POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's Degree Thesis

Xpective Dataset

Towards Robust Pose Estimation with Radar Sensing

Supervisors

Prof. Barbara CAPUTO

Doct. Dario FONTANEL

Student

Niccolò CAVAGNERO

October 2021

Abstract

In the last years, Computer Vision has achieved astonishing results, especially thanks to the development of Convolutional Neural Networks, in tasks such as Object Detection, Semantic Segmentation, Pose Estimation. Despite this success, the majority of RGB-based systems are still incapable of performing accurate predictions in adverse situations such as lack of light or in presence of occlusions, dust, fog and other unfavorable environmental conditions. On the other hand, even if radar-based systems are capable of overcoming some of these issues, their usage is still limited by low resolutions and difficulties in capturing the shapes of objects. In order to fill the gap between RGB and radar modalities, this work involves the acquisition of an heterogeneous dataset containing synchronized radar samples and the ground truth for keypoint 3D positions acquired by means of RGB sensors. Towards this objective, samples of different performed actions have been acquired from different people, in order to get as much variety as possible. At the time of writing this Thesis, in the literature there is not yet a public dataset available with these characteristics. After the collection of the dataset, we use it as a benchmark to evaluate state-of-art methods. In addition, we also propose a custom architecture able to outperform these methods by a large margin on the acquired dataset.

Acknowledgements

A special thanks to Dario Fontanel and Giuseppe Averta, for the precious advises in the development of this thesis and to professor Caputo for giving me the opportunity to work on this project. I also thank Inxpect for the funding received and for hosting me and to Ugo Bertacchini who helped me in the development of the Motion Capture System.

Finally, I dedicate this work to my friends and to my family for they are the reason I achieved this goal.

Table of Contents

List of Tables

List of Figures

Acronyms

ACC

Adaptive Cruise Control

AEB

Automatic Emergency Braking

AG

Agglomerative Clustering

AoA

Angle of Arrival

BCE

Binary Cross Entropy

BN

Batch Normalization

CNN

Convolutional Neural Network

DL

Deep Learning

ELLIS

European Laboratory for Learning and Intelligent Systems

ELU

Exponential Linear Unit

FC

Fully Connected

FFT

Fast Fourier transform

FMWC

Frequency-Modulated Continuous Wave

GELU

Gaussian Error Linear Unit

GNN

Graph Neural Network

IF

Intermediate Frequency

IIT

Italian Institute of Technology

IoT

Internet of Things

LR

Learning Rate

MIMO

Multiple-input Multiple-output

MLP

Multi-layer Perceptron

MSE

Mean Squared Error

mmWave

Millimeter-wave

NLP

Natural Language Processing

NN

Neural Network

NTP

Network Time Protocol

PReLU

Parametric Rectified Linear Unit

ReLU

Rectified Linear Unit

RF

Radio Frequency

RNN

Recurrent Neural Network

RReLU

Randomized Rectified Linear Unit

SGD

Stochastic Gradient Descent

UDP

User Datagram Protocol

Chapter 1

Introduction

The recent development of *Deep Learning* represented a breakthrough in many technological fields, which span from Natural Language Processing, up to Computer Vision and Robotics applications. Nowadays, Deep Learning systems are ubiquitous in our daily life and the performances in different tasks are already beyond the human level [1, 2, 3, 4, 5]. In particular, the availability of various large-scale well annotated public datasets was a key enabling factor in the achievement of astonishing results in *Pose Estimation*, which is a deeply studied Computer Vision tasks [6, 7, 8, 9] whose objective is the localization in 2D or 3D of various keypoints of the human body: head, shoulders, wrists, hips, knees and so on. Its applications range from Computer Graphics to Robot-human Interaction, industrial safety, autonomous driving and so on.

However, Deep Learning algorithms still suffer of a number of data related limitations, such as the corruption of the data samples. For example, most of the state-of-art vision-based approaches suffer of a dramatic reduction in performances when data are corrupted by poor-light conditions. This issue can cause these systems to be unsuitable for an actual application, especially in safety-critical scenarios such as industrial safety or autonomous driving: an autonomous car would not see a person in the fog, a surveillance system would not work in absence of light or a safety system may not detect a person in case of debris or smoke.

In order to enhance the robustness of the sensing systems, various sensors measuring different quantities may be adopted together, in a so called *Sensor Fusion* fashion, by merging their information from different domains or by exploiting the information coming from the right kind of sensor in the right

situation.

In this Thesis, we explored the applicability of radars for *Pose Estimation* purposes through the acquisition of a combined RGB and radar dataset. In particular, we adopted mmWave *Frequency-Modulated Carrier Wave* (FMCW) radars, which are popular for industrial and autonomous driving applications thanks to their limited size [10]. Indeed, radar technology is insensitive to weather conditions and may be exploited to enhance the robustness of Computer Vision systems by providing environmental perception in all kinds of situations.

As previously stated, a key enabling factor in the development of Pose Estimation, and Deep Learning in general, was the availability of large-scale public datasets, but at the time of writing this Thesis there is not a single public dataset for radar-based Pose Estimation. For this reason, we assembled a Motion Capture System and collected the first publicly available dataset for such purpose, with the aim of encouraging and facilitating the research community in the study and development of Pose Estimation and related research. After an evaluation of state-of-art models on our dataset, we also propose an ad hoc architecture which consistently outperforms these methods by a large margin.

Part I

Literature

Chapter 2

Prior

In the Internet of Things (IoT) era sensors are ubiquitous in our daily life and the trend is continuous growth, connecting devices and enabling the collection and exploitation of an always increasing amount of data. Sensors can be classified into two main categories:

- active sensors, if they employ their own signal to produce an output;
- passive sensors, if they rely on an external signal, usually called excitation signal, to generate their output.

Vision-based systems rely on passive sensors which require light as excitation signal and they are therefore ineffective in poor lighting conditions, adverse weather conditions or when the scene/target is occluded, i.e. when the excitation signal is absent or disturbed. Despite the recent astonishing results achieved in the last years in Computer Vision, this drawback is one of the reasons for which in safety critical scenarios, such as autonomous driving [10, 11], RGB sensors are deployed together with active sensors, such as Radar and Lidar, which do not rely on an external signal to operate, illuminating the scene by their own and thus providing environmental perception in all kinds of weather conditions [11].

A possible application would be aggregating and exploiting the information coming from different kinds of sensors and domains in a *Sensor Fusion* fashion. Sensor Fusion can be exploited in order to enhance the robustness and accuracy of predictions and measurements. For example it could increase the reliability to scene lighting and weather changes in detection systems thanks to an obvious statistical advantage and a better observability.

Figure 2.1: Radars produced by Inxpect are robust to light changes, debris, smoke and liquids. The image is taken from Inxpect official website.

However, it presents different drawbacks for which the actual implementation of this technique may be unfeasible, such as the increased energy and the memory needed for collecting, processing and storing the data increase or the increasing complexity of the sensing system.

2.1 Radar

Radar technology has been studied since the end of XIX century, but it had its major development during the World War II when the majority of the developed nations was secretly studying radio frequencies for detection purposes. The United Kingdom was the first Nation to deploy a radar-based defense system to detect incoming German attacks, later sharing the knowledge with the United States. This system was a crucial factor in the Britain Battle enabling the limited allied forces to counter the increasingly large Luftwaffe raids. Since then this technology has been employed for multiple different applications, both military and civilian, from air traffic control to industrial safety.

In this work, we focus on Frequency-Modulated Continuous Wave (FMCW) Millimeter-wave (mmWave) radars, which are popular in both industrial and automotive segments thanks to the Multiple-input Multiple-output (MIMO) technology which allows the synthesis of large virtual antenna arrays from a relatively small number of transmitting and receiving antennas [10].

2.1.1 Radar Signal Processing

An FMCW radar is essentially a time-of-flight sensor, usually consisting in an array with multiple TX and RX antennas, which resolves the target

distance, angle of arrival (AoA) and velocity by analyzing the reflections of its signal. The base unit of an FMCW radar is the so called chirp, i.e. a complex sinusoidal wave whose frequency increases linearly with time: the signal starts with a certain carrier frequency f_c and ends at time T with a final frequency $f_c + B$ where B is the chirp bandwidth. Namely,

$$f_T(t) = f_c + (B/T)t \qquad t \in [0, T]$$

The time during a chirp is usually referred to as fast time, while the time across multiple periods or chirps is referred to as slow time. Fig. 2.2 and Fig. 2.3 show a chirp in the A-t and in the f-t planes.

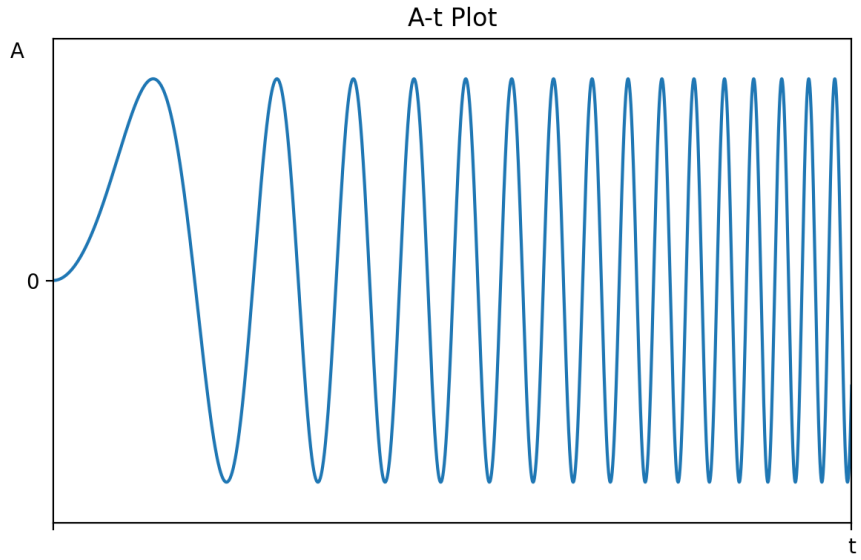

Figure 2.2: A chirp in the amplitude-time plane.

From a functional point of view, the radar is made up of different components which are schematized in Fig. 2.4:

- a synthesizer;
- transmitting antennas;
- receiving antennas;
- a mixer;
- a Band-pass filter;
- a final ADC.

The synthesizer generates the chirp signal which is transmitted by the set of TX antennas and is then received, delayed and attenuated after the reflection

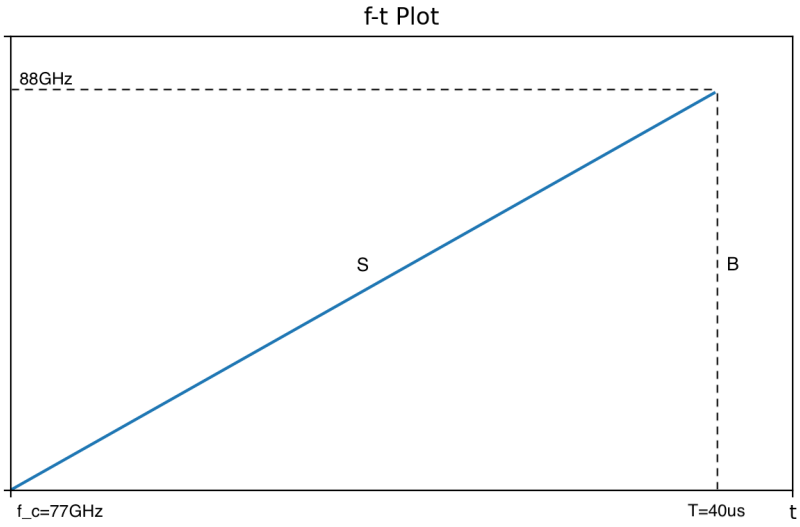

Figure 2.3: A chirp in the frequency-time plane.

on the target, by the set of RX antennas. Transmitted and received signals are then combined by the mixer, generating the Intermediate Frequency (IF) signal, also called Beat signal. The IF signal posses two interesting symmetric properties which can be exploited in the signal processing pipeline:

- the instantaneous frequency of the IF signal is the instantaneous difference of transmitted and received signals;
- the starting phase of the IF signal is the difference between the starting phases of transmitted and received signals.

Formally, from a mathematical point of view, these two properties can be formulated as:

$$\begin{aligned} x_{TX} &= sin(\omega_1 t + \phi_1) \\ x_{RX} &= sin(\omega_2 t + \phi_2) \\ x_{IF} &= sin((\omega_1 - \omega_2)t + (\phi_1 - \phi_2)) \end{aligned}$$

Therefore, the IF signal generated by a single object in front of the radar has a fixed frequency $S\tau$, where S is the slope of the signal in the frequency-time plane and $\tau = 2d/c$ is the round trip delay, i.e. two times the target distance divided by the speed of light. The IF signal is later passed through the Band-pass Filter in order to select only the desired frequencies of interest.

Finally the signal is converted from the analog domain to the digital one by means of the ADC for further preprocessing.

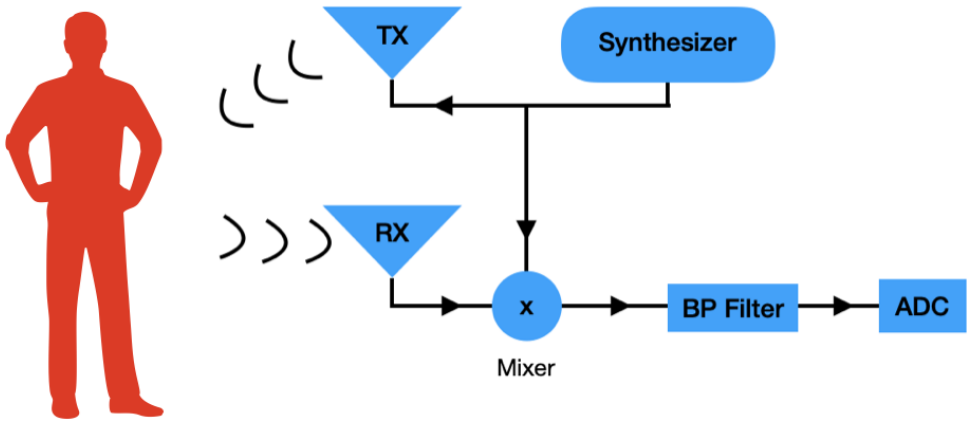

Figure 2.4: Radar scheme.

The core preprocessing technique for signals in general and also for RF data is the famous *Fourier transform*, specifically the fast Fourier transform (FFT) and short-time Fourier transform, which transforms a signal from the time domain to the frequency one. Formally, the mathematical formulation for an integrable function $f : \mathbb{R} \to \mathbb{C}$ is the following one:

$$\hat{f}(\xi) = \int_{-\infty}^{+\infty} f(t) e^{-2\pi i t \xi} \, dt$$

The raw data extracted from the digital signal are organized as a cube with dimensions fast time, slow time and channel. By applying a Fourier Transform along the fast time one can identify the target range. If we apply a second Fourier Transform along the slow time one can retrieve the Doppler Frequency and the velocity of the target while by applying it along the channel dimension one can retrieve the azimuth. The application of this two consecutive FFTs along two different dimensions of the raw data cube is equivalent to a single so called 2D-FFT. It is worthy to report that other techniques for azimuth estimation, such as the Capon method, are available.

