University of Strathclyde Engineering

EE581
Human Activity Recognition Using Multi-Modal Dataset

Duncan Wither - 201514824

Mario Emilio Manca – 201543775

17th of April 2020

Abstract

Providing feedback to patients performing an exercise is a time consuming and therefore expensive activity that requires a specialist to monitor the patients. This project explores the use of multi-modal classification techniques to categorise different exercises which aim to improve accuracy whilst reducing computational costs. The dataset that will be used is the MEx Multi-Modal Exercise Dataset [1] that provides 8 different rehabilitation exercises performed by 30 different patients with Musculoskeletal Disorders (MSD). Data was collected by four different sensors, a depth camera, a pressure mat and two accelerometers, one on the patient wrist and the other on the patient thigh. First, the data is acquired and represented on a 3D space. This process helps to visualise the data to have a better understanding of the degree of difficulty in the classification process. Then a deterministic method (kNN) is used to classify the data into the 8 exercises. This process will be used primarily as a benchmark for the deep learning algorithm performance. Finally, a deep learning unsupervised algorithm will be implemented for classification. For both methods, the classification process will be performed using individual sensors one at the time and then combined in a multi-channel approach. Results suggested better performance via the deep learning approach, in both speed to and accuracy, and they also demonstrated the multi-modal algorithm has a better accuracy compared with using the single sensors.

Acronyms and Abbreviations

Acronym	**Definition**
ACT	Thigh accelerometer
ACW	Wrist accelerometer
CNN	Convolutional Neural Network
DC	Depth Camera
DL	Deep Learning
kNN	k-Nearest Neighbour
MC-DCNN	Multi-Channel Deep Convolutional Neural Network
MC-NN	Multi-Channel Neural Network
MLP	Multi-Layer Perceptron
MSD	Musculoskeletal Disorder
PM	Pressure Mat

Contents

1. Introduction

Patients with Musculoskeletal Disorders (MSD) are usually prescribed physiotherapeutic exercises in order to recover or improve their range of movement. Assessing the exercise and its correctness usually requires a specialist overseeing the patient while they perform the exercise. This implies a doctor-patient ratio of one to one to ensure a good examination, meaning high costs and long waiting lists for the patient. This project wants to make the first step in using deep learning to evaluate the patient performance. At this stage attention will not focus on the quality of the exercise, but only on exercise recognition. The benefit in having the data already sorted in the different exercises reduces processing time and sets the base for a quality evaluating algorithm.

In the MEx Multi-Modal Exercise Dataset [1], which will be used during this project, thirty patients were recorded performing seven different exercises. One of them, however, is repeated for the left and right side and it was decided to count it separately. The dataset contains data from four sensors, a depth camera (DC) recording the patient from above, a pressure mat (PM) sensing the force applied from the patient to ground, and two accelerometers, one on the patient wrist (ACW) and the other one on the patient thigh (ACT).

The primary focus is to explore and compare multi-modal classification methods. The two approaches compared were the deterministic k-nearest neighbour (kNN) algorithm and unsupervised deep leaning using Convolutional Neural Networks (CNN) and Multi-Layer Perceptrons (MLP). Both methods will be implemented and compared in their single and multi-channel forms. The paper will first present the objective set at the start of the project and their relative progress. Then, in the methodology section, it will be explained how it was planned to accomplish them and the relative strategies adopted. The experimental set up will describe in technical details how the different algorithms were constructed and implemented. Finally, results will be compared and analysed.

2. Objectives

Shown below is a table of the objectives laid out in the statement of intent [2]. This includes the final status of each item. Of note was the stretch goal of exploring unsupervised classification, which was not met. However due to the achievement of the deep learning classifier, and the non-essentiality of this objective, this was considered acceptable.

Table 1. List of objectives along with final progression.

Objective	Why	Imp.	Progress
Visualising data	This is to provide some human context to the information. This will also allow for a better understand of what the classifiers are identifying and the difficulty of the task	Low	Done
Classify data using deterministic method	This will provide a benchmark to compare the deep learning method against.	High	Done
Classify data using the deep learning method	This is the primary objective of the project with reasons outlined in the introduction. The primary method would be the MC-DNN, however if this proved too consuming given the time constraints, a simpler deep convolutional neural network could be used.	High	Done
Explore effect of changing training set data	By changing the training set data in numerous ways different scenarios can be found. For example, changing the training and testing percentages to explore how the algorithm behaves.	High	Done

Look at unsupervised classification	A stretch goal of this project would be to look at unsupervised learning in order to classify the data without labels. This would be an interesting outcome given the different aspects of the data that could be interpreted, and how the different datapoints amalgamate.	Optional	Future Work

3. Methodology

3.1. Data Visualisation

A Python script was implemented for each sensor to accept user inputs and visually represent the data for a given sensor and exercise. For the two accelerometers, the 3D coordinates were plotted on a graph that displayed the path the sensor took over the execution of the exercise. The frames from the depth camera were concatenated creating a video, which speed was increased to reduce visualisation time. The data from the pressure mat was treated as an image, considering each pressure point as a pixel. Data was scaled accordingly and converted into a video as done for the depth camera.

