

Development of a Deep Learning Model for 3D Human Pose Estimation in Monocular Videos

Agnė Grinciūnaitė

Master's Degree Thesis



VILNIUS GEDIMINAS
TECHNICAL UNIVERSITY
FACULTY OF FUNDAMENTAL SCIENCES



VILNIUS GEDIMINAS TECHNICAL UNIVERSITY
FACULTY OF FUNDAMENTAL SCIENCES
DEPARTMENT OF GRAPHICAL SYSTEMS

Agnė Grinciūnaitė

Development of a Deep Learning Model for 3D Human Pose Estimation in Monocular Videos

Master's degree Thesis

Information Technologies study programme, state code 621E14004
Multimedia Information Systems specialization
Informatics Engineering study field

Vilnius, 2016



The work in this thesis was supported by Vicar Vision. Their cooperation is hereby gratefully acknowledged.



Copyright © Department of Graphical Systems
All rights reserved.

VILNIUS GEDIMINAS TECHNICAL UNIVERSITY
FACULTY OF FUNDAMENTAL SCIENCES
DEPARTMENT OF GRAPHICAL SYSTEMS

APPROVED BY
Head of Department

(Signature)

(Name, Surname)

(Date)

Agnė Grinciūnaitė

**Development of a Deep Learning Model for 3D
Human Pose Estimation in Monocular Videos**

Master's degree Thesis

Information Technologies study programme, state code 621E14004
Multimedia Information Systems specialization
Informatics Engineering study field

Supervisor

(Title, Name, Surname)

(Signature)

(Date)

Consultant

(Title, Name, Surname)

(Signature)

(Date)

Consultant

(Title, Name, Surname)

(Signature)

(Date)

Vilnius, 2016

Abstract

There exists a visual system which can easily recognize and track human body position, movements and actions without any additional sensing. This system has the processor called brain and it is competent after being trained for some months. With a little bit more training it is also able to apply acquired skills for more complicated tasks such as understanding inter-personal attitudes, intentions and emotional states of the observed moving person. This system is called a human being and is so far the most inspirational piece of art for today's artificial intelligence creators.

The most impressive results of complex computer vision and machine learning tasks were recently achieved by applying various deep learning methods. It is amazing how fast deep neural networks became popular and broadly used not only in research community but also in commercial world. The major impact was made by convolutional neural networks being able to beat some challenges in computer vision by quite a big margin and attract everybody's attention. These networks are motivated by the known neurophysiology of the brain and its functional properties required for cognition.

The goal of this thesis is to explore the capabilities of convolutional neural network to deal with easily manageable task for human-beings - perceiving other human's location in space-time from the perspective of the viewer. New approach of incorporating 3D convolutions to extract valuable features from motion data captured by monocular video camera and directly regress to joint positions in 3D camera coordinate space is used. This research shows the ability of such a network to achieve state of the art results on selected dataset.

The achieved results imply that improved realization could possibly be used in real-world applications such as human-computer interaction, augmented and virtual reality, robotics, surveillance, smart homes, etc.

Anotacija

Egzistuoja tokia vaizdo apdorojimo sistema, kuri geba lengvai atpažinti ir sekti žmogaus kūno poziciją, judesius ir veiksmus be jokių papildomų pojūčių. Šios sistemos procesorius tampa kompetentingas vos per kelis apmokymo mėnesius ir yra vadinamas smegenimis. Pasimokęs šiek tiek ilgiau, jis taip pat sugeba savo įgūdžius panaudoti sudėtingesnėms užduotims, pavyzdžiui, stebint judantį žmogų suprasti jo santykį su aplinka, asmeninius ketinimus bei emocinę būklę. Ši sistema yra vadinama žmogumi ir tai yra vienas labiausiai šių dienų dirbtinio intelekto kūrėjus įkvepiančių meno kūrinių.

Neseniai pasiekti rezultatai kompiuterinės vizijos ir sistemos mokymosi srityje naudojant įvairius giliojo mokymosi metodus išties daro įspūdį. Neįtikėtinais greitai gilieji neuroniniai tinklai tapo populiarūs ir plačiai naudojami ne tik mokslo bendruomenėje, bet ir komerciniame pasaulyje. Didžiausią įtaką tam turėjo būtent konvoliuciniai neuroniniai tinklai, dėl kurių buvo įveikti keli didžiausių kompiuterinės vizijos iššūkių. Tai ir pritraukė visų dėmesį. Šie neuroniniai tinklai yra įkvėpti žinomos smegenų neurofiziologijos ir jų funkcinėmis savybėmis, kurios reikalingos kognityvumui.

Šio darbo tikslas yra ištirti, ar konvoliucinis neuroninis tinklas gali susidoroti su lengvai žmogui „įkandama“ užduotimi – iš savo matymo perspektyvos suvokti kito žmogaus poziciją erdvėlaikyje. Šiuo darbu yra pristatomas naujas būdas inkorporuojant trimates konvoliucijas išgauti vertingas savybes iš judesio informacijos, užfiksuotos videomedžiagoje, ir tiesiogiai išvesti žmogaus kūno taškų pozicijas trimatėje kameros koordinačių sistemoje. Tyrimas parodo, kad siūloma neuroninio tinklo realizacija leidžia pasiekti geriausius rezultatus su pasirinktos duomenų bazės duomenimis.

Pasiekti rezultatai leidžia manyti, kad patobulinta realizacija galėtų būti sėkmingai taikoma tokiose srityse kaip žmogaus ir kompiuterio sąveika, papildyta ir virtuali realybė, robotika, sekimo technologijos, išmanieji namai ir pan.

Table of Contents

Acknowledgements	vii
1 Introduction	1
1-1 Thesis Objective and Research Questions	2
1-2 Report Structure	3
2 Theoretical Basis	4
2-1 Multi-Layer Neural Network	4
2-2 Convolutional Neural Network	6
3 Related Work	8
3-1 Classic CNN Architectures	8
3-2 Pose Regression CNN Architectures	10
3-3 Multi-task CNN Architectures	13
3-4 3D CNN Architectures	15
4 Dataset	18
4-1 Overview	19
4-1-1 Berkeley MHAD	19
4-1-2 Cornell Activity	20
4-1-3 CMU-MMAC	20
4-1-4 Human3.6M	20
4-1-5 HumanEva	21
4-1-6 INRIA RGB-D	21
4-1-7 MPI08	21

4-2	Human3.6M Dataset	21
4-2-1	Subjects	22
4-2-2	Actions	23
4-2-3	Video Data	24
4-2-4	Pose Data	24
4-2-5	Evaluation and Error Measure	25
4-3	Data Preprocessing	25
5	Three Dimensional Convolutional Neural Network	28
5-1	Data Sampling	28
5-2	Network's Input and Output Data	29
5-3	CNN Building	30
5-3-1	Activation Functions	31
5-3-2	Normalization Layer	32
5-3-3	Convolutional Layer	32
5-3-4	Pooling Layer	34
5-3-5	Fully Connected and Output Layers	35
5-3-6	3D CNN Architecture	35
5-4	CNN Training	36
5-4-1	Parameter Initialization	36
5-4-2	Cost Function	36
5-4-3	Learning Algorithm and Optimizations	37
5-4-4	Regularization	37
6	Experiments and Results	39
6-1	CNN Building Experiments	39
6-2	Output Tuning	40
6-3	Results	41
7	Conclusions	45
	Glossary	54
	List of Acronyms	54
	List of Symbols	55

List of Figures

2-1	Biological and artificial neuron	4
2-2	Schematic of a hierarchical sequence of categorical representations	6
3-1	Classic LeNet-5 Architecture	8
3-2	Krizhevsky's CNN Architecture	10
3-3	CNN-based regressor and refiner architectures	10
3-4	CNN of Heat-Map Models	11
3-5	Temporal Pose CNN	12
3-6	Deep expert pooling architecture for pose estimation	13
3-7	CNN architecture for binary classification	13
3-8	CNN architecture for a joint detection and regression tasks	14
3-9	Dual-Source CNN architecture	15
3-10	First 3D CNN architecture for action recognition	16
3-11	Reconfigurable 3D CNN architecture for action recognition	17
4-1	Subjects in Human3.6M dataset	22
4-2	Set of actions in Human3.6M dataset	23
4-3	Skeleton joints locations	26
4-4	Image preprocessing	27
4-5	Preprocessed data distribution by subject and action	27
5-1	Example of 3D Convolution	33
5-2	Example of 3D Max Pooling	34
5-3	Proposed 3D CNN Architecture	35
6-1	Selected good and bad results visualization	44

List of Tables

4-1	Publicly available datasets overview	19
4-2	Distribution of Human3.6M poses per scenario and type of action	24
5-1	Sampling parameters used for different experiments	29
6-1	Selected experimental CNN building steps	40
6-2	Results of all experiments evaluated on Human3.6M dataset's server	41
6-3	Results comparing with state of the art	42
6-4	Results comparison with recent work	43

Acknowledgements

I am very happy I got an opportunity to work on this thesis which actually started with the idea of Marten den Uyl. I would like to thank him for letting me give it a try.

I really enjoyed working at Vicar Vision and being supervised by Emrah and Amogh who guided me through all the process giving me the best tips and tricks at the right moments. I will never forget our exciting discussions about deep learning and the future of artificial intelligence. And thanks for reminding me that “Everything is going to be all right”.

Thanks to all the colleagues at Singel 160. It was a great pleasure working with them.

Big thanks to my friends Tomas, Viktoras, Eva and Alex for encouragement, positiveness, moral support and always cheering me up. Also all the people I met in Amsterdam who made my thesis period enjoyable.

I wouldn't have done it without limitless unconditional support and trust of my parents Violeta and Egidijus. Special thanks to them and my sister Inga who has always been my role model.

Vilnius Gediminas Technical University
June 7, 2016

Agnė Grinciūnaitė

“Intelligence is the art of good guesswork.”

— *Horace Basil Barlow*

Chapter 1

Introduction

Almost 40 years ago psychologist M. R. Jones stated that “humans are built to detect real-world structure by detecting changes along physical dimensions (i.e. contrasting values) and representing these changes as relations (i.e. differences) along subjective dimensions. Because change can only occur over time, it makes sense that time somehow be incorporated into a definition of structure” [1]. Ten years later Dr. Jennifer J. Freyd argued that temporal dimension is necessary and is coupled with spatial dimensions in human mental representations [2].

With the increased usage of Functional Magnetic Resonance Imaging (fMRI) it became possible to study human perception of motion by simultaneously monitoring the observer’s cortical activity. Since then we were able to get insight of how human brain processes motion information ([3], [4], [5]). Although it is still a challenge to explain motion perception from a computational neuroscience perspective, some of the main principles were successfully applied in today’s deep learning applications.

Breakthrough in the field of machine learning related to bio-inspired models has made it possible to model structured and abstract representations within multi-layered hierarchical networks. Searching the parameter space of deep architectures is still a difficult task, but their power in several object recognition and classification tasks has proven to be very promising if large amount of training data is available.

This thesis deals with a longstanding task in computer vision - human pose, represented by 3D joint positions, estimation in monocular videos. The challenges of this task include high dimensionality of the data, large variability of poses, motions and appearance, self occlusions and changes in illumination.

There were a number of studies carried out in human pose estimation field using different generative and discriminative approaches. However, most of the published works deal with still single ([6], [7], [8]) or depth images ([9], [10]). Also most often it is attempting to

estimate 2D full ([11], [12], [13]), upper body ([14], [15], [16], [17]) or single ([18], [19]) joint position in the image plane. Many approaches incorporate 2D pose estimations or features to then retrieve 3D poses ([20], [21], [22], [23]).

This work is built on the idea of necessity to involve time dimension in order to understand spacial location of the moving person. Successful attempt to accurately estimate space-time human body positions using only temporal video information would lead to effective applications in areas such as visual surveillance, human action and emotional state recognition, human-computer interfaces, video coding, ergonomics, video indexing and retrieval, human action prediction and others.

1-1 Thesis Objective and Research Questions

The main objective of this thesis is to build a discriminative 3D Convolutional Neural Network (CNN) model able to directly estimate human body pose in camera coordinate space using only Red-Green-Blue (color model based on additive color primaries) (RGB) video data. This section describes the main questions that will be researched in the course of achieving this objective.

The success of implementing well performing deep architecture largely depends on the correct hyper-parameter selection. It can be done manually or automatically using grid search, random search [24] or more sophisticated hyper-parameter optimization methods ([25], [26]). Due to the high computational cost of automatic hyper-parameter selection, all the choices have to follow the manual approach regarding this thesis. Therefore, one of the research questions is: *How well the model can cope with the defined task by using manually selected hyper-parameters based on theoretical knowledge and experience of others?*

It is known that deep learning models achieve better results when trained on more data. It can be stated that the lack of annotated video data was one of the main reasons why there are not enough deep learning related research done regarding formulated problem. This leads to the following question: *Are the existing publicly available annotated datasets sufficient for deep learning based experiments related to the objective of this thesis?*

This thesis aims to build a 3D CNN model coping with the task without using additional algorithms or processing steps slowing down application's speed. CNNs were successfully applied in classification tasks, such as human action recognition ([27], [28], [29], [30]), crowd behavior recognition ([31]), hand gesture recognition ([32]). It is the first attempt (to my knowledge) to utilize such a network for the formulated regression task. Therefore, the question that arises is: *Can 3D CNN be successfully applied to formulated regression task and be comparable to existing state of the art baselines?*

1-2 Report Structure

This thesis is structured in 7 chapters including introduction:

- Broad overview, motivation and theoretical basis of CNNs is given in Chapter 2.
- Chapter 3 reviews the related CNN implementations relevant to this work.
- Chapter 4 summarizes the review of available datasets, describes the dataset selected for this work. Also describes required data preprocessing steps.
- Detailed description of proposed 3D CNN architecture and its implementation is given in Chapter 5. More detailed explanations of CNN and its possible improvements are outlined.
- Chapter 6 explains completed experiments, provides technical details and describes results obtained.
- Finally, Chapter 7 concludes this thesis stating the goals achieved, limitations and future work.