2.1.2 Radar Signal Characteristics

mmWaves are able to traverse walls and therefore can detect a person even if it is totally occluded by an obstacle of this kind. However, while the

human body acts as a scatterer with visible light, mmWaves are specular to the human body, i.e. the wavelength is comparable to the roughness of the irradiated surface, which means that human bodies act as reflectors: this causes the reflected signals to not be always directed towards the RX antenna array, resulting in failed detections [12, 13]. While this is not a problem for the simple detection of a person, this issue can be a serious problem for a Pose Estimation task. In fact, while there are large body parts, such as the thorax, which are likely to reflect the signal in the right direction back to the radar, this may not be true for the limbs whose surface orientation may cause the signal to be reflected away making an accurate prediction of those keypoints a really challenging problem. For this reason, the state-of-art in radar-based Pose Estimation takes as input multiple consecutive RF frames exploiting the time dimension by means of 3D Convolutional Neural Networks (CNNs) [14]. It must also be noted not only that commercial radars have a much lower spatial resolution with respect to RGB sensors, making the information extraction process much more challenging, but also the fact that RGB works with just 2 dimensions, while radar data is intrinsically 3D or 4D, taking into account both the space and the time. This fact results in a more difficult handling and processing phase which is also likely to require significantly larger computational and storage capabilities. Moreover, the signal may be disturbed by the so-called multipath, a phenomenon for which non-direct reflections are received by the antennas causing a degradation of signal with detrimental effects for the sensing pipeline. Fig. 2.5 shows how this phenomenon can negatively impact on a sensing system by causing ghost targets to be detected.

2.2 Supervised Learning

Supervised Learning is a Machine Learning branch where algorithms infer knowledge from annotated data, usually by minimizing a loss function. Formally, we have a set of training data $\mathcal{D}_{train} = \{(X_1, Y_1), ..., (X_n, Y_n)\}$, a set of test data $\mathcal{D}_{test} = \{(X_{n+1}, Y_{n+1}), ..., (X_m, Y_m)\}$ and we aim to build a model which performs well on unseen samples by minimizing a loss function $\mathcal{L}(x, y, f(x))$ which measures the performance of the model $f(\cdot)$ on a certain sample x.

Before the *Deep Learning* (DL) era, shallow algorithms such as *Support Vector Machine* or *Decision Trees* where employed for prediction purposes

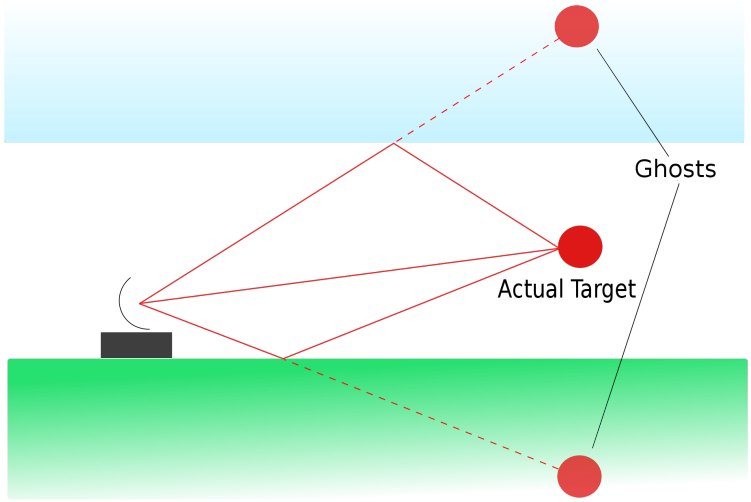

Figure 2.5: Multipath echoes from an actual target which generate 'ghost' targets. The image is taken from Wikipedia.

after a feature extraction step, which was manually performed. For years these methods shaded *Neural Networks* (NNs), which are both computationally and data intensive and were almost impossible to train and deploy. As the technology progressed and the amount of available data increased, there was a paradigm shift to hierarchical deep algorithms capable of automatically performing the feature extraction process, the Neural Networks. This kind of deep algorithms slowly started to outperform shallow ones in famous benchmark datasets such as MNIST [15], CIFAR [16], ImageNet [17]. Neural Networks, in their various forms, nowadays represent the state-of-art in a large number of fields, such as Computer Vision, Natural Language Processing or Robotics.

2.2.1 Neural Networks

The classic architecture for a Neural Network is constituted by a stacking of multiple non-linear Perceptrons [18] where each neuron of a layer is connected to all the neurons in the following one and each layer is generally followed by a non-linear activation function: this is the so called Multi-layer Perceptron (MLP). It has been shown that Neural Networks are universal function approximators able to deal with highly non-linear situations by

simply employing a finite linear combinations of Perceptron layers and non-linear activations (*sigmoids* at the time) [19]. Fig. 2.6 shows a scheme of a simple and small MLP architecture with only two hidden layers and a single output neuron.

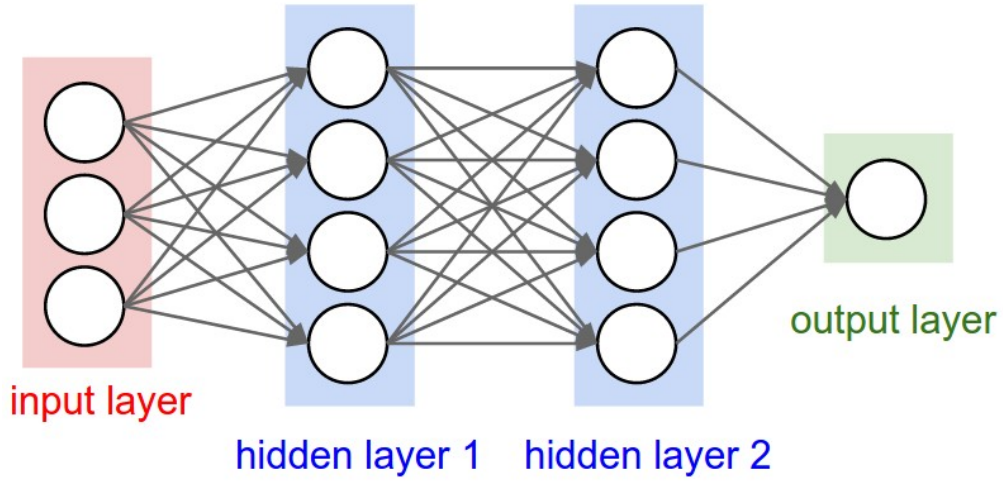

Figure 2.6: MLP architecture scheme. The image is taken from [20].

Namely, the mathematical formulation of the output vector of each layer is given by the following formula:

$$y = \sigma(\sum_{i=1}^{m} w_i x_i + b) \tag{2.1}$$

where $x \in \mathbb{R}^m$ is the input vector, $w \in \mathbb{R}^m$ is the weight vector, b is the bias and σ is the activation function providing non-linearity to the model, which would be otherwise equivalent to a 1-layer linear Perceptron. There exist various kinds of activation functions such as the *sigmoid* or *tanh* but the most famous and the most commonly used is the *ReLU* [21] activation, together with its derivatives such as Leaky ReLU, due to its improved robustness to the vanishing of the gradient [22] which results in a more efficient training and better performances. Fig. 2.7 contains various activation formulas and plots.

As said before, a Neural Network[1] is able to autonomously and automatically learn how to extract useful information and meaningful representations

[1]If properly trained with enough data.

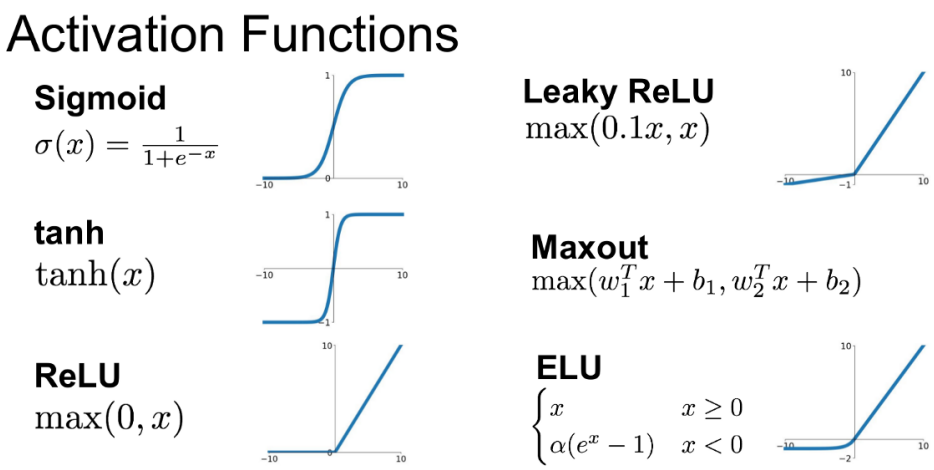

Figure 2.7: A scheme showing different kinds of activations. The image is taken from Medium.

from the data, almost completely removing the need for a manual, and laborious, feature extraction step. Moreover, these specialized features can be transferred to other networks by exploiting the so called *Transfer Learning*, either by applying the layers of a pretrained network for another different task or with a Teacher-Student approach, where a new Student network tries to mimic the behaviour of the trained Teacher one. Transfer Learning also enable a better model generalization and a faster training which also requires less annotated data, since the Neural Network is not trained from scratch but its weights already encode useful knowledge about the data domain.

Passing the time by, in addition to the classic MLP various other architectures emerged to tackle different problems: Convolutional Neural Networks (CNNs) [23, 24, 25, 26] for vision tasks, Transformers [27] initially for Natural Language Processing (NLP) and recently extended to vision [28, 29], Recurrent Neural Networks (RNNs) [30, 31] for sequential data, Graph Neural Networks (GNNs) [32] for graph-like data structures.

Training

A network is trained via *Error Backpropagation* [33] by minimizing a Loss function: after a *forward pass* in the network the error measured by the Loss is backpropagated up to the input layer updating the various weights. In fact, thanks to the chain rule it is possible to compute the various gradients

up to the input layer, since all operations that are applied on the data are differentiable. The Loss functions employed are usually convex, since this property is enough to guarantee the existence of a global minimum with obvious advantages in terms of optimization. However, there can be a certain number of local minima where the optimization algorithm can get stuck in a sub-optimal solution without reaching the real global minimum. Usually for the weight optimization the algorithm employed is Stochastic Gradient Descent (SGD) [34] or similar gradient-based algorithms such as Adagrad [35], AdaDelta [36], Adam [37] or AdamW [38].

Advanced Topics

Batch Normalization It is standard practice to apply a Batch Normalization (BN) layer between before the activation function. Despite the computational and memory overhead, normalizing the data is crucial and results in both higher accuracies and a significant speed up in the convergence of the training phase by reducing the vanishing gradient problem (up to 20 layers circa). Formally, the normalization procedure simply consists in the subtraction of the mean and the division for the standard deviation of the underlying distribution and can be formulated as:

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathbb{E}[x^{(k)}]}{\sqrt{Var[x^{(k)}]}} \tag{2.2}$$

where $\hat{x}^{(k)}$ is the normalized value of the k^{th} hidden unit, $\mathbb{E}[x^{(k)}]$ is its mean and $Var[x^{(k)}]$ is its variance. Since both the true mean and the true variance are not known, these parameters are estimated at training time and are then used to approximate the real values.

Overfitting One of the biggest drawbacks of Neural Networks, together with the need of a large amount of data and significant computational resources for the training phase, is in fact the tendency to overfit the training data which can result in a poor model generalization and therefore low performances at test time. Fig. 2.8 shows the effect of overfitting on the validation/test data.

In order to address and mitigate this issue, different techniques have been developed in addition to Transfer Learning, which still initially requires a large amount of training samples:

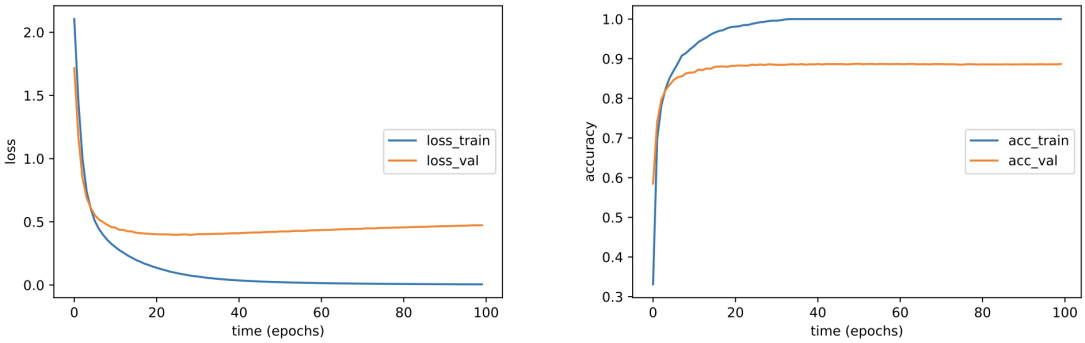

Figure 2.8: Training and validation losses and accuracies with overfitting. The image is taken from [39].

- Dropout [40], a simple procedure which consists in zeroing a fraction of the output of some layers during the forward pass in the training phase forcing the network to learn more robust features;
- Weight Decay [41], which consists in the addition of a term which penalizes the magnitude of the weights, formally $\lambda\|w\|$ where λ is a tunable hyper-parameter, to the classic loss suppressing this way "irrelevant components of the weight vector by choosing the smallest vector that solves the learning problem";
- Data Augmentation [42], which instead consists in slightly modifying the available data while maintaining the semantic information, e.g. by flipping, cropping or rotating an image or adding noise to a signal.

2.2.2 Convolutional Neural Networks

The literature about radar-based Pose Estimation exploits Convolutional Neural Networks, whose structure and functionality will now be briefly described. The history of modern CNNs started in 1989 when Yann LeCun at Bell Laboratories, leveraging the studies on Receptive Fields of the visual cortex [43], used a small backtrained convolutional architecture called LeNet [23] for handwritten digit recognition setting of the foundations of modern Computer Vision and Deep Learning in general. The second milestone is represented by AlexNet [24] which improved the overall architecture reaching the state-of-art on ImageNet [17], but it is only with the introduction of Residual Connections [26], also called Skip Connections or simply Residuals,

that we can really talk about *deep* Neural Networks. In fact, Residuals represent alternative ways for the gradient to flow back, effectively tackling the vanishing gradient problem and thus enabling the training of networks with hundreds of layers. Fig. 2.9 shows the architecture of a Residual Block.

