DeepConvLSTM on single accelerometer locomotion recognition

Henrik Sjöström

Henrik Sjöström
VT 2017
Bachelor thesis, 15 hp
Supervisor: Lars-Erik Janlert
Examiner: Juan Carlos Nieves Sanchez
Bachelor in Computing Science, 180 hp

Abstract

This project aims to evaluate the deep neural network architecture DeepConvLSTM to classify locomotive human activities using data from a single accelerometer. The evaluation involves comparisons to a simpler convolutional neural network and a hyperparameter evaluation in regards to the networks number of convolutional layers. The benchmark OPPORTUNITY dataset is used for training and evaluation from which triaxial accelerometer data from hips and legs are extracted. The results of the evaluation suggests that DeepConvLSTM outperforms simpler models on most locomotive activity recognition, especially at filtering out unclassified data. Further the results show that DeepConvLSTMs performance improves with a higher number of convolutional layer, but the number of limited by the architectures lack of padding and is compensated by longer training times.

Acknowledgements

I would like to thank my supervisor Lars-Erik Janlert for all the help and feedback. Further I would like to thank Fredrik Nilbrink at RISE Interactive Institute AB and Anna Nordström for their valuable discussion and interest in the project.

Contents

1 Introduction

One of the most serious challenges Europe is facing is the aging society[9]. While providing health care for a generally older population, health promotion and disease prevention activities for the elderly are often forgotten or overlooked. Effective public health strategies could help older adults to remain independent longer, improve their quality of life. For the purpose of optimizing the delivery and success of personalized interventions a means of monitoring and recognizing of physical activities would be beneficial, since it allow the specialists to get a hold of the necessary information at a rapid rate.

Recognizing human activities, such as walking or sitting down, and the context in which they occur from sensor data is at the core of smart assistive technologies[20], which can be classified using a machine learning technique. The raw data gathered from sensors over time takes the shape of a time series which when classified has a class assigned to each step of the series which identifies the activity performed at that moment.

One method of machine learning is deep learning. Deep learning refers to machine learning algorithms (more specifically artificial neural networks) that utilize high number of layers of nonlinear processing units, structured hierarchically, to allow for machine learning in more abstract and complex ways. It is a method that outperforms many conventional methods of classification.

The thesis project relates to the field of time series classification and deep neural networks, and how the deep neural network layout *DeepConvLSTM* performs while classifying locomotive, human activities using a single accelerometer.

1.1 Human Activity Recognition

Human activity recognition is the act of recognizing common human physical activities in a real life settings. It is based on the assumption that specific body movements translate into characteristic sensor signal patterns[17]. In this project we're interested in a single, on-body accelerometer for the purpose of recognizing locomotive activities.

Locomotive activities refer to physical activities that transports a person from one place to another, such as walking, running and jumping.

1.2 Neural network performance evaluation

The performance of a neural networks ability to classify an activity will be tracked with two values: accuracy and F_1 score.

Accuracy refers to the number of exactly matching classifications (true positives) divided by the total number of classifications, which ranges between 0 and 1. With accuracy one can evaluate a networks ability to make correct classifications.

F_1 score refers to the weighted average of the precision and recall, which ranges between 0 and 1. Precision is the number of true positives divided by the total number of positives. Recall is the number of true positives divided by the total number of true positives and false negatives. Much like accuracy, F_1 score allows one to evaluate a networks ability to make correct classifications, but it also takes into consideration the balance between false positives and false negatives.

1.3 Problem Description

The project relates to the field of time series classification and deep neural networks. The goal is to evaluate how the performance of a deep neural network framework that classifies time series of accelerometer data as locomotive, human activities. The framework in question is called DeepConvLSTM[17], a generic deep framework based on convolutional and LSTM recurrent units.

These questions are the ones that will be answered in this report:

- Does the DeepConvLSTM framework still outperform a baseline CNN in regards to accuracy and F_1 score at locomotive activity classification using only a single accelerometer?
- Given the scenario of the previous question, how is the DeepConvLSTM frameworks performance affected by the number of convolutional layers?

A baseline CNN refers to a deep neural network similarly structured to DeepConvLSTM that lacks recurrent layers, therefore representing a traditional, convolutional network.

1.4 Application areas

There are numerous possible applications for classification of physical activities. One example would be assessing physical activity and sedentary behaviour to fight obesity and related health risks. Until recent years the knowledge and thus the recommendations on physical activity have relied mainly on data obtained from surveys[5] but with modern phones containing accelerometers there's an opportunity to get reliable data, leading to better tracking and recommendations related to physical health.

Another possible usage for the data achieved through the classification of the physical activities would be retroactively tracking the effects of different physical lifestyles, which could be used to further the field of physical health.

1.5 Ethics

While dealing with utilization and gathering of sensor data ethical concerns for privacy and consent has to be considered. The act of classifying data itself adheres to these concerns since classifying data is not dependant on whose data it is. The responsibility falls instead on the one operating the network. In this project the ethically of the classified data provided for training and testing has been considered. Only the research leader will have access to social security numbers from where the identity can be traced from individual data. No individual data will be presented.

2 Background

In order to solve the task of classifying time series of accelerometer readings into categories of human, locomotive activities using neural networks, the field of deep neural networks needs to be explored. The act of training an artificial neural network, network architectures such as convolutional neural networks and recurrent neural networks, and the state of the art of this type of classification will be the topics of this section.

2.1 Training a Neural Network

The main feature of a neural network from the perspective of machine learning is its ability to learn from training. Rather than setting hard coded values into the network, the weights of a neural networks interconnections are updated through the process of training. Training a neural network is divided into two main approaches[24, pp. 19–20]: Supervised learning and unsupervised learning.

Supervised learning is an approach in which the set upon which the networks is trained includes both the data and the desired results. In the case of a network that is applying classification, supervised learning would imply that the network is trained using data that is accompanied by predetermined classification. Naturally, only during the training is the desired results required, since determining the result is the purpose of the network. [24, pp. 19–21]

Unsupervised learning is an approach in which the set upon which the networks is trained does not include the desired results. Unsupervised learning has been used for problems such as clustering[14], when the categorization (number of categories) is unclear beforehand. [24, pp. 19–21]

2.1.1 Backpropagation

On neural networks that consists solely of an input and an output layer, under a supervised training process, updating weights along the training process is as simple as calculating the error between the computed output and the desired output. On deep networks this method does not hold up due to there not being a direct connection between the input and output layers. The error needs to be propagated backwards through the network in order to update the weights of all layers. This method is known as backpropagation.

