Department of Computer Science
Duale Hochschule Baden-Württemberg Stuttgart

Human Activity Recognition with Neural Networks

Bachelor Thesis

Author:	Ahmad Elsagheer
Supervisors:	Prof. Dr. Dirk Reichardt
	Ahmed Elnaggar, M.Sc.
Submission Date:	8 August, 2017

Department of Computer Science
Duale Hochschule Baden-Württemberg Stuttgart

Human Activity Recognition with Neural Networks

Bachelor Thesis

Author:	Ahmad Elsagheer
Supervisors:	Prof. Dr. Dirk Reichardt
	Ahmed Elnaggar, M.Sc.
Submission Date:	8 August, 2017

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowlegement has been made in the text to all other material used

---

Ahmad Elsagheer
8 August, 2017

Acknowledgments

First and foremost, I would like to express my deepest and most sincere gratitude to my family for everything they have done for me and all the love the gave to me. My mother, father, aunt, sister and brother. No words can express my love for them.

I would like also to thank my supervisors Prof. Dr. Dirk Reichardt and Ahmed Elnaggar, M.Sc. for their support and guidance through my bachelor project.

Last but no least, I would like to thank Prof. Dr. Slim Abdennadher, the Dean of the Department of Computer Science and Engineering at the Germany University in Cairo, for his continuous encouragement and support during all my study years. I gratefully acknowledge his inspiring guidance.

Abstract

We propose a machine learning approach for the activity recognition dataset [3] that was used in the 21st European Symposium on Artificial Neural Networks, 2013. The dataset consists of recordings of 30 test participants performing six basic activities of daily life. The dataset contains 561 features extracted from raw data collected using the accelerometer and gyroscope of a smart phone. Our proposed approach is a four-layer neural network model. Results show that the model accuracy on the test set is 97.42% which outperforms the state-of-the-art approaches.

Furthermore, we explore the activity recognition problem using convolutional neural networks. We collected data for EMG signals using an armband sensor. Recordings from 11 test participants are included in the dataset. Raw data were preprocessed and sampled over time. A conv net model is applied to the sampled examples. Results show that EMG signals are hard to train and need sophisticated preprocessing techniques and machine learning algorithms for pattern recognition rather than simple classification.

Contents

Chapter 1

Introduction

1.1 Motivation

The last three decades have seen a great focus from researchers on the Human Activity Recognition (HAR) problem. Activity recognition has many real-life benefits such as different applications in Ambient Assisted Living (AAL), health care, security, and surveillance.

Neural networks have recently revolutionized the machine learning field, especially in the image classification problem. Applying the modern techniques of neural networks can improve the accuracy of the HAR systems.

1.2 Aim of the project

The aim of the project is to design and implement a neural network model that can classify activity records into six basic activity classes with an accuracy close to the state-of-the-art approaches.

Moreover, we aimed to apply a Convolutional Neural Network (CNN) model to Electromyography (EMG) signals collected from Myo armband sensor and check how accurate we can classify activities using these EMG signals.

1.3 Organization of the thesis

The work in this thesis is organized as follows:

Chapter 1, introduces the importance of the thesis and the aim of the project.

Chapter 2, is a literature review about human activity recognition, the researchers' efforts and the state-of-the-art approaches related to our work.

Chapter 3, discusses the background behind the devices used in our project which are smart phone motion sensors and Thalmic Myo armband sensor.

Chapter 4, states the work flow through out the project and the objectives in every milestone.

Chapter 5, explains the neural network model for the public domain HAR dataset showing the tests and their results.

Chapter 6, explains our effort in applying convolutional neural networks on the Myo EMG signals stating the results and conclusions we reached.

Chapter 7, suggests ideas and enhancements that can be done and implemented in the future.

Chapter 2

Literature Review

2.1 Introduction

Human-centered computing is an emerging research field. It aims to understand how people behave and provide easy integration between humans and computer systems. Since the 1980s, HAR has become one of the main interests of computer scientists as it has many different applications in medicine, security, Human-Computer Interface (HCI) and many other fields.

2.2 The Meaning of "Activity"

"*Activity is a unit of life, mediated by psychic reflection, the real function of which is that it orients the subject in the objective world. In other words, activity is not a reaction and not a totality of reactions but a system that has structure, its own internal transitions, and transformations, its own development.*" [20]

Though there is no unique definition for the meaning of activity, we consider it as any action that a person takes consciously with his/her own choice, not as a reflex. In general, we can think of activities as a composition of other simpler activities which are, in turn, compositions of much simpler activities, till we end up with very basic activities that we don't need to break down anymore. For example, "going to sleep" can be regarded as a composition of "entering the bedroom", "turning the lights off" and "laying in bed", where "entering the bedroom" consists of "opening the door", where "opening the door" is a composition of some "hand movements".

Furthermore, activities can be related according to the time intervals in which they occur. Figure 2.1 shows these relations [22].

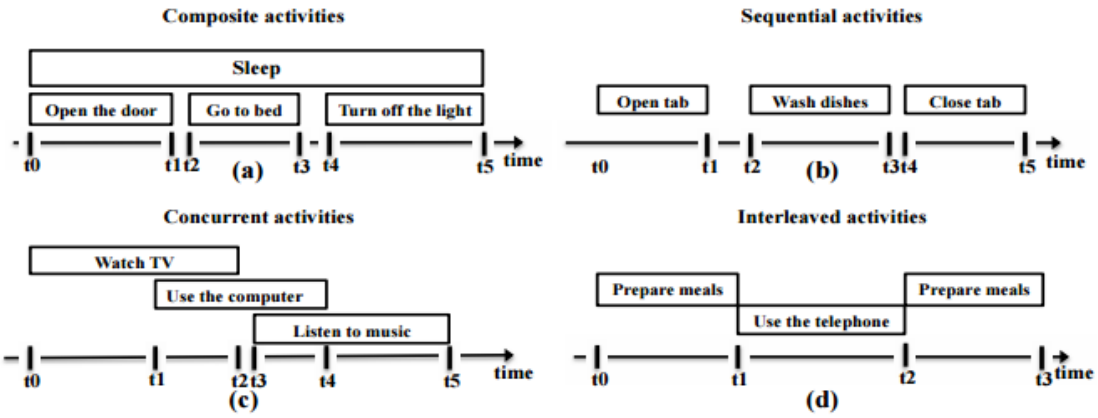


Figure 2.1: Relation between activities considering time intervals [22].

2.3 Human Activity Recognition

The main aim of activity recognition is to identify the actions of a person or multiple persons based on some observations from the person(s) and the surrounding environmental factors. Technically speaking, we have some activity records and want to assign for each record an activity label from a set of fixed categories. Such task may be trivial for humans to perform. However, it's a challenging task for computers as they need to be provided with enough data to identify the activity and need to be trained how to solve these tasks.

Similar to the image classification problem, it will be impractical to tell the computers how the activity record should look like to fall in a specific category. Here come the machine learning algorithms with their data-driven approaches. Analogous to how humans identify activities, computers are fed with a large set of examples associated with the correct labels, so they can learn how to identify activities for new unseen examples.

2.4 Process of HAR

Mainly, the HAR process consists of several steps [22][24]:

- Data Collection: collecting raw data from sensors. This raw data should be large and diverse enough to be able to generalize and produce accurate results when using the model in an application.
- Data Pre-processing: filtering noise and redundancy and handling missing values. Sometimes, this step includes data transformation such as normalizing the data to be able to relate data collected from different test participants.

- Data Segmentation: determining the important and effective data segments. Different approaches are used for data segmentation such as:
  - Activity-based segmentation: raw data are separated into different segments depending on the activity labels.
  - Time-based segmentation: raw data are separated into segments of specific time intervals. A sliding window can be used to let the segments overlap over time. This can be helpful in increasing the number of data records without decreasing the performance.
- Dimensionality Reduction: extracting the significant features that may affect the activity classification. This can be followed by reducing the number of features if some features are found to be insignificant to the classification process or if some features are homogeneous, such as a certain feature recorded over time, and we want to improve the computational speed. This can be regarded as *down sampling* of the features vector.
- Classification: identifying the activity.

Figure 2.2 summarizes the HAR process.

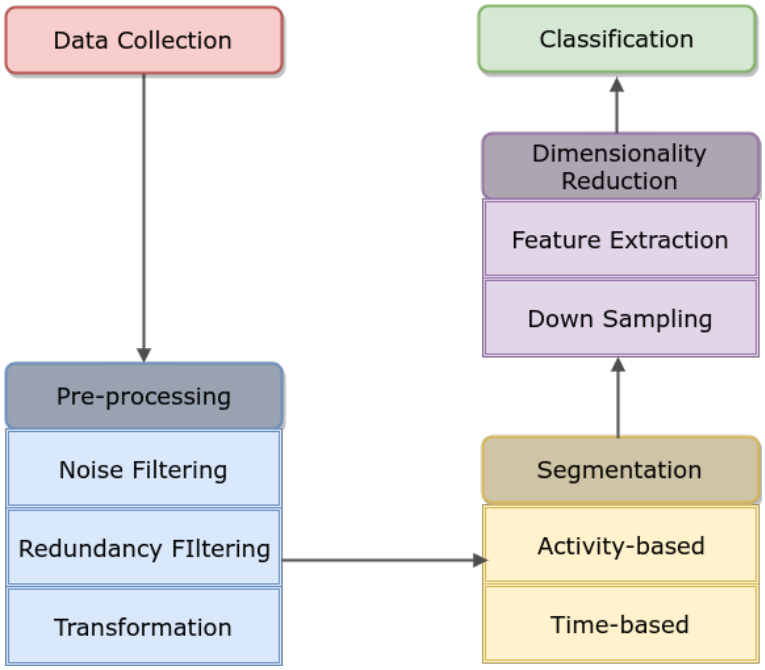


Figure 2.2: Process of HAR

2.5 Methods used in data collection and analysis

A lot of research has been done to provide practical solutions for many HAR problems. Many approaches depend on visual devices to identify and detect specific activities. Some of these approaches have showed good practical results in some applications such as AAL systems [14][7], security and surveillance [13][16], and health care monitoring [11][2]. One of the main advantages of visual-based systems is that they provide reliable data because the-state-of-the-art cameras have powerful resolutions. Furthermore, cameras can capture a wide range of the environment.