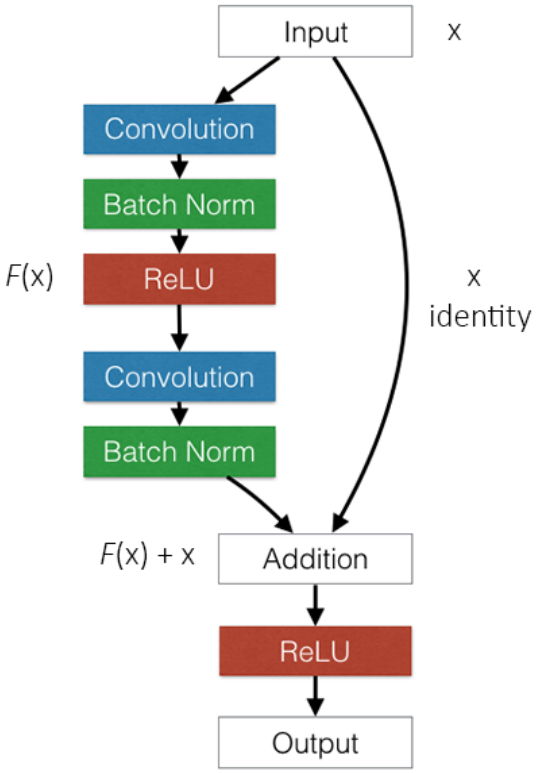

Figure 2.9: A scheme showing the structure of a Residual Block. The image is taken from [44].

The standard convolutional architecture consists in a sequence of Convolutional layer, generally followed by Batch Normalization, activations and Pooling layers with a final Fully Connected layer. It is possible to interpret the convolutional part as a feature extractor while the FC layer acts as classifier.

Convolutional Neural Networks perform particularly well in Computer Vision tasks, since their architecture intrinsically encodes the geometric priors associated with both translation invariance, given by the Convolutions, and scale separation, given by the Pooling layers [45]. However, CNNs do not take into account other types of transformation which preserve semantic

information, e.g. rotation or light changes [45]. While these priors can be easily encoded in the architecture exploiting Data Augmentation strategies, it must be noted that this way we are making the network invariant around the kind of transformations which are applied at training time, which in practice may not be the same at test time.

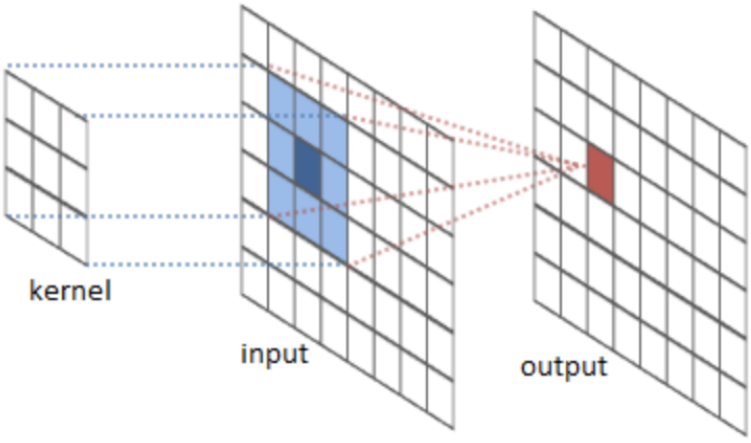

Figure 2.10: A scheme showing the application of a 3x3 kernel to an input feature map. The image is taken from Kaggle.

The core element is the Convolution: input data are convolved with learnable kernels, or filters, generating a certain number of feature maps which constitute the input for the next layer. These kernels, which act as a sliding window over the input tensors looking for patterns, are able to learn local features about the data which are more and more specific as the depth of network increases. Fig. 2.10 shows the application of a Convolutional filter to a feature map. Convolutions, thanks to weight sharing and being only locally connected with the previous layer, enable a significant reduction in the number of weight per layer which leads to a faster training phase and a decreased overfitting capability of the architecture. Formally, the output of a convolution followed by a ReLU [21] activation can be formulated as follows:

$$h_j^n = max(0, \sum_{k=1}^{K} h_k^{n-1} \cdot w_{kj}^n)$$

where h_k^{n-1} is the input feature map, h_j^n is the output feature map, w_{kj}^n is the kernel and the max operator refers to the ReLU activation. A

Convolutional layer is characterized by four main hyper-parameters which need be chosen:

- size, which is generally squared, e.g. 3x3, 5x5, 7x7 for 2D convolutions;
- stride, the step taken when sliding over the input tensor;
- padding, which increases the input tensor size in order to make it compatible with the kernel size and stride;
- number of filters, which also determines the number of output channels.

The Pooling layer acts instead as a downsampling element which reduces the resolution of the input tensor making it more manageable. A Pooling layer, which is applied channel-wise, generally does not possess learnable weights but, similarly to a kernel, it is characterized by a certain size and a stride. Fig. 2.11 shows the application of a Pooling layer to a feature map.

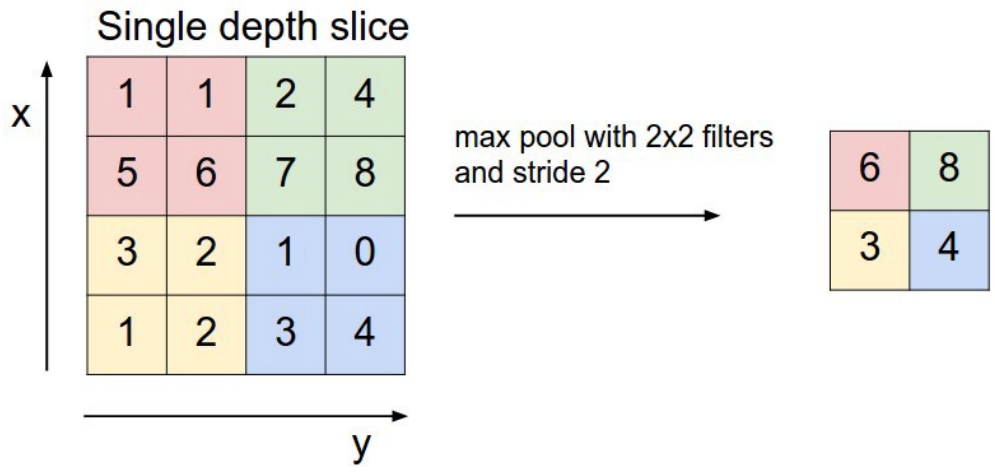

Figure 2.11: A scheme showing the application of MaxPool to an input feature map. The image is taken from [20].

There exist two main types of Pooling:

- Max Pooling, which retains only the highest value of the window and it is the most common one [46];
- Average Pooling, which instead considers all the window to produce its output.

More advanced architectures, such as ResNets [26], do also employ a so called Global Pooling layer which downsamples an entire feature map returning a single value, generally the average, for each input feature map leading to a large reduction in the number of parameters and allowing the network to take as input images of different dimensions. Global Pooling is intended to substitute the final Fully Connected layer in order to generate a fully Convolutional architecture, but in practice it is often employed before the last FC of the architecture.

The combination of downsampling and weight sharing given by the use of kernels allows a significant reduction in the number of weights in the network, leading to a lower overfitting capacity and allowing a faster training phase which also requires less computational resources.

2.3 Unsupervised Learning

While Supervised Learning relies on annotated data, Unsupervised Learning algorithms detect patterns or extract information exploiting the intrinsic properties of the samples, which are no labeled.

Clustering is the main family of Unsupervised Learning algorithms. The objective of Clustering techniques is to group the different data points into, possibly meaningful, categories according to similarities or distances across samples such that the intra-cluster distance is minimized while the inter-cluster distance is maximized. Clustering can be of two types:

- hierarchical, if generates a set of nested clusters which can be abstracted to a hierarchical tree;
- partitional, if generates a set of non overlapping clusters, i.e. each data point is in exactly one single cluster.

The major limitation of Clustering is the difficulty in assessing the fitness of the various clusters when external metrics are not available. Luckily, in our case we do possess the ground truth for our task in order to tune and test the algorithm.

2.3.1 Agglomerative Clustering

The data collection pipeline for the dataset also exploits *Agglomerative Clustering* (AG) [47] for the reconstruction of the 3D poses when multiple

people are in the scene. Agglomerative Clustering is a hierarchical Clustering technique where all the sample points start as a single cluster and at each iteration the two closest clusters are merged according to a certain metric distance. The algorithm ends when a certain number of clusters has been reached or when the minimum distance among the two closest clusters is greater than a predefined threshold value. There exist four main different kinds of Agglomerative Clustering, according to the criterion followed for the computation of the distance among different clusters:

- single linkage, which computes the distance between clusters as the minimum distance among their points;
- complete linkage, which is the opposite of the single linkage considering the maximum distance among the points of each cluster;
- average linkage, which instead computes the distance as the average distance of points of each cluster;
- ward linkage, where the merging criterion is instead given by the optimization of a function aiming to minimize the whithin-cluster variance.

Figure 2.12 shows the differences in performance among these methods by applying them to different kinds of data.

Single linkage is the only able to efficiently handle non-elliptical shapes, but is sensitive to noise and outliers while on the contrary the complete linkage tends to globular clusters but it is less sensitive to noise and outliers. Average linkage is instead the compromise between these two other methods being less biased to globular shapes while robust to noise. Ward's method, despite being biased to globular shapes too, is able to address clusters with different density, as we can see from the third row of Figure 2.12, but there exist other Clustering algorithms which are more suitable to handle cluster with different densities, such as DBSCAN [48].

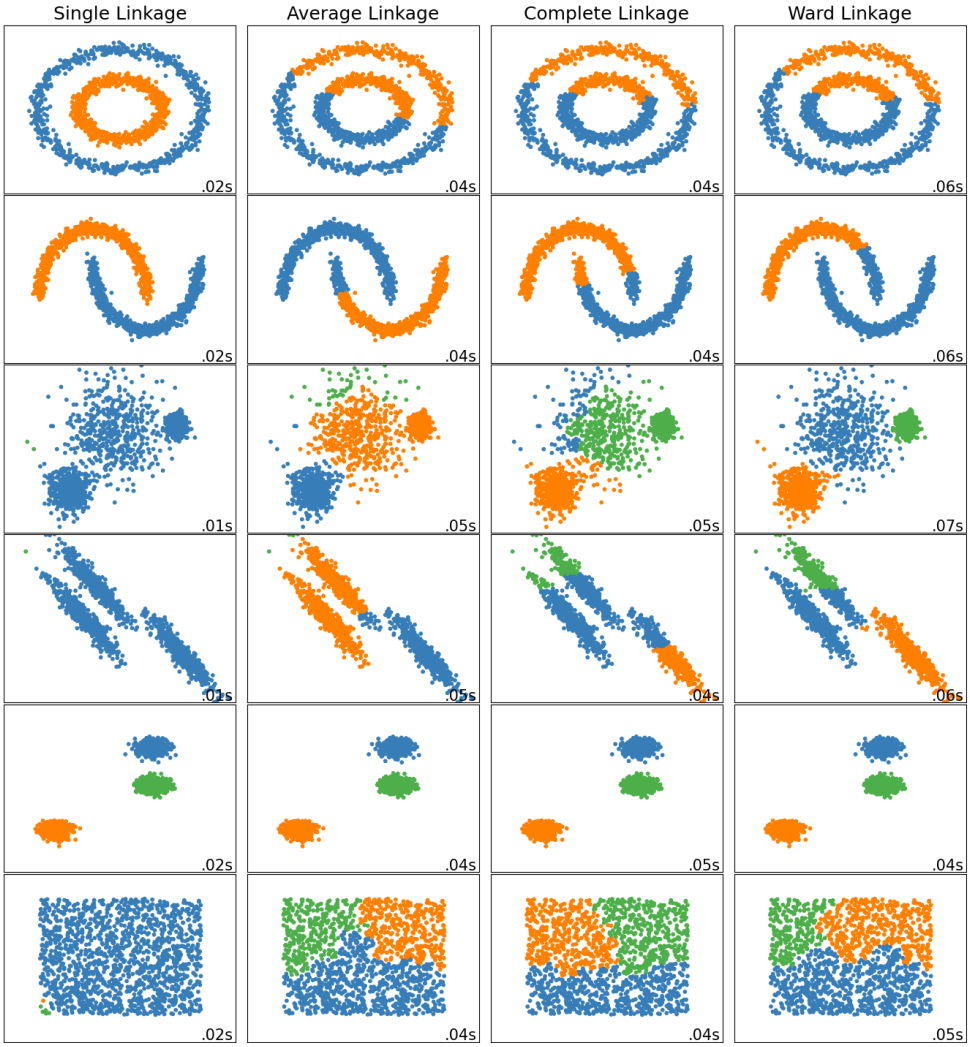

Figure 2.12: A scheme showing the performance of different types of Agglomerative Clustering. The image is taken from Sci-kit learn official website.

Chapter 3

Previous Work

3.1 Pose Estimation

Pose Estimation is a deeply studied Computer Vision task which consists in the recognition and localization of the different keypoints corresponding to multiple parts of the human body: head, neck, shoulders, wrists, knees, ankles and so on. The synthetic set of keypoints of a person is generally called *skeleton*. The possible application fields for this kind of technology range from Computer Graphics to surveillance, industrial safety, autonomous driving, activity recognition, gaming and so on.

During the last years astonishing results have been achieved in Pose Estimation, both in 2D, with OpenPose [7] or AlphaPose [6] for example, and in 3D with VoxelPose [9]. There exist two different approaches to address this task:

- top-down approach [6, 8], where first the area containing the subject is identified by means of Detectors, such as YOLO [49] (also employed internally by AlphaPose), and then the area is fed to a pose estimator to localize the keypoints of a single individual;

- bottom-up approach [7, 50], where first all the keypoints in the scene are identified exploiting postprocessing techniques, such as non-maximum suppression, and they are later associated to a single person.

3D Pose Estimation from a single view is still an unsolved problem and that is the reason why 3D Estimation is not generally performed relying on a single RGB frame but exploiting different techniques and technologies:

Figure 3.1: A sample from the COCO dataset.

- a multi-camera system [9], as the one developed in this Thesis and which will be described later;

- RGB-D cameras to retrieve depth information in a Sensor Fusion fashion, like the Microsoft Kinect [51].

The astonishing progress in this field was mainly enabled by the availability of different public large scale datasets for this purpose, such as COCO [52], developed and maintained by Microsoft, or Panoptic [53], which was used in this work in order to develop and tune the algorithms for 3D skeleton generation in the dataset.

COCO contains more than 200,000 images and 250,000 labeled 2D skeletons in a various range of environments, while Panoptic contains over 5.5 hours of data from a large range of sensors, HD cameras, VGA cameras, RGB-D cameras, Kinects with 1.5 millions labeled 3D skeletons in a single controlled setting.