The basic idea of backpropagation is to move the gradient information successively from the output layer to the input layer. The resulting weight update is computed as a scaled step in the opposite direction of the gradient, in other words the negative derivative is multiplied by a constant value known as the learning rate. [21]

2.1.2 Overfitting

Overfitting is the phenomenon where a models performance suffers due to the network being to complex for the task at hand. Overfitting can be prevented by adding additional information to the network, which is known as regularization.

There are multiple methods of regularization. L_2-regularization is a method in which every weight in the network is treated as $\frac{1}{2}\lambda w^2$ during error calculations, where w is the weight and λ is the strength of the regularization. L_1-regularization is a similar method were the weights are treated as $\lambda|w|$ instead.

Another method of regularization is called ”dropout”, a method that has been shown to be very effective on convolutional networks used for classification[15]. Dropout consists of setting the output of hidden neurons to zero with given probability. The neurons which are “dropped out” in this way do not contribute to the forward pass and do not participate in backpropagation. This random dropping of neurons forces the neurons in the network to not rely on a significant other neuron, thereby making the network as a whole to learn more robust features in the data. To compensate for the removed neurons the output weights of the neurons affected by dropout is multiplied with the same given probability.

Early stopping and overtraining

One network behaviour that occurs during the learning process is that the network goes beyond learning the gross structure of the data and starts learning the fine structure that is generated by noise in the training data. This harms the models performance. To avoid this behaviour a network the training should be stopped before the network starts finding learning the patterns of the noise. This act of stopping the training before complete convergence is called *early stopping*.

In order to determine the point at which the early stopping is to occur is one uses a validation set. This set is created by partitioning the training data into two separate sets: a training set and a validation set. The training set is used to train the neural network as normal, while the validation set is used to periodically test the how well your model has been trained. The validation set does not share any data with the training set, and the validation set is never used for actual training. When the testing on the validation set starts giving consistently worse results the system has very likely started to get overfit and an early stopping should be performed.**??**

Due to the existence of early stopping one does not have to specify for how long a network is to be trained. One can instead train the network indefinitely until an early stopping is reached.

This behaviour is by some referred to as overtraining[12], rather then considered a part of overfitting.

2.2 Convolutional Networks

Feature extraction is the process of reducing the size of raw input data[1] to a neural network without sacrificing the accuracy of the data. The intention of the process is essentially to remove redundancies and generalize the raw data into a more compact form, called "features". Common approaches to feature extraction is altering the raw datas space dimensionality and the extraction is generally considered a form of preprocessing. [10, pp. 2–4]. One method of feature extraction is the usage of deep convolutional neural networks.

A convolutional neural network (CNN) is a type of feed-forward network that makes the explicit assumption that its inputs are images, from which features are extracted . While a system can classify structures using raw data it has been shown that using the derived features often leads to higher performance for classification of images, speech, and time series. [6, pp. 276–278]

The architecture of a CNN consits of a filter stage (a convolutional layer), followed by pooling layer and a ReLU non-linearity). [18]

2.2.1 The convolutional layer

The convolutional layer is the core of convolutional neural network. This layer attempts to find features in the image by using sliding filters (or kernels) across the image and computes the dot products between the entries of the filters and the image at the position. Each filter will produce activation map of the responses it got at every position on the image. Ideally these filters will pick up on some type of visual feature in the image, like an edge with a given orientation. Stacking multiple layers will instead allow for the filters to eventually recognize more complicated pattern in the image. The layer as a whole returns the filters two 2-dimensional maps packed as a 3-dimensional volume. [3]

2.2.2 The pooling layer

The pooling layers' purpose is to progressively reduce the spatial size of the volume produced by the convolutions layer in order to reduce the amount of data and computations performed in the network. It is common to insert a pooling layer between successive convolutional layers. The layer resizes the volume using the MAX operation. Each map in the volume is divided into sub-maps, each entry in the map only belonging to a single sub-map. From these sub-maps the highest value, selected by the MAX operator, is returned to form a new, smaller map. See an example of this in Figure 1. In addition to max pooling, the pooling units can also perform other functions, such as average pooling or even L2-norm pooling.[3]

2.2.3 ReLU non-linearity

ReLU (Rectified Linear Units) is an activation function that formats a neuron's output f as $f(x) = max(0,x)$. Convolutional neural networks using ReLU has been shown to train several times faster than ones using tanh units ($f(x) = tanh(x)$). [15]

[1]data that has not been processed since it was generated

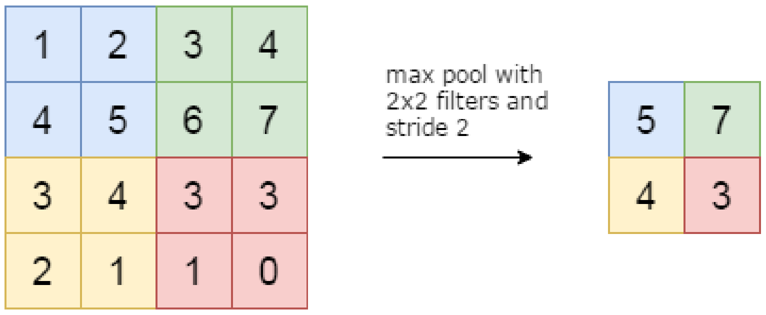

Figure 1: A 4x4 map resized in a max pooling with a filter size of 2x2 and a stride of 2.

2.3 Recurrent Networks

As opposed to a feed-forward network, recurrent networks have the ability to make use of previous data by being recurrent. In addition to the common input and output layer a recurrent network contains hidden neurons in between the input and output that are not strictly directed straight towards the output. Instead these hidden neurons can send signals towards any other neuron, including themselves. This allows the network to recurrently consider information from previous considerations. This ability to perceive back in time is useful when operating on sequences, like time series or sentences.

The shape of successively layered recurrent networks is very flexible and can be categorized using the terms: one-to-many, many-to-one and many-to-many (see Figure 2). One-to-many refers to layouts where an input gets recurrently operated on. This allows for extraction of multiple features that are order dependent, which for example is useful for image captioning. Many-to-one refers to layouts where a series of inputs gets recurrently operated on to produce a single output, such as determining the sentiment of a sentence. Many-to-many refers to layouts where a series of inputs gets recurrently operated on to produce a series of outputs, which for example is useful when translating a sentence to a different language.