Another set of approaches depends on sensor devices such as sensors that can record heart rate, human-voice streams and EMG signals. A network of sensors provides a set of consistent information about the user behavior. Sensor-based systems have been used in health care [23] and AAL systems [18], as well. Some approaches combined between visual and non-visual devices [22][33][10].

2.6 Challenges facing HAR

Although visual-based systems can provide good accuracies, they have some limitations. They cannot get specific details, for example, for hidden objects. Moreover, they are sensitive to light and brightness factors and need more processing time. On the other hand, data collected from sensors can be associated with noise resulting in unreliable data. Therefore, preprocessing is usually needed by applying noise filters [24]. More research is needed to improve the performance of the current HAR systems and to recognize more complex activities.

2.7 Conclusion

The current main focus in the HAR field is novel machine learning algorithms. Machine learning has become so popular to solve many problems that are easily solvable by humans but hard to solve by machines without explicitly programming them to do specific tasks.

Deep learning is one of the machine learning approaches that is considered a breakthrough in machine learning and artificial intelligence in general. Nowadays, deep learning, which re-branded neural networks, can achieve results with accuracy close to human brains. CNN has provided extremely significant enhancements for the image classification problem. Applying neural networks on HAR is at its early stages and can revolutionize the whole field.

Chapter 3

Activity Recognition Devices

3.1 Smartphone Motion Sensors

All smartphones, nowadays, have built-in motion sensors such as the accelerometer and gyroscope. The accelerometer is used to measure the linear acceleration of movement. The gyroscope can track the rotation by measuring the angular velocity. Figure 3.1 shows a smartphone with the axes drawn. Transitional motion is what the accelerometer can detect. Rotations around any arbitrary axes can be measured by the gyroscope using *pitch*, *roll* and *yaw* angles. Combining a 3-axis accelerometer with a 3-axis gyroscope provides a full 6 degree-of-freedom (DoF) motion tracking system [27].

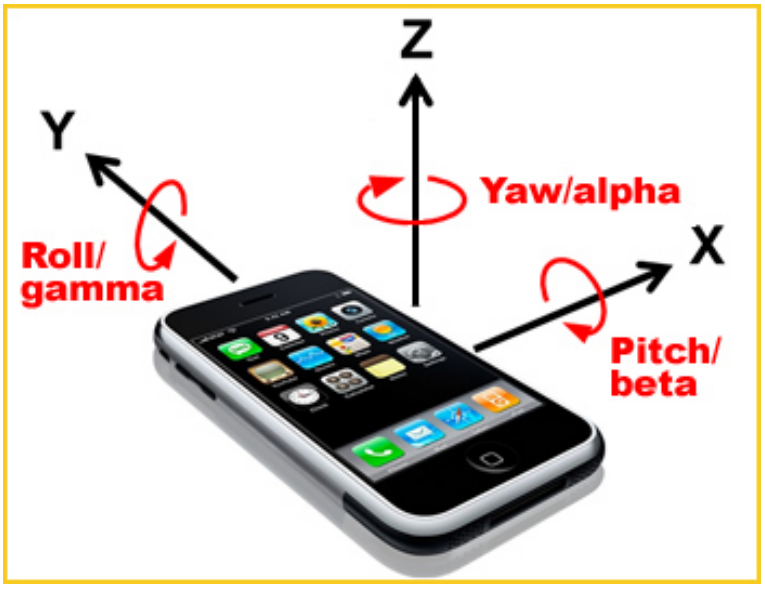


Figure 3.1: A smartphone with the three axes drawn [1]

3.2 Myo Armband Sensor

3.2.1 Architecture

The Myo armband sensor consists of medical grade stainless steel EMG sensors and highly sensitive nine-axis Inertial Measurement Unit (IMU) containing three-axis gyroscope, three-axis accelerometer, and three-axis magnetometer. It uses Bluetooth technology to connect to the computer [30].

Figure 3.2: Myo Armband Sensor [5]

3.2.2 Capabilites

The Myo sensor provides spatial data using IMU and gestural data using five already implemented gestures. In addition to that, it provides raw EMG data which is measured by the electrical sensors.

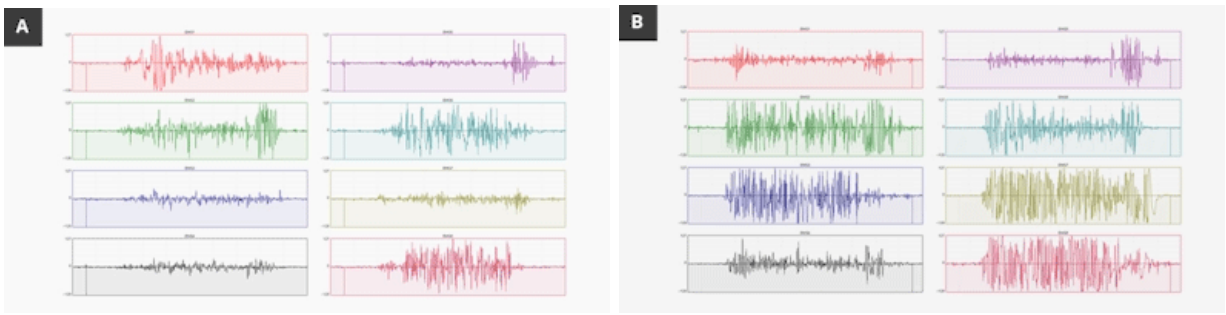
Advantages

The Myo sensor is a wearable sensor that is flexible and can be expanded between 17 - 34 cm forearm circumference. Also, it's light in weight (93 gm) and compatible with windows and iOS. [30]

Disadvantages

The Myo Software Development Kit (SDK) is not compatible with Linux which is the operating system we used for our project.

More importantly, raw EMG data are hard to deal with because these signals differ from one person to another. Figure 3.3 shows two raw EMG signals from two different persons performing the same gesture [4].



(a) (b)

Figure 3.3: EMG Signals from two different persons performing the same gesture [4]

3.2.3 The biology of EMG signals

EMG signals provide information about the muscle electrical activity. A specific type of neurons in human body called *motorneurons* controls muscle contractions. Electrical signals are carried by the motorneurons from the the central nervous system to the muscles causing the appropriate muscles to contract [9].

When muscle fibers receive the electrical impulse from motoneuron, muscle fiber action potential (AP) is generated which activates every segment of the muscle fiber. A single motoneuron is responsible for many muscle fibers. The motoneuron associated to it all of the muscle fibers responsible by that motoneuron is called a *motor unit*. The sum of electrical activity of all muscle fibers activated within the motor unit is known as motor unit action potential (MUAP) [25].

When the muscle contracts, some motor units are activated. The number of activated motor units depends on the required force. The higher the required force, the more motor units are activated. EMG signal at any moment of time is the composite electrical sum of all of the active motor units. The amplitude of EMG signal increases as the intensity of the muscular contraction increases [25].

It's worth noting that if the sensor electrodes are surface electrodes, deep muscle fibers have a little contribution to the recorded EMG signal. Surface EMG assesses muscle function by recording muscle activity from the surface above the muscle on the skin. Surface electrodes are able to provide only a limited assessment of the muscle activity [25]. In Myo sensor, surface EMG is recorded by 8 electrodes. Each of the 8 EMG channels displays the difference between two adjacent electrodes.

Chapter 4

Methodology

This chapter states the work flow of the project. The project was divided into three milestones.

4.1 Milestone 1: Deep Learning and Machine Learning Libraries

The first milestone was concerned with researching into deep learning and the-state-of-the-art techniques used in machine learning algorithms.

This was achieved by studying the course *Convolutional Neural Networks for Visual Recognition* which is offered online by Stanford University [32]. Several classifiers have been implemented during the study process such as *K-Nearest Neighbours (KNN), Multi-Class Support Vector Machine (SVM)* and *Softmax* classifiers. Moreover, a Two-Layer Neural Network and Convolutional Neural Network were implemented. These classifiers were applied on the CIFAR-10 dataset [19] for the image classification problem. This course helped to:

- understand and grasp the concepts of machine learning algorithms and particularly neural networks.
- understand the logic behind the functions implemented in the machine learning libraries such as *back propagation* because the implemented classifiers required implementing these low-level functions from scratch.
- learn best practice techniques when designing, implementing and evaluating neural network models.

In addition, we explored the following Python libraries:

- *pandas*: to organize and manipulate dataset files.
- *NumPy*: for most of the computations, especially computations that include high dimensional arrays and manipulation of matrices.
- *scikit-learn*: to use already implemented basic machine learning functions such as splitting the training dataset into folds for cross-validation.
- *matplotlib*: to plot and visualize results.
- *TensorFlow*: which we explored deeply because the whole project is based on this machine learning library. We used the low-level API to provide us with full control over our code. We also used *TensorBoard* to visualize and compare the results.