Theoretical Basis

This chapter introduces the reader to fundamentals of Deep Neural Network (DNN) and describes motivation and theoretical basis of CNN.¹

2-1 Multi-Layer Neural Network

The simplest form of deep Artificial Neural Network (ANN) is feed forward Multi-Layer Neural Network (MLNN). Similarly as biological neurons in our brain, artificial neuron is the elementary building block in an ANN (see Figure 2-1). Its function is to receive, process and transmit signal information. Artificial neuron receives one or more input units

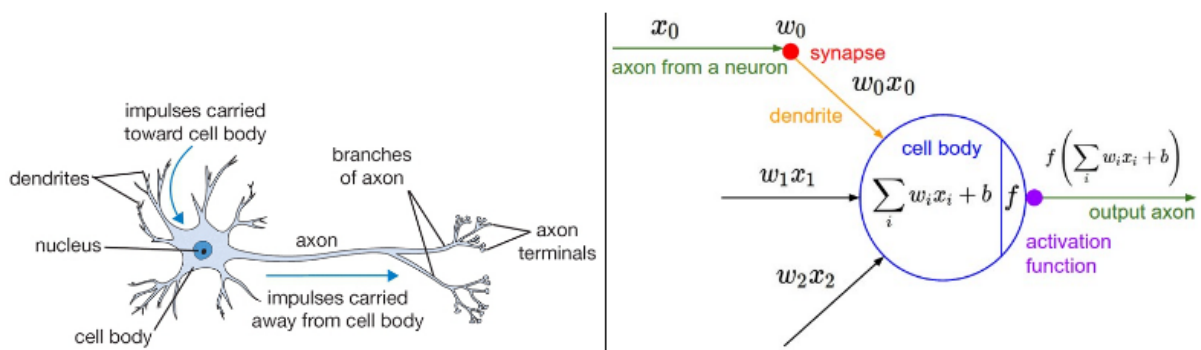


Figure 2-1: Analogy of biological neuron (left) and its mathematical model (right) [33]

corresponding to dendrites in the brain. i^{th} input unit will be denoted as x_i . Usually the

¹Readers familiar with the concepts of deep learning and convolutional neural networks may skip this chapter.

inputs are weighted by real numbers expressing the importance of the respective inputs to the output (denoted as w_i). Another important term is bias (denoted as b), which adds constant value to the input. In biological terms, a bias can be considered to be a measure of how easy is to get a neuron to fire.

When the weighted input is received, the artificial neuron performs three operations:

1. Summation of weighted inputs received:

$$\sum_{i=1}^N w_i x_i \equiv \mathbf{w} \cdot \mathbf{x}, \quad (2-1)$$

where N is number of input units and $\mathbf{w} \cdot \mathbf{x}$ is a dot product of weights and inputs vectors respectively.

2. Addition of the bias:

$$\sum_{i=1}^N w_i x_i + b \equiv \mathbf{w} \cdot \mathbf{x} + b \quad (2-2)$$

3. Application of nonlinear activation function:

$$a\left(\sum_{i=1}^N w_i x_i + b\right) \equiv a(\mathbf{w} \cdot \mathbf{x} + b), \quad (2-3)$$

where $a(\cdot)$ is activation or transfer function which is described in Section 5-3-1.

In this way artificial neuron produces the output (representing a biological neuron's axon) which is then transferred to other connected artificial neurons.

Feedforward MLNN is basically a collection of artificial neurons organized in layers and connected as a finite directed acyclic graph. Neurons belonging to one layer serve as input features for neurons in the next layer.

In each hidden layer, a non-linear transformation of the input from previous layer is computed. Therefore, the more hidden layers neural network has, the higher is its ability to learn more complex functions. In this simple form of deep neural network neurons between two adjacent layers are fully pairwise connected, but are not connected within the same layer. Such layers are called fully-connected. Because of the nonlinearity and high connectivity of the network, it is difficult to undertake theoretical analysis of MLNN.

To train the MLNN, a well-known backpropagation algorithm is used [34]. Briefly, training proceeds in two phases:

1. **Forward pass:** the weights and biases of the network are fixed and the input signal is propagated through the network layer by layer until it reaches the output. At the end, an error signal of the network is produced by comparing the output of the network with a desired response (ground truth).

2. **Backward pass:** the error signal is propagated back through the network layer by layer in the backward direction and the adjustments are applied to the weights and biases of the network in order to minimize the error function (cost).

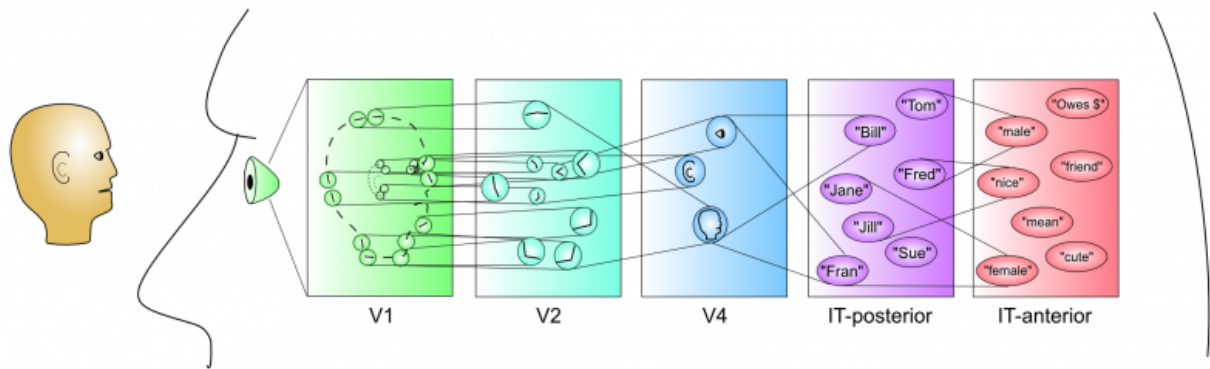


Figure 2-2: “Schematic of a hierarchical sequence of categorical representations processing a face input stimulus. Representations are distributed at each level (multiple neural detectors active). At the lowest level, there are elementary feature detectors (oriented edges). Next, these are combined into junctions of lines, followed by more complex visual features. Individual faces are recognized at the next level (even here multiple face units are active in graded proportion to how similar people look). Finally, at the highest level are important functional “semantic” categories that serve as a good basis for actions that one might take - being able to develop such high level categories is critical for intelligent behaviour.” [35]

2-2 Convolutional Neural Network

Although MLNN is able to approximate any function, it is not suitable when dealing with visual information, i.e. images. Firstly, the full-connectivity of the network leads to slow learning as the number of weights rapidly increases with the higher dimensionality of visual input. Secondly, the spatial organization of the visual input is not utilized in MLNN, since every pair of neurons between two layers has their own weight. For example, learning to recognize an object in one location wouldn’t transfer to the same object presented in a different location because separate weights would be involved in these calculations. Such drawbacks led to invention of CNN architecture, which exploits the spatial dimension properties of visual input whilst reducing the number of parameters to train.

The design of CNN was inspired by the structure of mammalian visual cortex where visual information received through the eyes is processed by neurons in the brain organized in hierarchical way. When visual stimuli reach the receptive field of a neuron it may be activated depending on its neuronal tuning. Neurons in the earlier visual areas have simpler tuning and smaller size of receptive field. Therefore, the most primitive visual forms such

as corners or edges are recognized in the primary visual cortex areas and more complex forms (feature groups, objects, object descriptions) - in the collateral areas (see Figure 2-2).

CNN is feed-forward supervised deep neural network first introduced in [36] in 1980. Since then a number of improvements were proposed and efficient methods developed to train this kind of network. Today CNNs are deployed in many practical applications in the fields of computer vision and natural language processing. CNNs were used by the winners of several competitions such as ImageNet, Kaggle Facial Expression, Kaggle Multimodal Learning, Kaggle CIFAR-10, German Traffic Signs, Connectomics.

In general, CNN is a special type of MLNN that has comparably much fewer connections and parameters and is easier to train. CNN can be applied to array data where nearby values are correlated, i.e. images, sound, time-frequency representations, video, volumetric images, RGB-Depth images. Although the most successful applications of CNNs were applied to 2D image data, recently there were some attempts to apply 3D convolutions on video and volumetric data (i.e. 3D medical scans). Despite 3D CNNs being harder to implement and visualize they can achieve very good performance if designed and calibrated well. The next chapters will cover some of CNN implementations that are most related to this work and more detailed explanations of how CNN works.

Related Work

This chapter gives an overview of different CNN architectures starting with the most common one and proceeding with more advanced and related to objective of this thesis. Most of the design and hyper-parameter choices of this work were made based on these examples.

3-1 Classic CNN Architectures

LeNet-5 The first CNN architecture which obtained state-of-the-art performance is shown in Figure 3-1 [37]. It is named LeNet-5 after the name of the author Y. LeCun. It was applied to handwritten digits recognition in 1998.

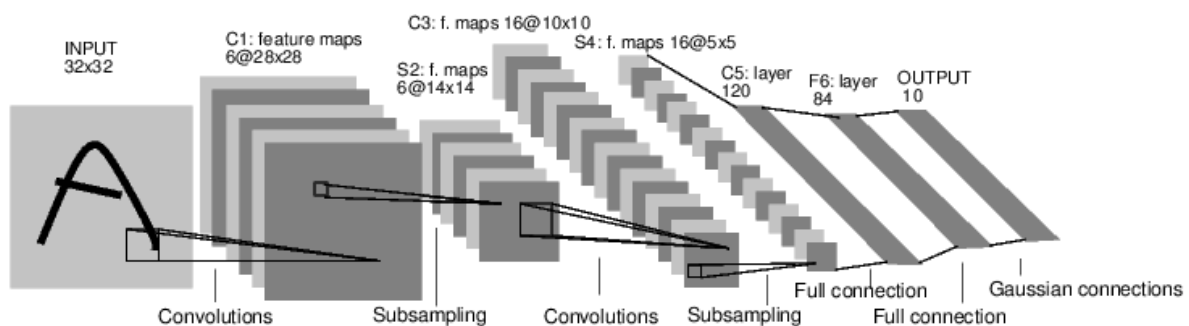


Figure 3-1: Architecture of CNN LeNet-5 [37]

It can be observed that at each convolutional or subsampling layer the number of feature maps is increased while the spatial resolution is reduced comparing to the corresponding previous layer. This approach gives translation invariance and tolerance to differences of

positions of object parts. Higher layers work on lower resolution inputs and process the already extracted high-level representation of the input. The last layers are fully connected layers that combine inputs from all positions to classify the overall inputs. The detailed explanation of different types of layers will be provided in Chapter 5.

Activation function used in LeNet-5 is scaled hyperbolic tangent and the output layer is composed of Euclidean Radial Basis Function (RBF) units for each class. Each output RBF unit computes the Euclidean distance between its input vector and its parameter vector. It can be interpreted as a penalty term measuring the fit between the input pattern and a model of the class associated with the RBF or as the unnormalized negative log-likelihood of a Gaussian distribution in the space of configurations of the previous layer. The loss function employed was the minimum Mean Squared Error (MSE). Training of this network was done by Stochastic Gradient Descent (SGD) algorithm.

Krizhevsky's architecture The more recent CNN architecture was proposed by A. Krizhevsky in 2012 [38]. It achieved outstanding results on a large benchmark dataset consisting of more than one million images - ImageNet [39]. This CNN architecture is the most often used as the basis for other modified CNN architectures (described later in this chapter) relevant to the problem of this thesis.

Comparing to LeNet-5, Krizhevsky's architecture (Figure 3-2) is deeper - it is comprised of five convolutional layers, three subsampling and three fully connected layers. The following novelties were introduced with this architecture:

- Activation function - Rectified Linear Unit (ReLU). The use of this activation function speeds up training which enables to experiment with such large neural networks.
- Training was carried out on two Graphical Processing Units (GPUs). Half of the neurons were stored in each GPU allowing GPUs to communicate only in certain layers. This means that, for example, the neurons of layer 3 take input from all feature maps in layer 2. However, neurons in layer 4 take input only from those feature maps in layer 3 which reside on the same GPU. The two-GPU network took slightly less time to train and achieved accuracy approximately 1.5% more than the one-GPU network.
- Local Contrast Normalization (LCN) applied after the first and second convolutional layers reduced over-fitting and error rate.
- Overlapping pooling was used instead of non-overlapping adjacent pooling units.

With the proposed architecture two over-fitting reduction techniques were used - dropout and data augmentation. Training was done by SGD with softmax loss function.

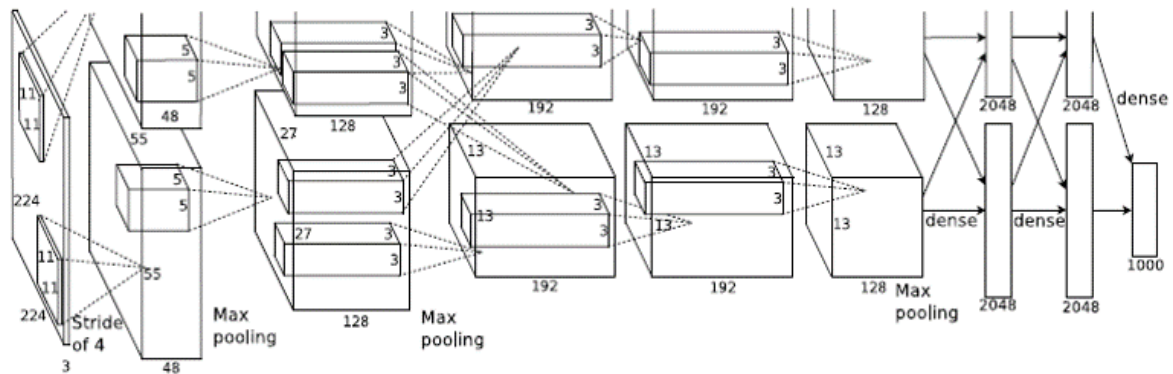


Figure 3-2: Architecture of Krizhevsky's CNN [38]

3-2 Pose Regression CNN Architectures

In this section four examples of CNNs applied to pose estimation problem will be reviewed.

DeepPose At the end of 2013, two researchers from Google, A. Toshev and C. Szegedy, formulated the Two-Dimensional (2D) pose estimation as a joint regression problem and showed how to cast it in CNN settings [16]. The full RGB input image is passed through 7-layered CNN to estimate the 2D location of each body joint. Predicted joint locations are then refined by using higher resolution sub-images as an input to a cascade of CNN-based pose predictors (see Figure 3-3).



Figure 3-3: CNN-based regressor and refiner architectures [16]

This architecture is based on the Krizhevsky's CNN described before. The difference is the loss function used. Instead of a classification loss, a linear regression is trained on top of the last CNN's layer by minimizing Euclidean distance between the prediction and the true pose. In order to achieve better precision of joint locations after the first stage, additional CNN regressors are trained to predict a displacement of the joint locations from previous stage to the true location. The input to these additional CNN regressors are sub-images

of the full image cropped around the predicted joint location from the previous stage. In this way, subsequent pose regressors are run on higher resolution images and thus learn features for finer scales which lead to higher precision. The CNN architecture is the same for all stages of the cascade.

Heat-map Models Another approach presented by J. Tompson and other researchers from New York University in 2014 [12] is based on the architecture shown in Figure 3-4. Presented model takes as an input 3 levels of RGB Gaussian pyramids (just two pyramids are shown in the figure) and outputs a heat-map for each body joint describing the per-pixel likelihood for that joint occurring in each output spatial location.