Much like other neural networks recurrent neural networks depend on backpropagation and gradient decent to classify its input. Using regular backpropagation is problematic since each layer of the backpropagation is defined as its relation with the next layer and recurrent networks allow for cyclical and self referential structures. This would lead the backpropagation to an infinite loop. Recurrent networks instead rely on an extension of backpropagation called backpropagation through time (BPTT)[1]. BPTT solves this problem by considering more than one step in the recurrent system at the same time, that way catching the cycles. It does this by unfolding the network into the shape of a feed-forward network and then operates a normal backpropagation on the unfolded network in sequential order.

One problem with training recurrent networks using BPTT is that unfolding highly cyclical networks can result in very deep feed-forward network, which causes a problem known as the vanishing gradient problem. The source of the problem is that in a large system each piece of information passing through the system will pass by many points of multiplication. Values below 1 will over time head towards 0 and detail in the data will be lost as computers will struggle to represent the smaller values with ample detail. The result of this is that memory gets harder and harder to store between steps, making long term memory difficult.[1]

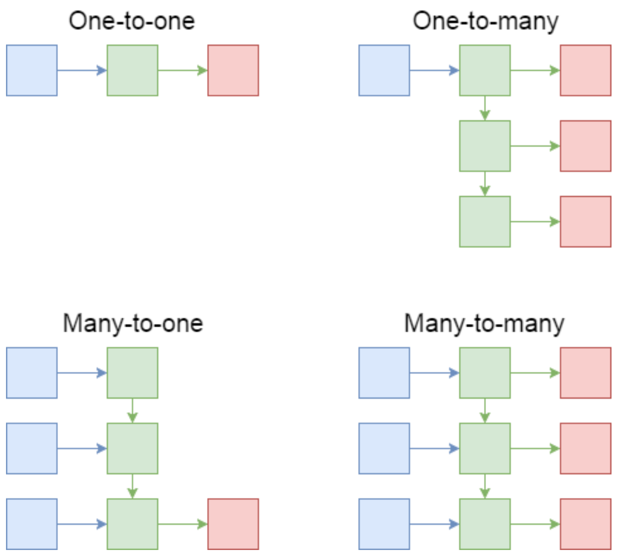

Figure 2: Examples of recurrent networks following different styles. One-to-one essentially being a normal neural network. One-to-many, for example used for image captioning. Many-to-one, for example used for sentence sentiment classification. Many-to-many, for example used for machine translation [4]

2.3.1 LSTM

Long short-term memory units (LSTMs) are a variation of the normal recurrent network, originally proposed as a solution to the vanishing gradient problem[11]. LSTM does this by maintaining a more constant error between steps which allows the network to maintain memory of previous operations over larger number of steps. LSTM manages this by storing additional information in a gated cell which is passed along each step. This cell can be stored in, written to, or read from by neurons in the network and acts much like a computer's memory. The neurons interacts with the cell via gates that can open and close. The gates act on the signals they receive, and similar to the neural network's neurons, they block or pass on information based on its strength and import, which they filter with their own sets of weights. Those weights are updated during the learning process. [1].

The gated cell contains three gates: the write/input gate, the read/output gate and the keep/forget gate. The input gate controls the extent to which a new value flows into the memory, the forget gate controls the extent to which a value remains in memory and the output gate controls the extent to which the value in memory is used to compute the output of the block.

2.4 Related work on deep networks for activity recognition

Deep neural networks, specifically convolutional neural networks, have been shown to have great potential in the art of identifying the various major patterns of human activity recognition data. With the addition of stacking multiple convolutional and pooling layers there's also the possibility for deriving less major patterns when the more primitive ones join together[25]. Convolutional neural networks also provide the ability to compute data in its raw form, without any preprocessing, since the convolutional network provides internal feature extraction. Experiments on several benchmark datasets have been performed, including the Skoda, OPPORTUNITY and Actitracker datasets with results showing that the convolutional neural networks approach outperforms other methods, such as restricted boltzmann machine and principal component analysis[27].

Video activity recognition is a field closely related to wearable activity recognition using, which this report touches upon. While the two fields aren't identical, they share properties such as the temporal dimension to the input data and the periodic shape of the activities that are to be classified. While video classification is commonly connected to convolutional networks, it has been shown that combining the convolutional network with LSTM provides significant performance improvements[26].

The field of combining convolutional neural networks with LSTM networks is fairly unexplored. A combination of the networks, called a long-term recurrent convolutional network (LRCN), has been proposed[8]. The LRCN architecture is suitable for large-scale visual prediction and classification and is end-to-end trainable. In the combination the CNN part is used to learn temporal dynamics and extract features while the LSTM adds the ability to learning long-term dependencies. LRCN has not yet been applied to fields other then video learning, but it does outperform competitors within the given field and should be applicable to other fields of classification that involve temporal dynamics.

Deep belief networks is an alternate network model to the convolutional network. It is a generative probabilistic network model composed of one visible layer and many hidden layers[16]. Deep belief networks can be efficiently trained using greedy layerwise training and have shown to perform well at human activity recognition[19].

2.4.1 DeepConvLSTM

DeepConvLSTM[17] is a generic deep neural network framework for activity recognition based on convolutional and LSTM recurrent units. It is suitable for multimodal wearable sensors, can perform sensor fusion naturally, does not require expert knowledge in designing features and explicitly models the temporal dynamics of feature activations. What sets it apart from other convolutional networks is its addition of LSTM recurrent units. It is somewhat similar to the LRCN architecture, but it differs in the dimension on which the convolutional layers map.

DeepConvLSTM includes convolutional, recurrent and softmax layers. The layout of the framework can be seen in Figure 3. It starts by passing a short time series extracted from the sensor data through four convolutional layers, processing the input along the axis representing time with a filter size of 5 time steps and a stride of 1. The computation of the feature maps uses 64 ReLUs per layer, whose non-linear function is defined as $\sigma(x) = max(0,x)$. They're followed by two recurrent LSTM layers. As input these layers use the elements of all of the feature maps at the last convolutional layer. Each LSTM layer possesses 128 LSTM units. The activation of the recurrent units is computed using the hyperbolic tangent function. The recurrent layers are followed by a dense layer with a softmax activation function yielding a class probability distribution for every single time step.