3.2. Deterministic Method

The algorithm used for the deterministic method, as specified in [2], is the kNN algorithm, using dynamic time warping (DTW). The kNN method was used because it is a simple concept, it makes no assumptions about the data and requires no training. This does come with some downsides, such as sensitivity to trivial features of the data [3], however, the larger issue is the computational cost. For smaller data sets with small data points this is not an issue, however, this scales poorly and became an issue, as MEx has relatively large individual data points. DTW was used to measure how alike two datasets were. This means the kNN operated without a feature map and can be considered scalar. DTW was used as theoretically would perform better than the simple Euclidian distance as DTW would better catch the structural similarities between the exercises [4]. The DTW works by comparing every time portion with another to find the lowest cost path between, and as such is performed in three distinctive steps:

1. Create cost matrix, by taking the RMS difference between datapoints
2. Path-find through matrix. This was achieved through the python pathfinding library.
3. Calculate costs.

Steps 2 and 3 could be done simultaneously, however the requirements of the pathfinding library used had specifications for the map matrix which made it simpler to separate these steps. A typical DTW calculation from the project can be seen in Figure 1 below.

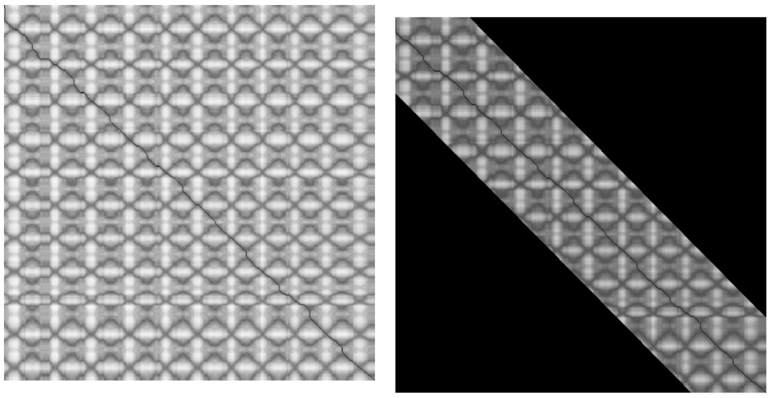


Figure 1. DTW visualisation, with (right) and without (left) the discussed optimisations.

As might be expected, using the DTW significantly increased the cost, as not only were there more calculations required for the initial graph, but also the pathfinding was computationally expensive. In order to improve computational efficiency several things were done:

- Create a search envelope by cutting off the corners from the map matrix. An example of this can be seen on the right of Figure 1. This is because opposite corners are where the timelines of the two sets are highly mis-aligned, and thus should not be traversed for alike sets. This improves both the pathfinding speed – by reducing the search space – but also the initial creation time of the cost matrix as less RMS calculations are required. It was found experimentally that using a 20% strip width cut map creation time to 33.5% of the original and cut the pathfinding time to 12.7% of the original.
- Using array data structures from the NumPy library for map creation significantly improves speed. Further similar speed improvements could be gained by using pre-compiled C code; however, the current speed was deemed enough.

The method described above effectively runs the data independently for each of the four sensors. However, in order to combine the results from each of the sensors exercises, a simplified Bayes optimal prediction strategy [5] was implemented, which assumes the results are independent. This involved taking a weighted vote based on the accuracy of the result. The benefit of this method is it not only to give an optimal result, but a confidence the kNN has in the result. The equation below shows how it was calculated:

$$Guess = argmax\left(A_{dc} * OH_{dc} + A_{pm} * OH_{pm} + A_{act} * OH_{act} + A_{acw} * OH_{acw}\right)$$

Where A_X is the accuracy of sensor X, and OH_X is the prediction from the sensor X data in one-hot encoding. The *argmax* function returns the index of the max value, which corresponds to the exercise number due to the one hot encoding. In order to calculate the confidence, the value is as follows:

$$Confidence = max\left(\frac{A_{dc} \times OH_{dc} + A_{pm} \times OH_{pm} + A_{act} \times OH_{act} + A_{acw} \times OH_{acw}}{A_{dc} + A_{pm} + A_{act} + A_{acw}}\right)$$

To get both A_X values for the kNN the testing was performed open, allowing the kNN to be run multiple times to get averaged results for A_X. This entire procedure was run with four separate k values, 1, 3, 5 and 7, to explore the optimal value.

