HUMAN ACTIVITY RECOGNITION USING DEEP LEARNING TECHNIQUES

Attila-Balázs Kis
Department of Computer Science
University of Stuttgart
st178772@stud.uni-stuttgart.de

Shunyi Deng
Department of Computer Science
University of Stuttgart
st175860@stud.uni-stuttgart.de

July 17, 2022

ABSTRACT

Human Activity Recognition (HAR) is an active research topic in the machine learning and deep learning communities. The aim of systems performing HAR is to classify and possibly predict, based on observed behavior, the current activity performed by the subject. Out of multiple alternatives the current approach is utilizing multiple data sources such as smart devices such as watches or bands. These devices collect sensor data throughout the day of the subjects in order to learn behavioral patterns and correlations to be able to perform the desired function. In this paper we present an approach utilizing deep learning techniques to perform HAR on a state-of-the-art dataset called HAPT [1]. We will present the methodology, the results and discuss future improvements, together with possible alternatives.

1 Introduction

In the following chapter we will describe briefly the used, from now on referred to as HAPT, dataset. We will also give a generic introduction in neural networks.

1.1 HAPT

The data set is obtained by collecting signal data of various activities of 30 experimental participants at different time. The dataset contains two parts. The one is the raw triaxial signals (3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz) from the accelerometer and gyroscope of all the trials with participants and related labels. The other is records of activity windows, which has a 561 *preprocessed feature vector* with time and frequency domain variables. And its associated activity labels. We use the latter one. The labels are *12 activities*, 3-3 static and dynamic activities, and 6 postural transitions that occurred between the static postures.

1.2 Neural networks

Neural networks (NN) are a subset of machine learning and are at the heart of deep learning algorithms. Its name and structure are inspired by the human brain. A NN consists of layers of nodes, consisting of an input layer, one or more hidden layers, and an output layer. Each node, also called an artificial neuron, is connected to another node with associated weights and biases. Neural networks are developed to successfully tackle both *classification* and *regression* tasks. The iterative process of network "learning" called optimization, and classification tasks as example steer the network to associate input data to a given set of labels. We are going to use networks to successfully classify sensorial input coming from the subjects to classify the action performed in the previously mentioned 12 classes.

In the following: Chapter 2 describes the methodology, development flow and details of the networks used. Chapter 3 is allocated to present and discuss details. Chapter 4 discusses possible future work and chapter 5 concludes our work on the subject of HAR using deep learning techniques.

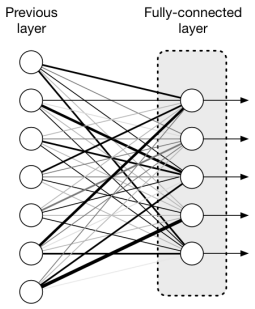

Figure 1: Fully connected layer[2]

2 Methodology

In this chapter we will describe the main building blocks of the neural networks, for each subsection we will provide a comprehensive theoretical ground, based on the difficulty level of the component, and will discuss the *application* of the block in our architecture development. Note that the subsections are placed in the order in which we used them to *enhance* the performance of the neural net. At the end of the chapter we will provide technical details about the networks, such as loss function and any additional detail which make out solution stand out from the existing ones.

2.1 Fully Connected Neural Networks

2.1.1 Theory

The fully connected layer is one of the basic building blocks of neural networks. Each neuron in the fully connected layer is fully connected to all the neurons in the previous layer. The fully connected layer plays the role of mapping inputs to different spaces, such as shaping input features to lower number of nodes, or fully connected layers are mostly utilized to map the learned "distributed feature representation" to the output label space. Fig. 1 presents a fully connected layer.

2.1.2 Application

The fully connected layers present the baseline of the networks developed throughout the development process. The baseline is created using two linear layers, one input and one output layers. The input shape and output shape are fixed since all the 561 features are going to be used, and since using a specific classification loss, 12 outputs are going to show the probabilities of the network predicting the class label. The inner hidden node size is going to be the focus of the updates, in the following way: The baseline is going to use 256 hidden nodes, and it's going to be referred to in the future as *FC*. Since we observed that the network tends to overfit, we adapted the hidden node size, thus coming up with the *FCs* - small networks, containing only 64 hidden nodes. Note that both FC networks use a nonlinear activation between the two layers, used mainly to learn non-linearities. We will analyze the difference between the two networks in Chapter 3.