The framework has been shown to outperform other CNN solutions on the OPPORTUNITY challenge, the challenge for which the OPPORTUNITY dataset was originally created, in regards to F_1 score. Further, it also outperformed all of the submissions to the challenge.[17]

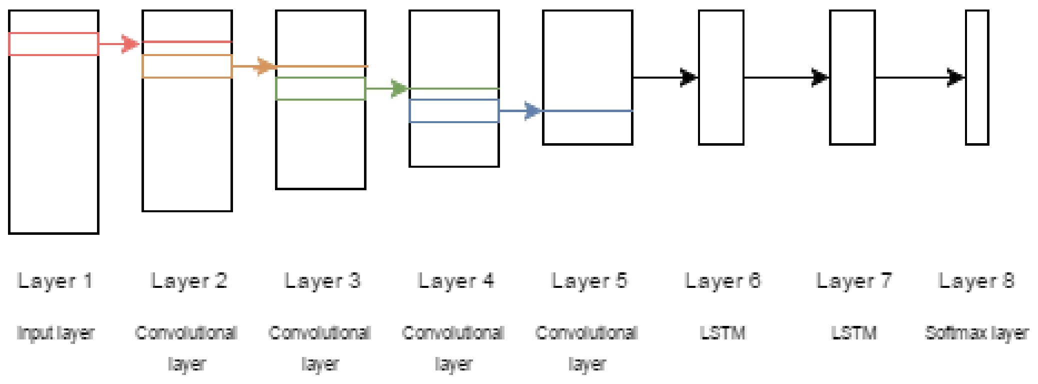

Figure 3: Architecture of the DeepConvLSTM framework as presented in its original article[17]. From the left, the signals coming from the wearable sensors are processed by four convolutional layers, followed by two dense layers and ending with a softmax logistic regression output layer.

3 Methods

In this section the implementation, training and evaluation of the neural networks, used for the activity classification tasks posed by the questions that this report is to answer, will be divulged.

3.1 Implementation

In order to answer the questions posed by this report the DeepConvLSTM framework will need to be evaluated, and consequently implemented. Additionally a baseline CNN network will be implemented as a basis of comparison for the DeepConvLSTM in order to answer the first question posed by this report:

- Does the DeepConvLSTM framework still outperform a baseline CNN in regards to accuracy and F_1 score at locomotive activity classification using only a single accelerometer?

The neural networks here described are implemented in Theano using Lasagne. Theano is a Python library that aims to improve both execution time and development time of machine learning applications, especially deep learning algorithms[7]. Lasagne is a lightweight library to build and train neural networks.

The decision to use Theano and Lasagne was made due to Theano having shown to perform better then most competitors in in regards to shorter training times and better classification accuracy[13].

3.1.1 DeepConvLSTM

The implementation of DeepConvLSTM follows the architecture previously described in the *Background* section. The layout of the network can be seen in Table 1. The number of convolutional layers can vary and is selected at runtime. Lasagne's convolutional layers include ReLU activation by default, which eliminates the need to add it as a separate layer after each convolutional layer. The convolutional layers are not followed by any pooling layers nor do they contain any padding[1] , in accordance with the architecture it is based on. The connection between the LSTM layers and the dense softmax layer is a many-to-one connection where only the last step in the sequence has a direct connection to the output layer. The rest of the sequence affects the layer only through the LSTM units passed within the LSTM layer.

Table 1 Model of the DeepConvLSTM implementation built using the library Lasagne. The model contains an input layer, a varied number of convolutional layers, two LSTM layers and dense softmax output layer. *C* refers to the number of axis/channels in the input data. *D* refers to the length of the input sequence. *N* refers to the number of different classes the sequence can be classified as.

Input *shape: CxD*
Convolutional *filter shape: 1x5, stride: 1, number of filters: 64*
...
LSTM *number of units: 128*
Dropout *probability: 0.5*
LSTM *number of units: 128*
Dropout *probability: 0.5*
Softmax Output *size: N*

[1]pad the input volume with zeros around the borders to maintain the size of the data

3.1.2 Baseline CNN network

The implementation of the baseline CNN is a near copy of the DeepConvLSTM implementation detailed previously+ . The main difference between the models is that the baseline CNN does not possess any recurrent layers. These layers have been replaced by dense layers of similar size. The layout of the network can be seen in Table 2. The number of convolutional layers can vary and is determined at runtime. Lasagne's convolutional layers include ReLU activation by default, which eliminates the need to add it as a separate layer after each convolutional layer. The convolutional layers are not followed by any pooling layers nor do they contain any padding[2], in accordance with the architecture it is based on.

Table 2 Model of a baseline CNN network implementation built using the library Lasagne. The model contains an input layer, a varied number of convolutional layers, two dense layers and dense softmax output layer. *C* refers to the number of axis/channels in the input data. *D* refers to the length of the input sequence. *N* refers to the number of different classes the sequence can be classified as.

Input *shape: CxD*
Convolutional *filter shape: 1 x 5, stride: 1, number of filters: 64*
...
Dense *size: 128*
Dense *size: 128*
Softmax Output *size: 5*

[2]pad the input volume with zeros around the borders to maintain the size of the data

3.2 Dataset

The frameworks are trained and evaluated on a benchmark dataset called the OPPORTUNITY activity recognition dataset[2].

The OPPORTUNITY activity recognition dataset[22] is a set of physical activities collected from 72 sensors, 12 of which are accelerometers, with over 25 hours of data. The data is stored as a time series with a frequency of 30 Hz. Each sample of the data is comprised of 113 real valued attributes, which are scaled to [0,1], as well as classifications in multiple categories. The accelerometer sensors are triaxial, thereby possessing three real values each. The dataset is divided into three different subjects whom wore the sensors when the data was generated.

The only category addressed in this report is *locomotion*. Locomotion includes 5 classes: Stand, Sit, Walk, Lie, and *Null*. The classes are explored in Table 3.

Table 3 Class labels for modes of locomotion recorded during the ADL runs in the OPPORTUNITY activity recognition dataset, including a description of the activity and the number of instances of the activity in the dataset. An instance refers to an uninterrupted sequence of a specific class, not to be confused with a number of time steps.