3.3. Deep Learning Method

Supervised learning was used for the deep learning approach, meaning that each entry was labelled based on the exercise it belonged.

Initially, the four sensors were treated separately. Each one of them had its own algorithm that categorised the input samples into the eight different exercises. Then, a multi-input network was developed to accommodate four channels to extract features from all the sensors at once.

Finally, different training and testing percentage were explored for both the single and multi-channel networks.

4. Experimental Setup

4.1. Data Visualisation

Three scripts were written for data visualisation, one for the accelerometers, one for the depth camera, and one for the pressure mat:

- Accelerometers – a Python script allows the user to choose between the thigh and wrist accelerometer, and to input patient and exercise number. Then, the code accesses the data from the MEx dataset accordingly. The CSV file is read line by line, during each iteration a line is drawn between the previous and the current location on a 3D graph. With the first iteration being an exemption where only the current position is plot.

- Depth Camera – as per the accelerometers script, the user is prompt to input the patient and the exercise number. Furthermore, there is the option to select the folder path where it is desired to save the video. After that, the code accesses the depth camera data frame by frame and converts it from the 1 x 192 form it was saved on the CSV file to the original 12 x 16. Then, each frame is temporary saved as a JPG after being upscaled to 240 x 320 and it is concatenated into an MP4 video. The single frames are then deleted.

- Pressure Mat - the script for the pressure mat is an adaptation of the one used for the depth camera. The only difference consists in the data dimension. The pressure mat outputs a 32 x 16 matrix, which is flattened in the CSV file to a 1 x 512 array. Again, each frame is converted back to its original form and upscaled to 480 x 240 for better visualisation, before being converted into an MP4 video file.

4.2. Deterministic Method

The experimental setup for evaluating the deterministic method was to run the kNN five times to determine the best value for k, and to get an estimate of the predictive accuracy of each sensor. The kNN would then be run again ten times using the previously found k values and accuracies to provide a finalised accuracy. Provisions within the code had been made to enable the splitting of the longer datasets into multiple instances, however this was not used as it increased the computational time significantly, and initial work had shown the accuracy gains were minimal. Instead the datasets were conditioned in other manners however, including down sampling of the accelerometer data, and limiting the size of the sets. These parameters are shown below in Table 2:

Table 2. Sensor Set lengths and down-sampling rates.

	Set Length	Down sampling Rate
Wrist Accelerometer	100	10
Thigh Accelerometer	100	10
Depth Camera	150	1
Pressure Mat	150	1

To compare to the original researchers the exercises from 25 of the patients were used as the training data, with the other 5 patients being used for the testing sets.

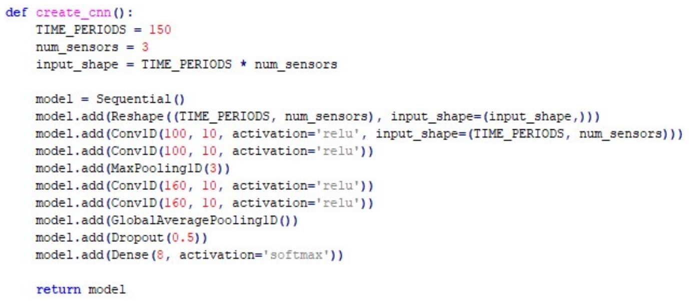
4.3. Deep Learning Method

4.3.1. Single Channel Neural Networks

The single channel networks were built as a reference point for the multi-input network. So, to have a fair comparison, the number of entries were limited. This was necessary because in the multi-channel network all channels must have the same number of entries, while in the MEx dataset the number of samples not only varies from each sensor, but also from patient to patient. This limits the number of entries to 150, which is the length of the shortest exercise in the dataset.

Initially, the chosen multi-channel network was a fully convolutional neural network (MC-DCNN) as proposed by Zheng et al. [6]. As such it was decided to implement a convolutional neural network (CNN) for each channel. A 1D CNN was implemented for the two accelerometers, the model architecture was the same for both thigh and wrist sensors [7]. Figure 2 shows the function in the code responsible of creating the CNN model and Figure 3 offers a graphical representation of it. Initially the data is fed as a 150 x 3 matrix (i.e. 150 samples, 3 axis). Then a 1D convolutional layer with 100 filters and kernel size 10 is applied. A second 1D convolutional layer with the same characteristic before pooling allows to extract more complex features.

