Università Politecnica Delle Marche

Department of Information Engineering

MASTER DEGREE IN BIOMEDICAL ENGINEERING

End-to-end semantic joint detection and limb-pose estimation from depth images of preterm infants in NICUs

Advisor:

Prof. Emanuele Frontoni

Candidate:

Co-Advisor: *Greta Vallasciani*

Prof. Lorenzo Scalise
Sara Moccia, PhD

Academic Year 2020 / 2021

If you lose today, win tomorrow.
In this never-ending spirit of challenge
is found the heart of a victor.

Daisaku Ikeda

Summary

Continuous monitoring of preterm infant's spontaneous motility is crucial for early recognition of cognitive and behavioural disorders, allowing timely therapies and treatments. Automatic infants' limbs pose estimation is an important step toward supporting clinicians in infant monitoring and improving patients' care.

This work proposes an end-to-end framework for limb-pose estimation based on the novel and high-performance region-based CNN) by Facebook AI Research (FAIR), named Mask R-CNN. The proposed framework was validated on a custom dataset of 6000 depth images from 30 videos acquired in a neonatal intensive care unit during the actual clinical practice. A leave-one-infant-out cross-validation with 19 folds is performed to evaluate the framework performance on each different video. Results for joint detection showed a mean average precision equal to 0.9 with a standard deviation of 0.2. For limb-pose estimation, a median root mean square error [pixel] equal to 6.8 (right arm), 6.7 (left arm), 6.5 (right leg), 6.5 (left leg) was achieved. The interquartile ranges [pixels] were 1.1, 1.2, 0.6, 1.2 for each limb, respectively. This end-to-end framework does not require any prior modeling of infants' body structure, neither any manual interventions, and it can represent a step toward embedded monitoring solutions for on-the-edge computation.

Contents

Chapter 1

Introduction

This chapter will introduce the preterm infants clinical background (Sec. 1.1), focusing on the long-term cognitive and cerebral disorders associated with prematurity. Then, it is explained the importance of the continuous monitoring of infant spontaneous movement for the early recognition of infant neuro-developmental disorders, pointing out the contribution of the present thesis to this issue (Sec. 1.2). Finally, the SINC project will be introduced (Sec. 1.3).

1.1 Preterm birth

Preterm birth is defined by the World Health Organization (WHO) as a birth before 37 completed weeks of gestation. An estimated 15 million babies born preterm every year. That is more than 1 in 10 babies. In almost all high-income countries, complications of preterm birth are the largest direct cause of neonatal deaths, accounting for the 35% of the world deaths a year [1].

The babies who survive have to face a wide range of morbidities associated with prematurity, with the frequency and severity of adverse outcomes rising with decreasing gestational age and decreasing quality of care. In the short-term, preterm infants may experience respiratory distress and intraventricular hemorrhage, while, in the longer term, they may have worse neuro-developmental performance outcomes.

There is a wide consensus that early interventions may have a key role to

impact neuro-development and functional attainment based on optimizing neuroplasticity of the developing brain soon after birth [2]. Thus, earlier diagnosis is urgently needed to take full advantage of critical windows of early brain development. Despite that, in most clinical settings, the average age for cognitive and motor disorders diagnosis is typically from 12 to 24 months [3], hampering early onsets of interventions and therapies during optimal period [4].

The only clinical currently available method to assess the infant's motor impairment in the first months of life, is the General Movements Assessment (GMA) tool [5].

1.2 General Movement Assessment

General movements (GMs) are part of the infant's repertoire of spontaneous movements, g.e. a wide range of movement patterns starting as early as 8 weeks of gestational age and continuing after birth, expression of infant's spontaneous neural activity [6]. Consequently, the GMA presented by Prechtl et al. [5] is based on the fact that the form of typical GMs changes as a result of developmental transformations of the nervous system, representing a high predictor for later motor and cognitive performance. Changes in the normal quality of GMs are a reliable indicator of brain dysfunction [5].

The application of GMA clinically involves the evaluation of the GMs specific spatial-temporal organization, through Gestalt perception of the observer. Nowadays, GMs follow-up is mainly limited to infants' visual inspection by trained clinicians directly in Neonatal Intensive Care Units (NICUs), with drawbacks as being qualitative, discontinuous, inaccurate and prone to inter- and intra-clinician variability [6].

As a result of the nominal use of GMA in neonatal follow-up programs, several studies have tried to automate this method to obtain earlier, quantitative, and more accurate clinically feasible GM assessments [6].

In particular, the estimation of preterm infants' pose is a relevant research problem in order to automatize GMs monitoring [7] and quantify the character-

istics of the movement.

Thus, a possible solution to attenuate the problem of qualitative monitoring is to use an automatic video-based system for infants' limb-pose estimation [8]. Following this paradigm, the acquisition setup can be designed as described by [9] as shown in Fig. 1 (Fig: 1.1). The setup consists of a depth camera placed over the infants' crib leaving healthcare operators and parents free to move and interact with the infants. Moreover, the setup is designed to interfere with the infant's spontaneous motility and the choice of the depth camera rather than a Red-Green-Blue (RGB) one, is concerning the infant privacy issues.

The automatic system implemented in [9] uses Artificial Intelligence (AI) techniques for the spatio-temporal analysis of depth images. The use of neural network architectures aims to use the information deriving from the analysis of video sequences to classify the presence or absence of movement of the limbs of the newborn.

The present thesis aims to develop an end-to-end deep learning framework for preterm infant joint detection and limb-pose estimation from the depth images acquired in NICUs through the previously described setup.

1.3 SINC

The development of the automatic vision system described in this thesis is part of the System Improvement for Neonatal Care (SINC) project. The SINC is a project of the Marche region carried out in collaboration with four companies, two research departments of the Università Politecnica delle Marche and of the "G. Salesi Hospital for Women and Children" in Ancona as coordinator of all regional NICUs. The project concerns neonatal care and aims to study, develop and test an innovative system on real cases that consists of new products to support a new organizational model. The SINC project aims to transform baby cribs into intelligent systems. An intelligent crib for preterm infants consists of a system that controls, via camera and contactless sensors, with the help of AI, heart and respiratory rate and temperature, movement and crying, as well as

Figure 1.1: *Sample of NICU acquisition setup. The Red-Green-Blue-Depth (RGB-D) camera (red box) is at approximately 40 cm over the infant's crib, avoiding hindering the operators [9]. On the top-left corner, an example of acquired depth image is showed.*

bilirubin, fundamental parameters for determining state of health of premature infants.The main objectives of the project are:

- Transform the cradle into a smart device by integrating new systems for the detection of the main physiological parameters.
- Creation and management of a cloud service that makes e. Services available in the Marche region utilities able to collect and integrate the monitoring data detected by the devices in order to favor diagnosis.
- Realization and experimentation of a new hospital / territorial model for the integrated management of neonatal care.

AI will be able to extract useful information on the activity of the newborn. In particular, this thesis regards the development of an AI algorithm for preterm infants' pose estimation, extracting information from the infants' video acquired through the camera of the intelligent crib in the NICU of G. Salesi Hospital

in Ancona. In Sec. 1.4, a brief introduction of the work done in this thesis is showed.

1.4 Thesis overview

This thesis presents an end-to-end framework based on deep learning, for estimating preterm infants' limb pose from depth video recordings acquired in the actual clinical practice. The thesis will be structured as follows:

- Chapter 2 encloses the main and more recent techniques for infant's motion analysis at the state-of-the art
- Chapter 3 describes the fundamentals of deep learning
- Chapter 4 describes the data and the developed deep learning framework
- Chapter 5 describes the experimental protocol
- Chapter 6 shows the obtained results
- Chapter 7 discusses the results and future developments
- Chapter 8 presents conclusions

Chapter 2

State of the art

Over the last few years, thanks to the advances in sensor technologies and computational resources, numerous computer-based solutions have been proposed to continuously monitor and analyze infants' movement.

In Sec. 2.1 different infant movement assessment approaches found in the literature are be briefly summarized, distinguishing the technique involving wearable sensors (Sec. 2.1.1) from the ones regarding contact-less approaches (Sec. 2.1.2).

In Sec. 2.2 the main limitations of the state-of-the-art are discussed, and the contribution of the present thesis is introduced.

2.1 Methods of analysis of the movement of the preterm infants

Movement recognition is the basis of the infant movement assessment and the timely recognition of cognitive and behavioural disorders. In clinical applications it aims at the automated detection, classification, and assessment of the quality of infants' limb movements focusing on indications for abnormalities [10]. Although this problem is approached in different ways, the pipeline is similar for most systems, and can be divided into *motion capture* and *motion analysis* [11]. Regarding the motion capture techniques, two groups can be identified: *visual*

sensor-based approaches and *motion sensor*-based approaches. The first group of approaches either use markers on human body region or exploit marker-less solutions by incorporating the image features such as color, edges, etc. to detect and track the different body parts in video data. The second family of approaches use motion wearable sensors to encode the motion information. Motion features extracted from captured movements, are then generally used for training a classifier to predict the outcome [11].

2.1.1 Sensor-based approaches

Literature on movement analysis based on inertial measurement unit (IMU) sensors has grown rapidly and a wide range of analytic tools have been developed to analyse movement activity at different levels.

In [12] a Bluetooth-connected infant sensor suit is developed, pairing a one-piece infant suit and rattle socks with six 6-axis IMUs powered by a coin cell battery. The suit incorporates 3 sensors per leg, placed on the thigh, shin, and foot to gather 3-axis acceleration and gyroscope data for each of the limb segments (Fig. 2.1). An activity detection algorithm is then used to quantify kicking activity derived from collected measurement data. The data are collected from term and low-risk preterm infants wearing the suit.

Figure 2.1: *Example of sensor placement for infant's left leg, by Fry et al. [12].*

Figure 2.2: *Photograph of the smart jumpsuit with four proximally placed movement sensors, by Airaksinen et al. [13].*

In [13], for tracking the posture and movement of infants, it is developed a multi-sensor "smart" jumpsuit that allows mobile accelerometer and gyroscope data collection during movements (Fig. 2.2). A total of four wireless sensors are mounted proximally in the upper arms and legs. Each sensor has a built-in IMU, consisting of a triaxial accelerometer and a gyroscope. Using this suit, movements in approximately 7-month old infants are recorded. These data were manually annotated for infant posture and movement based on video recordings of the sessions, and using a novel annotation scheme specifically designed to assess the overall movement pattern of infants in the given age group. A machine learning algorithm, based on deep CNNs was then trained for automatic detection of posture and movement classes using the data and annotations.

In [14] a custom movement measurement system is designed for use in infant biomechanical studies. The authors designed a network of 6 wearable movement sensors in the neck and head, the trunk, and in all extremities (Fig. 2.3). This sensor network was designed for use with infants during the peak manifestation of fidgety general movements from 12 to 20 weeks post term age. It is able both to quantify the presence or absence of fidgety movements, but also provide insight into the relative distribution of spontaneous and gross motor movements towards identification of left/right or upper/lower asymmetries that may be useful towards predicting distribution or severity of cognitive and be-

Figure 2.3: *Representation of the wearable sensor network during data collection presented by Redd et al. [14].*

havioural disorders. However, they presented the pilot data on a single infant, and the single trial is not sufficient to demonstrate diagnostic efficacy or potential impact within the population at large.