Class	**Description**	**Instances**
Stand	Standing upright without moving.	1093
Sit	Sitting in a chair at a table.	1095
Walk	Traversing upright on two legs.	90
Lie	Lie down in a lazy chair.	40
Null	Unclassified activity.	-

In this report the data from all subjects will be used together and the will include the *Null* class, since the goal of this analysis is to provide a reference performance rather then aiming to obtaining a high accurate. The report will also focus on the data belonging to a single sensor at a time. Since locomotive activities are the focus of this report and locomotion unusually involve the legs, the sensors that will be focused upon are the ones connected to the subjects legs and hips. Three sensors will therefore be included: hips (HIP), upper right knee (RKN$^\wedge$), and lower right knee (RKN$_{\_}$). See Figure 4 for a detailed look at the positions of the sensors on the subject. In this paper, the complete dataset will be split into subsets belonging to single sensors and referred to by the name of the sensor.

3.3 Training

The neural networks are trained using supervised training with a setup imitating the training used in its original report[17]. The network parameters are optimized by minimizing the cross-entropy loss function using mini-batch gradient descent with the RMSProp update rule. RMSProp is a gradient descent optimization algorithm and an adaptive learning rate method proposed by Geoff Hinton[23], commonly used for training recurrent neural networks. The training and testing data are segmented on mini-batches of 100 classifications and the accumulated gradient for the parameters is computed after every mini-batch rather then after every classification. The networks are trained with a learning rate of 0.001 and

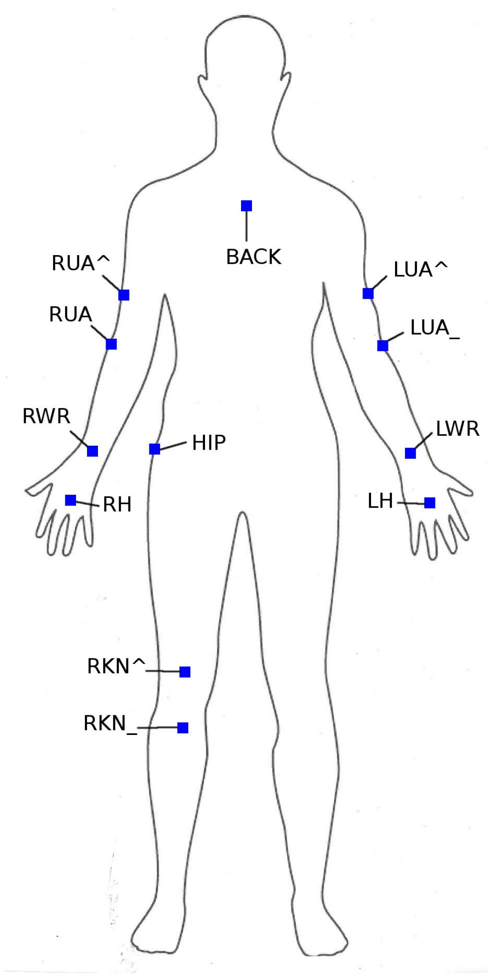

Figure 4: The locations of the on-body 12 Bluetooth acceleration sensors in the OPPORTUNITY dataset. [2]

a decay factor of 0.9. The networks weights are randomly orthogonally initialized before training.

The data subsets derived from the OPPORTUNITY activity recognition dataset is initially split into two sets: a training set and a testing set. The testing set consisted of the data gathered during the last two sessions for subject 2 and 3. The rest of the data belonged to the training set. The tailing fifth of the training data was used as a validation set. The training runs for an indefinite number of epochs with an early stopping patience of 10 epochs, after which the weights from the epoch with the lowest validation error are tested and exported.

3.3.1 Sliding window

While a network posed with the challenge of classifying a human activity could consider an indefinite amount of time steps into the past when classifying, considering steps very far into the past might not help with the process of classification. Due to the periodic nature of human activities, a sliding window approach will be used to give each steps classification a fitting amount of past information.

A sliding window, in the context of sequence classification, is the act of sliding a window of a given width across the sequence and only passing the subset of data defined by window to the network, thereby dividing the total data into smaller sequences. When dividing a series into smaller sequences with a sliding window, there are two parameters that needs to be set: the width of the window and the step size of the window. The width of the window defines the length of each sequence produces. The step size defines the number of steps along the series the window moves between each sequence production. This means that if the step size is smaller then the window size there will be overlapping between the sequences and if the step size is larger then the window size then no value will be included in multiple sequences.

The width of the sliding window on classifications using the OPPORTUNITY dataset, scaling between 400ms and 2750ms, has been shown to not affect performance significantly[17] and will therefore be set to 1000ms for training and evaluation. The step size will be set to half the size of the sliding window (500ms).

3.4 Evaluation

The purpose of the evaluation is to answer the question posed by this report:

- Does the DeepConvLSTM framework still outperform a baseline CNN in regards to accuracy and F_1 score at locomotive activity classification using only a single accelerometer?
- Given the scenario of the previous question, how is the DeepConvLSTM frameworks performance affected by the number of convolutional layers?

The two questions will be tackled independently from one another, identified as question 1 and question 2 (as in the order they were asked).

3.4.1 Question 1

- Does the DeepConvLSTM framework still outperform a baseline CNN in regards to accuracy and F_1 score at locomotive activity classification using only a single accelerometer?

In order to answer this question the models (DeepConvLSTM and baseline CNN) is both be trained and tested under identical circumstances, after which their accuracy and F_1 score will be compared. Using the dataset and training procedure detailed previously, and using the network models with four convolutional layer (like in DeepConvLSTM original model), these models architectures are to be compared and presented in a table.

3.4.2 Question 2

- Given the scenario of the previous question, how is the DeepConvLSTM frameworks performance affected by the number of convolutional layers?

In order to answer this question the DeepConvLSTM model is trained and tested using the dataset and training procedure detailed previously, after which the accuracy and F_1 score of the model will be documented. This process is done successively over multiple DeepConvLSTM models with numbers of convolutional layers ranging between one and five. The scores from these are to be compared and presented in a table.

4 Results

In this section results from the evaluation are presented. The results will be divided between the two questions the evaluation aims to answer. The question are referred to as question 1 and 2, just as in the *Methods* section. Details on methods of training and evaluation is described in the *Methods* section.