After pooling two more 1D convolutional layers are applied, this time with 160 filters. After a second pooling and the dropout phase a fully connected layer is implemented to categorise the data into the 8 exercises.



```python
def create_cnn():
    TIME_PERIODS = 150
    num_sensors = 3
    input_shape = TIME_PERIODS * num_sensors

    model = Sequential()
    model.add(Reshape((TIME_PERIODS, num_sensors), input_shape=(input_shape,)))
    model.add(Conv1D(100, 10, activation='relu', input_shape=(TIME_PERIODS, num_sensors)))
    model.add(Conv1D(100, 10, activation='relu'))
    model.add(MaxPooling1D(3))
    model.add(Conv1D(160, 10, activation='relu'))
    model.add(Conv1D(160, 10, activation='relu'))
    model.add(GlobalAveragePooling1D())
    model.add(Dropout(0.5))
    model.add(Dense(8, activation='softmax'))

    return model
```

Figure 2 Accelerometers CNN Model

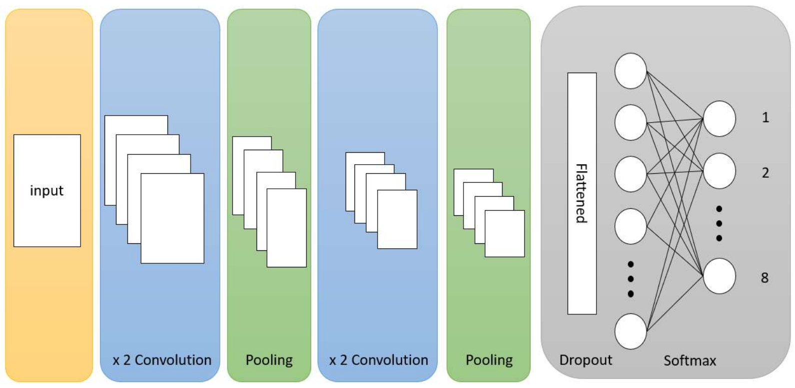


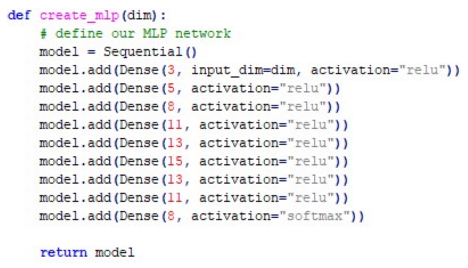
Figure 3 Accelerometers CNN Model Diagram

The algorithm was trained with 70% of the dataset, randomly selected, and tested with the remaining 30%. Epochs were set to 100. An average was taken over several simulations as the weights in the network are randomly selected leading to slightly different results. Table 3 shows the accuracy for the two CNN networks. The outcome was not what was expected.

Table 3 Accelerometers CNN Accuracy

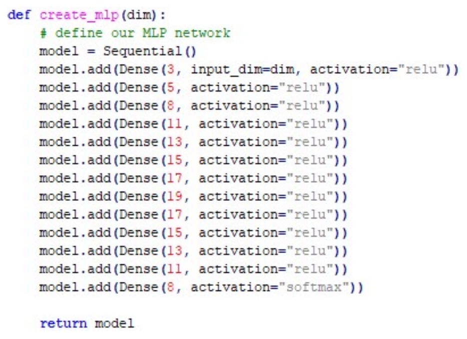
Data \ Simulations	Accuracy (%)					
	1	2	3	4	5	Average
ACT CNN	33.33	41.67	38.89	31.94	30.56	35.28
ACW CNN	8.33	8.33	8.33	8.33	8.33	8.33

