summarize the low-level information for the classification step. Therefore, without this step, the algorithm would not be able to abstract from small details to understand the high level meaning of the input.

Like with low-level descriptors, the detection and describing processes must be performed to produce the high-level descriptors. In this case, the detection process does not look into visual details in the input since it uses a set of low-level features. Therefore, the detection process usually performs an unsupervised procedure known as clustering, to find out the different patterns present in the data. Before doing this operation, it is necessary to compute a dictionary of patterns in a training set. This dictionary will describe the different patterns present in the training images. Therefore, at test time, the low-level features of the image will be compared with the patterns of the dictionary to select the closest one.

With the detected patterns, the representation process constructs a final descriptor which describes the whole image. Usually, this high-level descriptor is a vector whose length depends on the approach. Then, after this step, the input image has been transformed into a single vector which is able to describe the scene of the input.

To illustrate this process, Figure 2.4 shows a sketch of the process performed to compute the widely used Bag of Words (BoW) descriptor [126]. In this case, the dictionary is obtained with the K-Means algorithm [81] which is also used to find the closest patterns during test time. Those patterns are defined by the centroids obtained with the K-Means algorithm. Then, to build the feature vector for a new sample, a clustering operation is performed over the low-level features to find the closest patterns of the dictionary. After that, the feature vector is represented by the histogram of occurrences of each pattern. In this case, the length of the feature vector depends on the number of patterns or centroids of the dictionary.

### 2.1.3.1. Fisher Vectors

In this section, the use of Fisher Vectors as high-level descriptors is explained in detail as they are used in an approach developed in this thesis (Section 4.1). As described above, our low-level features are based on motion properties extracted from person-related local trajectories. In order to build a high-level gait descriptor, we need to summarize the local features. We propose here the use of

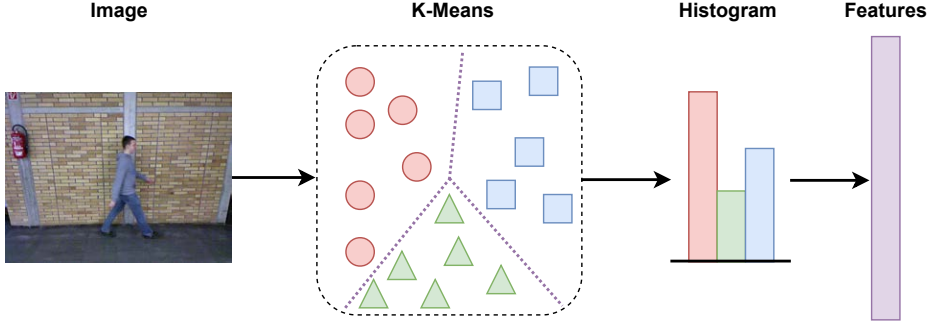


Figure 2.4: Bag of words

Fisher Vectors (FV) encoding [110].

The FV, that can be seen as an extension of the Bag of Words (BOW) representation [126], builds on top of a Gaussian Mixture Model (GMM), where each Gaussian corresponds to a visual word. Whereas in BOW, an image is represented by the number of occurrences of each visual word, in FV an image is described by a gradient vector computed from a generative probabilistic model. The dimensionality of FV is  $2ND$ , where  $N$  is the number of Gaussians in the GMM, and  $D$  is the dimensionality of the local motion descriptors  $x_t$ . In this thesis, we will use the term *Fisher Motion* (FM) to refer to the FV computed on a video from low-level motion features.

Assuming that our local motion descriptors  $\{x_t \in R^D, t = 1 \dots T\}$  of a video  $V$  are generated independently by a GMM  $p(x|\lambda)$  with parameters  $\lambda = \{w_i, \mu_i, \Sigma_i, i = 1 \dots N\}$ , we can represent  $V$  by the following gradient vector [108]:

$$G_\lambda(V) = \frac{1}{T} \sum_{t=1}^T \nabla_\lambda \log p(x_t|\lambda) \quad (2.4)$$

where  $T$  is the total number of local descriptors and  $\nabla_\lambda$  denotes the gradient operator with respect to  $\lambda$ .

Following the proposal of [110], to compare two videos  $V$  and  $W$ , a natural kernel on these gradients is the Fisher Kernel:  $K(V, W) = G_\lambda(V)^T F_\lambda^{-1} G_\lambda(W)$ , where  $F_\lambda$  is the Fisher Information Matrix. As  $F_\lambda$  is symmetric and positive definite, it has a Cholesky decomposition  $F_\lambda^{-1} = L_\lambda^T L_\lambda$ , and  $K(V, W)$  can be rewritten as a dot-product between normalized vectors  $\Gamma_\lambda$  with:  $\Gamma_\lambda(V) = L_\lambda G_\lambda(V)$ . Then,  $\Gamma_\lambda(V)$  is known as the Fisher Vector of video  $V$ . As stated in [110], the capability of description of the FV can be improved by applying it a signed

square-root followed by L2 normalization. So, we adopt this finding for our descriptor.

To clarify this process, Figure 2.5 shows a sketch of the process performed to compute Fisher Vectors. In this case, the dictionary is obtained with a Gaussian Mixture Model. Those patterns are defined by the means and variances of the gaussians. Then, to build the feature vector for a new sample, a gradient vector is computed in the dictionary. In this case, the length of the feature vector depends on the number of gaussians and the size of the input features.

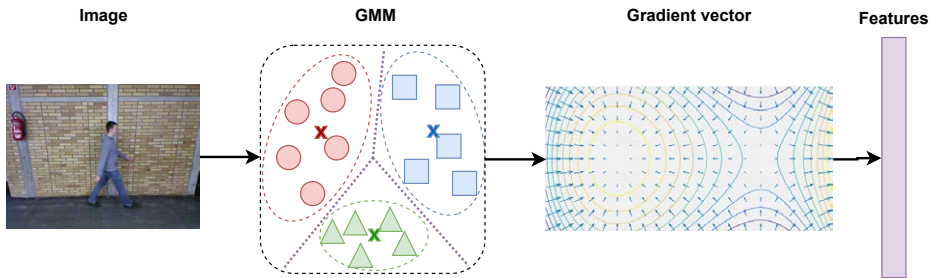


Figure 2.5: Fisher Vectors

#### 2.1.4. Classification

This is the final step and it takes the high-level descriptors to obtain categories or classes based on similarities extracted from the input samples. In a formal way, a classification algorithm maps an observation  $o$  to a class  $c$ . Note that we are going to focus on the classification as it is the problem developed on this thesis. However, there are many other kinds of tasks like regression, verification, etc, that are not described in this document.

The classification problem is an instance of supervised learning where the training set is fully annotated and those labels are available during training. There are many different kinds of classification algorithms adapted to different problems. Therefore, the selection of an adequate classifier is a critical point during the design of a computer vision approach.

Classification algorithms are usually divided into binary or multi-class algorithms depending on the number of classes allowed to use. A binary classifier only separates between two classes while a multi-class model supports more than two classes. However, binary classifiers can be extended to a multi-class setup using an one-vs-all (OvA) or an one-vs-one (OvO) strategy. In the one-vs-all strategy,



one classifier per class is trained with positive samples from the selected class and negative samples from the rest of the classes. Then, each classifier learns to separate its class from the others. In this case, the classifiers used must output a real-valued confidence score for its decision, allowing to combine decisions from multiple classifiers. In the one-vs-one, one classifier per pair of classes must be trained. Then, the number of classifiers is computed with Equation 2.5.

$$C = \frac{K(K-1)}{2} \quad (2.5)$$

Where  $K$  is the number of classes and  $C$  the number of classifiers. Each one of those classifiers is trained with positive samples from one class of the couple and negative samples from the other class of the pair. To perform an ensemble classification, a voting scheme is applied and the output class will be the most voted one.

Some examples of binary classifiers are Support Vector Machine (SVM) [27], Perceptron [118] or Linear Regressors. Regarding multi-class classifiers, the most common are Neural Networks [96], Naive Bayes [39] or Decision Trees [113].

#### 2.1.4.1. SVM

This section explains in detail the Support Vector Machines (SVM) [27] as they are used in the approach developed in this thesis. SVMs are a kind of supervised learning models used for binary classification problems. Like other binary classifiers, SVMs are used to solve linear problems where the samples are represented as points in the space and the model learns to separate samples from different classes with a line. However, SVM models can also solve non-linear problems using the kernel trick, which maps the samples to a high-dimensional space where samples from different classes are separated by a gap which is as wide as possible. In this case, the classes are separated by a hyperplane.

More formally, a SVM model is trained by minimizing the Equation 2.6.

$$\mathcal{L} = \frac{1}{2}w^T w + C \sum_{i=1}^N \xi_i \quad (2.6)$$

subject to  $y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i$  and  $\xi_i \geq 0, i = 1, \dots, N$ , where  $w$  is the vector of coefficients,  $C$  is the capacity constant,  $b$  is a constant,  $\xi_i$  represents parameters for handling non-separable data,  $N$  is the number of training samples,



$y$  represents the label of the training sample,  $x_i$  are the independent variables and  $\phi$  is the kernel used to map the input data to the feature space. With this formulation,  $w$  and  $b$  are updated during the training process until the algorithm converges.  $C$  must be selected carefully to avoid overfitting. Note that, as it is a binary problem, the possible label values are  $+1$  for a sample belonging to a class and  $-1$  for a sample belonging to another class. If the problem has multiple classes,  $P$  binary SVMs are trained (as many as different classes) in a *one-vs-all* strategy.

## 2.2. Deep learning

The term Deep Learning refers to a kind of machine learning algorithms that use a set of layers of nonlinear processing units for feature extraction and transformation, where each layer uses as input the output of the previous one. This kind of models can learn at the same time the features to represent the information at different levels and the layers used to solve the problem (classification, regression, etc). Thus, this approach can be seen as a black box which takes the input data (samples + labels) to produce an output automatically using a set of self-learned features.

This kind of models can be considered an extension of the traditional Artificial Neural Networks [96] but using more layers and, depending on the kind of architecture, a structured setup using spatial filters. The main difference with respect to traditional approaches is the number of trainable parameters. Traditional approaches have around one hundred of parameters maximum while deep models can have millions of them. Therefore, the amount of data required to optimize that large number of parameters is huge. Then, if the amount of training data is reduced, this kind of models tend to overfit having worse performance than traditional methods.

Deep models are defined by their architecture, which is a set of different layers, and the loss function, which depends on the kind of problem and it is used to compute the errors of the predictions. As there exist many kinds of deep models, this manuscript is going to focus on Convolutional Neural Networks because are used in the developed approach.

Convolutional Neural Networks (CNNs) are a kind of deep models specifically designed to work with images applying kernels or convolutions to obtain the features. In these networks, some layers have a spatial structure to simulate traditional kernels applied in convolutional operations. Then, the first layers tend

to produce low-level features focused on corners, edges, patterns, etc., and the last layers focus on high-level details with more meaning like eyes or faces (depending on the problem, the details will change). Figure 2.6 shows a general architecture of a CNN where the input frames are passed through a set of convolutions and pooling operations to produce a final vector. This vector is evaluated in the fully-connected layers (FC) and a final probability distribution is obtained.

To train a CNN, the first step is to prepare the training data and define the architecture of the network. This is a very important step to guarantee the convergence of the model. On the one hand, the data must be normalized, usually by subtracting the mean of the training data to each sample. On the other hand, the layers and loss function define the architecture and its behaviour during training and testing. The second step is the selection of the solver algorithm that will minimize the loss function to optimize the parameters of the model. Finally, the hyper-parameters of the model must be defined according to the specific characteristics of the problem. Then, the training process starts and, if everything works well, the model will converge. The training process can be split into three main steps which are performed iteratively until the model converges. The first training step is the forward pass where the input data is passed through the CNN to obtain the activations and loss error. Then, the second training step, called backpropagation, is executed to compute the partial derivatives of each parameter according to the error obtained from the loss function. These partial derivatives encode the direction and value of the parameter update needed to minimize the loss error. Finally, the parameter update is executed to minimize the loss error according to the derivatives obtained in the previous step.

In this thesis we designed a new deep learning architecture specifically for the gait recognition problem. It takes the optical flow as input to identify the subject appearing in a sequence.

Next sections explain with more details all these steps.

### 2.2.1. Layers

A layer is the fundamental part of a CNN due to it defines the behaviour of the model, learning capacity and training requirements. All CNNs are composed of a sequence of layers where every layer transforms the inputs through a differentiable function. When all layers are put together, they conform the architecture of the CNN.

Nowadays, there are many kinds of layers so this manuscript will comment only the most important layers which are more frequently used in the literature.

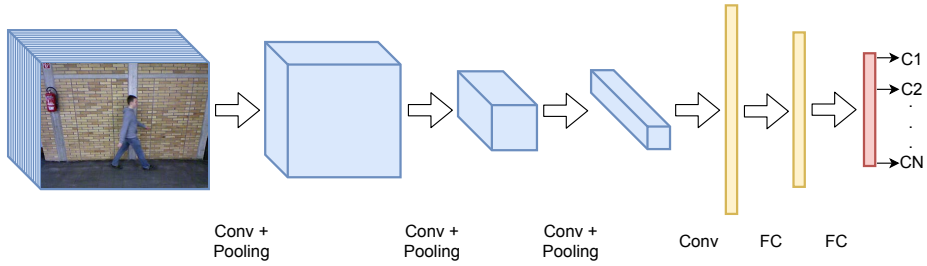


Figure 2.6: Example of a generic CNN architecture, where the input is a set of images and the output is a probability distribution with as many elements as classes included in the system

Specifically, the explained layers are convolution, pooling, normalization, fully-connected, dropout and batch-normalization.

### 2.2.1.1. Convolutional layer

Convolutional layers are the core of the CNNs and they perform most of the computations of the networks. The parameters of this layer consists of a set of learnable filters simulating a small spatial kernel. Then, during forward pass, every filter is convolved across the input sample. After this operation, each filter produces a 2-dimensional (for 2D convolutions) or 3-dimensional (for 3D convolutions) activation map with the responses to the spatial information of the input. Stacking the activations map of all filters, we obtain the output volume used as input by the following layer. Intuitively, the first layers of the network will learn filters that focus on some types of visual features such as corners or edges, on the other hand, higher layers will learn filters to focus on meaningful patterns like fingers or wheels for example.

To create a convolutional layer, a set of parameters must be defined:

- Kernel size. Defines the shape of the filter (width and height).
- Depth. Number of filters of the layer. Each filter will produce an activation map that will be stacked to conform the activation volume.
- Stride. Step size of the sliding window over the input data. Big stride values will produce less local features as there are bigger gaps between windows.
- Padding. Number of zeros added to each side of the input sample. It allows to control the shape of the output volume.

Then, the shape (width or height) of the output volume produced by a layer can be computed with Equation 2.7.

$$O = \frac{I - K + 2P}{S} + 1 \quad (2.7)$$

where  $I$  is the input size (width or height),  $K$  is the kernel size (width or height),  $P$  is the padding value,  $S$  is the stride value and  $O$  the output size.

From a neural point of view, each filter is implemented by a neuron whose weights encode the kernel. To implement the sliding window used in the traditional convolution, for every step there is a neuron which computes the convolution. Thus, there are neurons distributed along the two dimensions of the input to compute the convolution at the same time. This assumes a large number of parameters to optimize what would cause problems with overfitting. To overcome this problem, the number of parameters is reduced by making an assumption: if one kernel is useful in some position  $(x, y)$ , it should be useful also in another position  $(x+2, y+2)$ . By this way, all neurons of a kernel shares the same parameters what reduces the amount of trainable weights of the network.

Finally, like in layers of traditional Neural Networks, convolutional layers are usually followed by a non-linear activation function which helps the model to generalize or adapt to the data. There exists many different activation functions but the most common are:

- Sigmoid:  $f(x) = \frac{1}{1+e^{-x}}$ . Activation values in the range  $[0, 1]$ . A problem with this activation function happens when the values saturate to 0 or 1 because the gradients at these regions are almost zero. Therefore, during backpropagation, gradients will tend to zero and the convergence will be penalized. In addition, the parameter initialization plays an important role with this function to prevent saturation.
- Tanh:  $f(x) = \frac{e^{2x}-1}{e^{2x}+1}$ . Activation values in the range  $[-1, 1]$ . Like sigmoid, its activations saturate but it is zero-centered what makes it more preferable to sigmoid.
- ReLU (Rectified Linear Unit) [103]:  $f(x) = \max(0, x)$ . Activation values in the range  $[0, \infty]$ . This function is the most used today as it accelerates the convergence and is really fast to compute compared to previous functions.

Figure 2.7 shows the activation values of the three activation functions commented above.

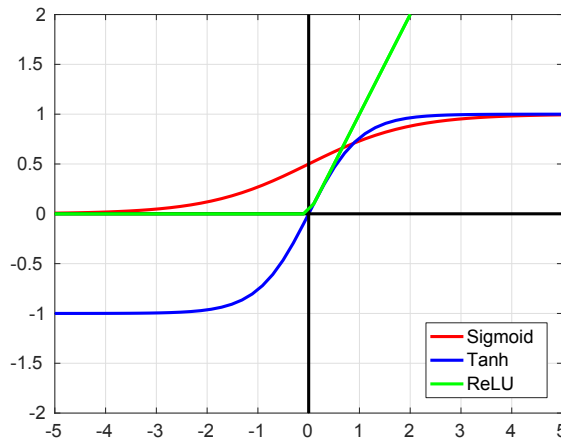


Figure 2.7: Activation functions

The backpropagation of a convolutional layer is also a convolution but with a flipped version of the filter parameters.

### 2.2.1.2. Pooling layer

The function of this layer is to reduce the shape of the input data to reduce also the number of parameters of the network to prevent overfitting. The natural position of this layer in the network is after convolutional layers to reduce the data size progressively. The computation of this layer is similar to the convolutional layer, the input traversed with a sliding window and an operation is applied to every window to obtain an output volume. In this case, the operation produces a single value for the whole window so the final size of the output is smaller than the input size.

Like for the convolutional layer, to create a pooling layer, a set of parameters must be defined:

- Window size. Defines the shape of the window (width and height).
- Stride. Step size of the sliding window over the input data. Big stride values will produce less local features as there are bigger gaps between windows.
- Padding. Number of zeros added to each side of the input sample. It allows to control the shape of the output volume.

- Operation. Operation performed to the window. The maximum is the most common operation but there are more operations such as minimum, average, L2, etc.

The backpropagation of this layer depends on the operation performed. For the *max* operation, which is the most common, the backward pass only has to route the gradient to the input with the highest value. Therefore, during forward, it is necessary to store the indices of the maximum values per window to perform the backpropagation.

### 2.2.1.3. Local Response Normalization layer

This layer simulates the lateral inhibition process that happens in the human brain. In broad terms, the lateral inhibition is the capacity of an excited neuron to reduce the activity of its neighbours. This process stops the spreading of nerve impulses from excited neurons to its neighbours in the lateral direction. This increases the contrast producing an increased sensory perception. Figure 2.8 illustrates this effect with the Match bands [83], which is an optical illusion where lateral inhibition makes the darker area falsely appear even darker and the lighter area falsely appear even lighter.

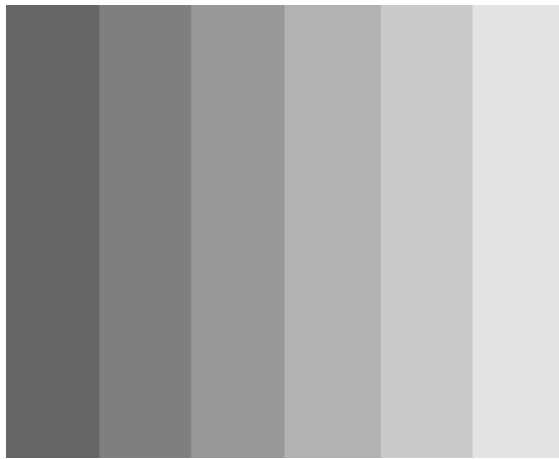


Figure 2.8: Mach bands

This effect is also useful in CNNs to boost neurons with higher activations. This is specially useful with ReLU layers because the activations are unbounded

and they must be normalized. The idea is to detect features with large activations. Then, when this normalization is applied to an excited neuron and its neighbourhood, the neuron becomes even more sensitive compared to the neighbour neurons. At the same time, it will restrict the activations that are uniformly large in a local neighborhood. If all the values are large, the normalization will reduce all of them.

Although this layer has been widely employed in the past and it is employed in state-of-the-art networks, its use has decreased in the last years as, in practice, its contribution is minimal in most of the cases.

#### 2.2.1.4. Fully-connected layer

This layer is the principal component of the traditional Neural Networks and it is also used in convolutional networks. In this layer, the neurons have connections to all activations in the previous layer unlike convolutional layers that only have local connections. This layer concentrates most of the parameters of the network and it is usually located in the later layers of the network, storing the high level knowledge used by the classifier.

Due to its large number of parameters, the amount of these layers included in a network is very limited to prevent overfitting. In fact, current networks usually have only one fully-connected layer in contrast to the two, three or even more fully-connected layers included in older models.

#### 2.2.1.5. Dropout layer

Dropout is a regularization technique used in neural networks to reduce overfitting. This layer implements this operation which is applied to the layer connected to it. During training, for each iteration, individual neurons or nodes are randomly dropped out with a probability of  $p$  or kept with a probability of  $1 - p$ . This operation is only applied during training to avoid co-dependency between neurons which curbs their individual activations leading to overfitting. During test, Dropout is not applied and all neurons of the network are taken into account.

This layer is specially useful with fully-connected layers because they concentrate most of the parameters of the network and overfitting is more significant in this kind of layers. Therefore, the natural position of this layer is after a fully-connected layer to decorrelate its neurons.

During backpropagation, this layer applies the same operation to the gradients, that is, the neurons deactivated during forward produce zero gradients during backward.

#### 2.2.1.6. Batch-normalization layer

During training, the input data is grouped into small blocks of information called batches. Thus, the information passed through the network may be very different in every iteration, penalizing the training process. For more details about batch and training process, please, check Sections 2.2.4 and 2.2.3.1. To overcome this problem, this layer normalizes the output of the layer to which it is connected. Like input data can have values in many different ranges, activations coming from previous layers can have different ranges of values. Therefore, like with input data, a normalization of the activations could speed up the convergence of the network. In addition, as the activations are normalized, the covariance shift problem is less important and the overfitting is reduced. For example, a network trained with images captured with artificial light will produce bad results with images captured with sunlight. Thus, there is a covariance shift between training and test sets. However, batch-normalization can help to overcome this problem.

To facilitate the convergence of the network, this layer is attached to every convolutional layer after the activation function to normalize the output data. Batch-normalization normalizes the activation by subtracting the mean of the batch samples and dividing by the standard deviation of the batch. In addition, there are two trainable parameters to scale and shift the data to keep fixed the mean and variance of the output values. Thus, activations are now in the same range and backpropagation has to deal with less oscillations when approaching the local minimum so it converges faster. Batch-normalization also helps to reduce the impact of earlier layers on the later layers of the network by keeping the mean and variance fixed, which makes layers more independent from the others. Moreover, it also helps to regularization, reducing overfitting, as covariance shift is minimized.

#### 2.2.2. Loss function

A loss function or cost function is a function that maps the output features of the model onto a real number that represents the cost associated to that output. During the training process, the optimizer tries to minimize that cost function to find the best possible solution. The loss function must be selected according



to the objective of the network. Thus, while for a classification problem the cost of the loss function represents the penalty applied to incorrect predictions, for a regression problem the cost of the loss function represents the distance from the predicted output to the objective.

In the following sections, the manuscript is going to focus on the classification and regression loss functions used in the developed approaches.

### 2.2.2.1. Cross-Entropy loss function

A Cross-Entropy loss function quantifies the difference between two probability distributions, the predicted and the ground-truth distribution. For this loss function, the ground-truth or label distribution must be encoded as an one-hot vector. That is, the vector which encodes the probability distribution has as many elements as classes and only the correct label is set to 1, the rest are set to 0.

Equation 2.8 is used to compute the cross-entropy loss for a multi-class setup.

$$\mathcal{L}(p, q) = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^C p_{ij} \log q_{ij}, \quad (2.8)$$

where  $q_i$  is a probability obtained by applying a softmax function to the *logits* of a classification layer for the sample  $i$ ,  $p_i$  is the ground-truth for the sample  $i$ ,  $N$  is the total number of samples and  $C$  the total number of classes.

### 2.2.2.2. Tukey's biweight loss function

Tukey's biweight loss function [11] is a robust regression loss function that reduces the influence of outliers during the learning process. The loss  $\mathcal{L}(\hat{y}_i, y_i) = \rho(r_i^{\text{MAD}})$  is based on the residual  $r_i$  of the training samples  $r_i = y_i - \hat{y}_i$  and is defined by the equations:

$$\rho(r_i) = \begin{cases} \frac{c^2}{6} [1 - (1 - (\frac{r_i}{c})^2)^3] & , \text{ if } |r_i| \leq c \\ \frac{c^2}{6} & , \text{ otherwise} \end{cases} \quad (2.9)$$

$$r_i^{\text{MAD}} = \frac{y_i - \hat{y}_i}{1.4826 \times \text{MAD}} \quad (2.10)$$

$$\text{MAD} = \text{median}_{k \in \{1, \dots, S\}} \left( \left| r_k - \text{median}_{j \in \{1, \dots, S\}} (r_j) \right| \right) \quad (2.11)$$

where constant  $c$  is set to 4.6851 as suggested in [11],  $S$  is the number of training samples, and the subscripts  $k$  and  $j$  are indexes referred to the training samples.

This layer may be replaced by another regression-like loss function, as the L2 distance:  $\mathcal{L}_a(\hat{y}_i, y_i) = \|\hat{y}_i - y_i\|_2^2$ . However, the performance of the Tukey's loss is better than L2 as it is more robust to outliers.

### 2.2.3. Optimizers

During training of a network, its parameters must be optimized to solve the problem for which it has been designed. This process implies to compute the gradients for a loss function and apply those gradients to update the parameters. The optimizer is the algorithm which performs those two operations to optimize the weights of the model.

Gradient descent is the most popular algorithm used to perform the optimization process. It minimizes an objective function  $J(\theta)$ , which is parameterized by the weights of the model  $\theta \in \mathbb{R}^d$ , by updating the weights in the opposite direction of the gradient of the objective function  $\nabla_{\theta} J(\theta)$  with respect to the parameters.

The following sections will explain the most frequently used algorithms for optimizing the weights of a deep learning model.