4.1 Results for question 1

- Does the DeepConvLSTM framework still outperform a baseline CNN in regards to accuracy and F_1 score at locomotive activity classification using only a single accelerometer?

Question 1 involves the two network models (DeepConvLSTM and baseline CNN) that were to be compared and the three subsets of the OPPORTUNITY dataset (HIP, RKNˆ, RKN_) in regards of classifying five classes of locomotive activities (Stand, Walk, Sit, Lie, *Null*).

Table 4 The results of evaluating a four convolutional layered DeepConvLSTM and baseline CNN on three subsets of the OPPORTUNITY dataset. The table shows the accuracy (upper number) and F_1 score (lower number) of the models when trained and tested on the subsets. The superior scores in each subset are bold.

Dataset				
Network		HIP	RKNˆ	RKN_
	Baseline CNN	0.467	0.612	**0.583**
		-	0.551	**0.533**
	DeepConvLSTM	**0.506**	**0.684**	0.565
		0.444	**0.678**	0.524

As can be seen in Table 4, DeepConvLSTM managed to to outperform the baseline CNN model on the HIP subset in both accuracy and F_1 score, with 50.6% correct classifications to 46.7%, and on the RKNˆsubset with 68.4% correct classifications to 61.2%. The baseline CNN model does manage to outperform DeepConvLSTM on one subset, the RKN_, with 58.3% correct classifications to 56.5%. The F_1 score of the baseline CNN model on the HIP subset is not defined since one of the classes in the classification lacks any representation (see Table 5a). Under those circumstances calculating the F_1 score is ill-defined.

Table 5 Confusion matrices from the tests of baseline CNN and DeepConvLSTM with four convolutional layers on subsets of the OPPORTUNITY dataset. The numbers represent the number of sliding window sequences classified as a given activity against its actual activity. The number is not to be confused with the number of time steps nor the number of instances of a given activity. The numbers marked with bold are correctly classified sequences.

(a) The baseline CNN on the HIP subset.

		Predicted Activity				
		Null	Stand	Walk	Sit	Lie
Actual Activity	*Null*	**233**	1072	1337	0	16
	Stand	11	**3326**	436	0	16
	Walk	63	548	**2003**	0	0
	Sit	1	2468	53	**0**	0
	Lie	81	302	54	0	**80**

(b) DeepConvLSTM on the HIP subset.

		Predicted Activity				
		Null	Stand	Walk	Sit	Lie
Actual Activity	*Null*	**800**	841	734	251	32
	Stand	105	**3136**	251	272	25
	Walk	361	901	**1298**	53	1
	Sit	25	1819	54	**622**	2
	Lie	239	7	0	1	**270**

(c) The baseline CNN on the RKN^subset.

		Predicted Activity				
		Null	Stand	Walk	Sit	Lie
Actual Activity	*Null*	**152**	1209	1125	147	25
	Stand	12	**3136**	582	58	1
	Walk	3	976	**1624**	11	0
	Sit	38	13	10	**2461**	0
	Lie	245	0	235	9	**28**

(d) DeepConvLSTM on the RKN^subset.

		Predicted Activity				
		Null	Stand	Walk	Sit	Lie
Actual Activity	*Null*	**1139**	760	616	123	20
	Stand	269	**2870**	624	26	0
	Walk	296	675	**1634**	9	0
	Sit	42	95	4	**2381**	0
	Lie	263	3	0	1	**250**

(e) The baseline CNN on the RKN$_{\_}$ subset.

		Predicted Activity				
		Null	Stand	Walk	Sit	Lie
Actual Activity	*Null*	**191**	1125	1093	208	41
	Stand	6	**2773**	792	218	0
	Walk	62	787	**1668**	97	0
	Sit	1	220	66	**2233**	2
	Lie	0	9	96	221	**191**

(f) DeepConvLSTM on the RKN$_{\_}$ subset.

		Predicted Activity				
		Null	Stand	Walk	Sit	Lie
Actual Activity	*Null*	**219**	1286	992	133	28
	Stand	8	**2932**	589	260	0
	Walk	43	961	**1540**	70	0
	Sit	4	575	55	**1887**	1
	Lie	7	13	230	6	**261**

4.2 Results of question 2

- Given the scenario of the previous question, how is the DeepConvLSTM frameworks performance affected by the number of convolutional layers?

Question 2 involves the network models DeepConvLSTM and its performance (accuracy and F_1 score) on the three subsets of the OPPORTUNITY dataset (HIP, RKNˆ, RKN_) with varying numbers of convolutional layers.

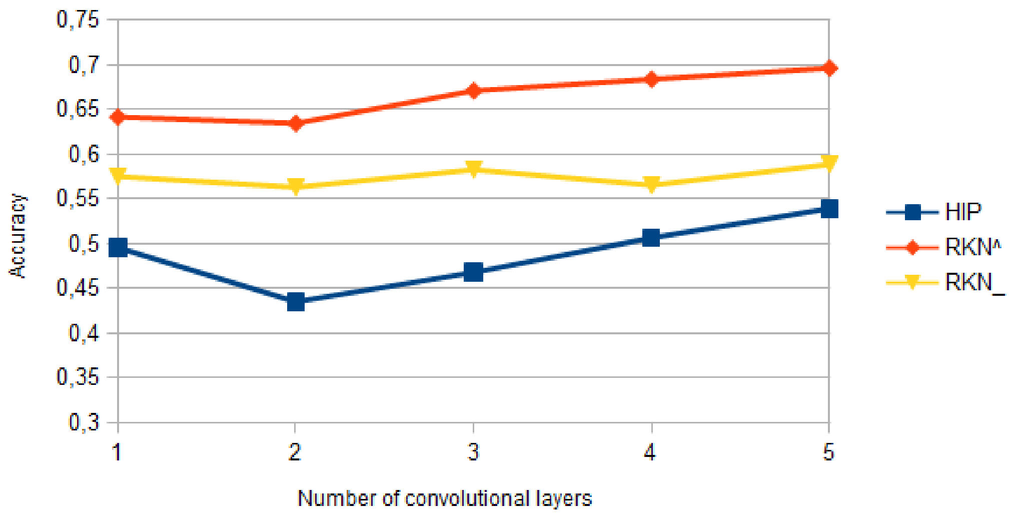

Figure 5: Accuracy of DeepConvLSTM on subsets of the OPPORTUNITY dataset with different numbers of convolutional layers.