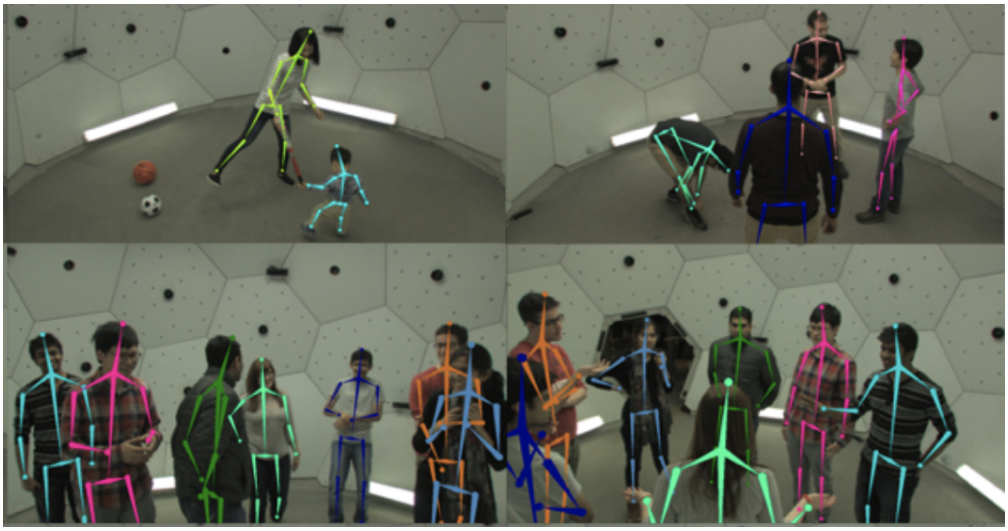

Figure 3.2: Some samples from the Panoptic dataset. The 3D skeletons are projected to camera views.

3.2 Radar-based Pose Estimation

While vision-based Pose Estimation is a well developed field, there are still few works involving Frequency-Modulated Continuous Wave radars for this purpose. There exist two different research branches employing wireless signals in order to decode the human pose:

- device-based, which identifies the keypoint positions by exploiting wireless devices or markers on the subjects;
- device-free, which instead does not require the subjects to wear any kind of sensors.

While device-based tracking is an easier task, device-free methods offer obvious advantages in terms of deployment and applicability, despite being more challenging. For this reason we chose this kind of technology as the focus of this Thesis work.

The first paper exploiting FMCW radars in this relatively new research area has been produced in 2015 at MIT[1], where a group of researches proposed

[1]MIT currently posses the largest, unfortunately private, heterogeneous dataset related to this task.

RF-Capture [12]. This work was just scratching the surface of the problem relying on shallow algorithms fed with radar heatmaps to only retrieve a coarse representation of a human body and not the actual localization of the various keypoints. The ouput of RF-Capture can be seen in Fig. 3.3.

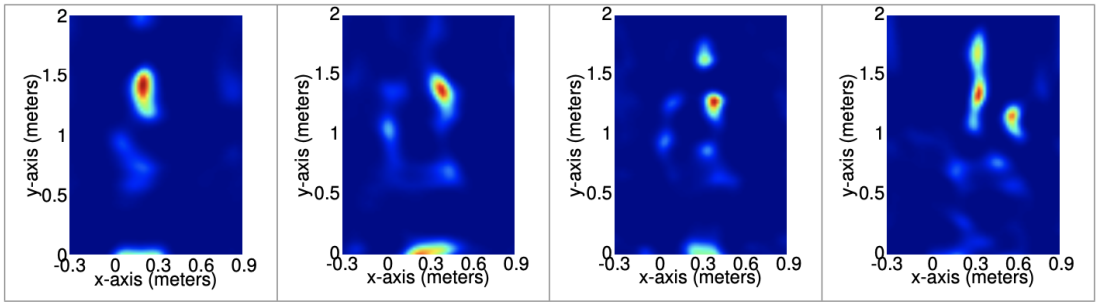

Figure 3.3: An example of the output of RF-Capture from the original paper.

The same research group continued working on this topic developing RF-Pose [54], the first paper in which Deep Learning is applied for actual radar-based 2D Multi-Person Pose Estimation. RF-Pose is a bottom up method which exploits an Encoder-Decoder architecture supervised by OpenPose [7], which generates the ground truth for the radar-based model in a cross-modal Teacher-Student fashion [55]. RF-Pose was then further improved to work in a 3D scenario with the development of RF-Pose 3D [14], which is instead a top-down approach. RF-Pose 3D exploits a ResNet [26] design for the backbone of the network and introduced a Region Proposal Network acting as Detector. The work was later extended with RF-Action [13], which is the first paper employing the radar technology for Action Recognition purposes, and with [56] which is instead the first work exploiting the radar signal for Human Mesh Recovery.

However, the MIT research group employed a very large 2D radar array, which enables the association of the received signal to a 3D spatial voxel, while Inxpect radars are small 1D arrays and are therefore limited to the range and azimuth plane, without elevation information. This issue is addressed by exploiting two identical 1D arrays, one of which rotated of 90 degrees, and combining the two signals. This is the approach adopted in mm-Pose [57], in [58], which we will refer to as PoseCapture, and also in this Master Thesis. It must also be noted that the size of the antenna array employed by MIT researchers is 60×18 cm, with advantages in terms of resolution but

with serious issues in terms of deployment. On the contrary, Inxpect radars are much smaller (9×9 cm) yielding a greater easiness in the deployment of such systems.

mm-Pose extracts the XY and YZ coordinates of the reflected points from the signals of the two radars and generates a 16x16x3 matrix from each of them as input for the CNN. Each entry of the matrix contains the coordinates of a reflected point, up to 256 points, in the first two channels and the intensity of the reflected signal in the third one, while the entries not corresponding to a reflection point are set to (0, 0, 0). The points are ordered following their time of arrival, i.e. by range dimension. This peculiar preprocessing ensure a consistent dimensionality reduction, as claimed by the authors. The two matrices are then fed into a branched CNN with a final Fully Connected layer which predicts the 3D position of the keypoints of a single person. It must be noted that the good results they achieved exploiting a single frame for the estimation process may be due to their simple setting, where the subject was mostly facing the radar while performing simple actions such as walking, waving or lifting the arms.

PoseCapture exploits a simpler and more standard preprocessing, feeding the network with 60 consecutive range-angle heatmaps from the two radars. The researchers adopted an Encoder-Decoder architecture with 3D convolutions. The output of the branched Encoder is concatenated and fed to the Decoder, which outputs a 2D heatmap. The position of the keypoints of the people in the scene is extracted by means of non-maximum suppression. However, it must be noted that the researchers limited their dataset and experiments to a single person scenario, for which the decoder may be redundant.

Both mm-Pose and PoseCapture will be later described more in details, as they constitute the baseline models adopted as a benchmark for the dataset.

3.2.1 Main Issues

Due to the specularity of the human body to GHz RF waves, the radar signals may be directed away from the receiving antennas resulting in missed detections. While this fact may not represent a problem for a coarse detection of a person or for the identification of body parts with a large reflective surface, such as the torso, it could represent a serious issue for the identification of smaller parts such as the limbs. This is due to the fact that large reflective surfaces are likely to reflect at least part of the signal in the right direction

while, on the contrary, smaller and more mobile parts such as the limbs are likely to not be detected since they may reflect the signal away from the RX antennas. That is the reason for which the literature exploited multiple consecutive frames as input for the networks: 3.3 seconds for RF-Pose3D and 2 seconds for PoseCapture. Since the time dimension is also considered, the architectures must employ 3D Convolutions which are much more computational demanding with respect to their simple 2D counterparts, causing a significant increase in terms of training time and model dimension. Another relevant issue is given by the much lower resolution of a radar with respect to RGB sensors, which makes the exact localization of the joints a very challenging task. Moreover, the presence of multipaths is going to further degrade information carried by the signal, especially in heavily cluttered areas.

Part II

Dataset Collection and Evaluation

Chapter 4

Capture System

In order to collect a significantly large dataset which could help the research community in the progress of radar-based Pose Estimation and related tasks, we developed a cheap and easily movable Capture System. Three different design choices for the data collection were taken into account, each one with its advantages and disadvantages:

- a VICON [59] system, which is the commercial golden standard but it is very expensive and cannot be easily moved to different settings;
- a Kinect [51], which was the easiest and fastest choice but it suffers the presence of occlusions and the deployment of multiple Kinects in order to overcome this issue can lead to interference in their signals;
- a multi-camera system, robust to occlusions and easily movable but which needed to be built from scratch.

A Vicon system was employed by the authors of RF-Pose 3D [14] but only as a benchmark for their movable multi-camera system, since they wanted to collect the data across different environments. The second approach with a single Kinect was instead followed, for example, by the authors of mm-Pose [57]. Despite the efforts required to build a multi-camera system, we opted for this last option following the design employed by the authors of RF-Pose 3D, which showed a very low error with respect to the Vicon system.

4.1 Hardware

4.1.1 Controlling Devices

The developed Capture System[1] is a modular ensemble of RaspberryPi 4, a famous small single-board computer developed in the United Kingdom. Currently the system consists in 5 devices -with more in deployment-, each one with its own batteries and a tripod in order to be easily moved across various settings to ensure model generalization.

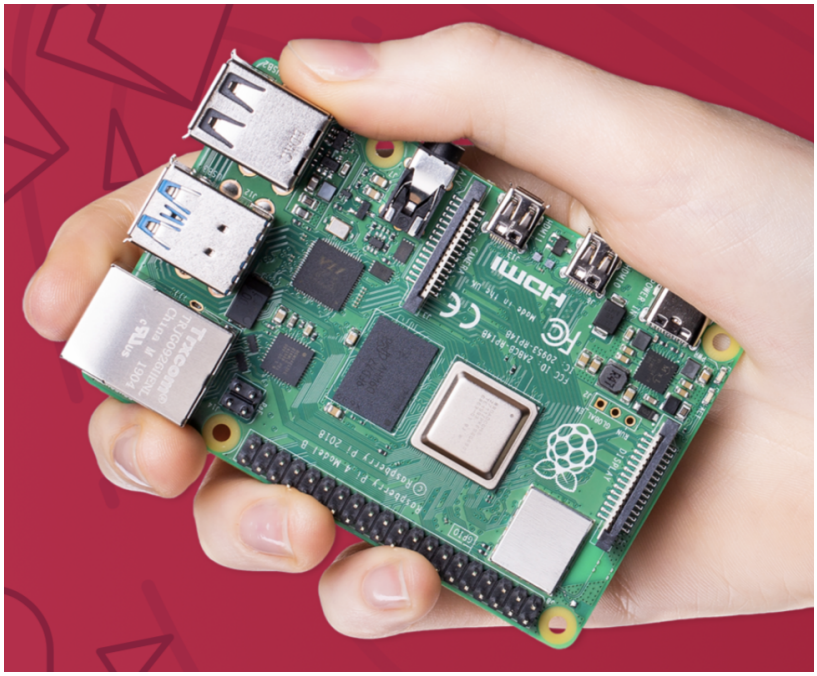

Figure 4.1: A photo of a RaspberryPi 4 from the official website.

Raspberry Pi boards offer multiple advantages such as the small size, which enables an easy deployment, and the very low prices which are in the order of 30-40 € circa, making the realization of the collection system relatively cheap and easily scalable to more environments or to a larger number of devices in the same setting to achieve better estimation performances.

Each device controls a PiCamera v2 which provides RGB supervision, while two of them also control the two radars deployed in correspondence of

[1]The system is completely funded by Inxpect.

the central device.

4.1.2 Sensors

PiCamera v2 are cheap camera sensors, less than 30 €, with 8 Megapixels. They mount a Sony IMX219 sensor with maximum resolution of 3280x2464 pixels and a pixel size of 1.12x1.12 μm. They can be either employed by means of simple Bash commands, raspistill and raspivid, or using various APIs such as the one for the Python programming language.

The mmWave radars employed are the SBV-01 radars produced by Inxpect. They operate at 60 GHz with 2 transmitting antennas, 4 receiving antennas and a response time lower than 100 milliseconds. Thanks to their limited size, such radars are suitable to be easily deployed in a large number of different scenarios, from industrial settings to autonomous driving. This kind of radar, however, possesses a low spatial resolution (40 cm in depth, 10-15 degrees in azimuth) which makes the Pose Estimation task even more challenging. We refer to Inxpect official website for other details.

Figure 4.2: An SBV-01 radar produced by Inxpect. The image is taken from Inxpect official website.

As previously said, in order to address the lack of elevation information in standard 1D radar arrays as the ones produced by Inxpect, the two radars

are arranged as a cross, as done in mm-Pose and PoseCapture, by means of an ad hoc 3D printed support realized by Inxpect.

4.2 Software

4.2.1 Synchronization

In order to associate the frames from the different devices, the RaspberryPis must be accurately synchronized. This is achieved by means of the Network Time Protocol (NTP) [60] which enables the exchange of time information exploiting the User Datagram Protocol (UDP) on the port 23. One of the RaspberryPis acts as the NTP server dictating the time to the other devices acting as NTP clients. NTP is able to maintain the error from Coordinated Universal Time over Internet in the order of tens of milliseconds while in a typical local subnet, as in our case, the delay between client and server is typically in the order of 2-3 milliseconds, thus ensuring the reliability of the timestamps used for the association of the frames coming from the different cameras and radars. Moreover, this synchronization is quite simple to realize and requires few commands and changes using a UNIX-like system like the ad hoc Linux version for RaspberryPi boards, once called Raspbian.

4.2.2 Camera Calibration

In order to reconstruct the 3D pose of an individual from multiple 2D views it is necessary to obtain the calibration data of the cameras. Calibrating a camera means computing the so called Extrinsic parameters, i.e. its Camera Matrix M, which is a 4x3 matrix mapping world points to image points and the Intrinsic parameters, which map from an XY image coordinate system to a pixel image coordinate system:

$$\begin{aligned} [\text{x,y,1}] &= [\text{X,Y,Z,1}]\text{M} \\ \text{M} &= [\text{R}|\text{t}]'\text{K} \end{aligned}$$

where lowercase coordinates refer to images point, uppercase coordinates refer to world points, R is a 3x3 rotation matrix, t a 3x1 translation vector and K is the Intrinsics Matrix. Generally, together with K another set of parameters is computed, the so called Distortion parameters, which, surprisingly, encode the lens distortion and are used to undistort the images by eliminating the lens effect.