2.2 Recurrent Neural Networks

2.2.1 Theory

Recurrent layers' outputs are composed by the usual components, having also an additional term, the previous (multiple) inputs. This way the recurrent layers compose a view about the correlations in the inputs built up throughout time. Any network containing a recurrent layer is called a recurrent network, i.e. RNN. Fig. 2 presents the layout and the unrolling of standard RNNs. These networks in their original form have serious short-term memory problems, and long-term data has little impact (even if it is important information).

Tackling the short and long term data issue, variant architectures such as *Long Short-Term Memory* (LSTM) and *Gated Recurrent Unit* (GRU) appeared. Their advantage is that long-term information can be effectively retained. And they select important information to keep, and unimportant information will choose to forget. Fig. 3 presents the way more complex architecture of LSTM networks, containing the forget and cell state incorporated into the recurrent structure.

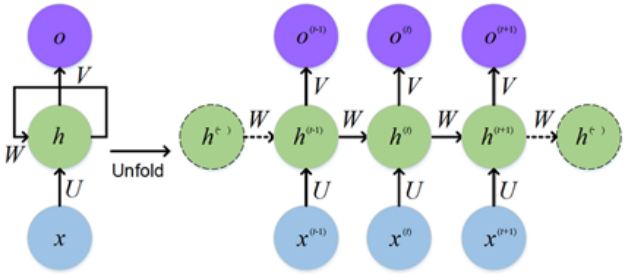

Figure 2: Recurrent layer unrolled layout

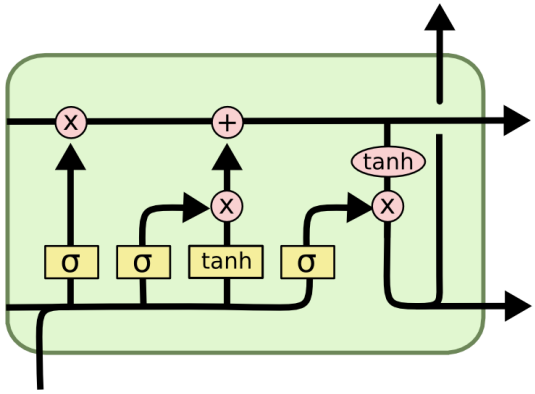

Figure 3: Long Short-Term Memory architecture [3]

One of the most important aspects of recurrent networks is how many data samples the network is going to process at once, i.e. what is going to be the look-back size of the network.

2.2.2 Application

The data provided by the dataset proved, since it's nature, to be a very good ground to extend the baseline network by using an LSTM layer in-between the input and output fully connected layers. The reason to insert the recurrent layer after the linear layer is based on size considerate, since the recurrent networks have a large size of parameters compared to the fully connected ones. Thus using the first linear layer to reduce the number of nodes to 256, and using *two stacked* LSTM layers, the network will be able to learn powerful temporal connections together with not getting quickly oversized. Note further that the activation after the LSTM layer is not needed, since it's architecture contains a hyperbolic tangent activation for each layer already.

2.3 Convolution Neural Networks

2.3.1 Theory

Convolutional neural networks are described in the literature as neural networks containing convolutional layers. The mathematical definition of the convolution states that the aforementioned operation is utilizing two functions, and produces a third function which expresses how the shape of one is modified by the other one [4]. In image processing, the same operation is described as the function that utilizes a kernel to transform the input image in a particular way to *highlight features*. Fig 4. presents the main idea of the convolution: *swipe* the kernel (3x3 in the image) over every pixel of the input and it's neighborhood, thus producing the convoluted result. Different kernels are capable

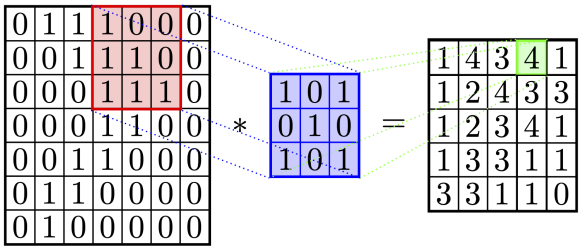

Figure 4: An example of convolution in 2D

of highlighting relevant information about the input data which might not be possible to uncover using conventional methods.

