TÉCNICO
LISBOA

Recognition of Human Activity in Domestic Environments Using Convolution Neural Networks

Tiago Rodrigues Borges Ferreira

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineer

Supervisor(s): Prof. Pedro Manuel Urbano de Almeida Lima
Prof. Jacinto Carlos Marques Peixoto do Nascimento

Examination Committee

Chairperson: João Fernando Cardoso Silva Sequeira
Supervisor: Jacinto Carlos Marques Peixoto do Nascimento
Member of the Committee: Pedro Manuel Quintas Aguiar

May 2016

Acknowledgments

First of all I want to thank my supervisors, Prof. Pedro Lima and Prof Jacinto Nascimento for all the help regarding this thesis. Second I thank my girlfriend (Carolina Freitas) for helping me in the writing process and on building the dataset. Third I also would like to thank the help of my colleagues Alexandre Gomes and Bernardo Marchante for helping me on bulding the dataset. And last but not least my family for the support not only in the thesis but along all my studies.

Resumo

Com o avanço da tecnologia os robots estão a tornar-se cada vez mais essenciais para a vida humana. De forma a tornar os robots mais autónomos, investigadores têm-se focado em criar algoritmos de processamento de imagem cada vez mais robustos e capazes de interagir com o mundo real. Esta tese aborda um *case study* próximo da realidade onde são aplicadas redes convolucionais. O conceito do *case study* trabalhado é a classificação de atividades levadas a cabo por humanos num ambiente doméstico. O objetivo é determinar, em tempo-real, se o comportamento do(s) sujeito(s) é normal ou anormal (potencialmente perigoso). A criação de um modelo como este permitirá a vigilância instantânea e contínua em todas as câmeras conectadas ao algoritmo. Nesta tese foi criado um dataset extenso sendo o mesmo aplicado a diferentes estruturas de redes de forma a determinar as arquiteturas que obtêm os melhores resultados. Os resultados finais mostram que redes convolucionais podem ser aplicadas com sucesso a problemas de processamento de imagem, obtendo resultados tão bons ou superiores aos obtidos com os métodos clássicos de processamento de imagem.

Palavras-chave: redes-convolucionais, machine-learning, processamento-de-imagem

Abstract

With the advance of technology robots are becoming essential in humans daily lives. In order to make robots more autonomous lots of efforts are being made by researchers to create robust and efficient image processing algorithms capable of interacting with the real world. This thesis addresses a semi real-live situation by applying convolution neural networks. The case study concept is the classification of humans activities performed in a domestic scenario. The main goal is to determine if the human real-time continuous behavior is either normal or abnormal (potentially dangerous). The creation of such model would allow instant and continuously surveillance in every connected camera. In this thesis an extensive dataset was created and applied to different net architectures to determine which obtain better results. The final results are very positive showing that convolution neural networks can successfully be applied to image processing situations obtaining results that easily equalise or even overcome classic methods from computer vision.

Keywords: convolution-neural-networks, image-processing, machine-learning

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

A new era of domestic robots is rising and many robotic investigation teams are focusing their efforts on developing robots that can better perceive the world around them, increasing their grade of automation. Over the last several years, Machine Learning techniques, particularly Neural Networks, have played an increasingly important role in the design of pattern recognition systems. This thesis follows Lecun et al. [1] that says that ”better pattern recognition systems can be built by relying more on automatic learning, and less in hand-designed heuristics”. This thesis is integrated in the SocRob@Home project which is a project from an ISR (Institute for Systems and Robotics) team that competes in robotic competitions such as RoCKIn[2] and RoboCup. This team focus is the scientific research in domestic robots. The main purpose of this study is to help the team developing a model that correctly identifies abnormal behaviors in indoor scenarios (ex: visitors in a house). The model must learn the correct classification using a labeled dataset containing both normal and abnormal behaviors.

1.2 State of the art

In this section is described how Artificial Neural Networks (ANNs) evolved until reaching the current state and in what situations ANNs proved to be successful.

ANN research has experienced three periods of extensive activity. The first peak in the 1940s was due to McCulloch and Pitts' pioneering The second occurred in the 1960s with Rosenblatt's perceptron convergence theorem and Minsky and Papert's work showing the limitations of a simple perceptron.

In machine learning and cognitive science, artificial neural networks are a family of models inspired by biological neural networks (the central nervous systems of animals, in particular the brain) which are used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown.

Artificial Neural Networks appeared in the 1940s due to McCulloch and Pitts' pioneering. The second pick occurred in the 1960s with Rosenblatt's perceptron convergence theorem and Minsky and Papert's work showing the limitations of a simple perceptron. The net could emulate a significant number of functions. However the algorithm was slow and not very efficient. In 2005 Deep Learning emerged. As the name suggests, the base in Deep Learning is in the use of deeper networks with much more layers and units per layer. Deep learning kept on growing with the creation of new powerful algorithms and techniques and with the increase of computational power. When trying to apply neural networks to more complex datasets, such as image processing, come the necessity of developing methods that could accept 2D inputs instead of only 1D. One of those methods is known as Convolutional Neural Networks (CNNs) [3]. Convolutional Neural Networks are mostly used in image and video processing. A big challenge in image processing is the size of the inputs. For instance, an image with a resolution of 100x100 pixels will have 10000 inputs times 3 channels (RGB). The first layers of a CNN are 2D and make the convolution between the image and the learned kernels (filters). The final layers are 1D (fully connected). The basic idea of CNNs is that the net aims to learn which kernels weights are the most effective. Each layer represents the input image in a different way. The first layers are considered low level since they extract elementary visual features such as oriented edges, end-points and corners. In deeper layers this features are then combined in order to detect high-order features (see Figure 1.1).

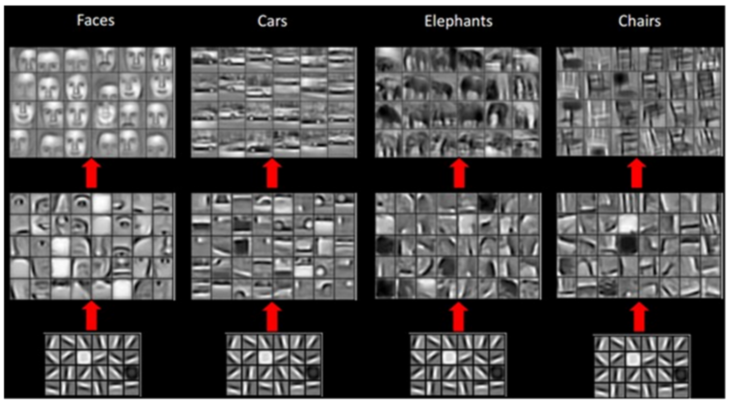

Figure 1.1: Feature extraction of kernels along the multiple layers [4]

Deep learning proved to give positive results in a wide variety of research areas. The beginnings of deep learning in 2006 focused on the MNIST [1] digit image classification problem [5] [6]. The latest records are still held by deep networks: Ciresan et al. [7] currently claim the title of state of the art for the unconstrained version of the task (e.g., using a convolutional architecture), with 0.27 percent error.

Microsoft released in 2012 a new version of their Microsoft Audio Video Indexing Service speech system based on deep learning[8]. These authors managed to reduce the word error rate on four major benchmarks by about 30 percent compared to state-of-the-art models based on Gaussian mixtures for the acoustic modeling and trained on the same amount of data (309 hours of speech).

These techniques have been applied in a big variety of datasets; not only in image processing but also in other areas, for example speech recognition. The Convolution Neural Networks were introduced by Fukushima et al.[9] and further developed by Lecun et al. [1] in 1998. Since then, many other researchers have published papers presenting results that outperformed the older methods that used

[1]Mixed National Institute of Standards and Technology database

hand-designed features selectors like SIFT [10] and HOG [11]. In Krizhevsky et al. [12], a deep Convolutional Neural Network was trained to classify 1.2 million high resolution images in the ImageNet LSVRC/2010 contest into 1000 different classes. Their network achieved top-1 and top-5 [2] test error rates of 37.5% and 17.0% where the best performance achieved during the ILSVRC-2010 competition was 47.1% and 28.2% with an approach that averages the predictions produced from six sparse/coding models trained on different features [13]. Krizhevsky et al. [12] used a network containing eight learned layers (five convolutional and three fully-connected). In order to train the network it was needed two NVIDIA GTX 580 GPU with 3GB of memory. The network was trained for 90 cycles through a training set of 1.2 million images, which took five to six days. In Girshick et al [14] it was proposed an object detection method that consisted in three modules: The first generates category-independent region proposals, the second is a large convolutional neural network that extracts a fixed-length feature vector and the third is a set of class specific linear SVMs (Support Vector Machines). In this paper the CNNS were use as a mean of feature extraction and not classifiers. The authors used a method that combined classical tools from computer vision with deep learning, achieving results of 30% relative improvement over the best previous results on PASCAL VOC 2012.

All the above information is relative to image processing, but as the purpose of this work is to classify behaviors from video sequences it is important to refer some work that has been done in the video processing area. CNNs can be modified to accept videos sequences as input. This can be achieved by modifying the filters on the first convolutional layer in the single-frame model by extending them to be of size $i * j * 3 * T$, where i,j are the pixels positions on the 3 the channels RGB and T represents some temporal extent. Andrej Karpathy et al. [15] provides an empirical evaluation of CNNS on a large scale video classification using a new dataset of 1 million YouTube videos belonging to 487 classes. In this work the authors observed that motion aware network clearly benefits from motion information in some cases. However these seem to be relatively uncommon because in most of the cases single-frame model already displays very strong performance.

1.3 Challenge Description

This thesis challenges can be categorized in two major topics: First, the study and understanding of deep learning and convolution neural networks, focusing on finding the best way to implement them and the best architectures to use. Second, applying this algorithms to a realistic scenario in order to determine how likely convolution neural networks can contribute for the increasing of automation in the area of robotics.

The realistic challenge of this study is to develop a model that correctly identifies abnormal behaviors of humans in a domestic scenario. The model must learn the correct classification using a labeled dataset containing both normal and abnormal behaviors.

As described in the RoCKIn Nutshell [16]: ”RoCKIn@Home is a competition that aims at bringing

[2]top-x error, also known as rank-x error, is the error taking in consideration if the correct classification is present in the top x predictions with the higher level of confidence

together the benefits of scientific benchmarking with the attraction of scientific competitions in the realm of domestic service robotics. The objectives are to bolster research in service robotics for home applications and to raise public awareness of the current and future capabilities of such robot systems to meet societal challenges like healthy ageing and longer independent living.” The basic idea of the scenario of the competition is that there is a elderly, named “Granny Annie”, who lives in an apartment together with some pets. Granny Annie is suffering from typical problems of aging people and the robot must be able to perform a big variety of tasks that help to increase her life quality. The environment is an ordinary European apartment equipped with a big variety of networkable devices such as: ceiling lamps, electric shutters, camera-based intercom at the front door and a net of surveillance cameras spread around the flat (figure 1.2).

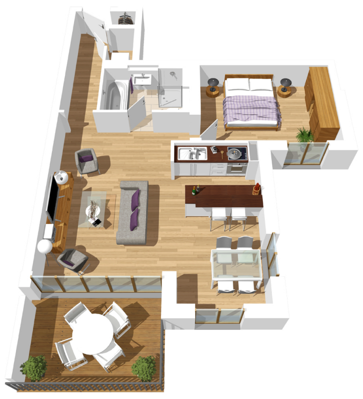

Figure 1.2: Virtual representation of @home competitions scenario [16]

If Granny Annie is not in conditions of standing up the robot has to be able to handle visitors, who arrive and ring the door bell. The robot must correctly decide which visitors are allowed to come in and which ones are not. Even if the visitor is allowed to come in, the robot must always be aware of his actions in order to make sure that the visitor is not doing nothing out of the ordinary.

1.4 Goals

The main purpose of the work developed in this thesis is to create a model that correctly classified actions performed by house visitors in order to identify abnormal behaviors. To reach this goal several intermediate goals were defined. The first intermediate goal was to develop a case study that in one hand was simple enough so that it could be easily implemented and analysed and in the other hand was robust enough so that the results could be compared with a real live scenario. The second intermediate goal was the creation of the dataset following the details of the case study. The dataset should be big enough so that it could be divided into a training and a test set and should be complex enough so that it could embrace a big variety of cases. The third sub-goal was to study different nets structures in order to find the one that best fits the case study and that provides the best outcome. Finally the last goal was to test the model in a real life scenario and measure the obtained results.

1.5 Thesis Outline

The structure of this thesis contains six chapters.

In Chapter 2 (Background) the Convolution Neural Networks background is discussed. First this chapter focus on introducing the Artificial Neural Networks and the algorithm of backpropagation with gradient descent (a common method of training artificial neural networks, the method calculates the gradient of a loss function with respect to all the weights in the network then the gradient is fed to the optimization method which in turn uses it to update the weights, in an attempt to minimize the loss function). Next there is an extensive study about Convolution Neural Networks, describing its main concepts: Local Connectivity (2.2.1), Parameter Sharing (2.2.2), Pooling and Sub-Sampling (2.2.3), Convolution (2.2.4), CNN backpropagation (2.2.5) and Dropout (2.2.6).