To overcome this problem, it was decided to explore the MLP approach [8]. Two different models were created, one for the thigh accelerometer and one for the wrist accelerometer. Figure 4 shows the two models. The ACT model consists of an input layer with 3 nodes (one per axis), 7 hidden layers, and an output layer for the 8 different exercises. The ACW model has a 3-node input layer, 11 hidden layers, and an 8-node output layer. The best architecture for both models was found empirically, experimenting with the depth and the number of nodes in the hidden layers.



```
def create_mlp(dim):
    # define our MLP network
    model = Sequential()
    model.add(Dense(3, input_dim=dim, activation="relu"))
    model.add(Dense(5, activation="relu"))
    model.add(Dense(8, activation="relu"))
    model.add(Dense(11, activation="relu"))
    model.add(Dense(13, activation="relu"))
    model.add(Dense(15, activation="relu"))
    model.add(Dense(13, activation="relu"))
    model.add(Dense(11, activation="relu"))
    model.add(Dense(8, activation="softmax"))

    return model
```



```
def create_mlp(dim):
    # define our MLP network
    model = Sequential()
    model.add(Dense(3, input_dim=dim, activation="relu"))
    model.add(Dense(5, activation="relu"))
    model.add(Dense(8, activation="relu"))
    model.add(Dense(11, activation="relu"))
    model.add(Dense(13, activation="relu"))
    model.add(Dense(15, activation="relu"))
    model.add(Dense(17, activation="relu"))
    model.add(Dense(19, activation="relu"))
    model.add(Dense(17, activation="relu"))
    model.add(Dense(15, activation="relu"))
    model.add(Dense(13, activation="relu"))
    model.add(Dense(11, activation="relu"))
    model.add(Dense(8, activation="softmax"))

    return model
```

Figure 4. Accelerometers MLP Models (ACT on the Left, ACW on the Right)

The advantage of the MLP is that it was possible to have more entries. The CNN takes a full exercise as an input because it looks at how each axis evolves over the full sampling period and tries to extract patterns from them. This means that, given there are 30 patients and 8 exercises, the total number of entries is 240. On the other hand, the MLP was set to observe the correlation between the 3 axes at each sample, resulting in 150 samples for 30 patients, for 8 exercises, which gives 36,000 entries. Given that deep learning heavily relies on the size of the dataset it is expected that the accuracy of the algorithm would increase. Table 4 shows the resulting accuracy. The same procedure, training percentage and epochs used for the CNN were implemented for the MLP approach.

Table 4 Accelerometers MLP Accuracy

Simulations / Data	Accuracy (%)					
	1	2	3	4	5	Average
ACT MLP	57.34	62.63	63.31	68.16	61.62	62.61
ACW MLP	50.34	44.93	33.87	30.62	51.39	42.23

As expected, accuracy significantly increased, with a 77% improvement for the thigh accelerometer and a 407% improvement for the wrist accelerometer. For these reasons, MLP was chosen over CNN for the two accelerometers.

For the depth camera and the pressure mat the CNN model implemented is the same, the only difference is the input data dimensions [9]. Figure 5 shows the model architecture for the aforementioned sensors. The input data goes through a convolutional layer, then the pooling phase the matrix is flattened and a fully connected 16-node layer is implemented before the final output layer were the data is categorised.

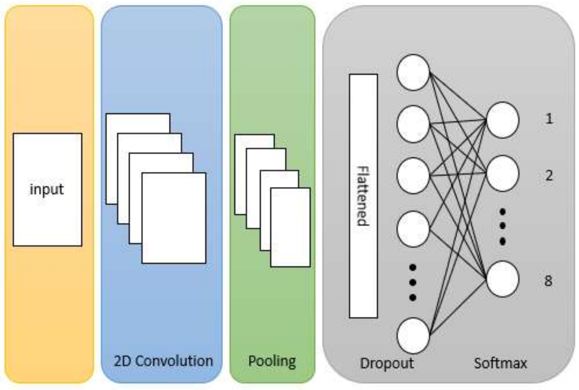


Figure 5 Depth Camera and Pressure Mat CNN Mode

4.3.2. Multi-Channel Neural Network

The multi-channel neural network (MC-NN) architecture was implemented by integrating together the models created with the help of a final MLP to combine, as shown in Figure 6. The single channels extract different features from different sensors and the last MLP classifies the data into the individual exercises.

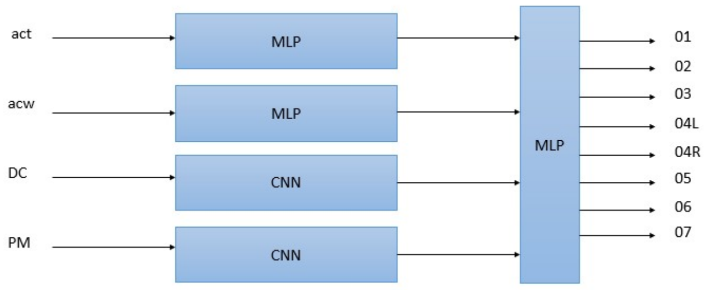


Figure 6 Multi-Channel Neural Network

5. Results and Discussion

5.1. Data Visualisation

As an example of what can be produced by the data visualisation code, Figure 7 shows the first exercise (knee-rolling) performed. For the depth camera and the pressure mat, critical frames were extrapolated from the video. From the depth camera (centre of figure) it is possible to see the patient laying on their back moving his/her knees first to the right and then to the left. The pressure mat data (bottom of figure) confirms that the weight of the lower back is shifted accordingly to the exercise. White regions are those subject to pressure (e.g. on the right it is possible to observe the constant pressure from the patient feet). The thigh accelerometer (top left) reflects the movement shown by the other sensors. The wrist accelerometer (top right) is not particularly useful in this case as the patient wrist stays still for the duration of the exercise.