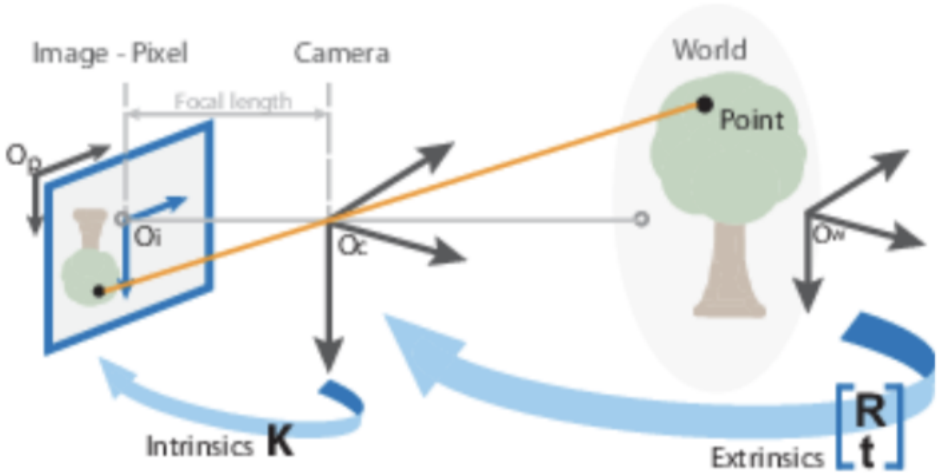

Figure 4.3: A scheme explaining how Intrisics and Extrinsics work. The image is taken from MATLAB official website.

While Intrinsics Calibration is already automatized in several libraries, such as OpenCV, Extrinsics Calibration (R and t) for more than two cameras is a much harder and time consuming task which required us to deploy targets with known coordinates (and thus to manual label the corresponding pixels) on the scene in order to obtain a global 3D reference system. Calibration data is then employed together with the 2D poses in order to reconstruct the 3D skeletons from the set of 2D skeletons in simultaneous frames captured by different cameras.

4.2.3 3D Skeleton Generation

For the 3D reconstruction, we mainly followed again the approach showed in RF-Pose3D. First, we retrieve the 2D poses from each camera by means of a well established vision-based Deep Learning algorithm, such as AlphaPose [6] which performs slightly better than the one adopted by MIT researchers, OpenPose [7]. Then the association of different 2D skeletons of a same person in simultaneous frames from different cameras is then performed by means of *Agglomerative Clustering* exploiting the scikit-learn library: the distance metric in our case consists in the distance between two skeletons, which is computed as the mean Euclidean distance between the 3D lines exiting from the cameras and intersecting some of the keypoints in 3D.

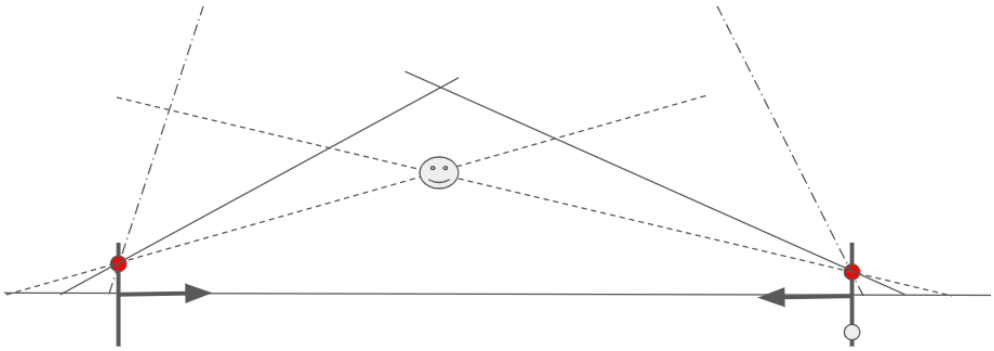

Figure 4.4: Camera Geometry

To achieve this goal, we first identify the 3D position of the pixel containing a certain keypoint in an image and we then compute the parametric equation of the line passing from such point and the focal center of the camera. The parametric equation of a line passing through two points x_1 and x_2 is simply the following one:

$$p = x_1 + (x_2 - x_1)t$$

with $t \in (-\infty, +\infty)$. Then the distance d between two lines is computed by means of the standard formula for the distance between two parametric lines:

$$d = \frac{n(r_1 - r_2)}{\|n\|}$$

where r_1 and r_2 are the direction vectors of the two lines and n is the cross-product of r_1 and r_2, namely $n = r_1 \times r_2$.

Once we have different 2D skeletons of the same person, the 3D Triangulation is performed similarly to what is described in RF-Pose3D, adding a weight γ for each camera: the 3D position p of a single keypoint is computed by minimizing the weighted distance from its 2D projections.

$$p = \arg\min_{p} \sum_{i \in I} \gamma_i ||C_i p - p^i||_2^2$$

"where the sum is over all cameras that detected that keypoint, and C_i is the calibration matrix that transforms the global coordinates to the image

coordinates in the view of camera i" [14] and γ_i is the confidence of the keypoint as seen by camera i. Moreover, only keypoints with a confidence greater than 0.6 are taken into account in the summation.

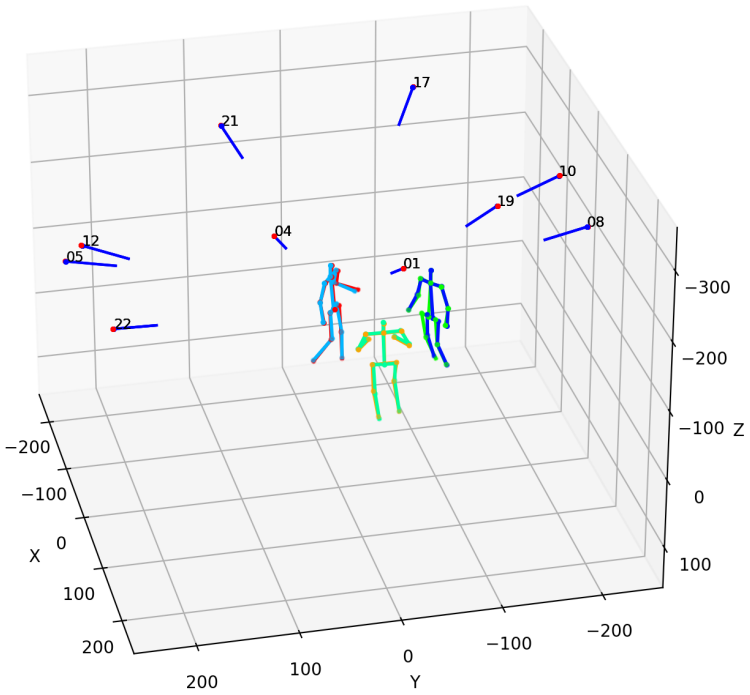

Figure 4.5: Sample of predicted skeletons over true skeletons. Numbered blue points are the camera centers, in red the focal centers and the lines represent the normal to each camera.

The optimization algorithm adopted for the minimization of the function is a standard Newton method implemented in the SciPy library. The Newton method is a sort of extension of the famous Gradient Descent algorithm which locally approximates with a quadratic function, instead of a linear one as Gradient Descent, generally leading to more accurate results and better performances thanks to a quadratic rate of convergence. However, this comes at the cost of being significantly more computationally intensive due to the computation of the Hessian matrix $\nabla^2 f$. Formally, the local function approximation can be formulated as follows:

$$f(x+p) \simeq f(x) + p^T \nabla f(x) + \frac{1}{2} p^T \nabla^2 f(x) p =: m(p)$$

If the Hessian $\nabla^2 f(x)$ is positive definite, p is a descent direction and $m(p)$ is a convex model for f around x, therefore the Newton step consists in the minimization of the local model $m(p)$ at each step:

$$\nabla m(p) = \nabla f(x) + \nabla^2 f(x)p = 0$$

As it is a common drawback of optimization algorithms, also Newton method is quite sensitive to the starting point, which is roughly set to the center of scene achieving this way convergence each time.

Moreover different techniques, such as Inexact Newton Method or Quasi-Newton Method, have been developed in order to mitigate the computational intensiveness of this algorithm making it suitable for large scale problems relying on approximations of the Hessian matrix $\nabla^2 f(x)$.
Clustering and Triangulation are tuned and tested on a scene from Panoptic dataset with three different groups of three subjects: 10463 skeletons and 3519 frames were considered in this sample. The results in centimeters for different numbers of cameras are reported in Tables 4.1 and 4.2.

4.2.4 Radar Data

With regards to radar data, raw samples from the two radars are extracted by means of two DCA100 and later processed by means of a proprietary library of Inxpect. Radar raw cubes are associated to RGB frames by means of timestamps and are then encoded in Base64 (since complex numbers are not json-serializable) in order to be saved into a .json file. Table 4.3 contains the radar configuration options chosen for the task by Inxpect Signal Processing Engineers.

Keypoint	Avg Error	Error Std	Max Error	Min Error
$Nose$	1.431	0.611	6.911	0.034
$Neck$	1.033	0.534	3.723	0.073
$Shoulder_R$	1.362	0.725	7.425	0.015
$Elbow_R$	2.105	0.934	10.252	0.028
$Wrist_R$	2.861	2.262	18.607	0.076
$Shoulder_L$	1.383	0.955	5.138	0.012
$Elbow_L$	2.319	1.299	9.853	0.082
$Wrist_L$	2.604	2.832	22.323	0.032
Hip_R	3.626	1.62	12.745	0.304
$Knee_R$	3.543	3.287	26.263	0.221
$Ankle_R$	2.942	1.498	36.961	0.184
Hip_L	3.572	2.159	11.678	0.201
$Knee_L$	3.646	1.38	15.016	0.393
$Ankle_L$	3.221	2.028	32.703	0.238
Eye_R	1.875	0.629	5.73	0.372
Eye_L	1.785	0.519	5.677	0.292
Ear_R	1.348	0.599	5.838	0.079
Ear_L	1.383	0.526	3.578	0.063

Table 4.1: Average, standard deviation, maximum and minimum error per keypoint on a scene from Panoptic Dataset using 10 cameras. Values in the table are in centimeters.

Keypoint	Avg Error	Error Std	Max Error	Min Error
$Nose$	1.558	0.841	9.963	0.028
$Neck$	1.177	0.586	5.667	0.012
$Shoulder_R$	1.299	0.736	9.827	0.062
$Elbow_R$	2.824	2.245	21.71	0.057
$Wrist_R$	3.18	3.33	29.971	0.062
$Shoulder_L$	1.489	0.817	10.977	0.062
$Elbow_L$	2.345	1.684	19.845	0.115
$Wrist_L$	3.534	4.386	29.937	0.03
Hip_R	3.776	1.586	21.647	0.346
$Knee_R$	4.264	3.991	28.882	0.238
$Ankle_R$	4.385	3.554	29.443	0.276
Hip_L	3.107	1.419	16.888	0.17
$Knee_L$	4.037	1.907	24.009	0.158
$Ankle_L$	5.243	4.492	29.536	0.137
Eye_R	1.951	0.831	9.36	0.439
Eye_L	1.917	0.742	9.851	0.18
Ear_R	1.562	0.658	9.342	0.137
Ear_L	1.506	0.575	5.974	0.107

Table 4.2: Average, standard deviation, maximum and minimum error per keypoint on a scene from Panoptic Dataset using 6 cameras. Values in the table are in centimeters.

Option	Value
Chirps per Frame	20
Samples per Chirp	256
Frame Period	25 ms
Start Frequency	60.6e9 Hz
Rampe Rate	40e12 Hz/s
Sampling Frequency	3000e3 sample/s

Table 4.3: Radar Configuration.

Chapter 5

Inxpect Dataset

Setting The dataset, which we called Xpective, was collected at Visual and Multimodal Applied Learning (VANDAL) laboratory at Politecnico di Torino. VANDAL is part of Italian Institute of Technology (IIT) and constitutes one of the units of European Laboratory for Learning and Intelligent Systems (ELLIS).

Currently, there is a single environment and the collected samples contain a single person at a time: multi-person scenarios and different environments are going to be added soon.

In order to have a realistic setting, the dataset is not collected in an empty environment but in our office, which is a heavily cluttered space likely to cause the emergence of a large number multipaths and ghost targets.

Content and Dimension Up to now, Xpective consists in 26 scenes of 3 minutes for a total of 78 minutes of data with over 135k labeled skeletons of 3 different people performing random movements, such as walking, waving, drinking, using the phone and so on. Each skeleton is composed of 18 different keypoints, following the "open" COCO format of AlphaPose [6] which is the vision-based model adopted to extract the 2D skeletons from the single camera views.

The dataset is organized in different environments which consist of various scenes together with the Calibration data of the cameras. For each scene, they are available the videos from the different cameras, the raw radar cubes from the two Inxpect SBV-01 radars, the 2D skeletons as they are generated by AlphaPose and the reconstructed 3D skeletons. Each environment also contains a recording for each radar when there are no people in the scene in

order to optionally eliminate static reflections and multipaths coming from the environment.

With regard to RGB data, we adopted a framerate of 30, the size of the frames is 1280x960 and the videos are saved in .mp4 extension. Frames can be easily extracted from the videos with a single Bash command by means of FFmpeg [61]. As said before, radar data is encoded in Base64 (since complex numbers are not json-serializable) and then stored to a .json file. In the Appendix it is possible to find a Python function which acts as object hooker to read the .json files.

Applications Since the dataset contains both RGB and radar data, it is suitable to be employed by using the single modalities only or in a Sensor Fusion fashion, constituting a useful tool in the development of Pose Estimation and related research. Moreover, a further labeling of the scenes may easily extend the use of this dataset to Activity Recognition or Person Identification tasks.

Chapter 6

Radar-based Pose Estimation Experiments

In order to establish an initial benchmark for the dataset, we took into consideration two[1] different models, mm-Pose [57] and PoseCapture [58]. While both the works exploit two identical 1D radar arrays to overcome the lack of elevation information, they employ different radar preprocessing techniques and different architectures, which will be now described in details. Moreover, we conducted an ablation study involving other architectures as backbones and an analysis regarding the time dimension.

6.1 Baselines

6.1.1 mm-Pose

Preprocessing As mentioned in the section about previous works in radar-based Pose Estimation, mm-Pose adopts a peculiar preprocessing strategy with the aim of reducing the size of the input data while preserving the valuable information to achieve real-time performances. By means of standard signal processing techniques, as the ones described in the Prior chapter, the authors extracted the XY and YZ coordinates of the reflected points, up to 256 points, from the two radars and they then generated a 3x16x16 matrix

[1]MIT research could not be replicated due to the different nature of radar systems employed, i.e. a single 2D radar array instead of two 1D radar arrays.

from each of them as input for their architecture. Each entry of the matrix contains the coordinates of a reflected point in the first two channels and the intensity of the reflected signal in the third one. Entries not corresponding to a reflection point are set to (0, 0, 0). The points are ordered following their time of arrival, i.e. by range dimension. Fig. 6.1 shows a sample of two paired input matrices.