Chapter 3 (Case Study and Dataset), describes the main problem that this work proposes to solve and how the dataset was obtained. It is specified the different actions present in the dataset (Actions Description 3.3.1) and some of the frames are illustrated. It is also detailed how the dataset was divided into Training Set and Validation Set (Spliting the Dataset 3.4.1).

In chapter 4 (Resutls) it is described how the results were obtained (section 4.1) and the experimental results are displayed together with their analyses in sections: Two Classes Dataset (section 4.2), Three Classes Dataset (section 4.3) and Four Classes Dataset (section 4.4).

Finally chapter 5 (Conclusion), provides an overview of the thesis, describes the most significant conclusions and proposes possible extensions to the developed work.

Chapter 2

Background

2.1 Artificial Neural Network

2.1.1 Basic Concept

An ANN structure is constituted by one input layer, one output layer and the hidden layers, which are between the first two. Both the unit inputs and the unit outputs are multiplied by weights. Each layer has a certain number of units. The output of a unit is the weighted sum of its inputs plus a bias term. This output is then passed in a sigmoid form function such as the $tanh$. In this section we will explain in more detailed how ANNs work. The net structure described is the Multi-Layer Perceptron (MLP) (Figure 2.1) using the back-propagation method with the algorithm gradient descent.

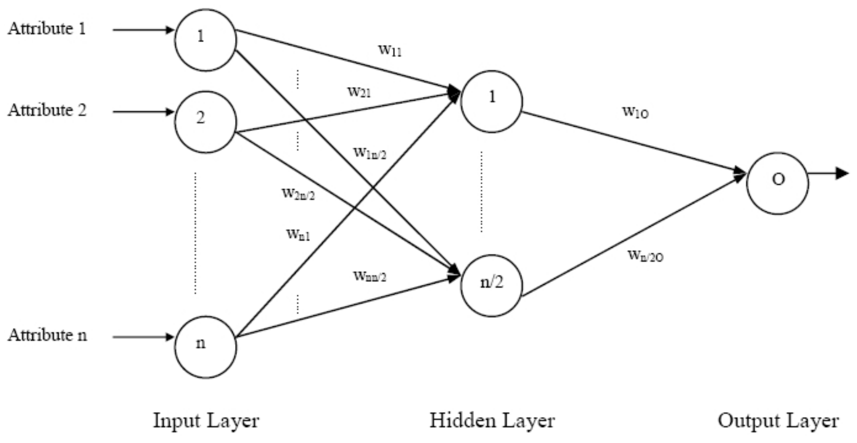

Figure 2.1: Representation of a Multi-Layer Perceptron

A MLP is a feedforward network and it is normally trained using a supervised method where the objective is to minimize a loss function by updating the weights. Once again the main purpose of the back propagation algorithm is to minimize an error function, such as:

$$e(k) = o(k) - d(k) \tag{2.1}$$

being k the training input, $e(k)$ the error, $o(k)$ the output and $d(k)$ the desired output. The loss function most commonly used is the "least square error":

$$E(k) = ||e(k)||^2 \tag{2.2}$$

where $E(k)$ is the loss function.

One common minimization algorithm is the gradient method. This method consists in iteratively updating the weights in order to minimize the loss function:

$$w(n+1) = w(n) - lr * \frac{\delta E}{\delta W} \tag{2.3}$$

being lr the learning rate, w the weights and $\frac{\delta E}{\delta W}$ the derivative of the loss function in relation to the weight.

Firstly, all the parameters of the feedforward net are calculated and the output is obtained. Having the output it is possible to calculate $E(k)$ and the derivative of the error is back-propagated. So the input of the back propagation network is $2 * e(k)$. Next $\frac{\delta E}{\delta W}$ must be calculated for each weight where $\frac{\delta E}{\delta W}$ is equal to the input of the feedforward network times the input of the back-propagation net in the same particularly branch. There are two types of training: the batch mode (offline training) and the online mode. The batch size is the number of samples that are used in each iteration, so in the batch mode all batch inputs are used in the weight update:

$$w(n+1) = w(n) - lr * (\frac{\delta E1}{\delta W} + ... + \frac{\delta En}{\delta W}) \tag{2.4}$$

In online mode the model does not have access to all the dataset at once and the training occurs each time that a new input is available, so:

$$w(n+1) = w(n) - lr * \frac{\delta Eni}{\delta W} \tag{2.5}$$

being ni the new input

2.2 Convolution Neural Networks

The architecture of a CNN can be divided in two parts. The first layers are 2D and make the convolution between the image and the learned kernels. After the convolution layers the net is constituted by 1D fully-connected layers. This last layers are basic artificial neural networks (similar to MLPs).

CNNS have four base concepts: local connectivity, parameter sharing, pooling and sub-sampling and convolution.

2.2.1 Local Connectivity

Local connectivity is the idea of connecting units to local receptive fields on the input. This idea goes back to the MLPs and was almost simultaneous with Hubel and Wiesel's discovery of locally-sensitive, orientation-selective neurons in the cat's visual system [17].

Local connectivity allows layers not to be fully connected to each other but only to be connected to a limited number of units in the next layer. This property allows the local processing of an image.

Figure 2.2 illustrates the property of local connectivity. As can be seen the input image is divided into different regions and each region is only connected to a specific unit from the next layer. When the input has multiple channels (ex: RGB) the same region in each channel is connected to a specific unit, as illustrated in figure 2.3

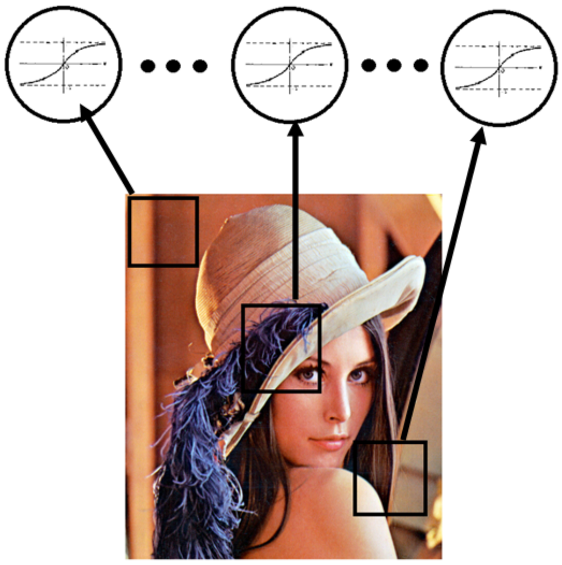

Figure 2.2: Representation of local connectivity

However distortion or shifts of the input can cause the position of salient features to vary. In addition, elementary feature detectors of salient features might be useful across the entire image. This can be accomplished by forcing a set of units, whose receptive fields are located at different places on the image, to have identical weights. This propriety is called parameter sharing.

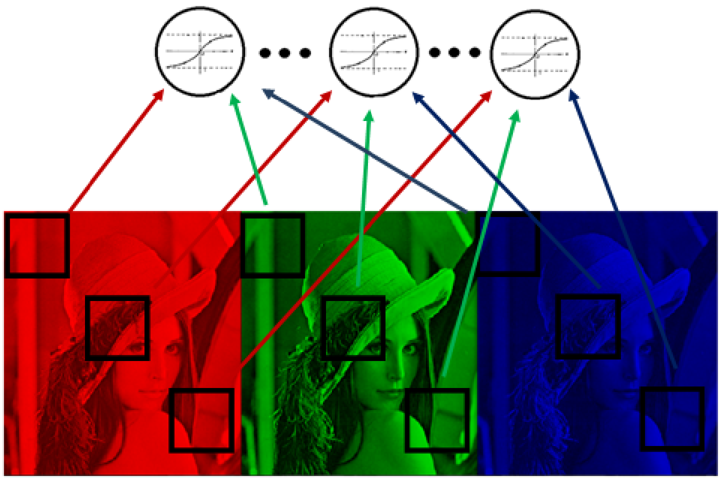
Figure 2.3: Representation of local connectivity in RGB

2.2.2 Parameter Sharing

Parameter Sharing is a property that allows each feature map to be applied to all the receptive fields of the input image. Therefore the weights of each feature map are updated based on all the connected receptive fields (as illustrated in figure 2.4) allowing a significant reduction on the net number of weights.

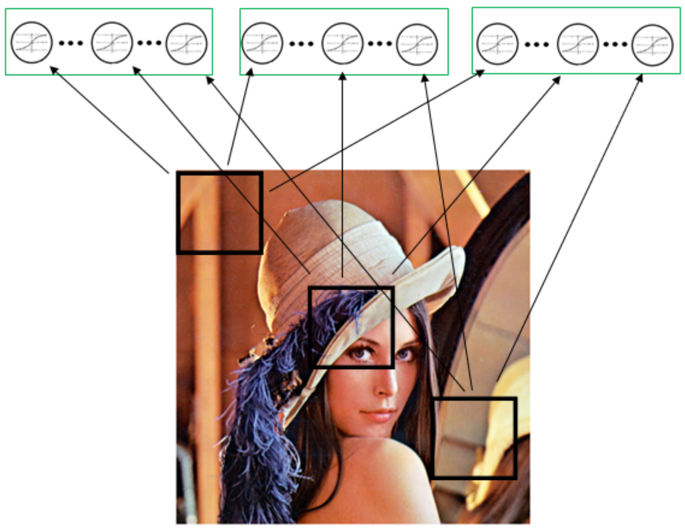

Figure 2.4: Representation of parameter sharing

2.2.3 Pooling and Sub-Sampling

Once the feature has been detected, its exact location becomes less important. It is only relevant the approximate position relative to other features. A good way to illustrate this statement is to think in the characters classification: once we know that the input image contains the endpoint of a roughly horizontal segment in the upper left area, a corner in the upper right area, and the endpoint of a roughly vertical segment in the lower portion of the image, we can tell the input image is a 7. All the other features beyond these are irrelevant and might even contribute to a poorer classification. Pooling and Sub-Sampling is a way to reduce the precision of the feature map in order to highlight the most representative features. The pooling technique takes from a patch of hidden units the most representative value. There are different kinds of pooling but the most used are the max-pooling and the average-pooling. The max-pooling as the name suggests takes the maximum value in a certain neighborhood:

$$y(i,j) = maxN(x(i,j)) \tag{2.6}$$

where $x(i,j)$ is the value of the pixel located in i,j and $N(x)$ is the neighborhood of $x(i,j)$

The average-pooling (equation 2.7), again as the name suggests, takes the average value of a certain neighborhood,

$$y(i,j) = (\frac{1}{m^2}) * \sum N(x(i,j)) \tag{2.7}$$

where m^2 is the squared value of the size of the reception field

The reason why it is also called sub-sampling is because there is a reduction in the feature map resolution. Besides contributing to the elimination of irrelevant features [1] this technique also contributes to save a lot of weights allowing to increase the number o kernels of the next layer without substantial increase of the processing power. In figure 2.5 is represented the result of an ”Average Pooling” and a ”Max Pooling”.

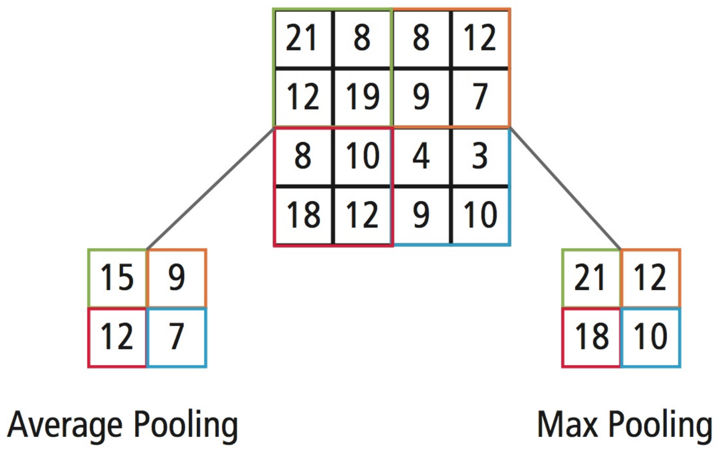

Figure 2.5: Representation of max pooling and average pooling [18] with a reduction of (2,2)

2.2.4 Convolution

The feature maps are created by the discrete convolution of the kernels with a portion of the input (receptive field).

The discrete convolution is very similar to the continuous one but with the difference that it is computed in a finite sequence of points.

$$\sum_{m=-inf}^{+inf} f[m]g[n-m] \tag{2.8}$$

Equation 2.8 describes the convolution between $f[m]$ and $g[m]$. The result will be the sum of the product of $f[m]$ and the reversed and shift n of $g[m]$ for every m.