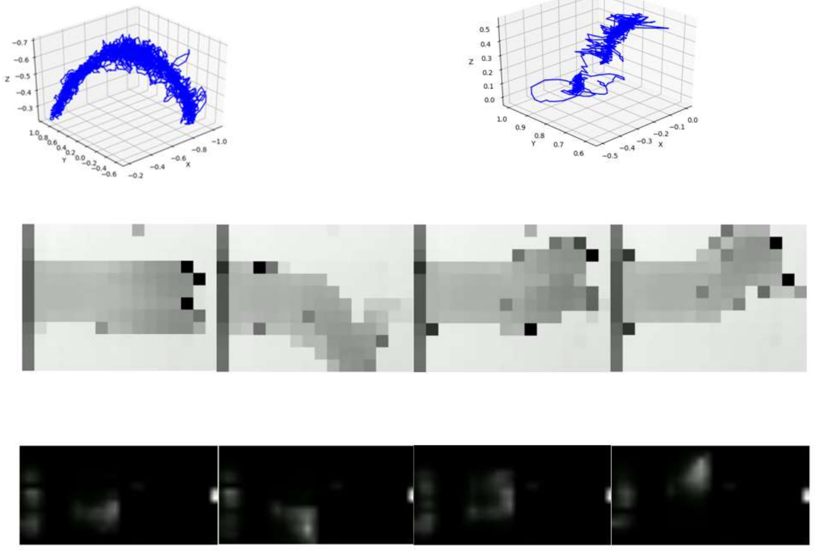


Figure 7 Patient 1 Exercise 1 visualisation. The plots shown are the thigh accelerometer (top left), wrist accelerometer (top right), depth camera (centre), pressure mat (bottom).

Visualising the data helped in understanding the difficulty of the task and clarified some of the results obtained from the different approaches. For example, as it will be discussed later, the wrist accelerometer algorithm was the one that struggled the most identifying the exercise in all the methods that were considered. This is because for the majority of the exercises performed in this dataset, the patient wrist stays in the same position, making categorisation challenging.

5.2. Deterministic Method

Evaluating the kNN using the method described after the first 5 runs the following accuracy values were found:

Table 5. Accuracies from initial set of runs, used as the prediction weighting in the latter runs.

k-Value / Sensor	Accuracy			
	1	3	5	7
ACT	0.633	0.6	0.667	0.7
ACW	0.4	0.467	0.5	0.467
DC	0.8	0.8	0.767	0.833
PM	0.333	0.367	0.267	0.2

The classifier was then run for another ten iterations using these accuracy values to classify the data. The results of both the individual sensors and the combined result are shown in Figure 8, along with the random chance of getting the correct result for reference. From this it can be seen the optimal k-value of 3 at 80%, however not by a large margin.

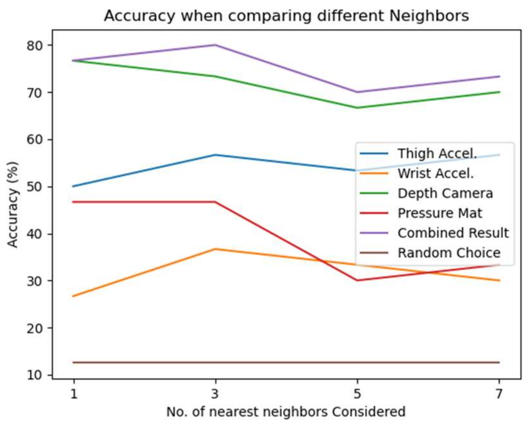


Figure 8. Results of the kNN classification analysis.

These results can be compared to the results achieved by Wijekoon et al. [10], who published the MEx dataset. Unfortunately, the results achieved by the kNN method proposed in this paper are worse than those achieved in [10] across all the sensors. However, they do show promise, as although there was no combined classification in [10], the method proposed in this report works better than any singular sensor as is shown in Figure 9.