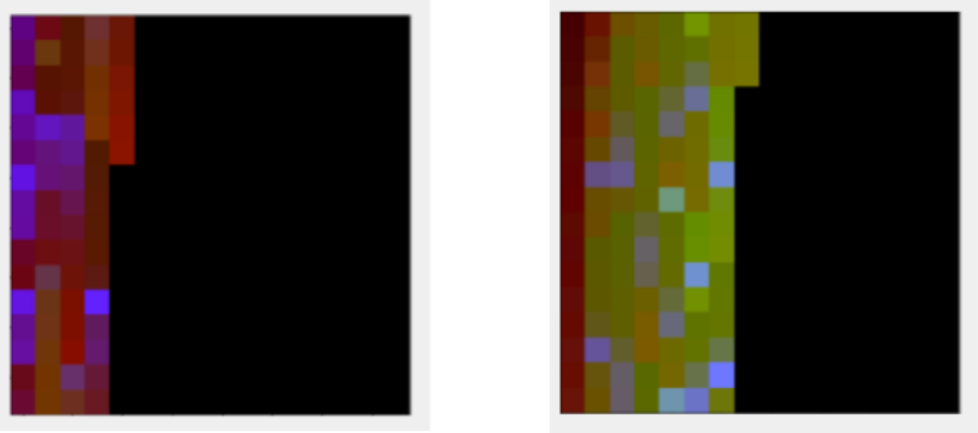

Figure 6.1: A representation of the input matrices from the original paper.

Architecture The architecture is made up of two simple 3-layers CNNs, whose output is concatenated and then fed to a small 3-layers MLP head before the last fully connected layer which outputs the 3D coordinates of different keypoints of a single person. Since the input is a single matrix per radar, i.e. they do not exploit temporal information across previous frames, the convolutions are 2D. The kernel size is 3x3 with stride 1 and same padding, while the number of filters per convolutional layer is 16, 32, 64. The number of nodes per layer in the MLP is instead 512, 256, 128. ReLU activation function is used after each layer but the last, convolutional layers are followed by a 0.2 rate dropout while the fully connected ones by a 0.3 rate dropout to avoid overfitting. The loss minimized at training time is a Mean Squared Error (MSE), and the optimizer adopted is Adam [37].

The good results achieved despite this simple method, which does not take into considerations multiple consecutive frames, may be due to the nature of the dataset collected by the authors, which is not particularly challenging since the subject is mostly oriented towards the radar while executing simple movements such as lifting an arm or waving.

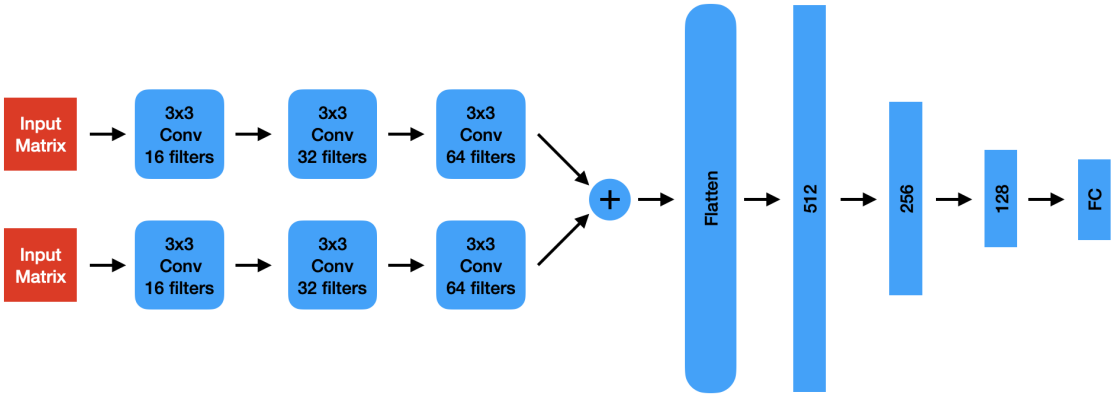

Figure 6.2: A scheme of the mm-Pose architecture.

6.1.2 PoseCapture

Preprocessing The authors of PoseCapture followed a more standard approach. They first extracted the range-angle matrices from the raw data and then they generated a range-azimuth and a range-elevation heatmaps which are fed to their architecture.

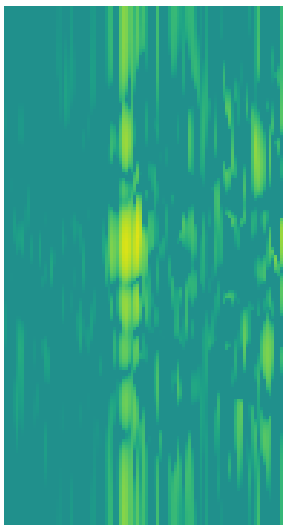

Figure 6.3: An example of a range-azimuth heatmap from our dataset.

Architecture The architecture follows an Encoder-Decoder design. Since they feed to the network two seconds of data, i.e. 60 frames with a framerate

of 30, to predict the pose their network adopts 3D convolutions. The Encoder is composed by two identical 5-layers CNN branches with kernels of size 9x5x5. Batch Normalization and ReLU activation are applied after each layer. Nothing is specified about stride, padding and number of filters per layer. The output of the two branches is concatenated and then passed to the Decoder, which generates a 2D heatmap containing the locations of the keypoints of the people in the scene. The Decoder is composed of 4 fractionally strided Convolutional layers with kernels of size 3x6x6 and 1x2x2 stride. The first 3 layers are followed by Parametric ReLU [1] while the last one is followed by a sigmoid in order to scale the outputs in the [0,1] range. The loss function minimized at training time is a Binary Cross Entropy (BCE), the batch-size is 8 and the optimizer adopted is again Adam.

Due to the lack of information, we had to make assumptions about stride, padding and number of filters in the Encoder:

- the stride is 1x2x2, as it is in the Decoder and in RF-Pose for all the layers but the last two which have stride 1, due to size compatibility with our heatmaps;
- no padding, since there is no reference to it;
- the number of filters starts from 8 for the first two convolutions and it is doubled after each layer, as it is standard practice.

The choice for a low number of filters is due to the slowness of the architecture. Moreover, since the adaptation of the Decoder in a 3D scenario would be computationally unfeasible for our GPUs, we removed this part of the architecture and substituted it with a simple FC layer acting as multi-regressor, as proposed by the authors of mm-Pose.

6.2 Ablation Study

Backbones Together with the two Baseline models, we also evaluated different standard architectures as backbones for the branches. Since proper 3D architectures are quite heavy to train and they often are too large even for a simple forward-pass with our GPUs, we modified standard 2D architectures by changing Convolutions, Pooling and BatchNorm layers changed from 2D

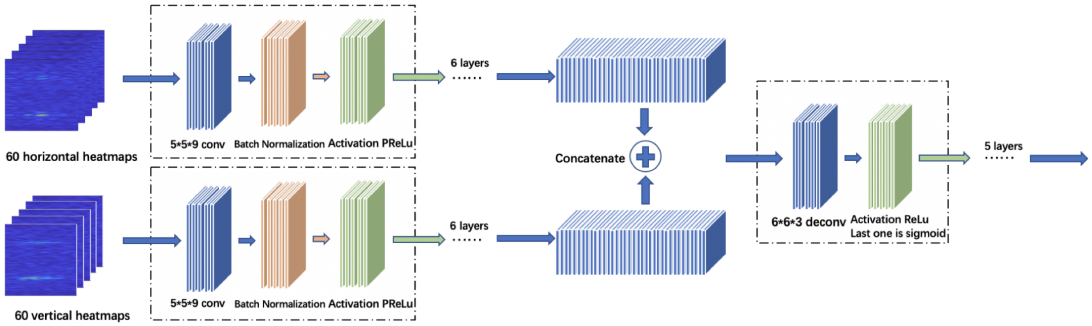

Figure 6.4: A scheme of the PoseCapture architecture from the original paper.

to 3D. A ResNet18 (2+1)D [62] is the only exception[2] since this model is both light and fast despite being originally developed for videos processing. Since 3D convolutions are quite computationally expensive, we reduced the number of filters of these models to make them trainable in a reasonable time on our GPUs. This reduction not only makes these architectures faster and lighter, but it also reduces the their overfitting capability with gains in terms of test error.

- In the ResNet [26] family, we explored ResNet18, ResNet34, ResNet50 and, as just said, a ResNet18 (2+1)D. The number of channels is divided by 8 from [64,128,256,512] to [8,16,32,64];
- In the DenseNet [64] family, we explored DenseNet40 and DenseNet76 with initial number of filters divided by 8 from 64 to 8 and the growth rate is divided by 3 from 12 to 4;
- Considering more shallow and simpler networks without Residuals, we explored a VGG16 [25] with Batch Norm, the Average Pooling performed only on the spatial dimensions and the number of channels divided by 8 from [64,128,256,512] to [8,16,32,64];
- For lightweight scenarios, we also explored a MnasNet0.25 [65] that should be more accurate than both MobileNetV1 [66] and MobileNetV2 [67] while being faster and with less parameters.

[2]Other standard 3D architectures, like X3D [63], were considered but they were far to heavy to train.

We refer to the original papers for an exhaustive description of these architectures, since it is out of the scope of this work.

All these architectures are fed with 30 Range-Angle (RA) heatmaps (e.g. 1 second) sampling a frame every two. We removed the classifier from each network and concatenated the output of the two convolutional branches. The concatenation is then passed to a final FC layer for the actual prediction, as done in mm-Pose.

Time Dimension Once we found the best performing architecture, we also investigated how many frames does it leverage to predict the actual pose by feeding 15, 30, or 60 frames (i.e. half, one or two seconds) per single inference. Moreover, since we expect the frames to be highly redundant, we also explored a simple sampling strategy by considering a frame every 3, 4 or 5.

6.3 Implementation and Training Details

All the architectures are fed with standard radar heatmaps. The models try to minimize the Mean Squared Error (MSE) by means of Adam [37] optimizer. The initial learning rate for all the networks is $1e-2$. Since both in mm-Pose or PoseCapture no information is given about the learning rate scheduling, we adopted Cosine Annealing with Warm Restarts [68] with $t_0 = 2$ and $t_{mult} = 2$ for all the models. The batch-size is 32 and the Weight Decay is set to 1×10^{-5}. All the models are implemented by means of the PyTorch [44] framework and are trained using two NVIDIA GTX 1070 for a maximum of 24 hours or 62 epochs (4 Warm Restarts), with the exception of mm-Pose which, being a lot faster due to the adoption of a 2D architecture, is simply trained for a maximum of 24 hours.

We made sure that the models do not see any of the test data at training time: all the scenes but a randomly chosen one are adopted for training while the test is performed on the remaining one (scene 8). The keypoints which have to be predicted are the the ones in the dataset with the exception of nose, ears and eyes that are averaged to obtain the head.

6.4 Results

Baselines Despite being an ad hoc architecture, mm-Pose miserably fails on the dataset and is the worst model among the tested ones achieving an average test error of 40.8 centimeters. This may be due to the necessary adaptation, e.g. the Global Average Pooling layer, we applied to the model to allow mm-Pose to be able to manage the radar heatmaps which are larger than the input matrix originally adopted in the original work. However, it must be noted not only that the original architecture is quite shallow and naïve, but also the fact that taking as input 1 single frame it is likely that some keypoints are not detected by the radars. Moreover, our dataset is more challenging than the one collected by the authors of mm-Pose and the environment is also quite cluttered, making an actual localization of the keypoints an even harder task.

On the other hand, PoseCapture, which has been heavily modified to be able to work in a 3D scenario, achieves far better results than mm-Pose with 15.13 centimeters of average test error. However, also this model is very shallow and naïve and indeed it is still outperformed by many standard architectures both in terms of precision and training time.

Analyzing the average error per keypoint it is clear that, while the larger and more static keypoints are easier to localize, the limbs are the most difficult parts to predict. This is due to two different factors: the limbs possess a higher degree of freedom and a smaller reflective surface. Indeed, the limbs are generally the hardest keypoints to predict also in classic RGB-based Pose Estimation due to their higher mobility, but in this scenario the error is even more pronounced due to their limited reflective surface which is also likely to reflect the signal away from the RX antennas, resulting in missing detections. Tables 6.1 and 6.2 show the average and median test error per keypoint for the two Baselines. Due to a non-symmetric distribution of the errors, the median is actually a better estimator than the average.

Backbones The other standard architectures which have been tested on average achieves better results than the state-of-art models, with also the two worst backbones, MnasNet0.25 [65] and DenseNet40 [64], scoring an average test error of 14.06 and 14.17 centimeters. VGG16 [25], despite being a relatively simple model without even Residuals, performs surprisingly well with an average test error of 13.34 centimeters but it is outperformed by more complex architectures. DenseNet70 achieves a good 13.08 centimeters

Keypoint	mm-Pose	PoseCapture
$Head$	38.26 ± 23.3	$\mathbf{10.81 \pm 8.25}$
$Neck$	35.62 ± 23.05	$\mathbf{8.74 \pm 8.08}$
$Shoulder_R$	40.56 ± 23.26	$\mathbf{11.71 \pm 9.51}$
$Elbow_R$	48.44 ± 23.75	$\mathbf{19.93 \pm 12.57}$
$Wrist_R$	55.75 ± 23.81	$\mathbf{28.86 \pm 18.29}$
$Shoulder_L$	35.88 ± 22.38	$\mathbf{10.80 \pm 8.17}$
$Elbow_L$	42.31 ± 22.25	$\mathbf{19.33 \pm 10.9}$
$Wrist_L$	50.24 ± 24.49	$\mathbf{29.8 \pm 16.88}$
Hip_R	38.44 ± 23.21	$\mathbf{9.8 \pm 9.03}$
$Knee_R$	39.7 ± 23.64	$\mathbf{12.1 \pm 11.13}$
$Ankle_R$	39.56 ± 24.39	$\mathbf{15.25 \pm 12.31}$
Hip_L	34.96 ± 22.81	$\mathbf{8.89 \pm 7.73}$
$Knee_L$	35.9 ± 23.62	$\mathbf{11.34 \pm 10.34}$
$Ankle_L$	35.6 ± 24.09	$\mathbf{14.5 \pm 11.58}$
$Mean$	40.8	**15.13**

Table 6.1: Average test error ± standard deviation per keypoint. All values are in centimeters. Best results are highlighted in bold.