The matrix convolution has some significant differences from the processes described above. Matrix convolution (figure 2.9) is used in image processing and it is between an image ($I[u,v]$) and a kernel ($k[i,j]$).

$$FM[i,j] = \sum N(I[u,v]) * k[i-u, j-v]) \tag{2.9}$$

The first step to calculate the convolution is to flip the rows and the columns of the Kernel, which means that instead of counting the kernel index from top right as being (0,0) we place the origin index in the bottom right corner. The next step is to overlap the center of the kernel with the image pixel that we want to calculate ($I[i,j]$) and obtain the sum of the overlapped values product.

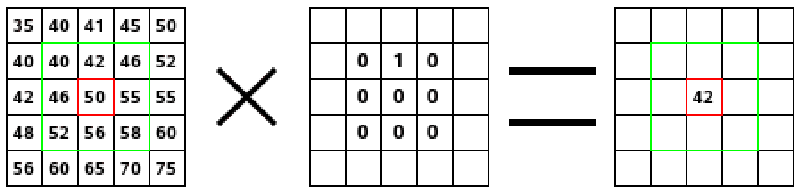

Figure 2.6: Representation of matrix convolution [19]

An helpful property of convolution is that it is shift invariant so if the neighborhood stays the same the image can be shifted and the result will always be very similar. If the input has three channels the output is going to be the sum of the result of the convolution in each channels. Then the result will be the input of a sigmoid (for example a $tanh$). The new feature map will be constituted by the outputs of the non-linear function.

2.2.5 CNN backpropagation

The algorithm of backpropagation in CNN's works in a very similar way than with 1D neural networks. Let's analyse how backpropagation works in convolution units and pooling units.

Convolution Units

The algorithm of back-propagation [9] in CNNs its actually very similar to the one applied to fully connected layers. The aim of each iteration is to obtain a new set of kernel weights ($k[i, j]$) that reduces the model error.

$$k(n+1) = k(n) - lr * \frac{\delta E}{\delta k} \tag{2.10}$$

where lr is the learning rate and $\frac{\delta E}{\delta k}$ is the derivative of the loss function in order of the weights k.

In order to calculate $\frac{\delta E}{\delta k}$ it is also needed to calculate the inputs of the backward net. As illustrated in figure 2.7 instead of determining $y(i,j)$ by making the convolution between $x(i,j)$ and $k(i,j)$ we want to calculate the opposite:

$$x(i,j) = y(i,j) * k(i,j) \tag{2.11}$$

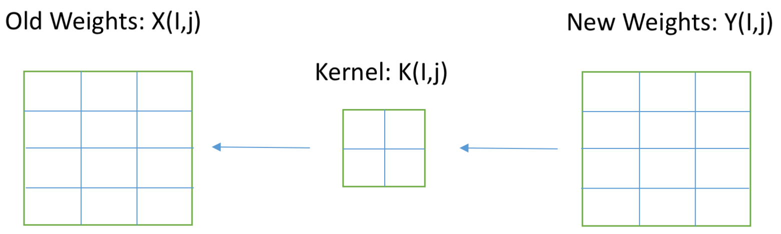

Figure 2.7: Simplified representation of backward propagation

Finally to calculate $\frac{\delta E}{\delta k}$ we need to calculate the convolution between the input of the unit in the feedforward net and the input of the same unit in the backward net.

Pooling Units

For the case of the pooling units let's analyse both the maxpooling and the average pooling situations. In the case of the maxpooling the error derivative will only be backpropagated to the units that were selected as the maximum in the forward propagation. All the other units will receive the value of zero.

```
Result: Backforward maxpooling
if i and j correspond to the max unit then
    x(i,j) = y(a,b)
else
    x(i,j) = 0;
end
```

In the case of the average pooling all the units will receive the same gradient value (upsampling) divided by the m^2 (m^2 is the squared value of the size of the reception field)

$$x(i,j) = \frac{1}{m^2} * y(a,b) \tag{2.12}$$

2.2.6 Dropout

Dropout was introduced by Srivastava et al.[20] as a "Simple Way to Prevent Neural Networks from Overfitting". When training a neural network, specially when having limited training data, there are some noise in the samples that might not be in the real data. This may cause the model to overfit to that noise. Dropout reduces substantially the overfitting by preventing each unit to depend on their nearest neighbours. The term ”dropout” refers from dropping out units which consists in temporarily removing them from the network while the training is occurring. This is accomplished by, for each training iteration, randomly setting to zero the weights of multiple units with a probability of p (the value of p is normally 0.25 or 0.5). The higher the value of p the higher the number of dropped units and consequently units deppend less on their neighbours. In figure 2.8 is illustraded the net architecture before and after dropout where the units with the cross were dropped.

At test time the entire network is used to make a prediction however its weights are a scale down version of the trained weights. That can be accomplished by multiplying them by the value of p used during the training process. Dropout units can also be used with the backprogation algorithm with gradient descent. In this case some of the units are dropped out and a thinned version of the network is used for the back and forward propagation. Each dropped out unit contributes with a gradient of 0 for the connected parameters.

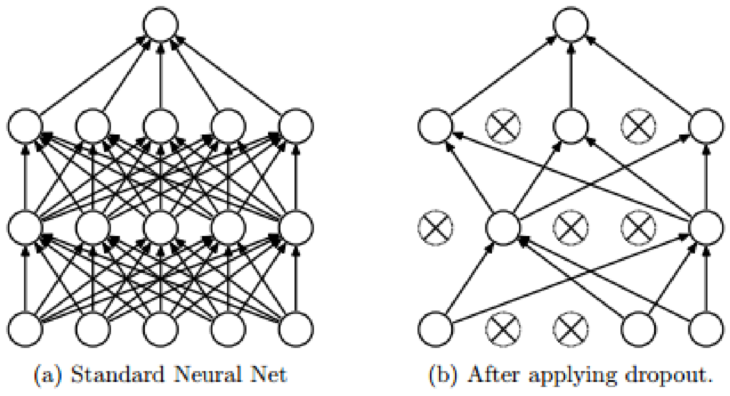

Figure 2.8: Representation of dropout taken from [20]

Chapter 3

Net Structure, Case Study and Dataset

3.1 Convolutional Net Characteristics

Finding the structures that performed the best for the current case study was one of the critical parts in this study. Surprisingly, it is very difficult to find literature about the topic of selecting the best net structures for each case. In most of the cases researches either use well known structures with proved results or conducted a variety of tests with multiple structures [21] in order to find the most efficient one. Since the used dataset was completely developed for this thesis it was not possible to use a well known structure. In this thesis two different structures were taken in consideration: SmallNet and BigNet. As the names suggests the big difference between these two structures is their size. The SmallNet has only one convolution layer while the BigNet follows some characteristics used in the paper of Alex Krizhevsky et al.[12] having five convolution layers. The main purpose on using these two different structures was to find how the number of layers could interfere with the final results

3.1.1 SmallNet

As it is represented in the fig 3.1 the SmallNet convolution section is constituted by one convolution layer with only one 3x3 kernel, followed by a non-linearity layer (relu), a maxpooling unit with a reducing scale of (2,2) and finally a dropout unit.

The fully-connected section is constituted by one fully-connected layer that receives 4536 inputs and has 100 outputs, followed by a non-linearity layer (relu), another fully-connected layers that is the output layer receiving 100 inputs and having $nclasses$ outputs. Where $nclasses$ depends on the used dataset configuration.

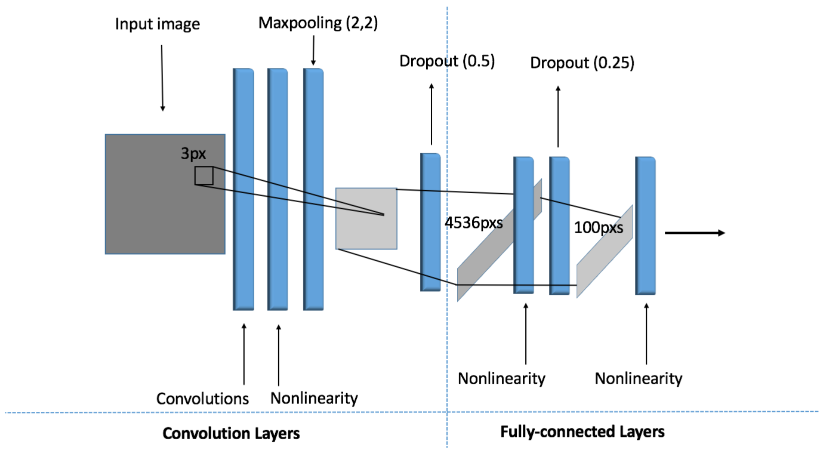

Figure 3.1: Smallnet Structure

3.1.2 BigNet

As it is represented in the fig 3.2 the BigNet convolution section is constituted by 5 convolution layers with 1,3,6,9 and 12 kernels where the first kernel is a 9x9 the second a 6x6 and the rest are 3x3. Each convolution layer is followed by the same layers as the SmallNet (non-linearity and maxpooling layers). The fully-connected structure is also similar to the SmallNet one having two dense layers where the first converts 36 inputs in 100 outputs and the second 100 inputs in $nclasses$ outputs.

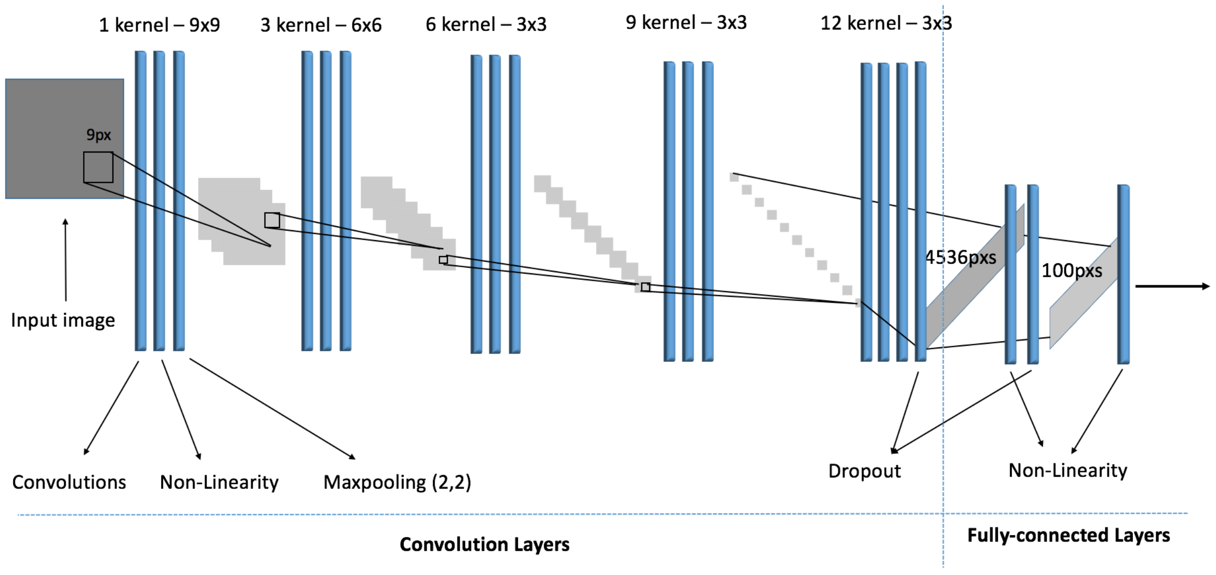

Figure 3.2: Bignet Structure

3.2 Case Study

The purpose of the case study was in one hand to be simple enough so that it could be easily implemented and analysed and in the other hand it needs to be robust enough so that the results could be compared with a real live scenario without being easily solved by applying the classic image processing techniques. In the @Home competition the robot must be able to handle the visitors so it must make a distinction between those visitors who are allowed to enter the house (for example: the plumber, the medic, the milk man ...) and the ones that are not. However even if the robot is capable to recognise the allowed guest with 100% of certainty, after they are inside the house the robot assumes that they are trustworthy which is an huge vulnerability since it allows the guests to perform any action without controlling them. This is exactly the problem that this thesis proposes to solve.

Although the visitors are allowed to enter, the robot must at all times be aware of their actions in order to determine if they are acting normally or if they are having abnormal behaviors that can arm Granny Annie or her belongings.

The scenario of this case study occurs in the apartment kitchen and the visitor is a plumber that comes to fix the kitchen sink. While in the kitchen the subject has a lot of different actions from which some are considered normal and others abnormal. The kitchen scenario is not static and changes through the videos in order to avoid the creation of patterns that associate actions to the position of the object.