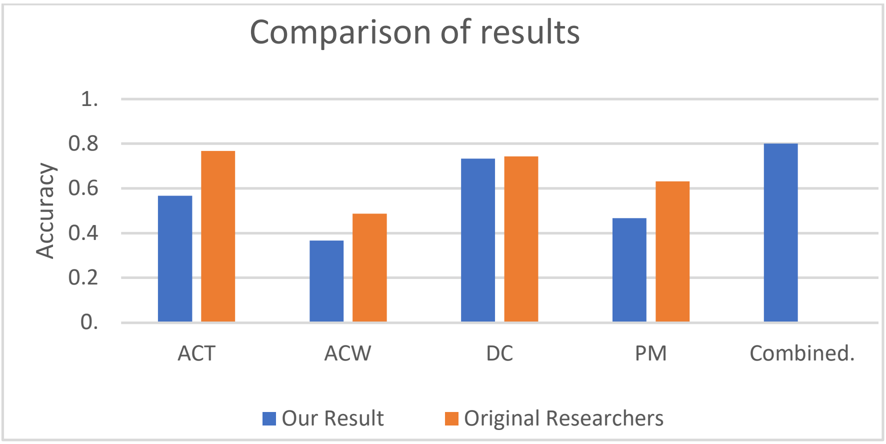


Figure 9. Comparison of Results with the original researchers.

The suspected reason for these discrepancies is the pre-processing of the information, especially for the accelerometer data. The researchers used a discrete cosine transform to create a feature set of the values, and thus performed nearest-neighbour calculation on the results. This method would also be more computationally efficient than the utilised method. On counterbalance however, the results for the deep learning method performed much better than both the researchers kNN and their other proposed methods.

5.3. Deep Learning Method

Each algorithm was trained and tested multiple times, and an average was taken across the different simulations. The complete list of values for each simulation is shown in Table 6 in Appendix A – Simulation Results. Figure 10 shows a comparison between the single channel networks and the multi-channel one. It is possible to notice that the two CNNs perform much better than the MLPs and achieve high accuracy. As discussed before, using CNNs for the accelerometers data was counterproductive in this particular case due to the size of the MEx dataset. The reason why it is possible to apply CNNs on the depth camera and pressure mat is because these two sensors produce 2D data that holds much more information compared with the 1D data coming from the accelerometer and the algorithm is capable of finding more correlation in the input data.

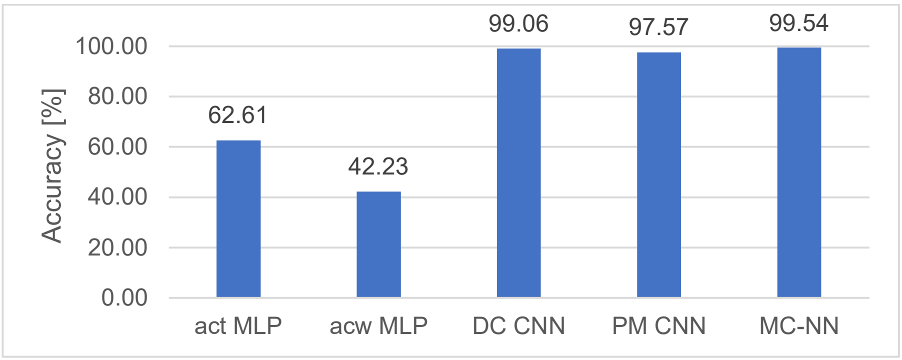


Figure 10 Single and Multi-Channel Comparison

Figure 11 graphically shows a comparison with the kNN approach. Deep learning performs better for each sensor with exception for the thigh accelerometer. For the combined approach, the multi-channel network performs over 50% better than the corresponding combined kNN approach. This proves that, as expected, deep learning accomplishes better results compared to kNN.