#### 2.2.3.1. Mini-batch Gradient Descent

The mini-batch gradient descent is the most used algorithm for optimizing deep learning models. This algorithm updates the weights of the model for every mini-batch of  $n$  training samples according to Equation 2.12. By this way, the variance of the updates is reduced and the convergence is more stable.

$$\theta \leftarrow \theta - \alpha \frac{1}{n} \sum_{i=1}^n (h_{\theta}(x^{(i)}) - y^{(i)}) x^{(i)} \quad (2.12)$$

where  $\theta$  are the parameters of the model,  $\alpha$  is the learning rate,  $n$  is the size of the mini-batch,  $h_{\theta}(x^{(i)})$  is the output value of the model for the  $x^{(i)}$  sample,  $y^{(i)}$  is the label of the sample  $i$  and  $x^{(i)}$  is the input sample.

### 2.2.3.2. Momentum

Mini-batch gradient descent has problems when the surface has ravines, that is, areas where the surface curves more in one dimension than in another. In these scenarios, mini-batch gradient descent oscillates between the sides of the ravine while the movement in the other axis is very slow. Therefore, the optimization process will take a long time in these cases.

In [112], the author proposes a new algorithm called Momentum. This algorithm is based on mini-batch gradient descent but adding a new parameter  $\gamma$  which is the fraction of the update of the past step used during the current step. This term helps to accelerate the convergence as it performs bigger updates on dimensions whose gradients have the same direction and smaller updates for dimensions with gradients pointing to different directions. Equation 2.13 shows the steps to update the parameters of the network using this approach.

$$\begin{aligned} v_t &= \gamma v_{t-1} + \alpha \frac{1}{n} \sum_{i=1}^n (h_{\theta}(x^{(i)}) - y^{(i)}) x^{(i)} \\ \theta &\leftarrow \theta - v_t \end{aligned} \tag{2.13}$$

where  $\theta$  are the parameters of the model,  $\alpha$  is the learning rate,  $n$  is the size of the mini-batch,  $h_{\theta}(x^{(i)})$  is the output value of the model for the  $x^{(i)}$  sample,  $y^{(i)}$  is the label of the sample  $i$ ,  $x^{(i)}$  is the input sample,  $\gamma$  is a decay constant,  $v_t$  is the current update and  $v_{t-1}$  is the past update.

### 2.2.3.3. Nesterov accelerated gradient

Nesterov accelerated gradient [104] (NAG) tries to go one step ahead of momentum by computing two steps in one. When momentum is used to update the weights of the model,  $\theta - \gamma v_{t-1}$  gives an approximation of the future position of the parameters which is completed when the gradients are aggregated. Thus, to go one step ahead, a new step can be done computing the gradients with respect to the approximate future position  $\theta - \gamma v_{t-1}$ . Equation 2.14 shows the steps to update the parameters of the model.

$$v_t = \gamma v_{t-1} + \alpha \nabla_{\theta} J(\theta - \gamma v_{t-1}) \tag{2.14}$$

$$\theta \leftarrow \theta - v_t \tag{2.15}$$

where  $\gamma$  is a decay constant,  $\theta$  are the parameters of the model,  $v_t$  is the current update and  $v_{t-1}$  is the past update.

#### 2.2.3.4. Adagrad

Adagrad [33] is an optimization algorithm based on mini-batch gradient descent that adapts the learning rate for each parameter. That way, it uses lower learning rates to perform smaller updates for parameters associated with frequent features and, higher learning rates to produce bigger updates for parameters associated with infrequent features.

Adagrad uses Equation 2.12 to update the weights as in mini-batch gradient descent but changing the learning rate value ( $\alpha$ ) by Equation 2.16.

$$lr_{t,i} = \frac{\alpha}{\sqrt{G_{t,ii}} + \epsilon} \quad (2.16)$$

where  $lr_{t,i}$  is the learning rate for the parameter  $i$  at step  $t$ ,  $\alpha$  is the base learning rate,  $G_{t,ii}$  is a diagonal matrix where each diagonal element  $(i,i)$  is the sum of the squares of gradients with regard to  $\theta_i$  and  $\epsilon$  is a smoothing term to avoid zero-divisions.

One of the main benefits of this algorithm is the auto-tuning of the learning rate, which is a critical point during the training process. Thus, with this approach, one only has to define an initial learning rate and it is fine-tuned during the training process for each parameter. However, this benefit is also a mayor problem during the training as to adapt the learning rate, squared gradients are aggregated in  $G_{t,ii}$  and, as all values are positive, these values grow indefinitely making the learning rate smaller in each step. Therefore, the learning rate will became small enough to stop the learning process.

#### 2.2.3.5. Adadelata

Adadelata [152] is an extension of Adagrad which solves the problem with the decreasing learning rate due to the aggregation of squared gradients. In this case, the aggregation of gradients is limited to a fixed windows size  $w$ . This way, the gradients are only accumulated during  $w$  steps what solves the problem with the learning rate. To execute this operation more efficiently, the sum of gradients is recursively defined as an exponentially decaying average of the  $w$  past squared gradients.

Thus, in Equation 2.16, the term  $G_{t,ii}$  is substituted by Equation 2.17.

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1 - \gamma)g_t^2 \quad (2.17)$$

where  $E[g^2]_t$  is the average of gradients for step  $t$ ,  $\gamma$  is a decay constant similar to the used in the momentum approach,  $E[g^2]_{t-1}$  is the average of gradients for the past step and  $g_t^2$  is the squared gradient for step  $t$ .

However, Equation 2.17 can be rewritten using a new decaying average defined over the squared parameter updates (Equation 2.18) instead of the squared gradients.

$$E[\Delta\theta^2]_t = \gamma E[\Delta\theta^2]_{t-1} + (1 - \gamma)\Delta\theta_t^2 \quad (2.18)$$

where  $E[\Delta\theta^2]_t$  is the average of parameter updates for step  $t$ ,  $\gamma$  is the same decay constant,  $E[\Delta\theta^2]_{t-1}$  is the average of parameter updates for the past step and  $\Delta\theta_t^2$  is the squared parameter update for step  $t$ .

Substituting  $G_{t,ii}$  in Equation 2.16 by Equation 2.18, the denominator can be approximated by the Root Mean Squared (RMS) error criterion of the gradient:  $RMS[\Delta\theta]_t = \sqrt{E[\Delta\theta^2]_t + \epsilon}$ . Since  $RMS[\Delta\theta]_t$  is unknown beforehand, it is approximated with the RMS of the parameter updates until the previous step. Thus, substituting all terms in the equations, the update rule for Adadelta is defined in Equation 2.19.

$$\theta_{t+1} \leftarrow \theta_t - \frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t \quad (2.19)$$

where  $\theta_{t+1}$  are the parameters in the step  $t+1$ ,  $\theta_t$  are the current parameters,  $RMS[\Delta\theta]_{t-1}$  is the root mean squared of parameter updates until the current step,  $RMS[g]_t$  is the root mean squared of current gradients and  $g_t$  are the current gradients.

### 2.2.3.6. Adam

Adaptive Moment Estimation (Adam) [67] is another approach that adapts the learning rate for each weight of the model. Like Adadelta, it stores the exponentially decaying average of past squared gradients  $v_t$ . In addition, Adam also stores an exponentially decaying average of past gradients  $m_t$  which is similar to momentum. Equations 2.20 and 2.21 are used to compute  $m_t$  and  $v_t$  respectively.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (2.20)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (2.21)$$

where  $\beta_1$  and  $\beta_2$  are decay constants,  $m_{t-1}$  and  $v_{t-1}$  are the decaying average of past gradients and the decaying average of past squared gradients, respectively, for step  $t-1$ ,  $g_t$  are the current gradients and  $g_t^2$  are the current squared gradients.

Parameters  $m_t$  and  $v_t$  are estimations of the first moment (mean) and the second moment (variance). Due to both parameters are initialized as zero-vectors, they are biased towards zero, specially during the first steps and when parameters  $\beta_1$  and  $\beta_2$  are close to 1. To solve that biasing problem, both parameters are re-computed with Equations 2.22 and 2.23.

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (2.22)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (2.23)$$

Finally the updating rule of Adam is shown in Equation 2.24.

$$\theta_{t+1} \leftarrow \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \quad (2.24)$$

where  $\theta_{t+1}$  are the parameters in the step  $t+1$ ,  $\theta_t$  are the current parameters,  $\alpha$  is the learning rate,  $\hat{m}_t$  is the bias corrected estimation of the first moment,  $\hat{v}_t$  is the bias corrected estimation of the second moment and  $\epsilon$  is a smoothing term.

#### 2.2.4. Hyper-parameters

Hyper-parameters are the variables that define high level concepts of the network such as complexity of the model or learning capacity. These parameters must be set before the training process as they cannot be optimized. To decide the best values of the hyper-parameters, a cross-validation process must be performed.

Regarding deep learning approaches, the most common hyper-parameters are:

- Architecture of the network. The architecture includes the number and kind of layers and the number of filters of each layer.
- Learning rate. Defines the update speed of the weights. Lower learning rates produce slower training processes and bigger learning rates accelerates the convergence but excessively big values could make the network not to converge.
- Weight decay. Parameter included in the weight update rule that causes the weights to exponentially decay to zero if no other update is performed. It helps to reduce overfitting.
- Batch size. Number of samples fed into the network during every training step.
- Stop criterion. Rule to stop the training. It can be a number of training iterations, a minimum accuracy, overfitting detection, etc.

## 2.3. Information fusion

When several sources of information are available, a method to fuse those sources can be used to improve the performance of the global approach. On the one hand, we can combine those sources of information before learning a classification model. This approach is usually known as *early fusion*. A typical example of early fusion is the concatenation of feature vectors. On the other hand, we can train independent classifiers for each source of information, and then, define a strategy to fuse the classification or confidence scores. This is known as *late fusion*.

In this thesis we applied six different fusion schemes to the approach based on handcrafted features and to the deep learning approach. In both cases, the fusion is able to help the training process to obtain better results.

In this section, both fusion schemes are explained together with some examples of each one.

### 2.3.1. Early fusion

This fusion is performed at feature level, that is, before learning the classification model. As features from different sources or modalities are combined,

the future classifier has more information to represent the classes and to obtain a better separation between them. However, depending on the combination approach, the fused features can be excessively large, penalizing the learning process. Therefore, a good fusion scheme is necessary to deal with large feature vectors.

### 2.3.1.1. Feature vector concatenation

The simplest method for information fusion is vector concatenation. Given a set of  $n$  row vectors  $\{f_1, f_2, \dots, f_n\}$ , each computed from a different type of feature, a new feature vector  $\hat{f}$  is defined as the concatenation of the  $n$  feature vectors. This approach can be considered as an early fusion method, since the combination of information is carried out before any learning/classification procedure.

In a deep learning approach, this scheme is easily carried out having a branch per source of information and concatenating them at some point in the architecture. Then, after the concatenation, all layers are common to all modalities. By this way, the model can learn features from all sources at the same time.

### 2.3.1.2. Bi-modal codewords

This kind of early fusion [149] builds a unique representation that fuses all the information provided by two different sources. In this manner, instead of concatenating the information, the method learns how to mix the information from different modalities into a unique and richer representation.

Given a set of  $n$  row vectors  $\{f_1, f_2, \dots, f_n\}$ , each computed from a different type of modality. The distance between the representation of each modality is computed and a clustering technique is applied to this information in order to select the best features that represent the correlation between modalities. The groups obtained in the clustering process are used to build a new feature vector  $\hat{f}$  that encodes the fused information from the original representation of each modality.

### 2.3.1.3. Multiple Kernel Learning

Multiple Kernel Learning (MKL) [132] is another approach for early fusion. It is based on the idea of applying a function, or kernel, that maps the input data to a higher dimensionality where the data are linearly separable. From this idea, the authors extend it to use a combination of kernels instead of using a



single kernel. Then, the method tries to find an optimal kernel that best maps the information to make it separable.

Given a set of  $k$  row vectors  $\{f_1, f_2, \dots, f_k\}$ , each computed from a different type of modality and being  $\delta_1, \dots, \delta_k$  the  $k$ -associated distance functions, where  $\delta_k = w_k^T f_k$ . The algorithm tries to find the optimal kernel  $K_{opt} = \sum_k d_k K_k$  where  $K_k$  is the  $k$ -th kernel matrix (*i.e.* function of  $\delta_k$ ) and  $d$  are the weights. The computation is performed as an SVM optimization framework where the primal problem can be formulated as:

$$\begin{aligned} \min_{w_k, b, d, \xi \geq 0} \quad & \frac{1}{2} \sum_k \frac{w_k^T w_k}{d_k} + C \sum_k \xi_k + \frac{\lambda}{2} \|d\|_p^2 \\ \text{s.t.} \quad & y_i \left( \sum_k w_k^T f_k + b \right) \geq 1 - \xi, i = 1, \dots, N \end{aligned} \quad (2.25)$$

where  $\|\cdot\|_p$  represents the Euclidean  $p$ -norm,  $\xi_k$  is the slack parameter,  $C$  is the regularization parameter and finally,  $w_i$  and  $b$  are the weights and bias of the SVM, respectively. Nevertheless, this formulation is equivalent to concatenate  $K$  modalities of each sample. The authors present a richer representation that uses the product of kernels instead of the sum.

### 2.3.2. Late fusion

This fusion scheme is carried out at score level, that is, after learning the classification model, using its probability distribution as input. In this case, the idea is to combine the classification scores from different sources to produce a better probability which combines all the information. By this way, each individual model for each modality is supposed to obtain the best accuracy possible for that modality. Thus, if multiple of those models are combined, the output probability should be better than the individual ones.

This fusion scheme can be applied to any kind of classification model either hand-crafted or deep model.

#### 2.3.2.1. Weighted Scores

Weighted scores (WS) is a late fusion method that uses the estimated accuracy of the individual models to assign a weight to each confidence score, obtaining in this way a new combined score.

Given a set of  $n$  confidence score vectors  $\{s_1, s_2, \dots, s_n\}$ , associated to  $n$  models, and their corresponding weighting factors  $\{a_1, a_2, \dots, a_n\}$ , the final score vector  $s_f$  is computed as follows:

$$s_f = \sum_{i=1}^n s_i * a_i \quad (2.26)$$

where the sum of the weighting factors is equal 1.

### 2.3.2.2. Classifier over the scores

Classifier over the scores is another late fusion method that builds a classifier using as input the confidence scores of all modalities obtained in a training set. By this way, the classifier is able to automatically learn a relation between modalities that maximizes the classification performance.

Given a set of  $n$  confidence score vectors  $\{s_1, s_2, \dots, s_n\}$ , obtained from  $n$  models, we build a new feature vector  $f_s$  by concatenating those scores. Then, a classifier is trained on those new feature vectors  $\{f_s\}$ , to obtain a final fusion score  $s_f$ .

### 2.3.2.3. Rank Minimization

The method proposed by Ye et al. [150], for the problems of object categorization and video event detection, can be classified into the category of late fusion methods.

Let  $s = [s_1, s_2, \dots, s_m]$  be a confidence score vector of a model on  $m$  samples. A pairwise relationship matrix  $T$  is constructed from  $s$  as:

$$T_{jk} = \text{sign}(s_j - s_k) \quad (2.27)$$

Given  $n$  models, the robust late fusion method of Ye et al. aims at optimizing the following problem:

$$\begin{aligned}
& \min_{\hat{T}, E_i} \|\hat{T}\|_* + \lambda \sum_{i=1}^n \|E_i\|_1, \\
& \text{s.t. } T_i = \hat{T} + E_i, i = 1, \dots, n, \\
& \quad \hat{T} = -\hat{T}^\top
\end{aligned} \tag{2.28}$$

Where  $T_i$  is the pairwise relationship matrix of the  $i$ -th model,  $\|\cdot\|_*$  denotes the nuclear norm of a matrix (*i.e.* the sum of the singular values of the matrix),  $E_i$  is a sparse matrix associated to the  $i$ -th model,  $\hat{T}$  is the estimated rank-2 pairwise relationship matrix consistent among the samples and models, and  $\lambda$  is a positive tradeoff parameter to be cross-validated. Such optimization problem is solved by inexact Augmented Lagrange Multiplier method [79].

As described in [150], given the estimated matrix  $\hat{T}$ , and assuming that  $\hat{T}$  is generated from  $\hat{s}$  as  $\hat{T} = \hat{s}e^\top - e\hat{s}^\top$ , the new score vector  $\hat{s}$  is computed as

$$(1/m)\hat{T}e = \arg \min_{\hat{s}} \|\hat{T}^\top - (\hat{s}e^\top - e\hat{s}^\top)\|_F^2, \tag{2.29}$$

treating  $(1/m)\hat{T}e$  as the recovered  $\hat{s}$  after the late fusion of the input scores.

## 2.4. Incremental learning

One of the main challenges in developing a visual recognition system targeted at real-world applications is learning classifiers incrementally, where new classes need to be learned continually. For example, a face recognition system needs to handle new faces to identify new people. This task needs to be accomplished without having to re-learn faces learned previously. While this is trivial to accomplish for most people (we learn to recognize faces of new people we meet every day), it is not the case for a machine learning system. Traditional models require all the data (corresponding to the old and the new classes) to be available at training time, and are not equipped to consider only the new data with a small selection of the old data. In an ideal system, the new classes should be integrated into the existing model, sharing the previously learned parameters.

Traditionally, this problem has been faced using hand-crafted features combined with SVMs. SVMs are specially useful for this problem as storing the support vectors of the current model, the current decision boundaries are kept

and, when new classes are added, the training process is easier as the system only has to separate the new classes from the old ones.

However, deep learning approaches are more powerful and obtains better results than SVMs, therefore, this manuscript focuses on incremental learning from a deep learning point of view

In this thesis we designed a new incremental learning approach based on deep learning. The pipeline and architecture proposed are able to obtain state-of-the-art results for the problems of image recognition and gait recognition.

### 2.4.1. Deep incremental learning

A truly incremental deep learning approach for classification is characterized by its:

- Ability to being trained from a flow of data, with classes appearing in any order, and at any time.
- Good performance on classifying old and new classes.
- Reasonable number of parameters and memory requirements for the model.
- End-to-end learning mechanism to update the classifier and the feature representation jointly.

Therefore, an ideal approach would be able to train on an infinitely-large number of classes in an incremental way, without losing accuracy, and having exactly the same number of parameters, as if it were trained from scratch.

Figure 2.9 shows an scheme of a general incremental learning approach where a deep model is used as feature extractor common to all classes (new and old), and new classification layers are attached to the feature extractor when new classes are added to the model. To train the model, it is necessary to include a loss function per classification layer with old classes. This loss function or distillation loss retains the knowledge from old classes. Finally, a classification loss is necessary to learn the new classes and the separability between old a new classes.

During training, all losses are combined to adapt the weights of the model retaining the old knowledge and including the new one.

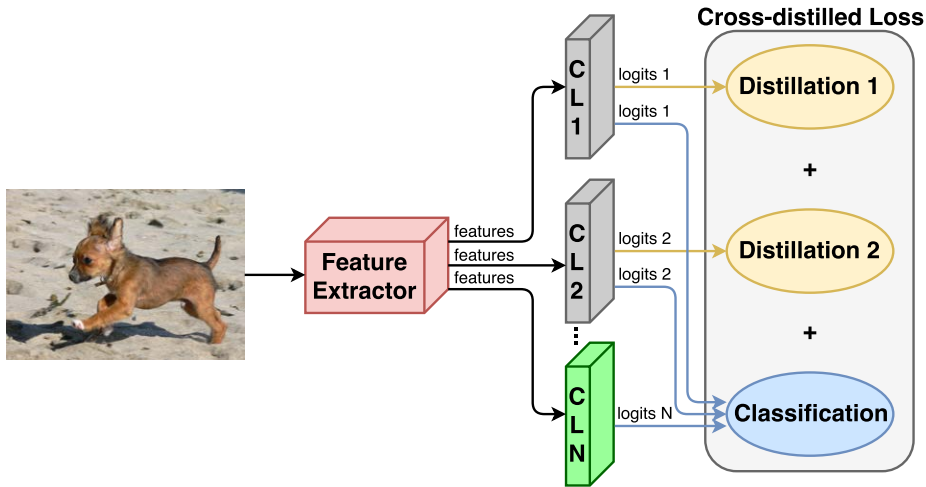


Figure 2.9: Incremental model. Given an input image, the feature extractor produces a set of features which are used by the *classification layers* ( $CL_i$  blocks) to generate a set of *logits*. Grey *classification layers* contain old classes and their *logits* are used for distillation and classification. The green *classification layer* (CLN block) contains new classes and its *logits* are involved only in classification.

## 2.5. Energy consumption of deep learning

From the point of view of High Performance Computing, Deep Learning applications are gaining momentum in the realm of Artificial Intelligence thanks to the use of accelerators that employ thousands of cores to carry out the costly computation of CNN models. An important aspect that must be evaluated when accelerators are applied to both CNN training and testing steps is the power consumption of the architecture. That way, the flagship performance metric is no longer GFLOPS (Giga Floating-Point Operations Per Second), but GFLOPS/w (GFLOPS per watt).

Thus, energy consumption has gained relevance among researchers during the big-data era as it can represent more than 20% of the budget in Data Centers nowadays. From a global point of view, costs in energy consumption have exceeded 5 billion dollars per year over the last decade only in the US [12], and it is predicted that the energy billing will increase in forthcoming years if power optimizations are not conducted in all levels, including operating systems, kernels and applications.

The industry is aware about the need of low-power CNN acceleration when using them extensively. Google is a clear example with Tensor Processing Unit (TPU) tailored to their TensorFlow framework in its data centers, claiming that they are able to reduce power an order of magnitude versus GPUs [63]. However, these TPU processors are not publicly available and only Google researchers and some selected research groups are allowed to use them. Moreover, the TPU processors are physically located in Google's datacenters precluding physical measurements of energy consumption.

FPGAs are other kind of accelerators commonly used in heavy computation processes. However, due to their expensive costs, hard programming and lack of frameworks, their use is very limited in the deep learning field.

On the other hand, GPUs provide a good computational capacity with a moderate cost. In addition, GPU manufactures are reducing the value of GFLOPS/w of these architectures in a significant way. Moreover, available GPU programming models are user friendly, what has allowed the creation of many frameworks that take advantage of their computational capacities. Thus, thanks to these benefits, GPUs are being extensively used for deep learning purposes.

In this thesis we carried out a comprehensive study of the power consumption incurred by several NVIDIA GPUs on different CNN models. To the best of our knowledge, our work is pioneer on measuring the actual power consumption of CNNs with wires and measurement devices physically plugged to the pinout of latest GPU generations and multi-GPU platforms, and even identifying the most expensive operators and functions in terms of energy budget.

### 2.5.1. Energy consumption on GPUs

To measure the energy consumption on GPUs, it is necessary to use specific devices, as GPUs do not include this capability. The developed system to measure current, voltage and wattage is based on a Beaglebone Black, an open-source hardware [10] combined with the Accelpower module [42], which has eight INA219 sensors [2]. Inspired by [60], wires taken into account are two power pins on the PCI-express slot (12 and 3.3 volts) plus six external 12 volt pins coming from the power supply unit (PSU) in the form of two supplementary 6-pin connectors (half of the pins used for grounding). See Figure 2.10 for details.

Accelpower uses a modified version of `pmlib` library [4], a software package specifically created for monitoring energy. It consists of a server daemon that collects power data from devices and sends them to the clients, together with a client library for communication and synchronization with the server.

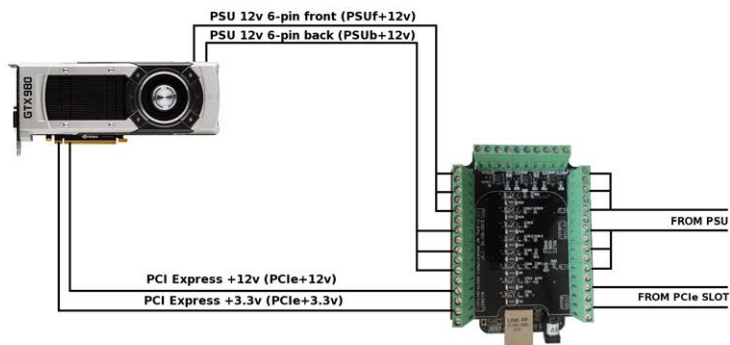


Figure 2.10: Wires, slots, cables and connectors for measuring energy on GPUs.





# 3

## Related work

---

This chapter reviews the related work for the gait recognition problem. To make this chapter more understandable, it has been divided into sections according to the approach described. Thus, Sections 3.1 and 3.2 reviews hand-crafted and deep learning approaches respectively. Sections 3.3 and 3.4 focuses on fusion and multi-task methods. Section 3.5 describes the incremental learning approaches. Finally, Section 3.6 reviews the energy consumption studies applied to deep learning approaches.

### 3.1. Hand-crafted approaches

Many research papers have been published in recent years tackling the problem of human gait recognition using different sources of data, like inertial sensors [107, 106], foot pressure [157], infrared images [146] or the traditional images. For example, in [59] we can find a survey on this problem summarizing some of the most popular approaches. Some of them use explicit geometrical models of human bodies, whereas others use only image features. A sequence of binary silhouettes of the body is adopted in many works as input data. In this sense, the most popular silhouette-based gait descriptor is the called Gait Energy Image (GEI) [47]. The key idea is to compute a temporal averaging of the binary silhouette of the target subject. Liu et al. [80], to improve the gait recognition performance, propose the computation of HOG descriptors from popular gait descriptors as the GEI and the Chrono-Gait Image (CGI). In [90], the authors try to find the minimum number of gait cycles needed to carry out a successful recognition by using the GEI descriptor. Martin-Felez and Xiang [91] [92], using GEI as

the basic gait descriptor, propose a new ranking model for gait recognition. This new formulation of the problem allows to leverage training data from different datasets, thus, improving the recognition performance. Another silhouette-based approximation is Motion Silhouette Image (MSI) [74]. This approach generates a gray-scale image where each pixel contains the temporal history of the motion of that pixel. As noisy silhouettes have a huge impact in this representation, Lee et al. [76] propose a new descriptor based on MSI, the Motion Energy Image (MEI). This new representation assigns to each pixel the mean energy of each silhouette within a fixed size window. By this way, the effect of noise in one frame is minimised with the energy of the other frames. Alternatively, in [3], Akae et al. propose a temporal super resolution approach to deal with low frame-rate videos for gait recognition. They achieve impressive results by using binary silhouettes of people at a rate of 1-fps. Hu proposes in [57] the use of a regularized local tensor discriminant analysis method with the Enhanced Gabor representation of the GEI. In addition, the same author defines in [56] a method to identify camera viewpoints at test time from patch distribution features. Recently, Lai et al. [73] proposed a novel discriminant subspace learning method (Sparse Bilinear Discriminant Analysis) that extends methods based on matrix-representation discriminant analysis to sparse cases, obtaining competitive results on gait recognition. In many works it is assumed that the target person follows a straight path, however, Iwashita et al. [61] explicitly focus on curved trajectories. Gong et al. [41] propose a method that uses dense local spatio-temporal features and a Fisher-based representation rearranged as tensors.