4.2 Milestone 2: Smartphone Motion Sensors and Deep Neural Network Model

In this milestone, we explored the public domain dataset [3] for HAR which contains activity records collected using smartphone motion sensors that were mentioned in section 3.1. Based on milestone 1, we implemented a deep neural network model for this dataset. The process of building this model was done in a professional way. Our final model achieved an accuracy better than the state-of-the-art approaches. At this point, we should mention two important processes done at this milestone:

4.2.1 Tuning Hyperparameters

Every machine learning model has some parameters that must be set before training such as the learning rate, the number of hidden layers and their dimensions. The model *depends* on these parameters to be able to train. Therefore, these parameters are not trainable parameters and we must tweak them to produce good results for our dataset.

We tuned the hyperparameters one by one while fixing all other parameters. For each hyperparameter, we tried different values and plotted the results for some variables to test the performance such as the loss function, the accuracy and the histograms for the distribution of weights at each layer.

4.2.2 Model Evaluation

Our main performance metric for our model is the *accuracy* which is the percentage of examples that are predicted correctly.

We cannot tune the hyperparameters using the training set and relying on the training accuracy only, because the algorithm can memorize all the training examples, but fails to produce good results when it is asked to predict labels for unseen examples. In this case, we say that the model *overfits* on the training set. That's why we used a validation set which is a subset of the training set but not used in the training process. This validation set is used to measure the accuracy and find the optimal choice for each hyperparameter.

Note that we didn't use the test set for tuning the hyperparameters, because we may tweak them to produce good results on the test set, but the algorithm will fail to generalize on other examples. So, we will only use the test set as a final performance measure after tuning all hyperparameters.

4.3 Milestone 3: EMG Signals and CNN model

This milestone was concerned with Myo armband sensor. Mainly, it consists of three parts:

1. understanding the biological background of EMG Signals which represent the raw data collected from the Myo sensor, in addition to understanding how the Myo sensor works and how to use its API to collect the data.
2. collecting and preprocessing raw EMG data for each activity class using the Myo armband sensor. Data was collected from 11 test participants.
3. Building a CNN model that can classify activity records based on the EMG signals.

It is worth noting that computing time for training the CNN was too much due to the limited capability of the processor (intel core i3), which we used for training the model. This caused progress in this milestone to be slow.

Chapter 5

Deep Network Model For ADL

5.1 ADL Dataset and the State-of-the-Art Approaches

In this chapter, we propose a deep neural network model for the public domain dataset [3] that targets the recognition of six basic Activities of Daily Living (ADL). The activities are are: *standing*, *sitting*, *laying*, *walking*, *walking downstairs* and *upstairs*. Raw data were collected from 30 test participants using the accelerometer and gyroscope of a smartphone. Then, the collected data were processed further to extract features such as mean, correlation and signal magnitude area. The final shape of the dataset is a set of 10299 examples, where each example is a vector of 561 features. 70% of the dataset is used for training while the other 30% is used for testing.

The state-of-the-art approach is One-Vs-One Multiclass linear SVM with majority voting which achieved an accuracy of 96.4% [26]. One of the implemented neural network models for this dataset used a bidirectional Long Short-Term Memory (LSTM) achieving an accuracy of 94% [8].

5.2 Deep Neural Network Approach

We applied modern techniques of neural networks to find out the best accuracy which we can reach on this feature-extracted HAR dataset with a neural network model. We split the training set into 7 equal folds so that each fold represents approximately 14% of the training set. While tweaking the hyperparameters, we either used the first fold as a validation set or performed cross-validation if the differences in the accuracy between the parameter choices are within the error of the validation accuracy. The validation set is important to tune the hyperparameters such as learning rate, learning method, the number of hidden layers and their dimensions. Consequently, the test dataset is only used after tuning all the hyperparameters.

5.2.1 Initial Network Configuration

Initially, we used a four-layer network model. The dimensions of the hidden layers are 1024, 512 and 64 (in the direction from the input layer to the output layer). Each hidden layer is followed by a Rectified Linear Unit (ReLU) unit. Furthermore, we used a 0.001 learning rate and 5,000 steps for the training process. At each step, only a mini batch of 256 examples is used in training to reduce the computational time. The mini batch is selected randomly at every training step. Figure 5.1 shows the architecture of the network.

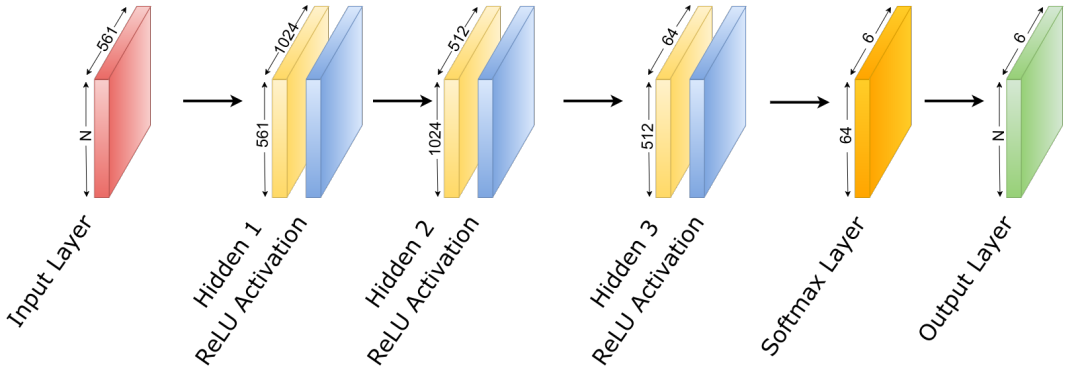


Figure 5.1: Initial Network Architecture

Each hidden layer is a fully connected layer. In other words, each input example is multiplied by the affine matrix of the layer. Afterward, a bias vector is added to the result. This can be represented by the following formula:

$$Y = f(X) = XW + B$$

where:

- X is the input matrix of dimensions $N \times M$.
- W is the weight matrix of dimensions $M \times K$.
- B is the bias vector of dimensions $1 \times K$.
- Y is the output matrix of dimensions $N \times K$

For each hidden layer, the parameter K is the hyperparameter of that layer. N and M depends on the input (possibly the output of the previous layer).

Using only fully connected layers will eventually result in a linear model. The ReLU units after each fully connected layer add non-linearity to the network and allow the model to learn from a big family of functions. The ReLU function can be represented mathematically as follows:

$$relu(X) = max(0, X)$$

The softmax layer is a fully connected layer followed by the softmax function. The output of the fully connected layer represents the scores assigned for each activity class. We interpret each activity score as the *log probability* that this activity is the correct label for the input example. When applying the softmax function, the scores are converted into *normalized probabilities*. For each example x with scores $s_1, s_2, ..., s_m$ where m is the number of activities, the softmax function for activity i is:

$$\sigma(x)_i = \frac{e^{s_i}}{\sum_j^m e^{s_j}}$$

The output layer is an $N \times 6$ matrix, where cell y_{ij} represents the probability that the j-th activity is the correct label for the i-th example. For each example, the activity with the highest probability is the predicted activity.

5.2.2 Learning Optimizer

The core idea of machine learning algorithms is to minimize a loss function which compares the real labels with the predicted labels.

Loss Function Since we are using a softmax layer, the loss function we are using is the cross-entropy function which can be formulated as follows:

$$l(\boldsymbol{w}) = -\sum_i^N \sum_j^M p_{ij} \log q_{ij}$$

where:

- N is the number of examples.
- M is the number of activity classes.
- $\boldsymbol{w}$ is the values of the weights variables.
- p_{ij} is the true probability (from the dataset) that the j-th activity is the correct label for the i-th example. Note that for a fixed example, only one activity will have $p = 1$ while other activities will have $p = 0$.
- q_{ij} is the estimated probability (by the network) that the j-th activity is the correct label for the i-th example.

We tested and compared the performance of some popular optimizers:

Gradient Descent Gradient descent is the simplest way to minimize the loss function. Weights are updated along the negative direction of the gradient. The size of the update step is equal to the learning rate. Actually, we used Stochastic Gradient Descent (SGD) instead of true Gradient Descent to reduce the computing time [6]. If our learning rate is γ, then the update step of Gradient Descent can be represented as follows:

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} - \gamma \cdot \nabla l(\boldsymbol{w}^{(k)})$$

Adagrad Adagrad is an adaptive learning rate method that was proposed by [12]. It accumulates the squared gradients and uses the accumulated sum (cache) in scaling the learning rate. The learning rate is divided by the square root of the cache. In this way, if the gradient is high, the learning rate is reduced whereas if the gradient is small, the learning rate is increased. Formally:

$$\boldsymbol{c}^{(k+1)} = \boldsymbol{c}^{(k)} + \nabla l(\boldsymbol{w}^{(k)})^2$$

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} - \gamma \cdot \frac{\nabla l(\boldsymbol{w}^{(k)})}{\sqrt{\boldsymbol{c}^{(k+1)}}}$$

RMSProp RMSProp is a modification for Adagrad. It applies a decaying function to the cache, to reduce the decay of the learning rate [31]. We can represent it mathematically as follows:

$$\boldsymbol{c}^{(k+1)} = \beta \cdot \boldsymbol{c}^{(k)} + (1-\beta) \cdot \nabla l(\boldsymbol{w}^{(k)})^2$$

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} - \gamma \cdot \frac{\nabla l(\boldsymbol{w}^{(k)})}{\sqrt{\boldsymbol{c}^{(k+1)}}}$$

Figure 5.2 compares the loss functions of the above learning optimizers. Adagrad has better convergence than Gradient Descent. Noticeably, RMSProp has the lowest curve for the loss function and approaches zero loss. However, its loss function has some glitches that cause an unstable behavior, because the final model after all training steps may settle at one of these glitches. We will investigate these glitches in sections 5.2.6 and 5.2.7.