All the aforementioned articles share the same limitations related to the use of contact measurement techniques: the application of wearable sensors on the preterm infant's skin can lead to baby's discomfort and pain, also interfering with infants' mobility. It should be also noted that these studies are referring only to at term or low-risk preterm infants, with small datasets or even with a single trial.

2.1.2 Vision-based approaches

Other than sensor-based approaches, vision-based approaches do not measure motions directly. These techniques use color images, depth information or both for movement analysis. Some approaches involves markers attached to the infant's body parts to represent the joints' locations, and use them to detect and track the skeleton in a video to encode the motion information. The others camera-based approaches exploit image features such as color, shape, and edges to estimate the joints' locations for movement analysis.

Marker-based

In Miyagishima et al. [15], spontaneous movements of the infants are recorded using a 3D motion capture system Vicon 512 (Oxford Metrics). Eight infrared cameras mounted on tripods recorded synchronously the movement of the single markers. 6.0 mm diameter markers are located at hands and foots. They normalize the marker displacement data using the limb length in each infant. The distance between both hands and between both feet and the height of both hands and feet are used as indexes of antigravity movements [15].

Methods relying on attached markers present some of the same challenges as wearable sensors. They require human intervention for marker attachment and wearing a large number of sensors or markers may cause discomfort to the young patients which may affect their natural body part movements. Still, they use computer vision for tracking the pose of the infants. The high cost of the system, the complex setup and calibration, and the occlusion problems stand against the highly accurate tracking of joints in 3D. Due to practical limitations these systems are most commonly seen in the research setting, being not easily adaptable to the clinical environment [16].

Camera-based

Cameras, opposed to motion sensors, are cheap, easy to use, require no setup or calibration, and can be easily integrated into standard examinations while not influencing infants' movements. This makes them more suitable for use in clinical or even domestic environments. Camera-based methods, used in the current state-of-the-art in infant motion analysis, involves pixel tracking methods or infant pose estimation for movement detection and machine learning or deep learning algorithms for classification of GM.

The proposed solutions can be divided according to the method of the extraction of features describing the child's movement, distinguishing those based on features extracted directly from the recording (optical flow, background subtraction) from pose-based features, in which the extraction of features is preceded

by the process of locating individual body segments [17].

In Ilhen et al. [18], a five-step procedure is proposed for the processing of the RGB video recording: video screening, pre-processing, pixel tracking using Large Displacement Optical Flow (LDOF), segmentation of six body parts, and extraction of vertical and horizontal coordinates of body part's movements. LDOF is used to track pixel movements and a manual annotation is performed on 500 frames to identify the pixel center of six parts of the infant's body (i.e., arms, legs, head, and torso). The body parts' mean movement frequency, amplitude, and covariation is computed and finally resulted in a set of 990 features describing all the infant's movement repertoire. Linear discriminative analysis (LDA) is applied to classify movements typically found in children with or without cerebral palsy [18].

Tsuji et al. [19], inspired by [20], use frame differencing of RGB videos to estimate movements by tracking the centroid of motion, similarly to [21]. The video processing consist in two steps: first, the background frames are averaged and the resulting background image is subtracted from each frame of the video sequence in order to obtain a sequence of *binary-frames* using a brightness threshold (where 0-black is the background and 1-white is the infant's body); then the *difference-frame* is obtained by performing the pixel by pixel difference between time-adjacent frames, in order to generate additional binary images using a brightness threshold (where 1-white represents a pixel in which infant movement has been detected) (Fig. 2.4).

As highlited by the authors, binarization implies a radical loss of eidetic information for each frame of a sequence. Features relate to changes in body posture and movement, the velocity and fluctuation of the body centre are extracted and fed into a feedforward-type neural network called the log-linearized Gaussian mixture network to classify the limbs movements.

Both these approaches proposed by these authors lack of robustness in dealing with illumination changes and body part dimensions [22]. They are threshold-based, not fully automatic or only used for the whole body movement analysis. Also, limitations concerning the infants' privacy issues are related with the use

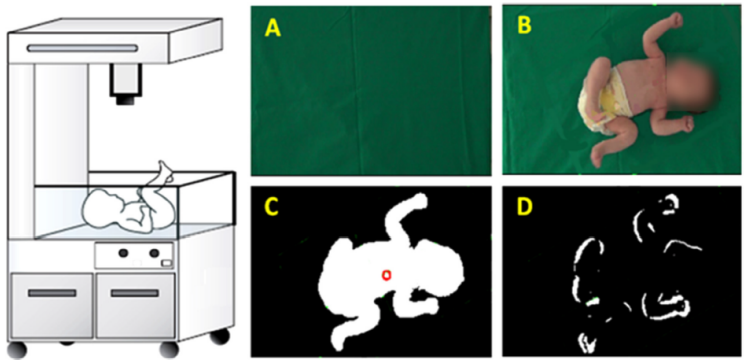

Figure 2.4: *The experimental setup system of Tacchino et al. [21] at left, and the relative pre-processing of the video sequences at right. The background and one frame of a video sequence at the panel A and B respectively. One binary image, also including as a red circle the instantaneous position of the centroid of the body silhouette, at panel C. An example of "difference-frame" at panel D.*

of RGB camera.

Pose estimation

Researchers have recently started to evaluate the effectiveness of pose-based assessment. The automated estimation of human pose from 2D images is an active research area, with several significant recent contributions. With the continued progression in deep learning techniques, various robust frameworks have been proposed which can accurately estimate human poses from 2D images [22].

In 2017, Cao et al. [23] introduced the OpenPose framework. It is a deep learning framework trained to detect human joint locations, namely keypoints, on single images, detecting both 2D position and orientation of human limbs using a non-parametric representation referred to as Part Affinity Fields (PAFs) [24]. The availability of this ready-made human pose estimation libraries leads to an improvement of the pose-based features extraction capability [17], proven by the extensive use of the OpenPose framework customized for infants in the most recent studies [25, 22, 17, 24].

Marchi et al. [25] produced skeleton videos and extracted motion features, such as the position and speed information of upper limb movements of 8-17

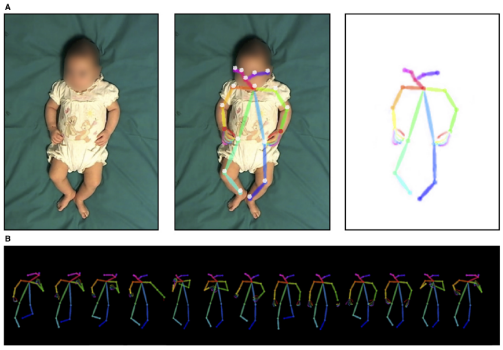

Figure 2.5: *Pose estimation procedure from the video image to skeleton videos by [25] In the original video image (A, left side), anatomical key points are identified (A, middle and right side with dots in infant's joints) and a skeleton is formed by drawing lines between these key points. This is repeated for all consecutive video frames (B).*

week old infants from 21 conventional videos to identify atypical movements, customizing the OpenPose framework (Fig. 2.5). However, they indicated that pose tracking errors occurred in 14 out of 21 recorded videos, concluding that their approach would improve significantly by the use of 3D instead of 2D camera technology.

Also in [17] pose estimation is performed using the OpenPose library, then, using the trajectories of the distal parts of the limbs, a set of features is determined based on the parameters of the ellipse circumscribed on each trajectory (Fig. 2.6): the FMA (factor of movement's area), FMS (factor of movement's shape), and CMA (center of movement's area) were determined. Machine learning methods are used to detect writhing movements on RGB video recordings of children on the second and third day after birth.

In 2019, Hesse et al. [11] created the Moving INfants In RGB-D (MINI-RGBD) dataset containing 12 sequences of real infant movements with varying realistic textures, shapes and backgrounds. Using this dataset, they evaluated their Random Ferns based 3D pose estimation method. Later, the MINI-RGBD dataset is used by McCay et al. [22] to establish histogram-based pose features, such as Histogram of Joint Orientation 2D (HOJO2D) and Histogram of Joint Displacement 2D (HOJD2D) to identify atypical movements with the use of the

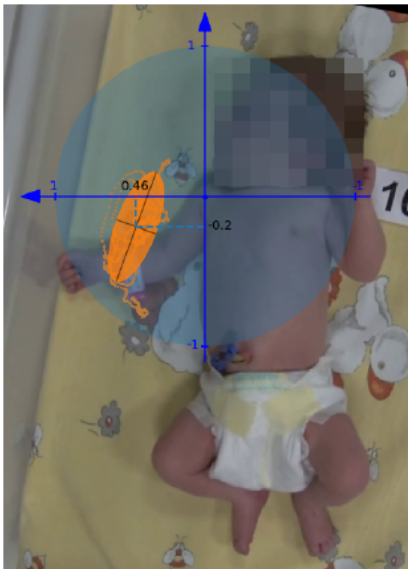

Figure 2.6: *Visualization of the coordinate system associated with the right shoulder used by Doroniewicz et al. [17] to normalize the parameters of the ellipse circumscribed on the trajectory of the right wrist. The blue circle shows the possible range of motion used to normalize the value of the area of the ellipse circumscribed on the trajectory (orange). Minor and major axes are marked inside the ellipse.*

OpenPose framework. The hand-crafted features generated are then fed into a deep learning framework for classification [24].

All the aforementioned articles are based on recording conventional 2D color videos. The single camera setup resulted in a reduction of the three-dimensional motion of the limbs to a bidimensional space implying a reduction of information, and these RGB cameras also raised issues concerning the infants' privacy protection. With the advent of low cost RGB-D sensors, motion analysis approaches started taking advantage of depth information.

In 2017, Hesse et al. [26] proposed a method for estimating pose and shape of infants, learning a statistical 3D Skinned Multi-Infant Linear body model (SMIL) from incomplete, low-quality RGB-D sequences of freely moving infants. Quantitative experiments show that SMIL faithfully represents the RGB-D data and properly factorizes the shape and pose of the infants.

In a recent work of Moccia et al. [8] a preterm infants' limb pose estimation technique is proposed, consisting of two consecutive CNNs extracting pose

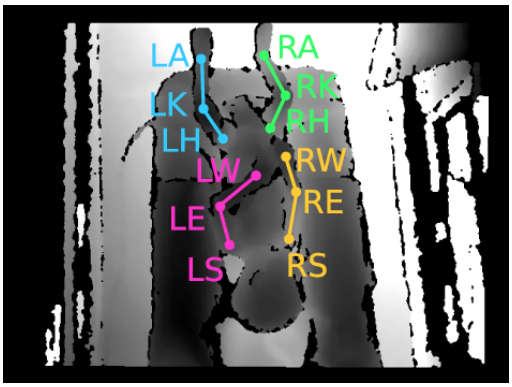

Figure 2.7: *Infant model proposed by Moccia et al. [8].*

directly from depth images acquired in the NICU during the actual clinical practice (Fig. 2.7), including spatio-temporal information. The approach uses the babyPose dataset, consisted of 16 depth videos of 16 preterm infants.

2.2 The problem of infant dataset

There are multiple reasons why no public repository of infant 3D scans exists. Protection of privacy of infants is more strict as compared to adults. The high cost of 3D scanners prevents them from being widespread. Creating a scanning environment that takes into consideration the special care required by infants, like warmth and hygiene, requires additional effort. Finally, infants can not be instructed to strike poses on demand, which is usually required in standard body scanning protocols [26].