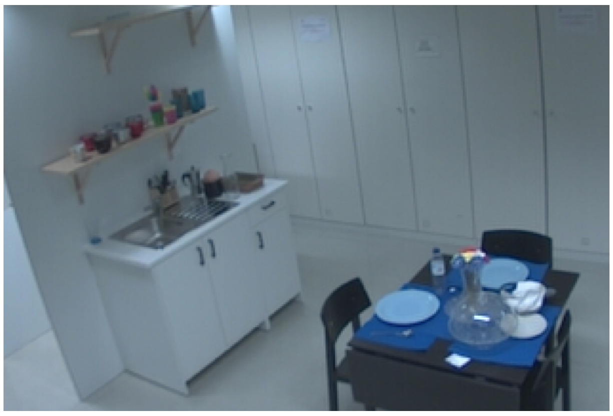

Figure 3.3: Kitchen scenario with table with plates and silverware

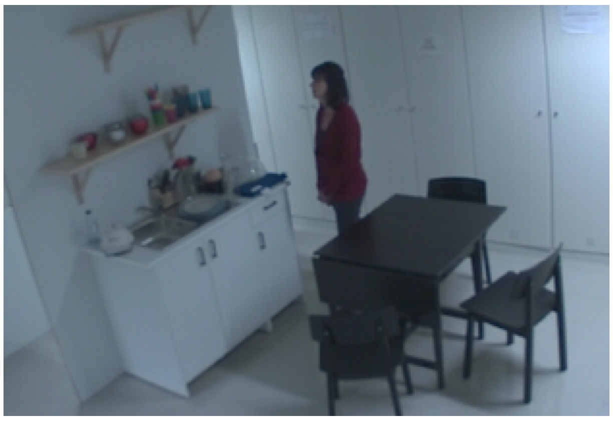

Figure 3.4: Kitchen scenario with table without anything on

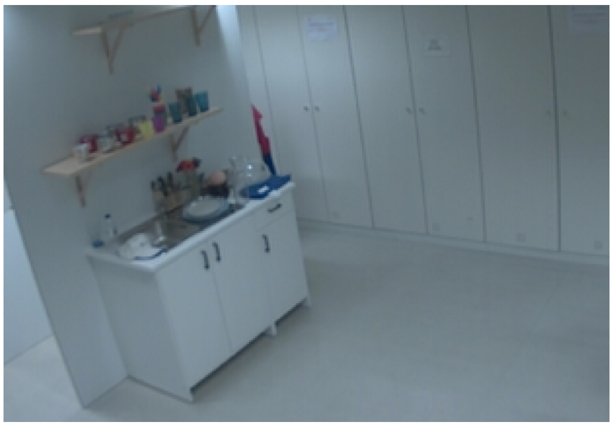

Figure 3.5: Kitchen scenario without the table

3.3 Creating the Dataset

Obtaining the dataset was one of the most important parts of this thesis since that having a good set of data is crucial for the training and testing process of CNNs. A lot of the most used datasets contain a huge amount of data and are collected by big research teams and used by many researches. A few examples of the most fascinating collections of data are: ”ImageNet” [22], containing more than 15 million images, separated in 22 thousand categories (very well explained by Dr. Fei Fei Li in the Ted Talk ”How We Teach Computers to Understand Pictures” [23]), ”Caltech-101” [24], containing 101 classes of 300x200pxs images and ”CIFAR” 10 and 100 [25] with 60 thousand images of 32x32 pxs. The datasets referenced above are all constituted by single frames, however there are also some videos databases, such as: ”CAVIAR” [26], that include 384x288pxs images recorded at 25 frames per second and includes different behaviors like walking, leaving bags unattended, people in groups, etc ... or ”LIRIS” [27] that was recorded using 2 different cameras (a kinect and a Sony consumer camcorder) and contains 10 different classes spread in 828 actions. Although very complete none of this examples could be used in this thesis because of the need of applying the model to a specific context. In the sections bellow is described the process of obtaining the dataset.

3.3.1 Actions Description

The first step for building the dataset was to record a big variety of actions performed in the kitchen of the ISR test bed scenario (see figures 3.3, 3.4, 3.5). Generally all the recorded frames needed to have a big variety of actions and features in order to avoid over-fitting (model become stuck in a local minimum). In order to guarantee the variety of the dataset multiple actions were recorded with 4 different subjects (three male and one female), where the same subject would use different clothes and the disposal of the scenario would change from video to video. The result was a total of 471 actions where each action was constituted by 30 frames summing up to a total of 14130 frames. In order to make the dataset flexible enough so it could be split into various classes each action was classified with a specific label such as: ’sit’, ’sit on the floor’,’sit on the table’,’sit on the sink’, ’walk’,’run’,’steal’, ’throw a chair’,’throw object’,’throw’,’fight’,’move arms’,’jump’,’shake’, ’fix’, ’drink’ and ’stop’. It’s important to notice that each of these labels contain multiple videos where these actions were performed multiple times by the subjects. Bellow there are some examples of the collected frames.

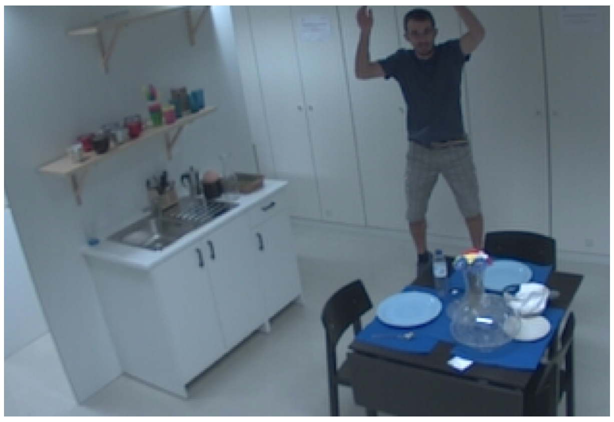
Figure 3.6: Male subject 1 performing the action of jumping

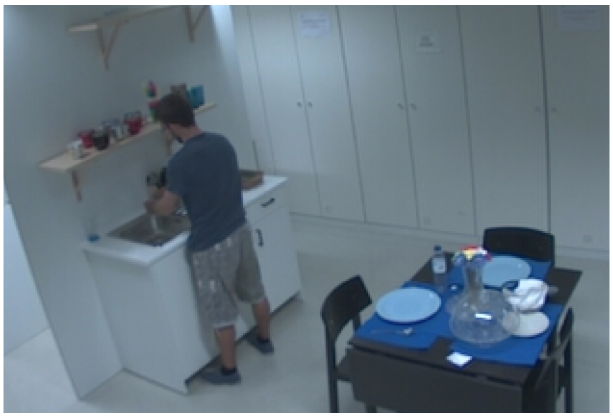
Figure 3.7: Male subject 1 performing the action of fixing the sink tap

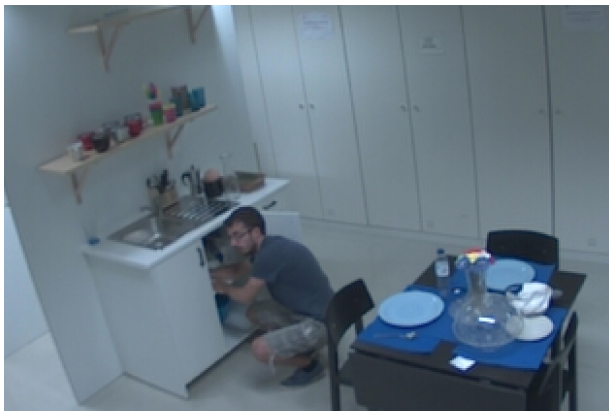
Figure 3.8: Male subject 1 performing the action of fixing under the sink

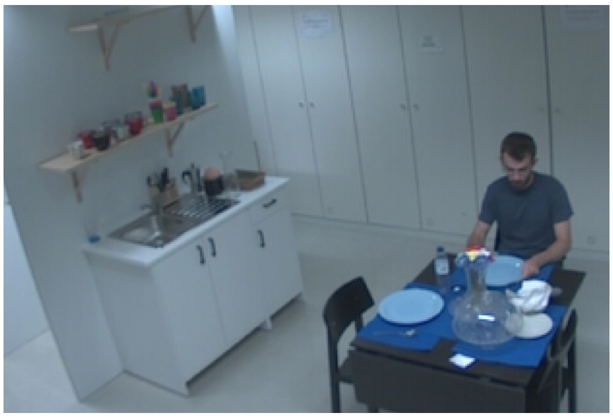
Figure 3.9: Male subject 1 sited on the chair

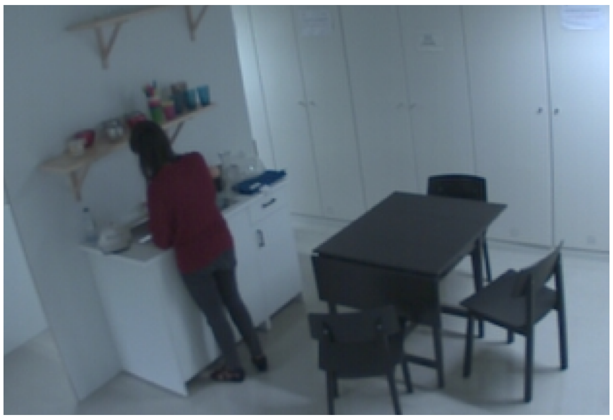

Figure 3.10: Female subject fixing the sink tap

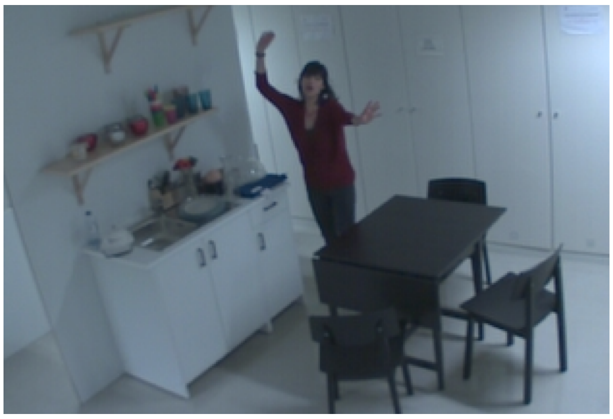

Figure 3.11: Female subject moving her arms

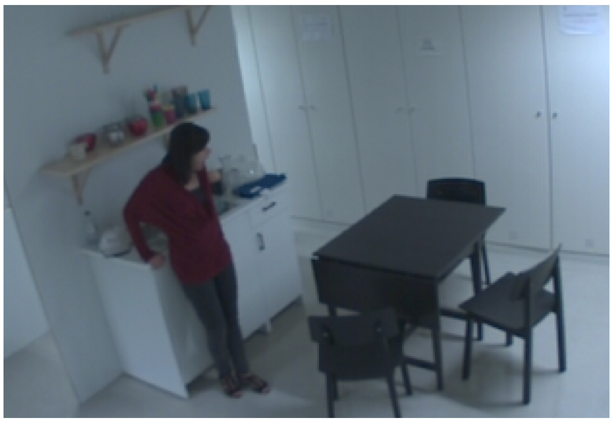

Figure 3.12: Female subject standing still near the sink

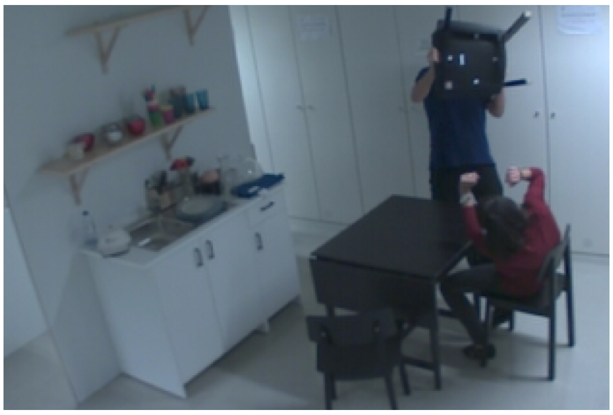
Figure 3.13: Male subject 2 fighting with female subject

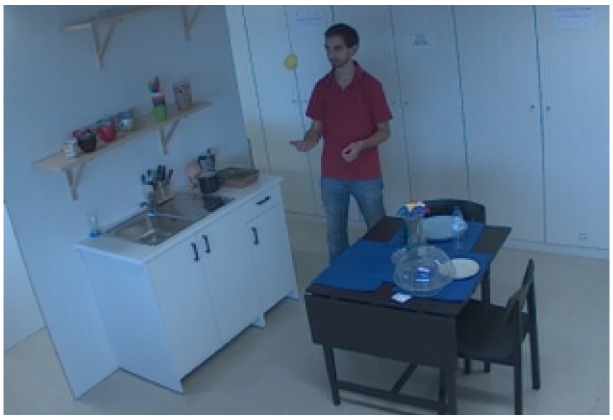
Figure 3.14: Male subject 3 performing the action of throwing glass

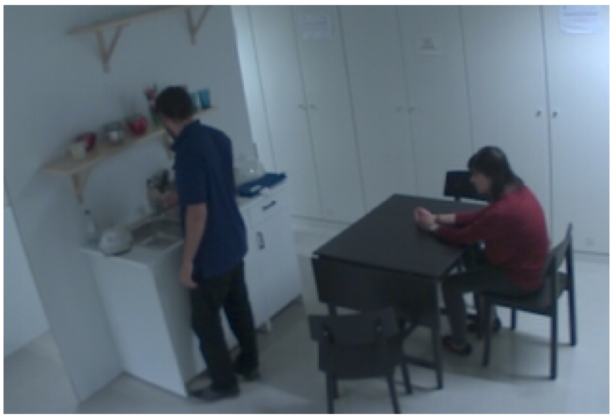
Figure 3.15: Male subject 2 fixing the sink tap while female subject is sited