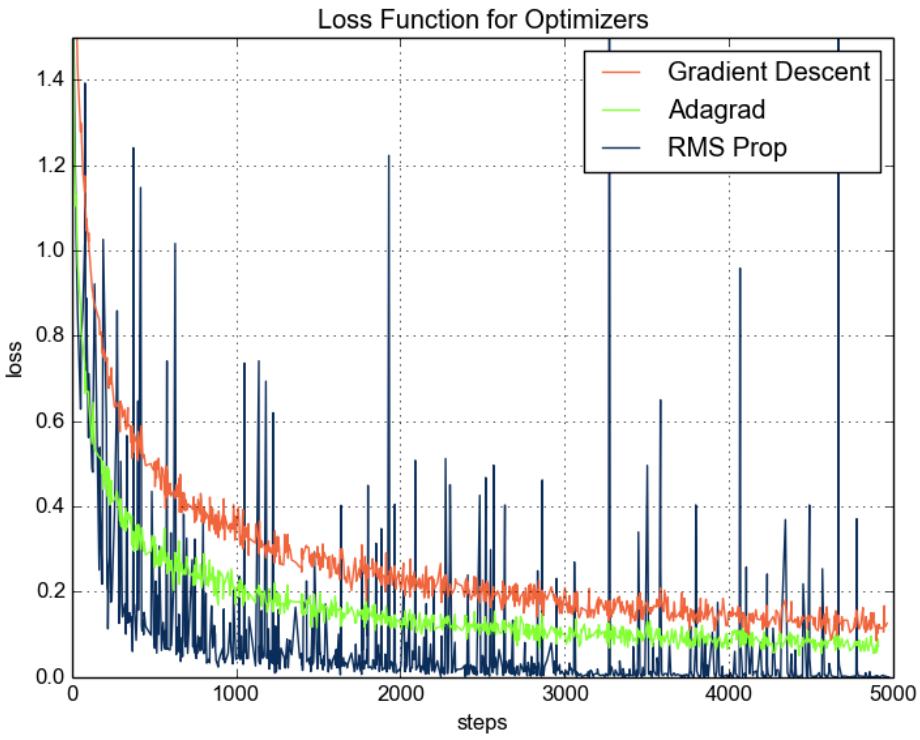
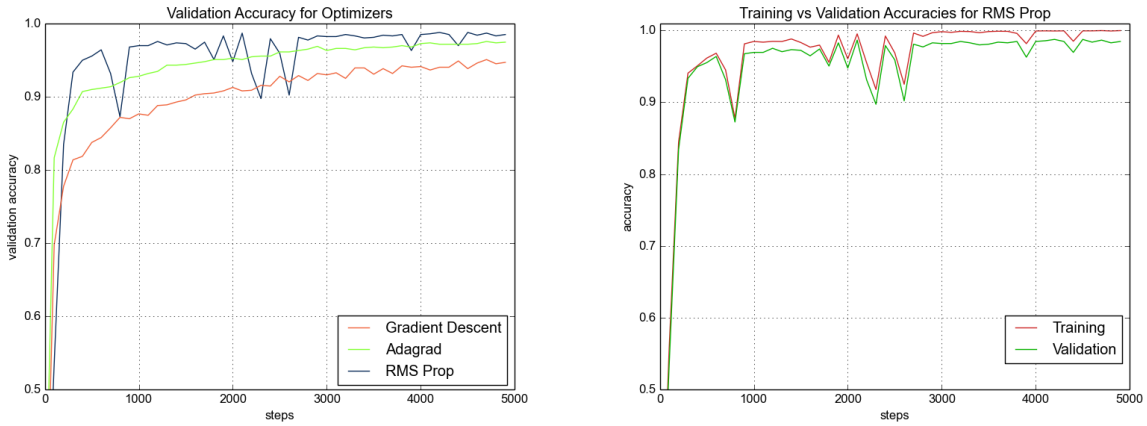


Figure 5.2: Loss function for optimizers

Checking the performance of each optimizer on the validation set, we found out that RMSProp outperforms both Adagrad and Gradient Descent, as shown in figure 5.3a. Note that there is a little acceptable overfitting as shown in 5.3b. In conclusion, we will use RMSProp for our model and to tune the hyperparameters.



(a) Validation accuracies for optimizers

(b) Training and validation accuracies for RMSProp

Figure 5.3: Accuracy plots for optimizers

5.2.3 Weights and Biases Initialization

One of the most important things to take into consideration when building a neural network is the initialization of the trainable parameters. A good initialization will help the network to learn faster and achieve better accuracy.

Using a single validation set results in accuracies with 1.5% error. So, we used cross-validation to compare results using mean accuracies.

We tried five initialization methods and monitored their behavior and accuracies. The methods and their results are shown in table 5.1. Truncated Normal Distribution (N.D) was used in all methods.

#	Weights	Biases	Train Acc (%)	Val Acc (%)
1	N.D, $\sigma = 0.1$	N.D, $\sigma = 0.1$	99.96	98.63
2	N.D, $\sigma = 0.1$	zeros	99.91	98.65
3	N.D, $\sigma = 0.01$	zeros	99.87	98.42
4	N.D divided by $\sqrt{fan\_in/2}$, $\sigma = 0.01$	zeros	99.72	98.41
5	N.D divided by $\sqrt{fan\_in/2}$, $\sigma = 0.1$	zeros	99.97	98.49

Table 5.1: Comparison between initialization methods

Figure 5.4 shows the cross validation results for different methods. For each method, validation accuracy for each fold is plotted and the standard deviation is calculated. The green line connects the mean accuracy of all methods.

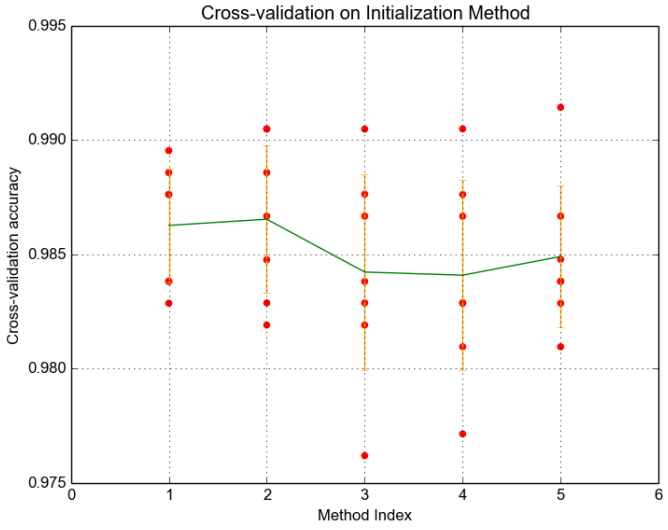


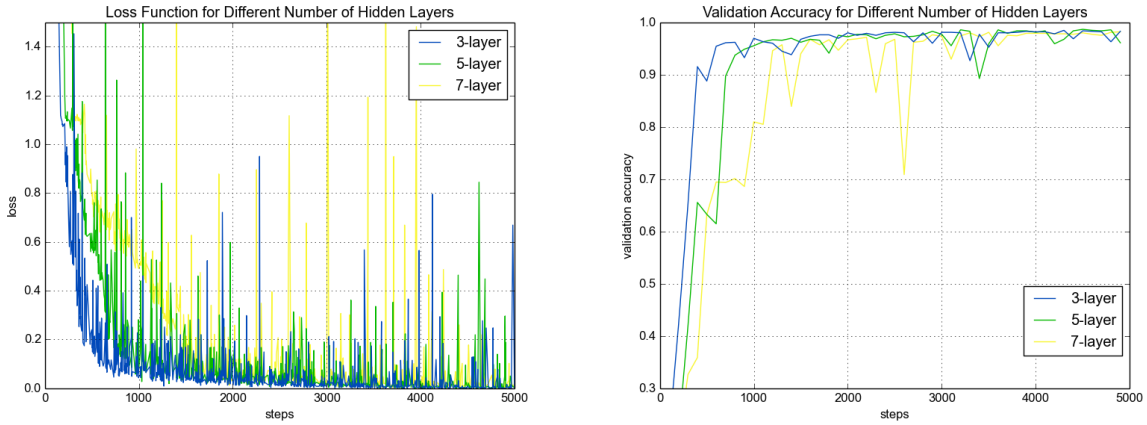
Figure 5.4: Cross-Validation for initialization methods

Note that we didn't include methods that initialize the weights with zeros. In general, this is a bad practice because all neurons will be computing the same values and will have equal gradients. So, we need to break this symmetry using random values.

It was suggested by [15] to initialize weights by method 4 or 5 (σ depends on the dataset) as it has shown, on image classification, better distribution of the weight values especially for deep networks. But, since our network is not very deep, the results showed that method 1 and 2 are more accurate on the validation set. In conclusion, we will use method 2 for our network weight initialization.

5.2.4 Number of Hidden Layers and Their Dimensions

Here, we tried to set a good architecture for the network. We tested networks with 3, 5, and 7 hidden layers where the dimensions of the layers are powers of two. It has been found that choosing power-of-two dimensions achieves faster computation [21]. Figure 5.5 compares the performance of the tested architectures.