2.3 Main contributions

The methods present in the state of the art show a series of limitations:
(i) Wearable sensors could irritate the infant's skin, cause incorrect measurements following recalibration, but even worse, generate unwanted movements
(ii) The algorithms to analyze the images acquired by the cameras, are non-invasive, but in most cases they rely on semi-automatic approaches, with high computational costs and complex pipelines.

To overcome the limitations of the state of the art, based on the study of Moccia et al. [8], the work done in this thesis aims to develop an end-to-end deep learning framework for preterm infant joint detection and limb-pose estimation from depth images acquired in NICUs. The high performance novel framework for human pose estimation proposed by Facebook AI Research (FAIR) is used in this work to develop the end-to-end model easily deployable in a clinical or domestic environment.

Chapter 3

Deep Learning

This chapter will introduce the basic concepts behind Deep Neural Networks (Sec. 3.1 - 3.4). In Sec. 3.5 region-based convolutional neural network architectures, that precedes the one used in the present work, are described. Finally, the deep residual network (Sec. 3.6) and the feature-pyramid network (Sec. 3.7) are briefly introduced.

3.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are biologically inspired computer programs designed to simulate the way neurons in our brain work and propagate the information, thus the human brain learning process [27]. The result is a powerful learning method: ANNs are able to inductively acquire concepts from examples

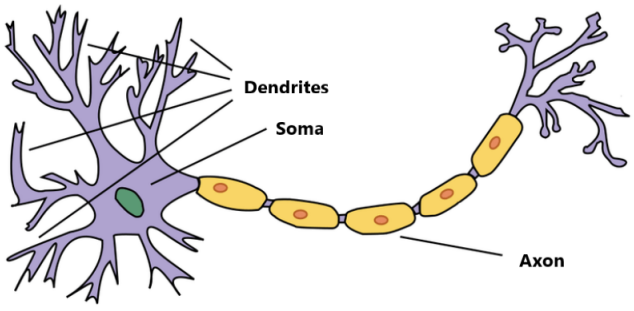

Figure 3.1: *Biological neuron.*

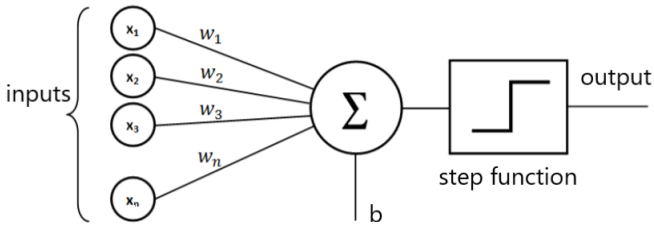

Figure 3.2: *Perceptron.*

[28], i.e., to learn and generalize from data, mimicking the human capability to learn from experience. ANNs are used to solve problems with many multivariate parameters, when the exact analytical model does not exist or it reveals too complex.

An ANN structure is charachterized by hundreds of single units, i.e. artificial neurons, organized as networks whose interconnections are similar in some respects to the way in which neurons are inter connected in the visual cortex of mammals [29].

Let consider a biological neuron in order to understand the basic idea behind neural networks. The neuron (Fig. 3.1) is able to receive inputs (i.e. electrical signals in our brain), perform processing and produce an output (also an electrical signal). It is important to note that the inputs and outputs are binary (0 or 1). A single neuron receive inputs, usually from other neurons, through its dendrites. Dendrites connect with other neurons through a space, named *synapse*, which assigns weight to a particular input. All the received inputs are summed and processed together in the cell body, or *soma*. Neurons exhibit a *all-or-nothing* behavior: only if the combination of inputs exceeds a certain threshold, an output signal is produced. In case the neuron activates, the output travels along the *axon* to the *axon terminals*, which are connected to the dendrites of other neurons through synapses.

The simplest form of ANN is the *perceptron*, proposed by Frank Rosenblatt [30]. The perceptron model, represented as in Figure 3.2, consists of binary inputs (x_m), usually given as a vector, interacting with synapses (i.e.weights w_m) through the multiplication $w_m * x_m$. Then, the weighted inputs are summed

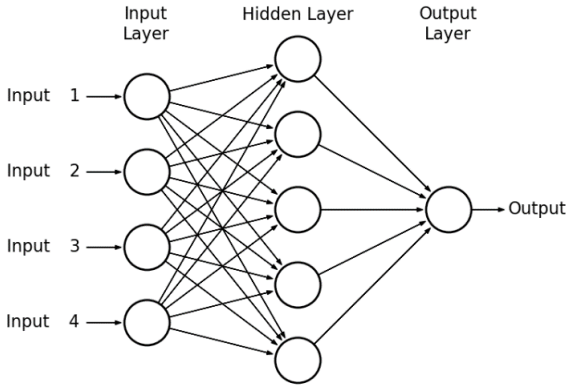

Figure 3.3: *Multi Layer Perceptron.*

together in the artificial neuron body, also adding another constant factor, called *bias* (b). The weighted sum is modeled by, an activation function (g), that limits the amplitude of the output of the neurons. The activation function for perceptron is the *step function*: if the input is greater than or equal to 0, then the output is 1, otherwise is 0. Finally, we can express the output ($\hat{y}$) like this:

$$output = \hat{y} = g(\sum_{m=1}^{M} w_m x_m + w_b b) \tag{3.1}$$

with M the number of inputs.

However, the perceptron is effective only when the features of the input data are linearly separable (i.e. separable by a hyperplane). In 80s-90s there was the introduction of the Multi-Layer Perceptron (MLP), a multi-layered network to solve non linearly separable data. A layer in the network is the set of nodes (neurons) in a column of the network [29]. All the nodes in the network are perpeptron-like, except for the input layer, whose nodes are the components of an input pattern vector x [29]. Each layer in the network can have a different number of nodes, but each node has a single output. The key innovation is that at least one layer is *hidden*, as it is neither an input layer nor an output layer, allowing the network to build an internal representation of the data, in a feature space, before producing an output. The Fig. 3.3 shows an MLP with three layers. In the standard form, the MLP is *fully-connected*: all the neurons'

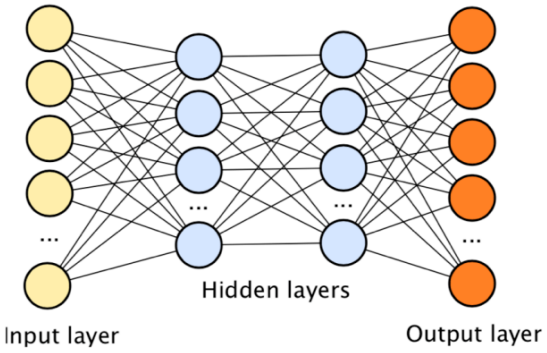

Figure 3.4: *Esample of Deep Neural Network architecture.*

outputs of a layer are connected to all neurons' inputs of the next layer. The activation function of the neurons in the hidden layer are nonlinear. Generally, neural networks with a single hidden layer are referred to as *shallow neural network* [29], while networks with two or more hidden layers are named *Deep Neural Network* (Fig. 3.4).

With the MLP, it was introduced also the idea of error back-propagation for the adjustment of the weight in the hidden layer. Unlike the pattern recognition approaches relying on feature engineering techniques to extract feature from raw data, neural networks can use back-propagation to automatically learn representations suitable for recognition, starting with raw data [29]. Each layer in the network "refines" the representation into more abstract levels [29]. This type of multilayered learning is commonly referred to as *Deep Learning*. In the two decades following the introduction of back-propagation, neural networks have been used successfully in a broad range of applications, such as medical diagnosis, speech and pattern recognition, becoming also integral part of our everyday life [29].

3.2 Training Neural Networks

According to the above, neural network is characterized by its weights, biases, and activation function. Training a neural network refers to using one or more sets of training patterns to estimate these parameters [29]. ANN are trained

primarily to solve two problems:

- Regression: the algorithm outputs numerical values in response to input.
- Classification: the algorithm associates one or more classes to each of the inputs, assigning them one or more label.

This thesis is dealing only with a training approach called *Supervised Learning*: both the data and the expected output for each data (its label) are fed as input to the network. The first step of the training process is the initialization of the network weight with random values. The training samples are then fed as input to the network and processed through the hidden layers, and finally the related output is obtained, called prediction. The prediction is compared with the desired output by calculating the prediction error, defined by a *loss function*. Thus, the network training consists in the minimization of the loss, that is representing the difference between the expected output (the label) and the actual prediction of the network. The loss function, or *Loss*, can be defined as:

$$L(W) = \frac{1}{N} \sum_{i=1}^{N} (Li(f(x_i, W), y_i)) \tag{3.2}$$

where:

- N: number of training samples
- W: weights matrix
- $f(x_i, W)$: prediction for sample x_i
- y_i: expected output for sample x_i
- $Li(f(x_i, W), y_i)$: prediction error

The Loss is dependent on the weights and biases values. Thus, in other words, the training of a neural network is the process of finding the configuration of weights and biases that maximize the performance of the model and minimize the Loss value. The Loss minimization problem is defined as follow:

$$min\left\{L(W)\right\} = min\left\{\frac{1}{N}\sum_{i=1}^{N}(Li(f(x_i, W), y_i))) + \lambda R(W)\right\} \tag{3.3}$$

where $R(W)$ and λ are the regularization term and parameter.

Before going into the discussion, it is good to make the following clarifications. The training set can be divided into *batches*, each one containing a number of examples equal to the *batch-size*. With the term *iteration* we want to indicate the passage in feed-forward of only one batch. When each batch of the training set is seen by the network, we will have the end of an *epoch*.

Once the ANN has been *trained* with a set of examples, it can be used to predict the outcome of another new set of similar input data.

3.2.1 Gradient descent and back-propagation

The training of ANNs is an optimization problem: we are searching the values of W to minimize the Loss (formula 3.3). The optimization algorithm used to solve this problem is the *gradient descent*. Considering the weight space, i.e. the space in which each point represents the loss value resulting from specific combination of weights, the problem consist in finding the direction towards the global minimum of that space, thus a direction of descent. The gradient is the vector of the partial derivatives of the Loss along each dimension w_i:

$$\nabla_w L(W) = [\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, ..., \frac{\partial L}{\partial w_n}] \tag{3.4}$$

where each partial derivative gives the contribution of the relative weight w_i to the Loss. Mathematically the direction of descent is given by the negative gradient: $(-\nabla_w L(W))$. The gradient descent is an iterative procedure, which at each step updates the weight vector W moving towards the minimum of the Loss, as follow:

$$W_{new} = W_{old} - \eta \nabla_w L(W) \tag{3.5}$$

where η is a positive, configurable hyper-parameter, called *learning rate*, regulating the weight update. It gives the dimension of the step done in the weight space toward the new position. A too high learning rate can lead to excessive 'jumps', cause a lack of convergence. Conversely, the choice of an excessively small learning rate could slow down a lot the convergence process, requiring a higher number of epochs in order to achieve acceptable results. There are numerous approaches that attempt to find optimal learning rates, but ultimately this is a problem-dependent parameter that involves experimenting. A reasonable approach is to start with a small value of (e.g., 0.01), then experiment with vectors from the training set to determine a suitable value in a given application. We have no way for computing the gradients of the weights in the hidden nodes. The back-propagation algorithm solves this issues, by propagating the output error back into the network. The process that starts from the weights coefficients to calculate the predictions is known as *forward propagation*. Conversely, the process that allows to optimize the coefficients starting from the error previously calculated is called *backward propagation* or *back-propagation*. Training by back-propagation involves four basic steps: (1) inputting the pattern vectors; (2) a forward pass through the network to classify all the patterns of the training set and determine the classification error; (3) a backward pass that feeds the output error back through the network to compute the changes required to update the parameters; and (4) updating the weights and biases in the network. These steps are repeated until the error reaches an acceptable level [29].