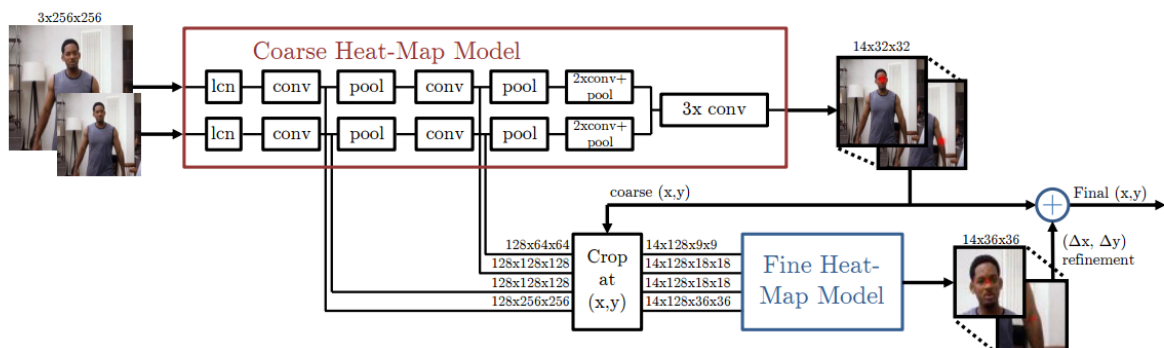


Figure 3-4: J. Tompson's CNN architecture [12]

Similarly as in [16], after predicting the heat-maps of all joints locations, these predictions are used to crop out a window centered at the predicted joints locations from the first two convolutional feature maps of each resolution. The contextual size of the windows is kept constant by scaling the cropped area at each higher resolution level. These feature maps are then propagated through a fine heat-map model to produce an offset within the cropped sub-window. Finally, the position refinement is added to the first predicted location producing a final 2D localization for each joint.

The fine heat-map model is a Siamese network [40] of instances corresponding to a number of joints, where weights and biases of each module are shared. These convolutional sub-networks are applied to each joint independently because the sample location for each joint is different and convolutional features do not share the same spatial context. The heat-map model and fine heat-map model are trained jointly by minimizing modified MSE function between the predicted heat-map and target heat-map which is a 2D Gaussian of constant variance centered at the ground-truth joint location (x, y) .

Pose Regression CNN The third example of CNN architecture (see Figure 3-5) is designed for video input and exploits temporal information from multiple frames. It

was presented by a joint group of researchers from University of Oxford and University of Leeds in 2014 [15].

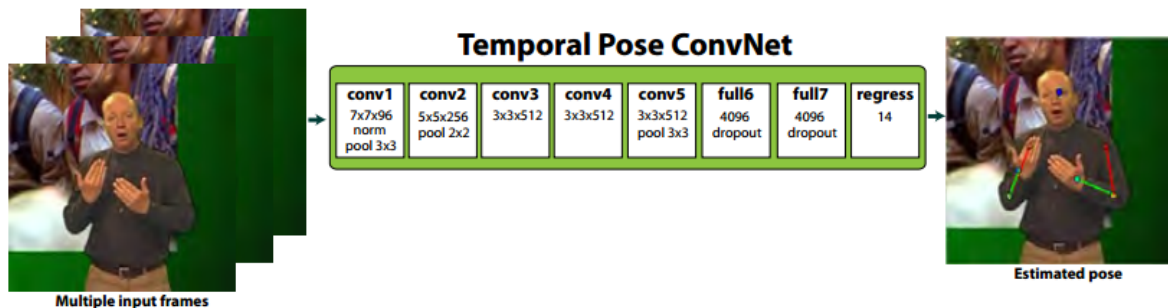


Figure 3-5: CNN architecture for video inputs [15]

The goal of their work was to track the 2D upper human body pose over long gesture videos. The overall architecture is very similar to the first one presented in this subsection ([16]), except for the input layer where multiple frames (or images of their differences) are inserted into the data layer color channels. For example, a network with three input frames contains 9 color channels in its data layer. Also, the mean image of over 2,000 sampled frames for each video in a dataset was precomputed in order to overcome over-fitting to the static background behind the person. Then, the video-specific mean was subtracted from each input image of corresponding video. The network's weights were also learned using mini-batch SGD as in the previous examples.

After one year the same research group presented some improvements of this architecture introducing some novelties (see Figure 3-6):

1. Spatial fusion layers that learn an implicit spatial model.
2. Optical flow used to align heat map predictions from neighboring frames.
3. Final parametric pooling layer that learns to combine the aligned heat maps into a pooled confidence map.

CNN for Binary Classification Similar, though not so deep, architecture proposed by A. Jain in 2013 is designed to perform independent binary body-part classification with one network per feature (see Figure 3-7).

The inputs of these networks are 64×64 pixel RGB image patches with applied LCN. CNNs are implemented as sliding windows to overlapping regions of the input. A window of pixels is mapped to a single binary output (logistic unit), representing the probability of the body part being present in that patch. Such approach enables to use much smaller CNNs and retain the advantages of pooling at the expense of having to maintain a separate set of

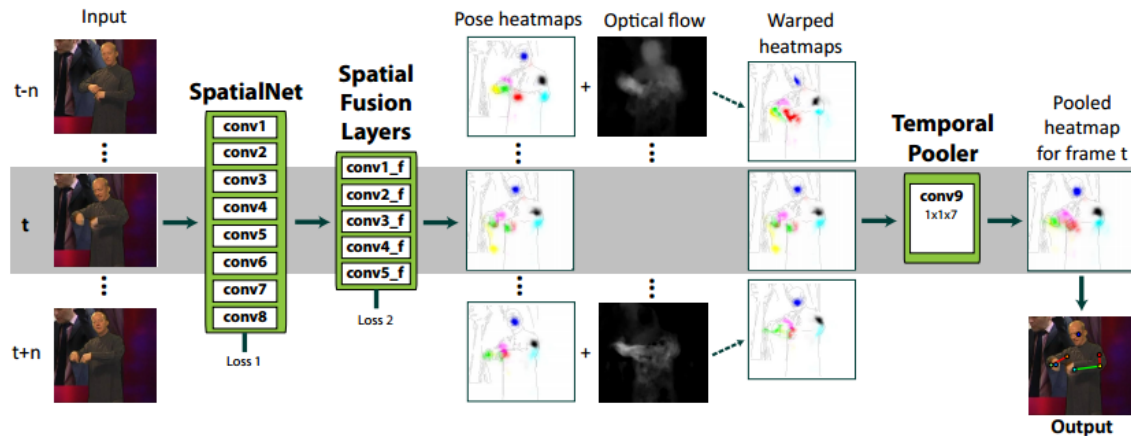


Figure 3-6: Deep expert pooling architecture for pose estimation [14]

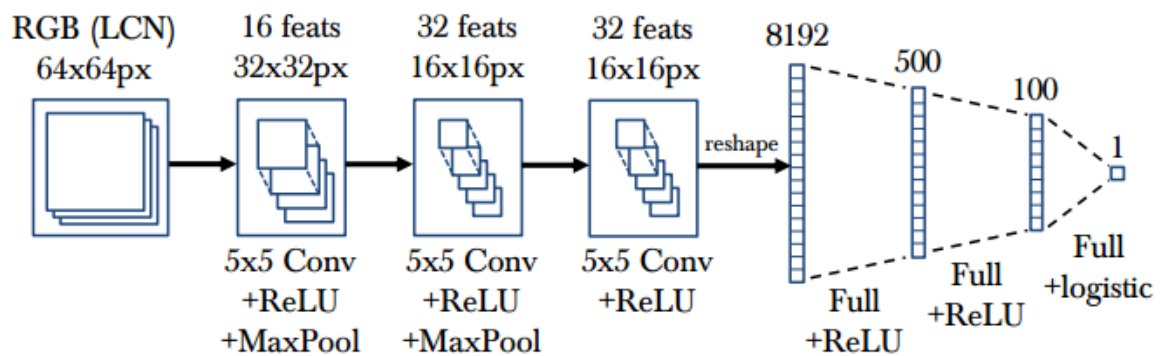


Figure 3-7: CNN architecture for binary classification [18]

parameters for each body part. Of course, a series of independent part detectors cannot enforce consistency in pose in the same way as a structured output model, which produces valid full-body configurations. Therefore, after training these CNNs with standard batch SGD, a method enforcing pose consistency using parent-child relationships is applied [18].

3-3 Multi-task CNN Architectures

It is interesting to review another type of CNN architecture designed not only for pose estimation but also other tasks such as joint or human body detection and action recognition.

CNN for Detection & Regression Tasks Researchers from City University of Hong Kong constructed such architecture for human pose estimation in 2014 [7]. Their framework

consists of two types of tasks - a joint point regression and detection tasks (Figure 3-8). The inputs for both tasks are the bounding box images containing human subjects. The goal of regression task is to estimate the positions of 3D joints relative to their parents' joints in camera coordinate system. The aim of detection task is to classify whether one local window contains the specific joint or not. One detection task is associated with one joint point and one local window.

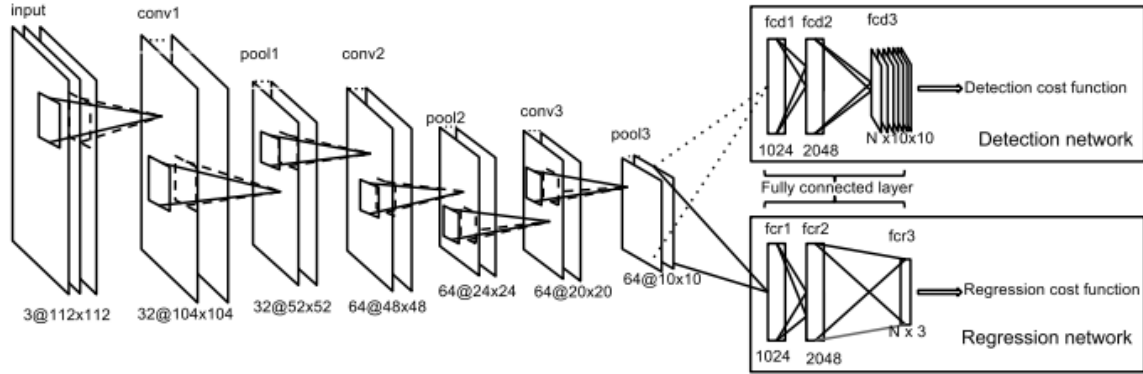


Figure 3-8: CNN architecture for detection and regression tasks [7]

It is worth mentioning that this CNN architecture was trained on the same dataset selected for this thesis (see Chapter 4). The whole CNN consists of 3 convolutional layers followed by subsampling layers that are shared by both regression and detection networks, 3 fully connected layers for the regression network, and 3 fully connected layers for the detection network. ReLUs are used for the first two convolutional layers and the first two fully connected layers for both regression and detection networks. Hyperbolic tangent as the activation function is used for the last regression layer. The LCN layer is added after the second convolutional layer to make the network robust to pixel intensity.

There were two approaches used to train this CNN. First, both regression and detection networks were trained jointly with the global cost function using backpropagation. In this case the shared network tends to learn features that benefit both tasks. Second, training was first performed on the detection network alone and then training for pose regression was initialized using the weights (of the convolutional layers) learned from the detection task. At the end, approximately the same performance was achieved by both strategies, although pre-training had longer running time. When either using pre-training or sharing features, the detection task helped to regularize the training of the regression network and guided it to the better local minimums.

Multi-task CNN Another approach presented by researchers from University of California, Berkeley in 2014 [8] also jointly trains single CNN for multiple tasks. Each task is associated with a loss function for person detection, pose estimation or action classification.

While jointly training this network the final loss function is simply the sum of all three loss functions multiplied by some coefficient. A higher value of this coefficient is given for action classification to make sure that the task has a significant contribution to the total loss, since there is significantly fewer training data for action compared to detection and pose. The joint network for the three tasks performs on average similarly to the networks trained for specific tasks individually, but it is much faster. The inputs for this CNN were object proposals (either segments or bounding boxes).

Dual-Source CNN Another recently proposed architecture for multi-task learning is called Dual-Source CNN [19]. The training of this network is run on two types of inputs - the local part object proposals and the full body images. In this way, a unified learning is performed to achieve both joint detection, which determines whether an object proposal contains a body joint, and joint localization, which finds the exact location of the joint in the object proposal. In the testing stage, the multi-scale sliding windows are used to provide local part information in order to avoid the performance degradation resulted from the uneven distribution of object proposals. Based on the networks outputs, the joint detection results from all the sliding windows are combined to construct a heat map that reflects the joint location likelihood at each pixel. The final estimation of each joint location is achieved by calculating a weighted average of the joint localization results at the high-likelihood regions of the heat-map (see Figure 3-9).

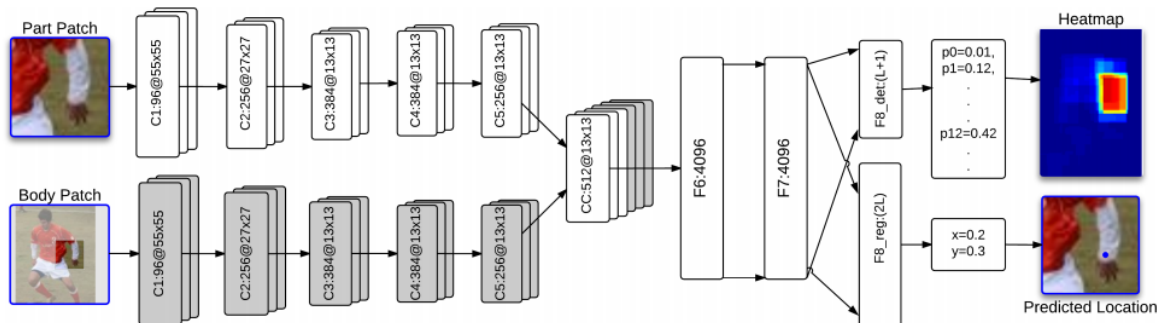


Figure 3-9: Dual-Source CNN architecture [19]

3-4 3D CNN Architectures

Since the interest of this thesis is to build a CNN model for temporal video data to determine if motion information helps to predict human pose location, it is important also to review some CNN architectures that exploit an operation of 3D convolution.

3D CNN The first (to my knowledge) such architecture was proposed in 2013 and applied to human action recognition in real-world environment [27]. It was proposed to perform 3D

convolutions in the convolution stages of CNNs to compute features not only from spatial dimensions but also from the temporal one. The 3D convolution is achieved by convolving a 3D kernel to the cube formed by stacking multiple contiguous frames together. By this construction, the feature maps in the convolution layer are connected to multiple contiguous frames in the previous layer, thereby capturing motion information.

It is noted that “a 3D convolutional kernel can only extract one type of features from the frame cube, since the kernel weights are replicated across the entire cube. A general design principle of CNNs is that the number of feature maps should be increased in late layers by generating multiple types of features from the same set of lower-level feature maps. Similar to the case of 2D convolution, this can be achieved by applying multiple 3D convolutions with distinct kernels to the same location in the previous layer”.

The proposed 3D CNN architecture is shown in Figure 3-10. Inputs to this network are 7 frames of size 60×40 centered on the current frame. Firstly, a set of hardwired kernels is applied in order to generate multiple channels of information from the input frames. This results in 33 feature maps in the second layer in 5 different channels known as gray, gradient-x, gradient-y, optflow-x, and optflow-y. The gray channel contains the gray pixel values of the 7 input frames. The feature maps in the gradient-x and gradient-y channels are obtained by computing gradients along the horizontal and vertical directions, respectively on each of the 7 input frames. The optflow-x and optflow-y channels contain the optical flow fields along the horizontal and vertical directions respectively, computed from adjacent input frames. This hardwired layer is used to encode prior knowledge of the features. Described scheme led to better performance as compared to random initialization. Finally, The output layer consists of the same number of units as the number of actions, and a linear classifier is applied on the 128D feature vector for action classification.