(a) Loss function

(b) Validation Accuracy

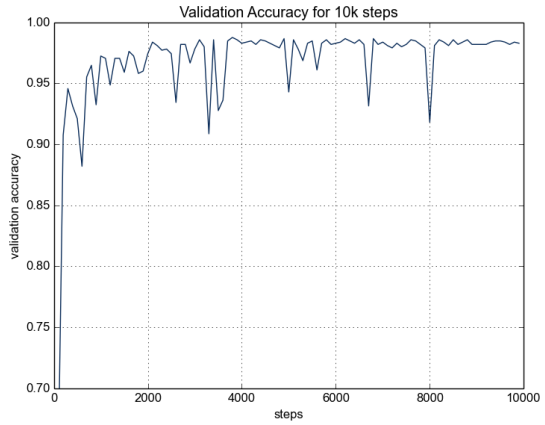
Figure 5.5: Performance of different network architectures

From the plots, we can see that all networks converge to almost the same loss value and achieve close accuracies on the validation set. However, a network with a fewer number of layers converges faster than a network with more number of layers. Furthermore, the fewer the number of layers, the faster the computation is, because the number of matrix multiplications is reduced in this case. To conclude, we will use the 3-hidden-layer network model for our approach.

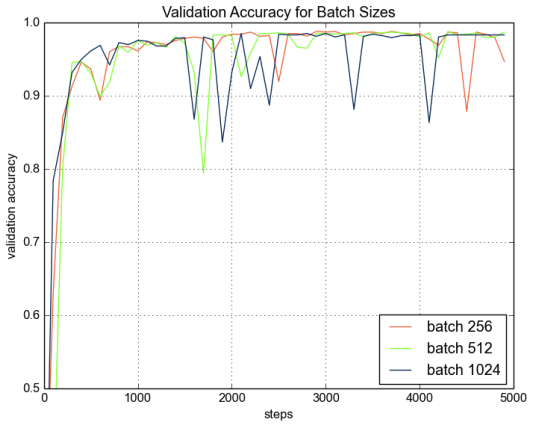
5.2.5 Training Steps and Batch Size

The time the model takes to train is essential in machine learning problems because the model is usually fed with an enormous dataset. We ran our model for 10K steps and plotted the validation accuracy as shown in figure 5.6a. We can see from the figure that accuracy reached its maximum value and saturates at 5K steps. So, there is no need to run the model for more than 5K steps.

Another point to consider is the number of examples used to train the model at each training step. We tried batches of sizes 256, 512 and 1024, and plotted the results in figure 5.6b. Notice that bigger batches have larger glitches. Mini batches with size 256 were enough to train the model.



(a) Validation accuracy over 10K steps



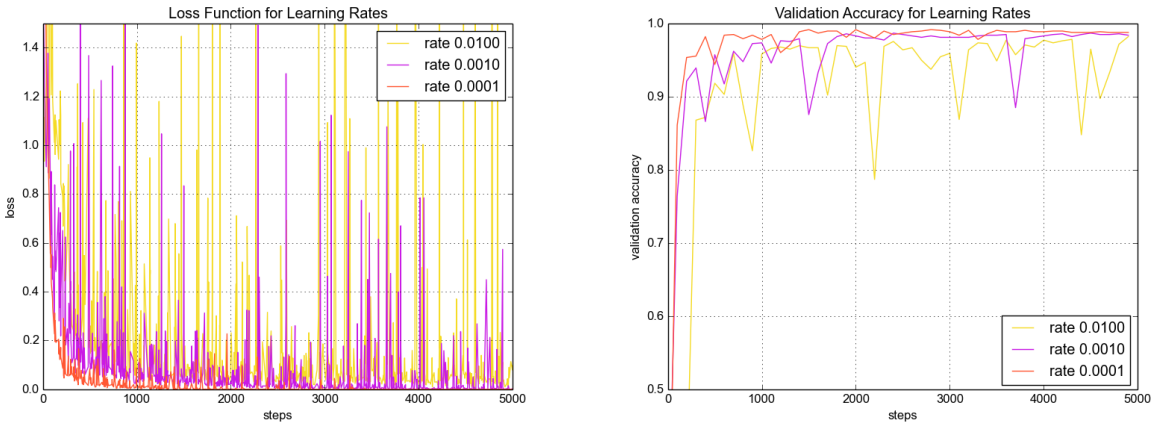
(b) Validation accuracies for different batch sizes

Figure 5.6: Validation accuracies for number of training steps and batch size

5.2.6 Learning rate

The rate at which weights move towards the global minimum of the loss function needs to be finely tuned. High learning rates may result in a model overshooting the global minimum. On the other hand, low learning rates may make the model take too much time till it reaches the global minimum.

We test different values for the learning rate (0.01, 0.03, 0.07, 0.001, 0.0007, 0.0001). Figure 5.7 shows the results for learning rates 0.01, 0.001 and 0.0001. We can see from the figure that the model is more stable with smaller learning rates and glitches decrease when the learning rate decreases. This is because learning rate indicates the size of the step each weight variable moves towards the global minimum. So, when a weight is being trained in an incorrect direction for a single train step, it will not move too far away from the correct direction as it will be fixed in the next step. Though Gradient Descent and Adagard performed well with learning rate 0.001, these results show that RMSProp is more sensitive to the learning rate. Furthermore, lower learning rates have better results on the validation set than higher learning rates.



(a) Loss function

(b) Validation accuracy

Figure 5.7: Comparison between different learning rates

5.2.7 Batch Normalization

As we have seen that low learning rates produced better results, it will be too slow to try very low learning rates as that will require more training steps. Batch normalization, which was originally proposed by [17], was tested in our network model. We added a normalization layer after the activation function of each hidden layer. Batch normalization normalizes each feature using mean and variance of the mini-batch examples, then it allows scaling and shifting. This can be represented by the following formula:

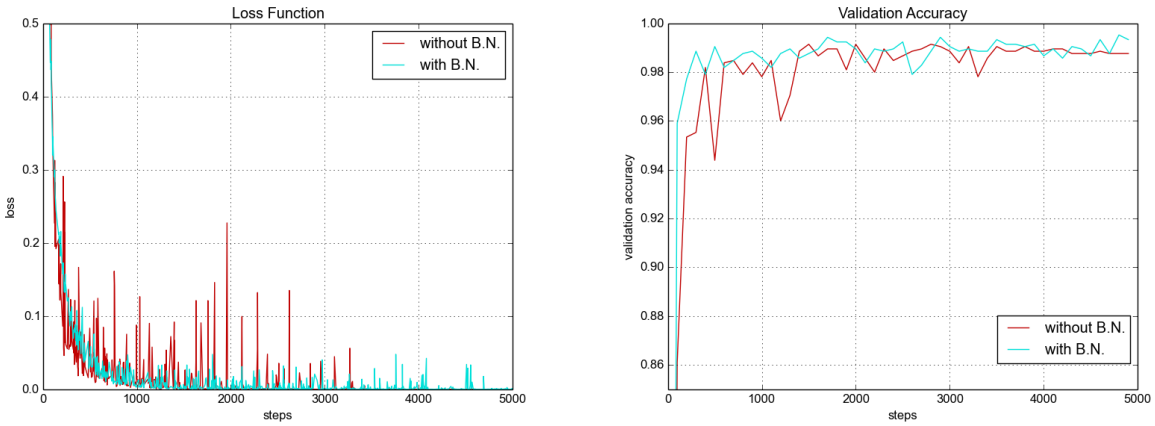
$$y = \alpha\hat{x} + \beta$$

where:

- $\hat{x}$ is the normalized feature.
- α is the scaling factor.
- β is the shifting factor.
- y is the output feature.

We allow the parameters α and β of each layer to be trainable.

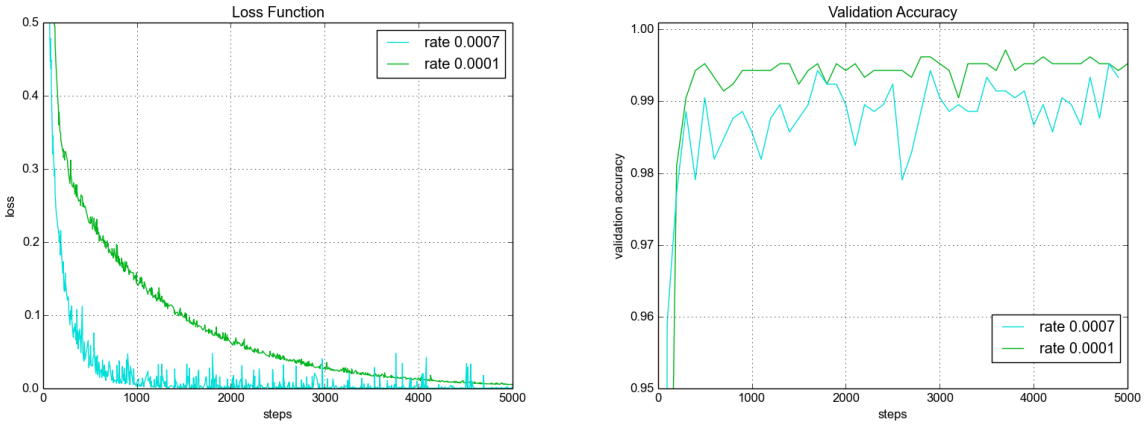
Figure 5.8 shows a comparison between a network with batch normalization and learning rate 0.0007 with another network without batch normalization and learning rate 0.0001. As we can see from the figure, the network with batch normalization has a more stable loss function and higher accuracy on the validation set.