The gradient can be computed at each iteration considering all the training sample, but it is infeasible for large datasets. Alternatively, the gradient can be computed on a single sample randomly chosen, reducing the executive time but leading to gradient's oscillations avoiding the convergence to the minimum Loss value. Finally, the best compromise results in *mini batch gradient descent*, computing gradients on a small number of data samples. The update of this good but not perfect gradients moves the weights overall in the right direction.

3.2.2 Loss functions

Cross-entropy

Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. As the predicted probability approaches 1, log loss slowly decreases. As the predicted probability decreases, however, the log loss increases rapidly. Cross-entropy is defined as:

$$\mathcal{L} = \sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{3.6}$$

where M is the number of classes, *log* is the natural log, y is binary indicator (0 or 1) if class label c is the correct classification for observation o, and p is the predicted probability that observation o belongs to c.

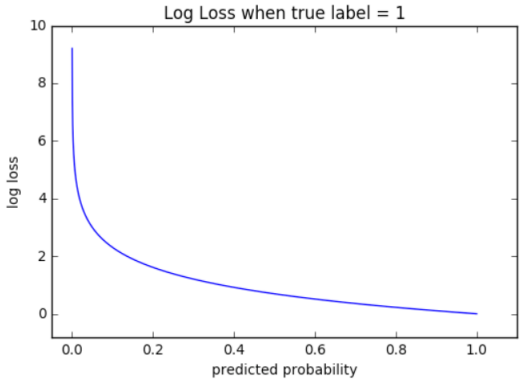

Figure 3.5: *Cross-entropy function.*

Mean Absolute Error or L1

Mean Absolute Error *(MAE)* loss function, or L1, is the sum of absolute differences between the target (y_{true}) and predicted ($y_{predicted}$) variables.

$$\mathcal{L}_1 = \sum_{i=0}^{n} |y_{true} - y_{predicted}| \tag{3.7}$$

3.3 Activation functions

Activation functions are used to determine the output of neural network. The Activation Functions can be basically divided into 2 types: linear and non-linear. In this section are described the most used non-linear activation functions, shown in Fig. 3.6

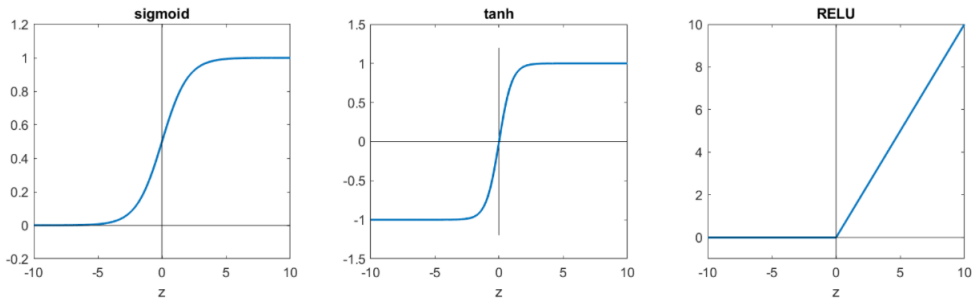

Figure 3.6: *Three exapmles of activaction functions: at left panel the Sigmoid function, at the center the hyperbolic tangent (tanh), at right the Rectified Linear Unit (ReLU)*

3.3.1 Sigmoid

The Sigmoid function curve looks like a S-shape (Fig. 3.6 left panel). The input to the function is transformed into a value between 0.0 and 1.0. Inputs that are much larger than 1 are transformed to the value 1, similarly, values much smaller than 0 are snapped to 0. Therefore, it is especially used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice. Mathematically we can represent the sigmoid in this way:

$$sigmoid(x) = \frac{1}{(1 + e^{-}x)} \tag{3.8}$$

3.3.2 Hyperbolic tangent

The hyperbolic tangent function (tanh), in middle panel of Fig. 3.6, is a similar shaped nonlinear activation function that outputs values between -1.0 and 1.0.

$$\tanh(x) = \frac{sinh(x)}{cosh(x)} \tag{3.9}$$

3.3.3 Rectifed Linear Unit

The function in right panel of Fig. 3.6 is called the rectifier function (ReLU). It is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It is the activation function most used in ANNs and has replaced over the years the hyperbolic and sigmoid tangent functions. The main reason it is due to the fact that tanh and sigmoid have a very small first derivative, which quickly tends to zero. Since the training of a neural network is based on the gradient descent, the multiplication for a value close to zero lead to slower learning.

$$ReLU(x) = max(0, x) \tag{3.10}$$

3.3.4 Softmax

The softmax function allows to calculate the probability distribution of an event on n different events. In general, this function is used for classification problems and calculates the probability of an example to belong to a certain target class. The main advantage of using softmax is that the probability range is in the range [0,1], and the sum of all probabilities will be equal to one. For classification problems with only two classes (binary classification), the sigmoid activation function can be used. Instead, if the number of classes is greater, the softmax activation function can defined as follows:

$$Softmax(x) = \frac{e^{x_i}}{\sum_{i=1}^{C} e^{x_i}} \tag{3.11}$$

where C indicates the total number of classes of the problem in question.

3.4 Convolutional Neural Network

As mentioned in the previous section, neural networks are capable to learn features directly from training data, thus reducing the need for "engineered" features. CNNs are a class of neural networks primarily used for pattern recognition within images. CNNs learn 2-D features directly from raw image data. The image is considered as a matrix of neurons in which the neuron in place *(i, j)* will be associated with the intensity of pixel in position *(i, j)* of the input image. In the fully connected network seen before, the output of every neuron in a layer is directly fed into the input of every neuron in the next layer. Differently, in the CNNs each neuron of one layer receives a single value, that is obtained by the *convolution* between a region of the input image (i.e. the output of the previous layer) called *receptive fields*, and a set of weight arranged in the shape of the receptive field, called *kernel*.

The receptive field is sliding over the input image and, at each location, it is computed the sum of products between the pixels contained in the receptive field and the set of kernel's weights, as shown in Fig. 3.7. A kernel is a smaller-sized matrix in comparison to the input dimensions of the image, and the kernel's values are learned during the training phase. Each neuron of the hidden layer uses the same weights and the same bias: the sharing of weights and bias for all neurons of the convolutional layer means they learns the same characteristic, albeit in different areas of the image. Thus, a certain feature can be detected even when located at different positions. This highlights the invariance of CNNs with respect to translations. With respect to fully connected network, the use of kernel reduces the number of parameters to the filter dimensions, regardless of the size of the input image. The output of the convolution between the input image and the kernel is called *feature map*, referring to the role performed by convolution operation, i.e. to extract features such as edges, points, and blobs from the input. Within a convolutional layer, there is often not only a single filter, but many are used: each of them will learn a characteristic of the image. The convolution operation is defined by its *stride* and *padding*. The stride is the

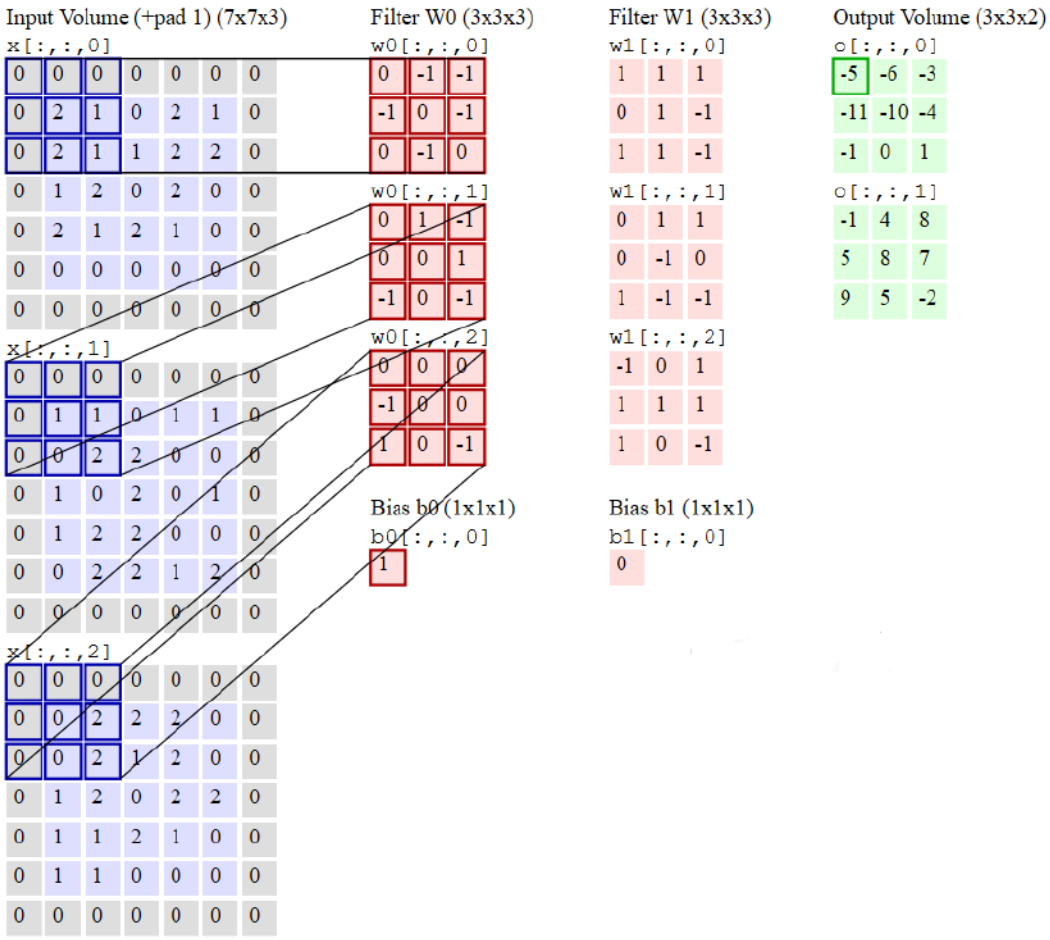

Figure 3.7: *Example of convolution between the input matrix* $7 \times 7 \times 7$ *and two filters (*W_0 *and* W_1*)* $3 \times 3 \times 3$ *with stride equal to 2.*

number of spatial increments by which a receptive field is moved. Strides greater than one is used for data reduction. For example, changing the stride from one to two reduces the image resolution by one-half in each spatial dimension, resulting in a three-fourths reduction in the amount of data per image. Otherwise, stride can be used as substitute for subsampling to reduce system sensitivity to spatial translation. Sometimes filter does not perfectly fit the input image. *Padding* is to pad the picture with zeros (zero-padding) to allow this fit. Once the feature map is obtained the ReLU activation function is applied, setting to zero all nodes with negative values. Often a CNN has multiple convolutional layers: starting from the input layers, the first convolutional layers extract low level characteristics,

such as angles, horizontal and vertical lines, while the latest convolutional layers allow to extract significant features of the image, such as faces or objects. Once all the operations involving the convolutional and di layers have been completed pooling, you will get a matrix of dimension $W \times H \times D$. By means of an operation called *flattening*, it is possible to transform it into a vector of dimension $W \cdot H \cdot D$. At this point, this vector is fed into fully-connected layers. The last fully-connected layer will have a number of nodes equal to the total number of classes defined for the classification problem. The output vector, generated by softmax classification function, will match the probability associated with the network that the object belongs to the i-th class.