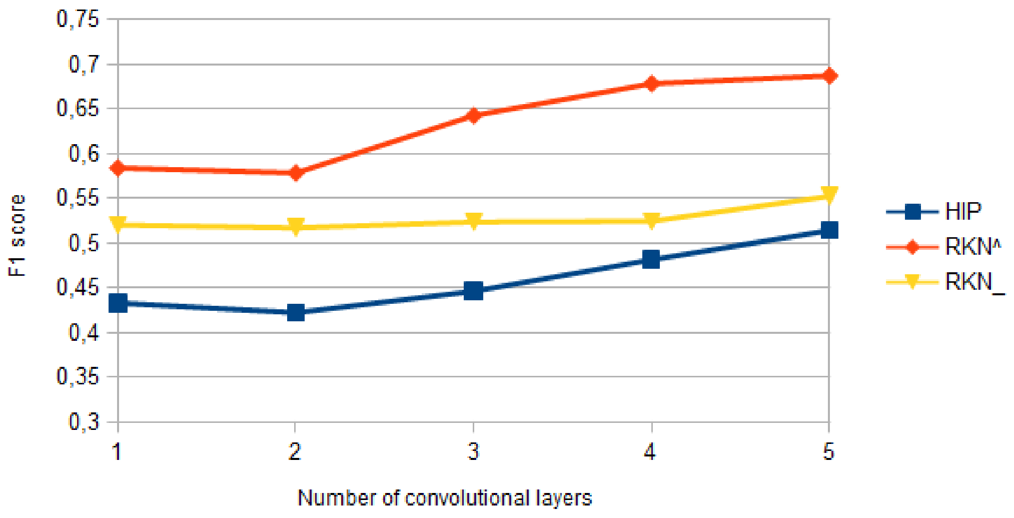

Figure 6: F_1 score of DeepConvLSTM on subsets of the OPPORTUNITY dataset with different numbers of convolutional layers.

See Figure 5 and 6 for the results of the evaluation. In regards to both accuracy and F_1 score, the best performance on each subset is reach with the models using five convolutional layers, the highest number of convolutional layers tracked. Excluding the step from one to two convolutional layers, the performance on the HIP and RKNˆsubsets constantly improves as more convolutional layers are added, with accuracy improving by around 2-3% with each added layer. The RKN$_-$ is comparatively unaffected by the increasing number of convolutional layers, but too has its best performance at the highest number of layers.

5 Discussion

In this section the results of the project will be discussed. This will target problems that occurred during training and evaluation, and conclusions reached about the question raised by this report. This section is concluded with a summary of the most important conclusions.

5.1 Training

The training of the models went without any major problems. The number of epochs it took for the networks to stop ranged from 5 to 40, excluding the epochs of patience. The number it took was generally higher on systems with more convolutional layers, but it did not seem to differ much which model that was trained. The order in which the mini-batches were ordered during training was randomly set without the use of a preset seed for the random generation. Giving all training processes the same seed might make for a more uniformed comparison between the models.

The performance improvements between epochs seemed very chaotic in nature. The system could go for long periods of no improvement to suddenly seeing a huge spike of improvement over multiple epochs. The fairly high patience of the early stopping is implemented to catch these spikes, but the fact that they arrive so spontaneously leaves the possibility that each trained variation of the models could possibly be far from perfectly trained. This is not necessarily a problem since all of the models were trained under the same circumstances and the main purpose was to compare the models, rather then to find the performance of a perfectly trained network. There is the possibility to further train an already weighted network, and preliminary attempts did show that the networks could see slight performance improvements with further training. This would be the equivalence of increasing the patience of the system.

5.2 Conclusions

Here I'll attempt to answer the questions posed by this report using the results from the Results section.

5.2.1 Question 1

In the results (see Table 4), neither of the two models consistently outperform the other. Comparing the models ability to classify individual classes, which is displayed in the confusion matrices of the trained and tested models, can give more insight into the nature of this inconsistency.

In the models performances on the HIP subset, seen in Table 5a and 5b respectively, one can see that DeepConvLSTM outperforms the baseline CNN at correctly classifying the classes *Null*, Sit and Lie, while being outperformed on Walk. Most of the misses by the baseline CNN comes from mistaking Sit and Lie as Stand. From the sensors perspective mistaking these would make sense since during all of them the sensor, located on the hip, would be stationary, making them seem very similar. DeepConvLSTM on the other hand does not seem to struggle as much from this problem, which could either be a result of the baseline CNN being undertrained or DeepConvLSTM being able to find the existence of (or lack of) minor temporal patterns in the features that the baseline CNN could not find, such as differences in breathing and twitching.

The baseline CNNs performances on the RKN^subset, seen in Table 5c and 5d respectively, does not struggle at classifying Sit like in the HIP subset, but it is still outperformed by DeepConvLSTM at classifying Lie and *Null*.

While the baseline CNN does outperform DeepConvLSTMs performance on the RKN_ subset in regards to accuracy and F_1 score the gap between the scores is far lower then that of the other comparisons (within 2%) and is likely within the realm of random variation due to the random setup of the training process. Assuming this would imply that the two models performance at correctly classifying the classes would be fairly even, which shows in Table 5e and 5f where the difference in confusion is very similar. The largest difference between them is in Sit and Lie class where the baseline CNN respectively DeepConvLSTM outperforms the other.

A common trend with all subsets is that DeepConvLSTM always outperforms the baseline CNN at classifying the class *Null*, the unclassified activities, and on the HIP and RKN^subsets by a large margin. While the weighted nature of the F_1 score counts all *Null* activities as one unanimous class they could be considered a group of different activities. If the purpose of the network is to perform well while scaled up to account for more unique activities, the ability to differentiate the tested datasets activities from non-included activities would be ever more important. An ability that DeepConvLSTM succeeds the baseline CNN at.

5.2.2 Question 2

As Figure 6 shows, the performance of the models tends to improve with the increasing number of convolutional layers. The improvement with each layer seems to be somewhat consistent and is likely to continue past the tested parameters. Other convolutional neural networks, such as GoogLeNet and ResNet, have used more then 20 convolutional layers in their models with beneficial results so deeper networks are not unheard of. With all that, the improvements must eventually plateau and training times increase with deeper networks. Due to this one will have to decide on a balance between accuracy and training times.