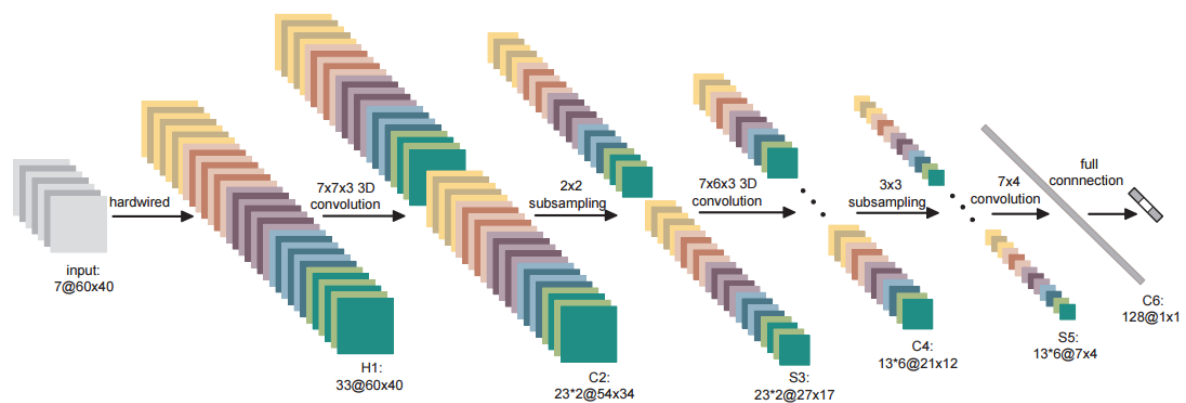


Figure 3-10: 3D CNN architecture for action recognition [27]

Reconfigurable CNN One more interesting architecture using 3D convolutions is illustrated in Figure 3-11. It is designed for automatic activity recognition from RGB-D

videos. The model consists of several network cliques that are the subparts of the network stacked up for several layers. In particular, each clique extracts features from one decomposed video segment associated to one separated sub-action from the complete activity. Specifically, for each clique, two 3D convolutional layers are first built upon the raw input (gray scale and depth data) and then followed by one 2D convolutional layer. Max pooling operator is applied on each 3D convolutional layer making the model robust to local body deformations and surrounding noises. Afterwards, the convolution results generated by different cliques are merged and concatenated into a long feature vector, upon which two fully connected layers are built to associate with the activity labels.

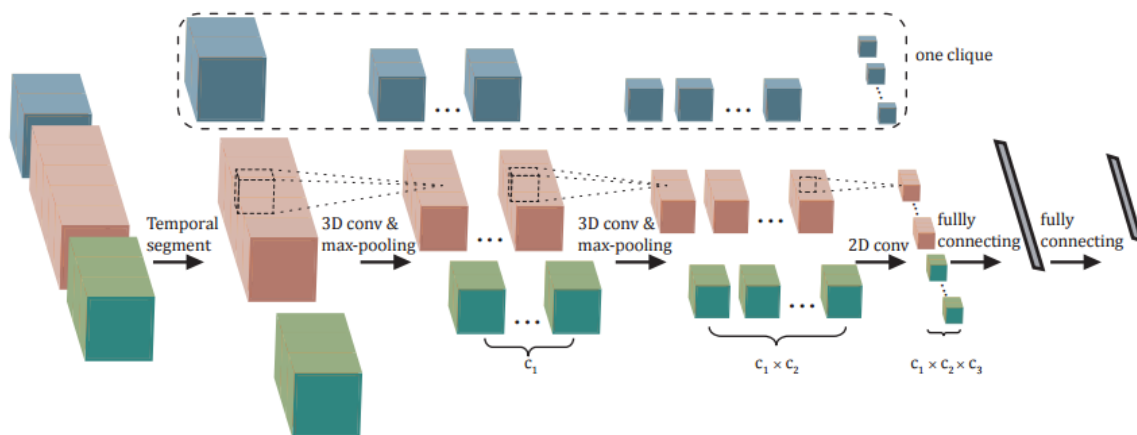


Figure 3-11: Reconfigurable CNN architecture [28]

Traditional SGD training method could not be applied to this kind of architecture. Therefore, a new method was proposed - Latent Structural Back Propagation (LSBP) which iterates with two steps:

- Fixing the current model parameters, it performs activity classification while discovering the temporal composition (i.e. determining the separated actions) for each training example.
- Fixing the decompositions of input videos, it learns the parameters in each layer of the network using the back-propagation algorithm.

In summary, reviewing different existing CNN architectures gives a good insight of possible ways to employ CNNs to different vision tasks. However, there are just some implementations of 3D CNN on video data. Also most of the work for human pose estimation is done on 2D image data.

Chapter 4

Dataset

In order to train a CNN, large dataset with annotated ground truth information is needed. For the formulated task the dataset essentially should meet the following requirements:

- Data should be in video format where single person performing different actions is captured;
- Each video frame should be annotated with ground truth full human body joint positions in 3D camera coordinate system;
- The resolution of images should be large enough in order to obtain a proper bounding box of human body.
- The dataset should be available to download and use for research purposes for free.

The desirable features of dataset include:

- larger number of different persons captured;
- variety of actions performed;
- larger number of total video sequences available;
- larger number of different camera views;
- possibility to easily obtain bounding boxes of human body.

This chapter will cover the dataset selection process and preprocessing steps needed to prepare network's input data. Firstly, the overview of available datasets meeting the necessary requirements is given. Secondly, the selected dataset is described in more detail. Finally, data preprocessing steps are depicted.

4-1 Overview

Selection of the dataset for training deep learning models meeting the requirements listed in chapter's introduction appeared to be not so simple task. The choice of training data is very important and has to be well thought of beforehand. Analysis of selected data format, arrangement, extraction and its preprocessing is a time-consuming, necessary and relevant process. Therefore, it is very undesirable to change the data selection decision later.

Unfortunately, there is no broadly used, well benchmarked dataset designed for 3D full human pose estimation in video format as, for example, the well-known ImageNet [39] for images. The list of eligible datasets found with their main features is given in Table 4-1.

Table 4-1: Publicly available datasets overview

Dataset	Year	Resolution	#Cameras	#Subjects	#Sequences	#Scenarios	#Joints
Berkeley MHAD	2013	640×480	2	12	660	11	43
Cornell Activity	2009	320×240	1	8	180	22	11
CMU-MMAC	2009	1024×768	3	43	645	5	22
Human3.6M	2014	1000×1000	4	10	1200	15	32
HumanEva	2007	640×480	3	4	56	6	15
INRIA RGB-D	2015	640×480	1	1	12	-	15
MPI08	2010	1004×1004	8	4	54	4	22

Recently, the comprehensive review of existing benchmark datasets containing 3D human skeleton data was provided in [41]. The authors enlist the datasets categorized by the type of devices used to acquire skeleton information. Although this survey was not available before deciding what dataset to use for this work (which could have facilitated the dataset selection process), nearly all of the enlisted references were reviewed and the most suitable for this task will be shortly described in the following subsections.

4-1-1 Berkeley MHAD

The Berkeley Multimodal Human Action Database (MHAD) [42] contains 660 video sequences (~82 min of total recording time) capturing 7 male and 5 female subjects performing 11 different actions. Each action was performed 5 times. 640×480 resolution RGB images are obtained using two Kinect cameras placed in opposite directions with a frame rate of 30 Hz. Kinect data was calibrated with motion capture system. The 3D ground truth skeleton coordinates are provided in the world coordinate system. Thus, in order to obtain the ground truth for this task, data preprocessing and analysis of the demo code provided with the dataset is needed.

The dataset was primarily collected for the purpose of providing the computer vision community with multi-modal data for action recognition task. In the context of this work, the dataset has drawbacks of relatively low resolution and the need for preprocessing to

obtain bounding boxes of human body and ground truth coordinates in camera view. On the other hand, the dataset has quite large number of sequences and a variety of subjects and scenarios captured.

Most of the papers citing this dataset deal with human action recognition task (at the time of writing this thesis). Therefore, the results of this project could not be compared with other researchers' results, making this dataset undesirable to select.

4-1-2 Cornell Activity

The Cornell Activity Dataset (CAD) consists of CAD-60 [43] and CAD-120 [44] datasets containing in total 180 RGB-D video sequences recorded using Kinect. RGB data has the resolution of 320x240 and frame rate of 30 Hz. 3D ground truth locations of 15 joints are provided in the world coordinate system. Only 11 joints also have orientation allowing to obtain the positions in camera coordinate system. Bounding boxes of human body are not provided and possibly should be obtained using the ground truth.

The positive feature of this dataset is that it has various activities captured in different environments. However, the dataset is more suitable for action recognition task and there is no work reported to compare the results of this work with.

4-1-3 CMU-MMAC

The CMU Multimodal Activity (CMU-MMAC) Dataset [45] is another multi-modal dataset containing videos capturing 43 subjects performing meal preparation and cooking actions. There are 3 cameras capturing high resolution (1024x768) images at 30 Hz frames rate from 3 different views (including one from the top view) where full human body with some occlusions is visible. Calibrated motion capture data is provided in C3D file format [46], which requires a thorough understanding to be able to manipulate the ground truth data.

The main drawback of this dataset is the type of activities captured as the task of this thesis preferably requires more diverse human body movements.

4-1-4 Human3.6M

Human3.6M Dataset [47] is so far the largest publicly available motion capture dataset. It consists of high resolution 50 Hz video sequences from 4 calibrated cameras capturing 10 subjects performing 15 different actions. 3D ground truth joint locations are provided in camera coordinate system. Additionally, bounding boxes of human bodies are provided. The ground truth data for 3 subjects is withheld and used for results evaluation on the server. These features determined this dataset to be selected for this task. More detailed description of this dataset is given in the next section.

4-1-5 HumanEva

The HumanEva dataset [48] contains training data of 56 color video sequences of 640×480 resolution capturing 4 subjects performing 6 predefined actions in three repetitions. There are $\sim 14,000$ synchronized video frames and ground-truth 3D joint locations available for training and validation and $\sim 30,000$ for testing. Background subtraction code is provided with the dataset. However, it was complicated to run the code on Windows 10 using Matlab R2012b version.

Regarding this task HumanEva dataset is too small for training CNN but is appealing for testing as there is a number of other papers reporting 3D pose estimation results using this dataset.

4-1-6 INRIA RGB-D

INRIA RGB-D dataset [49] has 12 video sequences of one person performing daily life activities in a scene with occlusions. There are 3D ground truth positions available for 15 joints. The dataset is quite new and attractive for testing the ability of tracking algorithms to deal with severe occlusions. However, it is quite small, there is just one person captured and bounding boxes of the human body are not provided.

4-1-7 MPI08

The indoor motion capture dataset (MPI08) [50], [51] provides research community with multi-view video sequences obtained from 8 calibrated cameras together with 3D laser scans and registered meshes with inserted skeleton. Videos are of high resolution where 4 subjects are captured performing 4 different actions. The data structure and Matlab demo script provided require time to be understood and possibly additional effort is needed to obtain final data for the task of this thesis.

4-2 Human3.6M Dataset

As mentioned in previous section, Human3.6M Dataset was selected for this task because of its size, high resolution, multi-camera views, a number of different actions captured, ready to download segments of human body and 32 joints locations in camera coordinate system. Moreover, there is availability to officially test the results on the dataset's server for the fair comparison with other methods. Hereafter, more detailed description of Human3.6M data is provided.

4-2-1 Subjects

The actions were performed by 11 professional actors, 6 male and 5 female, chosen to span a body mass index from 17 to 29. There are three subjects (S2, S3 and S4) selected for testing. For these subjects no ground truth data is provided and evaluation is available only through the server of dataset providers. Video data of subject no. 10 is not provided due to certain privacy concerns, therefore it will not be used for this work. Evaluation server allows testing both with and without S10. The pictures of all the subjects can be seen in Figure 4-1.



Figure 4-1: Subjects in Human3.6M dataset

4-2-2 Actions

Each actor performed 15 different everyday scenarios in two trials. These scenarios include various movements containing walking with many types of asymmetries (e.g. walking with a hand in a pocket, walking with a bag on the shoulder, walking with a dog or with another person), sitting and lying down poses, various types of waiting poses and others. The actors were instructed before acting about examples of the poses in different scenarios, but were given quite a bit of freedom in moving naturally over a strict, rigid interpretation of the tasks. Examples of the poses from different scenarios are shown in Figure 4-2.

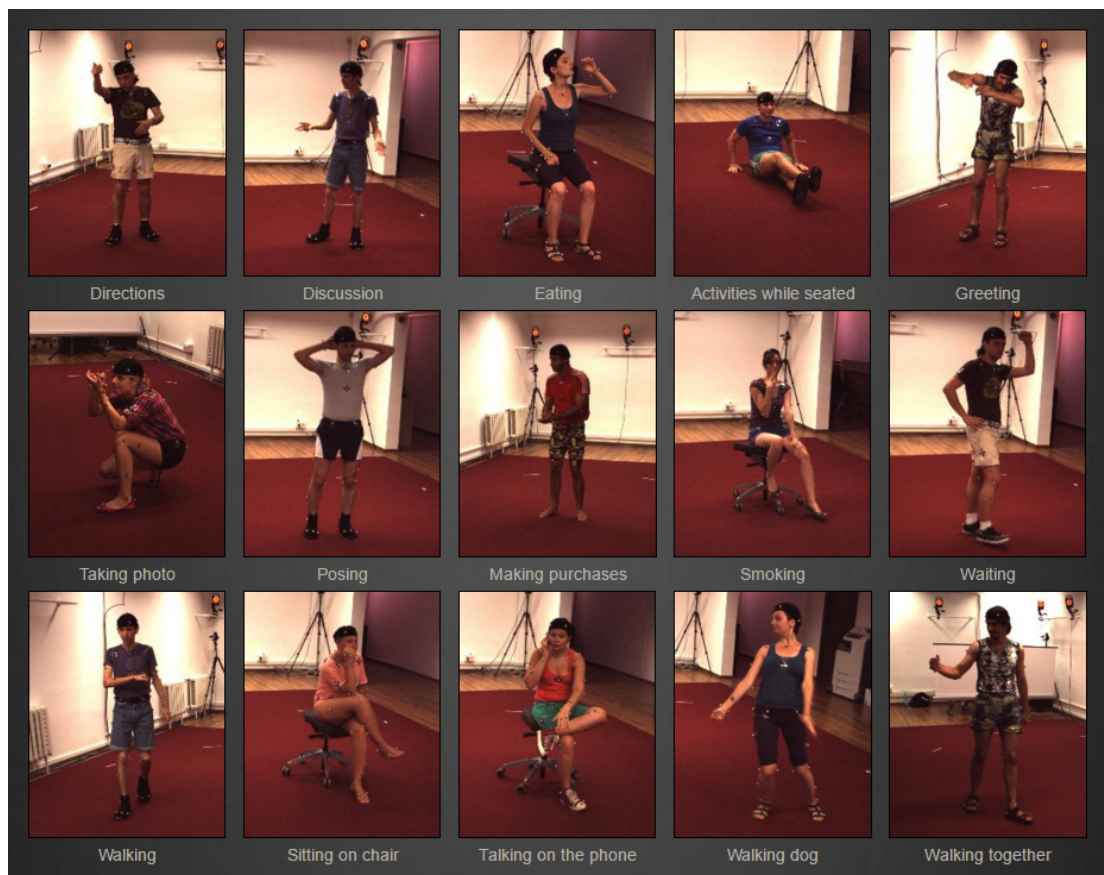


Figure 4-2: Set of actions in Human3.6M dataset [52]

Scenarios can be grouped by the type of movements they represent. This grouping and percentage of total training and testing video frames are shown in Table 4-2-2.