3.5 Region-based CNN

Region-based CNNs (R-CNNs) are approaches applying deep learnning to object detection [31]. A naive approach to solve detection problem would be to take different regions of interest from the image, and use a CNN to classify the presence of the object within that region. The drawback is that the objects of interest might have different spatial locations within the image and different aspect ratios. Hence, a huge number of regions have to be selected and this could computationally blow up.

To overcame this problem, Ross Girshick et al. [31] proposed the use of an

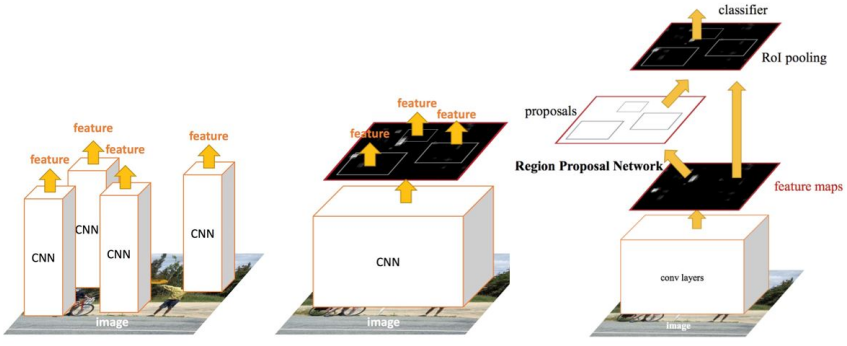

Figure 3.8: *Comparison between R-CNN (left panel), Fast R-CNN (central panel), and Faster R-CNN (right panel) architectures.*

external algorithm, named *Selective search*, aimed to extract from the image just 2000 regions. These regions are called *region proposals* and represent the first bounding box candidates.

The selective search performs a bottom-up non-object segmentation, based on filtering approach: pixels are grouped in small segmented areas according to color similarities, texture similarities, region size and region filling, and then the small segmented areas are merged together to form larger segmented areas. The 2000 region proposals are then warped into 224 x 224 squares, and fed into a CNN generating a 4096-dimensional vector for each of them. All these vectors should be saved to disk, in order to finally use them as input to a support vector machine that classifies the presence of the object within each candidate proposal, and to a linear regressor classifier helping in adjusting the bounding box of the region proposals.

This approach presents lots of drawbacks. Firstly, there is no training phase and learning from selective search, lacking of flexibility and leading to bad generated proposal. The warping of region proposals to squared size leads to the loss of bounding box aspect ratio and of image information. Moreover, due to number of windows it processed, it takes 47 seconds for each test image, which is not good enough for a real-time object detection system.

3.5.1 Fast RCNN

Compared with the R-CNN, in the Fast R-CNN [32] the input of the CNN for feature extraction is the entire image, rather than individual region proposals, generating a convolutional feature map. From the convolutional feature map, the region of proposals are identified applying the Selective search and then are warped into squares by using a *region of interest (RoI) pooling* layer. The reshape into a fixed size is needed to feed them into the following fully connected layer, that is used for classification and bounding box regression From the RoI feature vector, we use a softmax layer to predict the class of the proposed region and also the offset values for the bounding box.

Fast R-CNN is faster than R-CNN because we don't have to feed 2000 region

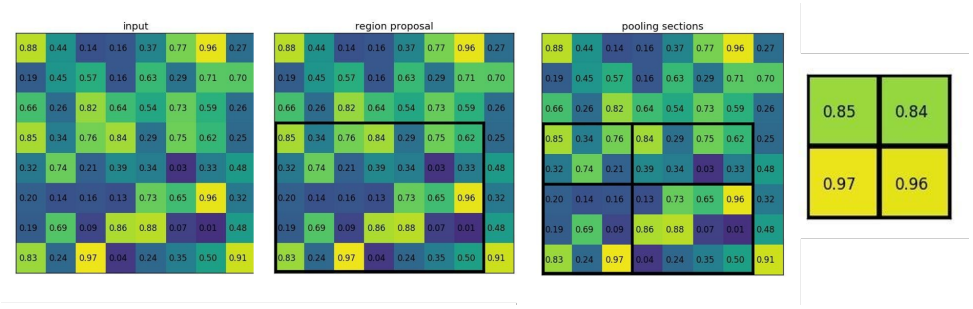

Figure 3.9: *RoI pooling applied on a single 8×8 matrix, one RoI 5x7 and an output size of 2×2. Notice that the size of the region of interest doesn't have to be perfectly divisible by the number of pooling sections. The max values in each of the sections are saved in the 2x2 output matrix.*

proposals to the CNN every time. Instead, the convolution operation is done only once per image and a feature map is generated from it.

RoI pooling

ROI pooling produces the fixed-size feature maps from non-uniform inputs by doing max-pooling on the inputs. Differently from the pooling layer introduced in Section 3.4., in the RoI pooling layer it is possible to directly specify the output shape.

Let's consider a small example to see how it works. It is considered a RoI with height h and weight w and it is subdivided by a $h_2 \times w_2$ grid of subwindows, where the shape of each subwindow is approximately $(h/h_2) \times (w/w_2)$. In practice, the height and width of any subwindow shall be rounded up, and the largest element shall be used as the output of the subwindow. Therefore, the region of interest pooling layer can extract features of the same shape even when RoI have different shapes. An illustrative example is shown in Fig. 3.10

3.5.2 Faster RCNN

Both of the algorithms of R-CNN and Fast R-CNN use selective search to find out the region proposals.

Selective search is a slow and time-consuming process affecting the performance of the network. Therefore, [33] came up with an object detection algo-

rithm that eliminates the selective search algorithm and lets the network learn the region proposals. It is called Faster R-CNN [33].

Similar to Fast R-CNN, in Faster R-CNN the image is provided as an input to a CNN which provides a convolutional feature map. Instead of using selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals. The predicted region proposals are then reshaped using a RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes. Faster R-CNN is much faster than it's predecessors. Therefore, it can even be used for real-time object detection [33].

Region Proposal Network

Next to the last layer of the first CNN within Faster R-CNN, a sliding window having dimension nxn (n = 3, [33]) will slide along the feature map generated by the backbone network in order to determine the region proposal boxes, denoted also as anchors. The RPN generates k anchor boxes with different scales and aspect ratios, centered to each point in the feature map (*anchor point*). By default there are 3 different scale values and 3 different aspect ratios, for a total of 9 anchors for each sliding window. For a feature map having dimension W x H, $W \cdot H \cdot k$ will be extracted anchor boxes. Then, two tasks are achieved through two convolution layers on the feature map: the classification of each anchor box whether it is foreground or background; the shape offsets for anchor boxes are learnt to fit them for objects, since the boxes need to be at image dimensions, whereas the feature map is reduced depending on the backbone. The final proposals are propagated forward through the ROI pooling layer and fully connected layers. These layers are equivalent to those proposed by Fast R-CNN, including the softmax classifier and the regressor of bounding box.

The region proposal network is jointly trained with the rest of the model. As a result of the end-to-end training, the region proposal network learns how to generate high-quality region proposals, so as to stay accurate in object detection with a reduced number of region proposals that are learned from data.

Loss functions

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{box} \tag{3.12}$$

$$\mathcal{L}(\{\hat{p}_i\}\{\hat{t}_i\}) = \frac{1}{N_{cls}} \sum_i \mathcal{L}_{cls}(\hat{p}_i, p_i) + \frac{\lambda}{N_{box}} \sum_i p_i smooth_{L1}(\hat{t}_i - t_i) \tag{3.13}$$

where $\mathcal{L}_{cls}$ is the binary log-loss; $\hat{p}$ is the predicted probability of an anchor to belong a class; $\hat{t}$ is the predicted coordinates; p and t are the corresponding ground true values; and N_{cls} and N_{box} are two normalization terms. λ is balancing the two losses. $Smooth_{L1}$ is defined as:

$$smooth_{L1} = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

3.6 Deep Residual Network

One of the problems of deep neural networks is the fact that, as it increases depth of the network, the accuracy of the model decreases. An innovative idea was proposed in 2015 through the introduction of the so-called Residual Neural Networks (ResNet) [34].

In the CNN, given an input x, its output value will be denoted with the notation F(x). The next layer receives as input the value returned by the previous

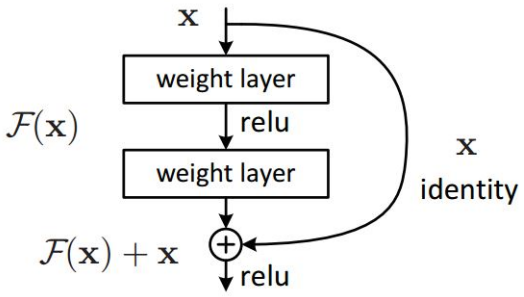

Figure 3.10: *ResNet single residual block.*

output, that is F(x). In a ResNet, the output will no longer be equal to F(x), but will assume the value of its residual, that is, $\mathcal{H}(x) = \mathcal{F}(x) + x$. A block of operations of this type it is also called a *building block*.

3.7 Feature Pyramid Network

Feature Pyramid Network (FPN) extracts feature maps and feeds them into the RPN, for object detection.

FPN provides a top-down pathway to construct higher resolution layers from a semantic rich layer. While the reconstructed layers are semantic strong but the locations of objects are not precise after all the downsampling and upsampling. We add lateral connections between reconstructed layers and the corresponding feature maps to help the detector to predict the location betters. It also acts as skip connections to make training easier (similar to what ResNet does).

The bottom-up pathway uses ResNet to construct the bottom-up pathway. It composes of many convolution modules each has many convolution layers. As we move up, the spatial dimension is reduced by 1/2 (i.e. double the stride). The output of each convolution module is labeled as Ci and later used in the top-down pathway.

In top-down pathway, the higher resolution features is upsampled by a factor of 2. The feature maps from bottom-up pathway undergoes 1×1 convolutions to reduce the channel dimensions, and then each lateral connection merges feature maps of the same spatial size from the bottom-up pathway and the top-down pathway, by element-wise addition. Finally, a 3×3 convolution is appended on each merged map to generate the final feature map, which is to reduce the aliasing effect of upsampling.

Chapter 4

Methods

This chapter presents the BabyPose dataset (Sec. 4.1) and the R-CNN models used for preterms' limbs pose estimation: the Mask RCNN [35] (Sec. 4.2).