Convolutional layers are a big part of the success of the deep learning in the area of computer vision, utilizing the convolution operation as their core functionality. Carefully constructing kernels and applying them sequentially utilizing additional layers such as *pooling*, different features of the inputs can be *highlighted* and passed forward in the network.
Convolutional neural networks can start with low-level features such as edges or corners, and learn more and more complex features, which is the key of building a proper understanding of the input data for the model [5]. This particular idea is highly applicable in two-dimensional (image) input scenarios, but found a big success in one-dimensional, i.e. signal data. The approach is similar, but utilizing the convolution with a kernel in one dimension, find temporal patterns in data consecutive data points.

The size and the type of the kernel highly differs on the application, but kernels such as *Sobel* or *Gaussian* and their variations are just a couple examples of the multitude of kernels developed such that a convolutional layer can produce multiple "representations" of the input data which are passed forward onto the next layer. Modern utilization of the convolutional layers are mostly focusing on *applying multiple kernels* at once at the same layer and pass the different results onwards, such that as many features are highlighted as possible.

2.3.2 Application

Applying one-dimensional convolutions over the time dimension of the input data is a wide-spread solution for using CNNs. Thus we decided to apply a (3x1) kernel shape over the sequence of the input, taking extra care to return the same length of sequence input. Since the error is computed over the whole output sequence of the network, the batch and time dimensions cannot be changed. Thus utilizing the *stride* and *padding* of the convolutional layer, the network outputs from the initial layer the convoluted, but same size output, which is going to be processed by the recurrent layers, added in the previous chapter to the network structure.

Another valid alternative, and highly dependent on the design choice, is using the convolutional kernel after the recurrent layers, but we opted for the first solution, since the convolutional layer outputs 256 kernel convolutions in our implementation, the convoluted output of the layer might divulge connections which might not be obvious using only input in the recurrent layers.

2.4 Technical details of the neural networks

In this subchapter we are going to give some technical details of the networks, training process and the overall project.

- When preparing the sequenced data for the recurrent networks, we decided to create a *one-by-one* sliding when creating input data for the models;
- The network output and evaluation is made in a *many-to-many* manner, and the inference output of the network is the last element of the outputted sequence;
- Training hyperparameters: 150 epochs, 1e-4 learning rate;
- use *Adam* optimizer, for it's proven speed and performance;
- classification task requires the use of *CrossEntropyLoss*;
- in order to avoid overfitting, *Early Stopping* regularization is used.

The code together with documentation can be found at the following link: GitHub repository. The documentation shows ways to train different architectures, evaluate and visualize their performance using different functions such as SHAP (SHapley Additive exPlanations) and state/weight visualizations.

3 Results

In the following chapter we will present the results achieved using the previously presented layers in different structures together with a representative additional metric for assessing neural network models: *the number of trainable parameters*.

All of the models were evaluated using the HAPT pre-processed test dataset. The three metrics which are going to be presented are *accuracy* (how often prediction equals the correct labels), *F1 score* (harmonic mean of the precision and recall) and *AUC* (area under Receiver Operatinv Characteristic curve) - all of them in *percentage*. The bigger the percentage of the metrics, the better the model performance.

We will present three hypotheses and will use a comprehensive table to validate these:

1. recurrent networks are performing better than fully connected ones;
2. sequence length is highly relevant in recurrent networks, even more than convolutional parameters;
3. convolutional layers enhance the results of the networks.