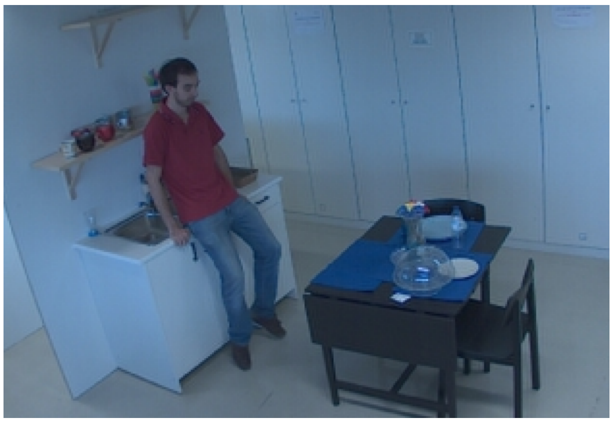

Figure 3.16: Male subject 3 sited on the sink

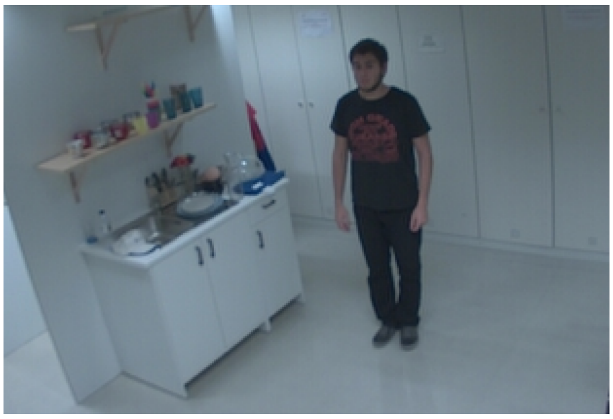

Figure 3.17: Male subject 2 standing still in scenario without table

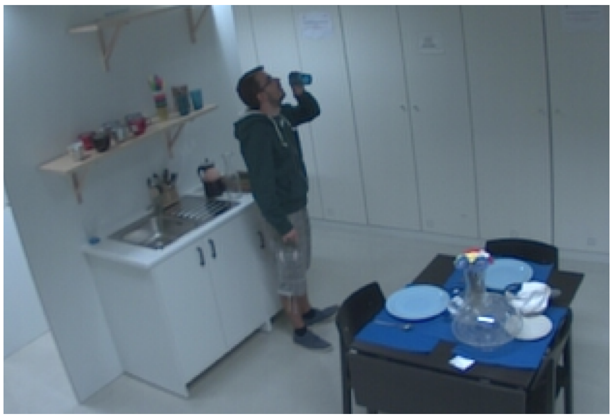

Figure 3.18: Male subject 1 performing action of drinking

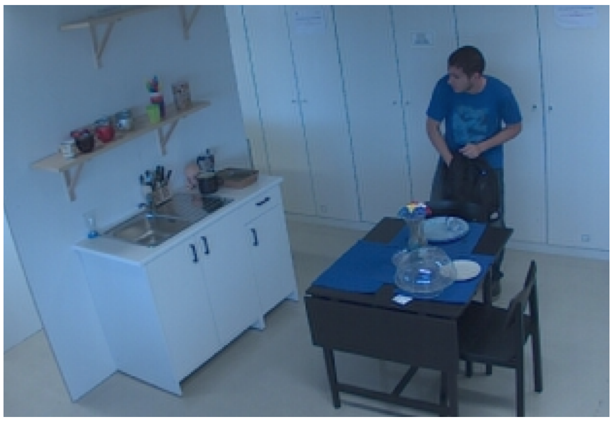

Figure 3.19: Male subject 2 performing action of stealing

Of course that the fact of giving to each action such a specific label allowed the clustering of labels into different classes. In this thesis 3 different datasets were created by clustering the labels in different global classes.

Two classes dataset

This dataset was constituted only by to classes: normal and abnormal.

- normal = 'sit', 'sit on the floor', 'walk','run','fix', 'drink' and 'stop'
- abnormal = 'sit on the table', 'sit on the sink', 'run', 'steal', 'throw a chair', 'throw object', 'throw', 'fight', 'move', 'arms', 'jump', 'shake'.

Three classes dataset

- fast movements = 'shake','throw the chair','throw object', 'run' 'throw','fight', 'move the arms', 'jump'
- slow movements = 'walk', 'fix','steal', 'drink'
- stop movements = 'stop','sit', 'sit on the floor','sit on the table','sit on the sink'

Four classes dataset

- seated = 'sit', 'sit on the floor','sit on the table','sit on the sink'
- moving = 'walk','run','steal'
- fighting = 'throw chair','throw object','throw','fight','move arms','jump','shake'
- still = 'fix', 'drink', 'stop'

3.3.2 Frames Characteristics

The camera used to record the dataset was an ip RGB axis camera with a static position fixed to the ceiling. The camera was recording with a resolution of 320x240 pixels, however the frames used in the dataset were resized to 160x120 pixels because it did not require so much RAM from the computer which allowed the use of a bigger net. The ip axis camera has an off-the-shelf driver that allows an integration with ROS [28] (Robot Operative System). The topic generated by the camera allows to access to live images which is perfect for the integration with the @Home challenge.

3.3.3 Recording the dataset

This thesis required a lot of software building in order to fulfill all the requirements. All the software was written in python and is available for consultation in bitbucket cloud[29]. Above is described how the recording of the dataset was implemented in the code. Figure 3.20 is a diagram that illustrates the flow of the capture of the video.

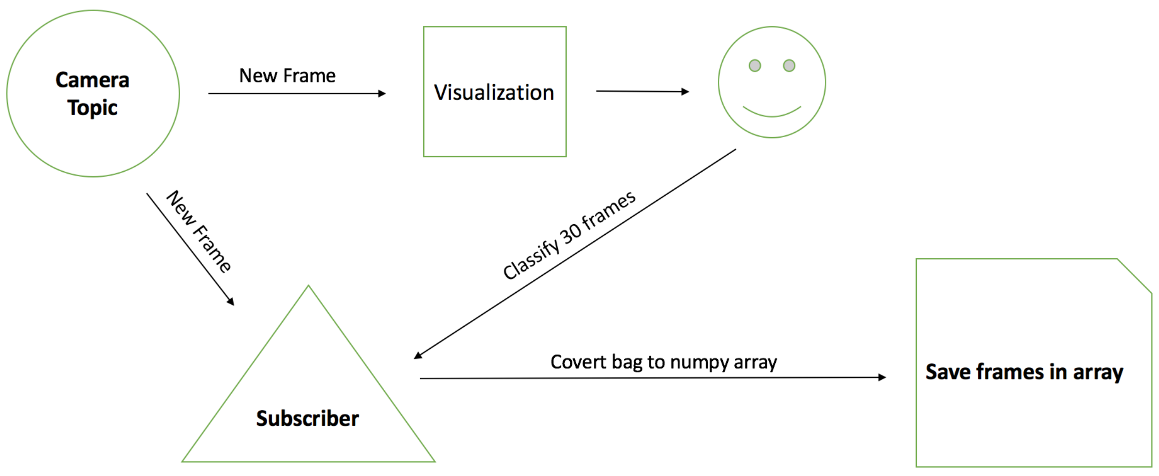

Figure 3.20: Recording Dataset

The camera publishes the information in a topic that is subscribed by two modules. The first is the viewer module that is an open source software (*image_view*) and displays the video. The second is a subscriber developed by implementing the library ”rospy” (implements ROS for python language). This module is always saving the last 30 frames. The user that its visualizing the video through the visualization module classifies each group of 30 frames with a label. Before saving the labeled frames the subscriber module also rescales each individual image and removes the frames headers that per default contain the time from when the image was taken.

To facilitate the classification process the videos were recorded in multiple .bag files[1] with a rate of 20fps. However, the processing module does not save all the consecutive frames, always leaving

[1].bag correspond to rosbag files which are built in files from ROS that record all the information that was published to a certain topic during the live occurrence of the system

an interval of 10 frames between them. This interval has two important roles: one is to make sure that the user is actually classifying the images that he is seeing at the same moment, because the processing model is slower than the visualization model (which means that without the interval the user would classify outdated images); the other role is to prevent the recording of redundant images since consecutive images are very similar.

3.4 Training Process

3.4.1 Splitting the dataset

In the literature the dataset is normally divided into three types: training set (a subset that is used for training the model and that is propagated through the net), the validation set (that is also used during the training process but to implement the criteria of early stopping, generating an more independent error) and finally the test set (used after the training to measure the model error against data never used in the training process)

Since the dataset used in this thesis was not big enough the validation set was not used. Instead, it was only used the training set and the test set (the test set had both role of test set and validation set). The method used to generate both sets from the dataset is as follows:

1. Queue all frames (grouped by class) in an array
2. Start iterating through the actions, from left to right, splitting the dataset so that 1/3 was test set and the rest 2/3 training set. Which means that 1 for every 3 actions would be test set and the others training set.

It is important to notice that each action/video is constituted by 30 frames. The result is two datasets that even though both having videos with the same label each of them have different frames.

3.4.2 Avoiding Overfitting

As the dataset used is small it was really important to avoid overfitting at all costs because if not the model could only be applied to the current training set. The first technique used was dropout layers (as described before). The technique of early stopping was also used. To implement early stopping, for each iteration we should measure both the training error and the validation error. As long as the validation error is getting lower the train might proceed. However the train must stop when the validation error starts to increase, even if the training error is still reducing (as pointed out on figure 3.21).

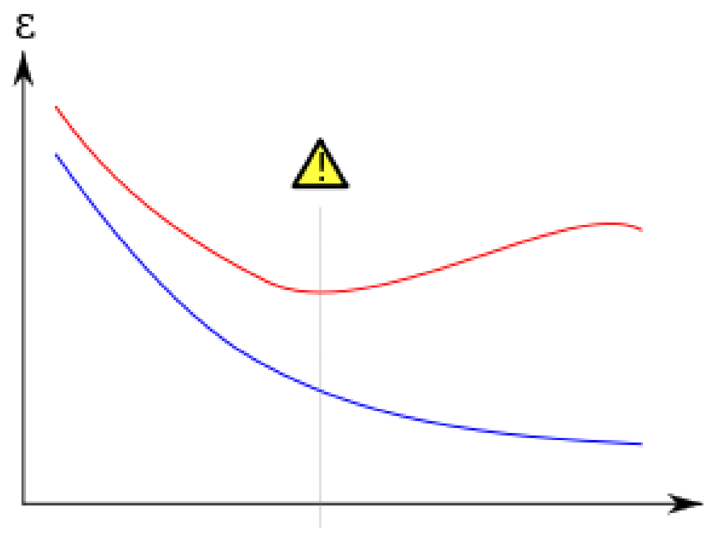

Figure 3.21: This figure shows the evolution of the training error (blue) and test error (red) through each training iteration. The yellow triangle shows where the trainning should stop

Chapter 4

Results

In this section the results for the three analysed datasets will be presented for the two used net structures. In this thesis it was used the library Keras[30] to implement the training and prediction using CNNs.

4.1 Obtaining Error Rate

To better understand the results bellow it is important to describe how they were obtained. Even though the inputs of the nets are single frames and the output does not depend on past frames, this case study involves video processing and consequently its classification should depend on the classification of all the frames in it. To accomplished that the frames belonging to a certain video were fed as input to the model in the same order as they appear, the global classification of the video was the average of all the belonging frames classification. As described in section 3.3.3, all the frames of the dataset are divided in sets of 30. This means that the frames from 1 to 30 belong to the same action and then frames from 31 to 60 belong to a different action.

Taking that in consideration the process to obtain the error rate is described bellow:

```
Result: Error Rate
while nFrame less then size(dataset) do
    if nFrame is multiple of 30 then
        top1 = average(30 last most probable classifications);
        top2 = average(30 last second most probable classifications);
        if top1 != ground truth then
            error1 ++;
            if top2 != ground truth then
                error2 ++;
            end
        end
    else
        nFrame ++;
    end
end
Error1 = error1/Total number of frames * 100;
Error2 = error2/Total number of frames * 100;
```

4.2 Two Classes Dataset

This dataset was constituted by only two classes: Normal and Abnormal. And it was the most simple case used in this study. The software developed in this thesis allowed, in an automated way, to test different variations in the training parameters in order to see which combinations generated the best results. The parameters tested in the train were the learning rate (using values of 0.01, 0.05 and 0.2) and the batch size (using values of 30, 100 and 500). In this study was also tested how different weights initializations could influence the results. Two different initializations were tested:

- The "lecun uniform"[31] initialization. This algorithm of initialization states that the weights should not be to large so that the sigmoid would saturate resulting in small gradients and consequently in a slow learning process. Neither to small causing the same effect. The weights should be intermediate so that the sigmoid is primarily active in its linear region.
- Pre-trained initialization. This type of initialization states that applying weights obtained in the training of other models, will lead to better and faster results.