4.1 BabyPose dataset

Firstly, the data acquisition setup and video characteristics are described (Sec. 4.1.1) and then the data preparation and the limb-pose infant model are illustrated (Sec. 4.1.2)

4.1.1 Dataset

In this work, the dataset comprises 30 depth videos of 19 preterm infants. The choice of acquiring depth frames over RGB frames was made to protect patient privacy. The video capture setup, shown in Fig. 1.1, was designed to not hinder healthcare professionals in clinical practice. The video recordings, each of 300 seconds, were acquired for each infant using an Astra Mini Series - Orbbec, with a frame rate of 30 frames per second and an image size of 640x480 pixels. A frame selection is performed: for each video, 200 frames are chosen ensuring a certain variability in the child's pose and excluding frame with operator limbs hiding the infant. Dataset challenges (Fig. 4.1) includes: 1) different distance between camera and infants 2) varying illumination level, 3) the presence of self and external-occlusion and 4) different number of visible joints in the camera

field of view.

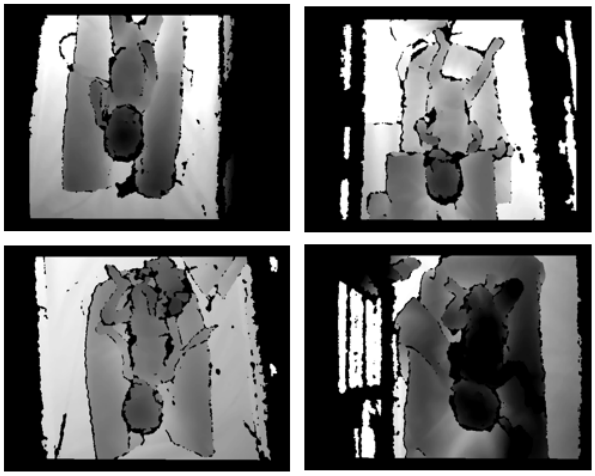

Figure 4.1: *Example of dataset challenges: different number of visible joints (top-left), self occlusion (top-right), external occlusion (bottom-left), varying infant-camera distance (bottom-right).*

4.1.2 Infant's model

Following the previous work of Moccia et al. [9], considering the importance of monitoring legs and arms for evaluating preterm infants' cognitive and motor development, the proposed limb-pose infant model considers each of the four limbs as a set of three connected joints (i.e., wrist, elbow and shoulder for arms, and ankle, knee and hip for legs). Joint annotation was performed on the overall 6000 frames (200 frames for each of 30 videos) using the COCO annotator tool, publicly available online [36]. The ground truth was indicated by manually annotating one keypoint for each of the twelve joints (left and right hand, left and right shoulder, left and right elbow, left and right foot, left and right knee, left and right hip). The annotator discriminates between visible and not-visible keypoints. Then, the ground truth for the bounding box containing the infant was also annotated. An example of annotation is reported in Fig. 4.2.

The COCO annotator tool allows to directly export a JSON file in the COCO format, with information about images, categories and annotations and then other dataset's information and licenses are added:

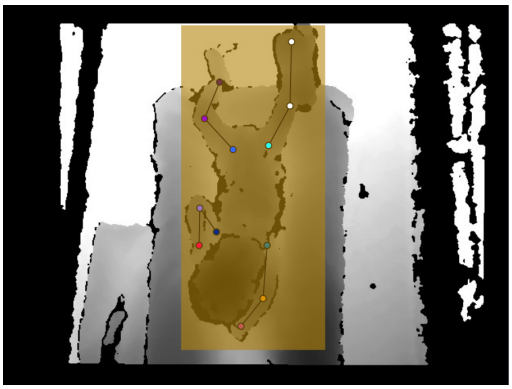

Figure 4.2: *Sample of annotation on a depth frame. Twelve joints were annotated on left and right limbs: hands, shoulders, elbows, feet, knees and hips. Colored circle for visible joints, white circle for externally occluded joints (by the limb plaster). The yellow rectangle is the bounding box annotation.*

Info — Description and versioning information about dataset.

Licenses — List of licenses with unique IDs to be specified by your images.

Categories — Classification categories each with a unique ID. In this work, only the *infant* category is present.

Images — List of images in the dataset and relevant metadata including unique image ID, file path, file name, file height and width.

Annotations — List of annotations each with a unique ID and the image ID it relates to. Here, the bounding box information and the coordinates pair (x and y) with a visibility flag (2 or 1 or 0) for each keypoint are stored. The number of labeled keypoints are also indicated. This field also stores bounding box *area* and *iscrowd* indicating a large bounding box surrounding multiple objects of the same category which is used for evaluation.

4.2 Mask R-CNN

The high performance novel framework for human pose estimation proposed by Facebook AI Research (FAIR), named Mask R-CNN [35], was used in this work to develop the end-to-end framework for semantic joint detection and limb-pose estimation from depth images of preterm infants.

4.2.1 The architecture

The Mask R-CNN [35] (Fig. 4.3) extends Faster R-CNN [33] detection framework, adding a mask prediction branch in parallel with the existing branch for classification and bounding box recognition.

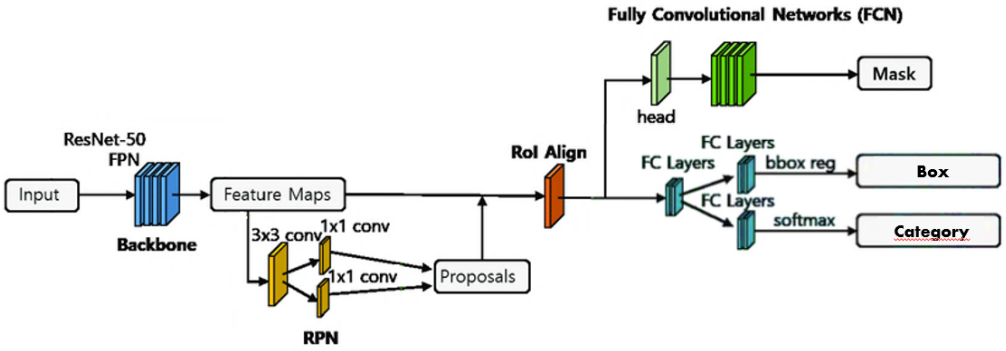

Figure 4.3: *The Mask R-CNN architecture is shown. FPN: Feature Pyramid Network; RPN: Region Proposal Network; FC: Fully Connected.*

Firstly, the 640x480 depth image of the infant is fed into the network. Two backbones are intended to extract accurate feature maps: the ResNet with 50 layers and the Feature Pyramid Network (FPN). Then this feature maps are fed into a two-stage architecture. The first stage consists in a Region Proposal Network (RPN), using a CNN to generate the multiple Region of Interest (RoI). Then, each region proposal is sent to the RoI Align layer. RoI Align is an operation for extracting a small feature map from each RoI. It is a simple, quantization-free layer, replacing the Faster R-CNN RoI Pooling, to avoid misalignments. Warped features outputs from the RoI Align are then fed into fully connected layers: they outputs the bounding box prediction for the infant and the label for *infant* category is assigned with the relative prediction confidence by the classification branch. In parallel, warped features are also fed into the mask branch properly adapted for keypoint detection task. The keypoint's location is modeled as one-hot mask: for a single input image with all twelve visible keypoints, the output is a set of twelve masks, one for each keypoint. The joints are also connected as previously described. The final output consists in

the depth frame with the superimposed bounding box, classification confidence, joints location and connections, as shown in Fig.4.4.

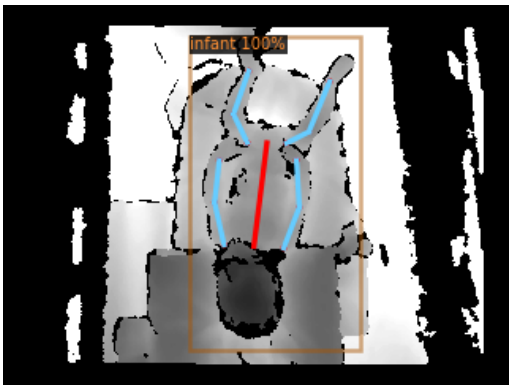

Figure 4.4: *Example of framework output. The blue lines connecting joints represents the limbs; the orange rectangle is the bounding box, and the category "infant" is assigned with a prediction confidence of 100 %. The red line is linking the middle points between hips and shoulders.*

4.2.2 The Loss function

The loss function relating to each RoI extracted will be equal to the sum of the three losses, one for each branch:

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{box} + \mathcal{L}_{mask} \tag{4.1}$$

where $\mathcal{L}_{cls}$ e $\mathcal{L}_{box}$ are the losses already used by Faster R-CNN, by the classification and the regression branches, respectively. $\mathcal{L}_{mask}$ is the averaged cross-entropy functions of keypoints detection head.

Chapter 5

Experimental Protocol

This chapter presents the experimental protocol: in Sec. 5.1 the dataset split strategy and the cross-validation methods are described. In Sec. 5.2 and Sec. 5.3, the framework training settings and the performance metrics are presented, respectively.

5.1 Leave-One-infant-Out Cross Validation

A Leave-One-infant-Out Cross-Validation (LOOCV) approach was used to train the framework, evaluating the performance on the different infants' videos. In the LOOCV the data of one infant's video are left out of training set and used for test set, also another one for validation set, and all the remaining data are used for training. A total of nineteen-fold cross-validation is performed. The occurrence of wrong positioning of the camera led to the exclusion of two folds. Due to the presence of multiple videos recording the same infant, particular attention was paid to remove from the training set the videos of the same baby selected for the test set. In view of that, the training dataset results in 5600 frames, 4800 frames, or 4600 frames, according to the number of videos removed. The Table 5.1 reports the number of frames considered for the training sets of each fold.

Table 5.1: *Dataset split: the number of frames used for training set, validation set and test set are indicated, with the relative fold number.*

Fold number	Training	Validation	Test
1-10	5600	200	200
11,12,14,15,17	5400	200	200
13,16	5200	200	200

5.2 Training settings

Firstly, a data augmentation was performed applying random brightness, random contrast and size alterations to the images. This types of data augmentation allow to face the limited dimension of the available dataset, increasing robustness of the prediction in presence of the challenging conditions previously mentioned in Sec. 4.4.1 (Fig. 4.1). At the beginning, the weight of a pre-trained model were loaded, with two frozen layers by default. The number of iterations for training was set to 3000, with a batch size consisting of 2 images per GPUs (a single GPU was used for this work). The SGD was selected as optimizer and the LR, starting from a base value of 0.01, scheduler to decrease by a factor 0.2 every 200 iterations in case of a plateau of the validation loss. The confidence threshold for bounding box is set to 0.75. The analysis was performed on a NVIDIA Tesla K80 GPU.

5.3 Performance metrics

For infant detection, the Intersection Over Union *(IoU)* was used to quantify the amount of overlapping area between the true and the predicted bounding box areas. It is defined as the ratio between the area of intersection and the area of union of the two bounding boxes (Fig. 5.1):

$$IoU = \frac{A \cap B}{A \cup B} \tag{5.1}$$

Figure 5.1: *Example of computing IoU at different bounding box predictions.*

IoU is ranging from 0 (completely wrong prediction) to 1 (perfect prediction). An IoU threshold value α is chosen and used to determine if the detection is valid or not. If $IoU \geq \alpha$, the detection is correct, and it is called *true positive* (TP). Otherwise, for $IoU < \alpha$, the detection is not valid and it is defined *false positive* (FP). *False negative* (FN) is also defined a ground truth missed by the model.