(a) Loss function (b) Validation accuracy

Figure 5.8: Network with and without batch normalization

Another comparison is shown in figure 5.9 for two networks with batch normalization but two different learning rates: 0.0007 and 0.0001. Although the network with learning rate 0.0007 has a better decrease in the loss function at the beginning, it saturates. On the other hand, 0.0001-rate network has a good loss function and its accuracy on the validation set exceeds 99.5% which is an extremely significant improvement for the network.



(a) Loss function (b) Validation accuracy

Figure 5.9: Batch Normalization with different learning rates

5.3 Conclusion

Our final model is a four-layer neural network. The dimensions of the hidden layers are 1024, 512 and 64 (in the direction from the input layer to the output layer). Each hidden

layer is followed by a Rectified Linear Unit (ReLU) unit, which is followed by a batch normalization layer. The learning rate is 0.0001 with an RMSProp Optimizer and 5,000 steps are used for the training process with a batch size of 256 examples. Figure 5.10 shows the architecture of the network.

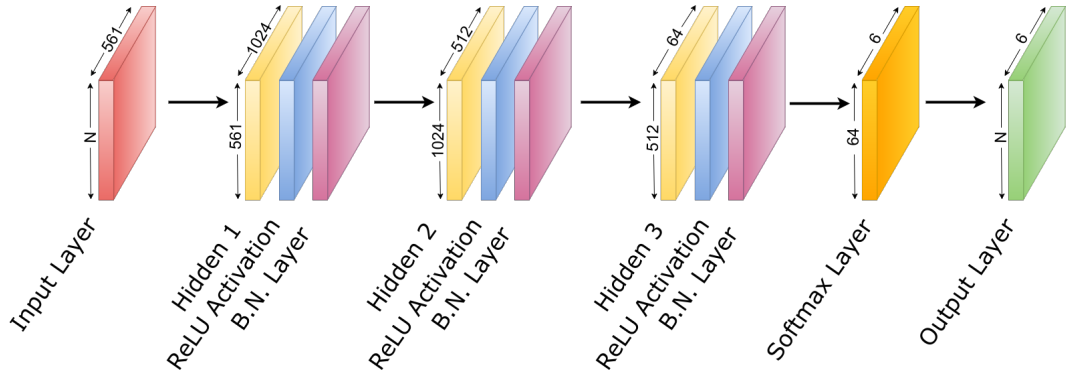


Figure 5.10: Final Network Architecture

The final best accuracy we could achieve with our neural network model is 97.42%. The confusion matrix is show in figure 5.2. Rows represent the real labels while columns represent the predicted labels. Rows and columns labels are abbreviated as follows: WA-*walking*, UP-*walking upstairs*, DO-*walking downstairs*, SI-*sitting*, ST-*standing*, LA-*laying*.

Activity	WA	UP	DO.	SI	ST	LA	Accuracy (%)
WA	491	2	3	0	0	0	98.99
UP	4	467	0	0	0	0	99.15
DO	2	10	408	0	0	0	97.14
SI	0	2	0	452	37	0	92.05
ST	0	0	0	16	516	0	96.99
LA	0	0	0	0	0	537	100.00
Precision (%)	98.79	97.09	99.27	96.58	93.31	100.00	**97.42**

Table 5.2: Confusion matrix for the neural network model

We can see from the confusion matrix that *sitting* and *standing*, as expected, are close to each other. *Sitting* has the worst accuracy as many of its records are mistakenly predicted as *standing*. On the other hand, *Laying* records are very clear and easily distinguished from other activities. Other wrong predictions are among the three walking activities which is logically possible. A single strange result is predicting two *sitting* records as a *walking upstairs* activity. The proposed network outperforms the state-of-the-art approaches.

Chapter 6

Convolutional Neural Networks on EMG Signals

CNN has become an important tool for object recognition in Computer Vision [28]. It has revolutionized the way the image classification problem is tackled. It is currently used by Facebook to detect faces in a photo and produces awesome results [29]. We tried to apply CNN on the activity recognition problem.

6.1 Data Collection

6.1.1 Experiment Procedure

We collected data from 11 23-years-old test participants using Myo armband sensor worn in the right forearm. The sensor was worn such that channel 4 and the status LED faces the wrist (see section 3.2). For each participant, we recorded 4-5 seconds for each activity. 10 activities have been recorded. The activities are:

1. *walking*: most of the humans swing their arms while walking such that each arm swings with the motion of the opposing leg. The walking activity recorded making sure that all participants swing their arms while walking.
2. *standing*: not making any movements or gestures with the hand.
3. *talking in phone*: by putting the cell phone on the right ear.
4. *typing on keyboard*: using all fingers.
5. *writing on desk*: in a notebook with the right hand on the desk.
6. *drawing on board*: while standing.

7. *brushing teeth*: making horizontal and vertical moves.
8. *holding sphere*: a ball that can be gripped by hand.
9. *holding cylinder*: a cylindrical shape with radius 2.5 cm.
10. *holding thin sheet*: a thin sheet with thickness 3 cm.

6.1.2 Preprocessing of Raw Data

The Myo sensor collects data with frequency 200 Hz. 800-1000 time frames have been recorded for each participant activity. So, each participant has 10 Comma-Separated Valus (CSV) files, one for each activity. Rows in each file correspond to the time frames while columns correspond to the 8 EMG values associated with each time frame.

This raw data need further preprocessing before feeding it to the classifier. We need to organize our records into examples such that each example has:

1. Specific number of time frames.
2. EMG values for each time frame.
3. Activity label.

Each activity record is sampled into a set of examples using a sampling rate of 128 Hz with 50% overlapping between consecutive examples. The activity labels are set manually. Each example has dimensions 128×8. The time dimensions is said to be the *width* of the example while the EMG channels dimension is said to be the *depth*. Examples are pickled into two multi-dimensional arrays:

- Features array: with dimensions $N \times M \times C$ where:
  - N is the number of examples.
  - M is the width of the example.
  - C is the depth of the example.
- Labels array: with dimensions $N \times V$ where:
  - N is the number of examples.
  - V is the number of activity classes.

For each row (example), only one column (activity class) has value 1 while other columns have value 0. This is known as *one-hot encoding*.

6.2 Network Configuration

Our network consists of 2 convolutional layers with 16 and 32 filters, respectively. Each convolutional layer is followed by a ReLU function followed by a Batch Normalization (BN) layer. After the convolutional part, there are 3 fully connected layers with dimensions (64, 16, 10). The first two fully connected layers are followed by a ReLU function followed by BN layer. The last fully connected layer is followed by the softmax function. We used RMSProp optimizer with learning rate 0.001 and momentum 0.7. Figure 6.1 shows the architecture of the network.

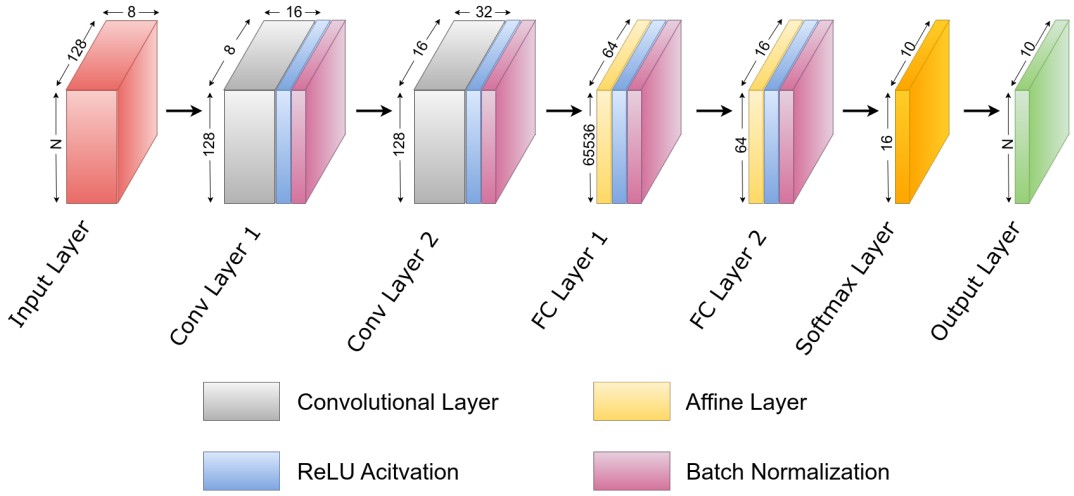


Figure 6.1: CNN architecture

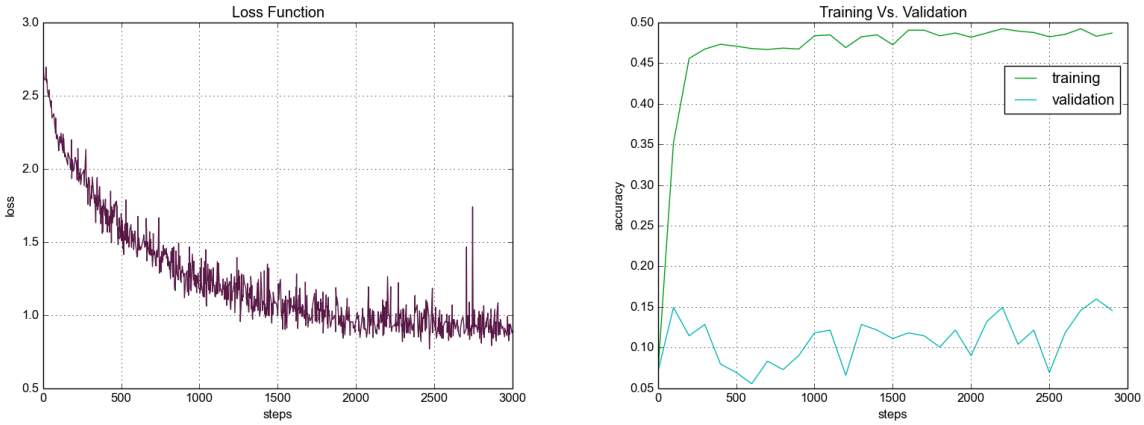
Each convolutional layer consists of a number of filters. Each filter is a 1×7 matrix and slides with stride equal to 1 during the convolution process. The output of the convolutional layer has the same dimensions as the input layer except for the depth dimension which is equal to the number of filters in the output.