### 3.2. Deep learning approaches

A new realm of the feature learning field for recognition tasks started with the advent of Deep Learning (DL) architectures [43]. Recently, DL approaches based on CNN have been used on image-based tasks with great success [72, 124, 153]. In the last years, deep architectures for video have appeared, specially focused on action recognition, where the inputs of the CNN are subsequences of stacked frames. The very first approximation of DL applied to stacked frames is proposed in [75], where the authors apply a convolutional version of the Independent Subspace Analysis algorithm to sequences of frames. By this way, they obtain low-level features which are used by high-level representation algorithms. A more recent approach is proposed in [66], where a complete CNN is trained with sequences of stacked frames as input. In [123], Simonyan and Zisserman proposed to use as input to a CNN a volume obtained as the concatenation of two channels: optical flow in the  $x$ -axis and  $y$ -axis. To normalize the size of the inputs, they split

the original sequence in subsequences of 10 frames, considering each subsample independently. Donahue *et al.* [32] propose a new viewpoint in DL using a novel architecture called ‘Long-term Recurrent Convolutional Networks’. This new architecture combines CNN (specialized in spatial learning) with Recurrent Neural Networks (specialized in temporal learning) to obtain a new model able to deal with visual and temporal features at the same time. Recently, Wang *et al.* [137] combined dense trajectories with DL. The idea is to obtain a powerful model that combines the deep-learned features with the temporal information of the trajectories. They train a traditional CNN and use dense trajectories to extract the deep features to build a final descriptor that combines the deep information over time. On the other hand, Perronnin *et al.* [109] proposed a more traditional approach using Fisher Vectors as input to a Deep Neural Network instead of using other classifiers like SVM. Recently, He *et al.* [48] proposed a new kind of CNN, named ResNet, which has a large number of convolutional layers and ‘residual connections’ to avoid the vanishing gradient problem.

Although several papers can be found for the task of human action recognition using DL techniques, few works apply DL to the problem of gait recognition. In [55], Hossain and Chetty propose the use of Restricted Boltzmann Machines to extract gait features from binary silhouettes, but a very small probe set (*i.e.* only ten different subjects) were used for validating their approach. Yan *et al.* [147] use GEI descriptors, computed on completed walking cycles, as input data for a Convolutional Neural Network (CNN). The proposed CNN is able to extract high-level features that are used in a multi-task framework, where the goals are gait, angle view and scene recognition. A more recent work, [144], uses a random set of binary silhouettes of a sequence to train a CNN that accumulates the calculated features in order to achieve a global representation of the dataset. In [40], raw 2D GEI are employed to train an ensemble of CNN, where a Multilayer Perceptron (MLP) is used as classifier. Similarly, in [5] a multilayer CNN is trained with GEI data. A novel approach based on GEI is developed on [143], where the CNN is trained with pairs of gallery-probe samples and using a distance metric.

Despite most CNNs are trained with visual data (*e.g.* images or videos), there are some works that build CNNs for different kinds of data like inertial sensors or human skeletons. Holden *et al.* [53] propose a CNN that corrects wrong human skeletons obtained by other methods or devices (*e.g.* Microsoft Kinect). Neverova *et al.* [105] build a temporal network for active biometric authentication with data provided by smartphone sensors (*e.g.* accelerometers, gyroscope, etc.).

Recently, some authors have proposed the use of 3D convolutions to extract visual and temporal data from videos. Tran *et al.* [130] define a new network composed of 3D convolutions in the first layers that has been successfully applied

to action recognition. Following that idea, Wolf *et al.* [141] build a CNN with 3D convolutions for gait recognition. Due to the high number of parameters that must be trained (3D convolutions implies three times more parameters per convolutional layer), Mansimov *et al.* [87] show several ways to initialize a 3D CNN from a 2D CNN.

### 3.3. Fusion approaches

Since there are different descriptors for representing the same data, an interesting idea would be to try to combine those descriptors into a single one that could benefit from the original descriptors. To perform this task, several methods have appeared [8, 142]. Also, the emergence of new cheaper devices that record multimodal spectrums (*e.g.* RGB, depth, infrared) has allowed to investigate how to fuse that information to build richer and more robust representations for the gait recognition problem. Traditionally, fusion methods are divided into *early fusion* methods (or feature fusion) and *late fusion* (or decision fusion). The first ones try to build descriptors by fusing features of different descriptors, frequently, using the concatenation of the descriptors into a bigger one as in [25]. On the other hand, late fusion tries to fuse the decisions obtained by each classifier of each modality, usually, by applying arithmetic operations like sums or products on the scores obtained by each classifier as in [25, 52]. Fusion has been also employed with CNN to improve the recognition accuracy for different computer vision tasks. For example, two independent CNNs fed with optical flow maps and appearance information (*i.e.* RGB pixel volumes) are employed in [123] to perform action recognition. Then, class score fusion is used to combine the softmax output of both CNNs. In a similar way, Eitel *et al.* [35] have proposed a DL approach for object recognition by fusing RGB and depth input data. They concatenate the outputs of the last fully-connected layers of both networks (those processing RGB and depth data) and process them through an additional fusion layer. Wang *et al.* [134] also employ a multimodal architecture composed by two CNN networks to process RGB-D data. They propose to learn two independent transformations of the activations of the second fully-connected layer of each network, so correlation of color and depth features is maximized. In addition, these transformations are able to improve the separation between samples belonging to different classes.

### 3.4. Multi-task approaches

Although in *gait recognition* the main goal is to recognize the identity of people by the way they walk, other tasks can be defined from the gait patterns as gender recognition or age estimation. Generally, those additional tasks are addressed independently of the identification task [52]. In contrast, in other contexts as face recognition, works as [127] and [155] show that simultaneous learning of several tasks provides benefit to the main task. Therefore, we explore in this thesis the benefits of a novel multi-task CNN-based architecture for gait recognition. In the context of gait recognition, we can find the recent work of Yun *et al.* [147] that uses GEI as input of a CNN to predict the subject identity, the camera viewpoint and the scenario (*i.e.* normal, with coat, with bag). Note that only identity is a biometric feature and the input is the hand-crafted GEI descriptor.

### 3.5. Incremental learning approaches

We now describe relevant methods in the field of incremental learning by organizing them into traditional ones using a fixed feature set, and others that learn the features, *i.e.* , through deep learning frameworks, in addition to training classifiers.

#### Traditional approaches

Many initial methods for incremental learning targeted the SVM classifier [27], exploiting its core components: support vectors and Karush-Kuhn-Tucker conditions. Some of the methods [119] retain the support vectors, which encode the classifier learned on old data, to learn the new decision boundary together with new data. Cauwenberghs and Poggio [24] proposed an alternative to this by retaining the Karush-Kuhn-Tucker conditions on all the previously seen data (which corresponds to the old classes), while updating the solution according to the new data. While these early attempts showed some success, they are limited to a specific classifier and also do not extend to the current paradigm of learning representations and classifiers jointly.

Another relevant approach is learning concepts over time, in the form of lifelong [129] or never-ending [98, 26, 31] learning. Lifelong learning is akin to transferring knowledge acquired on old tasks to the new ones. Never-ending learning,

on the other hand, focuses on continuously acquiring data to improve existing classifiers or to learn new ones. Methods in both these paradigms either require the entire training dataset, *e.g.* , [26], or rely on a fixed representation, *e.g.* , [31]. Methods such as [97, 117, 121] partially address these issues by learning classifiers without the complete training set, but are still limited due to a fixed or engineered data representation. This is achieved by: *(i)* restricting the classifier or regression models (*e.g.* , those that are linearly decomposable [121]), or *(ii)* using a nearest mean classifier (NMC) [97], or a random forest variant [117]. Incremental learning is then performed by updating the bases or the per-class prototype, *i.e.* , the average feature vector of the observed data, respectively.

Overall, the main drawback of all these methods is the lack of a task-specific data representation, which results in lower performance.

## Deep learning approaches

This class of methods provides a natural way to learn task-specific features and classifiers jointly [116, 123, 13]. However, learning models incrementally in this paradigm results in *catastrophic forgetting*, a phenomenon where the performance on the original (old) set of classes degrades dramatically [95, 44, 38, 7, 114, 82, 77, 115, 122]. Initial attempts to overcome this issue were aimed at connectionist networks [95, 38, 7], and are thus inapplicable in the context of today's deep architectures for computer vision problems.

A more recent attempt to preserve the performance on the old tasks was presented in [77] using distillation loss in combination with the standard cross-entropy loss. Distillation loss, which was originally proposed to transfer knowledge between different neural networks [50], was adapted to maintain the responses of the network on the old tasks whilst updating it with new training samples [77]. Although this approach reduced forgetting to some extent, in particular, in simplistic scenarios where the old and the new samples come from different datasets with little confusion between them, its performance is far from ideal. This is likely due to a weak knowledge representation of the old classes, and not augmenting it with an exemplar set. Works such as [115, 131] demonstrated this weakness of [77] showing significant errors in a sequential learning scenario, where samples from new classes are continuously added, and in particular when the new and the old samples are from related distributions.

Other approaches using distillation loss, such as Jung *et al.* [64] propose to freeze some of the layers corresponding to the original model, thereby limiting its adaptability to new data. Triki *et al.* [131] build on the method in [77] using

an autoencoder to retain the knowledge from old tasks, instead of the distillation loss. This method was also evaluated in a restrictive scenario, where the old and the new networks are trained on different datasets, similar to [77]. Distillation loss was also adopted for learning object detectors incrementally [122]. Despite its success for object detection, the utility of this specific architecture for more general incremental learning scenarios we target here is unclear.

Alternative strategies to mitigate catastrophic forgetting include, increasing the number of layers in the network to learn features for the new classes [120, 128], or slowing down the learning rate selectively through per-parameter regularization [68]. Xiao *et al.* [145] also follow a related scheme and grow their tree-structured model incrementally as new classes are observed. The main drawback of all these approaches is the rapid increase in the number of parameters, which grows with the total number of weights, tasks, and the new layers.

Rebuffi *et al.* [115] present iCaRL, an incremental learning approach where the tasks of learning the classifier and the data representation are decoupled. iCaRL uses a traditional NMC to classify test samples, *i.e.*, it maintains an auxiliary set containing old and new data samples. The data representation model, which is a standard neural network, is updated as and when new samples are available, using a combination of distillation and classification losses [50, 77].

### 3.6. Energy consumption studies on deep learning

Moons *et al.* [100] propose methods at system and circuit level based on approximate computing. They always perform training using 32-bit, lowering precision during the test phase. They claim energy gains up to  $30x$  without losing classification accuracy and more than  $100x$  at 99% classification accuracy, compared to a commonly used 16-bit fixed point number format. Cai *et al.* propose NeuralPower [16], a layer-wise predictive framework based on sparse polynomial regression, for predicting the serving energy consumption of a CNN deployed on different GPU platforms and Deep Learning software tools, attaining an average accuracy of 88.24% in execution time, 88.34% in power, and 97.21% in energy. Andri *et al.* introduce YodaNN [6], an energy and area efficiency accelerator based on ASIC hardware optimized for BinaryConnect CNNs which basically removes the need for expensive multiplications during training, also reducing I/O bandwidth and storage. Yang *et al.* [148] propose an energy-aware pruning algorithm for CNNs that directly uses energy consumption estimation of a CNN to guide the pruning process. The energy consumption of AlexNet and GoogLeNet are reduced by  $3.7x$  and  $1.6x$ , respectively, with less than 1% top-5

accuracy loss. Results are obtained via a energy estimation tool for Deep Neural Networks publicly available in [1]. Lin et al. [78] propose PredictiveNet to skip a large fraction of convolutions in CNNs at runtime without modifying the CNN structure or requiring additional branch networks. An analysis supported by simulations is provided to justify how to preserve the mean square error (MSE) of the nonlinear layer outputs. Energy savings are attained by reducing the computational cost by a factor of  $2.9x$  compared to a state-of-the-art CNN, while incurring marginal accuracy degradation.

Moving away from estimators, predictors and simulators, we may find examples of real energy measurements and studies on low-power devices like DSPs [94] and FPGAs [14], even for CNN applications [138, 93]. But to the best of our knowledge, there are no works measuring the actual power consumption of CNNs with wires and measurement devices physically plugged to the pinout of latest GPU generations and multi-GPU platforms, and even identifying the most expensive operators and functions in terms of energy budget.



# 4 Published Work

---

This chapter collects the set of papers published during the PhD. Specifically, four papers have been published in journals indexed in the Journal of Citation Report (JCR) and two of the most important conference paper. Moreover, other three papers have been published in international conferences. Thus, nine papers are the outcome of this PhD.

## 4.1. List of Published Papers

### ■ Journal papers:

- Marín-Jiménez, M. J., Castro, F. M., Carmona-Poyato, Á., Guil, N. (2015). On how to improve tracklet-based gait recognition systems. *Pattern Recognition Letters*, 68, 103-110. Doi: 10.1016/j.patrec.2015.08.025.
- Castro, F. M., Marín-Jiménez, M. J., Guil, N. (2016). Multimodal features fusion for gait, gender and shoes recognition. *Machine Vision and Applications*, 27(8), 1213-1228. Doi: 10.1007/s00138-016-0767-5.
- Castro, F. M., Marín-Jiménez, M. J., Mata, N. G., Muñoz-Salinas, R. (2017). Fisher motion descriptor for multiview gait recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 31(01), 1756002. Doi: 10.1142/S021800141756002X.
- Castro, F. M., Guil, N., Marín-Jiménez, M. J., Pérez-Serrano, J., Ujaldón, M. (2018). Energy-based Tuning of Convolutional Neural

Networks on Multi-GPUs. *Concurrency and Computation: Practice and Experience* (to appear). Doi: 10.1002/cpe.4786.

■ **Conference papers:**

- Castro, F. M., Marín-Jiménez, M. J., Guil, N. (2015). Empirical study of audio-visual features fusion for gait recognition. In International Conference on Computer Analysis of Images and Patterns (CAIP) (pp. 727-739). Doi: 10.1007/978-3-319-23192-1\_61.
- Castro, F. M., Marín-Jiménez, M. J., Guil, N., López-Tapia, S., Pérez de la Blanca, N. (2017). Evaluation of CNN architectures for gait recognition based on optical flow maps. In International Conference of the Biometrics Special Interest Group, BIOSIG 2017, Darmstadt, Germany, September 20–22, 2017 (pp. 251-258). Doi: 10.23919/BIOSIG.2017.8053503.
- Marín-Jiménez, M. J., Castro, F. M., Guil, N., de la Torre, F., Medina-Carnicer, R. (2017). Deep multi-task learning for gait-based biometrics. In Image Processing (ICIP), 2017 IEEE International Conference on (pp. 106-110). Doi: 10.1109/ICIP.2017.8296252.
- Castro, F. M., Marín-Jiménez, M. J., Guil, N., de la Blanca, N. P. (2017). Automatic learning of gait signatures for people identification. In International Work-Conference on Artificial Neural Networks (IWANN) (pp. 257-270). Doi: 10.1007/978-3-319-59147-6\_23.
- Castro, F. M., Marín-Jiménez, M. J., Guil, N., Schmid, C., Ahahari, K. (2018). End-to-End Incremental Learning. European Conference on Computer Vision (ECCV), (to appear). Doi: 10.1007/978-3-030-01258-8\_15.

## 4.2. Summary of the papers that support this thesis

This section presents a summary of each one of the journal papers that support this thesis. For each one of them, we attach a brief summary and a full copy of the published document.

#### 4.2.1. Reference [89] ‘On how to improve tracklet-based gait recognition systems’

In [89] we study different proposals to improve the previous work published in [22] based on gait recognition using dense trajectories. A first improvement is the use of RootDCS instead of the common DCS descriptor. Thus, this new descriptor applies an element wise square root to the L1-normalized DCS descriptors. This regularization stabilizes the variance between bin values of the kinematic features histogram what improves the representation capacity of the descriptors. Another way of improvement is the use of sparse trajectories instead of dense ones because not all trajectories contributes to describe the gait. Therefore, the idea is to discard trajectories whose mean velocity vectors are similar to the median velocity of the person. This way, the most relevant trajectories are stored (*e.g.* arm and leg swing) and the superfluous ones are removed (horizontal displacement of the body). Since the final descriptors used in [22] have a large dimensionality, a PCA dimension reduction is applied to reduce that dimensionality. However, as we are facing a labeled problem, a semi-supervised approach can be used to increase the separability of the reduced descriptors. Finally, as we are using a set of SVMs in an one-vs-all approach, the output scores can be very similar sometimes. Thus, we propose the use of a Rank Minimization step whose aim is to combine the outputs of different classifiers to obtain a final and more robust decision on a set of test samples.

A thorough set of experiments is performed on three challenging datasets to validate our proposals. In all cases, the proposed improvement overcomes the results obtained by the baseline method.

#### 4.2.2. Reference [18] ‘Multimodal features fusion for gait, gender and shoes recognition’

Since there can be multiple sources of information or modalities recorded at the same time when a person is walking, our objective is to fuse those different modalities to improve the recognition results obtained with a single source of information. In this paper, we study six different ways to fuse the information of different modalities, where three of them are late fusions (performed at score level) and the other three are early fusion (performed at descriptor level).

Regarding the early fusion approaches, we propose three versions: vector concatenation, where the feature vectors are concatenated to produce a larger one; bimodal codewords where a common dictionary is learned to produce a final

feature vector which mixes information from all modalities; and, multiple kernel learning where an optimization process is performed to obtain an optimal kernel which maps the information of all modalities into a higher feature space where the classes are more separable.

Regarding the late fusion approaches, we propose three options: weighted scores, where a weighted sum is applied to the output scores of each individual modality; SVM over the scores, where a SVM model is trained over the probability distributions obtained from each isolated modality; and, rank minimization, where a minimization process is done to obtain better combined probabilities.

According to the experimental results, the modality fusion helps to improve the accuracy of the system as long as the combined features are representative themselves. That is, if a modality obtains poor results, fusing that modality with another one will produce poor results as well.

#### **4.2.3. Reference [21] ‘Fisher motion descriptor for multi-view gait recognition’**

In this work, we extend the conference version paper [22]. This version, extends the experimental section to four datasets with multiple-views and walking conditions such as carrying a bag, wearing different clothes, coating shoes, treadmills, etc. Therefore, the objective of this paper is to explore the proficiency of the previously proposed approach which, due to length limitations, was impossible to test. In addition, some changes are performed to improve the accuracy of the global pipeline.

The thorough experimentation confirms that our approach is able to deal with changes in the shape and multiple views, obtaining state-of-the-art results in all datasets.

#### **4.2.4. Reference [17] ‘Energy-based Tuning of Convolutional Neural Networks on Multi-GPUs’**

In this study, we analyze the energy consumption during the training process of a CNN. Since this is an expensive process which uses a lot of resources, it is interesting to consider the energy as a parameter which can be minimized such as the loss. This way, apart from the final accuracy, the energy is also taken into account in the designing process of the CNN architecture because the energy consumed has an economical impact, and a network that consumes less energy

can be a great saving.

As many possible architectures can be designed, we focus on the following ones: an AlexNet [72] version adapted to the problem of gait recognition and the original AlexNet used for image recognition, a ResNet [48] network for image recognition on ImageNet and an adapted version for gait recognition. This way, we are studying two kinds of networks used as baseline in most of the current approaches, and two of the most common kinds of input (images and video). Moreover, we also analyze the impact of the batch size to the accuracy and energy consumption. Finally, we also consider the impact of doing a multi-GPU training together with different GPU architectures (Pascal and Maxwell in our case).

According to the results, in small datasets, minimizing power consumption and accuracy is very complex since small batch sizes are required to obtain a good training process what penalizes energy consumption. However, networks based on AlexNet are an exception supporting larger batch sizes allowing to minimize power consumption while maintaining good accuracy. In large databases, large batch sizes can be used without lowering accuracy, which allows consumption and accuracy to be optimized together. Regarding GPU architectures, Pascal achieves better consumption and higher frequencies as expected. Finally, configurations with multiple GPUs yield good results with two devices. With more than two devices, the problem has to be studied individually.

## 4.3. Additional papers

This section presents a summary of two conference papers that contains techniques not published in journals. For each one of them, we attach a brief summary and a full copy of the published document.

### 4.3.1. Reference [88] ‘Deep Multi-Task Learning for Gait-based Biometrics’

In this work, we explore the use of gait information for different tasks, not only for identification. Although gait is mainly used for people identification, other biometric tasks can be defined based on gait, *e.g.* gender recognition or age estimation. However, not much attention has been paid to the fact that those tasks are closely related and can benefit ones from the others when are jointly considered during training.

We propose a multi-task learning approach based on a combined loss function for training CNNs from optical flow data. By this way, the gait signatures or gait features obtained by the network are simultaneously suitable for all the tasks trained during the multi-task learning process.

According to the results, the multi-task learning speeds up the the velocity of convergence of the objective function, it also improves the capacity of generalization of the shared gait signatures obtained for the biometric tasks and, it is able to improve the previous state-of-the-art results.

#### 4.3.2. Reference [20] ‘End-to-End Incremental Learning’

Although deep learning approaches achieve state-of-the-art results when they are trained with huge datasets or, at least, using the entire dataset, they have problems when training new classes incrementally. This problem is known as *catastrophic forgetting*, which is a huge decrease in the performance of the network when new classes are trained incrementally.

We propose an end-to-end approach that learns deep neural networks incrementally, using the new data from the new classes and only a small exemplar set of samples from the old classes. The training process is based on a loss function composed of a distillation measure to retain the knowledge from the old classes, and a cross-entropy loss to learn the new classes.

According to the results, our end-to-end approach is able to obtain state-of-the-art results on two challenging datasets (*i.e.* CIFAR-100 and ImageNet).

# 5 Unpublished Work

---

This chapter contains those works which are not published yet. Specifically, Section 5.1 explains the fusion approach developed for deep learning applied to gait recognition and, Section 5.2 describes the incremental learning approach developed for gait recognition.

## 5.1. Multimodal gait recognition with CNNs

In this work, the experimental study is directed towards three main objectives. The first objective is the identification of good architectures that, using as input 2D spatial information from a sequence of video frames or 3D spatio-temporal information from a finite subset of video frames, are capable of achieving high scores in the task of gait recognition. To this effect we design 2D-CNN and 3D-CNN architectures with different depth (*i.e.* layers). In addition, as previous works [72] have shown that deeper CNN models achieve better generalization power, we have also designed a ResNet architecture based on [48]. The second objective is the use of diverse types of raw input features (*i.e.* volumes of gray-level pixels, optical flow maps and depth maps) to automatically derive gait signatures. And, the last objective is to assess if the combination of information derived from different inputs allows to obtain better models for the task of gait recognition.

Therefore, the main contributions of this work are:

- A comparative study of state-of-the-art CNN architectures using as input

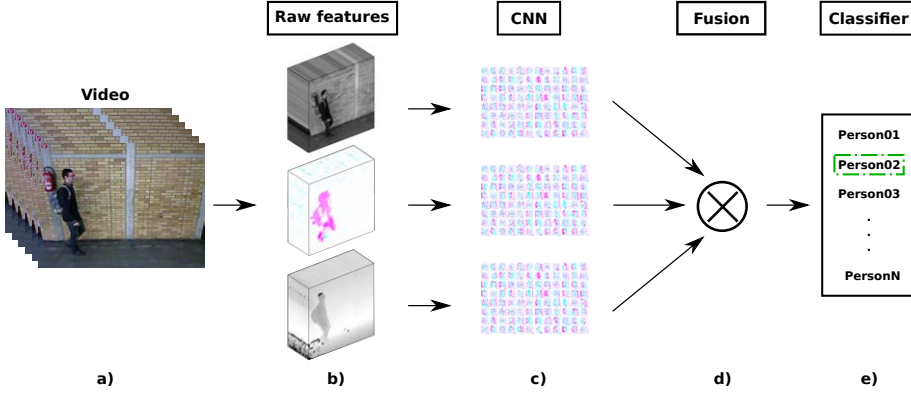


Figure 5.1: **Pipeline for gait recognition.** a) The input is a sequence of RGB-D video frames. b) Low-level features are extracted along the sequence and stacked building volumes. c) Volumes are passed through the CNN to obtain gait signatures. d) CNN outputs are combined. e) A final decision is taken to output an identity.

2D or 3D information blocks representing spatial and spatio-temporal low-level information, respectively, from data.

- A thorough experimental study to validate the proposed framework on the standard TUM-GAID and CASIA-B datasets for gait identification.
- An extensive experimental study of low level feature fusion.
- State-of-the-art results on both datasets, being our fusion scheme the best approach.

### 5.1.1. Proposed approach

In this section we describe our proposed framework to address the problem of gait recognition using CNNs. The pipeline proposed for gait recognition based on CNNs is represented in Figure 5.1:

1. Gather low-level features along the whole sequence.
2. Crop the obtained OF maps to obtain square-shaped frames, and stack  $L$  consecutive OF maps.



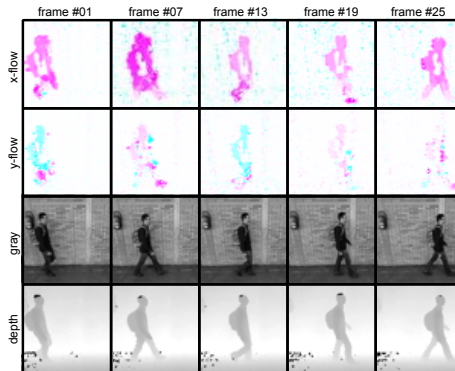


Figure 5.2: **CNN input data.** Sample frames extracted from a subsequence of 25 frames. **(top rows)** Optical flow in  $x$ -axis and  $y$ -axis. where positive flows are displayed in pink and negative flows in blue (best viewed in color). **(bottom rows)** Gray pixels and depth maps of the same sequence.

3. Build up a data cuboid from consecutive low-level feature maps.
4. Feed the CNN with the low-level feature cuboid to extract the gait signature.
5. Fuse information from the different inputs.
6. Apply a classifier to decide the subject identity.

#### 5.1.1.1. Input data

We describe here the different types of low-level features used as input for the proposed CNN architecture. In particular, we use optical flow, gray pixels and depth maps. An example of the three types of low-level features is represented in Figure 5.2. Our intuition is that this set of low-level features will cover both *motion* (*i.e.* optical flow) and *appearance* information (*i.e.* pixels and depth) of people.

**Optical flow.** The use of optical flow (OF) as input data for action representation in video with CNN has already shown excellent results [123]. Nevertheless human action is represented by a wide, and usually well defined, set of local motions. In our case, the set of motions differentiating one gait style from another is much more subtle and local.