It is known that the group of activities where subjects perform different actions when sitting on the floor (A9) is the most challenging. This is because of the high rate of self-occlusion and bounding box aspect ratio changes. Sitting on the chair (A8) scenario is also challenging due to the use of a chair. Complexity of taking a photo (A11) and walking with a dog (A14) scenarios is also the cause of bounding box variations. These motions are also less repeatable and more liberty was granted to the actors performing them.

Table 4-2: Distribution of Human3.6M poses per scenario and type of action

Type of Action	Scenario (Abbr.)	% of Total Training Poses	% of Total Testing Poses
Upper body movement		17%	19%
	Directions (A1)	6%	9%
	Discussion (A2)	11%	10%
Full body upright variations		26%	32%
	Greeting (A4)	5%	6%
	Posing (A6)	5%	6%
	Making Purchases (A7)	4%	4%
	Taking Photo (A11)	5%	7%
	Waiting (A12)	7%	9%
Walking instructions		18%	15%
	Walking (A13)	8%	7%
	Walking with a dog (A14)	5%	4%
	Walking together (A15)	5%	4%
Variations while seated on a chair		32%	27%
	Eating (A3)	7%	7%
	Talking on the phone (A5)	8%	7%
	Sitting on the chair (A8)	8%	7%
	Smoking (A10)	9%	6%
Sitting on the floor	Activities while seated (A9)	8%	8%

4-2-3 Video Data

RGB video data was acquired using 4 digital cameras placed in the corners of the effective capture space of approximately 4m×3m. Video frame rate is 50 Hz and resolution is 1000×1000. Videos are available to download in MP4 format. Corresponding bounding boxes of a human body can be obtained from the binary masks available to download in MAT format.

4-2-4 Pose Data

32 joints locations were acquired using 10 motion capture cameras. The 3D motion capture system Vicon tracks small reflective markers attached to the subject's body. Tracking maintains the label identity and propagates it through time from an initial pose that is labeled either manually or automatically. A fitting process uses the position and identity of each of the body labels and proprietary human motion models to infer accurate pose parameters.

The Vicon system exports the joint angles. Joint positions in a 3D coordinate system are obtained from these angles by applying forward kinematics on the skeleton of the subject. For this work the transformed 3D positions for monocular prediction using camera parameters are used. These positions are available to download in CDF file format. Positions are relative to a specially designated joint called the root corresponding to the pelvis bone position. It is taken as a center of the coordinate system. Projections of the skeleton onto the image plane are also available to download in CDF format. The skeleton metadata is provided in XML file.

4-2-5 Evaluation and Error Measure

Evaluation on the dataset's server is executed by uploading specific results file, which can be generated utilizing the code provided together with the dataset. All 3D pose estimations for the three testing subjects (S2, S3 and S4) are written to this file. The Mean per Joint Position Error (MPJPE) is used to evaluate performance.

For the joint estimations of one frame m_e and corresponding ground truth locations m_{gt} MPJPE is computed as:

$$MPJPE = \frac{1}{N} \sum_{i=1}^N \|m_e(i) - m_{gt}(i)\|_2 \quad (4-1)$$

where N is the number of joints measured in skeleton. For a set of frames the error is the average of the MPJPEs of all the frames.

Despite the 32 available joint locations, evaluation is performed for the base skeleton of 17 joints. This limitation of the number of joints helps discard the smallest links associated to details for the hands and feet, going as far down the kinematic chain to only reach the wrist and the ankle joints. These 17 joints are marked in red in Figure 4-3.

After results submission the measurements are reported in millimeters per every action separately.

4-3 Data Preprocessing

The goal of data preprocessing is to obtain clean data, from which it should be easy to sample input data for CNN. The following preprocessing steps were performed to the downloaded video, binary masks for human body segmentation and 3D/2D pose data:

- video frames are cropped using bounding box binary masks and extended to the larger side to make the crop squared;
- in case the crop exceeds image boundaries, it is padded with the corresponding edge pixel values;
- cropped images are resized to 128×128 resolution that was chosen arbitrarily;
- 2D image plane joint positions are adjusted accordingly.

Each preprocessed video was then stored to HDF5 file and named uniquely. The code and more detailed information of data preprocessing and storing is provided together with this thesis. The results of cropping can be seen in Figure 4-4.

The total number of frames and size of the data with ground truth information can be seen in Figure 4-5.

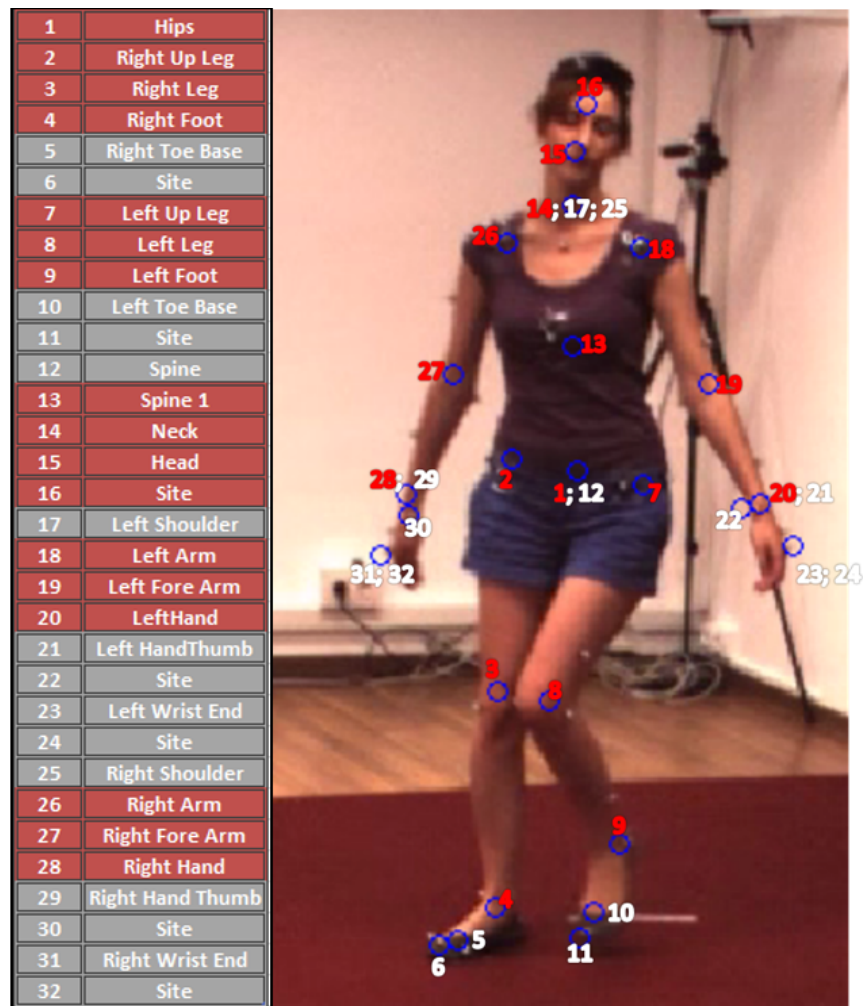


Figure 4-3: Skeleton joints locations

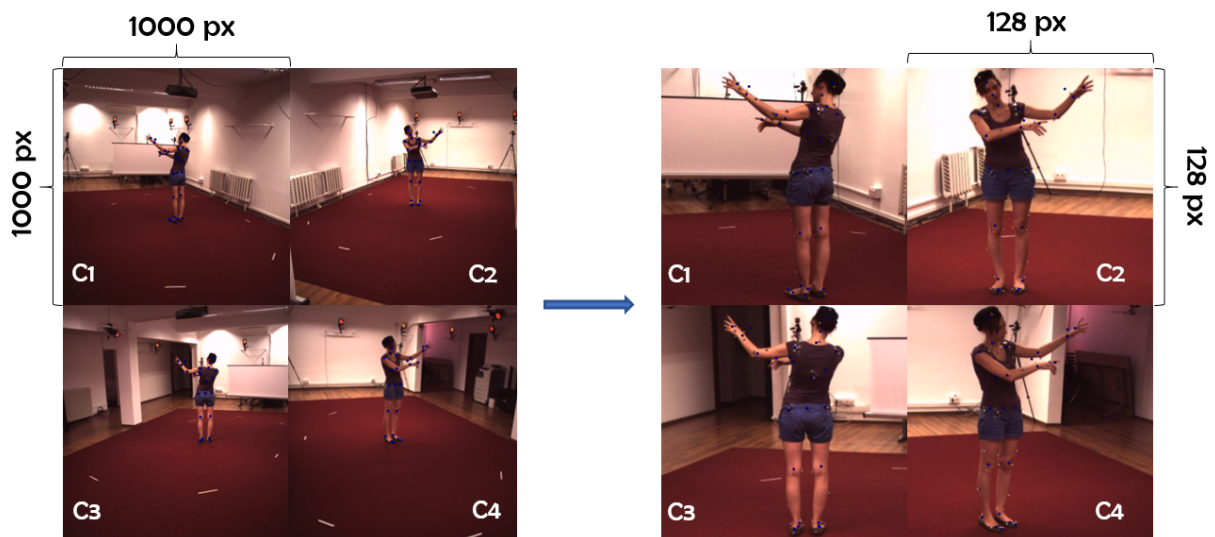


Figure 4-4: Image preprocessing from 4 camera views capturing subject no. 1 performing action "Directions"

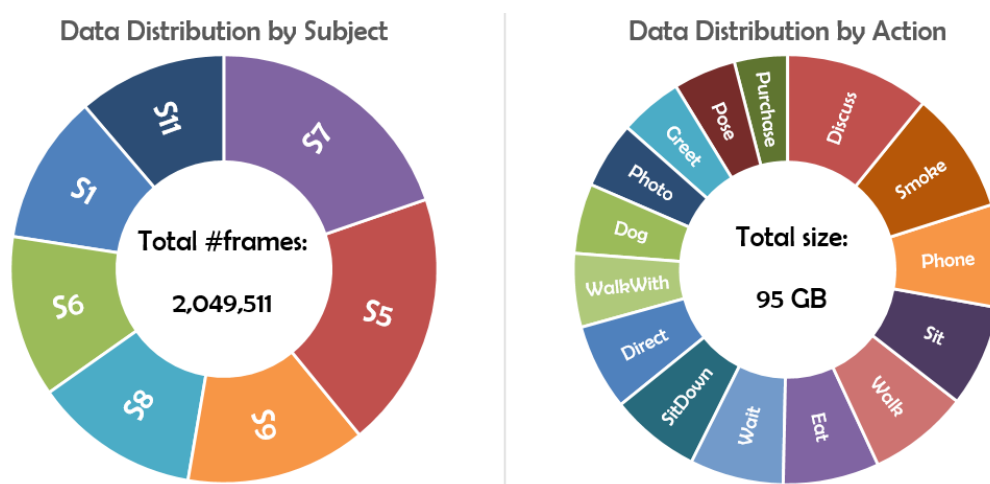


Figure 4-5: Preprocessed data distribution by subject and action

Three Dimensional Convolutional Neural Network

This chapter will cover implementation of the proposed 3D CNN model, its training and testing details. All the Python code created for realization is provided together with this thesis.

5-1 Data Sampling

Due to the large amount of data, limited memory and time available, it is important to perform data sampling from the preprocessed data files (see Section 4-3). Before sampling it is necessary to decide on some parameters to be used. They are the following:

- subsets or sets of 7 subjects, 15 actions, 4 camera views and 2 trials;
- number of color channels (either gray scale or RGB);
- video frames sequence length in one sample;
- frame rate (how many frames to skip between consecutive sampled frames);
- image size ($\geq 128 \times 128$);
- frames per video to be selected;
- number of joint locations to be considered (≤ 32)
- either select the first video frames or perform random selection.

Depending on these input parameters, the output of sampling is then stored in HDF5 file containing three datasets - one of the samples selected and the other two of 2D and 3D ground truth data (if it exists).

Table 5-1: Sampling parameters used for different experiments

Parameter	Setting 1	Setting 2	Setting 3	Setting 4
Set of training subjects	{S1, S6, S7, S8, S9}	{S1, S5, S6, S7, S8, S9, S11}	{S1, S5, S6, S7, S8}	{S1, S5, S6, S7, S8, S9, S11}
Set of validation subjects	{S5}	-	{S9}	-
Set of testing subjects	{S11}	-	{S11}	-
Set of actions			All	
Set of camera views			All	
Set of trials			All	
Number of color channels			3 (RGB)	
Frames sequence length			5	
Frames rate			13	
Image size			128×128	
Frames per video	175/85/86	124	350/200/210	300
Number of joints	17/11	17/11	17/11	17
First frames or random selection			Random	

For experiments, sampling was done in 4 different parameter settings shown in Table 5-1. It can be seen that some parameters were constant for all the experiments - all samples were composed of 5 sequential (skipping 3 frames in between to obtain frame rate of 13) color images with resolution of 128×128. Random selection was done from every chosen training, validation and testing subjects' videos to ensure that all the possible poses are selected. For the experiments in Setting 1, subjects S5 and S11 were chosen for testing and validation in order to see the results for both - female and male subjects. Later on, subject S5 was changed to S9 in order to compare results with those of other researchers. There was no need to perform a separate sampling for the upper-body trainings (using 11 joint locations) as this was accomplished at the time of loading data to the network by removing unused ground truth locations.

For the official testing data (of subjects S2, S3 and S4), the sampling is also performed to obtain data of the same shape as the network's input data. In this case, all the video frames are processed without random selection and leaving out the ground truth data that is not provided.

5-2 Network's Input and Output Data

The 3D CNN network, trainable with mini-batch Stochastic Gradient Descent (SGD) algorithm, during one training iteration (forward and backward pass) takes a number of image sequences as an input represented by a 5D tensor $\mathbf{X} \in \mathbb{N}_{[0,255]}^{b \times f \times c \times w \times h}$, where b is the size of mini-batch, f - number of frames in one sequence, c - number of color channels, w - image width, h - image height. The output of this network is represented by a 4D tensor $\mathbf{Y} \in \mathbb{R}^{b \times f \times j \times d}$, where j is the number of joints and d is the number of joint coordinates.