6.3 Tests and Results

We tried to run the network on different subsets of the activity classes. Below we show the results of three of these trials.

Comparison 1: *walking*, *talking in phone* and *drawing on board*

We ran the network for examples with the activities: WA-*walking*, TA-*talking in phone* and DR-*drawing on board*. Figure 6.2 shows the loss function and accuracies. As shown in the figure the training accuracy didn't exceed 50% and the validation accuracy is 16% by the maximum.



(a) Loss function

(b) Accuracies

Figure 6.2: Loss and accuracies for CNN comparison 1

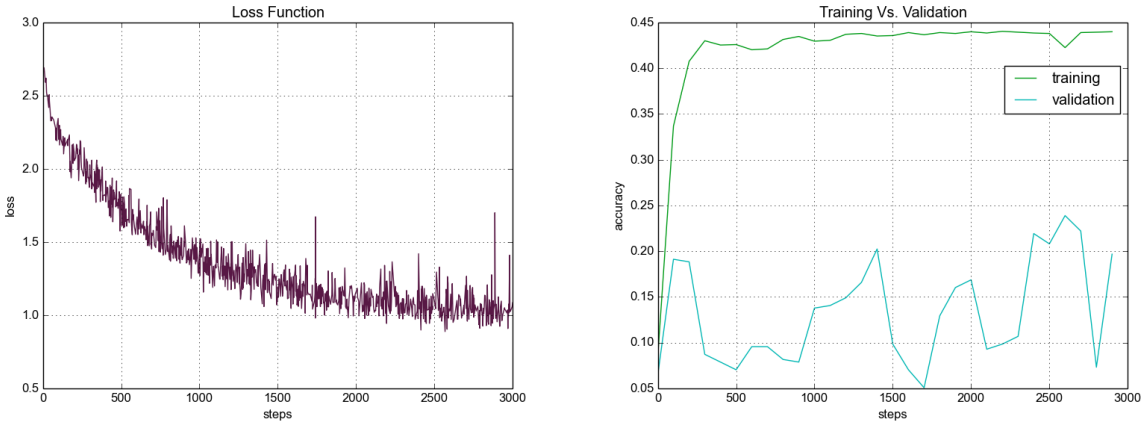
Table 6.1 shows the confusion matrix on the training set. The *walking* activity was predicted correctly with a reasonable accuracy. More than half of the *talking in phone* activity records were predicted correctly, however, a great portion was mistakenly predicted as *walking*. *Drawing on board* has the worst accuracy.

Activity	WA	TA	DR	Accuracy (%)
WA	346	73	4	81.80
TA	263	367	2	58.07
DR	319	231	118	17.66
Precision (%)	37.28	54.69	95.16	48.23

Table 6.1: Confusion matrix for CNN comparison 1

Comparison 2: *talking in phone*, *typing on keyboard* and *writing on desk*

In comparison 2, we tested TA-*talking in phone*, TP-*typing on keyboard* and WR-*writing on desk*. In figure 6.3, it is shown that we have a worse accuracy on the training set, but a better accuracy on the validation set, than comparsion 1.



(a) Loss function (b) Accuracies

Figure 6.3: Loss and accuracies for CNN comparison 2

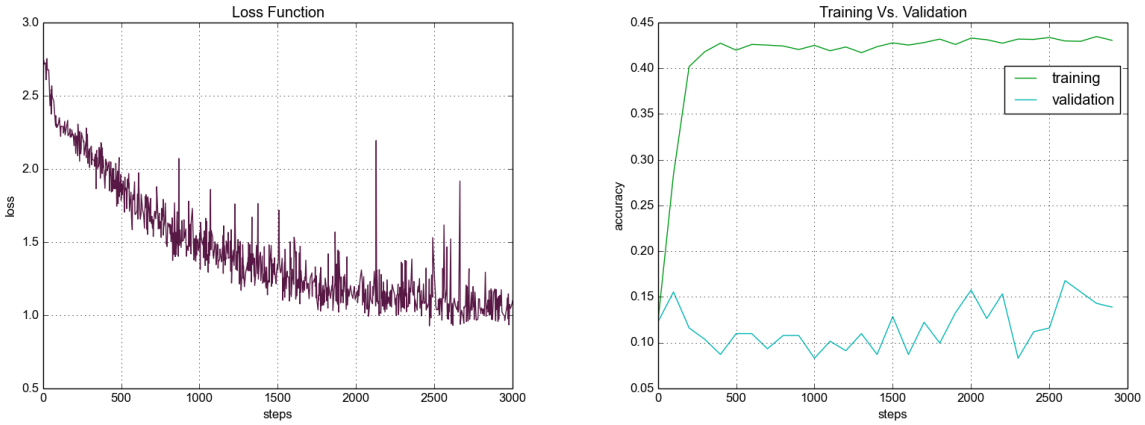
Table 6.2 shows that *writing on desk* has good precision. That might be due to the difference in the fingers configuration in *writing on desk* from *talking in phone* or *typing on keyboard*, because holding the pen makes the fingers close to each other.

Activity	TA	TP	WR	Accuracy (%)
TA	308	307	21	48.43
TR	221	483	14	67.27
WR	188	452	139	17.84
Precision (%)	42.96	38.89	79.88	43.60

Table 6.2: Confusion matrix for CNN comparison 2

Comparison 3: *holding sphere*, *holding cylinder* and *holding thin sheet*

Our last comparison is for static holding positions: HS-*holding sphere*, HC-*holding cylinder* and HT-*holding thin sheet*. We expected the network to produce better results. However, figure 6.4 shows that results are no better than the previous comparisons.



(a) Loss function (b) Accuracies

Figure 6.4: Loss and accuracies for CNN comparison 3

Table 6.1 shows the confusion matrix on the training set. Differentiating between these three holding activities was a challenging task for our model.

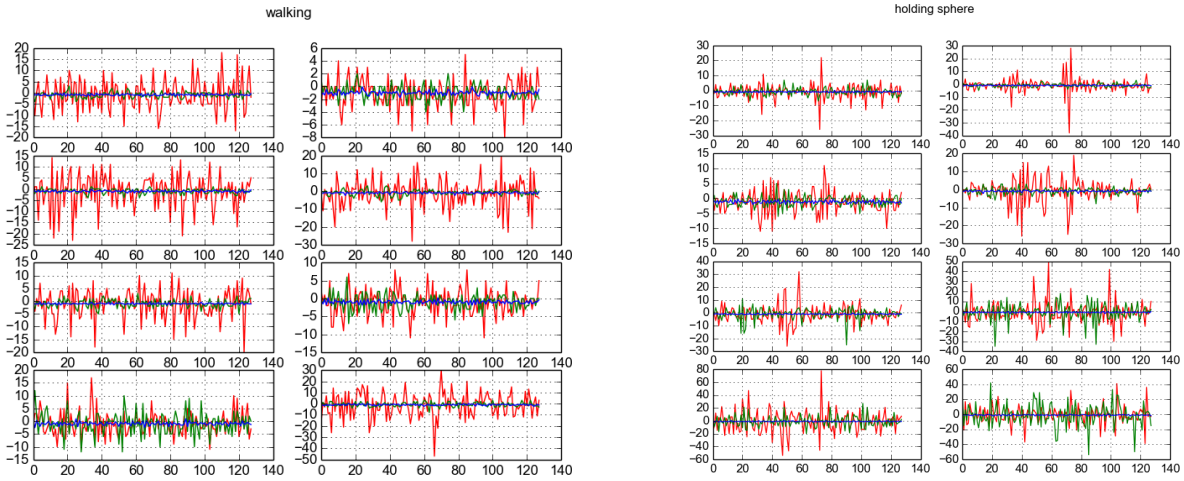
Activity	HS	HC	HT	Accuracy (%)
HS	336	269	372	34.39
HC	204	333	359	37.17
HT	136	297	587	57.55
Precision (%)	49.70	37.04	44.54	43.42

Table 6.3: Confusion matrix for CNN comparison 3

6.4 Conclusion

The network cannot exceed 50% accuracy on the training dataset or 25% accuracy on the validation set when classifying only 3 activity classes.

We took some random samples, averaged the EMG signals over time and plotted the averaged example against each sample. Figure 6.5 shows these plots for *walking* and *holding sphere*, respectively. The blue line represents the averaged example while the red and green lines represent two random examples from the dataset. As shown in the figure, there is a deviation from the averaged example in some channels while other channels have values close to the averaged signal. Although these signals represent the same activity, they have a little in common.



(a) Walking (b) Holding sphere

Figure 6.5: EMG Signals for similar activities

Furthermore, we plotted two examples with different activities which are *talking in phone* and *typing on keyboard*. As shown in figure 6.6, the EMG signals look very close to each other.