Let  $F_t$  be an OF map computed at time  $t$  and, therefore,  $F_t(x, y, c)$  be the value of the OF vector component  $c$  located at coordinates  $(x, y)$ , where  $c$  can be either the horizontal or vertical component of the corresponding OF vector. The input data  $I_L$  for the CNN are cuboids built by stacking  $L$  consecutive OF maps  $F_t$ , where  $I_L(x, y, 2k - 1)$  and  $I_L(x, y, 2k)$  corresponds to the value of the horizontal and vertical OF components located at spatial position  $(x, y)$  and time  $k$ , respectively, ranging  $k$  in the interval  $[1, L]$ .

Since each original video sequence will probably have a different temporal length, and CNN requires a fixed size input, we extract subsequences of  $L$  frames from the full-length sequences. In Figure 5.2 we show five frames distributed every six frames along a subsequence of twenty-five frames in total (*i.e.* frames 1, 7, 13, 19, 25). The first row shows the horizontal component of the OF ( $x$ -axis displacement) and second row shows the vertical component of the OF ( $y$ -axis displacement). It can be observed that most of the motion flow is concentrated in the horizontal component, due to the displacement of the person. In order to remove noisy OF located in the background, as it can be observed in Figure 5.2, we might think in applying a preprocessing step for filtering out those vectors whose magnitude is out of a given interval. However, since our goal in this work is to minimize the manual intervention in the process of gait signature extraction, we will use those OF maps as returned by the OF algorithm.

**Gray-level pixels.** When using CNNs for object detection and categorization, the most popular low level features are raw pixels [72]. In contrast to [123], that uses single RGB frames for action recognition, we build cuboids of gray pixels with the aim of better capturing the important features of the subject appearance. Note that in gait recognition, color is not as informative as it is for object recognition. Therefore, using only gray intensity will eventually help CNN to focus just on the gait-relevant information. An example can be seen in the corresponding row of Figure 5.2.

**Depth maps.** The use of depth information has not been explored much in the field of gait recognition. In [52] they basically use depth to segment people from background and compute the *Gait Energy Volume* descriptor [125]. Castro *et al.* [18] represent depth information in a gray-scale image where the intensity of a pixel is the depth value scaled to  $[0, 255]$ . In our opinion, depth information is rich and should be studied in depth for this problem. Therefore, given a sequence of depth maps, we extract depth volumes that will be used as input data for the corresponding CNN architecture. An example of depth maps can be seen in the bottom row of Figure 5.2.

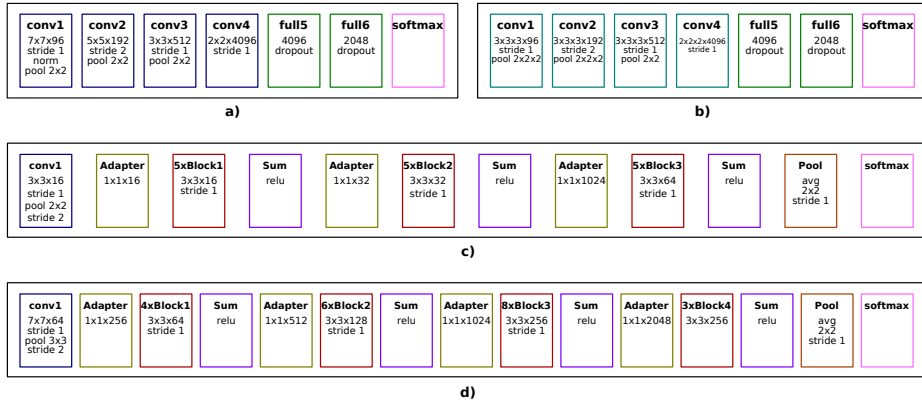


Figure 5.3: **Proposed CNN architectures for gait signature extraction.** a) **2D-CNN:** linear CNN with four 2D convolutions, two fully connected layers and a softmax classifier. b) **3D-CNN:** 3D CNN four 3D convolutions, two fully connected layers and a softmax classifier. c) **ResNet-A:** residual CNN with a 2D convolution, three residual blocks (red boxes), an average pooling layer and a final softmax classifier. d) **ResNet-B:** residual CNN with a 2D convolution, four residual blocks (red boxes), an average pooling layer and a final softmax classifier. More details in the main text.

#### 5.1.1.2. CNN architectures for gait signature extraction

We have selected the three architectures that most frequently appear in the bibliography and produce state-of-the-art results in different topics (*e.g.* action recognition, object detection, etc.). The three proposed architectures are:

- A linear CNN with 2D convolutions (*2D-CNN*), which is the traditional and most common architecture.
- A linear CNN with 3D convolutions and pooling (*3D-CNN*), which is specially designed to capture information in videos.
- A 2D very deep residual CNN (*ResNet*), which produces state-of-the-art results in most challenging tasks.

The input to our CNN is a volume of gray pixels, OF channels or depth maps of size  $N \times N \times L$ . See Section 5.1.2 for the actual values of  $N$  and  $L$  used in the experiments.

We describe below the four models compared in the experimental section (Sections 5.1.4 and 5.1.5). Note that, along this section, we use the term ‘softmax layer’ to refer to a fully-connected layer with as many units as classes followed by a softmax exponential layer.

**2D-CNN:** This CNN is composed of the following sequence of layers (Figure 5.3a):

- ‘conv1’, 96 filters of size  $7 \times 7$  applied with stride 1 followed by a normalization and max pooling  $2 \times 2$ .
- ‘conv2’, 192 filters of size  $5 \times 5$  applied with stride 2 followed by max pooling  $2 \times 2$ .
- ‘conv3’, 512 filters of size  $3 \times 3$  applied with stride 1 followed by max pooling  $2 \times 2$ .
- ‘conv4’, 4096 filters of size  $2 \times 2$  applied with stride 1.
- ‘full5’, fully-connected layer with 4096 units and dropout.
- ‘full6’, fully-connected layer with 2048 units and dropout.
- ‘softmax’, softmax layer with as many units as subject identities.

All convolutional layers use the rectification (ReLU) activation function.

**3D-CNN:** As optical flow has two components and the CNN uses temporal kernels, the network is split into two branches:  $x$ -flow and  $y$ -flow. Therefore, each branch contains half of the total filters described below. Then, this CNN is composed by the following sequence of layers (Figure 5.3b):

- ‘conv1’, 96 filters of size  $3 \times 3 \times 3$  applied with stride 1 followed by a max pooling  $2 \times 2 \times 2$ .
- ‘conv2’, 192 filters of size  $3 \times 3 \times 3$  applied with stride 2 followed by max pooling  $2 \times 2 \times 2$ .
- ‘conv3’, 512 filters of size  $3 \times 3 \times 3$  applied with stride 1 followed by max pooling  $2 \times 2 \times 2$ .
- ‘conv4’, 4096 filters of size  $2 \times 2 \times 2$  applied with stride 1.
- ‘concat’, concatenation of both branches ( $x$ -flow and  $y$ -flow).
- ‘full5’, fully-connected layer with 4096 units and dropout.

- ‘*full6*’, fully-connected layer with 2048 units and dropout.
- ‘*softmax*’, softmax layer with as many units as subject identities.

All convolutional layers use the rectification (ReLU) activation function.

**ResNet-A:** This CNN is composed by the following sequence of layers and residual blocks (a sequences of two convolutions of size  $3 \times 3$ , as defined in [48] for CIFAR Dataset). This model is specially designed for small datasets with low variability because this kind of networks tends to overfit due to its high number of layers. As our architecture follows the indications defined by the authors [48], we only describe the main blocks (Figure 5.3c):

- ‘*conv1*’, 16 filters of size  $3 \times 3$  applied with stride 1 followed by a max pooling  $2 \times 2$  and stride 2.
- ‘*block 1*’, 5 residual blocks with convolutions of 16 filters of size  $3 \times 3$  applied with stride 1.
- ‘*block 2*’, 5 residual blocks with convolutions of 32 filters of size  $3 \times 3$  applied with stride 1.
- ‘*block 3*’, 5 residual blocks with convolutions of 64 filters of size  $3 \times 3$  applied with stride 1.
- ‘*average pooling*’, size  $8 \times 8$  with stride 1.
- ‘*softmax*’, softmax layer with as many units as subject identities.

All convolutional layers use the rectification (ReLU) activation function and batch normalization.

**ResNet-B:** This model is an extension of the model ResNet-A. The number and size of layers of this model is increased and it is specially designed for datasets with high variability (*e.g.* CASIA-B). This CNN is composed by the following sequence of layers and residual blocks (a sequence of three convolutions of size  $1 \times 1$ ,  $3 \times 3$  and  $1 \times 1$ , as defined in [48]). As our architecture follows the indications defined by the authors, we only describe the main blocks (Figure 5.3d):

- ‘*conv1*’, 64 filters of size  $7 \times 7$  applied with stride 1 followed by a max pooling  $3 \times 3$  and stride 2.
- ‘*block 1*’, 4 residual blocks with convolutions of 64 filters of size  $3 \times 3$  applied with stride 1.

- ‘*block 2*’, 6 residual blocks with convolutions of 128 filters of size  $3 \times 3$  applied with stride 1.
- ‘*block 3*’, 8 residual blocks with convolutions of 256 filters of size  $3 \times 3$  applied with stride 1.
- ‘*block 4*’, 3 residual blocks with convolutions of 256 filters of size  $3 \times 3$  applied with stride 1.
- ‘*average pooling*’, size  $2 \times 2$  with stride 1.
- ‘*softmax*’, softmax layer with as many units as subject identities.

All convolutional layers use the rectification (ReLU) activation function and batch normalization.

### Model training

For 2D and 3D models, we perform an incremental training to speed up and to facilitate the convergence. In this incremental process, initially, we train a simplified version of each model (*i.e.* less units per layer and no dropout) and, then, we use its weights for initializing the layers of a more complex version of that previous model (*i.e.* 0.1 dropout and more filters and units). By this way, we train three incremental versions using the previous weights until we obtain the final model architecture represented in Figure. 5.3.

During CNN training, the weights are learned using mini-batch stochastic gradient descent algorithm with momentum equal to 0.9 in the first two incremental iterations of the 2D and 3D models, and 0.95 during the last one. Note that ResNet-A and ResNet-B are trained from scratch in just one iteration (without incremental training) so momentum for these nets is set to 0.9. We set weight decay to  $5 \cdot 10^{-4}$  and dropout to 0.4 (when corresponds). The number of epochs is limited to 20 in TUM-GAID and the learning rate is initially set to  $10^{-2}$  and it is divided by ten when the validation error gets stuck. Due to the specifics of the ResNet models, the initial learning rate is set to 0.1.

In CASIA-B we limit the training stage to 30 epochs, the learning rate is initially set to  $10^{-3}$  and it is divided by two when the validation error gets stuck. At each epoch, a mini-batch of 150 samples is randomly selected from a balanced training set (*i.e.* almost the same proportion of samples per class). Note that for ResNet models we use a mini-batch of 64 samples. When the CNN training has converged, we perform five more epochs on the joint set of training and validation samples.

### 5.1.1.3. Single modality

Once we have obtained the gait signatures, the final stage consists in classifying those signatures to derive a subject identity. Although the softmax layer of the CNN is already a classifier (*i.e.* each unit represents the probability of belonging to a class), the fully-connected layers can play the role of gait signatures that can be used as input of a Support Vector Machine (SVM) classifier. Since we are dealing with a multiclass problem, we define an ensemble of  $C$  binary SVM classifiers with linear kernel in an ‘one-vs-all’ fashion, where  $C$  is the number of possible subject identities. Previous works (*e.g.* [21]) indicate that this configuration of binary classifiers is suitable to obtain top-tier results in this problem. Note that we  $L2$ -normalize the top fully-connected layer before using it as feature vector, as early experiments shown improved results.

In Section 5.1.1.1, we split the whole video sequence into overlapping subsequences of a fixed length, and those subsequences are classified independently. Therefore, in order to derive a final identity for the subject walking along the whole sequence, we apply a *majority voting* strategy on the labels assigned to each subsequence.

An alternative way for obtaining a final label for a video  $v$  from the set of subsequences  $\{s_i\}$  is to derive the identity from the product of softmax vectors (*i.e.* probability distributions  $P_i$ ) obtained:

$$P(v = c) = \prod_{i=1}^t P_i(s_i = c), \quad (5.1)$$

where  $t$  is the number of subsequences extracted from video  $v$ ,  $P(v = c)$  is the probability of assigning the identity  $c$  to the person in video  $v$  and  $P_i(s_i = c)$  is the probability of assigning the identity  $c$  to the person in subsequence  $s_i$ .

### 5.1.1.4. Multiple modalities

In the case where several low-level features have been used, we explore different approaches for combining the outputs of the CNN.

**Late fusion.** Focusing on the softmax scores returned by each CNN, we explore the following approaches to combine them: product and weighted sum. These approaches are considered as ‘late fusion’ ones, as fusion is performed on the classification scores.

A) *Product of softmax vectors.* Given a set of  $n$  softmax vectors  $\{P_i\}$  obtained

from a set of different modalities  $\{m_i\}$ , a new score vector  $S_{\text{prod}}$  is obtained as:

$$S_{\text{prod}}(v = c) = \prod_{i=1}^n P_i(m_i = c) \quad (5.2)$$

where  $n$  is the number of modalities used to classify video  $v$ ,  $S_{\text{prod}}(v = c)$  can be viewed as the probability of assigning the identity  $c$  to the person in video  $v$  and  $P_i(m_i = c)$  is the probability of assigning the identity  $c$  to the person in modality  $m_i$ .

*B) Weighted sum of softmax vectors.* Given a set of  $n$  softmax vectors obtained from a set of different modalities  $\{m_i\}$  a new score vector  $S_{\text{ws}}$  is obtained as:

$$S_{\text{ws}}(v = c) = \sum_{i=1}^n \beta_i P_i(m_i = c), \quad (5.3)$$

where  $n$  is the number of modalities used to classify video  $v$ ,  $S_{\text{ws}}(v = c)$  can be viewed as the probability of assigning the identity  $c$  to the person in video  $v$ ,  $P_i(m_i = c)$  is the probability of assigning the identity  $c$  to the person in modality  $m_i$  and  $\beta_i$  is the weight associated to modality  $m_i$ , subject to  $\beta_i > 0$  and  $\sum_{i=1}^n \beta_i = 1$ .

$\beta$  values are selected empirically by cross-validation. Note that the values used for each experiment are specified in its corresponding section.

**Early fusion.** The fusion performed at descriptor level is known as ‘early fusion’.

In our case, as we are working with CNNs, early fusion could be performed at any layer before the ‘softmax’ one. Depending on the layer, the combined descriptors are matrices (fusion before a convolutional layer) or vectors (fusion before a fully-connected layer). We have tried all the possible fusion locations for our CNNs and we have selected the best solution according to the results obtained. In our case, the best early fusion location is after layer ‘full6’ of each modality. The activations of those layers are concatenated and fed into a new set of layers to perform the actual fusion. Therefore, we extend the 2D and 3D networks shown in Figure 5.3 with the set of additional layers summarized in Figure 5.4:

‘**concat:**’ concatenation layer;

‘**full7:**’ fully-connected layer with 4096 units, ReLU and dropout;

‘**full8:**’ fully-connected layer with 2048 units, ReLU and dropout;



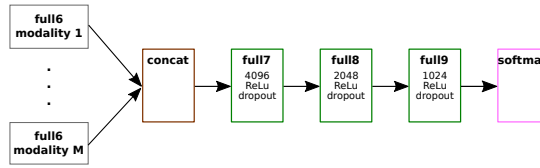


Figure 5.4: **Proposed set of layers for early fusion.** A concatenation layer and three fully-connected layers are followed by a softmax classifier used to directly derive an identity. More details in the main text.

‘*full9:*’ fully-connected layer with 1024 units, ReLU and dropout; and,  
‘*softmax:*’ softmax layer with as many units as subject identities.

During the training process, the weights of the whole CNN (the branch of each modality and the fusion layers) are trained altogether, automatically learning the best combination of weights for the modalities. From our point of view, this kind of fusion is considered early as it is not performed at classification-score level, as done above.

For ResNet models, due to their high number of layers, we do not stack more fully-connected layers to prevent overfitting. Therefore, the selected early fusion architecture is the same as for the rest of models but without fully-connected layers:

‘*concat:*’ concatenation layer;

‘*softmax:*’ softmax layer with as many units as subject identities and dropout.

### 5.1.2. Implementation details

We ran our experiments on a computer with 32 cores at 2.3 GHz, 256 GB of RAM and a GPU NVidia Titan X Pascal, with MatConvNet library [133] running on Matlab 2016a for Ubuntu 16.04.

For the following experiments with CNN, we resized all the videos to a common resolution of  $80 \times 60$  pixels, keeping the original aspect ratio of the video frames. Preliminary experiments support this choice [23], as this size exhibits a good trade-off between computational cost and recognition performance. Note that resolution  $80 \times 60$  is 4 times lower than original CASIA-B and 8 times lower than TUM-GAID one.

Given the resized video sequences, we compute dense  $OF$  on pairs of frames by using the method of Farneback [36] implemented in OpenCV library [15]. In parallel, people are located in a rough manner along the video sequences by background subtraction [65]. Then, we crop the video frames to remove part of the background, obtaining video frames of  $60 \times 60$  pixels (full height is kept) and to align the subsequences (people are  $x$ -located in the middle of the central frame, #13) as in Figure 5.2.

Finally, from the cropped  $OF$  maps, we build subsequences of 25 frames by stacking  $OF$  maps with an overlap of  $\mathcal{O}\%$  frames. In our case, we chose  $\mathcal{O} = 80\%$ , that is, to build a new subsequence, we use 20 frames of the previous subsequence and 5 new frames. For most state-of-the-start datasets, 25 frames cover almost one complete gait cycle, as stated by other authors [9]. Therefore, each  $OF$  volume has size  $60 \times 60 \times 50$ .

The same process described above is applied to the gray pixels and depth inputs, obtaining volumes of size  $60 \times 60 \times 25$ . Before feeding the CNN with those data volumes, the mean of the training set for each modality is subtracted to the input data. Both gray and depth values are normalized to the range  $[0, 255]$ . Note that in CASIA-B, due to the high variability between viewpoints, it is necessary to normalize gray values to the range  $[0, 1]$ .

To increase the amount of training samples we add mirror sequences and apply spatial displacements of  $\pm 5$  pixels in each axis, obtaining a total of 8 new samples from each original sample.

### 5.1.3. Performance evaluation

For each test sample, we return a sorted list of possible identities, where the top one identity corresponds to the largest scored one. Therefore, we use the following metrics to quantitative measure the performance of the proposed system: *rank-1* and *rank-5*. Metric *rank-1* measures the percentage of test samples where the top one assigned identity corresponds to the right one. Metric *rank-5* measures the percentage of test samples where the ground truth identity is included in the first five ranked identities for the corresponding test sample. Note that *rank-5* is less strict than *rank-1* and, in a real system, it would allow to verify if the target subject is any of the top five most probably ones. Final results at sequence level are obtained by applying a majority vote strategy except in the product of softmax scores which is the only case where we have probabilities between 0 and 1 and therefore, we can multiply them for obtaining a sequence probability.

Along this section, we are going to use the following notation: ‘SM-Vote’: softmax decision followed by majority voting to obtain the sequence level results; ‘SM-Prod’: softmax decision followed by the product of the scores to obtain the sequence level results; ‘SVM-L2’: SVM with L2 normalization of the features; ‘SVM-SM’: SVM trained with the scores of the softmax.

#### 5.1.4. Experimental results on TUM-GAID

In ‘TUM Gait from Audio, Image and Depth’ (TUM-GAID) 305 subjects perform two walking trajectories in an indoor environment. The first trajectory is performed from left to right and the second one from right to left. Therefore, both sides of the subjects are recorded. Two recording sessions were performed, one in January, where subjects wore heavy jackets and mostly winter boots, and the second in April, where subjects wore different clothes. The action is captured by a Microsoft Kinect sensor which provides a video stream with a resolution of  $640 \times 480$  pixels with a frame rate of approximately 30 fps. Some examples can be seen in the left part of Figure 5.5 depicting the different conditions included in the dataset.

Hereinafter the following nomenclature is used to refer each of the four walking conditions considered: *normal* walk ( $N$ ), carrying a *backpack* of approximately 5 kg ( $B$ ), wearing coating *shoes* ( $S$ ), as used in clean rooms for hygiene conditions, and *elapsed time* ( $TN$ - $TB$ - $TS$ ). Each subject of the dataset is composed of: six sequences of normal walking ( $N1$ ,  $N2$ ,  $N3$ ,  $N4$ ,  $N5$ ,  $N6$ ), two sequences carrying a bag ( $B1$ ,  $B2$ ) and two sequences wearing coating shoes ( $S1$ ,  $S2$ ). In addition, 32 subjects were recorded in both sessions (*i.e.* January and April) so they have 10 additional sequences ( $TN1$ ,  $TN2$ ,  $TN3$ ,  $TN4$ ,  $TN5$ ,  $TN6$ ,  $TB1$ ,  $TB2$ ,  $TS1$ ,  $TS2$ ). Therefore, the overall amount of videos is 3400.

We follow the experimental protocol defined by the authors of the dataset [52]. Three subsets of subjects are proposed: training, validation and testing. The training set is used for obtaining a robust model against the different covariates of the dataset. This partition is composed of 100 subjects and the sequences  $N1$  to  $N6$ ,  $B1$ ,  $B2$ ,  $S1$  and  $S2$ . The validation set is used for validation purposes and contains 50 different subjects with the sequences  $N1$  to  $N6$ ,  $B1$ ,  $B2$ ,  $S1$  and  $S2$ . Finally, the test set contains other 155 different subjects used in the test phase. As the set of subjects is different between the test set and the training set, a new training of the identification model must be performed. For this purpose, the authors reserve the sequences  $N1$  to  $N4$ , from the subject test set, to train the model again and the rest of sequences are used for testing and to obtain the

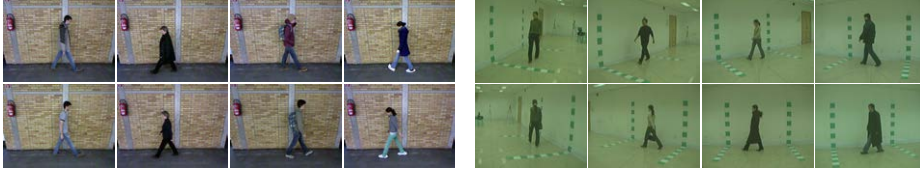


Figure 5.5: **Datasets for gait recognition.** (left) **TUM-GAID.** People walking indoors under four walking conditions: normal walking, wearing coats, carrying a bag and wearing coating shoes. Top and bottom rows show the same set of subjects but in different months of the same year. (right) **CASIA-B.** People walking indoors recorded from eleven camera viewpoints and under three walking conditions: normal walking, wearing coats and carrying a bag.

accuracy of the model. In the *elapsed time* experiment, the temporal sequences ( $TN1$ ,  $TN2$ ,  $TN3$ ,  $TN4$ ,  $TN5$ ,  $TN6$ ,  $TB1$ ,  $TB2$ ,  $TS1$ ,  $TS2$ ) are used instead of the normal ones and the subsets are: 10 subjects in the training set, 6 subjects in the validation set and 16 subjects in the test set.

In our experiments, after parameter selection, the validation sequences are added to the training set for fine-tuning the final model.

In this section, we first examine the impact of CNN architectures in automatic extraction of gait signatures from diverse low-level features, studying which one is the more convenient for the different scenarios. Afterwards, we evaluate the impact of combining gait signatures from different low-level features for people identification. Finally, we compare our results to the state-of-the-art ones.

#### 5.1.4.1. Architecture and feature evaluation

In this experiment, we evaluate the individual contribution of each low-level feature (*i.e.* gray pixels, optical flow and depth maps) and each architecture (*i.e.* 2D, 3D and ResNet) for extracting discriminative gait signatures. Note that, as this dataset only contains a single viewpoint, ResNet models tend to overfit due to the lack of variability in the training data. Therefore, we use ResNet-A (see Section 5.1.1.2 for more details) which is shallower than traditional ResNet models. Tabs. 5.1, 5.2 and 5.3 summarize the identification results obtained on TUM-GAID with each modality: *Gray*, *OF* and *Depth*. Each column contains the results for rank-1 (R1) and rank-5 (R5) for each scenario. The last column ‘AVG’ is the average of each case (temporal and non temporal) weighted by the

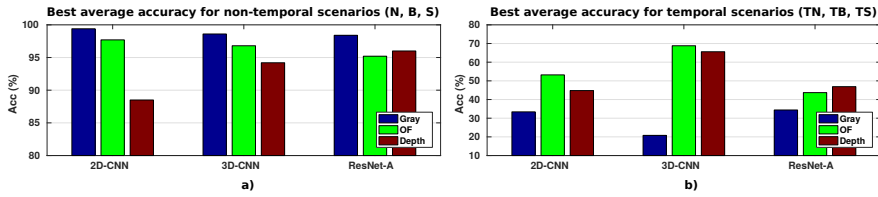


Figure 5.6: **Best average accuracy.** a) non-temporal scenarios (N, B, S) b) temporal scenarios (TN, TB, TS)

number of classes.

Table 5.1: **Feature selection on TUM-GAID *Gray*-modality.** Percentage of correct recognition by using *rank-1* (R1) and *rank-5* (R5) metrics. Each row corresponds to a different classifier and modality. Each column corresponds to a different scenario. Best average results are marked in bold.

		<i>N</i>		<i>B</i>		<i>S</i>		<i>TN</i>		<i>TB</i>		<i>TS</i>		<i>AVG</i>	
		R1	R5	R1	R5	R1	R5	R1	R5	R1	R5	R1	R5	R1	R5
2D-CNN	<i>SM-Vote</i>	99.4	100	99.0	99.7	98.4	99.7	31.3	53.1	34.4	65.6	34.4	62.5	92.8	96.1
	<i>SM-Prod</i>	100	100	99.7	99.7	98.4	99.7	28.1	62.5	37.5	71.9	34.4	59.4	<b>93.2</b>	96.5
	<i>SVM+L2</i>	100	100	99.7	99.7	98.4	99.7	34.4	68.8	31.3	78.1	34.4	68.8	<b>93.2</b>	<b>97.2</b>
	<i>SVM-SM</i>	99.4	99.7	99.4	99.4	97.4	98.7	28.1	65.6	34.4	71.9	34.4	68.8	92.5	96.4
3D-CNN	<i>SM-Vote</i>	99.7	100	98.4	99.7	96.8	99	21.9	50	21.9	46.9	12.5	43.8	90.9	94.6
	<i>SM-Prod</i>	97.7	97.7	93.9	94.2	91.3	91.6	18.8	37.5	21.9	37.5	12.5	31.3	87.1	89
	<i>SVM+L2</i>	100	100	98.1	99.4	97.7	99	18.8	56.3	28.1	62.5	15.6	62.5	91.3	95.8
	<i>SVM-SM</i>	99.7	99.7	98.4	99	96.8	97.7	21.9	43.8	21.9	53.1	12.5	43.8	90.9	93.9
RESNET-A	<i>SM-Vote</i>	99.4	100	95.8	99.4	96.1	99	25	62.5	34.4	98.8	25	59.4	90.6	96.1
	<i>SM-Prod</i>	99	100	96.5	99.4	95.5	99	28.1	56.3	34.4	68.8	25	56.3	90.7	95.8
	<i>SVM+L2</i>	100	100	97.4	99	97.7	100	34.4	53.1	34.4	53.1	34.4	50	92.4	95.2
	<i>SVM-SM</i>	100	100	95.8	97.7	97.1	98.7	25	50	25	62.5	25	56.3	90.8	94.8

In Figure 5.6 appears the best average performance for non-temporal and temporal scenarios per modality. If we focus on the non-temporal scenarios (*N*, *B* and *S*), we can see that features based on *Gray* or *Depth* are able to outperform the results obtained with *OF*. On the other hand, if we focus on the temporal scenarios (*TN*, *TB* and *TS*), the worst results are obtained with *Gray*. These results evidence the weakness of appearance models under conditions with high variability between training and test samples (like our temporal experiment). However, *OF* models have a better sturdiness against appearance changes on the

Table 5.2: **Feature selection on TUM-GAID *OF*-modality.** Percentage of correct recognition by using *rank-1* (R1) and *rank-5* (R5) metrics. Each row corresponds to a different classifier and modality. Each column corresponds to a different scenario. Best average results are marked in bold.