The IoU was used to compute the Average Precision *(AP)*. The AP represents the area under the curve of the *precision-recall* curve, where the precision and the recall are defined as follow:

$$Precision = \frac{TP}{\text{all detections}} \tag{5.2}$$

$$Recall = \frac{TP}{\text{all ground truths}} \tag{5.3}$$

The precision is the ability to identify relevant objects only. It is the proportion of TP detections. The recall measures the ability to find all relevant cases, that is the proportion of TP detected among all ground truths. In order to calculate AP, the *precision-recall* curve is calculated at varying IoU thresholds, and the average is taken for each class across all of the *IoUs*.

For keypoint detection task, COCO metric [37] was used to evaluate the framework performance. The AP for keypoint detection was computed considering the Object Keypoint Similarity *(OKS)* [37], a similarity measure analogous

of IoU for bounding box. The detection is defined positive or negative according to OKS distance threshold.

$$OKS = \frac{\sum_i \exp(\frac{-d_i^2}{2s^2k_i^2})\delta(v_i > 0)}{\sum_i \delta(v_i > 0)} \tag{5.4}$$

For each object, ground truth keypoints have the form [x1,y1,v1,...,xk,yk,vk], where x,y are the keypoint locations and v is a visibility flag defined as v=0: not labeled, v=1: labeled but not visible, and v=2: labeled and visible [37]. The d_i are the Euclidean distances between each corresponding ground truth and detected keypoint and the v_i are the visibility flags of the ground truth. To compute OKS, d_i is passed through an unnormalized Guassian with standard deviation sk_i, where s is the object scale (i.e., the square root of the object segment area) and k_i is a per-keypoint constant that controls falloff. For each keypoint this yields a keypoint similarity that ranges between 0 and 1. These similarities are averaged over all labeled keypoints (keypoints for which $v_i > 0$). Perfect predictions will have $OKS = 1$ and predictions for which all keypoints are off by more than a few standard deviations si will have $OKS \approx 0$.

The OKS is analogous to the IoU. Given the OKS, we can compute AP just as the IoU allows to compute AP metrics for bounding box detection.

The the $AP^{IoU=[.50:.95]}$ was computed in the present work. It represents the average AP computed by varying the OKS threshold from 0.50 to 0.95 increasing it by 0.05 for each complete evaluation of the dataset.

To evaluate the limb-pose detection performance, the Root Mean Square Error *(RMSE)* [pixels] for each infants' limb was computed. It is defined as:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{5.5}$$

representing the distance between the ground truth a(y_i) and the predicted (y_i) limb joint connections (Fig. 5.2).

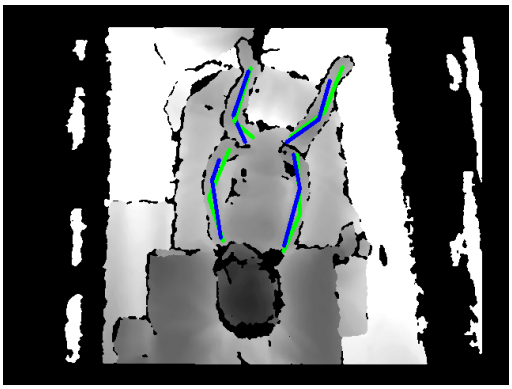

Figure 5.2: *Example of limb-pose prediction (blue lines) and ground truth (green lines). The RMSE [pixels] between the blue and green lines was measured for each limb.*

5.4 Comparison with other architectures

The present work was compared against [9], which is the closest work with respect to this one. We use the same training settings in terms of dataset split and computational resources for a fair comparison.

Chapter 6

Results

This chapter shows the results achieved for joint detection and limb-pose estimation with the Mask R-CNN.

Joint detection task was evaluated by AP metrics. In Fig. 6.1, the graph shows the $AP^{IoU=[.50:.95]}$, for each cross-validation fold. Two folds of the 19 were excluded from the results since affected by the wrong positioning of the camera. Mean AP achieved 86% ± 17%. Detection time was on average 0.7 s per image. The joint detection allowed to show the infant joint spatial displacement over time (Fig 6.2).

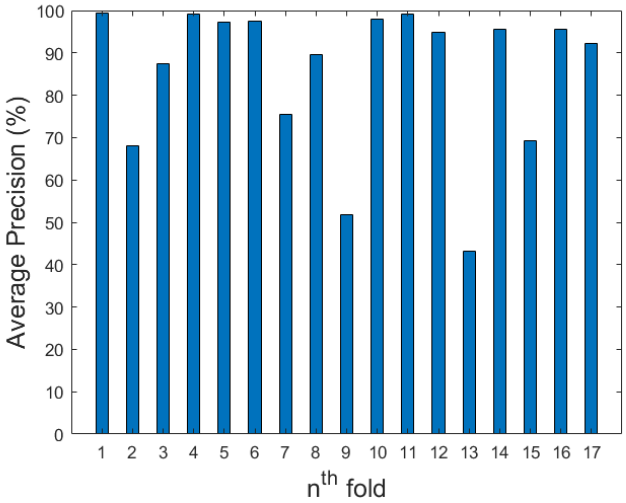

Figure 6.1: *$AP^{IoU=[.50:.95]}$ (%) values resulted from each LOOCV fold.*

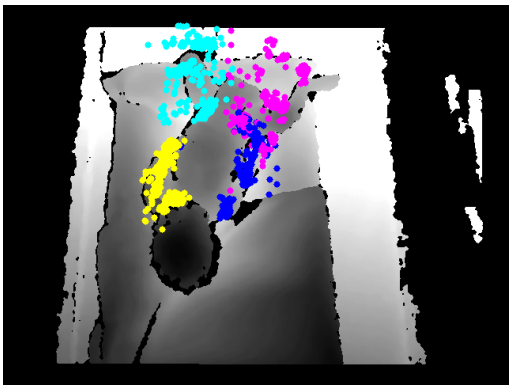

Figure 6.2: *Temporal evolution of joint position for each infants' limb. Each color refers to a different limb.*

The median *RMSE* for limb-pose estimation are showed in Table 6.1. Boxplots for *RMSE* are showed in Figure 6.3.

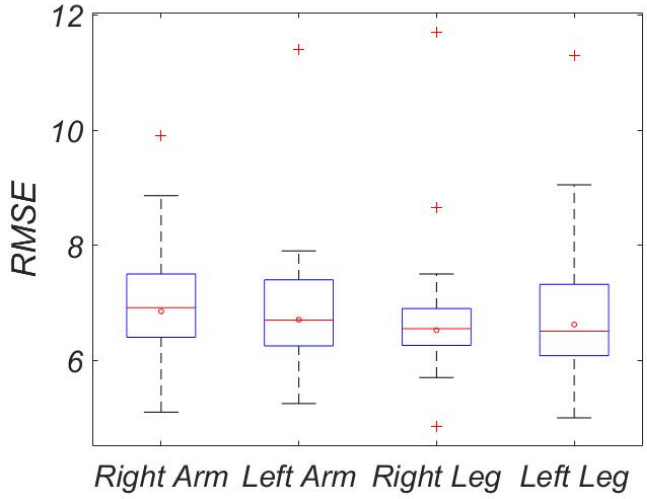

Figure 6.3: *Boxplots of the RMSE computed for the four limbs separately. Red line indicates the median, red circles are the mean values and the red crosses are the outliers.*

Qualitative results are showed in Fig. 6.4 also on challenging frames with more homogeneous pixel intensity.

The results of the [9] detection CNN training are showed in Fig. 6.5 for two folds.

Table 6.1: Limb-pose estimation performance in terms of median *RMSE* computed with respect to ground-truth pose, reported separately for each limb. IQR is reported in brackets.

Median RMSE			
Right arm	Left arm	Right leg	Left leg
6.8 (1.1)	6.7 (1.2)	6.5 (0.6)	6.5 (1.2)

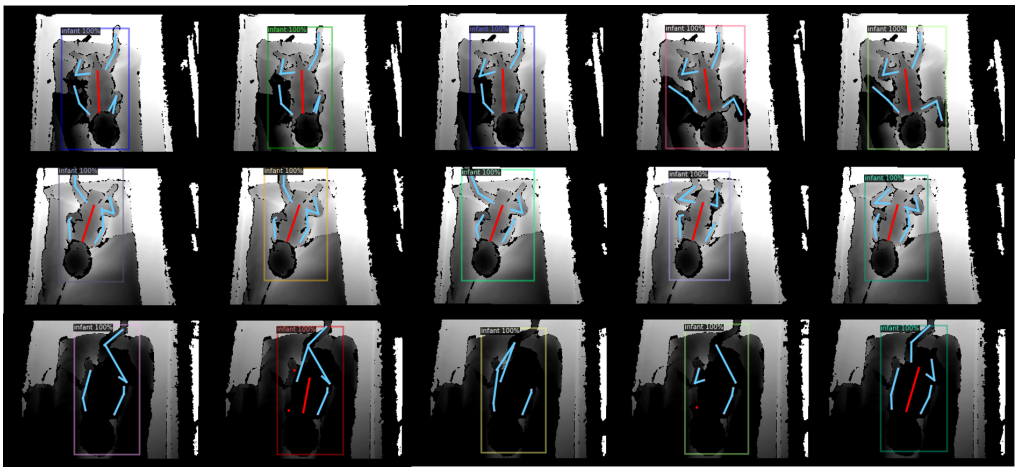

Figure 6.4: *Sample pose-estimation results. Sample of limb-pose (blu lines) and bounding box prediction with classification confidence (%) and infant body axes (red lines). Correct predictions are showed on first and second row of images, lower performance on challenging cases are showed in the third raw of images.*

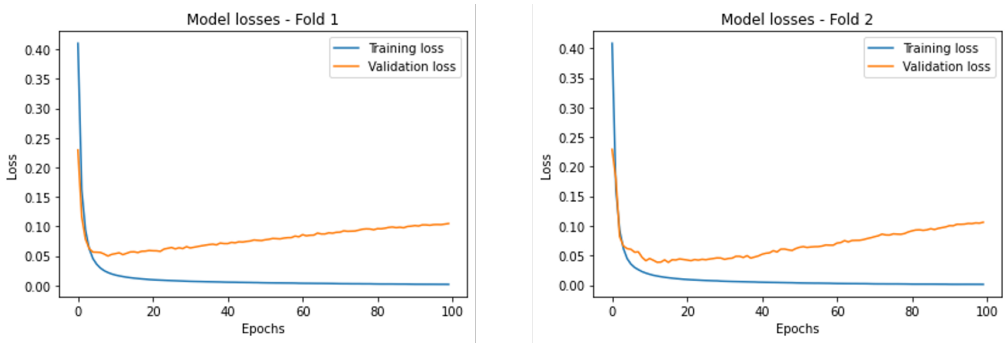

Figure 6.5: *Training and validation losses for [9] detection CNN on fold 1 (left panel), and fold 2 (right panel).*

Chapter 7

Discussion

Monitoring preterm infants' limb movements proven to be crucial to assess infant's health and for an early recognition of cognitive or motor disorders. Nowadays, this monitoring relies on qualitative and sporadic observation by trained clinicians at the crib side in NICUs.

In the literature, researchers have proposed different solutions to automate GMA tool, such as exploiting contact sensors. Concerning this approach, it entails infants' stress, discomfort, and pain, as well as hindering the actual clinical practice of operators dealing with infants.