Keypoint	mm-Pose	PoseCapture
$Head$	32.94 ± 35.26	$\mathbf{9.06 \pm 6.9}$
$Neck$	29.76 ± 33.32	$\mathbf{6.71 \pm 5.82}$
$Shoulder_R$	36.77 ± 32.4	$\mathbf{9.71 \pm 7.56}$
$Elbow_R$	46.25 ± 36.74	$\mathbf{17.59 \pm 13.13}$
$Wrist_R$	52.23 ± 36.74	$\mathbf{25.19 \pm 21.29}$
$Shoulder_L$	30.07 ± 34.5	$\mathbf{9.29 \pm 7.42}$
$Elbow_L$	39.53 ± 36.61	$\mathbf{16.9 \pm 12.66}$
$Wrist_L$	45.47 ± 36.1	$\mathbf{26.83 \pm 20.87}$
Hip_R	33.76 ± 34.82	$\mathbf{7.81 \pm 6.4}$
$Knee_R$	35.14 ± 37.16	$\mathbf{9.24 \pm 7.83}$
$Ankle_R$	32.65 ± 37.96	$\mathbf{12.75 \pm 10.61}$
Hip_L	28.44 ± 35.01	$\mathbf{7.18 \pm 5.91}$
$Knee_L$	30.35 ± 38.49	$\mathbf{8.98 \pm 7.4}$
$Ankle_L$	28 ± 39	$\mathbf{11 \pm 10.48}$

Table 6.2: Median test error ± interquartile range per keypoint. All values are in centimeters. Best results are highlighted in bold.

of average test error, which is the best result achieved by a standard 2D model modified in 3D. This result is even more surprising considering how

few parameters this architecture possesses but the training and processing time are still very long due to the large number of input channels in each Convolution. In the ResNet [26] family, as we could expect for the modified 2D models we get more accurate results as the depth of the network increases but the best performing architecture is actually the only proper 3D architecture, ResNet18 (2+1)D [62] which is the only model able to achieve an average error lower than 13 centimeters outperforming not only deeper ResNets but also all the other models. Indeed, the (2+1)D Convolutions are able to better model the phenomenon but this comes at the cost of a triplicated training time with respect to the other ResNet18.

Table 6.3 contains the average test error, the number of parameters and the needed training time for the different backbones.

Model	Avg Error [cm]	Parameters [M]	Time [h]
mm-Pose	40.8	0.28	12.3
PoseCapture	15.13	1.34	16.75
ResNet18	13.88	1.05	5.5
ResNet34	13.82	2	2.5
ResNet50	13.11	1.49	15.3
ResNet18 (2+1)D	**12.76**	0.99	15.25
DenseNet40	14.17	0.1	12
DenseNet76	13.08	0.22	20.25
VGG16	13.34	1.39	10
MnasNet0.25	14.06	1.08	9.5

Table 6.3: Average test error in centimeters, number of parameters in millions and time needed to achieve the lowest test error in hours for the different models. Best result is highlighted in bold.

Time Dimension With regard to the time dimension, the experiments showed that the best backbone, ResNet18 (2+1)D, leverages frames up to 1 second away in order to predict the pose. In particular, the performances are slightly lower with 15 or 45 frames. The difference is however in the order of 1-2 millimeters, so it could be useful to feed only 15 frames in order to reduce training and inference time. Table 6.4 shows the average test error for the different clip lengths.

With respect to the different sampling steps, step 2 seems to represent the right compromise between subsampling the input size and preserving

Measure	15 Frames	30 Frames	45 Frames
Avg Error [cm]	12.9	**12.76**	12.81

Table 6.4: Average test error in centimeters for the different input clip lengths. Best result is highlighted in bold.

the useful information. Indeed, both with step 1 and step 3 it is possible to see a decrease in terms of precision. While the decrease with step 3 is easily attributed to the lower resolution, the decrease with step 1 is harder to motivate. We suppose it could be due to the network which tends to overfit focusing too much on the time dimension in place of the spatial ones. Table 6.4 shows the average test error for the different sampling step lengths.

Measure	Step 1	Step 2	Step 3
Avg Error [cm]	13.53	**12.76**	13.42

Table 6.5: Average test error in centimeters for the different sampling steps. Best result is highlighted in bold.

Chapter 7

R3D-Pose

After an evaluation of the current state-of-art together with other possible architectures, we also developed an ad hoc model, which we called **R**adar-based **3D Pose** Estimation (**R3D-Pose**), which is able to consistently outperform the other methods. Due to the results obtained in the ablation study, we opted for a ResNet [26] design as the base backbone. The adopted architecture and its modifications will be later described in details. Moreover, in order to achieve a better generalization we introduced changes not only in the architecture but also in the way the network is trained:

- AdamW [38] is adopted as optimizer in place of the classic Adam [37];
- Weight Decay is strongly increased to from 10^{-5} to 10^{-1};
- Data Augmentation is applied to the training labels in the form of Gaussian noise;
- Huber [69] loss is minimized in place of the standard MSE.

These modifications highly enhance the generalization capability of the network reducing the overfitting and leading to more accurate predictions. Moreover, the adoption of Huber loss reduces the weight of outliers caused by errors in the 3D reconstruction. Indeed, despite resulting in an unbiased estimator for the mean, MSE tends to be dominated by outliers while Huber is much less sensitive to them. Moreover, at the best of our knowledge, this is the first time Data Augmentation is applied in radar-based Pose Estimation.

Architecture The architecture of a standard ResNet is the following one:

- a first Convolutional layer with 64 filters, kernel size 7x7, stride 2 and padding 3, followed by Batch Norm layer and ReLU [21] activation;
- a Max Pooling layer with kernel size 3x3, stride 2 and padding 1;
- four groups of Residual Blocks or Residual Bottlenecks with number of filters doubled after each group and stride 2 for all the groups but the first whose stride is instead 1;
- a Global Average Pooling layer;
- a final FC layer acting as classifier or regressor.

Fig. 7.1 shows the differences between a standard Residual Block and a Bottleneck at the same depth in the network.

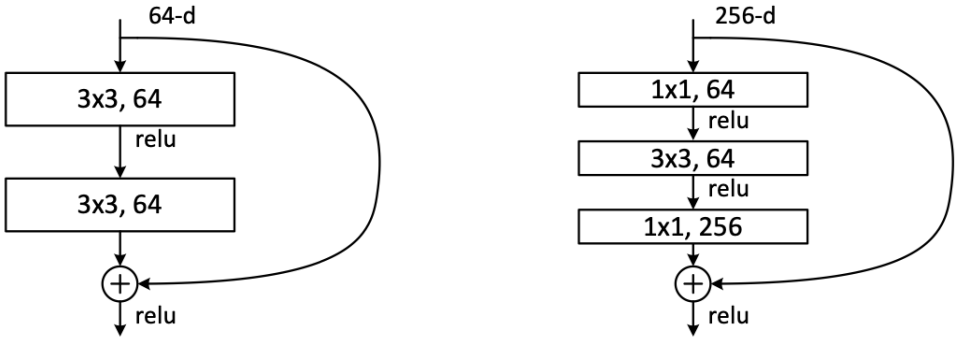

Figure 7.1: Left: Residual Block. Right: Residual Bottleneck. The image is taken from [26].

After various experiments we adopted a simple ResNet34, with all the layers but the last Residual Block modified from 2D to 3D and a reduced number of filters. Somehow with the applied modifications it performs better than even ResNet18 (2+1)D [62], despite being the latter the best performing model in the ablation study.

First, we modified the first convolution following the design in [62]: the kernel size is 3x7x7, stride 1x2x2 and padding 1x3x3. Then, the Max Pooling operation is performed only on the time dimension, since with radar heatmaps we do not need scale equivariance on the spatial dimensions, while on the time one scale equivariance makes the model invariant with respect to the speed of

movements. Moreover, the Global Average Pooling layer is removed and the output of the last Convolution of each branch are directly concatenated. We believe the translation invariance bias of the Global Pooling may worsen the results making our regressor insensitive to shifts in the feature maps. Before feeding the concatenation to the final Fully Connected layer, we applied another Residual Block with stride 2 in order to downsample the feature maps entering the Fully Connected layer, which would otherwise explode without the dimensional reduction performed by the Global Pooling. The absence of a Global Pooling is perhaps the reason why the 34-layer network performs better than the 50-layer one, whose Convolutions output a tensor 4 times larger. Moreover Convolutional layers are initialized with a Kaiming Normal [1], which has been specifically developed for ReLU-like activations, instead of the default LeCun Normal [70]. We also tested different activation functions replacing the ReLU: Randomized Rectified Linear Unit (RReLU) [71], Parametric (PReLU) [1], Gaussian Error Linear Unit (GELU) [72] and Exponential Linear Unit (ELU) [73]. After various experiments, we adopted GELU which is a new type of high performance activation function which further enhances the generalization capability of the model by "weighting inputs by their values, rather than gating inputs by their sign as in ReLUs".

7.1 Implementation and Training Details

As for the Baselines and the ablation study, we adopted Cosine Annealing with Warm Restarts [68] as scheduler with initial learning rate 10^{-2}, $t_0 = 2$ and $t_{mult} = 2$. The batch-size is 32. The model is implemented by means of the PyTorch [44] framework and trained using two NVIDIA GTX 1070 for a maximum of 24 hours or 62 epochs (i.e. 4 Warm Restarts).

We made sure that the network does not see any of the test data at training time: as before, all the scenes are used for the training phase but scene 8 which is the one which the model is tested on.

7.2 Results

R3D-Pose results are far more accurate than the previous models, achieving an average test error of 11.18 centimeters. In particular, it achieves an error reduction of 30% circa with respect to PoseCapture and 70-80% with respect to mm-Pose. The major drawback is due to the higher number of parameters

which is increased of 25% with respect to the base ResNet34 but it still converges circa 5 times faster than the state-of-art models. Table 7.1 contains the average test error, the number of parameters and the needed training time for the Baselines, the ResNets and R3D-Pose.

Model	Avg Error [cm]	Parameters [M]	Time [h]
mm-Pose	40.8	0.28	12.3
PoseCapture	15.13	1.34	16.75
ResNet18	13.88	1.05	5.5
ResNet34	13.82	2	2.5
ResNet50	13.11	1.49	15.3
ResNet18 (2+1)D	12.76	0.99	15.25
R3D-Pose (Ours)	**10.65**	2.43	22

Table 7.1: Average test error in centimeters, number of parameters in millions and time needed to achieve the lowest test error in hours for the different models. The best result is highlighted in bold.

Relatively to the error per keypoint we can notice the same behaviour observed before, with the limbs showing a significantly larger error with respect to the other body parts. However, on average R3D-Pose achieves better results on each single keypoint and the difference in performance is even more emphasized in the limbs. Tables 7.2 and 7.3 show the average and median test error per keypoint for the two Baselines and R3D-Pose.

Keypoint	mm-Pose	PoseCapture	R3D-Pose (Ours)
$Head$	38.26 ± 23.3	10.81 ± 8.25	$\mathbf{8.3 \pm 7.98}$
$Neck$	35.62 ± 23.05	8.74 ± 8.08	$\mathbf{6.51 \pm 7.78}$
$Shoulder_R$	40.56 ± 23.26	11.71 ± 9.51	$\mathbf{8.29 \pm 9.57}$
$Elbow_R$	48.44 ± 23.75	19.93 ± 12.57	$\mathbf{13.73 \pm 12.51}$
$Wrist_R$	55.75 ± 23.81	28.86 ± 18.29	$\mathbf{20.02 \pm 16.77}$
$Shoulder_L$	35.88 ± 22.38	10.80 ± 8.17	$\mathbf{7.47 \pm 7.55}$
$Elbow_L$	42.31 ± 22.25	19.33 ± 10.9	$\mathbf{12.95 \pm 10.6}$
$Wrist_L$	50.24 ± 24.49	29.8 ± 16.88	$\mathbf{21.1 \pm 15.94}$
Hip_R	38.44 ± 23.21	9.8 ± 9.03	$\mathbf{7.2 \pm 9.35}$
$Knee_R$	39.7 ± 23.64	12.1 ± 11.13	$\mathbf{8.43 \pm 10.47}$
$Ankle_R$	39.56 ± 24.39	15.25 ± 12.31	$\mathbf{10.5 \pm 10.11}$
Hip_L	34.96 ± 22.81	8.89 ± 7.73	$\mathbf{6.31 \pm 7.55}$
$Knee_L$	35.9 ± 23.62	11.34 ± 10.34	$\mathbf{8.07 \pm 9.55}$
$Ankle_L$	35.6 ± 24.09	14.5 ± 11.58	$\mathbf{10.2 \pm 9.92}$
$Mean$	40.8	13.88	**10.65**

Table 7.2: Average test error ± standard deviation per keypoint. All values are in centimeters. Best results are highlighted in bold.

Keypoint	mm-Pose	PoseCapture	R3D-Pose (Ours)
$Head$	32.94 ± 35.26	9.06 ± 6.9	$\mathbf{6.23 \pm 5.08}$
$Neck$	29.76 ± 33.32	6.71 ± 5.82	$\mathbf{4.73 \pm 3.83}$
$Shoulder_R$	36.77 ± 32.4	9.71 ± 7.56	$\mathbf{5.89 \pm 4.87}$
$Elbow_R$	46.25 ± 36.74	17.59 ± 13.13	$\mathbf{9.97 \pm 8.98}$
$Wrist_R$	52.23 ± 36.74	25.19 ± 21.29	$\mathbf{14.8 \pm 15.09}$
$Shoulder_L$	30.07 ± 34.5	9.29 ± 7.42	$\mathbf{5.76 \pm 4.78}$
$Elbow_L$	39.53 ± 36.61	16.9 ± 12.66	$\mathbf{10.03 \pm 9.61}$
$Wrist_L$	45.47 ± 36.1	26.83 ± 20.87	$\mathbf{16.77 \pm 16.2}$
Hip_R	33.76 ± 34.82	7.81 ± 6.4	$\mathbf{4.8 \pm 3.89}$
$Knee_R$	35.14 ± 37.16	9.24 ± 7.83	$\mathbf{5.44 \pm 5.05}$
$Ankle_R$	32.65 ± 37.96	12.75 ± 10.61	$\mathbf{7.74 \pm 7.96}$
Hip_L	28.44 ± 35.01	7.18 ± 5.91	$\mathbf{4.39 \pm 3.57}$
$Knee_L$	30.35 ± 38.49	8.98 ± 7.4	$\mathbf{5.5 \pm 4.54}$
$Ankle_L$	28 ± 39	11 ± 10.48	$\mathbf{7.41 \pm 6.62}$

Table 7.3: Median test error ± interquartile range per keypoint. All values are in centimeters. Best results are highlighted in bold.

Chapter 8

Conclusions and Future Works

The poor environmental robustness of RGB based models causes these systems to be often unsuitable for actual applications in safety-critical scenarios. In particular a Pose Estimation system is likely to fail in scarce light conditions, or if the signal is perturbed by weather phenomena such as intense rain or fog. Radar technology can indeed mitigate this issue by providing robust environmental perception in a large number of adverse conditions. Towards this objective, in this thesis I collected a dataset with radar data and 3D ground truth skeletons and after an evaluation of state-of-art techniques I proposed a custom architecture, R3D-Pose, able to consistently outperform them by a large margin with an error decrease of 70-80% with respect to mm-Pose and 20-30% with respect to PoseCapture. This work is a step forward in the development of this promising research field. However, despite the positive results achieved in this work, there are still drawbacks in the application of radars systems for Pose Estimation purposes. In particular, the limited spatial resolution represents the major drawback in the development of high precision models, especially for the localization of body parts with a small reflective surface. Radars with higher resolution could in theory fill the gap between radar-based and RGB-based models in standard conditions, where RGB-based models still outperform radar-based solutions.