		<i>N</i>		<i>B</i>		<i>S</i>		<i>TN</i>		<i>TB</i>		<i>TS</i>		<i>AVG</i>	
		R1	R5	R1	R5	R1	R5	R1	R5	R1	R5	R1	R5	R1	R5
2D-CNN	<i>SM-Vote</i>	99.4	100	97.4	100	96.4	99.4	53.1	96.9	43.8	87.5	56.3	93.8	93.4	<b>99.1</b>
	<i>SM-Prod</i>	99.4	100	97.7	100	96.1	99.4	56.3	87.5	43.8	84.4	59.4	90.6	93.6	98.7
	<i>SVM+L2</i>	99.4	100	96.5	99.4	96.8	99.4	50.0	90.6	56.3	84.4	43.8	90.6	93.1	98.6
	<i>SVM-SM</i>	99.0	100	96.8	98.4	95.5	98.7	53.1	78.1	50.0	81.3	56.3	91.3	93.0	97.6
3D-CNN	<i>SM-Vote</i>	99	99.4	95.5	99.7	94.2	98.1	65.6	90.6	65.6	93.8	59.4	87.5	93.2	98.3
	<i>SM-Prod</i>	98.7	99.7	97.1	99.4	94.5	98.7	71.9	87.5	68.8	87.5	65.6	84.4	<b>94.1</b>	98.1
	<i>SVM+L2</i>	98.7	99.4	93.9	99	92.6	98.4	65.6	87.5	65.6	81.3	56.3	90.6	92	97.8
	<i>SVM-SM</i>	98.7	99	95.5	99.4	94.2	97.1	65.6	90.6	65.6	81.3	59.4	84.4	93.1	97.3
RESNET-A	<i>SM-Vote</i>	94.5	99.7	81	98.4	85.1	97.7	34.4	93.8	34.4	90.6	37.5	93.8	82.1	98.1
	<i>SM-Prod</i>	95.2	99.4	81	98.7	86.1	97.7	34.4	96.7	40.6	93.8	43.8	93.8	83	98.2
	<i>SVM+L2</i>	99.4	99.4	93.9	98.1	92.2	98.1	37.5	87.5	40.6	81.3	53.1	90.6	90.4	97.4
	<i>SVM-SM</i>	97.4	98.7	89.7	96.5	89.6	92.6	37.5	75	43.8	84.4	46.9	75	87.6	95.4

Table 5.3: **Feature selection on TUM-GAID *Depth*-modality.** Percentage of correct recognition by using *rank-1* (R1) and *rank-5* (R5) metrics. Each row corresponds to a different classifier and modality. Each column corresponds to a different scenario. Best average results are marked in bold.

		<i>N</i>		<i>B</i>		<i>S</i>		<i>TN</i>		<i>TB</i>		<i>TS</i>		<i>AVG</i>	
		R1	R5	R1	R5	R1	R5	R1	R5	R1	R5	R1	R5	R1	R5
2D-CNN	<i>SM-Vote</i>	98.4	100	65.8	90.0	96.8	99.7	34.4	93.8	34.4	93.8	50.0	84.4	82.6	96.0
	<i>SM-Prod</i>	98.7	100	66.1	90.7	96.8	99.7	43.8	90.6	40.6	87.5	46.8	84.4	83.1	95.9
	<i>SVM+L2</i>	99.0	99.7	69.4	85.8	97.1	99.7	46.9	84.4	37.5	81.3	50.0	84.4	84.4	94.0
	<i>SVM-SM</i>	98.7	99.0	65.8	77.7	96.8	98.4	34.4	68.8	40.6	59.4	43.8	68.8	82.7	89.3
3D-CNN	<i>SM-Vote</i>	97.7	98.4	84.2	96.5	96.8	99.4	68.8	100	50	100	75	100	90.3	<b>98.3</b>
	<i>SM-Prod</i>	98.4	100	86.8	96.1	97.4	99.4	62	87.4	53.1	96.9	78.1	100	<b>91.4</b>	98.2
	<i>SVM+L2</i>	96.6	98.7	78.7	91.6	96.8	99.4	71.9	96.9	46.9	90.6	68.8	96.9	88.1	96.4
	<i>SVM-SM</i>	98.4	99	83.6	88.4	96.4	98.4	68.8	81.3	53.1	65.6	75	87.5	90.3	93.7
RESNET-A	<i>SM-Vote</i>	77.7	99.4	60	91.6	71.8	97.4	56.3	81.3	37.5	81.3	46.9	90.6	67.7	95
	<i>SM-Prod</i>	77.1	98.4	60	91.3	70.9	96.1	56.3	78.1	34.4	81.3	46.9	87.5	67.1	94.1
	<i>SVM+L2</i>	99.4	99.7	91.6	97.7	97.1	99.4	31.3	50	21.9	46.9	21.9	53.1	89.4	94.4
	<i>SVM-SM</i>	98.4	99.4	86.5	96.5	87.9	93.6	50	62.5	34.4	53.1	46.9	62.5	86.5	93

inputs. On average, the best results are obtained when using optical flow (*OF*) as base for extracting the gait signature.

With regard to the type of architecture, the behaviour of all of them is very similar on the non-temporal scenarios. However, for the temporal scenarios, 3D-CNN offers its best results in combination with either *OF* or *Depth*, whereas 2D-CNN and ResNet work better with *Gray*. Considering the average accuracy over all the scenarios, 2D-CNN works better with *Gray*, and 3D-CNN with both *OF* and *Depth*.

Finally, note that all the strategies employed for obtaining the identity at video level offer similar performance. However, *SM-Prod* seems to work slightly better on average. Recall that it is defined as the product of probabilities obtained at the softmax layer (see Section 5.1.1.3), what does not require to train an additional classifier as SVM.

#### 5.1.4.2. Feature fusion

As we can use three types of low-level features from TUM-GAID, we study here the benefits of fusing information from the different sources. We are going to use as basis the data of Tabs. 5.1, 5.2 and 5.3, concretely, data obtained with the *product of softmax vectors* on each modality. We have experimented with three types of fusion methods for all the combinations that include optical flow, chosen due to its sturdiness under all walking conditions.

Firstly, we analyse the results within each type of architecture. The results of Tab. 5.4 correspond to 2D-CNN and indicate that, in general, the best option is to combine all three modalities for all fusion methods except for *SM Prod* where it is better to use only *OF* and *Gray*. Note that for *Weighted Sum*, we have used the weights 0.4, 0.3 and 0.3 for *OF*, *Gray* and *Depth*, respectively, when we fuse all modalities. In the case of only two modalities, we use weights 0.6 and 0.4 for *OF* and the other modality, respectively. According to the results, in non temporal scenarios we only improve the results with respect to single modalities in scenario *S* while in the other scenarios we obtain similar results. Regarding the fusion strategy, the proposed *Early* fusion CNN provides on average the best results.

Focusing on the results obtained with the 3D-CNN (Tab. 5.5), the best average accuracy is reported by the combination of all modalities by *W Sum*. However, it is only slightly better than the best result obtained by using only *OF*. Due to the low accuracy obtained with *Gray*, combining it with other features worsen the fused results.

Table 5.4: **Fusion strategies in TUM-GAID with 2D-CNN.** Percentage of correct recognition for different modalities and fusion methods. Each row corresponds to a different fusion strategy. Best results are marked in bold.

Fusion	Modalities	$N$	$B$	$S$	$TN$	$TB$	$TS$	$AVG$
Single	Gray	100	99.7	98.4	28.1	37.5	34.4	93.2
	OF	99.4	97.7	96.1	56.3	43.8	59.4	93.6
	Depth	98.7	66.1	96.8	43.8	40.6	46.8	83.1
SM Prod	OF-Gray	99.7	99.7	99.0	40.6	37.5	53.1	94.3
	OF-Depth	92.9	88.1	98.7	59.4	40.6	46.9	89.1
	All	92.9	90.0	99.0	56.3	56.3	50.0	90.2
W. Sum	OF-Gray	99.4	98.4	98.7	50.0	34.4	53.1	93.9
	OF-Depth	97.7	93.9	99.0	53.1	43.8	59.4	92.7
	All	99.0	98.1	99.7	50.0	34.4	53.1	94.0
Early	OF-Gray	99.4	98.7	97.7	56.3	43.8	43.8	93.9
	OF-Depth	99.0	96.1	96.4	53.1	53.1	46.9	92.9
	All	99.4	98.4	98.7	56.3	53.1	46.9	<b>94.5</b>

Finally, the ResNet architecture (see Tab. 5.6) shows unexpected low fusion results. It may indicate that the probability distribution on the classes obtained at the softmax layer does not show clearly defined maxima, and small changes in those values cause important changes in the final classes. However, *Early Fusion* improves the results on average for the combination *OF* and *Gray*, what indicates that adding more inputs to the training process can be beneficial to avoid local minima.

In summary, by using multimodal information the recognition accuracy improves 0.9% with respect to the best single modality (*i.e.* *OF*).

#### 5.1.4.3. State-of-the-art on TUM-GAID

In Tab. 5.7, we compare our results with state-of-the-art in TUM-GAID under all modalities previously employed (*Gray*, *OF*, *Depth* and *Fusion*). First of all, we would like to remark that our approach uses a resolution of  $80 \times 60$  while the rest of methods use  $640 \times 480$ . Therefore, our method uses 64 times less information. If we focus on the visual modality (*Gray* in our case), we can see that our method outperforms previous results in non temporal scenarios establishing a new state-of-the-art. On the other hand, in the temporal scenarios we have lower results



Table 5.5: **Fusion strategies in TUM-GAID with 3D-CNN.** Percentage of correct recognition for different modalities and fusion methods. Each row corresponds to a different fusion strategy. Best results are marked in bold.

Fusion	Modalities	$N$	$B$	$S$	$TN$	$TB$	$TS$	$AVG$
Single	Gray	97.7	93.9	91.3	18.8	21.9	12.5	87.1
	OF	98.7	97.1	94.5	71.9	68.8	65.6	94.1
	Depth	98.4	86.8	97.4	62	53.1	78.1	91.4
SM Prod	OF-Gray	93.5	84.8	83.5	12.5	12.5	15.6	80.4
	OF-Depth	92.2	97.4	96.8	78.1	62.5	15.6	91.4
	All	78.4	84.2	83.5	12.5	21.9	12.5	75.8
W. Sum	OF-Gray	97.4	98.1	96.1	71.9	50	53.1	93.6
	OF-Depth	95.5	96.5	96.8	65.6	68.8	53.1	93.1
	All	96.8	98.4	97.1	65.6	65.6	59.4	<b>94.3</b>
Early	OF-Gray	99.4	96.8	94.5	62.5	50	56.3	93.1
	OF-Depth	84.8	97.4	97.4	71.9	68.8	71.9	91.1
	All	99.7	98.7	97.7	34.4	25	31.3	92.3

than the other methods due to the high variability in visual information. Then, if we focus on *OF*, we can see that the best results are obtained by PFM [21] with a resolution of  $640 \times 480$ . Nevertheless, if we apply PFM with a resolution of  $80 \times 60$ , its results worsen dramatically and our CNN is able to outperform it in all scenarios. If we compare our CNN with other deep learning approaches presented in the literature, only MTaskCNN-7NN [88] is able to improve our approach. This model has been trained in a multi-task fashion so, during training, there are more information available to optimize the weights. If we focus on the other deep learning approaches, we can see that we obtain similar results (only a 0.2% lower) on average but, we obtain the state-of-the-art for temporal scenario. In *Depth* modality, we can see that our method obtains better results than other methods, which use full resolution frames, in all cases except  $N$ . Nevertheless, on average, we are able to obtain more than a 10% of improvement. Finally, if we fuse information from all modalities with a CNN, the average score achieved by both scenarios (temporal and non-temporal) beats all the methods shown in Tab. 5.7 with the exception of PFM ( $640 \times 480$ ) [21] and MTaskCNN-7NN [88], where we are 1.5% and 1.1% below, respectively. However, if we apply the same 7NN approach as in [88], and we fuse the probabilities obtained, we set a new state-of-the-art (96.5% vs 96.0%) for all scenarios with our 3D-CNN-7NN-All

Table 5.6: **Fusion strategies in TUM-GAID with ResNet.** Percentage of correct recognition for different modalities and fusion methods. Each row corresponds to a different fusion strategy. Best average results are marked in bold.

Fusion	Modalities	$N$	$B$	$S$	$TN$	$TB$	$TS$	$AVG$
Single	Gray	99.0	96.5	95.5	28.1	34.4	25.0	90.7
	OF	95.2	81.0	86.1	37.5	40.6	43.8	83.1
	Depth	77.1	60.0	71.0	56.3	34.4	46.9	67.2
SM Prod	OF-Gray	84.8	77.7	79.3	46.9	40.6	50	77.3
	OF-Depth	71.2	63.6	69.6	53.1	37.5	53.1	66.2
	All	79.9	80.7	81.9	56.3	34.4	56.3	77.9
W. Sum	OF-Gray	72.8	60.7	64.4	31.3	31.3	40.6	63
	OF-Depth	68.3	53.9	62.5	37.5	46.9	56.3	60.2
	All	72.5	60	64.7	31.3	28.1	46.9	62.9
Early-RES	OF-Gray	99.4	94.8	97.7	40.6	34.4	43.8	<b>91.9</b>
	OF-Depth	95.8	93.2	96.1	40.6	37.5	43.8	89.9
	All	80.3	87.1	88.4	40.6	50	50	81.7

using Softmax Product as fusion.

### 5.1.5. Experimental results on CASIA-B

In CASIA-B 124 subjects perform walking trajectories in an indoor environment (right part of Figure 5.5). The action is captured from 11 viewpoints (*i.e.* from  $0^\circ$  to  $180^\circ$  in steps of  $18^\circ$ ) with a video resolution of  $320 \times 240$  pixels. Three situations are considered: normal walk (*nm*), wearing a coat (*cl*), and carrying a bag (*bg*). The authors of the dataset indicate that sequences 1 to 4 of the ‘*nm*’ scenario should be used for training the models. Whereas the remaining sequences should be used for testing: sequences 5 and 6 of ‘*nm*’, 1 and 2 of ‘*cl*’ and 1 and 2 of ‘*bg*’. Therefore, we follow this protocol in our experiments, unless otherwise stated. This makes a total of 496 video sequences for training, per camera viewpoint.

We focus here on CASIA-B dataset, which offers different covariate factors and multiple viewpoints. Note that, for the sake of comparison with other methods, we train our models with all cameras and the test is performed only with the  $90^\circ$

Table 5.7: **State-of-the-art on TUM GAID**. Percentage of correct recognition on TUM-GAID for diverse methods published in the literature. Bottom rows of each modality correspond to our proposal, where instead of using video frames at  $640 \times 480$ , a resolution of  $80 \times 60$  is used. Each column corresponds to a different scenario. Best results are marked in bold. (See main text for further details).

Modality	Input Size	Method	N	B	S	Avg	TN	TB	TS	Avg	Global Avg
Visual Data	$640 \times 480$	SDL [154]	-	-	-	-	96.9	-	-	-	-
		GEI [52]	99.4	27.1	52.6	59.7	44.0	6.0	9.0	19.7	56.0
		SEIM [140]	99.0	18.4	96.1	71.2	15.6	3.1	28.1	15.6	66.0
		GVI [140]	99.0	47.7	94.5	80.4	62.5	15.6	62.5	46.9	77.3
		SVIM [140]	98.4	64.2	91.6	84.7	65.6	31.3	50.0	49.0	81.4
		RSM [46]	100	79.0	97.0	92.0	58.0	38.0	57.0	51.0	88.2
OF	Gray $80 \times 60$	2D-CNN-SMP (ours)	100	99.7	98.4	99.4	28.1	37.5	34.4	33.3	93.2
		PFM [21]	99.7	99.0	99.0	99.2	78.1	62.0	54.9	65.0	96.0
	$80 \times 60$	PFM [21]	75.8	70.3	32.3	59.5	50.0	40.6	25.0	38.5	57.5
		OF-CNN-NN [23]	99.7	98.1	95.8	97.9	62.5	56.3	59.4	59.4	94.3
		OF-ResNet-B [19]	99	95.5	97.4	97.3	65.6	62.5	68.8	65.6	94.3
		MTaskCNN-7NN [88]	99.7	97.4	99.7	98.9	59.4	62.5	68.8	63.6	95.6
Depth	$640 \times 480$	3D-CNN-SMP (ours)	98.7	97.1	94.5	96.8	71.9	68.8	65.6	<b>68.8</b>	94.1
		DGHEI [52]	99.0	40.3	96.1	78.5	50.0	0.0	44.0	31.3	74.1
	$80 \times 60$	3D-CNN-SMP (ours)	98.4	86.8	97.4	94.2	62.0	53.1	78.1	64.4	91.4
Fusion	$640 \times 480$	DGHEI + GEI [52]	99.4	51.3	94.8	81.8	66.0	3.0	50.0	39.7	77.9
		2D-CNN-All (ours)	99.4	98.4	98.7	98.8	56.3	53.1	46.9	52.1	94.5
	$80 \times 60$	3D-CNN-7NN-All (ours)	100	99.4	99.4	<b>99.6</b>	75	62.5	62.5	66.7	<b>96.5</b>

camera as done in state-of-the-art approaches [143, 21].

### 5.1.5.1. Architecture and feature evaluation

As this dataset contains eleven viewpoints, ResNet models have enough variability in the training data. Therefore, we use ResNet-B (see Section 5.1.1.2 for more details) which is deeper than ResNet-A. Tab. 5.8 summarizes the identification results obtained on CASIA-B 90° with each modality: *Gray* and *OF*. Note that this dataset does not provide depth information. R1 and R5 columns contain the results for rank-1 (R1) and rank-5 (R5) for each scenario. The last column ‘*AVG*’ is the average of all scenarios. The results at sequence level are obtained by multiplying the scores of the softmax layer. Note that as in CASIA-B there is no training partition to build the model, we have split the dataset into a training set composed of the first 74 subjects and a test set composed of the 50 remaining subjects, following the indications in [143]. During the training process, all viewpoints and training samples are used.

Table 5.8: **Feature selection on CASIA-B 90°: *Gray* and *OF* modalities.** Percentage of correct recognition by using *rank-1* (R1) and *rank-5* (R5) metrics. Each row corresponds to a different classifier and modality, grouped by architecture. Each column corresponds to a different scenario. Best average results are marked in bold.

			<i>nm</i>		<i>bg</i>		<i>cl</i>		<i>AVG</i>	
			R1	R5	R1	R5	R1	R5	R1	R5
Gray	2D	<i>SM-Vote</i>	91	98	82	95	37	82	70	91.7
		<i>SM-Prod</i>	92	100	85	98	45	90	74	96
	3D	<i>SM-Vote</i>	72	93	69	87	33	76	58	85.3
		<i>SM-Prod</i>	81	92	73	90	45	77	66.3	86.3
	RES	<i>SM-Vote</i>	94	100	89	98	42	83	75	93.7
		<i>SM-Prod</i>	96	100	91	98	46	98	<b>77.7</b>	<b>98.7</b>
OF	2D	<i>SM-Vote</i>	99	99	76	90	28	51	67.7	80
		<i>SM-Prod</i>	99	99	78	93	27	62	68	84.7
	3D	<i>SM-Vote</i>	98	99	86	99	37	70	73.7	89.3
		<i>SM-Prod</i>	98	100	88	98	36	67	74	88.3
	RES	<i>SM-Vote</i>	94	100	83	98	47	73	74.7	90.3
		<i>SM-Prod</i>	93	100	85	98	46	71	74.7	89.7

According to the results obtained, we can see that our model is able to identify people with a high accuracy in scenarios *nm* and *bg* while in scenario *cl* we have lower precision due to the high appearance changes. If we focus on the modality used, on average, *Gray* is the best option most of the time. In this dataset, with huge variations between points of view, the shape of the subject seems to be important and it helps to classification. In scenario *cl* our models experiment a huge decrease in accuracy mainly caused by the high variability of coats worn by the subjects. This can be seen in Figure 5.5 on the right part of the last row. In these pictures, the coat occludes the legs and if we add the fact that we have different kind of coats with different number of occurrences, our CNN is not able to learn good features for this scenario due to the high variability and low number of samples.

On the other hand, *OF* seems that it is not able to find a good representation if the shape of the subject changes drastically. We think that this is because of the high variability in the appearance of the subjects seen from the different cameras used for training. Therefore, as the models receive different flow vectors, the training process cannot produce a view-independent model and the global performance decreases. For example, frontal-views produce vectors whose main movement is focused on Y-axis (there is no horizontal displacement of the subject) while lateral-views produce vectors whose movement is focused on X-axis.

If we focus on the different architectures, according to the mean results, it is clear that ResNet-B obtains the best results for each modality. That shows that ResNet is the more powerful model if data with enough variability is available. On the other hand, 3D-CNN obtains really good results for *OF* modality while 2D-CNN achieves good results for *Gray* modality.

#### 5.1.5.2. Feature fusion

In this case, as we only have two modalities, fusion is performed using both. It can be observed in Tab. 5.9 that the best method for fusing *Gray* and *OF* features is, on average, Softmax product followed by weighted sum with weights 0.5 and 0.5 for *Gray* and *OF*, respectively. Focusing on the three architectures, again, the best option is ResNet as it obtains the best results in all cases.

In this dataset, the late fusion of both modalities improves or obtains the same result as single modality CNNs in all cases, apart from special cases where the difference between the accuracy of the fused modalities is huge (*e.g.* *cl* for 2D-CNN). Anyway, on average, fusion always obtains the best results with improvements of more than a 3%. In this case, early fusion is not able to improve

the single modality results. In our opinion, this is due to high variability between viewpoints. In addition, we have observed that the two branches of the network have different convergence speeds, hence the final features are not fused properly producing bad representations.

Table 5.9: **Fusion strategies in CASIA-B 90°**. Percentage of correct recognition with different fusion methods. Each row corresponds to a different fusion method, but the two top rows that correspond to the baseline cases. Best average results are marked in bold.

	<i>2D-CNN</i>				<i>3D-CNN</i>				<i>ResNet</i>			
	<i>nm</i>	<i>bg</i>	<i>cl</i>	<i>AVG</i>	<i>nm</i>	<i>bg</i>	<i>cl</i>	<i>AVG</i>	<i>nm</i>	<i>bg</i>	<i>cl</i>	<i>AVG</i>
<i>Gray</i>	92	85	45	74	81	73	45	66.3	96	91	46	77.7
<i>OF</i>	99	78	27	68	98	88	36	74	93	85	46	74.7
<i>SM-Prod</i>	99	95	41	<b>78.3</b>	98	96	49	<b>81</b>	98	97	63	<b>86</b>
<i>W. Sum</i>	99	94	39	77.3	98	95	46	79.7	98	96	60	84.7
<i>Early</i>	83	61	26	56.7	76	74	46	65.3	67	63	38	56

### 5.1.5.3. State-of-the-art on CASIA-B

In Tab. 5.10, we compare our results with state-of-the-art in CASIA-B under all modalities used before (*Gray* and *OF*) and their fusion. First of all, we would like to remark that our approach uses a resolution of  $80 \times 60$  while the rest of methods use  $320 \times 240$ . Therefore, our method uses 16 times less information. If we focus on the visual modality (*Gray* in our case), we can see that our ResNet-B obtains the best results compared to the state-of-the-art even using the lowest resolution. Indeed, our model sets a new state-of-the-art for visual data. If we focus on *OF*, the best results are obtained by PFM [21] with a resolution of  $640 \times 480$ . Nevertheless, if we apply it with a resolution of  $80 \times 60$ , its results worsen dramatically and our ResNet-B is able to outperform it in all scenarios. With this modality, our model sets the second best result in the state-of-the-art (apart from PFM with full resolution). Finally, our fusion (softmax product) sets the best result and it improves our ResNet-B for *Gray* modality by a 8.3%.

Focusing on [143], which is the closest approach to ours as they use also CNNs, our best average result improves a 16.3% with respect to their best average accuracy. Focusing on individual scenarios, they only improve our results in *cl* scenario if we use a single modality, probably due to the use of a gallery-probe

scheme. During test time, they must compare the test sample with all the probe samples to get all distances, then, they select the class of the probe sample with the lowest distance. This approach is easier but slower than our approach where we only need to propagate the test sample through the CNN to obtain the class. However, if we use our fusion approach, we beat them in all scenarios and we minimize the changes in the shape of *cl* scenario.

### 5.1.6. Conclusions

We have presented a comparative study of multi-feature systems based on CNN architectures for the problem of people identification based on the way the walk (gait). The evaluated architectures are able to extract automatically gait signatures from sequences of gray pixels, optical flow and depth maps. Those gait signatures have been tested on the task of people identification, obtaining state-of-the-art results on two challenging datasets, *i.e.* TUM-GAID and CASIA-B, that cover diverse scenarios (*e.g.* people wearing long coats, carrying bags, changing shoes or camera viewpoint changes).

With regard to the type of input features, we may conclude that, under similar viewpoints (*e.g.* TUM-GAID) the weakest one is *gray pixels*, as it is highly appearance dependant. However, as it could be expected *optical flow* is the one that better encodes body motion. Depth maps work fairly well if changes in appearance are small (*i.e.* *Shoes* scenario). In datasets with multiple viewpoints (*e.g.* CASIA-B), *gray pixels* achieve the best results, probably due to *optical flow* produces extremely different vectors depending on the viewpoint so, during training, the optimization process is not able to build a good multiview representation of the subjects.