To obtain the full set of input and output data of a defined shape, data acquired after sampling (Section 5-1) is processed again and stored in the final HDF5 files (one for training, one for validation and one for testing) to be used for training. These files contain a complete set of defined number of batches and ground-truth joint locations.

For all completed experiments the following parameter values were used:

- mini-batch size = 10;
- image sequence length of one sample = 5;
- number of color channels = 3;
- image width = image height = 128;
- number of joints = 17 (or 11);
- number of joint coordinates = 3 (corresponding to x , y , z).

For the Settings 1 and 2, the total number of batches used was 10,000 for training, 1,000 for validation and 1,000 for testing. In Settings 3 and 4, the numbers were increased to 20,000, 2,000 and 2,000 respectively.

It has to be noted that during this procedure ground truth joint positions were centered to the pelvis bone position (first joint) and all z coordinates were increase by 4000 to avoid negative values.

When running experiments for the upper body positions, joints of the lower body can be removed from the output data when loading them to the network leaving 11 trainable joint locations.

5-3 CNN Building

This section is dedicated to the proposed 3D CNN architecture and its building parts. All the components used in the well performing architecture will be reviewed in separate subsections together with a theoretical basis of each. The final model of network's architecture was made up by starting with the small basic network with only three hidden convolutional layers and building it up when testing with the small subset of data. Decisions on the construction parts and hyper-parameter selection were made by analyzing experimental results and utilizing similar choices reported in related work reviewed in Chapter 3.

During the last years, there were many articles and tutorials released with recommendations and optimization techniques for building and training deep learning models. Due to the time and available hardware constraints, only the most relevant and acknowledged techniques were implemented in this work. However, available optimizations that would be useful (and interesting) to implement and test in the future are outlined in this chapter.

5-3-1 Activation Functions

Every activation function (or non-linearity) $a()$ takes a single input x and performs a certain fixed mathematical operation on it. Selection of activation function to be used in CNN is made based on its properties. Most common activation functions used in Deep Neural Network (DNN) are sigmoid, hyperbolic tangent and Rectified Linear Unit (ReLU). In CNNs related to this work usually the ReLU is used. ReLU is simply a threshold at zero:

$$a(x) = \max(0, x) \quad (5-1)$$

ReLU became very popular because of its properties:

- It was found that it greatly accelerates the convergence of SGD, is much simpler and requires less computational power compared to the sigmoid or hyperbolic tangent functions. It is argued that this is due to its linear, non-saturating form [38].
- When using ReLUs, it is more likely to have a true zero activations; this results in a large number of neurons not being activated for any one case. This property is more biologically plausible [53] and has been demonstrated to improve the accuracy of DNNs [54].

Despite of the listed ReLUs advantages, this type of activation can easily become weak during training and can “die”. For example, a large gradient going through a ReLU activation could cause the weights to update in such a way that the neuron will never activate on any data point again. If this happens, then the gradient going through the unit will forever be zero from that point on. In this way, the ReLU units can irreversibly “die” during training since they can get knocked off the data manifold. If the learning rate is set too high, it is possible that a lot of neurons were never activated across the entire training dataset. With a proper setting of the learning rate, this is less frequently an issue.

To overcome this kind of problem, Parametric Rectified Linear Unit (PReLU) (or Leaky ReLU) was introduced in [55]. Instead of the activation function being zero when $x < 0$, a PReLU will instead have a small negative slope p :

$$a(x) = \begin{cases} x, & \text{if } x > 0 \\ px, & \text{if } x \leq 0 \end{cases} \quad (5-2)$$

Coefficient p can be manually set to a small value or adaptively learned. Some researchers report success with this form of activation function, but the results are not always consistent [56].

In this thesis all the activations are PReLU with p set to 0.01.

During the past 3 years, other types of non-linearities were introduced like Maxout [57], Network in Network [58] and Adaptive Piecewise Linear Units [59]. These approaches are not broadly used thus it might be interesting to experiment with them.

To conclude, in literature rarely different types of neurons are mixed in the same network, even though there is no fundamental problem in doing so.

5-3-2 Normalization Layer

To reduce the variability that DNNs need to account during training, input data is usually preprocessed by applying Global Contrast Normalization (GCN) or Local Contrast Normalization (LCN). The overall added value of GCN and LCN is dependent on the kind and size of the dataset used as there are controversial findings reported in different papers ([38], [60], [61]).

Given the input image $\mathbf{X}$, GCN outputs a modified image $\mathbf{X}'$, defined as:

$$\mathbf{X}' = \frac{\mathbf{X} - \bar{\mathbf{X}}}{\max(\varepsilon, \sqrt{\lambda + (\mathbf{X} - \bar{\mathbf{X}})^2})}, \quad (5-3)$$

where $\bar{\mathbf{X}}$ is the mean intensity of the one entire image, λ is a small, positive regularization parameter to bias the estimate of the standard deviation and ε is usually zero or very small value that can be used to avoid normalization by very small values.

Experiments carried out in this thesis showed that predictions accuracy slightly increased when GCN was applied before first convolutional layer. Applying LCN after first, middle or last convolutional layers (in different configurations) did not show significant improvement. Therefore, in the final architecture only GCN is applied. It should be noted that the added value of GCN was observed when testing on the small subset of the dataset and may not be relevant when used with more data. However, this has not been tested. Parameters λ and ε were set to 10 and 10^{-8} respectively and were not changed.

5-3-3 Convolutional Layer

Convolutional layer is the main part of a CNN. It is responsible for applying mathematical computations of discrete convolution (denoted with an asteric $*$) on the input images or feature maps (if it is not the first convolutional layer) ($\mathbf{X}$) using kernels (or filters) ($\mathbf{K}$). The output of convolutional layer is a predefined number of so called feature maps. The following is a mathematical expression of discrete convolution applied to three dimensional data using three dimensional flipped kernels:

$$(\mathbf{K} * \mathbf{X})_{i,j,k} = \sum_m \sum_n \sum_l \mathbf{X}_{i-m,j-n,k-l} \mathbf{K}_{m,n,l} \quad (5-4)$$

The kernel is flipped to obtain the commutative property of convolution operation which leads to less variation of valid values of m , n and l . As a result, it is more convenient to implement in machine learning library. A simple case of 3D convolution is visualized in Figure 5-1.

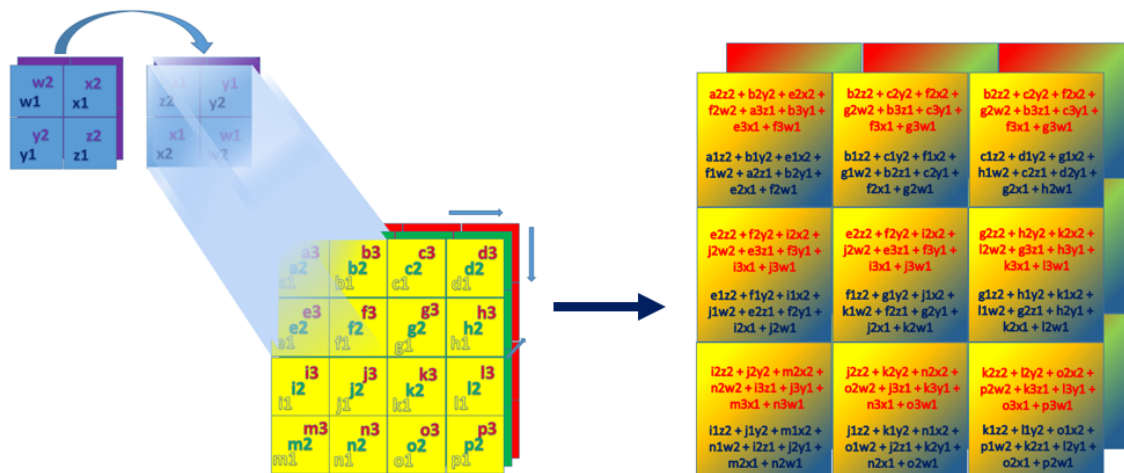


Figure 5-1: An example of 3D convolution applied to 3D tensor of size $3 \times 4 \times 4$ using flipped kernel of size $2 \times 2 \times 2$ outputting one feature map of size $2 \times 3 \times 3$. This is the case of 'valid' mode when the kernel is applied wherever it completely overlaps with the input. It generates outputs of shape: input shape - kernel shape + 1.

Before training, the filters are initialized in some way and then adjusted at every training epoch by propagating back the derivatives with respect to the cost calculated using predicted and ground truth values at the end of the network. To get one feature map, the same filter is applied to the input. This feature is called *parameter sharing* and it helps to save memory and increase networks' efficiency.

Another important characteristic of convolutional networks is *sparse connectivity* which is due to the small kernel sizes. Unlike in fully connected layers where each input neuron interacts with output neuron, in convolutional layer one filter is applied to small regions of the input. In this way small, meaningful features such as edges or corners can be detected with filters that occupy much less memory. Intuitively, the network learns filters that activate when some specific type of feature at some spatial position in the input is detected.

There are three hyper-parameters which control the output size of convolutional layer:

- Kernel size: it controls the number of neurons in the convolutional layer that connects to the same region of the input tensor.

- **Stride:** it specifies how many positions apart a filter is moved across the input. If the stride is high then the receptive fields will overlap less and the output will have smaller dimensions.
- **Zero-padding:** it is the size of zero-padding performed on the input's borders.

In this implementation the stride is always equal to 1 and there is no zero-padding performed. Experiments have been completed with different kernel sizes and a number of convolutional layers in the network. The best performance was achieved with 5 convolutional layers with kernel sizes $3 \times 5 \times 5$, $2 \times 5 \times 5$, $1 \times 5 \times 5$, $1 \times 3 \times 3$ and $1 \times 3 \times 3$ respectively. It can be seen that convolutions across time dimensions were applied just for the first two layers as the number of frames per sample is relatively small. For the future, there is still a lot of room left for trying different compositions with kernel sizes and numbers of layers.

5-3-4 Pooling Layer

After performing the convolution, linear output activations are run through a nonlinear activation function described in Subsection 5-3-1. The next step is pooling (or subsampling) operation which replaces the input at certain location with a summary statistic of the nearby input values.

Similarly as in convolutional layer, pooling is performed by sliding a kernel over the output from the previous convolutional layer and computing one single value (maximum, average or other) from the region which has the same size as the kernel (see Figure 5-2). The desired effect of pooling is to transform the representation of the feature map discarding irrelevant information while retaining important information.

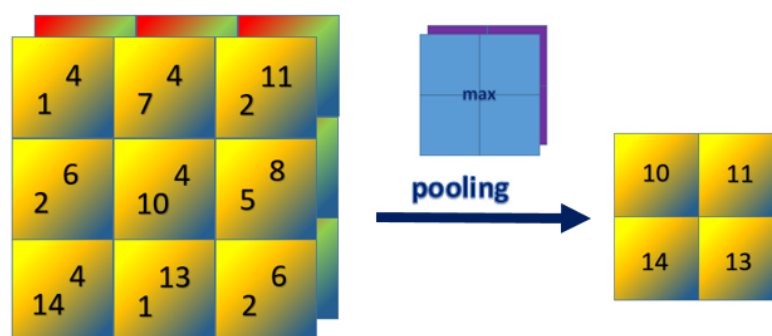


Figure 5-2: An example of 3D max pooling applied to 3D tensor of size $2 \times 3 \times 3$ using kernel of size $2 \times 2 \times 2$ outputting a reduced feature map of size $1 \times 2 \times 3$

The most common is max or average pooling [62] that outputs the maximum or average value of the rectangular neighborhood. In this network only max pooling was used.

Nevertheless, there are many other types to try such as L^2 norm or weighted average based on the distance from the central pixel pooling, more complicated stochastic [63], spatial pyramid pooling [64] or most recent fractional max pooling [65]. There is also a proposal to remove the pooling layer in favor of architecture that only consists of repeated convolutional layers. To reduce the size of the representation, it is suggested to use larger stride in convolutional layer once in a while [66].

In the tuned architecture proposed in this thesis, max pooling is performed after the first, second and fifth convolutional layers only on the image space with the kernel of size 2×2 .

5-3-5 Fully Connected and Output Layers

Finally, after several convolutional and pooling layers, the high-level reasoning in the neural network is done via fully connected layers. A fully connected layer simply takes all neurons in the previous layer and connects them to every single neuron it has.

In the proposed architecture, the output of the last pooling layer is flattened to one dimensional vector of size 9680 and then is fully connected to the output layer of size 255 (see Subsection 5-2). It was an attempt to add two fully connected layers, but this did not result in significant improvements.

5-3-6 3D CNN Architecture

Complete 3D CNN architecture is shown in Figure 5-3. “C” stands for convolutional layer, “P” for pooling layer. Kernel sizes are specified in parenthesis. Second row shows the size of corresponding layer’s output.

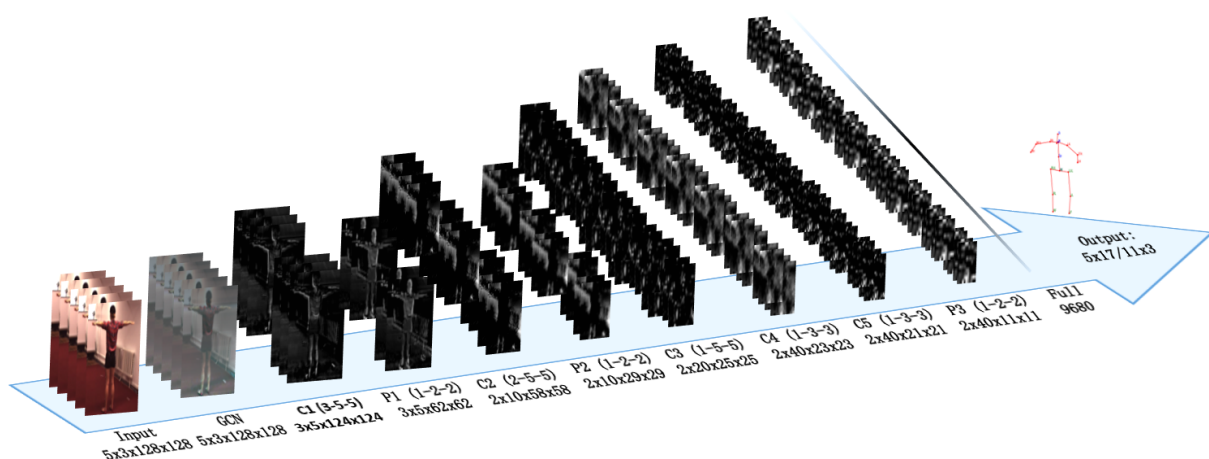


Figure 5-3: Proposed 3D CNN Architecture

5-4 CNN Training

In the previous section, building blocks of 3D CNN architecture were presented. Next, to make it do the magic it has to be trained. This section will cover the methods used to train the proposed architecture including other training related decisions such as cost function, parameter initialization and regularization. As before, the other not tested existing techniques and possible improvements will be shortly outlined too.