Regarding future works, there is a series of possible paths to be taken to extend both the dataset and the architecture. First of all, the dataset could be expanded to a multi-person scenario and to different environments in order to obtain a more realistic setting. Moreover, by labeling the various subjects the

dataset would be suitable to perform Person Identification, while by labeling the different actions it can be adopted for Action Recognition. Finally, RGB and radar signals can be jointly exploited in a Sensor Fusion Fashion. In particular, this last path is quite interesting to explore since radars solve the two major issues of a RGB sensor, the lack of depth information and environmental robustness, while the latter could provide a better spatial resolution on the height and width dimensions. Moreover, a mutual help in the avoiding of false positives can occur: a radar would not see at all the image of a person in a picture or reflected on a window pane and an RGB sensor would not fail due to ghost reflections caused by multipaths.

Appendix A

Appendix

Object hooker function to read the radar files:

```
    # adapted from: https://stackoverflow.com/questions
    /27909658/json-encoder-and-decoder-for-complex-numpy-arrays

    import numpy as np
    import base64

    def json_numpy_obj_hook(dct):
    """
    Decodes a previously encoded numpy ndarray
    with proper shape and dtype
    :param dct: (dict) json encoded ndarray
    :return: (ndarray) if input was an encoded ndarray
    """
    if isinstance(dct, dict) and '__ndarray__' in dct:
        data = base64.b64decode(dct['__ndarray__'].encode())
        return np.frombuffer(data, dct['dtype']).reshape(dct['
    shape'])
    return dct
```

Bibliography

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015. arXiv: 1502.01852 [cs.CV] (cit. on pp. 1, 43, 52).

[2] David Silver et al. *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*. 2017. arXiv: 1712.01815 [cs.AI] (cit. on p. 1).

[3] Azalia Mirhoseini et al. «A graph placement methodology for fast chip design». In: *Nature* 594.7862 (2021), pp. 207–212 (cit. on p. 1).

[4] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. «Deepface: Closing the gap to human-level performance in face verification». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1701–1708 (cit. on p. 1).

[5] Noam Brown and Tuomas Sandholm. «Superhuman AI for heads-up no-limit poker: Libratus beats top professionals». In: *Science* 359.6374 (2018), pp. 418–424 (cit. on p. 1).

[6] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. «RMPE: Regional Multi-person Pose Estimation». In: *ICCV*. 2017 (cit. on pp. 1, 21, 32, 38).

[7] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. «OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019) (cit. on pp. 1, 21, 24, 32).

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. «Mask R-CNN». In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322 (cit. on pp. 1, 21).

[9] Hanyue Tu, Chunyu Wang, and Wenjun Zeng. «VoxelPose: Towards Multi-Camera 3D Human Pose Estimation in Wild Environment». In: *European Conference on Computer Vision (ECCV)*. 2020 (cit. on pp. 1, 21, 22).

[10] Shunqiao Sun, Athina P. Petropulu, and H. Vincent Poor. «MIMO Radar for Advanced Driver-Assistance Systems and Autonomous Driving: Advantages and Challenges». In: *IEEE Signal Processing Magazine* 37.4 (2020), pp. 98–117. DOI: 10.1109/MSP.2020.2978507 (cit. on pp. 2, 4, 5).

[11] Sujeet Milind Patole, Murat Torlak, Dan Wang, and Murtaza Ali. «Automotive radars: A review of signal processing techniques». In: *IEEE Signal Processing Magazine* 34.2 (2017), pp. 22–35. DOI: 10.1109/MSP.2016.2628914 (cit. on p. 4).

[12] Fadel M. Adib, C. Hsu, Hongzi Mao, D. Katabi, and F. Durand. «Capturing the human figure through a wall». In: 2015 (cit. on pp. 9, 24).

[13] Tianhong Li, Lijie Fan, Mingmin Zhao, Yingcheng Liu, and Dina Katabi. «Making the Invisible Visible: Action Recognition Through Walls and Occlusions». In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 872–881. DOI: 10.1109/ICCV.2019.00096 (cit. on pp. 9, 24).

[14] Mingmin Zhao, Yonglong Tian, Hang Zhao, Mohammad Abu Alsheikh, Tianhong Li, Rumen Hristov, Zachary Kabelac, Dina Katabi, and Antonio Torralba. «RF-Based 3D Skeletons». In: *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. SIGCOMM '18. Budapest, Hungary: Association for Computing Machinery, 2018, pp. 267–281. ISBN: 9781450355674. DOI: 10.1145/3230543.3230579. URL: https://doi.org/10.1145/3230543.3230579 (cit. on pp. 9, 24, 28, 34).

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. «Gradient-based learning applied to document recognition». In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on p. 10).

[16] Alex Krizhevsky. «Learning Multiple Layers of Features from Tiny Images». In: (2009) (cit. on p. 10).

[17] O. Russakovsky et al. «ImageNet Large Scale Visual Recognition Challenge». In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y (cit. on pp. 10, 14).

[18] F. Rosenblatt. «The perceptron: A probabilistic model for information storage and organization in the brain.» In: *Psychological Review* 65.6 (1958), pp. 386–408. ISSN: 0033-295X. DOI: 10.1037/h0042519. URL: http://dx.doi.org/10.1037/h0042519 (cit. on p. 10).

[19] George Cybenko. «Approximation by superpositions of a sigmoidal function». In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314 (cit. on p. 11).

[20] A. Karpathy. *Convolutional Neural Networks (CNN / ConvNets)*. 2018. URL: http://cs231n.github.io/classification/ (cit. on pp. 11, 17).

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». In: (2012). Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf (cit. on pp. 11, 16, 51).

[22] H. Ide and T. Kurita. «Improvement of learning for CNN with ReLU activation by sparse regularization». In: *International Joint Conference on Neural Networks (IJCNN)* (2017), pp. 2684–2691. DOI: 10.1109/IJCNN.2017.7966185 (cit. on p. 11).

[23] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. «Backpropagation Applied to Handwritten Zip Code Recognition». In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541 (cit. on pp. 12, 14).

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». In: *Commun. ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: https://doi.org/10.1145/3065386 (cit. on pp. 12, 14).

[25] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV] (cit. on pp. 12, 44, 46).

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV] (cit. on pp. 12, 14, 18, 24, 44, 48, 50, 51).

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL] (cit. on p. 12).

[28] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV] (cit. on p. 12).

[29] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. *Training data-efficient image transformers distillation through attention*. 2021. arXiv: 2012.12877 [cs.CV] (cit. on p. 12).

[30] Sepp Hochreiter and Jürgen Schmidhuber. «Long short-term memory». In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cit. on p. 12).

[31] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. arXiv: 1406.1078 [cs.CL] (cit. on p. 12).

[32] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. «The graph neural network model». In: *IEEE transactions on neural networks* 20.1 (2008), pp. 61–80 (cit. on p. 12).

[33] David E. Rumelhart and James L. McClelland. «Learning Internal Representations by Error Propagation». In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. 1987, pp. 318–362 (cit. on p. 12).

[34] Léon Bottou. «Large-Scale Machine Learning with Stochastic Gradient Descent». In: (2010). Ed. by Yves Lechevallier and Gilbert Saporta, pp. 177–186 (cit. on p. 13).

[35] John Duchi, Elad Hazan, and Yoram Singer. «Adaptive subgradient methods for online learning and stochastic optimization.» In: *Journal of machine learning research* 12.7 (2011) (cit. on p. 13).

[36] Matthew D. Zeiler. *ADADELTA: An Adaptive Learning Rate Method*. 2012. arXiv: 1212.5701 [cs.LG] (cit. on p. 13).

[37] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: `1412.6980 [cs.LG]` (cit. on pp. 13, 41, 45, 50).

[38] Ilya Loshchilov and Frank Hutter. «Fixing Weight Decay Regularization in Adam». In: *CoRR* abs/1711.05101 (2017). arXiv: `1711.05101`. URL: `http://arxiv.org/abs/1711.05101` (cit. on pp. 13, 50).

[39] Shaeke Salman and Xiuwen Liu. *Overfitting Mechanism and Avoidance in Deep Neural Networks*. 2019. arXiv: `1901.06566 [cs.LG]` (cit. on p. 14).

[40] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. «Dropout: A Simple Way to Prevent Neural Networks from Overfitting». In: *Journal of Machine Learning Research* (2014), pp. 1929–1958 (cit. on p. 14).

[41] Anders Krogh and John A. Hertz. «A Simple Weight Decay Can Improve Generalization». In: *Proceedings of the 4th International Conference on Neural Information Processing Systems*. NIPS'91. Denver, Colorado: Morgan Kaufmann Publishers Inc., 1991, pp. 950–957. ISBN: 1558602224 (cit. on p. 14).

[42] Connor Shorten and T. Khoshgoftaar. «A survey on Image Data Augmentation for Deep Learning». In: *Journal of Big Data* 6 (2019), pp. 1–48 (cit. on p. 14).

[43] David H Hubel and Torsten N Wiesel. «Receptive fields, binocular interaction and functional architecture in the cat's visual cortex». In: *The Journal of physiology* 160.1 (1962), pp. 106–154 (cit. on p. 14).

[44] Sam Gross and Michael Wilber. *Training and investigating Residual Nets*. 2016. URL: `http://torch.ch/blog/2016/02/04/resnets.html` (cit. on pp. 15, 45, 52).

[45] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. 2021. arXiv: `2104.13478 [cs.LG]` (cit. on pp. 15, 16).

[46] Dominik Scherer, Andreas C. Müller, and Seven Behnke. «Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition». In: (2010) (cit. on p. 17).

[47] Daniel Müllner. *Modern hierarchical, agglomerative clustering algorithms*. 2011. arXiv: `1109.2378 [stat.ML]` (cit. on p. 18).

[48] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. «A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise». In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231 (cit. on p. 19).

[49] Joseph Redmon and Ali Farhadi. «YOLOv3: An Incremental Improvement». In: *ArXiv* abs/1804.02767 (2018) (cit. on p. 21).

[50] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter Gehler, and Bernt Schiele. *DeepCut: Joint Subset Partition and Labeling for Multi Person Pose Estimation*. 2016. arXiv: 1511.06645 [cs.CV] (cit. on p. 21).

[51] Zhengyou Zhang. «Microsoft Kinect Sensor and Its Effect». In: *IEEE MultiMedia* 19.2 (2012), pp. 4–10. DOI: 10.1109/MMUL.2012.24 (cit. on pp. 22, 28).

[52] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. cite arxiv:1405.0312Comment: 1) updated annotation pipeline description and figures; 2) added new section describing datasets splits; 3) updated author list. 2014. URL: http://arxiv.org/abs/1405.0312 (cit. on p. 22).

[53] Hanbyul Joo et al. «Panoptic Studio: A Massively Multiview System for Social Interaction Capture». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017) (cit. on p. 22).

[54] Mingmin Zhao, Tianhong Li, Mohammad Abu Alsheikh, Yonglong Tian, Hang Zhao, Antonio Torralba, and Dina Katabi. «Through-Wall Human Pose Estimation Using Radio Signals». In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7356–7365. DOI: 10.1109/CVPR.2018.00768 (cit. on p. 24).

[55] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. *SoundNet: Learning Sound Representations from Unlabeled Video*. 2016. arXiv: 1610.09001 [cs.CV] (cit. on p. 24).

[56] Mingmin Zhao, Yingcheng Liu, Aniruddh Raghu, Hang Zhao, Tianhong Li, Antonio Torralba, and Dina Katabi. «Through-Wall Human Mesh Recovery Using Radio Signals». In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 10112–10121. DOI: 10.1109/ICCV.2019.01021 (cit. on p. 24).

[57] Arindam Sengupta, Feng Jin, Renyuan Zhang, and Siyang Cao. «mm-Pose: Real-Time Human Skeletal Posture Estimation Using mmWave Radars and CNN». In: *IEEE Sensors Journal* 20.17 (2020), pp. 10032–10044 (cit. on pp. 24, 28, 40).

[58] Guangzheng Li, Ze Zhang, Hanmei Yang, Jin Pan, Dayin Chen, and Jin Zhang. «Capturing Human Pose Using mmWave Radar». In: *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. 2020, pp. 1–6. DOI: 10.1109/PerComWorkshops48775.2020.9156151 (cit. on pp. 24, 40).

[59] L. Sigal, A. Balan, and M. J. Black. «HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion». In: *International Journal of Computer Vision* 87.1 (Mar. 2010), pp. 4–27 (cit. on p. 28).

[60] D.L. Mills. «Internet time synchronization: the network time protocol». In: *IEEE Transactions on Communications* 39.10 (1991), pp. 1482–1493. DOI: 10.1109/26.103043 (cit. on p. 31).

[61] Suramya Tomar. «Converting video formats with FFmpeg». In: *Linux Journal* 2006.146 (2006), p. 10 (cit. on p. 39).

[62] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. *A Closer Look at Spatiotemporal Convolutions for Action Recognition*. 2018. arXiv: 1711.11248 [cs.CV] (cit. on pp. 44, 48, 51).

[63] Christoph Feichtenhofer. *X3D: Expanding Architectures for Efficient Video Recognition*. 2020. arXiv: 2004.04730 [cs.CV] (cit. on p. 44).

[64] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. *Densely Connected Convolutional Networks*. 2018. arXiv: 1608.06993 [cs.CV] (cit. on pp. 44, 46).

[65] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. *MnasNet: Platform-Aware Neural Architecture Search for Mobile*. 2019. arXiv: 1807.11626 [cs.CV] (cit. on pp. 44, 46).

[66] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV] (cit. on p. 44).

[67] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2019. arXiv: 1801.04381 [cs.CV] (cit. on p. 44).

[68] Ilya Loshchilov and Frank Hutter. «SGDR: Stochastic Gradient Descent with Restarts». In: *CoRR* abs/1608.03983 (2016). arXiv: 1608.03983. URL: http://arxiv.org/abs/1608.03983 (cit. on pp. 45, 52).

[69] Peter J Huber. «Robust estimation of a location parameter». In: *Breakthroughs in statistics*. Springer, 1992, pp. 492–518 (cit. on p. 50).

[70] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. «Efficient backprop». In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48 (cit. on p. 52).

[71] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. *Empirical Evaluation of Rectified Activations in Convolutional Network*. 2015. arXiv: 1505.00853 [cs.LG] (cit. on p. 52).

[72] Dan Hendrycks and Kevin Gimpel. «Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units». In: *CoRR* abs/1606.08415 (2016). arXiv: 1606.08415. URL: http://arxiv.org/abs/1606.08415 (cit. on p. 52).

[73] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*. 2016. arXiv: 1511.07289 [cs.LG] (cit. on p. 52).