Regarding the type of architecture, 2D-CNN produces better results in most cases; 3D-CNN is specially useful in scenarios with appearance changes; ResNet models are designed to be very deep, therefore, they need huge datasets with high variability between samples to perform well. This has been demonstrated in our experiments where ResNet-A produces worse results than the other two architectures for TUM-GAID (dataset with low variability) but, on the other hand, ResNet-B produces the best results for CASIA-B (dataset with high variability).

Finally, the experimental results show that the fusion of multiple features allows to boost the recognition accuracy of the system in many cases or at least, it matches the best results achieved by using a single modality.

As final recommendation and, according to the results obtained, the best models are 3D-CNN and ResNet, being the latter the best option if the dataset

Table 5.10: **State-of-the-art on CASIA-B, camera 90°**. Percentage of correct recognition for several methods on camera 90°. Bottom rows of each modality correspond to our proposal, where instead of using video frames at  $640 \times 480$ , a resolution of  $80 \times 60$  is used. Acronyms: ‘#subjs’ number of subjects used for test; ‘#train’ number of sequences per person used for training; ‘#test’ number of sequences per person used for test. Best results are marked in bold.

<i>Modality</i>	<i>Input Size</i>	<i>Method</i>	<i>#subjs</i>	<i>#train</i>	<i>#test</i>	<i>nm</i>	<i>b<sub>g</sub></i>	<i>cl</i>	<i>Avg</i>
Visual Data	$320 \times 240$	GEI [151]	124	4	2	97.6	52.0	32.7	67.8
		GEI [151]	124	4	2	97.6	52.0	32.7	67.8
		iHMM [58]	84	5	1	94.0	45.2	42.9	60.7
		CGI [135]	124	1	1	88.1	43.7	43.0	58.3
		SDCNN [5]	124	4	2	95.6	-	-	-
	126 × 126	DCNN [144]	124	4	2	81.5	-	-	-
	88 × 128	LBCNN [143]	50	4	2	91.5	63.1	54.6	69.7
	Gray 80 × 60	ResNet-B (ours)	50	4	2	96.0	91.0	46.0	<b>77.7</b>
	320 × 240	PFM [21]	124	4	2	100	100	85.5	<b>95.2</b>
		PFM [21]	124	4	2	88.3	66.5	44.0	66.3
OF	80 × 60	ResNet-B (ours)	50	4	2	93.0	85.0	46.0	74.7
Fusion	80 × 60	ResNet-B-SMP (ours)	50	4	2	98.0	97.0	63.0	<b>86.0</b>



contains enough training data. Regarding to fusion methods, the best option is late fusion approaches and, in our case, product of the softmax scores. As future work, we plan to study in depth our early fusion approach to solve the problem of different convergence rates in the branches.

## 5.2. Incremental learning for gait recognition

Let us suppose we have a system which is able to recognize  $M$  subject identities by their way of walking. Now, we want to add  $N$  new subjects to that system without forgetting the previous  $M$  ones. And after those  $N$ , we want to add  $O$  more and so on. Thus, the objective of the incremental learning is to add new subjects or classes to a trained model without training it from scratch. That is, the new model should be very similar to the original one, but with small differences to deal with the new classes. In this work, we assess the incremental learning applied to gait recognition. Thus, in each incremental step, a set of new subjects is added to the system keeping the knowledge about the old ones. The developed approach is based on our conference paper [20] but applied to gait recognition in videos instead of image recognition. Thus, the core of this approach is the same as the published one and it will be explained in the following sections.

Section 5.2.1 describes our model and Section 5.2.2 shows the incremental learning details of our approach. Section 5.2.3 explains the experiments performed to validate our approach for the gait recognition problem and Section 5.2.4 summarizes the results obtained. Finally, the obtained conclusions are presented in Section 5.2.5. In summary, these conclusions indicate that our approach is able to obtain similar results to models trained in one step, without incremental learning.

### 5.2.1. Our model

Our end-to-end approach uses a deep network trained with a cross-distilled loss function, i.e., cross-entropy together with distillation loss. The network can be based on the architecture of most deep models designed for classification, since our approach does not require any specific properties. A typical architecture for classification can be seen in Figure 5.7, with one *classification layer* and a classification loss. This *classification layer* uses features from the feature extractor to produce a set of *logits* which are transformed into class scores by a softmax layer (not shown in the figure). The only necessary modification is the loss function, described in Section 5.2.1.2. To help our model retain the knowledge acquired

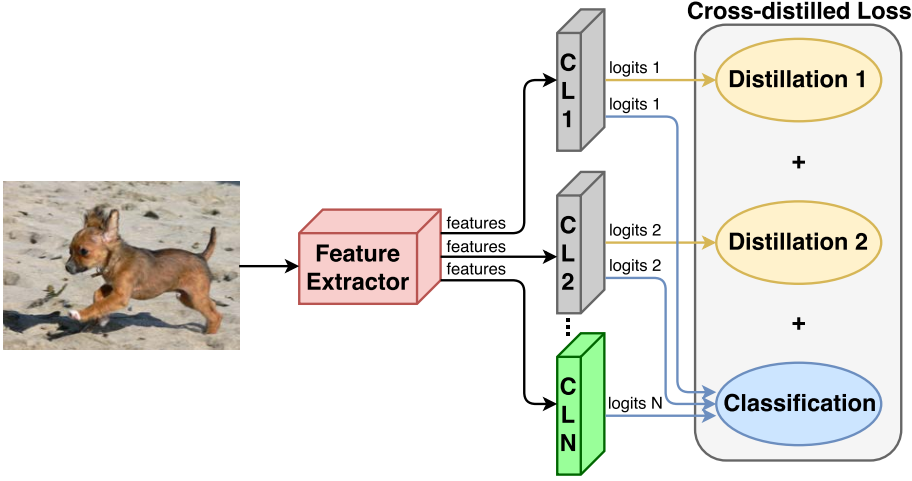


Figure 5.7: **Our incremental model.** Given an input image, the feature extractor produces a set of features which are used by the *classification layers* (CLi blocks) to generate a set of *logits*. Grey *classification layers* contain old classes and their *logits* are used for distillation and classification. The green *classification layer* (CLN block) contains new classes and its *logits* are involved only in classification. (Best viewed in color.)

from the old classes, we use a representative memory that stores and manages the most representative samples from the old classes. In addition to this we perform data augmentation and a balanced fine-tuning. All these components put together allow us to get state-of-the-art results.

#### 5.2.1.1. Representative memory

When a new class or set of classes is added to the current model, a subset with the most representative samples from them is selected and stored in the representative memory. We investigate two memory setups in this work. The first setup considers a memory with a limited capacity of  $K$  samples. As the capacity of the memory is independent of the number of classes, the more classes stored, the fewer samples retained per class. The number of samples per class,  $n$ , is thus given by  $n = \lfloor K/c \rfloor$ , where  $c$  is the number of classes stored in memory, and  $K$  is the memory capacity. The second setup stores a constant number of exemplars per class. Thus, the size of the memory grows with the number of classes.

The representative memory unit performs two operations: selection of new samples to store, and removal of leftover samples.

**Selection of new samples.** This is based on *herding selection* [139], which produces a sorted list of samples of one class based on the distance to the mean sample of that class. Given the sorted list of samples, the first  $n$  samples of the list are selected. These samples are most representative of the class according to the mean. This selection method was chosen based on our experiments testing different approaches, such as random selection, histogram of the distances from each sample to the class mean. The selection is performed once per class, whenever a new class is added to the memory.

**Removing samples.** This step is performed after the training process to allocate memory for the samples from the new classes. As the samples are stored in a sorted list, this operation is trivial. The memory unit only needs to remove samples from the end of the sample set of each class. Note that after this operation, the removed samples are never used again.

### 5.2.1.2. Deep network

**Architecture.** The network is composed of several components, as illustrated in Figure 5.7. The first component is a *feature extractor*, which is a set of layers to transform the input image into a feature vector. The next component is a *classification layer* which is the last fully-connected layer of the model, with as many outputs as the number of classes. This component takes the features and produces a set of *logits*. During the training phase, gradients to update the weights of the network are computed with these *logits* through our cross-distilled loss function. At test time, the loss function is replaced by a softmax layer (not shown in the figure).

To build our incremental learning framework, we start with a traditional, *i.e.*, non-incremental, deep architecture for classification for the first set of classes. When new classes are trained, we add a new *classification layer* corresponding to these classes, and connect it to the *feature extractor* and the component for computing the cross-distilled loss, as shown in Figure 5.7. Note that the architecture of the *feature extractor* does not change during the incremental training process, and only new *classification layers* are connected to it. Therefore, any architecture (or even pre-trained model) can be used with our approach just by adding the incremental classification layers and the cross-distilled loss function when necessary.

**Cross-distilled loss function.** This combines a distillation loss [50], which

retains the knowledge from old classes, with a multi-class cross-entropy loss, which learns to classify the new classes. The distillation loss is applied to the *classification layers* of the old classes while the multi-class cross-entropy is used on all *classification layers*. This allows the model to update the decision boundaries of the classes. The loss computation is illustrated in Figure 5.7. The cross-distilled loss function  $L(\omega)$  is defined as:

$$L(\omega) = L_C(\omega) + \sum_{f=1}^F L_{D_f}(\omega), \quad (5.4)$$

where  $L_C(\omega)$  is the cross-entropy loss applied to samples from the old and new classes,  $L_{D_f}$  is the distillation loss of the *classification layer*  $f$ , and  $F$  is the total number of *classification layers* for the old classes (shown as grey boxes in Figure 5.7).

The cross-entropy loss  $L_C(\omega)$  is given by:

$$L_C(\omega) = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^C p_{ij} \log q_{ij}, \quad (5.5)$$

where  $q_i$  is a score obtained by applying a softmax function to the *logits* of a *classification layer* for sample  $i$ ,  $p_i$  is the ground truth for the sample  $i$ , and  $N$  and  $C$  denote the number of samples and classes respectively.

The distillation loss  $L_D(\omega)$  is defined as:

$$L_D(\omega) = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^C p_{dist_{ij}} \log q_{dist_{ij}}, \quad (5.6)$$

where  $p_{dist_i}$  and  $q_{dist_i}$  are modified versions of  $p_i$  and  $q_i$ , respectively. They are obtained by raising  $p_i$  and  $q_i$  to the exponent  $1/T$ , as described in [50], where  $T$  is the distillation parameter. When  $T = 1$ , the class with the highest score influences the loss significantly, e.g., more than 0.9 from a maximum of 1.0, and the remaining classes with low scores have minimal impact on the loss. However, with  $T > 1$ , the remaining classes have a greater influence, and their higher loss values must be minimized. This forces the network to learn a more fine grained separation between them. As a result, the network learns a more discriminative representation of the classes. Based on our empirical results, we set  $T$  to 2 for all our experiments.

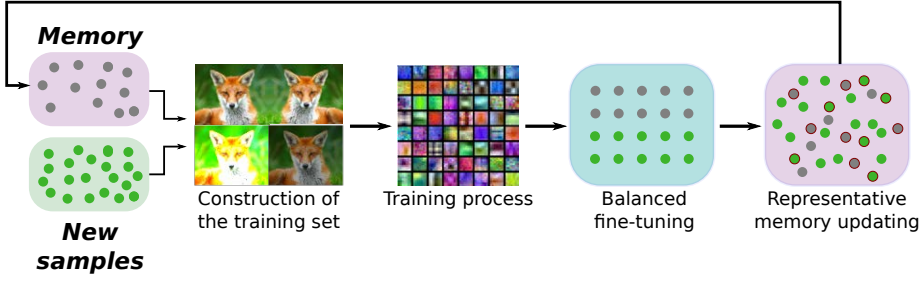


Figure 5.8: **Incremental training.** Grey dots correspond to samples stored in the representative memory. Green dots correspond to samples from the new classes. Dots with red border correspond to the selected samples to be stored in the memory. (Best viewed in color.)

### 5.2.2. Incremental learning

An incremental learning step in our approach consists of four main stages, as illustrated in Figure 5.8. The first stage is the construction of the training set, which prepares the training data to be used in the second stage, the training process, which fits a model given the training data. In the third stage, a fine-tuning with a subset of the training data is performed. This subset contains the same number of samples per class. Finally, in the fourth stage, the representative memory is updated to include samples from the new classes. We now describe these stages in detail.

**Construction of the training set.** Our training set is composed of samples from the new classes and exemplars from the old classes stored in the representative memory. As our approach uses two loss functions, *i.e.*, classification and distillation, we need two labels for each sample, associated with the two losses. For classification, we use the one-hot vector which indicates the class appearing in the image. For distillation, we use as labels the *logits* produced by every *classification layer* with old classes (grey fully-connected layers in Figure 5.7). Thus, we have as many distillation labels per sample as *classification layers* with old classes. To reinforce the old knowledge, samples from the new classes are also used for distillation. This way, all images produce gradients for both the losses. Thus, when an image is evaluated by the network, the output encodes the behaviour of the weights that compose every layer of the deep model, independently of its label. Each image of our training set will have a classification label and  $F$  distillation labels; cf. Eq. 5.4. Note that this label extraction is performed in each incremental step.

Consider an example scenario to better understand this step, where we are performing the third incremental step of our model (Figure 5.7). At this point the model has three *classification layers* ( $N = 3$ ), two of them will process old classes (grey boxes), i.e.,  $F = 2$ , and one of them operates on the new classes (green box). When a sample is evaluated, the *logits* produced by the two *classification layers* with the old classes are used for distillation (yellow arrows), and the *logits* produced by the three *classification layers* are used for classification (blue arrows).

**Training process.** Our cross-distilled loss function (Eq. 5.4) takes the augmented training set with its corresponding labels and produces a set of gradients to optimise the deep model. Note that, during training, all the weights of the model are updated. Thus, for any sample, features obtained from the feature extractor are likely to change between successive incremental steps, and the *classification layers* should adapt their weights to deal with these new features. This is an important difference with some other incremental approaches like [77], where the *feature extractor* is frozen and only the *classification layers* are trained.

**Balanced fine-tuning.** Since we do not store all the samples from the old classes, samples from these classes available for training can be significantly lower than those from the new classes. To deal with this unbalanced training scenario, we add an additional fine-tuning stage with a small learning rate and a balanced subset of samples. The new training subset contains the same number of samples per class, regardless of whether they belong to new or old classes. This subset is built by reducing the number of samples from the new classes, keeping only the most representative samples from each class, according to the selection algorithm described in Section 5.2.1.1. With this removal of samples from the new classes, the model can potentially forget knowledge acquired during the previous training step. We avoid this by adding a temporary distillation loss to the *classification layer* of the new classes.

**Representative memory updating.** After the balanced fine-tuning step, the representative memory must be updated to include exemplars from the new classes. This is performed with the selection and removing operations described in Section 5.2.1.1. First, the memory unit removes samples from the stored classes to allocate space for samples from the new classes. Then, the most representative samples from the new classes are selected, and stored in the memory unit according to the selection algorithm.

### 5.2.3. Experiments

We perform two experiments on TUM-GAID dataset [52] (more details about the dataset in Section 5.1.4). In the first one, we train from scratch a CNN using samples from the 155 subjects set. Since there are three different walking conditions, the CNN has to be trained with samples from all conditions. Therefore, we create a new training set used for this incremental problem. For training, we take the first half of the two first sequences of each subject and walking condition and, for testing, we take the other halves of the sequences. By this way, we are using two sequences of normal (N), bag (B) and shoes (S) divided into two subsequences, one for training and one for testing. Note that we have to use information from both sequences as the subjects walk in different directions in each one. Therefore, if a full sequence was used for training and the other for testing, the performance of the model would be poor, as the test conditions would differ from the training ones. Intuitively, applying the mirror to the original sequences should help to deal with this problem as we would have samples with subjects walking in both directions. However, early experiments showed that the performance does not improve and it is necessary to adopt the partitioning policy explained above. For this experiment, the memory size of the incremental model is fixed to 1240 samples (8 samples per subject in the last incremental step).

In the second experiment, we start from a network pretrained on the 150 subjects set. This training set contains samples from all walking conditions so the network knows how to deal with them. Thus, during the training process, we have to retain the old knowledge to deal with the walking conditions and, at the same time, add new subject identities using the standard training samples (*i.e.* 4 sequences of normal gait (N)). To do this, we use the pretrained network to initialize the weights of the network used during the incremental process for the 155 subjects. By this way, the incremental network has notions about how to treat samples with different walking conditions. This experiment allows us to compare the final performance of our incremental approach with the non-incremental approaches published in the literature. For this experiment, the memory size is fixed to 1240 samples (8 samples per subject in the last incremental step).

For both experiments, we use the following incremental steps: 5, 10, 20, 31<sup>1</sup>, 50 to compare the impact of the number of new classes during the training. The architecture, hyper-parameters and training process is the same as the one performed in [20]. Note that we focus on the non-temporal experiment of TUM-

---

<sup>1</sup>Note that we chose this particular value due to 31 and 5 are the divisors of 155, which is the number of classes of the problem.

GAID as the temporal one is composed of a very limited number of subjects.

#### 5.2.4. Results

Table 5.11 contains the accuracy for the first experiment and the baseline accuracy using a network trained from scratch (without incremental learning). Note that the accuracy shown in the table corresponds to the accuracy obtained in the last incremental step, that is, when the model is trained for the 155 subjects that compose the dataset. By this way, we can compare the incremental results with the non-incremental ones. Moreover, for brevity, the accuracy is computed for the whole test set without differentiating between walking conditions.

According to the results, step sizes of 10, 20 and 31 classes obtain similar results, being 31 the best option with an accuracy of 97.3%. Comparing these values with the obtained by the baseline approach using the same data partition, the accuracy of our incremental method only decreases in 2.1%. Therefore, our incremental approach obtains similar results to a model trained from scratch with all classes at the same time. This way, using our incremental learning approach, it is not necessary to keep all the training data (we only need to store a small subset of representative samples) and the training process can be performed incrementally as new classes arrive.

Table 5.12 contains the accuracy for the second experiment and the baseline accuracy using a network trained from scratch (without incremental learning). Like in the previous experiment, the accuracy shown in the table corresponds to the accuracy obtained in the last incremental step, that is, when the model is trained for the 155 subjects that compose the dataset. Moreover, for brevity, the accuracy is computed for the whole test set without differentiating between walking conditions.

According to the results, a step size of 5 classes is the best option with an accuracy of 85.2%. Comparing this value with the obtained by the baseline approach using the same data partition, the accuracy of our incremental method only decreases in 2.2%. This drop in the accuracy is comparable to the obtained in the first experiment, showing again that our approach is a good solution that obtains results close to the model trained from scratch. In general for this case, the performance of both models is lower because only data from the normal walking condition are available during the incremental training.



Table 5.11: **Incremental learning results for experiment 1.** Percentage of correct recognition with different step size. Each row represents a different model. Each column represents a different step size. More details in the main text.

Model / # classes	5	10	20	31	50
Ours-Experiment1	95.8	97.1	97.1	97.3	91.4
Baseline	99.4				

Table 5.12: **Incremental learning results for experiment 2.** Percentage of correct recognition with different step size. Each row represents a different model. Each column represents a different step size. More details in the main text.

Model / # classes	5	10	20	31	50
Ours-Experiment2	85.2	74.8	78.5	77.4	80.4
Baseline	87.4				

### 5.2.5. Conclusions

The experimental results on TUM-GAID, presented above, lead to the following conclusions:

- Our end-to-end approach is useful applied to gait recognition using optical flow and videos. Therefore, we have demonstrated that it can be used in different situations besides image recognition.
- The selection of the training set is critical. When train and test sets contain similar conditions (experiment 1), our model and the baseline obtain better results than when the conditions are different (experiment 2). However, if we compare our approach with the baseline, in both cases, the performance is very similar, around 2% lower than baseline accuracy. Thus, the results between them are comparable even though our models are trained incrementally.

As future work, we plan to explore new selection algorithms to find better representatives of the each class. In addition, we also want to explore the option

of adding more weights in the intermediate layers of the model during the incremental steps. By this way, the model will have more weights to store knowledge.

# 6 Conclusions

---

This chapter presents the final ideas of this thesis. Section 6.1 summarizes the conclusions obtained from the previously described work and Section 6.2 describes the future lines of work that we plan to explore.

## 6.1. Conclusions

The gait recognition problem can be addressed using many different kind of techniques. Most of the approaches previously published in the literature are based on binary silhouettes or descriptors derived from them. However, we have proposed a set of solutions that use raw data or optical flow as input, instead of highly preprocessed images. This way, our approaches can disengage from the shape of the subject, which is the information represented by the silhouettes, and focus on better characteristics of the gait. Throughout this thesis, we have proposed different solutions to the gait recognition problem that, depending on the specific conditions of the setup, one can have better performance than the other.

The proposed hand-crafted approach and later improvements obtain state-of-the-art results in multi-view conditions and scenarios where the clothes of each subject can vary. Moreover, the little need for training data combined with its robustness facing changing conditions, allow this hand-crafted approach to be used in realistic environments. However, this approach needs high resolution (*i.e.*  $320 \times 240$  or bigger) to perform well and the execution time is slow, what difficulties a real time approximation.

To solve those shortcomings, we have proposed a deep learning approach which uses optical flow or raw data as input. This method can be used in real time with small resolution videos (*i.e.*  $60 \times 60$  is enough to obtain state-of-the-art results in many cases). Compared with other approaches, our model obtains the best results under similar resolutions but, without this constraint, it performs worse in some conditions. However, this problem is not a limitation of the method and it can be solved using a bigger training set to allow the CNN to learn a better representation.

To take advantage of datasets with multiple modalities or sources of information, we have proposed a set of different fusion schemes for the hand-crafted and deep learning approaches. Thus, the use of a fusion scheme improves the final accuracy of hand-crafted and deep learning approaches what indicates that the use of multiples inputs helps to obtain better models. Moreover, we have proposed a multi-task approach that uses additional information in the form of labels. This way, the multi-task approach uses multiple labels for the same input to produce multiple outputs. This approach also helps to the learning process as the description of a subject involves more information and it can be described in a better way.

Due to the importance of deep learning approaches in the last years, we have studied the energy consumption during training of a deep learning approach for gait recognition. The objective of this study is to help other researchers to save energy and money during the training process which, usually, may take weeks or even months. Thus, our study considers the energy a hyper-parameter that can be considered during the design of the architecture of the model. Therefore, depending on the problem, some rules can be applied to minimize the energy consumption without penalizing the accuracy of the model.

Finally, we have proposed an incremental approach that can be used in real scenarios where the number of classes can be incremented along the time. Thus, with our approach, instead of retraining the whole model from scratch using the full training set, we propose a shorter training step using a reduced number of training samples from the previously trained classes. By this way, the new model starts from the old one and the new classes are added while the previous knowledge is retained. Then, a company or institution only has to store a limited amount of data in order to retain the current behaviour of the network while the number of classes is incremented.

## 6.2. Future Work

As lines of study derived from this thesis, we plan to study the performance of our approach with multiple persons in the scene and the impact of occlusions, a new approximation for incremental learning, an implementation based on Long short-term memory (LSTM) [51] and an in depth study about energy consumption for deep learning approaches in embedded systems.

For the first line of study, there are no datasets with occlusions or multiple subjects in the scene. Therefore, we would have to build a new dataset including that conditions. To avoid recording people walking and to take advantage of the existing datasets, we plan to create a semi-synthetic dataset combining the real subjects from a previous dataset with synthetic information added to the original video. Thus, to include more subjects in the scene, we will combine the subjects from different videos into a single one. For the occlusions, the process is similar and we will add synthetic objects to the original video. By this way, we do not have to record a complete dataset and we could build a new dataset with endless combinations.

For the second line, we plan to study the effect of the number of parameters of the CNN during the incremental process. Intuitively, there should be a point during the incremental learning where the number of parameters is insufficient to learn new classes. Then, when this limit point is approached, the number of parameters should grow to store new knowledge. By this way, during the incremental steps, the CNN will grow when necessary and the incremental learning will perform better.

Finally, as third line of study, due to the cyclical nature of the gait, the use of recurrent networks and LSTMs is specially interesting. Thus, we want to study the performance of this kind of networks compared to the traditional CNNs.

Finally, as fourth line of study, energy consumption is also an important issue in embedded architectures. Thus, the design of applications in these platforms requires a trade-off between computing performance and consumption. We plan to develop a methodology that follows previous deal to deploy efficient deep learning application. It should take into account both the different hardware setups available in many heterogeneous architectures and several work scheduling strategies that could be applied.



# Apéndice A

## Resumen en español

---

El análisis del paso ha sido aplicado desde hace mucho tiempo en estudios médicos y biomecánicos. Hacia 1890, Braune y Fischer [37] fueron pioneros analizando el paso de humanos caminando con y sin carga encima, para optimizar el equipamiento de la infantería alemana. En 1970, Morrison [102] analizó el paso y parametrizó el movimiento de las articulaciones y extremidades involucradas en el mismo. Durante los últimos años, el análisis del paso se ha usado en entornos médicos, ya que es un método no invasivo que permite detectar enfermedades que afectan a la locomoción, como pueden ser el Parkinson [101], escoliosis [69] o artrosis [99]. Otro ámbito en el que se está utilizando el análisis del paso es el estudio del rendimiento en deportistas de élite para desarrollar nuevos equipamientos (zapatillas, bicicletas, etc) o mejorar la técnica del deportista.

Recientemente, el análisis del paso ha sido utilizado como patrón biométrico para identificar personas. De hecho, las personas son capaces de identificar otras personas a cierta distancia por su forma de caminar, sin necesidad de ver la cara del sujeto, lo que permite una identificación no invasiva y que no requiere de la colaboración del sujeto. Por contra, la mayoría de técnicas usadas hoy en día para identificar personas como pueden ser el reconocimiento facial, del iris o de la huella dactilar, requieren de la colaboración del sujeto a identificar, que debe mirar a una cámara o poner su dedo en un sensor. Sin embargo, la identificación mediante la forma de caminar no necesita de esta colaboración y puede usarse en entornos de video-vigilancia como aeropuertos.

En todos estos casos, y en cualquier aplicación de visión artificial, el proceso general seguido por los algoritmos consta de tres fases: adquisición de los datos, análisis de los datos y, por último, la toma de decisiones.

El proceso de adquisición de datos ha ido evolucionando a lo largo de los años con la llegada de nuevas tecnologías. Así, en el comienzo de los estudios del paso, los datos se tomaban a mano por los investigadores que observaban la escena y tomaban los apuntes que consideraban oportunos. Esto limitaba la cantidad y calidad de los datos obtenidos y de las posteriores conclusiones de los estudios. Con la aparición de los primeros sensores mecánicos, el proceso de adquisición de datos se hizo más fácil ya que, aunque estos sensores eran muy rudimentarios, ayudaban a discretizar el movimiento y facilitar su descripción. Finalmente, con la aparición de las computadoras, sensores electrónicos y cámaras digitales, el proceso de adquisición de datos se hizo mucho más asequible pudiendo tomar datos a frecuencias muy altas y con mucha precisión. Por lo tanto, las conclusiones extraídas de este tipo de datos podían ser mucho más sólidas que las que se podían realizar varias décadas atrás.