Using the models presented in this report one cannot stack more and more convolutional layers indefinitely. Since the convolutional layers lack padding[1] the size of each layers output gets progressively smaller until it's too small to filter, and since the filters are mapped across the temporal axis (rather then the sensor channels) time steps will be compressed with each feature extraction until they're too few. With the setup used in this implementation (1x5 filter and 30 time steps) the network will lose four time steps with each feature extraction for

[1]pad the input volume with zeros around the borders to maintain the size of the data

a maximum of seven convolutional layers. Using DeepConvLSTM, while there are benefits of continuously stacking convolutional layers, such as extraction more complex features, the LSTM layers following them will have less temporal information to deal with which makes them all the more pointless. Since both the convolutional layers and the LSTM layers focus on finding temporal patterns they'll naturally be needing the same data and a balance needs to be struck between the two. Simply adding padding in order to avoid this issue has problems of its own since without any max pooling (which DeepConvLSTM does not include) this padding will be passed on to the LSTMs and might give empty information to the edges of the LSTM layers.

5.3 Future work

While the effects of additional convolutional layers has been explored by this report, many other hyperparameters have not been touched, such as: number of LSTM layers, number of LSTM units and size of the filters in the convolutional layers. Like previously mentioned, the balance between number of convolutional layers and LSTM layers might have an effect on the performance and as such these hyperparameters would need further investigation.

Since the size of the sliding window has been shown to have little effect on the performance of the model and it possesses recurrent properties, the model might be a good candidate for variable sized input. This does bump into the padding problem once again and it is not covered by its current architecture, but it justifies further investigation.

5.4 Summary

- The DeepConvLSTM architecture performs equaly or better then a baseline CNN at locomotive activity recognition on single accelerometer data from the OPPORTUNITY dataset.
- The DeepConvLSTM architecture outperforms a baseline CNN at filtering out uncategorized locomotive activities from a single accelerometer dataset.
- The DeepConvLSTM architectures ability to classify locomotive activities improves with higher number of convolutional layers.

References

[1] A Beginner's Guide to Recurrent Networks and LSTMs. https://deeplearning4j.org/lstm (visited 2017-05-11).

[2] Activity Recognition Challenge - Dataset. http://www.opportunity-project.eu/challengeDataset/ (visited 2017-05-20).

[3] CS231n Convolutional Neural Networks for Visual Recognition. https://cs231n.github.io/convolutional-networks/ (visited 2017-05-10).

[4] Notes for CS231n Recurrent Neural Network. http://www.yuthon.com/2016/10/30/Notes-for-CS231n-RNN/ (visited 2017-05-11).

[5] Minna Aittasalo, Henri Vähä-Ypyä, Tommi Vasankari, Pauliina Husu, Anne-Mari Jussila, and Harri Sievänen. Mean amplitude deviation calculated from raw acceleration data: a novel method for classifying the intensity of adolescents' physical activity irrespective of accelerometer brand. *BMC Sports Science, Medicine and Rehabilitation*, 7(1):18, 2015.

[6] M.A. Arbib. *The Handbook of Brain Theory and Neural Networks*. A Bradford book. CogNet, 2003.

[7] James Bergstra, Frédéric Bastien, Olivier Breuleux, Pascal Lamblin, Razvan Pascanu, Olivier Delalleau, Guillaume Desjardins, David Warde-Farley, Ian Goodfellow, Arnaud Bergeron, et al. Theano: Deep learning on gpus with python. In *NIPS 2011, BigLearning Workshop, Granada, Spain*, volume 3. Citeseer, 2011.

[8] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[9] Commission E. Taking forward the strategic implementation plan of the european innovation partnership on active and healthy ageing, 2013. http://ec.europa.eu/health//sites/health/files/ageing/docs/com_2012_83_en.pdf (visited 2017-05-22).

[10] Isabelle Guyon and André Elisseeff. *An Introduction to Feature Extraction*, pages 1–25. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, Nov 1997.

[12] Alexander I. Luik Igor V. Tetko, David J. Livingstone. Neural network studies. 1. comparison of overfitting and overtraining. *J. Chem. Inf. Comput. Sci.*, 35(5):826–833, 1995.

[13] Vassili Kovalev, Alexander Kalinovsky, and Sergey Kovalev. Deep learning with theano, torch, caffe, tensorflow, and deeplearning4j: Which one is the best in speed and accuracy? 2016.

[14] Sanjay Krishnan, Animesh Garg, Sachin Patil, Colin Lea, Gregory Hager, Pieter Abbeel, and Ken Goldberg. Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning. In *International Symposium of Robotics Research. Springer STAR*, 2015.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[16] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1096–1104. Curran Associates, Inc., 2009.

[17] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16:115, 2016.

[18] Dimitri Palaz, Mathew Magimai.-Doss, and Ronan Collobert. Analysis of cnn-based speech recognition system using raw speech as input. In *Proceedings of Interspeech*, number Idiap-RR-23-2015, pages 11–15. ISCA, ISCA, September 2015.

[19] Thomas Plötz, Nils Y Hammerla, and Patrick L Olivier. Feature learning for activity recognition in ubiquitous computing. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[20] P. Rashidi and D. J. Cook. Keeping the resident in the loop: Adapting the smart home to the user. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 39(5):949–959, Sept 2009.

[21] Martin Riedmiller. Advanced supervised learning in multi-layer perceptrons — from backpropagation to adaptive learning algorithms. *Computer Standards & Interfaces*, 16(3):265 – 278, 1994.

[22] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, J. Doppler, C. Holzmann, M. Kurz, G. Holl, R. Chavarriaga, H. Sagha, H. Bayati, M. Creatura, and J. d. R. Millàn. Collecting complex activity datasets in highly rich networked sensor environments. In *2010 Seventh International Conference on Networked Sensing Systems (INSS)*, pages 233–240, June 2010.

[23] Sebastian Ruder. An overview of gradient descent optimization algorithms. http://sebastianruder.com/optimizing-gradient-descent/ (visited 2017-05-11).

[24] F.M. Soares and A.M.F. Souza. *Neural Network Programming with Java*. Packt Publishing, Limited, 2016.

[25] Jian Bo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 3995–4001. AAAI Press, 2015.

[26] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[27] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang. Convolutional neural networks for human activity recognition using mobile sensors. In *6th International Conference on Mobile Computing, Applications and Services*, pages 197–205, Nov 2014.