More reliable and unobtrusive alternatives involve marker-less solutions based on video analysis, growing, in recent years, the interest in pose estimation approaches, too. In the previous work, Moccia et al. [9, 8] proposed an innovative approach for limb-pose estimation from spatio-temporal features extracted from depth videos acquired in NICUs. However, the introduction of temporal information and the use of two different CNNs (one for joint detection and one another for joint regression) results in a long and complex framework, increasing computational costs.

Based on the Moccia et al. approach [9, 8], the present thesis aims to develop an end-to-end deep learning framework for joint detection and limb-pose estimation from depth images of preterm infants.

The Mask R-CNN was chosen and trained for this purpose. The dataset

comprised 6000 frames from 30 videos recording the movements of 19 preterm infants. It presented several challenges, such as varying illumination levels, self- or external occlusions, and a different number of joints in the camera field of view. The framework was validated through a LOOCV, for a total of 19 trainings.

The proposed framework achieved satisfactory performance for both joint detection and limb pose estimation. Concerning joint detection, a mean $AP = 86\% \pm 17\%$ was achieved.

The occurrence of wrong positioning of the camera led to the exclusion of two folds from the results, and has affected the performance of folds 9 and 13. In folds 2, 7, 15 , the framework under-performed in detecting one joint with respect to the others. This was indicated by a prediction confidence score almost ten order of magnitude lower than correctly detected keypoints. Observing Fig. 6.3, achieved similar results for the pose estimation of the four limbs. Barring the outliers, the best performance is characterized by a RMSE equal to 5 pixels achieved for right arm, while the worse performance corresponds to 9 pixels, for left leg. The median RMSE among the four limbs is 6.6 pixels with IQR equal or lower than 1.2 pixel. The overall methodology required s per image, hence being compatible with real-time infants' monitoring.

Fig. 6.5 reveals the overfitting of the detection CNN of [9], after few epochs. The present framework uses pre-trained weights on COCO dataset [37], while [9] does not rely on transfer learning. Moreover, differently from [9], where the keypoints detection is performed over the entire input images, Mask-RCNN [35] allows for joint detection within the more limited infant bounding box area. These observations justify the higher performance of the present work, compared to [9].

The present approach, despite the limited dataset dimensions, overcame some of the literature drawbacks. Hence, it allowed to directly estimate limb-specific pose with an end-to-end framework, being computationally efficient and easily deployable in clinical environment, representing an important step toward an on-the-edge computation for home monitoring.

In future works, the results of the home monitoring will be integrated into

the SINC system to allow clinicians to assess the evolution of the infants' health status and timely intervene. Moreover, the network could be improved for detection of both infant and operator and to distinguish the infants' behaviour in response to tactile stimulation from parents or clinicians in NICU.

Chapter 8

Conclusion

This thesis presents a deep learning framework for joint detection and limb-pose estimation from depth images of preterm infants in NICUs. The work is based on the approach of [9], improving results in terms of pipeline complexity and computational costs. The training of Mask R-CNN is revealed as highly performing for that purpose, obtaining encouraging results. Differently from what proposed in the state-of-the-art, the approach presented in this thesis is: (i) an end-to-end framework (ii) completely automatic (iii) non-invasive (iv) reliable (v) limb specific. This may be a step toward new research scenarios aimed at developing embedded monitoring solutions for on-the-edge computation.

List of Figures

List of Tables

Bibliography

[1] A. Polito, S. Piga, P. E. Cogo, C. Corchia, V. Carnielli, M. D. Fr'e, D. D. Lallo, I. Favia, L. Gagliardi, and F. M. et al.:, "Increased morbidity and mortality in very preterm/vlbw infants with congenital heart disease." *Intensive Care Medicine*, 2013.

[2] H. C. Glass, Y. Li, M. Gardner, A. J. Barkovich, M. C. E. Novak, I., and E. E. Rogers, "Early identification of cerebral palsy using neonatal mri and general movements assessment in a cohort of high-risk term neonates," *Pediatric neurology*, vol. 118, pp. 20–25, 2021.

[3] A. te Velde, C. Morgan, I. Novak, E. Tantsis, and N. Badawi, "Early diagnosis and classification of cerebral palsy: An historical perspective and barriers to an early diagnosis," *Journal of Clinical Medicine*, vol. 8, no. 10, p. 1599, 2019.

[4] H. A. Parikh, N. A. and M. Altaye, "Early detection of cerebral palsy using sensorimotor tract biomarkers in very preterm infants," *Pediatric neurology*, vol. 98, pp. 53–60, 2019.

[5] F. Ferrari, G. Cioni, and H. Prechtl, "Qualitative changes of general movements in preterm infants with brain lesions," *Early Human Development*, vol. 23, no. 3, pp. 193–231, 1990.

[6] I. Zuzarte, A. H. Gee, D. Sternad, and D. Paydarfar, "Automated movement detection reveals features of maturation in preterm infants," *42nd Annual*

International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 600–603, 2020.

[7] K. Raghuram, S. Orlandi, V. Shah, T. Chau, M. Luther, R. Banihani, and P. Church, "Automated movement analysis to predict motor impairment in preterm infants: a retrospective study," *Journal of Perinatology*, vol. 39, pp. 1–8, 10 2019.

[8] S. Moccia, L. Migliorelli, V. Carnielli, , and E. Frontoni, "Preterm infants' pose estimation with spatio-temporal features," *IEEE*.

[9] S. Moccia, L. Migliorelli, R. Pietrini, and E. Frontoni, "Preterm infants' limb-pose estimation from depth images using convolutional neural networks," in *2019 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE, 2019, pp. 1–7.

[10] C. Marcroft, A. Khan, N. D. Embleton, M. Trenell, and T. Plötz, "Movement recognition technology as a method of assessing spontaneous general movements in high risk infants," *Frontiers in neurology*, vol. 5, 2015.

[11] N. Hesse, S. Pujades, M. J. Black, M. Arens, U. G. Hofmann, and A. S. Schroeder, "Learning and tracking the 3d body shape of freely moving infants from rgb-d sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, pp. 2540–2551, 2020.

[12] K. E. Fry, Y. Chen, and A. Howard, "Detection of infant motor activity during spontaneous kicking movements for term and preterm infants using inertial sensors," *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 5767–5770, 2018.

[13] M. Airaksinen, O. Räsänen, E. Ilen, T. Häyrinen, A. Kivi, V. Marchi, A. Gallen, S. Blom, A. Varhe, N. Kaartinen, L. Haataja, and S. Vanhatalo, "Automatic posture and movement tracking of infants with wearable movement sensors," *Scientific Reports*, vol. 10, no. 1, 2020.

[14] C. B. Redd, L. A. Barber, R. N. Boyd, M. Varnfield, and M. K. Karunanithi, "Development of a wearable sensor network for quantification of infant general movements for the diagnosis of cerebral palsy," in *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2019, pp. 7134–7139.

[15] S. Miyagishima, T. Asaka, K. Kamatsuka, N. Kozuka, M. Kobayashi, L. Igarashi, T. Hori, and H. Tsutsumi, "Spontaneous movements of preterm infants is associated with outcome of gross motor development," *Brain amp; development*, vol. 40, no. 8, p. 627—633, September 2018.

[16] N. Hesse, C. Bodensteiner, M. Arens, U. Hofmann, R. Weinberger, and A. Schroeder, *Computer Vision for Medical Infant Motion Analysis: State of the Art and RGB-D Data Set: Munich, Germany, September 8-14, 2018, Proceedings, Part VI*, 01 2019, pp. 32–49.

[17] I. Doroniewicz, D. J. Ledwoń, A. Affanasowicz, K. Kieszczyńska, D. Latos, M. Matyja, A. W. Mitas, and A. Myśliwiec, "Writhing movement detection in newborns on the second and third day of life using pose-based feature machine learning classification," *Sensors*, vol. 20, no. 21, 2020.

[18] E. A. Ihlen, R. Støen, L. Boswell, R.-A. de Regnier, T. Fjørtoft, D. Gaebler-Spira, C. Labori, M. C. Loennecken, M. E. Msall, U. I. Möinichen *et al.*, "Machine learning of infant spontaneous movements for the early prediction of cerebral palsy: A multi-site cohort study," *Journal of clinical medicine*, vol. 9, no. 1, p. 5, 2020.

[19] T. Tsuji, S. Nakashima, H. Hayashi, Z. Soh, A. Furui, T. Shibanoki, K. Shima, and K. Shimatani, "Markerless measurement and evaluation of general movements in infants," *Scientific Reports*, vol. 10, 2020.

[20] L. Adde, J. L. Helbostad, A. R. Jensenius, G. Taraldsen, and R. Støen, "Using computer-based video analysis in the study of fidgety movements," *Early human development*, vol. 85, no. 9, pp. 541–547, 2009.

[21] C. Tacchino, M. Impagliazzo, E. Maggi, M. Bertamino, I. Blanchi, F. Campone, P. Durand, M. Fato, P. Giannoni, R. Iandolo, M. Izzo, P. Morasso, P. Moretti, L. Ramenghi, K. Shima, K. Shimatani, T. Tsuji, S. Uccella, N. Zanardi, and M. Casadio, "Spontaneous movements in the newborns: a tool of quantitative video analysis of preterm babies," *Computer Methods and Programs in Biomedicine*, vol. 199, p. 105838, 2020.

[22] K. D. McCay, E. S. Ho, H. P. Shum, G. Fehringer, C. Marcroft, and N. D. Embleton, "Abnormal infant movements classification with deep learning on pose-based features," *IEEE Access*, vol. 8, pp. 51 582–51 592, 2020.

[23] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," 2017.

[24] M. Li, F. Wei, Y. Li, S. Zhang, and G. Xu, "Three-dimensional pose estimation of infants lying supine using data from a kinect sensor with low training cost," *IEEE Sensors Journal*, vol. 21, no. 5, pp. 6904–6913, 2021.

[25] V. Marchi, A. Hakala, A. Knight, F. D'Acunto, M. L. Scattoni, A. Guzzetta, and S. Vanhatalo, "Automated pose estimation captures key aspects of general movements at 8-17 weeks from conventional videos," *Acta Paediatrica*, vol. 108, 2019.

[26] N. Hesse, A. S. Schröder, W. Müller-Felber, C. Bodensteiner, M. Arens, and U. G. Hofmann, "Body pose estimation in depth images for infant motion analysis," in *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2017, pp. 1909–1912.

[27] D. JE and D. JM, *Artificial neural networks— opening the black box.* Cancer, 2001.

[28] R. Nayak, L. Jain, and B. Ting, "Artificial neural networks in biomedical engineering: A review," in *Computational Mechanics–New Frontiers for the New Millennium*, S. Valliappan and N. Khalili, Eds. Oxford: Elsevier, 2001, pp. 887–892.

[29] R. Gonzalez and R. Woods, *Digital Image Processing*. Pearson, 2018. [Online]. Available: https://books.google.it/books?id=0F05vgAACAAJ

[30] F. Rosenblatt:, "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological review*, vol. vol. 65, p. p.386, 1958.

[31] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 11 2013.

[32] R. Girshick, "Fast r-cnn," 2015.

[33] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[35] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[36] J. Brooks, "COCO Annotator," https://github.com/jsbroks/coco-annotator/, 2019.

[37] X. Chen, H. Fang, T. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick, "Microsoft coco captions: Data collection and evaluation server," *arXiv:1504.00325*, 2015.