Model Name	Acc	F1	AUC	#params	Comment
FC	91.96	91.95	91.01	146,956	baseline model
FCs	92.78	92.76	91.02	36,748	smaller hidden size
RNN_LSTM_sl1	91.49	91.42	88.15	1,199,628	sl = sequence length
RNN_LSTM_sl5	95.44	95.42	94.30	1,199,628	
RNN_LSTM_sl20	**97.23**	**97.21**	94.97	1,199,628	
CRNN_LSTM_sl5	95.82	95.78	94.07	1,486,860	C = convolutional
CRNN_LSTM_sl20	96.34	96.30	**95.43**	1,486,860	

3.1 Discussion

In the current subsection we are going to assess the previously presented hypotheses, and will discuss further details, such as considering size, since the use-cases of HAR have to take in account the portability of models.

1. Hypothesis 1
   - as seen clearly from the result table, recurrent networks manage to *clearly outperform* their fully connected counterparts;
   - on the other hand, a recurrent network using sequences of one value at once can be seen to *underperform* the fully connected networks, because of the lack of temporal connections, and the fact that way more gradient calculations affect the performance of the network.
2. Hypothesis 2
   - we can see from the table above that the sequence length given to the recurrent layer is a *relevant hyperparameter* in developing the most optimal network. 5 samples do a way better job than one, and sequence length of 20 massively outperforms the previous sampling. An interesting experiment is to find the maximum sequence length, since 20 is clearly not too many samples. A visible turn-around point of sequence length is the decline in network performance, signaling the fact that too many data points are considered as a sequence, at once;
   - keeping the convolution kernel size always (3x1) is a quite rigid design choice, since the target of the convolutional layer is to discover hidden correlations. For sequence length of 5 the convolutional layer is an upgrade, but still doesn't manage to get to the performance of the 20 sequence length simple recurrent network. Furthermore the results suggest that for sequence length of 20, there might be a kernel which enhances the network performance, rather than only marginally doing so, as presented in the result table.

3. Hypothesis 3
    - comparing recurrent and convolutional recurrent networks for the same sequence length, the sequence length of 5 with (3x1) convolution kernel proves to be a clear enhancement;
    - for sequence length of 20, the (3x1) convolution kernel only marginally enhances the performance, visible by the greater AUC score only. Further research on optimal kernel size might be necessary.

The results presented by us outline clearly the truth of the three hypotheses presented previously, together with the clear image that neural networks can learn the activity classification task on the HAPT dataset.

4 Future work

We can categorize future work in two sub-categories, namely: data & network research.

4.1 Data

We utilized the preprocessed HAPT dataset, including a large variety of feature enhancements created using mathematical and statistical means. We believe that the high number of features can be lowered using different means such as *feature analysis* and different heuristics.

Furthermore using the provided raw data in the HAPT dataset, we believe that utilizing only the actual signal data together with preprocessing, a more comprehensive dataset can be built, thus networks might be able to generate even higher performances. An exciting aspect to tackle is the slightly unbalanced nature of the dataset, which, by preprocessing, can be corrected up to a point.

4.2 Network Research

The networks we presented in the paper are clearly highlighting the steps forward by utilizing more and more complex layers, but still doesn't utilize relevant modern enhancements of neural networks such as:

- an alternative of LSTMs, namely GRUs;
- a powerful alternative of recurrent networks, *transformers*;
- skip connections.

An interesting aspect to look more closely to is the size of the networks, together with their inference speed, since the task of Human Activity Recognition is highly requested in mobile environments such as smart bands and smartphones, usually not having as big of a computing power accessible as computers used for model training.

Furthermore experimenting with different optimizers, together with learning rate schedulers, we believe would enhance the performance of developed networks.

5 Conclusions

We successfully proved that *Human Activity Recognition* can be performed with a relevant performance on the *HAPT* dataset. We created a baseline network, and enhanced it using different techniques and neural network layers, so that we achieved a maximum *97%* testing accuracy. Furthermore we compared and discussed the development choices converging towards the best solution.

References

[1] Human activities and postural transitions dataset.

[2] A Bhat. Fully connected layer.

[3] Nick McCullum. Long short-term memory networks (lstms).

[4] Madhushree Basavarajaiah. Six basic things to know about convolution, Mar 2022.

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