5-4-1 Parameter Initialization

As stated before, the CNN network has to optimize its weights that form the kernels in convolutional layers and affect the outputs in fully connected layers. Before first training iteration, these weights and biases have to be initialized. The choice of initialization strategy can determine the convergence of training algorithm, how fast and how accurately it converges. It also has an effect on network's ability to generalize.

The common goal of weights initialization is to set them in a way that each neuron produces different activation. This motivates to initialize the weights in some random way depending on the activation function used for nonlinearity. The common way is to initialize the weights randomly from a zero mean standard normal distribution. For ReLU activations modified "Xavier" initialization [67] has been proved to be a good initialization decision in [55]. It was used in this work and it is simply zero mean normal distribution with a standard deviation of $\frac{2}{\sqrt{n}}$, where n is the number of connections of response from the previous layer.

Alternatively, having more computational resources, the initial scale of each layer's weights can be treated as a hyper-parameter and be tuned using, for example, optimization technique recently proposed in [68].

As it is generally recommended, all the biases in convolutional layers were set to zero. In the last fully connected layer they are set to 4000 to obtain the right statistics of the output coordinates (see Section 5-2).

5-4-2 Cost Function

Cost function (or loss function) to be minimized during training is simply the Mean per Joint Position Error (MPJPE) shown in 4-2-5. This squared difference between the true and desired joint locations is a good indication of performance and it satisfies the regression goal of this thesis.

Selection of the cost function can be more complicated for classification tasks when using sigmoid activations. As this is out of the scope of this thesis, it will not be discussed.

5-4-3 Learning Algorithm and Optimizations

Learning algorithm chosen to train the proposed network is vanilla mini-batch SGD which uses the gradient information from a small number of random training samples (so called mini-batches) to update network's parameters. In this way the gradient is approximated for the entire training dataset in one training epoch. It is the most often used algorithm in today's deep learning research and there were many improvements introduced to overcome some challenges that arise using it:

- Difficulty to choose a proper learning rate. A learning rate that is too small leads to slow convergence, while a learning rate that is too large can prevent convergence and cause the loss function to fluctuate around the minimum or even to diverge.
- Difficulty to avoid being stuck in local minima or saddle points where one dimension slopes up and another slopes down [69].

In this thesis the learning rate was selected by manual experiments and set to 0.00001. The only optimization implemented was momentum of 0.9 [70] which resulted in significant results improvement. Momentum is a method that helps to speed up SGD to move to desired direction by introducing a so called velocity. Velocity is the direction and speed at which the parameters move through parameter space. It is set to an exponentially decaying average of the negative gradient. A momentum parameter determines how fast the contributions of previous gradients exponentially decay.

There exist many other optimization methods for SGD such as Nesterov Accelerated Gradient (NAG) or algorithms with adaptive learning rates - Adagrad [71], Adadelta [72], RMSprop [73], Adam [74].

5-4-4 Regularization

Due to a larger number of hyper-parameters, there is a big risk for model to overfit on training data. Having a very large and diverse training data may overcome this risk. However, this is not always possible. Therefore, many regularization techniques have been developed to prevent overfitting.

The simplest method is to monitor the accuracy of the network and stop the training when it is no longer increasing. This procedure is called early stopping. In order to monitor the accuracy and prevent overfitting on the test set, a common practice is to use a validation set.

Recently proposed promising technique is batch normalization [75]. It provides a way of reparametrizing a deep network which significantly reduces the problem of coordinating updates across many layers. Batch normalization can be applied to any input or hidden layer in a network. Other techniques include well known $\mathbf{L}_1$ and $\mathbf{L}_2$ Regularization, Dropout and DropConnect [76].

Due to the large amount of samples provided with Human3.6M dataset, in this work only the early stopping technique was used with patience set to 15 epochs. However, in the future it would be beneficial to try some of the other regularization techniques to improve the results, especially if the model is going to be tested on other datasets.

Experiments and Results

This chapter describes the experiments done to build a good performing CNN model and the results achieved on the selected Human3.6M dataset. It also covers network's output tuning techniques used to improve the results.

The obtained results were officially evaluated on the selected dataset's website. Comparison was also done with other recently reported results. However, they were obtained by testing on the subset of dataset to which ground truth information was provided and cannot be objectively compared using the evaluation server.

70% of the experiments were performed on Nvidia GeForce GT 755M¹ and the rest on Nvidia GeForce GTX 760M², both with 2GB of memory. The implementation is written in Python using Theano library [77].

The training and testing speed of one sample, consisting of 5 RGB video frames having 128×128 resolution, is ~0.025s and ~0.014s on GTX 760M and ~0.045s and ~0.025s on GT 755M respectively.

6-1 CNN Building Experiments

To select the structure of network's layers, convolutional and pooling kernel sizes and number of feature maps, experiments were completed with different settings starting with small simple network and small subset of data.

Some of the experiments with network's structure are shown in Table 6-1. One column represents one architecture, where C stands for convolutional layer followed by a number

¹www.geforce.com/hardware/notebook-gpus/geforce-gt-755m

²www.geforce.com/hardware/notebook-gpus/geforce-gtx-760m

of output feature maps and kernel size, P - pooling layer followed by kernel size and F - fully connected layer. The table heading shows how much the error decreased compared with the starting base model.

For the final architecture, the size of feature maps was reduced in order to be able to train the model with more data. Generally, the design choices were made more arbitrarily based on personal experience rather than following some structure.

Table 6-1: Selected experimental CNN building steps

Base	-8%	-16%	-18%	-21%	-30%	Final
C-10-(3,5,5)	C-10-(3,5,5)	C-10-(3,5,5)	C-10-(3,5,5)	C-10-(3,5,5)	C-10-(3,5,5)	C-5-(3,5,5)
P-(2,2,2)	P-(2,2,2)	P-(2,2,2)	P-(2,3,3)	P-(2,2,2)	P-(2,2,2)	P-(1,2,2)
C-20-(2,5,5)	C-20-(2,5,5)	C-20-(2,5,5)	C-20-(2,5,5)	C-20-(2,5,5)	C-20-(2,5,5)	C-10-(2,5,5)
P-(1,2,2)	P-(1,2,2)	P-(1,2,2)	-	P-(1,2,2)	P-(1,2,2)	P-(1,2,2)
C-40-(1,5,5)	C-40-(1,5,5)	C-40-(1,5,5)	C-40-(1,5,5)	C-40-(1,5,5)	C-40-(1,5,5)	C-20-(1,5,5)
-	-	-	C-60-(1,5,5)	C-60-(1,3,3)	C-60-(1,3,3)	C-40-(1,3,3)
-	-	-	C-60-(1,5,5)	C-60-(1,3,3)	C-60-(1,3,3)	C-40-(1,3,3)
-	-	-	C-60-(1,5,5)	-	-	-
P-(1,2,2)	P-(1,2,2)	P-(1,2,2)	P-(1,3,3)	P-(1,2,2)	P-(1,2,2)	P-(1,2,2)
-	-	F	-	-	-	-
F	F	F	F	F	F	F

6-2 Output Tuning

As described in Section 5-2, the shape of the output is 4D tensor containing estimated human body positions for 5 frames. In this way, for one video frame there are up to 5 different pose estimations obtained when testing model with all possible samples of one video. In order to get the final output, it is possible to apply some simple statistics for those multiple estimations, such as minimum, maximum, average, median or just to select middle frame's estimation. After experimenting with testing subjects, average showed the best results. Also all the center pelvis bone locations were set to zero.

Full-body training results showed that it was hard for the network to estimate locations of hands. To overcome this challenge it was tried to train the same network just with upper body positions and then update (overwrite) the results obtained from the "full body" network. Updates were performed for the all upper body positions and just for two hand joint locations. Results of such approaches are shown in the next section.

No additional output tuning techniques were used in this work.

6-3 Results

There were 8 successful network trainings completed using different data parameters (see Section 5-1). All the results evaluated on dataset's server are shown in Table 6-3. Number "1" in Network's name (first column) stands for training without momentum, "Mom" - with added momentum (see Section 5-4-3). "UpperBody"/"Hands" defines if results were updated with upper body/hands estimations as explained in previous section. The best results were obtained by the network trained on more data with momentum and with updated upper body positions.

Table 6-2: Results of all experiments evaluated on Human3.6M dataset's server

Network	Data used	Average error, mm
3DCNN-1	Setting 1	143
3DCNN-Mom	Setting 1	139
3DCNN-Mom-UpperBody	Setting 1	137
3DCNN-Mom-Hands	Setting 1	138
3DCNN-Mom	Setting 2	130
3DCNN-Mom	Setting 3	132
3DCNN-Mom-UpperBody	Setting 3	129
3DCNN-Mom-Hands	Setting 3	130

In Table 6-3 the best results are compared with state of the art reported on the dataset's website. The latter results were obtained by linear (random feature) approximation of the kernel dependency estimation method using a pyramid of Scale Invariant Feature Transform (SIFT) features extracted on images on which a background subtraction mask was applied.

It can be seen that CNN performs better on 9 actions and the Mean per Joint Position Error (MPJPE) is 3% smaller on average. However, the model performs worse on the actions where people are sitting on the chair or on the ground showing difficulties to deal with body part occlusions. All the numbers are MPJPEs in millimeters.

Some selected examples of good (left) and bad (right) pose estimation results are shown in Figure 6-3.

By the time of working on this thesis there were two other papers released which report 3D pose estimation results on Human3.6M dataset.

A discriminative approach to 3D human pose estimation using spatiotemporal features (HOG-KDE) is presented in [78]. It consists of the following steps:

- A person is detected in 24 consecutive frames;
- The corresponding image windows are shifted so that the subject remains centered;
- A data volume is formed by concatenating these aligned windows;

Table 6-3: Results comparing with state of the art

		3DCNN-Mom-UpperBody	KDE [47]
Directions	15%	100	117
Discussion	9%	98	108
Eating	-21%	110	91
Greeting	13%	112	129
Phoning	-15%	120	104
Posing	12%	114	130
Purchases	-3%	138	134
Sitting	-18%	159	135
Sitting Down	-19%	238	200
Smoking	-5%	123	117
Taking Photo	17%	162	195
Waiting	13%	115	132
Walking	7%	107	115
Walking With Dog	7%	150	162
Walking Together	26%	115	156
AVERAGE	3%	129	133

- A pyramid of 3D HOG features is extracted densely over the volume;
- The 3D pose in the central frame is obtained by Kernel Dependency Estimation (KDE) method.

More similar to this work is 3D pose estimation framework (2DCNN-EM) presented in [21]. It estimates 3D positions by performing the following steps:

- A CNN is trained to predict the uncertainty maps of the 2D joint locations similarly as in [14];
- Expectation-Maximization algorithm is used over the entire sequence to estimate 3D camera parameters. It is shown that the 2D joint location uncertainties can be marginalized out during inference.

The main drawback of both approaches is that they utilize a large number of frames in a sequence comparing to the proposed 3D CNN method. On the other hand, the results reported are better. It is disappointing that official Human3.6M dataset’s evaluation server was not used to objectively evaluate results of mentioned works. Comparable results on two subjects (S9 and S11) are shown in Table 6-3. The proposed method shows better results only on “Posing” action.

Table 6-4: Results comparison with recent works on S9, S11 subjects data

	2DCNN-EM	HOG-KDE	3DCNN
Directions	87	102	104
Discussion	109	148	131
Eating	87	88	125
Greeting	103	127	126
Phoning	116	118	140
Posing	107	114	105
Purchases	100	108	147
Sitting	125	136	174
Sitting Down	199	206	252
Smoking	107	118	133
Taking Photo	143	185	172
Waiting	118	147	123
Walking	79	66	96
Walking With a Dog	114	128	165
Walking Together	98	77	117
AVERAGE	113	124	140

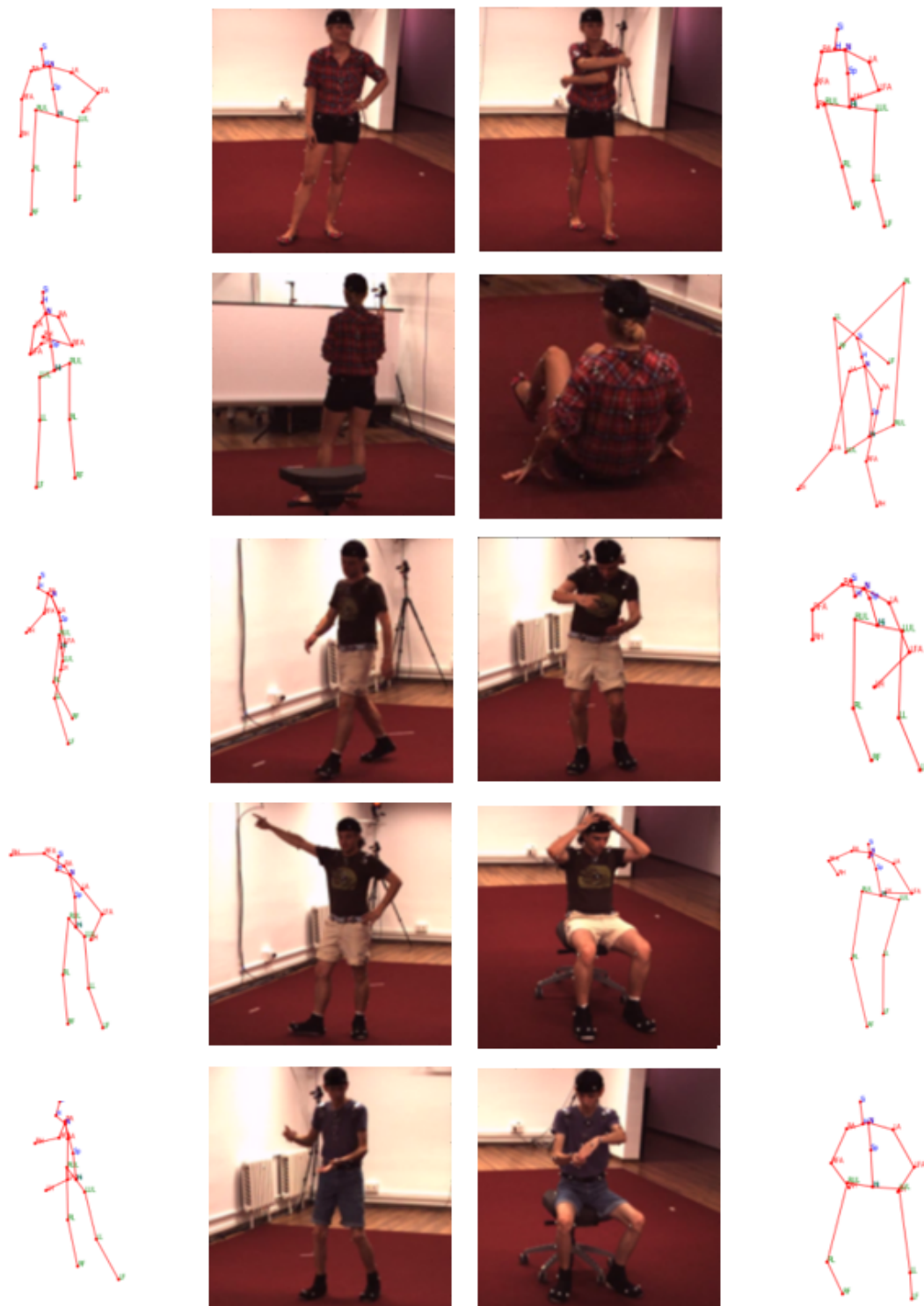


Figure 6-1: Visualization of some good (left) and bad (right) 3D pose estimation results

Conclusions

In this thesis a discriminative 3D CNN model was implemented for the task of human pose estimation in camera coordinate space using RGB video data. It is the first attempt to utilize 3D convolutions for the formulated task.