In the graphs above is shown the error percentage for the multiple training variations performed in this thesis. There are four different error types:

- Ts n p - Testset without pre-training
- Ts p - Testset with pre-training

- Trs n p - Training set without pre-training
- Trs p - Training set with pre-training

4.2.1 SmallNet Results

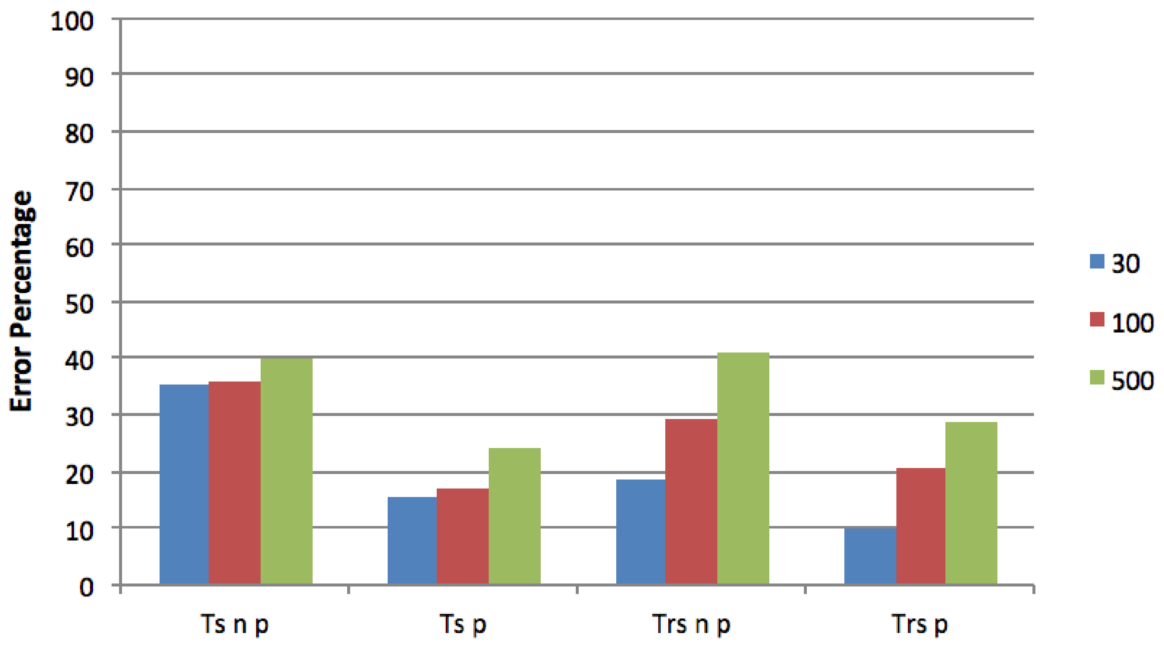

Figure 4.1: SmallNet with learning rate of 0.01 for the different batch sizes

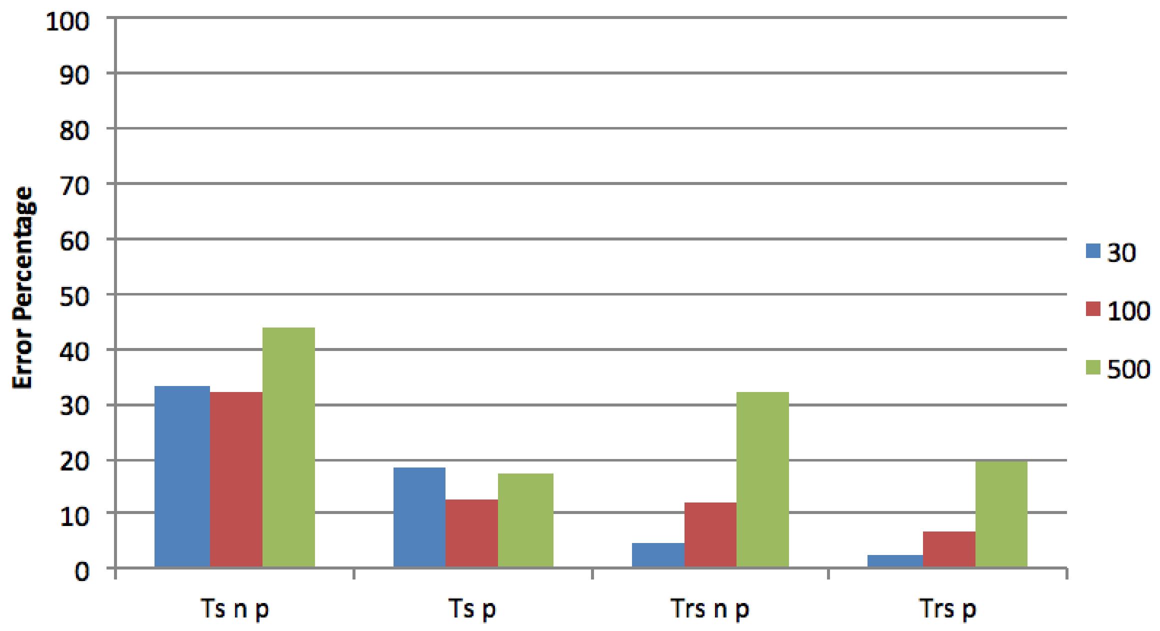

Figure 4.2: SmallNet with learning rate of 0.05 for the different batch sizes

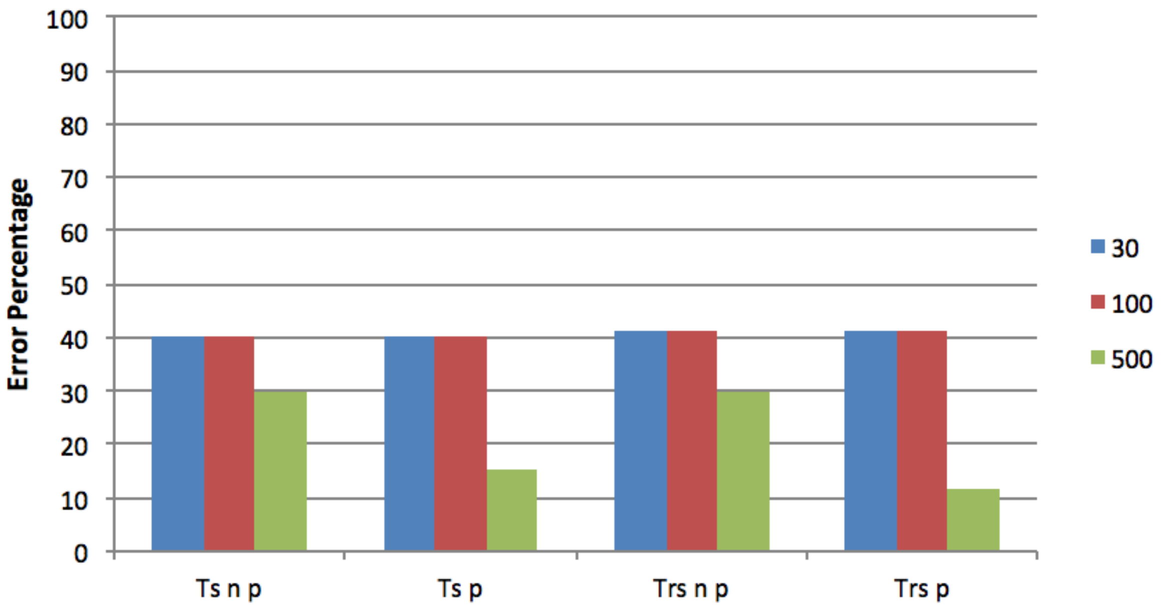

Figure 4.3: SmallNet with learning rate of 0.2 for the different batch sizes

For the SmallNet the pre-trained was obtain by training the same net in a smaller dataset (less frames) and then applying the resulting weights as initialization of the train with the entire dataset. This process helps the net to converge faster although with more epochs the training with no initialization would probably reach the same results.