El proceso de análisis de los datos, como el proceso de adquisición, era manual hasta que aparecieron las computadoras. A medida que su capacidad de cómputo iba creciendo, los análisis que se podían realizar sobre los datos eran más complejos. Así, los análisis básicos como modelos estadísticos obtenidos a partir de cientos de datos, evolucionaron a métodos más complejos obtenidos a partir de miles o decenas de miles de datos. Los investigadores se dieron cuenta de que cuanto mayor cantidad de datos y modelos más complejos fueran usados, mejores resultados se obtenían. Esto conllevó a una carrera para obtener más datos para construir modelos cada vez más complejos. Hoy en día, enormes modelos entrenados con miles de millones de muestras obtienen resultados que parecerían imposibles años atrás.

Finalmente, el proceso de toma de decisiones, a diferencia de los dos anteriores, no ha cambiado sustancialmente. Aunque hay muchos más datos disponibles y los modelos utilizados son muy complejos, los objetivos principales que condujeron a realizar los análisis son los mismos. A grosso modo, hay tres tipos principales de decisiones: clasificación, regresión y detección, y, todas ellas, están presentes en problemas del siglo XIX y en problemas actuales. Sin embargo, es cierto que hoy en día nos enfrentamos a problemas más ambiciosos gracias a las computadoras, nuevos sensores y procesos de análisis. Así, lo que solía ser un gran problema como una segmentación o una clasificación, hoy en día el problema es el reconocimiento de acciones o la detección de eventos en vídeos.

La motivación de esta tesis parte de que hoy en día, los sistemas de seguridad y de control de acceso automáticos están ganando importancia y se usan cada vez más. La mayoría de la soluciones desarrolladas para resolver estos problemas se basan en la identificación biométrica usando un amplio abanico de características físicas de los sujetos como pueden ser el iris, la huella dactilar o la cara.



Sin embargo, todas estas técnicas tienen una serie de limitaciones, por un lado, requieren la colaboración por parte del sujeto a identificar (reconocimiento mediante la huella dactilar [86] o el iris [29]) o bien son muy sensibles a cambios en la forma (reconocimiento facial [156]). Sin embargo, el reconocimiento del paso es una forma no invasiva de implementar estos controles de seguridad y que no necesita de la colaboración del sujeto. Además, es robusto frente a cambios en la forma del sujeto ya que se centra en el movimiento.

Centrándonos en el problema de la identificación de personas a partir de su forma de caminar, la mayoría de trabajos publicados usan representaciones basadas en siluetas [47]. Este tipo de entrada tiene muchos problemas cuando la forma de los sujetos cambia, es decir, llevan distinta ropa o el punto de vista cambia. Por lo tanto, el gran interés que hay en el reconocimiento del paso combinado con la falta de buenos métodos disponibles en la literatura para resolver este problema, han hecho que nos planteemos las siguientes motivaciones para desarrollar este estudio:

- Desarrollar un método robusto frente a cambios en la forma de los sujetos a partir de diferentes tipos de información.
- Ser capaz de identificar las personas desde múltiples puntos de vista. En este caso, nuestro método debe ser capaz de identificar la persona sin conocer el punto de vista de antemano.
- Debido a la importancia que está tomando el *deep learning*, hemos decidido crear uno o más enfoques basado en técnicas de *deep learning* para compararlos con los métodos basados en características desarrolladas a mano.
- Aprovechar las diferentes fuentes de datos y la información sobre los sujetos que están disponibles hoy en día en muchas bases de datos y entornos reales.

A partir de las motivaciones anteriores, nos planteamos el objetivo principal de esta tesis, que es desarrollar un nuevo método para la identificación de personas a partir de la forma de caminar en entornos de múltiples vistas. Como entrada vamos a usar el flujo óptico que proporciona una información muy rica sobre el movimiento del sujeto mientras camina, lo que permite construir mejores representaciones y modelos. Además, como el flujo óptico se centra en el movimiento, nuestros métodos son más robustos frente a cambios en la forma de los sujetos. Como parte de este objetivo principal, definimos un conjunto de objetivos específicos englobados en el mismo:

1. Desarrollar un método basado en características desarrolladas a mano, específicamente, usando flujo óptico como entrada para construir una repre-

sentación de bajo nivel para, finalmente, construir una representación de alto nivel que es usada para entrenar un clasificador tradicional. Si hubiera múltiples tipos de información disponibles, todas esas informaciones de entradas serán usadas a la vez por el modelo.

2. Desarrollar un segundo método basado en *deep learning* construyendo arquitecturas para el problema de la identificación a partir de la forma de caminar. En este caso, nuestro plan es explorar las mejores arquitecturas disponibles en la literatura para otras tareas. Como en el objetivo anterior, si hubiera múltiples tipos de información disponibles, todas esas informaciones de entradas serán usadas a la vez por el modelo.
3. Desarrollar una metodología de aprendizaje incremental para añadir nuevas clases a un modelo previamente entrenado usando solamente un pequeño subconjunto de muestras de las clases antiguas. Esta metodología es útil en situaciones donde el conjunto de muestras de entrenamiento es demasiado grande para almacenarlo de forma indefinida o cuando el proceso de entrenamiento completo requiera de un tiempo excesivo de cómputo. En esas situaciones, cada paso incremental con nuevas clases requiere un pequeño conjunto de datos y mucho menos tiempo de entrenamiento que el modelo completo.
4. Realizar un estudio de consumo de energía en los modelos de *deep learning* para caracterizarlos desde un punto de vista no explorado anteriormente y que es muy importante desde un punto de vista monetario.

Para conseguir estos objetivos, se han seguido las siguientes fases de desarrollo:

1. Primero se ha revisado el estado del arte para seleccionar los mejores métodos y bases de datos usados para evaluar nuestro enfoque y compararnos con los mejores métodos. De acuerdo con los datos publicados, tamaño de las bases de datos y objetivos propuestos, se ha decidido utilizar las bases de datos AVAMVG [30], MoBo [45], CASIA-B y C [151] y TUM-GAID [52].
2. Se ha desarrollado un método basado en características desarrolladas a mano en el que se han seleccionado las representaciones de bajo y alto nivel para codificar el movimiento y la información sobre la forma de caminar. Finalmente, la representación del vídeo se introduce en un clasificador para obtener la identidad del sujeto. Como había múltiples tipos de datos de entrada en las bases de datos utilizadas, se exploraron diferentes tipos de fusión de información para mejorar los resultados de nuestro trabajo inicial.

Los métodos de fusión se han dividido en dos esquemas diferentes: fusión tardía, realizada a partir de las probabilidades obtenidas de los diferentes clasificadores y, fusión temprana, realizada a partir de las características obtenidas por el método antes de llegar al clasificador.

3. Se ha desarrollado en método de *deep learning*. Como arquitectura base se ha utilizado AlexNet [71] adaptada al reconocimiento del paso. Además, se ha comparado esta arquitectura base con otros dos tipos de arquitecturas muy utilizadas en la literatura: ResNet [49] y una red basada en convoluciones 3D [130]. Para realizar una comparación justa, los mismos experimentos se han ejecutado en todas las redes para encontrar la mejor para el problema de la identificación de sujetos a partir de su forma de caminar. Además, se han comparado los métodos basados en *deep learning* con el método desarrollado anteriormente basado en características desarrolladas a mano para ver si se cumple la tendencia de que los métodos de *deep learning* obtienen mejor precisión que los métodos tradicionales. Como en el punto anterior, se han evaluado diferentes formas de fusión para el enfoque basado en *deep learning*. En este caso, se han usado tres tipos de fusión tardía (a nivel de probabilidades obtenidas de los clasificadores). En el caso de la fusión temprana (a nivel de características), como la red construye sus propias características, la fusión temprana solo puede realizarse cuando se define la arquitectura de la red. En nuestro caso, las características de cada tipo de entrada se concatenan a diferentes niveles dentro de la red. En este paso, también decidimos explorar los beneficios de entrenar una red con una función de pérdida combinada para entrenar múltiples tareas a la vez.
4. Hemos estudiado el problema del reconocimiento del paso desde un punto de vista basado en el aprendizaje incremental. En este caso, se quería añadir más clases (*i.e.* sujetos) a un modelo entrenado previamente sin tener que re-entrenarlo desde cero. Para ello, se ha construido una función de pérdida combinada, una parte para retener el conocimiento previo y otra para aprender nueva información sobre las nuevas clases.
5. Finalmente, analizamos el consumo de energía durante el proceso de entrenamiento de diferentes tipos de arquitecturas CNN y el impacto de los hiper-parámetros en el consumo de energía y la precisión de los modelos. La energía consumida se a medido físicamente con un dispositivo externo conectado a las GPUs usadas durante el entrenamiento.

En esta lista de fases, la primera fase es necesaria para cumplir todos los objetivos puesto que las bases de datos se usan en todos ellos. Las fases 2 y 3

permiten cumplir los objetivos 1 y 2 respectivamente. Finalmente, las fases 4 y 5 permiten realizar los objetivos 3 y 4.

En cuanto a las contribuciones de esta tesis, hemos desarrollado diferentes enfoques usando diferentes técnicas para buscar la mejor solución posible. Por lo tanto, hemos intentado resolver el problema empezando desde diferentes puntos de vista aprovechando las últimas técnicas desarrolladas en otros ámbitos y la información incluida en las bases de datos utilizadas. Así, se han desarrollado enfoques basados en características diseñadas a mano y basados en *deep learning*. Además, hemos propuesto diferentes formas de fusión de información para aprovechar múltiples tipos de entrada. Finalmente, para hacer nuestros enfoques más atractivos para la industria, hemos desarrollado un enfoque incremental que es capaz de aprender nuevas clases incrementalmente y un estudio sobre el consumo de energía de los enfoques basados en *deep learning* durante su entrenamiento.

Por lo tanto, de forma más específica, las contribuciones de esta tesis son:

- Para el objetivo 1, en enfoque basado en características diseñadas a mano, las contribuciones son:
  1. Un nuevo descriptor de la forma de caminar (PFM) que combina el potencial de los descriptores HAR con la rica representación proporcionada por la codificación FV. Este nuevo descriptor usa el flujo óptico como entrada en lugar de las siluetas usadas por el resto de métodos disponibles.
  2. El descriptor propuesto es capaz de manejar múltiples puntos de vista durante el test, incluso si el modelo es entrenado con vistas diferentes a las testeadas.
  3. Un enfoque unificado que usa audio, información de profundidad e información visual para resolver el problema del reconocimiento a partir de la forma de caminar.
  4. Un enfoque multimodal para el reconocimiento de la edad y de los zapatos.
  5. Resultados que superan al estado del arte en las bases de datos AVAMVG, MoBo, CASIA-B, CASIA-C y TUM-GAID.
- Para el objetivo 2, el enfoque basado en *deep learning*, las contribuciones son:
  1. Una fase de pre-procesamiento para extraer, organizar y normalizar la entrada del flujo óptico.

2. Un conjunto de arquitecturas CNN que pueden usarse directamente como clasificadores o para extraer descriptores de la forma de caminar a partir del flujo óptico.
  3. El método propuesto es capaz de manejar múltiples puntos de vista.
  4. Un conjunto de arquitecturas CNN para la fusión de datos.
  5. Resultados que mejoran el estado del arte en las bases de datos TUM-GAID y CASIA-B.
- Para el objetivo 3, el enfoque incremental, las contribuciones son:
    1. Un enfoque *end-to-end* diseñado específicamente para el aprendizaje incremental. Este enfoque se puede aplicar a cualquier tipo de red profunda.
    2. Resultados que mejoran el estado del arte en CIFAR-100 e ImageNet.
  - Para el objetivo 4, el estudio de consumo, las contribuciones son:
    1. Un análisis del consumo de energía y el rendimiento en una configuración multi-GPU usando dos de las arquitecturas CNN más populares. Este estudio se centra en los pasos de *forward*, *backward* y actualización de pesos que se dan en el proceso de entrenamiento de una CNN.
    2. Estadísticas de precisión para descubrir las mejores configuraciones basadas en tiempo, consumo de energía y el producto energía-retraso.
    3. Comparación entre las arquitecturas Maxwell y Pascal.
  - Una implementación pública en MATLAB de todos los métodos desarrollados.

## A.1. Publicaciones

Durante esta tesis se han publicado un total de cuatro artículos en revistas internacionales indexadas en el *Journal of Citation Report* (JCR). Además, se han publicado cinco artículos en conferencias internacionales. Por lo tanto, un total de nueve artículos se han obtenido durante esta tesis.

A continuación se presenta la lista de artículos publicados:

- **Artículos de revista:**

- Marín-Jiménez, M. J., Castro, F. M., Carmona-Poyato, Á., Guil, N. (2015). On how to improve tracklet-based gait recognition systems. *Pattern Recognition Letters*, 68, 103-110.
- Castro, F. M., Marín-Jiménez, M. J., Guil, N. (2016). Multimodal features fusion for gait, gender and shoes recognition. *Machine Vision and Applications*, 27(8), 1213-1228.
- Castro, F. M., Marín-Jiménez, M. J., Mata, N. G., Muñoz-Salinas, R. (2017). Fisher motion descriptor for multiview gait recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 31(01), 1756002.
- Castro, F. M., Guil, N., Marín-Jiménez, M. J., Pérez-Serrano, J., Ujal-dón, M. (2018). Energy-based Tuning of Convolutional Neural Networks on Multi-GPUs. *Concurrency and Computation: Practice and Experience* (to appear).

■ **Artículos de conferencias internacionales:**

- Castro, F. M., Marín-Jiménez, M. J., Guil, N. (2015). Empirical study of audio-visual features fusion for gait recognition. In International Conference on Computer Analysis of Images and Patterns (CAIP) (pp. 727-739).
- Castro, F. M., Marín-Jiménez, M. J., Guil, N., López-Tapia, S., Pérez de la Blanca, N. (2017). Evaluation of CNN architectures for gait recognition based on optical flow maps. In International Conference of the Biometrics Special Interest Group, BIOSIG 2017, Darmstadt, Germany, September 20–22, 2017 (pp. 251-258).
- Marín-Jiménez, M. J., Castro, F. M., Guil, N., de la Torre, F., Medina-Carnicer, R. (2017). Deep multi-task learning for gait-based biometrics. In Image Processing (ICIP), 2017 IEEE International Conference on (pp. 106-110).
- Castro, F. M., Marín-Jiménez, M. J., Guil, N., de la Blanca, N. P. (2017). Automatic learning of gait signatures for people identification. In International Work-Conference on Artificial Neural Networks (IWANN) (pp. 257-270).
- Castro, F. M., Marín-Jiménez, M. J., Guil, N., Schmid, C., Ahahari, K. (2018). End-to-End Incremental Learning. European Conference on Computer Vision (ECCV), (to appear).

A continuación presentamos un resumen de los artículos de revista que avalan esta tesis además de los resúmenes de dos artículos de conferencia internacional que contienen técnicas no publicadas en revistas.

#### **A.1.1. Referencia [89] ‘On how to improve tracklet-based gait recognition systems’**

En [89] estudiamos diferentes propuestas para mejorar nuestro trabajo previo publicado en [22] basado en el reconocimiento del paso usando trayectorias densas. Una primera mejora es el uso de RootDCS en lugar del descriptor DCS original. De esta forma, este nuevo descriptor aplica una raíz cuadrada a los elementos del descriptor DCS original después de aplicarle una normalización L1. Esta regularización estabiliza la varianza entre valores del histograma de características cinemáticas, lo que mejora la capacidad de representación del descriptor. Otra forma de mejorar es usar trayectorias dispersas en lugar de densas porque no todas las trayectorias contribuye a describir la forma de caminar. Por lo tanto, la idea es descartar trayectorias cuyos vectores de velocidad media son similares a la velocidad media de la persona. De esta forma, se mantienen las trayectorias más relevantes (movimiento de brazos y piernas) y las superfluas se eliminan (movimiento horizontal del cuerpo). Puesto que los descriptores usados en [22] tienen una dimensionalidad muy grande, se aplicaba PCA para reducirla. Sin embargo, como estamos resolviendo un problema etiquetado, un enfoque semi-supervisado puede usarse para incrementar la separabilidad de los descriptores. Finalmente, como estamos usando un conjunto de SVMs en un enfoque uno frente a todos, las probabilidades de salida pueden ser muy similares en muchas ocasiones. Por lo tanto, proponemos usar un proceso de Rank Minimization cuyo objetivo es combinar las salidas de los diferentes clasificadores para obtener una salida final más robusta.

Una experimentación exhaustiva es realizada en tres bases de datos complejas para validar nuestras propuestas. En todos los casos, la mejora propuesta supera a los resultados obtenidos por el método original.

#### **A.1.2. Referencia [18] ‘Multimodal features fusion for gait, gender and shoes recognition’**

Puesto que múltiples fuentes de información o modalidades pueden ser grabadas a la vez mientras que una persona camina, nuestro objetivo es fusionar esas modalidades para mejorar la precisión del reconocimiento obtenida a par-

tir de una sola fuente de información. En este artículo, estudiamos seis formas diferentes de fusionar la información de diferentes modalidades, donde tres de ellas son fusiones tardías (a nivel de probabilidades) y las otras tres son fusiones tempranas (a nivel de descriptores).

Con respecto a las fusiones tempranas, proponemos tres versiones diferentes: concatenación de vectores, donde los vectores de características se combinan para producir uno más grande; diccionarios bi-modales donde un diccionario común aprende a producir un vector final de características mezclando información proveniente de todas las modalidades; y, por último, el aprendizaje de kernel múltiple donde un proceso de optimización obtiene un kernel óptimo que representa la información de todas las modalidades en un espacio de características mayor donde las clases son más separables.

Con respecto a las fusiones tardías, proponemos otras tres opciones: probabilidades pesadas, donde una suma pesada se aplica a las probabilidades de salida de cada modalidad individual; SVM sobre las probabilidades, donde se entrena un SVM con las probabilidades de cada modalidad individual para producir una probabilidad combinada; y, finalmente, la minimización de ranking, donde un proceso de minimización obtiene la mejor combinación de probabilidades.

De acuerdo con los resultados experimentales, la fusión de modalidades ayuda a mejorar la precisión del sistema siempre que las características fusionadas sean representativas por sí mismas. Esto es, si una modalidad obtiene resultados muy pobres, la fusión de esa modalidad con otra también producirá malos resultados.

### **A.1.3. Referencia [21] ‘Fisher motion descriptor for multi-view gait recognition’**

En este trabajo, extendemos el artículo de conferencia anteriormente publicado [22]. Esta versión amplía la sección experimental a cuatro bases de datos con múltiples vistas y condiciones de paso como llevando mochilas, diferente ropa, fundas en los zapatos, cintas de correr, etc. Por lo tanto, el objetivo de este artículo es explorar las capacidades del método anteriormente publicado que, debido a limitaciones de espacio, fueron imposibles de comprobar. Además, se han realizado algunos cambios para mejorar la precisión del proceso global.

La experimentación exhaustiva ha confirmado que nuestro enfoque es capaz de manejar cambios en la forma de los sujetos y múltiples vistas, obteniendo resultados en el estado del arte para todas las bases de datos.



#### A.1.4. Referencia [17] ‘Energy-based Tuning of Convolutional Neural Networks on Multi-GPUs’

En este estudio, analizamos la energía consumida durante el proceso de entrenamiento de una CNN. Puesto que este es un proceso costoso y largo que necesita muchos recursos, es interesante considerar la energía como un parámetro que puede ser minimizado como si fuera una función de pérdida. De esta forma, además de la precisión final, la energía también se tiene en cuenta en el proceso de diseño de la arquitectura de una CNN porque la energía consumida tiene un impacto económico y una red que consuma menos energía puede suponer un gran ahorro económico.

Puesto que se pueden diseñar infinitas arquitecturas, nosotros nos centramos en las siguientes: una versión de AlexNet [72] adaptada al problema del reconocimiento del paso y la versión original de AlexNet para el reconocimiento de imágenes, una red ResNet [48] para el reconocimiento de imágenes en ImageNet y una versión adaptada al reconocimiento del paso. De esta forma, hemos estudiado dos tipos de redes diferentes usadas como base en la mayoría de métodos actuales, y dos de los tipos de datos de entrada (vídeo e imágenes) más usados en la actualidad. Además, también analizamos el impacto del tamaño del batch en la precisión y el consumo de energía. Finalmente, también consideramos el impacto de realizar el entrenamiento en un entorno con múltiples GPUs y el efecto en el consumo de diferentes arquitecturas GPU (Pascal y Maxwell en nuestro caso).

De acuerdo con las conclusiones extraídas, en bases de datos pequeñas, la opción de optimizar el consumo de energía y la precisión de forma conjunta es muy compleja ya que se requieren tamaños pequeños de batch para obtener un buen entrenamiento, lo que penaliza el consumo de energía. Sin embargo, hay una excepción en las redes basadas en AlexNet, ya que admiten tamaños mayores de batch permitiendo minimizar la energía manteniendo una precisión buena. En bases de datos grandes, se pueden usar tamaños de batch grande sin que la precisión baje lo que permite optimizar energía y precisión conjuntamente. En cuanto a las arquitecturas de GPU, Pascal obtiene mejores consumo y frecuencias más altas como se esperaba. Por último, las configuraciones con múltiples GPUs arrojan buenos resultados con dos dispositivos. Con más de dos dispositivos, el problema tiene que ser estudiado individualmente.

### A.1.5. Referencia [88] ‘Deep multi-task learning for gait-based biometrics’

En este trabajo exploramos el uso de la información del paso para diferentes tareas, no solo para la identificación de personas. Aunque la forma de andar se usa principalmente para la identificación de personas, otras tareas biométricas pueden ser definidas en base a la forma de andar, como por ejemplo, el reconocimiento del género de una persona o la estimación de su edad. Sin embargo, no se ha prestado mucha atención al hecho de que esas tareas están muy relacionadas y se pueden beneficiar unas de las otras cuando son consideradas conjuntamente durante el entrenamiento.

Nosotros proponemos un proceso de aprendizaje multi-tarea basado en una función de pérdida combinada para entrenar CNNs a partir del flujo óptico. De esta forma, las características de la forma de andar obtenidas por la red, son útiles para todas las tareas entrenadas.

De acuerdo con los resultados, el proceso de aprendizaje multi-tarea acelera la convergencia de la red, mejora la capacidad de generalización de las características o descriptores obtenidos por la red para tareas biométricas y, es capaz de mejorar los resultados del estado del arte.

### A.1.6. Referencia [20] ‘End-to-End Incremental Learning’

Aunque los enfoques basados en *deep learning* obtienen los mejores resultados del estado del arte cuando son entrenados con bases de datos enormes o, al menos, usando la base de datos de forma completa, tienen problemas cuando son entrenados con nuevas clases de forma incremental. Este problema es conocido como *catastrophic forgetting*, un gran empeoramiento del rendimiento del modelo cuando se le añaden nuevas clases de forma incremental.

Nosotros proponemos un enfoque *end-to-end* que aprende redes de forma incremental, usando nueva información de las nuevas clases y solamente un pequeño conjunto de datos de las clases antiguas. El proceso de entrenamiento se basa en una función de pérdida compuesta de una medida de destilación para retener el conocimiento de las clases antiguas y, de una función de pérdida basada en la entropía cruzada para entrenar las nuevas clases.

De acuerdo con los resultados, nuestro enfoque obtiene los mejores resultados del estado del arte en dos bases de datos (CIFAR-100 e ImageNet). Además, es robusto frente a bases de datos compuestas de clases similares, permitiendo que

el método pueda ser utilizado en problemas como el reconocimiento de caras.

## A.2. Conclusiones

El problema de la identificación de personas a partir de su forma de caminar puede resolverse usando muchas técnicas diferentes. La mayoría de los enfoques publicados anteriormente en la literatura usan siluetas o descriptores derivados a partir de ellas. Sin embargo, nosotros hemos propuesto un conjunto de soluciones que usan como entrada los datos en bruto o el flujo óptico, en lugar de imágenes pre-procesadas. De esta forma, nuestros métodos pueden abstraerse de la forma del sujeto, que es la información representada por las siluetas, y centrarse en mejores características de la forma de caminar. A lo largo de esta tesis, hemos propuesto diferentes soluciones al problema del reconocimiento del paso para obtener los mejores resultados posibles dependiendo de la configuración del entorno.

El método propuesto basado en características desarrolladas a mano y las mejoras presentadas posteriormente obtienen resultados que mejoran el estado del arte en entornos con múltiples vistas y escenarios donde la ropa que llevan los sujetos puede variar. Además, la poca necesidad de datos de entrenamiento junto con su robustez frente a condiciones cambiantes, permiten que este método pueda usarse en entornos realistas. Sin embargo, este enfoque necesita una resolución relativamente grande ( $320 \times 240$  o mayor) para tener un buen rendimiento y su tiempo de ejecución es bastante lento, dificultando su aplicación en una aproximación en tiempo real.

Para resolver esos problemas, hemos propuesto un enfoque basado en *deep learning* que usa como entrada el flujo óptico o datos en bruto. Este método puede usarse en tiempo real con vídeos a baja resolución (en muchos casos,  $60 \times 60$  es suficiente para obtener resultados en el estado del arte). Comparado con otros métodos, nuestro modelo obtiene los mejores resultados usando resoluciones de entrada similares pero, sin esta limitación, obtiene peores resultados en algunos casos. Sin embargo, este problema no es una limitación del método y puede resolverse usando un conjunto de entrenamiento mayor para permitir que la CNN obtenga una mejor representación.

Para aprovechar las bases de datos con múltiples modalidades o fuentes de información, hemos propuesto un conjunto de diferentes esquemas de fusión para los enfoques basados en características desarrolladas a mano y en *deep learning*. De esta forma, el uso de un esquema de fusión mejora la precisión final de am-

bos métodos (basado en características desarrolladas a mano y basado en *deep learning*), lo que indica que el uso de múltiples entradas ayuda a obtener mejores modelos. Además, hemos propuesto un enfoque multi-tarea que usa información adicional en forma de etiquetas. Así, el método multi-tarea usa múltiples etiquetas para una misma entrada común para producir múltiples salidas. Este enfoque también ayuda al proceso de aprendizaje puesto que el sujeto se describe con más información.