Through this thesis, an extensive review of publicly available datasets that could be used for defined task was conducted. It has shown that there is a lack of available benchmark datasets applicable for large-scale 3D human body representation learning methods. There is also big diversity in ground truth skeleton data formats and the way it is provided which complicates consolidation of data coming from different sources. There is also a lack of unified evaluation protocols. Based on dataset review, the most applicable and largest dataset was chosen for this thesis. After analysis of the selected dataset, data preprocessing, sampling and network's input preparation tasks were completed.

The 3D CNN model was built having limited resources (in terms of computational power, time and available data variety) and based on related literature and review of similar CNN research works. It was shown that such a model can cope with 3D human pose estimation in videos and outperform the existing methods on the selected dataset. Manual selection of hyper-parameters and theoretical knowledge proved to serve well for this thesis objective.

Proposed model was officially tested on dataset provider's evaluation server and compared with other reported results. Empirical comparison with recently presented results of other two approaches showed that proposed model performs better only on one action and thus has limitations. Limitations of the proposed model include difficulties in estimating highly varied hands locations, also coping with self occlusions and complex poses especially when a person is sitting or lying.

In summary, this thesis is a proof of concept that a compact 3D CNN model can be successfully applied for 3D human pose representation learning and can be further developed.

There is a number of possible future work directions extending this work:

- Implementation of novel CNN training techniques outlined but not tested in this thesis could possibly lead to more accurate estimations;
- Exploration of defined model's weaknesses and possibilities of related improvements;
- Testing model's capabilities on other available datasets;
- Combining the proposed model with Recurrent Neural Network for human body pose tracking and prediction tasks;
- Model's usability analysis for the real-world applications.

Bibliography

- [1] Mart Riess Jones. Time, our lost dimension: Toward a new theory of perception, attention, and memory. *Psychological Review*, 83:323–355, 1976. 1
- [2] Jennifer J Freyd. Dynamic mental representations. *Psychological review*, 94(4):427, 1987. 1
- [3] Lars Michels, Markus Lappe, and Lucia Maria Vaina. Visual areas involved in the perception of human movement from dynamic form analysis. *Neuroreport*, 16(10):1037–1041, 2005. 1
- [4] Marie-Hélène Grosbras, Susan Beaton, and Simon B Eickhoff. Brain regions involved in human movement perception: A quantitative Voxel-Based Meta-Analysis. *Human brain mapping*, 33(2):431–454, 2012. 1
- [5] Seth B Agyei, FR Ruud van der Weel, and Audrey LH Van der Meer. Development of visual motion perception for prospective control: Brain and behavioral studies in infants. *Frontiers in psychology*, 7, 2016. 1
- [6] Chunyu Wang, Yizhou Wang, Zhouchen Lin, Alan Yuille, and Wen Gao. Robust estimation of 3D human poses from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2361–2368, 2014. 1
- [7] Sijin Li and Antoni B. Chan. 3D human pose estimation from monocular images with deep convolutional neural network. In *Computer Vision - ACCV 2014 - 12th Asian Conference on Computer Vision, Singapore, Singapore, November 1-5, 2014, Revised Selected Papers, Part II*, pages 332–347, 2014. 1, 3-3, 3-8
- [8] Georgia Gkioxari, Bharath Hariharan, Ross B. Girshick, and Jitendra Malik. R-CNNs for pose estimation and action detection. *CoRR*, abs/1406.5212, 2014. 1, 3-3

- [9] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Hands deep in deep learning for hand pose estimation. *arXiv preprint arXiv:1502.06807*, 2015. 1
- [10] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013. 1
- [11] Yonghui Du, Yan Huang, and Jingliang Peng. Full-Body human pose estimation from monocular video sequence via Multi-Dimensional boosting regression. In *Computer Vision-ACCV 2014 Workshops*, pages 531–544. Springer, 2014. 1
- [12] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. *CoRR*, abs/1411.4280, 2014. 1, 3-2, 3-4
- [13] Arjun Jain, Jonathan Tompson, Yann LeCun, and Christoph Bregler. Modeep: A deep learning framework using motion features for human pose estimation. In *Computer Vision-ACCV 2014*, pages 302–315. Springer, 2014. 1
- [14] Tomas Pfister, James Charles, and Andrew Zisserman. Flowing convnets for human pose estimation in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1913–1921, 2015. 1, 3-6, 6-3
- [15] Tomas Pfister, Karen Simonyan, James Charles, and Andrew Zisserman. Deep convolutional neural networks for efficient pose estimation in gesture videos. In *Asian Conference on Computer Vision (ACCV)*, 2014. 1, 3-2, 3-5
- [16] Alexander Toshev and Christian Szegedy. DeepPose: Human pose estimation via deep neural networks. *CoRR*, abs/1312.4659, 2013. 1, 3-2, 3-3, 3-2, 3-2
- [17] Sijin Li, Zhi-Qiang Liu, and Antoni Chan. Heterogeneous Multi-Task learning for human pose estimation with deep convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 482–489, 2014. 1
- [18] Arjun Jain, Jonathan Tompson, Mykhaylo Andriluka, Graham W. Taylor, and Christoph Bregler. Learning human pose estimation features with convolutional networks. *CoRR*, abs/1312.7302, 2013. 1, 3-7, 3-2
- [19] Xiaochuan Fan, Kang Zheng, Yuewei Lin, and Song Wang. Combining local appearance and holistic view: Dual-Source deep neural networks for human pose estimation. *CoRR*, abs/1504.07159, 2015. 1, 3-3, 3-9
- [20] Feng Zhou and Fernando De la Torre. Spatio-Temporal matching for human pose estimation in video. 2016. 1

- [21] Xiaowei Zhou, Menglong Zhu, Spyridon Leonardos, Kosta Derpanis, and Kostas Daniilidis. Sparseness meets deepness: 3D human pose estimation from monocular video. *arXiv preprint arXiv:1511.09439*, 2015. 1, 6-3
- [22] Tsz-Ho Yu, Tae-Kyun Kim, and Roberto Cipolla. Unconstrained monocular 3D human pose estimation by action detection and Cross-Modality regression forest. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649, 2013. 1
- [23] Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Reconstructing 3D human pose from 2D image landmarks. In *Computer Vision–ECCV 2012*, pages 573–586. Springer, 2012. 1
- [24] James Bergstra and Yoshua Bengio. Random search for Hyper-Parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012. 1-1
- [25] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2011. 1-1
- [26] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012. 1-1
- [27] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3D convolutional neural networks for human action recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(1):221–231, January 2013. 1-1, 3-4, 3-10
- [28] Keze Wang, Xiaolong Wang, Liang Lin, Meng Wang, and Wangmeng Zuo. 3D human activity recognition with reconfigurable convolutional neural networks. *CoRR*, abs/1501.06262, 2015. 1-1, 3-11
- [29] Gül Varol, Ivan Laptev, and Cordelia Schmid. Long-term temporal convolutions for action recognition. 2015. 1-1
- [30] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential deep learning for human action recognition. In *Human Behavior Understanding*, pages 29–39. Springer, 2011. 1-1
- [31] Divya R Pillai and P Nandakumar. Crowd behavior analysis using 3D convolutional neural network. 2014. 1-1
- [32] Pavlo Molchanov, Shalini Gupta, Kihwan Kim, and Jan Kautz. Hand gesture recognition with 3D convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–7, 2015. 1-1

- [33] CS231n convolutional neural networks for visual recognition. <http://cs231n.github.io/neural-networks-1/>. (Accessed on 05/27/2016). 2-1
- [34] Paul J Werbos. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990. 2-1
- [35] Randall C. O'Reilly, Yuko Munakata, Michael J. Frank, Thomas E. Hazy, and Contributors. *Computational Cognitive Neuroscience*. Wiki Book, 1st Edition, URL: <http://ccnbook.colorado.edu>, 2012. 2-2
- [36] Kunihiro Fukushima. Neocognitron: A Self-Organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980. 2-2
- [37] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998. 3-1, 3-1
- [38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 3-1, 3-2, 5-3-1, 5-3-2
- [39] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 3-1, 4-1
- [40] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a “siamese” time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993. 3-2
- [41] Fei Han, Brian Reily, William Hoff, and Hao Zhang. Space-Time representation of people based on 3D skeletal data: A review. *CoRR*, abs/1601.01006, 2016. 4-1
- [42] Rene Vidal, Ruzena Bajcsy, Ferda Ofli, Rizwan Chaudhry, and Gregorij Kurillo. Berkeley MHAD: A comprehensive multimodal human action database. In *Proceedings of the 2013 IEEE Workshop on Applications of Computer Vision (WACV)*, WACV '13, pages 53–60, Washington, DC, USA, 2013. IEEE Computer Society. 4-1-1
- [43] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Human activity detection from RGBD images. In *In AAAI workshop on Pattern, Activity and Intent Recognition (PAIR)*, 2011. 4-1-2

- [44] Hema Swetha Koppula, Rudhir Gupta, and Ashutosh Saxena. Learning human activities and object affordances from RGB-D videos. *Int. J. Rob. Res.*, 32(8):951–970, July 2013. 4-1-2
- [45] Fernando de la Torre, Jessica K. Hodgins, Javier Montano, and Sergio Valcarcel. Detailed human data acquisition of kitchen activities: the CMU-Multimodal activity database (CMU-MMAC). In *CHI 2009 Workshop. Developing Shared Home Behavior Datasets to Advance HCI and Ubiquitous Computing Research*, 2009. 4-1-3
- [46] C3D.ORG. <https://www.c3d.org/>. (Accessed on 05/13/2016). 4-1-3
- [47] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014. 4-1-4, 6-3
- [48] Leonid Sigal, Alexandru O. Balan, and Michael J. Black. HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *Int. J. Comput. Vision*, 87(1-2):4–27, March 2010. 4-1-5
- [49] Abdallah Dib and François Charpillet. Pose Estimation For A Partially Observable Human Body From RGB-D Cameras. In *IEEE/RJS International Conference on Intelligent Robots and Systems (IROS)*, page 8, Hamburg, Germany, September 2015. 4-1-6
- [50] Gerard Pons-Moll, Andreas Baak, Thomas Helten, Meinard Müller, Hans-Peter Seidel, and Bodo Rosenhahn. Multisensor-fusion for 3D full-body human motion capture. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2010. 4-1-7
- [51] Andreas Baak, Thomas Helten, Meinard Müller, Gerard Pons-Moll, Bodo Rosenhahn, and Hans-Peter Seidel. Analyzing and evaluating markerless motion tracking using inertial sensors. In *European Conference on Computer Vision (ECCV Workshops)*, September 2010. 4-1-7
- [52] Human3.6M dataset. <http://vision.imar.ro/human3.6m/description.php>. (Accessed on 05/15/2016). 4-2
- [53] Rodney J. Douglas and Kevan A.C. Martin. Recurrent neuronal circuits in the neo-cortex. *Current Biology*, 17(13):R496 – R500, 2007. 5-3-1
- [54] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey J. Gordon and David B. Dunson, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, volume 15, pages 315–323. Journal of Machine Learning Research - Workshop and Conference Proceedings, 2011. 5-3-1

- [55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing Human-Level performance on ImageNet classification. *CoRR*, abs/1502.01852, 2015. 5-3-1, 5-4-1
- [56] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 5-3-1
- [57] Ian J. Goodfellow, David Warde-farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *In ICML*, 2013. 5-3-1
- [58] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013. 5-3-1
- [59] Forest Agostinelli, Matthew Hoffman, Peter J. Sadowski, and Pierre Baldi. Learning activation functions to improve deep neural networks. *CoRR*, abs/1412.6830, 2014. 5-3-1
- [60] Kevin Jarrett, Koray Kavukcuoglu, Marc’Aurelio Ranzato, and Yann LeCun. What is the best Multi-Stage architecture for object recognition? In *ICCV*, pages 2146–2153. IEEE, 2009. 5-3-2
- [61] Neslihan Bayramoglu, Juho Kannala, and Janne Heikkilä. Human epithelial type 2 cell classification with convolutional neural networks. In *15th IEEE International Conference on Bioinformatics and Bioengineering, BIBE 2015, Belgrade, Serbia, November 2-4, 2015*, pages 1–6, 2015. 5-3-2
- [62] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *Artificial Neural Networks—ICANN 2010*, pages 92–101. Springer, 2010. 5-3-4
- [63] Matthew D Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*, 2013. 5-3-4
- [64] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1458–1465. IEEE, 2005. 5-3-4
- [65] Benjamin Graham. Fractional Max-Pooling. *arXiv preprint arXiv:1412.6071*, 2014. 5-3-4
- [66] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014. 5-3-4

- [67] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feed-forward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010. 5-4-1
- [68] Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015. 5-4-1
- [69] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in High-Dimensional Non-Convex optimization. In *Advances in neural information processing systems*, pages 2933–2941, 2014. 5-4-3
- [70] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999. 5-4-3
- [71] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011. 5-4-3
- [72] Matthew D Zeiler. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. 5-4-3
- [73] Yann N Dauphin, Harm de Vries, Junyoung Chung, and Yoshua Bengio. RMSProp and equilibrated adaptive learning rates for Non-Convex optimization. *arXiv preprint arXiv:1502.04390*, 2015. 5-4-3
- [74] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5-4-3
- [75] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 5-4-4
- [76] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013. 5-4-4
- [77] Theano Development Team. Theano: A Python Framework for Fast Computation of Mathematical Expressions. *arXiv e-prints*, abs/1605.02688, May 2016. 6
- [78] Bugra Tekin, Xiaolu Sun, Xinchao Wang, Vincent Lepetit, and Pascal Fua. Predicting people’s 3D poses from short sequences. *arXiv preprint arXiv:1504.08200*, 2015. 6-3

Glossary

List of Acronyms

2D	Two-Dimensional
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DNN	Deep Neural Network
fMRI	Functional Magnetic Resonance Imaging
GCN	Global Contrast Normalization
KDE	Kernel Dependency Estimation
LCN	Local Contrast Normalization
LSBP	Latent Structural Back Propagation
MLNN	Multi-Layer Neural Network
MPJPE	Mean per Joint Position Error
MSE	Mean Squared Error
NAG	Nesterov Accelerated Gradient
PReLU	Parametric Rectified Linear Unit
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RGB	Red-Green-Blue (color model based on additive color primaries)

SGD Stochastic Gradient Descent

SIFT Scale Invariant Feature Transform