4.2.2 BigNet Results

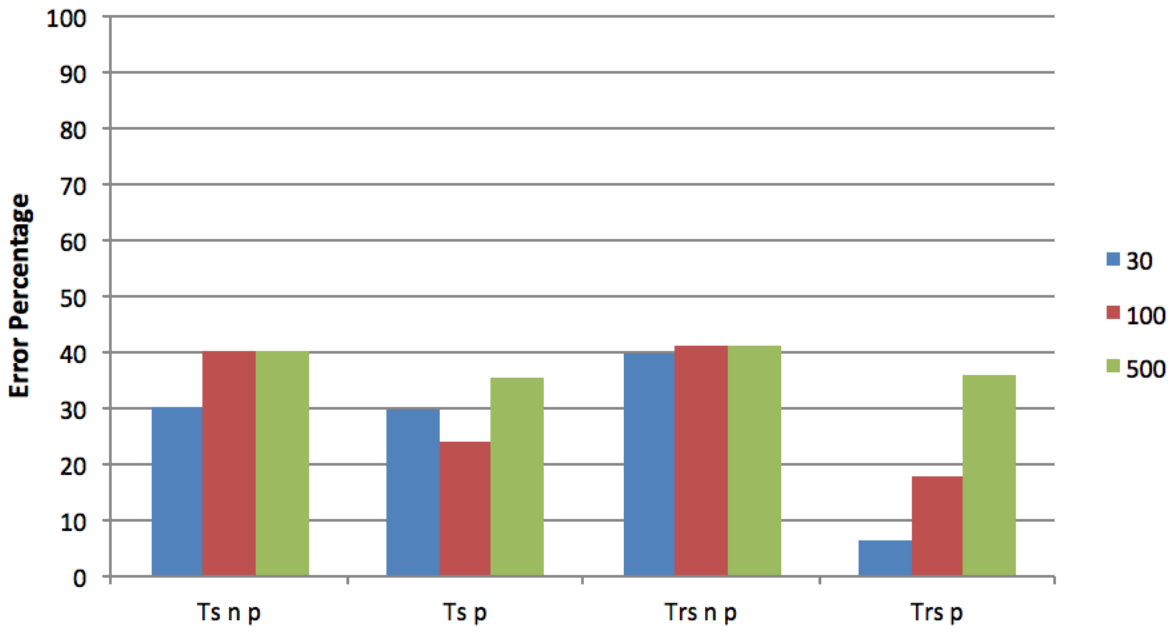

Figure 4.4: BigNet with learning rate of 0.01 for the different batch sizes

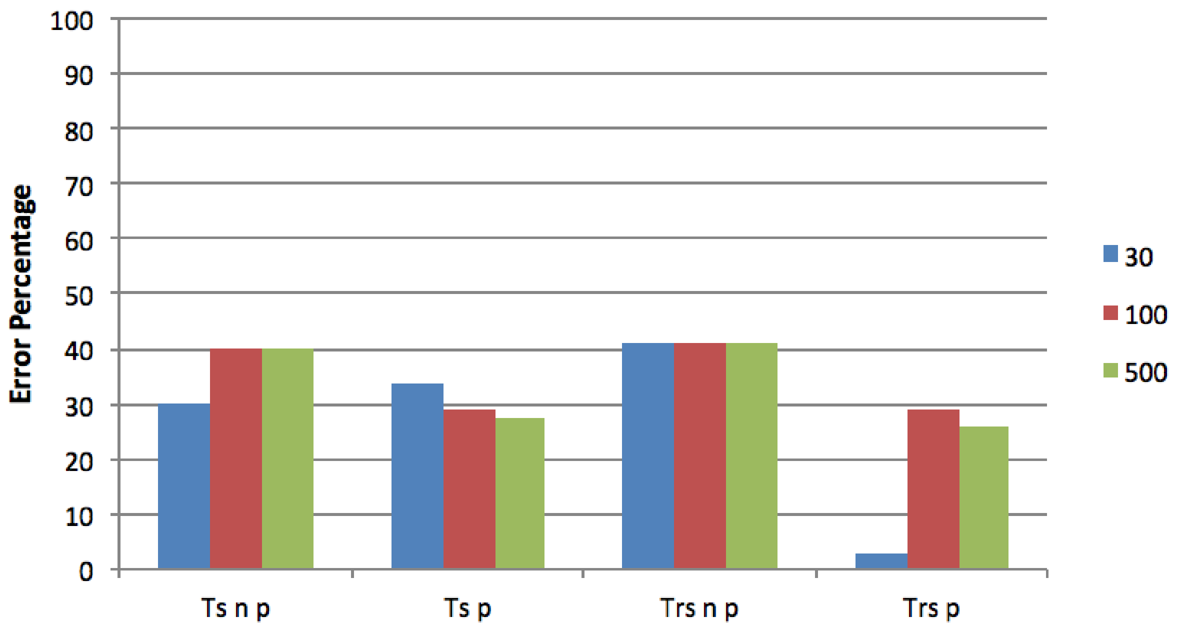

Figure 4.5: BigNet with learning rate of 0.05 for the different batch sizes

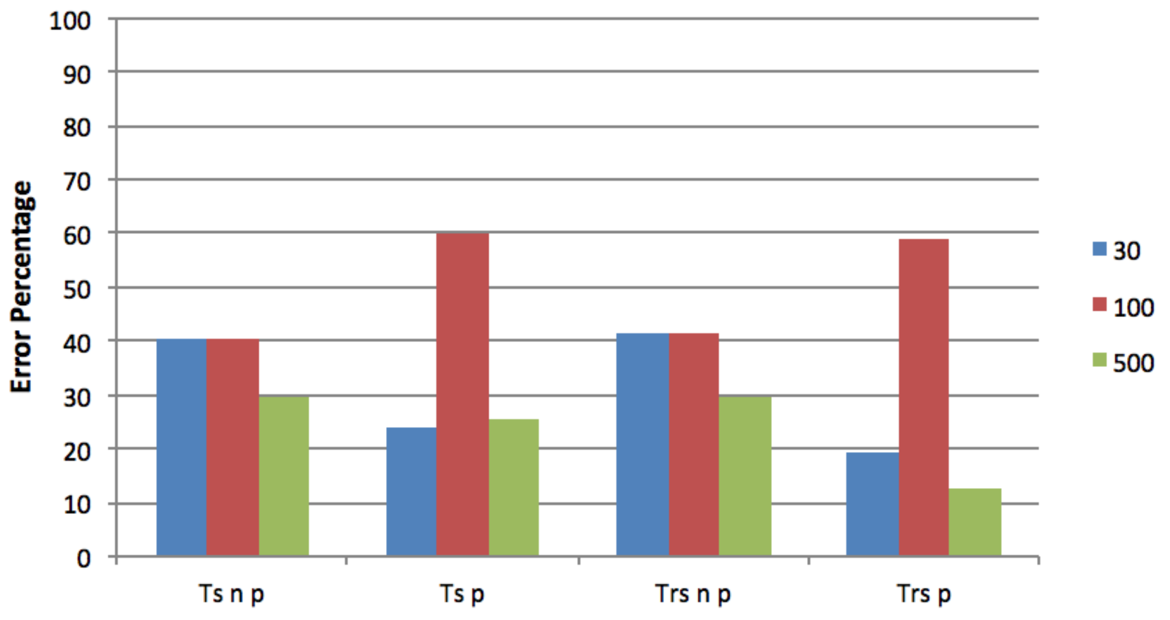

Figure 4.6: BigNet with learning rate of 0.2 for the different batch sizes

BigNet was trained layer by layer. Which means that firstly all convolution layers but the first one were removed and trained for 10 epochs, then the second one was added and the first layer was initialized with the just learned weights. The process is repeated for the rest of the layers. As can be seen in the graphics above, without weights initialization the training has poor results and never converges however with the weights initialization the error rate decreases significantly (see figures 4.4, 4.5 and 4.6).

4.2.3 Results Analysis

SmallNet	30	100	500	**BigNet**	30	100	500
0.01	35,211%	35,915%	40,140%	0.01	30,281%	40,140%	40,140%
0.05	33,098%	32,394%	43,661%	0.05	30,281%	40,140%	40,140%
0.2	40,140%	40,140%	29,577%	0.2	40,140%	40,140%	29,577%

Table 4.1: Summarized results of the error rate of the test set without pre-training

SmallNet	30	100	500	**BigNet**	30	100	500
0.01	18,367%	29,387%	41,224%	0.01	39,591%	41,224%	41,224%
0.05	4,897%	12,2446%	32,244%	0.05	41,224%	41,224%	41,224%
0.2	41,224%	41,224%	11,836%	0.2	41,224%	41,224%	29,795%

Table 4.2: Summarized results of the error rate of the train set without pre-training

SmallNet	30	100	500	**BigNet**	30	100	500
0.01	15,492%	16,901%	23,943%	0.01	29,577%	23,943%	35,211%
0.05	18,309%	12,676%	17,605%	0.05	33,802%	28,873%	27,464%
0.2	40,140%	40,140%	15,492%	0.2	23,943%	59,859%	25,352%

Table 4.3: Summarized results of the error rate of the test set with pre-training

SmallNet	30	100	500	**BigNet**	30	100	500
0.01	9,795%	20,408%	28,571%	0.01	6,530%	17,959%	35,918%
0.05	2,448%	6,938%	19,591%	0.05	2,857%	28,873%	25,714%
0.2	41,224%	41,224%	11,836%	0.2	19,183%	58,775%	12,653%

Table 4.4: Summarized results of the error rate of the train set with pre-training

From both net results it can be concluded that in order to increase the learning rate the batch size has also to be increased. This can be observed by comparing figure 4.1 with figure 4.3 (for the SmallNet) and figure 4.4 with figure 4.6 (for the BigNet). In both figures 4.1 and 4.4 the results are better with the lower batch size, however in figures 4.3 and 4.6 the better results are obtain with the higher batch size. This makes sense because having a higher batch size means that the algorithm uses a bigger variety of inputs in each update so the learning rate can be reduced because each update is more moderate.

It can also be observed in figures 4.3, 4.4, 4.5 and 4.6 that in some cases different batch sizes have the exact same error rate. This is due to both combinations had encountered the same local minimum, not being able to reduce the error any more after that. Finally in figure 4.6 both error rates from training set and test set with pre-training with the batch size of 100 standout for being much higher than the others. This can also be explained by the encounter of a local minimum during the training process that prevented the model from learning the dataset.

4.3 Three Classes Dataset

In order to raise the complexity of the tests other datasets were created which included more classes (as mentioned in section 3.3.1). Here are described the results for the dataset with three classes: ”stop movements”, ”slow movements”, ”fast movements”. Eventhought for the Two Classes Dataset the best results were obtain with a combination of learning rate equal to 0.05 and batch size equal to 100, after several tests the best combination for this case was obtain with a lower batch size and learning rate. The values used were 0.01 for the learning rate and 30 for the batch size. Also in this situation using the same methods of pre-initialization used for the two classes dataset proofed to be inefficient leading to poor results. The best results were obtain by using the *lecun uniform* initialization.

4.3.1 SmallNet Results

For the SmallNet were obtain for the training error: 2.5477% for top-1 error and 0.955% for top-2 error. Regarding the test error were: 26.7515% for top-1 error and 7.6433% for top-2 error.

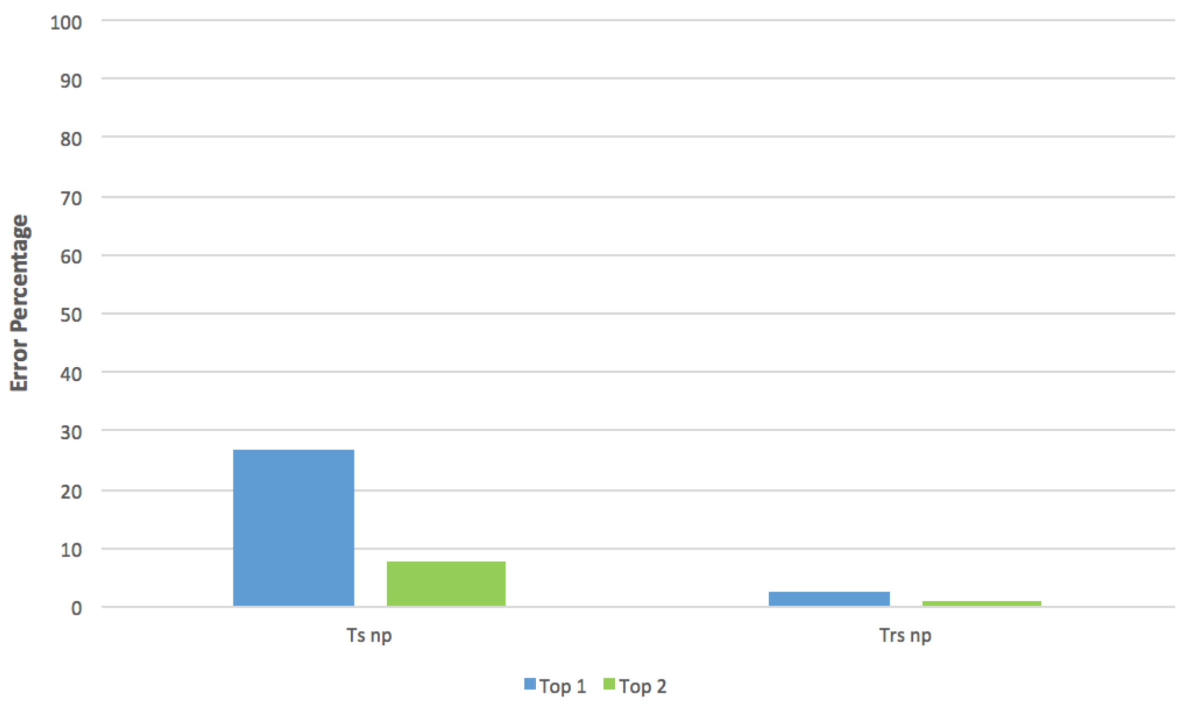

Figure 4.7: SmallNet error percentage for dataset with 3 classes

This model was trained using 400 epochs using the early-stop criteria to determine when to stop.

4.3.2 BigNet Results

This model was trained using 600 epochs using the early-stop criteria to determine when to stop. The error rate obtain for the test set was, in the top-1 33,12% and for the top-2 was 8,91%. For the training set the obtain error was 12.738% for the top-1 and 3.18% for the top-2.

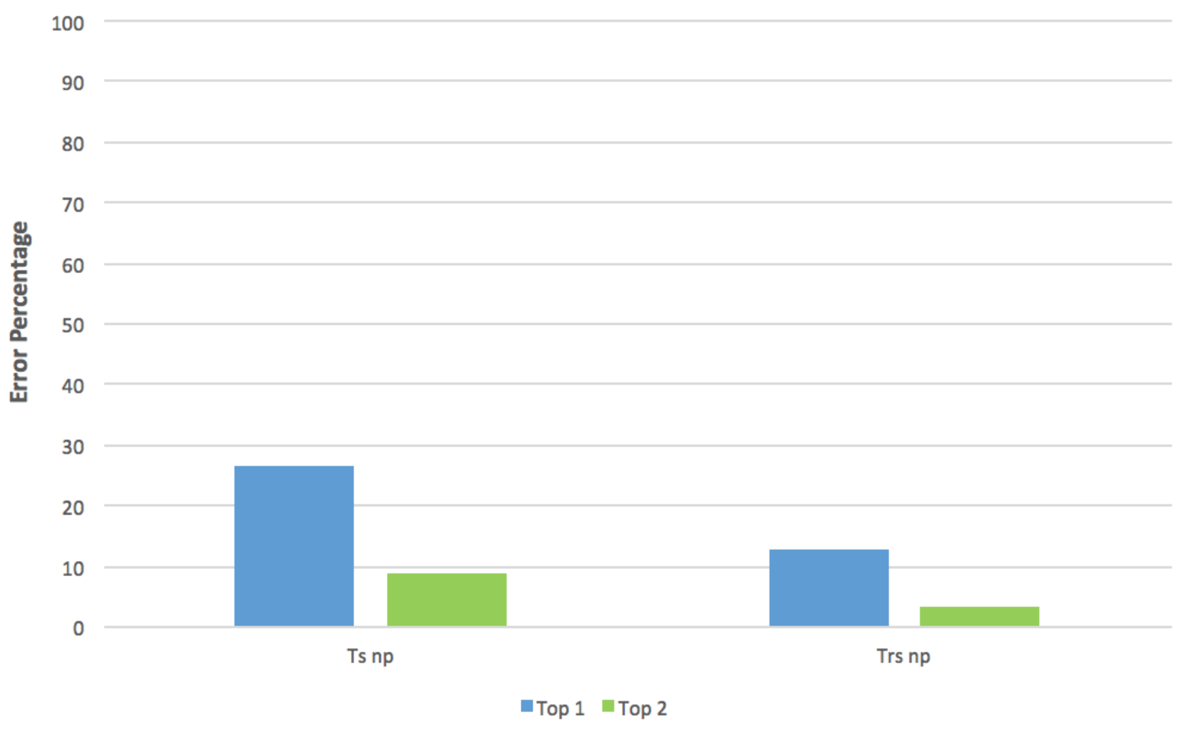

Figure 4.8: BigNet error percentage for dataset with 3 classes

4.3.3 Results Analysis

SmallNet	Top 1	Top 2	**BigNet**	Top 1	Top 2
test set	26,7515%	7,6433	test set	33%	8,91%
training set	2,5477%	0,9555%	training set	12,738%	3,18%

Table 4.5: Summarized results of *Three Classes Dataset*

Even though the training error increased comparing to the two classes dataset the results are still very positive. It is also very interesting to see how the results of the SmallNet are as good or even better than the results of the Bignet meaning that this case study does not requires a lot of neurons to create a model that represents this dataset. Although the ”lecun uniform” initialization tries to find intermediate weights the algorithm is still random so in some situations the training process would be stuck in a local minimum. In that case the only solution was to restart the training in order to obtain a better initialization.