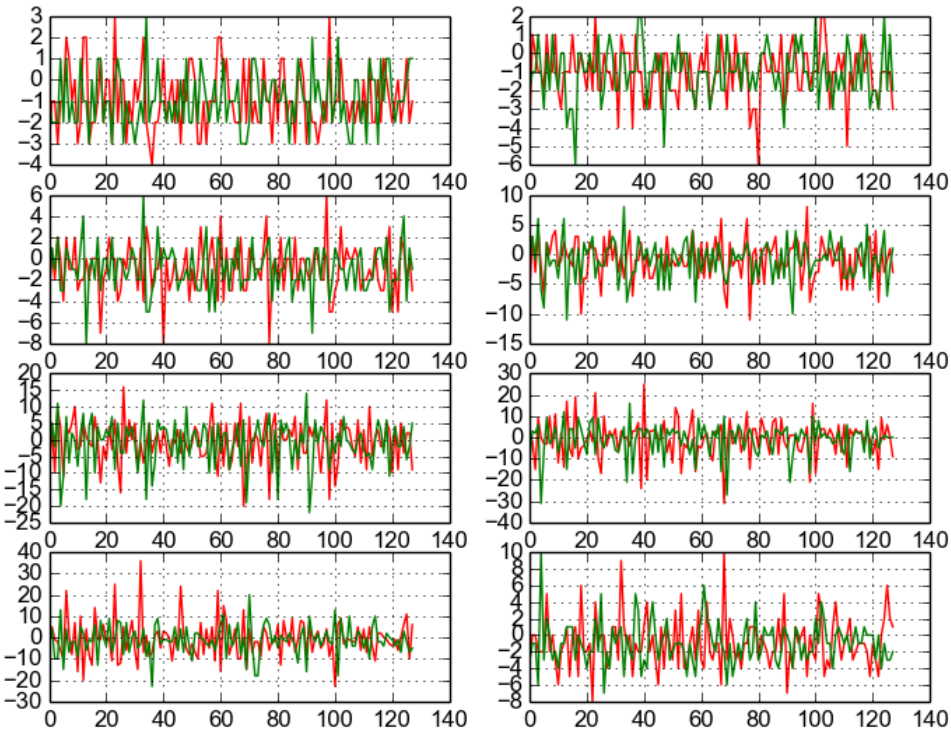


Figure 6.6: EMG Signals: Talking in phone vs Typing on keyboard

When training the network with all activity classes, we got a 17.9% accuracy on the training set and 10.8% on the test set. Raw EMG signals are hard to train. The algorithm cannot even overfit on the training set. We believe that applying sophisticated preprocessing techniques to the raw data can improve the results. Furthermore, normalization might help the algorithm compare data from different test participants.

Chapter 7

Future Work

The neural network model we proposed in chapter 5 can be modified to achieve higher results on the ADL dataset. We didn't try activation functions other than the ReLU function. Changing the network architecture and the dimensions of hidden layers can improve the network or reduce the computing time while keeping the same accuracy.

Furthermore, EMG signals seem to be interesting data to train. Preprocessing the signals to filter noise and decrease redundancy can show better results. In addition, integrating EMG signals with features collected from other sensors can help the network train better.

Appendix

Appendix A

Lists

AAL	Ambient Assisted Living
ADL	Activities of Daily Living
AP	action potential
BN	Batch Normalization
CNN	Convolutional Neural Network
CSV	Comma-Separated Valus
DoF	degree-of-freedom
EMG	Electromyography
HAR	Human Activity Recognition
HCI	Human-Computer Interface
IMU	Inertial Measurement Unit
KNN	K-Nearest Neighbours
LSTM	Long Short-Term Memory
MUAP	motor unit action potential
N.D	Normal Distribution
ReLU	Rectified Linear Unit
SDK	Software Development Kit
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine

List of Figures

List of Tables

Bibliography

[1] Sense and sensor-bility: access mobile device sensors with JavaScript. https://mobiforge.com/design-development/sense-and-sensor-bility-access-mobile-device-sensors-with-javascript, 2012. [Online; accessed 3-August-2017].

[2] A Al-Marakeby. Camera-based wireless sensor networks for e-health. *International Journal of Advanced Research in Computer and Communication Engineering*, 2:4757–4761, 2013.

[3] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *ESANN*, 2013.

[4] Co-Founders at Thalmic Labs. Announcing Raw EMG Data for Developers from the Myo Armband. http://developerblog.myo.com/big-data/, 2014. [Online; accessed 3-August-2017].

[5] Paul Bernhardt. #MyoCraft: EMG in the Bluetooth Protocol. http://developerblog.myo.com/myocraft-emg-in-the-bluetooth-protocol/, 2015. [Online; accessed 3-August-2017].

[6] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[7] Alexandros Andre Chaaraoui, José Ramón Padilla-López, Francisco Javier Ferrández-Pastor, Mario Nieto-Hidalgo, and Francisco Flórez-Revuelta. A vision-based system for intelligent monitoring: Human behaviour analysis and privacy by context. *Sensors*, 14(5):8895–8925, 2014.

[8] Guillaume Chevalier. HAR-stacked residual bidirectional LSTM . https://github.com/guillaume-chevalier/HAR-stacked-residual-bidir-LSTMs, 2017.

[9] Ph.D. & Robynne Boyd Craig Freudenrich. How Your Brain Works. http://www.sensorwiki.org, 2001. [Online; accessed 1-July-2017].

[10] Donato Di Paola, David Naso, Annalisa Milella, Grazia Cicirelli, and Arcangelo Distante. Multi-sensor surveillance of indoor environments by an autonomous mobile robot. *International journal of intelligent systems technologies and applications*, 8(1-4):18–35, 2009.

[11] Anastasios Doulamis, Nikoalos Doulamis, Ilias Kalisperakis, and Christos Stentoumis. A real-time single-camera approach for automatic fall detection. *ISPRS Commission V, Close Range Image measurements Techniques*, 38:207–212, 2010.

[12] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[13] Clinton Fookes, Simon Denman, Ruan Lakemond, David Ryan, Sridha Sridharan, and Massimo Piccardi. Semi-supervised intelligent surveillance system for secure environments. In *Industrial Electronics (ISIE), 2010 IEEE International Symposium on*, pages 2815–2820. IEEE, 2010.

[14] Robert Hartmann, Fadi Al Machot, Philipp Mahr, and Christophe Bobda. Camera-based system for tracking and position estimation of humans. In *Design and Architectures for Signal and Image Processing (DASIP), 2010 Conference on*, pages 62–67. IEEE, 2010.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[16] Jun-Wei Hsieh, Yung-Tai Hsu, Hong-Yuan Mark Liao, and Chih-Chiang Chen. Video-based human movement analysis and its application to surveillance systems. *IEEE Transactions on Multimedia*, 10(3):372–384, 2008.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[18] Maarit Kangas, Antti Konttila, Per Lindgren, Ilkka Winblad, and Timo Jämsä. Comparison of low-complexity fall detection algorithms for body attached accelerometers. *Gait & posture*, 28(2):285–291, 2008.

[19] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[20] Aleksei Nikolaevich Leontiev. Activity and consciousness. *Dialectus*, 2016.

[21] M Marchesi, N Benvenuto, G Orlandi, F Piazza, and A Uncini. Design of multi-layer neural networks with powers-of-two weights. In *Circuits and Systems, 1990., IEEE International Symposium on*, pages 2951–2954. IEEE, 1990.

[22] Qin Ni, Ana Belén García Hernando, and Iván Pau de la Cruz. The elderly's independent living in smart homes: A characterization of activities and sensing infrastructure survey to facilitate services development. *Sensors*, 15(5):11312–11362, 2015.

[23] George Okeyo, Liming Chen, and Hui Wang. Combining ontological and temporal formalisms for composite activity modelling and recognition in smart homes. *Future Generation Computer Systems*, 39:29–43, 2014.

[24] Suneth Ranasinghe, Fadi Al Machot, and Heinrich C Mayr. A review on applications of activity recognition systems with regard to performance and evaluation. *International Journal of Distributed Sensor Networks*, 12(8):1550147716665520, 2016.

[25] Gordon Robertson, Graham Caldwell, Joseph Hamill, Gary Kamen, and Saunders Whittlesey. *Research methods in biomechanics, 2E*. Human Kinetics, 2013.

[26] Bernardino Romera-Paredes, Min SH Aung, and Nadia Bianchi-Berthouze. A one-vs-one classifier ensemble with majority voting for activity recognition. In *ESANN 2013 proceedings, 21st European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 443–448, 2013.

[27] SensorWiki: Accelerometer and Gyroscope. http://www.sensorwiki.org, 2016. [Online; accessed 3-August-2017].

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[29] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[30] Tech Specs: Myo Battery Life, Dimensions, Compatibility, and More. https://www.myo.com/techspecs. [Online; accessed 3-August-2017].

[31] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[32] Stanford University. Convolutional Neural Networks for Visual Recognition. hhttp://cs231n.stanford.edu/2016/, 2016. [Online; accessed March-2017].

[33] Michel Vacher, Dan Istrate, François Portet, Thierry Joubert, Thierry Chevalier, Serge Smidtas, Brigitte Meillon, Benjamin Lecouteux, Mohamed Sehili, Pedro Chahuara, et al. The sweet-home project: Audio technology in smart homes to improve well-being and reliance. In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pages 5291–5294. IEEE, 2011.