Debido a la importancia de los métodos basados en *deep learning* en los últimos años, hemos estudiado el consumo de energía durante el entrenamiento de diferentes enfoques basados en *deep learning* aplicados al reconocimiento del paso. El objetivo de este estudio es ayudar a otros investigadores a ahorrar energía y dinero durante el proceso de entrenamiento que, normalmente, puede durar semanas o incluso meses. Por lo tanto, nuestro estudio considera la energía consumida como un hiper-parámetro que debe tenerse en cuenta durante el diseño de la arquitectura de la red. De esta forma, dependiendo del problema, algunas reglas pueden ser aplicadas para minimizar el consumo de energía sin penalizar la precisión del modelo.

Finalmente, hemos propuesto un enfoque incremental que puede usarse en entornos reales donde el número de clases puede crecer a lo largo del tiempo. De esta forma, con nuestro enfoque, en lugar de volver a entrenar el modelo completo desde cero usando el conjunto completo de entrenamiento, nosotros proponemos realizar un entrenamiento más corto usando un número reducido de muestras de entrenamiento de las clases ya entrenadas. Así, el nuevo modelo parte del modelo anterior al que se le añaden las nuevas clases mientras el proceso de entrenamiento trata de mantener el conocimiento anterior. Por lo tanto, una compañía o institución solo tiene que almacenar un conjunto limitado de datos de entrenamiento para mantener el conocimiento actual mientras que se añaden nuevas clases al sistema.

### A.3. Trabajo Futuro

Como líneas de estudio derivadas de esta tesis, planeamos estudiar el rendimiento de nuestro enfoque en escenas donde haya múltiples personas a la vez junto con el impacto de las oclusiones de los sujetos, una nueva aproximación para el aprendizaje incremental, una implementación basada en LSTM [51] y un estudio en profundidad sobre el consumo de energía de aplicaciones *deep learning* en sistemas embebidos.

Para la primera línea de estudio, actualmente no existen bases de datos con oclusiones o múltiples personas en la escena. Por lo tanto, tendríamos que construir la base de datos de cero incluyendo esas condiciones. Para evitar tener que grabar a personas andando y aprovechar las bases de datos existentes, nuestra intención es crear una base de datos semi-sintética combinando los sujetos reales de una base de datos con información sintética añadida al vídeo original. De esta forma, para incluir más sujetos en una escena, tendremos que combinar los sujetos de diferentes vídeos en un vídeo común. Para las oclusiones, el proceso es similar y tendremos que añadir objetos sintéticos al vídeo original. Así, no tenemos que grabar una base de datos completa desde cero y podemos construir una base de datos con combinaciones infinitas.

Para la segunda línea de estudio, queremos estudiar el efecto del número de parámetros de la CNN durante el entrenamiento incremental. Intuitivamente, debería haber un punto durante el aprendizaje incremental donde el número de parámetros es insuficiente para aprender nuevas clases. Entonces, cuando este límite es alcanzado, el número de parámetros debería crecer para almacenar más conocimiento. De esta forma, durante los pasos incrementales, la CNN crecerá cuando sea necesario y el proceso incremental tendrá mejor rendimiento.

Para la tercera línea de estudio, debido a la naturaleza cíclica del paso, el uso de redes recurrentes y LSTMs es muy interesante. De esta forma, queremos estudiar el rendimiento de este tipo de redes comparadas con las CNNs tradicionales.

Finalmente, como cuarta línea de estudio, el consumo de energía es un problema importante en arquitecturas embebidas. Por lo tanto, el diseño de aplicaciones en ese tipo de plataformas requiere de un compromiso entre el rendimiento computacional y el consumo de energía. Queremos desarrollar una metodología que permita desplegar aplicaciones eficientes de *deep learning*.



# Bibliography

- [1] Deep Neural Network Energy Estimation Tool. <https://energyestimation.mit.edu>.
- [2] Lady Ada. Adafruit INA219 Current Sensor Breakout. <https://learn.adafruit.com/adafruit-ina219-current-sensor-breakout/>.
- [3] Naoki Akae, Al Mansur, Yasushi Makihara, and Yasushi Yagi. Video from nearly still: an application to low frame-rate gait recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1537–1543, 2012.
- [4] P. Alonso, R.M. Badía, J. Labarta, M. Barreda, M.F. Dolz, R. Mayo, E.S. Quintana-Ortí, and R. Reyes. Tools for power-energy modelling and analysis of parallel scientific applications. In *Proceedings 41st Intl. Conference on Parallel Processing (ICPP'12)*, pages 420–429. IEEE Computer Society, September 2012.
- [5] M. Alotaibi and A. Mahmood. Improved gait recognition based on specialized deep convolutional neural networks. In *IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pages 1–7, Oct 2015.
- [6] R. Andri, L. Cavigelli, D. Rossi, and L. Benini. Yodann: An ultra-low power convolutional neural network accelerator based on binary weights. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 236–241, July 2016.
- [7] Bernard Ans, Stéphane Rousset, Robert M French, and Serban Musca. Self-refreshing memory in artificial neural networks: Learning temporal sequences without catastrophic forgetting. *Connection Science*, 16(2):71–99, 2004.

- [8] Pradeep K. Atrey, M. Anwar Hossain, Abdulmotaleb El Saddik, and Mohan S. Kankanhalli. Multimodal fusion for multimedia analysis: a survey. *Multimedia Systems*, 16(6):345–379, 2010.
- [9] Olivier Barnich and Marc Van Droogenbroeck. Frontal-view gait recognition by intra- and inter-frame rectangle size distribution. *Pattern Recognition Letters*, 30(10):893 – 901, 2009.
- [10] BeagleBone. Beaglebone black. <http://beagleboard.org/BLACK>.
- [11] Vasileios Belagiannis, Christian Rupprecht, Gustavo Carneiro, and Nassir Navab. Robust optimization for deep regression. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2830–2838, 2015.
- [12] S. Benedict. Energy-aware performance analysis methodologies for hpc architectures — an exploratory study. *Journal of Network and Computer Applications*, 35(6):1709–1719, 2012.
- [13] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *PAMI*, 35(8):1798–1828, 2013.
- [14] M. Bettoni, G. Urgese, Y. Kobayashi, E. Macii, and A. Acquaviva. A convolutional neural network fully implemented on fpga for embedded platforms. In *2017 New Generation of CAS (NGCAS)*, pages 49–52, Sept 2017.
- [15] G. Bradski. OpenCV library. *Dr. Dobb’s Journal of Software Tools*, 2000.
- [16] Ermao Cai, Da-Cheng Juan, Dimitrios Stamoulis, and Diana Marculescu. Neuralpower: Predict and deploy energy-efficient convolutional neural networks. *CoRR*, abs/1710.05420, 2017.
- [17] Francisco M Castro, Nicolás Guil, Manuel J Marín-Jiménez, Jesús Pérez-Serrano, and Manuel Ujaldón. Energy-based tuning of convolutional neural networks on multi-gpus. *Concurrency and Computation: Practice and Experience (to appear)*, 2018.
- [18] Francisco M. Castro, Manuel J. Marín-Jiménez, and Nicolás Guil. Multimodal features fusion for gait, gender and shoes recognition. *Machine Vision and Applications*, 27(8):1213–1228, Nov 2016.
- [19] Francisco M. Castro, Manuel J. Marín-Jiménez, Nicolás Guil, Santiago López-Tapia, and Nicolás Pérez de la Blanca. Evaluation of cnn architectures for gait recognition based on optical flow maps. In *BIOSIG*, pages 251–258, 2017.



- [20] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *European Conference on Computer Vision (ECCV)*, (to appear), 2018.
- [21] Francisco M. Castro, M.J. Marín-Jiménez, N. Guil Mata, and R. Muñoz Salinas. Fisher motion descriptor for multiview gait recognition. *International Journal of Patt. Recogn. in Artificial Intelligence*, 31(1), 2017.
- [22] Francisco M. Castro, M.J. Marín-Jiménez, and Rafael Medina-Carnicer. Pyramidal Fisher Motion for multiview gait recognition. In *Proceedings of the International Conference on Pattern Recognition*, pages 1692–1697, 2014.
- [23] Francisco Manuel Castro, Manuel J. Marín-Jiménez, Nicolás Guil, and Nicolás Pérez de la Blanca. Automatic learning of gait signatures for people identification. *Advances in Computational Intelligence: 14th International Work-Conference on Artificial Neural Networks (IWANN)*, pages 257–270, 2017.
- [24] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In *NIPS*, 2000.
- [25] Yanmei Chai, Jie Ren, Huimin Zhao, Yang Li, Jinchang Ren, and Paul Murray. Hierarchical and multi-featured fusion for effective gait recognition under variable scenarios. *Pattern Analysis and Applications*, pages 1–13, 2015.
- [26] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. NEIL: Extracting visual knowledge from web data. In *ICCV*, 2013.
- [27] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
- [28] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.
- [29] John Daugman. How iris recognition works. In *The essential guide to image processing*, pages 715–739. Elsevier, 2009.
- [30] David López-Fernández, Francisco J. Madrid-Cuevas, Ángel Carmona-Poyato, Manuel J. Marín-Jiménez and Rafael Muñoz-Salinas. The AVA Multi-View Dataset for Gait Recognition. In Pier Luigi Mazzeo, Paolo

- Spagnolo, and Thomas B. Moeslund, editors, *Activity Monitoring by Multiple Distributed Sensing*, Lecture Notes in Computer Science, pages 26–39. Springer International Publishing, 2014.
- [31] Santosh Divvala, Ali Farhadi, and Carlos Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *CVPR*, 2014.
- [32] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [33] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.
- [34] Sheila A Dugan and Krishna P Bhat. Biomechanics and analysis of running gait. *Physical Medicine and Rehabilitation Clinics*, 16(3):603–621, 2005.
- [35] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller, and Wolfram Burgard. Multimodal deep learning for robust RGB-D object recognition. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, pages 681–687. IEEE, 2015.
- [36] Gunnar Farneback. Two-frame motion estimation based on polynomial expansion. In *Proc. of Scandinavian Conf. on Image Analysis*, volume 2749, pages 363–370, 2003.
- [37] Otto Fischer and Braune Wilhelm. *Der Gang des Menschen: Versuche am unbelasteten und belasteten Menschen I - II*. Hirzel Verlag, 1895.
- [38] Robert M French. Dynamically constraining connectionist networks to produce distributed, orthogonal representations to reduce catastrophic interference. In *Cognitive Science Society Conf.*, 1994.
- [39] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2-3):131–163, 1997.
- [40] Bence Gálai and Csaba Benedek. Feature selection for lidar-based gait recognition. In *Computational Intelligence for Multimedia Understanding (IWCIM), 2015 International Workshop on*, pages 1–5, 2015.

- [41] Wenjuan Gong, Michael Sapienza, and Fabio Cuzzolin. Fisher tensor decomposition for unconstrained gait recognition. In *Proc. of Tensor Methods for Machine Learning, Workshop of the European Conference of Machine Learning*, 2013.
- [42] J.D. González-Rincón. Sistema basado en open source hardware para la monitorización del consumo de un computador. *Master Thesis Project. Universidad Complutense de Madrid*, 2015.
- [43] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [44] I.J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *ArXiv e-prints, arXiv 1312.6211*, 2013.
- [45] Ralph Gross and Jianbo Shi. The CMU Motion of Body (MoBo) Database. Technical Report CMU-RI-TR-01-18, Robotics Institute, June 2001.
- [46] Yu Guan and Chang-Tsun Li. A robust speed-invariant gait recognition system for walker and runner identification. In *Intl. Conf. on Biometrics (ICB)*, pages 1–8, 2013.
- [47] Ju Han and Bir Bhanu. Individual recognition using gait energy image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(2):316–322, 2006.
- [48] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.
- [49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [50] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NIPS workshop*, 2014.
- [51] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [52] Martin Hofmann, Jürgen Geiger, Sebastian Bachmann, Björn Schuller, and Gerhard Rigoll. The TUM Gait from Audio, Image and Depth (GAID) database: Multimodal recognition of subjects and traits. *Journal of Visual Communication and Image Representation*, 25(1):195 – 206, 2014.

- [53] Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 Technical Briefs*, page 18, 2015.
- [54] Wouter Hoogkamer, Shalaya Kipp, Jesse H. Frank, Emily M. Farina, Geng Luo, and Rodger Kram. A comparison of the energetic cost of running in marathon racing shoes. *Sports Medicine*, 48(4):1009–1019, Apr 2018.
- [55] Emdad Hossain and Girija Chetty. Multimodal feature learning for gait biometric based human identity recognition. In *Neural Information Processing*, pages 721–728, 2013.
- [56] H. Hu. Multiview gait recognition based on patch distribution features and uncorrelated multilinear sparse local discriminant canonical correlation analysis. *Circuits and Systems for Video Technology, IEEE Transactions on*, 24(4):617–630, April 2014.
- [57] Haifeng Hu. Enhanced gabor feature based classification using a regularized locally tensor discriminant model for multiview gait recognition. *Circuits and Systems for Video Technology, IEEE Transactions on*, 23(7):1274–1286, July 2013.
- [58] Maodi Hu, Yunhong Wang, Zhaoxiang Zhang, De Zhang, and J.J. Little. Incremental learning for video-based gait recognition with LBP flow. *Cybernetics, IEEE Transactions on*, 43(1):77–89, Feb 2013.
- [59] Weiming Hu, Tieniu Tan, Liang Wang, and Steve Maybank. A survey on visual surveillance of object motion and behaviors. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(3):334–352, 2004.
- [60] F.D. Igual, L.M. Jara, J.I. Gómez, L. Piñuel, and M. Prieto. A Power Measurement Environment for PCIe Accelerators. *Computer Science - Research and Development*, 30(2):115–124, May 2015.
- [61] Yumi Iwashita, Koichi Ogawara, and Ryo Kurazume. Identification of people walking along curved trajectories. *Pattern Recognition Letters*, 48(0):60–69, 2014.
- [62] Mihir Jain, Herve Jegou, and Patrick Bouthemy. Better exploiting motion for better action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2555–2562, 2013.

- [63] N. Jouppi. Google supercharges machine learning tasks with TPU custom chip. <https://cloudplatform.googleblog.com/2016/05/Google-supercharges-machine-learning-tasks-with-custom-chip.html>.
- [64] Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetting learning in deep neural networks. *ArXiv e-prints, arXiv 1607.00122*, 2016.
- [65] Pakorn KaewTraKulPong and Richard Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Video-based surveillance systems*, pages 135–144. Springer, 2002.
- [66] Andrej Karpathy, George Toderici, Sachin Shetty, Tommy Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732. IEEE, 2014.
- [67] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [68] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proc. National Academy of Sciences*, 114(13):3521–3526, 2017.
- [69] Inès A. Kramers-de Quervain, Roland Müller, A. Stacoff, Dieter Grob, and Edgar Stüssi. Gait analysis in patients with idiopathic scoliosis. *European Spine Journal*, 13(5):449–456, Aug 2004.
- [70] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.
- [71] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
- [72] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
- [73] Z. Lai, Y. Xu, Z. Jin, and D. Zhang. Human gait recognition via sparse discriminant projection learning. *Circuits and Systems for Video Technology, IEEE Transactions on*, 24(10):1651–1662, 2014.

- [74] Toby HW Lam and Raymond ST Lee. A new representation for human gait recognition: Motion silhouettes image (msi). In *International Conference on Biometrics*, pages 612–618. Springer, 2006.
- [75] Quoc V Le, Will Y Zou, Serena Y Yeung, and Andrew Y Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3361–3368. IEEE, 2011.
- [76] Heesung Lee, Sungjun Hong, Imran Fareed Nizami, and Euntai Kim. A noise robust gait representation: Motion energy image. *International Journal of Control, Automation and Systems*, 7(4):638–643, 2009.
- [77] Zhizhong Li and Derek Hoiem. Learning without forgetting. *PAMI*, 2018.
- [78] Y. Lin, C. Sakr, Y. Kim, and N. Shanbhag. Predictivenet: An energy-efficient convolutional neural network via zero prediction. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, May 2017.
- [79] Zhouchen Lin, Minming Chen, and Yi Ma. The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.
- [80] Yushu Liu, Junping Zhang, Chen Wang, and Liang Wang. Multiple HOG templates for gait recognition. In *Proceedings of the International Conference on Pattern Recognition*, pages 2930–2933. IEEE, 2012.
- [81] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982.
- [82] David Lopez-Paz and Marc’Aurelio Ranzato. Gradient episodic memory for continual learning. In *NIPS*, 2017.
- [83] R Beau Lotto, S Mark Williams, and Dale Purves. Mach bands as empirically derived associations. *Proceedings of the National Academy of Sciences*, 96(9):5245–5250, 1999.
- [84] D. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 1150–1157, September 1999.
- [85] N. Lynnerup and P. K. Larsen. Gait as evidence. *IET Biometrics*, 3(2):47–54, June 2014.

- [86] Davide Maltoni, Dario Maio, Anil K Jain, and Salil Prabhakar. *Handbook of fingerprint recognition*. Springer Science & Business Media, 2009.
- [87] Elman Mansimov, Nitish Srivastava, and Ruslan Salakhutdinov. Initialization strategies of spatio-temporal convolutional neural networks. *CoRR*, abs/1503.07274, 2015.
- [88] M. J. Marín-Jiménez, F. M. Castro, N. Guil, F. de la Torre, and R. Medina-Carnicer. Deep multitask learning for gait-based biometrics. In *Proceedings of the IEEE International Conference on Image Processing*, 2017.
- [89] Manuel J. Marín-Jiménez, Francisco M. Castro, Ángel Carmona-Poyato, and Nicolás Guil. On how to improve tracklet-based gait recognition systems. *Pattern Recognition Letters*, 68, Part 1:103 – 110, 2015.
- [90] R. Martin-Felez, J. Ortells, and R.A. Mollineda. Exploring the effects of video length on gait recognition. In *Proceedings of the International Conference on Pattern Recognition*, pages 3411–3414, 2012.
- [91] Raúl Martín-Félez and Tao Xiang. Gait recognition by ranking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 328–341, 2012.
- [92] Raúl Martín-Félez and Tao Xiang. Uncooperative gait recognition by learning to rank. *Pattern Recognition*, 47(12):3793 – 3806, 2014.
- [93] M. Mathew, K. Desappan, Kumar-Swami P., S. Nagori, and B Moothedath. Sparse, quantized, full frame cnn for low power embedded devices. In *Proceedings Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [94] M. Mathew, K. Desappan, Kumar-Swami P., Nagori S., and B Moothedath. Embedded low-power deep learning with tidl. *Texas Instruments Technical Report*, 2018.
- [95] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109 – 165, 1989.
- [96] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.
- [97] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *PAMI*, 35(11):2624–2637, 2013.

- [98] Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapa Nakashole, Emmanouil Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard Wang, Derry Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. Never-ending learning. In *AAAI*, 2015.
- [99] Michael A Mont, Thorsten M Seyler, Phillip S Ragland, Roland Starr, Jochen Erhart, and Anil Bhawe. Gait analysis of patients with resurfacing hip arthroplasty compared with hip osteoarthritis and standard total hip arthroplasty. *The Journal of arthroplasty*, 22(1):100–108, 2007.
- [100] B. Moons, B. De Brabandere, L. Van Gool, and M. Verhelst. Energy-efficient convnets through approximate computing. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8, March 2016.
- [101] Meg E. Morris, Robert Iansek, Thomas A. Matyas, and Jeffery J. Summers. The pathogenesis of gait hypokinesia in parkinson’s disease. *Brain*, 117(5):1169–1181, 1994.
- [102] J.B. Morrison. Bioengineering analysis of force actions transmitted by the knee joint. In *Biomedical Engineering*, pages 164–170, 1968.
- [103] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.
- [104] Y. Nesterov. A method for solving the convex programming problem with convergence rate  $o(1/\sqrt{k})$ . *Soviet Mathematics Doklady*, 27:372–376, 1983.
- [105] Natalia Neverova, Christian Wolf, Griffin Lacey, Lex Fridman, Deepak Chandra, Brandon Barbello, and Graham Taylor. Learning human identity from motion patterns. *IEEE Access*, 4:1810–1820, 2016.
- [106] Thanh Trung Ngo, Yasushi Makihara, Hajime Nagahara, Yasuhiro Mukaigawa, and Yasushi Yagi. The largest inertial sensor-based gait database and performance evaluation of gait-based personal authentication. *Pattern Recognition*, 47(1):228 – 237, 2014.
- [107] Trung Thanh Ngo, Yasushi Makihara, Hajime Nagahara, Yasuhiro Mukaigawa, and Yasushi Yagi. Similar gait action recognition using an inertial sensor. *Pattern Recognition*, 48(4):1289 – 1301, 2015.



- [108] Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2007.
- [109] Florent Perronnin and Diane Larlus. Fisher vectors meet neural networks: A hybrid classification architecture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3743–3752, 2015.
- [110] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 143–156, 2010.
- [111] Janez Perš, Vildana Sulić, Matej Kristan, Matej Perše, Klemen Polanec, and Stanislav Kovačič. Histograms of optical flow for efficient representation of body motion. *Pattern Recognition Letters*, 31(11):1369–1376, 2010.
- [112] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.
- [113] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.
- [114] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990.
- [115] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR*, 2017.
- [116] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.
- [117] M. Ristin, M. Guillaumin, J. Gall, and L. V. Gool. Incremental learning of ncm forests for large-scale image classification. In *CVPR*, 2014.
- [118] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.
- [119] S. Ruping. Incremental learning with support vector machines. In *ICDM*, 2001.

- [120] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *ArXiv e-prints, arXiv 1606.04671*, 2016.
- [121] P. Ruvolo and E. Eaton. ELLA: An efficient lifelong learning algorithm. In *ICML*, 2013.
- [122] K. Shmelkov, C. Schmid, and K. Alahari. Incremental learning of object detectors without catastrophic forgetting. In *ICCV*, 2017.
- [123] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014.
- [124] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2018.
- [125] Sabesan Sivapalan, Daniel Chen, Simon Denman, Sridha Sridharan, and Clinton Fookes. Gait energy volumes and frontal gait recognition using depth images. In *Biometrics (IJCB), 2011 International Joint Conference on*, pages 1–6. IEEE, 2011.
- [126] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 2, pages 1470–1477, October 2003.
- [127] Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation by joint identification-verification. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, pages 1988–1996. Curran Associates, Inc., 2014.
- [128] Alexander V. Terekhov, Guglielmo Montone, and J. Kevin O'Regan. Knowledge transfer in deep block-modular neural networks. In *Biomimetic and Biohybrid Systems*, pages 268–279, 2015.
- [129] Sebastian Thrun. Lifelong learning algorithms. In *Learning to Learn*, pages 181–209, 1998.
- [130] Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the International Conference on Computer Vision (ICCV)*. IEEE, 2015.

- [131] Amal Rannen Triki, Rahaf Aljundi, Matthew B. Blaschko, and Tinne Tuytelaars. Encoder based lifelong learning. In *ICCV*, 2017.
- [132] Manik Varma and Bodla Rakesh Babu. More generality in efficient multiple kernel learning. In *ICML*, page 134, 2009.
- [133] A. Vedaldi and K. Lenc. MatConvNet – Convolutional Neural Networks for MATLAB. In *Proceeding of the ACM Int. Conf. on Multimedia*, 2015.
- [134] Anran Wang, Jiwen Lu, Jianfei Cai, Tat-Jen Cham, and Gang Wang. Large-margin multi-modal deep learning for RGB-D object recognition. *Multimedia, IEEE Transactions on*, 17(11):1887–1898, Nov 2015.
- [135] Chen Wang, Junping Zhang, Liang Wang, Jian Pu, and Xiaoru Yuan. Human identification using temporal information preserving gait template. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2164–2176, 2012.
- [136] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action Recognition by Dense Trajectories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3169–3176, June 2011.
- [137] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4305–4314, 2015.
- [138] Y. Wang, L. Xia, T. Tang, B. Li, S. Yao, M. Cheng, and H. Yang. Low power convolutional neural networks on a chip. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 129–132, May 2016.
- [139] Max Welling. Herding dynamical weights to learn. In *ICML*, 2009.
- [140] Tenika Whytock, Alexander Belyaev, and NeilM. Robertson. Dynamic distance-based shape features for gait recognition. *Journal of Mathematical Imaging and Vision*, 50(3):314–326, 2014.
- [141] T. Wolf, M. Babaei, and G. Rigoll. Multi-view gait recognition using 3D convolutional neural networks. In *Proceedings of the IEEE International Conference on Image Processing*, pages 4165–4169, Sept 2016.
- [142] Shengli Wu. Applying statistical principles to data fusion in information retrieval. *Expert Systems with Applications*, 36(2):2997–3006, March 2009.

- [143] Z. Wu, Y. Huang, L. Wang, X. Wang, and T. Tan. A comprehensive study on cross-view gait based human identification with deep CNNs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(2):209–226, 2017.
- [144] Zifeng Wu, Yongzhen Huang, and Liang Wang. Learning representative deep features for image set analysis. *IEEE Trans. on Multimedia*, 17(11):1960–1968, Nov 2015.
- [145] Tianjun Xiao, Jiaxing Zhang, Kuiyuan Yang, Yuxin Peng, and Zheng Zhang. Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In *ACM Multimedia*, 2014.
- [146] Zhaojun Xue, Dong Ming, Wei Song, Baikun Wan, and Shijiu Jin. Infrared gait recognition based on wavelet transform and support vector machine. *Pattern Recognition*, 43(8):2904 – 2910, 2010.
- [147] C. Yan, B. Zhang, and F. Coenen. Multi-attributes gait identification by convolutional neural networks. In *International Congress on Image and Signal Processing (CISP)*, pages 642–647, Oct 2015.
- [148] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6071–6079. IEEE, 2017.
- [149] Guangnan Ye, I-Hong Jhuo, Dong Liu, Yu-Gang Jiang, D. T. Lee, and Shih-Fu Chang. Joint audio-visual bi-modal codewords for video event detection. In *ICMR*, page 39, 2012.
- [150] Guangnan Ye, Dong Liu, I-Hong Jhuo, and Shih-Fu Chang. Robust late fusion with rank minimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3021–3028, 2012.
- [151] Shiqi Yu, Daoliang Tan, and Tieniu Tan. A framework for evaluating the effect of view angle, clothing and carrying condition on gait recognition. In *Proceedings of the International Conference on Pattern Recognition*, volume 4, pages 441–444, 2006.
- [152] Matthew D Zeiler. Adadelata: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

- [153] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.
- [154] Wei Zeng, Cong Wang, and Feifei Yang. Silhouette-based gait recognition via deterministic learning. *Pattern Recognition*, 47(11):3568 – 3584, 2014.
- [155] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 94–108. Springer, 2014.
- [156] Wenyi Zhao, Rama Chellappa, P Jonathon Phillips, and Azriel Rosenfeld. Face recognition: A literature survey. *ACM computing surveys (CSUR)*, 35(4):399–458, 2003.
- [157] Shuai Zheng, Kaiqi Huang, Tieniu Tan, and Dacheng Tao. A cascade fusion scheme for gait and cumulative foot pressure image recognition. *Pattern Recognition*, 45(10):3603 – 3610, 2012.