4.4 Four Classes Dataset

This last dataset was also the most complex of this thesis. It was divided into 4 different classes: ”still”, ”sited”, ”moving” and ”fighting”. The training process was identical to the one used for the ”Three Classes Dataset”.

4.4.1 SmallNet Results

The results of the SmallNet were very positive. The error rate for the test set was 21,65% for the top-1 and 7% for the top-2. Regarding the training set the error rate was 0,636% for the top-1 and 0,32% for the top-2.

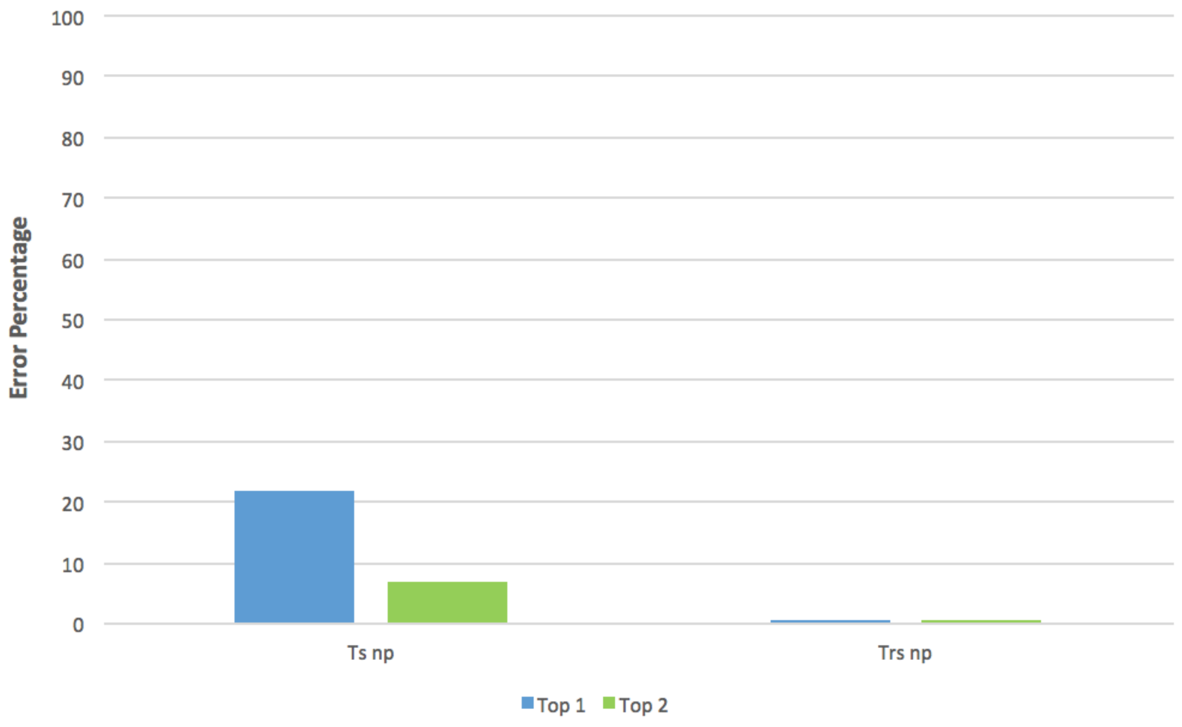

Figure 4.9: SmallNet error percentage for dataset with 4 classes

4.4.2 BigNet Results

Although the results were a little worse than the SmallNet results they are still very positive. The error rate for the test set was 31,21% for the top-1 and 15,92% for the top-2. Regarding the training set the error rate was 20,06% for the top-1 and 7% for the top-2.

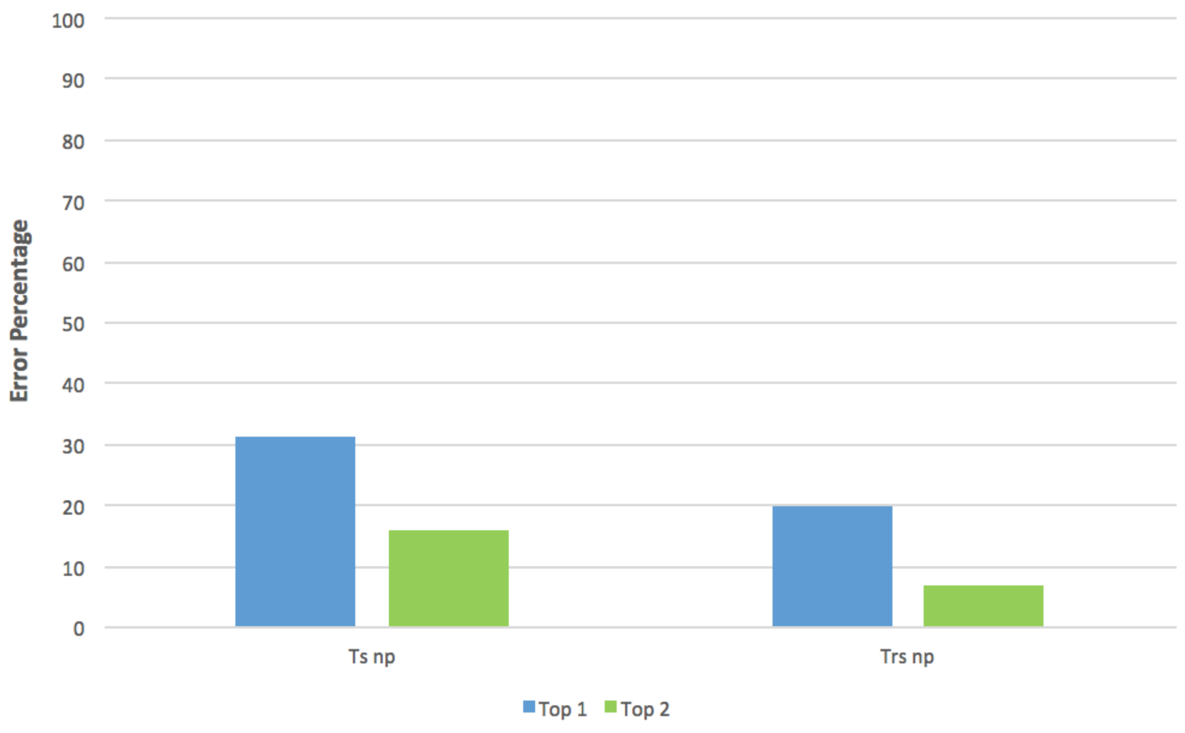

Figure 4.10: bigNet error percentage for dataset with 4 classes

4.4.3 Results Analysis

SmallNet	Top 1	Top 2	**BigNet**	Top 1	Top 2
test set	21,65%	7%	test set	31,21%	15,92%
training set	0,636%	0,32%	training set	20,06%	7%

Table 4.6: Summarized results of *Four Classes Dataset*

The results of this dataset turned out to be very positive even with a small dataset in comparison with the state of the art ones. Once again the results from the SmallNet are better than the results from the BigNet.

4.5 Global Results Analyses

In this thesis the same dataset was divided into different classes, starting with only two classes until four different ones. In the ”Two classes dataset” results a big variety of combinations were tested and the conclusion was that having a small learning (0.01) rate together with an also small batch size (30) turn out to be the combination that led to better results. Choosing a batch size of 30 turned out to be very efficient mostly because each action of the dataset was constituted by 30 frames. Another interesting discover was that the SmallNet structure achieved better results than the BigNet which goes against the majority of the literature where the bigger the net the better the results. This results are also very positive because the training and predicting process of the SmallNet is much faster than the BigNet. Of course that the size of the net depends on the complexity of the case study and probably bigger datasets with more classes would require bigger nets.

Chapter 5

Conclusion

5.1 Thesis Overview

In this thesis convolution neural networks were used to learn to correctly distinguish different behaviors in a real life situation. The case study is a real life problem adapted from the competition RockIn@Home. Before this thesis this problem has never been tackled so all the work done started from ground zero and was constructed along the way. In this thesis is presented how the case study dataset was constructed and how the methods of CNNs were applied in order to obtain our results.

5.2 Thesis Conclusions

In this thesis the same dataset was used in three different situations by changing how the data was split. The model was tested against a dataset divided into two, three and four classes. The results obtain for each case were positive and allowed taking the following conclusions: Firstly it is definitely possible to apply CNNs into the current case study. The model will autonomously find a way to classify the dataset with less error possible. However there are no thumb rules that determine the structure of the net that should be used to each case study so in order to find the best model is required to make a several tests with different net structures and parameters. Secondly not always a deeper net obtains better results. For this thesis case study the SmallNet structure obtain better results in less time than the bigger net, bigger nets should only be used in more complex cases where the number of outputs are bigger or for a situation that requires to give more relevance to small details. Thirdly keeping both training error and test error in mind while training the dataset turned to be very useful to find the best models. Using techniques like early-stopping or cross-validation is very important to obtain the best models. Fourthly in video processing it is very important to obtain a big and miscellaneous dataset. By doing so it will avoid the appearance of patterns that lead to over-fit and prevent the model from obtaining good results in real life situations. As a last conclusion the results of this thesis show once more the potential of autonomously learning methods to solve complex problems where human intuition is involved. There is a lot of good documentation and machine learning libraries that allow researches to quickly start testing their models and obtain good results.

5.3 Future Work

There are multiple ways to expand this work. In this thesis the dataset was only split into training set and test set. Using a validation set in the training process could improve the results, also the method of cross-validation could also be applied. In order to make sure that the model would obtain good classifications in real live scenarios it is important to keep improving the dataset by collecting more videos with more variety and to test the model in online mode (which means measuring the model error by making live classifications). In the future the model should be capable of doing ”online learning”. This means that it should be continuously learning and collecting data and could be corrected by an Human every time that it made poor classifications. Finally, as in video processing, the past frames are important for the present classifications. Using recurrent neural networks could also improve the results and allow more complex associations between frames.

Bibliography

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219. doi: 10.1109/5.726791.

[2] Rockin. URL `http://rockinrobotchallenge.eu`.

[3] B. Hu, Z. Lu, H. Li, and Q. Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2042–2050, 2014. URL `http://papers.nips.cc/paper/5550-convolutional-neural-network-architectures-for-matching-natural-language-sentences`.

[4] Kernels along the layers. URL `http://stats.stackexchange.com/questions/146413/why-convolutional-neural-networks-belong-to-deep-learning`.

[5] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006. ISSN 0899-7667. doi: 10.1162/neco.2006.18.7.1527. URL `http://dx.doi.org/10.1162/neco.2006.18.7.1527`.

[6] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press, 2007. URL `http://papers.nips.cc/paper/3048-greedy-layer-wise-training-of-deep-networks.pdf`.

[7] J. Schmidhuber. Multi-column deep neural networks for image classification. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 3642–3649, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-1-4673-1226-4. URL `http://dl.acm.org/citation.cfm?id=2354409.2354694`.

[8] F. Seide, G. Li, and D. Yu. Conversational speech transcription using context-dependent deep neural networks. In *Interspeech 2011*. International Speech Communication Association, August 2011. URL `http://research.microsoft.com/apps/pubs/default.aspx?id=153169`.

[9] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

[10] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. URL `http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94`.

[11] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 886–893, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2372-2. doi: 10.1109/CVPR.2005.177. URL `http://dx.doi.org/10.1109/CVPR.2005.177`.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[14] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. URL `http://arxiv.org/abs/1311.2524`.

[15] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[16] Rocking@home competition. URL `http://rockinrobotchallenge.eu/home.php`.

[17] D. H. HUBEL and T. N. WIESEL. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160:106–154, Jan. 1962. ISSN 0022-3751. URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1359523/`.

[18] Max pooling and average pooling. URL `http://www.embedded-vision.com/sites/default/files/technical-articles/CadenceCNN/Figure7.jpg`.

[19] Matrix convolution. URL `https://docs.gimp.org/en/plug-in-convmatrix.html`.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

[21] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and

K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 568–576. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5353-two-stream-convolutional-networks-for-action-recognition-in-videos.pdf`.

[22] Imagenet, . URL `http://www.image-net.org/`.

[23] How we teach computers to understand pictures — fei fei li — ted talks, . URL `https://www.youtube.com/watch?v=40riCqvRoMs`.

[24] Caltech 101. URL `http://www.vision.caltech.edu/Image_Datasets/Caltech101/`.

[25] Cifar-10-100. URL `http://www.cs.toronto.edu/~kriz/cifar.html`.

[26] Caviar dataset. URL `http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/`.

[27] Liris dataset. URL `http://liris.cnrs.fr/voir/activities-dataset/videoframes.html`.

[28] Robotic operative system. URL `http://www.ros.org/`.

[29] Thesis code repository. URL `https://bitbucket.org/tiagorbf/deep-video-behavior-classifier-thesis`.

[30] Keras documentation. URL `http://keras.io`.

[31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.