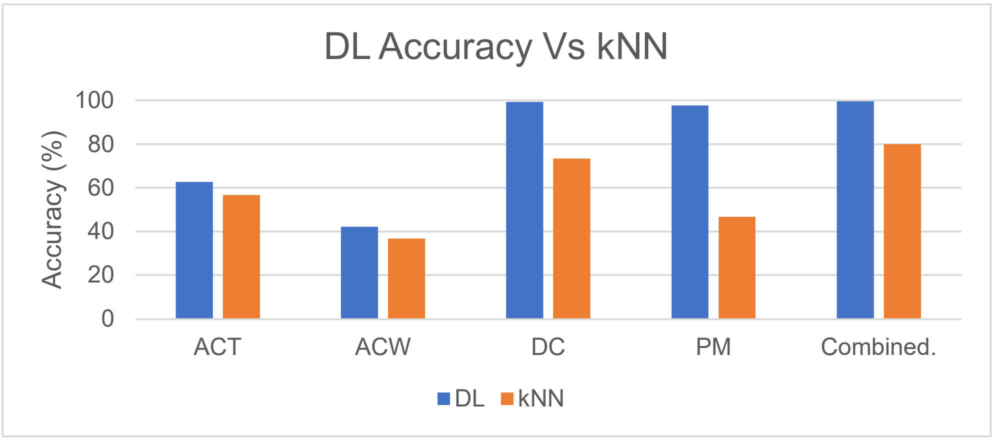


Figure 11 Deep Learning and kNN Comparison

Figure 12 shows a comparison between the different networks varying the training and testing percentages. As expected, the lower the training percentage the less chance the algorithm must learn from the dataset, hence its accuracy decreases. It is possible to notice how the multi-channel network performs better compared with any of the other single channel networks up to a threshold, which in this case seems to be

10% for training. After that, its accuracy is brought down by the two MLPs and the single CNNs perform better. On the other hand, it is worth considering that at this point the accuracy is below 95% so it is marginally important which one is performing better.

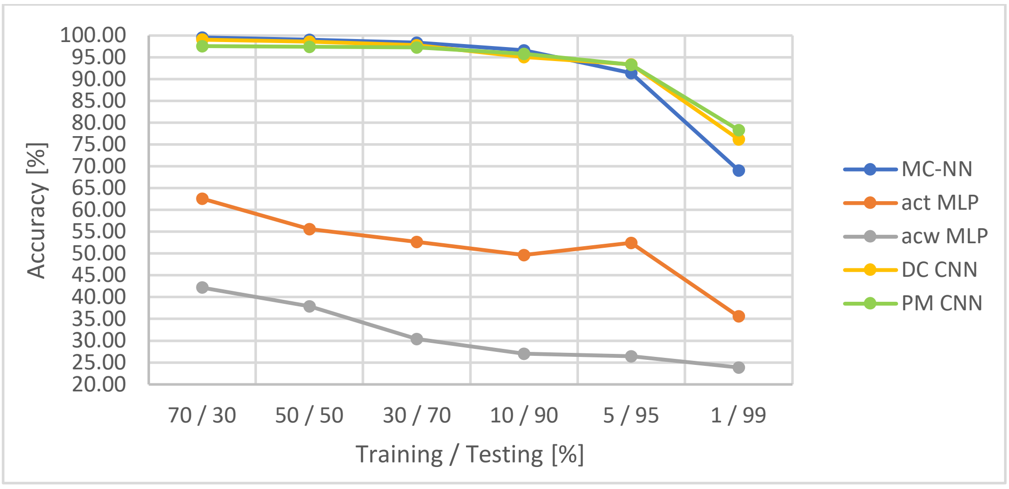


Figure 12 Different Training and Testing Percentages

6. Future Work

One of the main limitations in this project was the limited size of the database. Deep learning requires large numbers in terms of input data to extract, recognise, and categorise. The following pieces of future work assume access to use a database with more information. One item that would have been optimal to explore further is the implementation of CNN algorithms for the accelerometer sensors. This would allow the full implementation of a Multi-Channel Deep Convolutional Neural Network. Convolutional networks are notably better performers than MLP and would add value to the multi-modal algorithm implemented.

Explore unsupervised learning and compare it to the kNN and supervised learning approach. Unsupervised learning is very useful when the amount of exercises to classify is very large and it is not already labelled. In the dataset provided, the data was already structured in a folder system that made easy labelling the single exercise. Unsupervised learning could potentially take an unlabelled, randomised set of data and extract features from it. This would effectively remove the need to carefully store and label each exercise.

Next step to automate the evaluation process would be to implement a quality assessing algorithm that returns feedback on the exercise. The idea is to use a reference exercise performed by the specialist as ground truth, and compare the exercises performed by the patients to it. The algorithm could return a pass or fail mark and some feedback on the overall exercise, e.g. "Pass, but the patient should keep both feet in contact with the ground for the entire duration when performing the knee-rolling exercise".

7. Deliverables

One of the primary deliverables is the pre-trained multimodal deep neural network with the superior classification power to alternatives developed by Wijekoon et al. Another key deliverable developed is the multimodal deep learning algorithm. This can be modified to suit different data sources. The high accuracy of the MM-NN shows the potential for further exploration. Furthermore, within the codebase is the code for the four CNNs and the 2 MLPs for independent classification of the dataset. Other deliverable items include the modules developed for both the kNN functionality and to calculate the DTW cost which, although targeted towards the Mex dataset, can be easily modified to perform classification on other datasets. The documented code, files and the pre-trained model used for this project can be found online at [12].

8. Reflection and Conclusion

Upon reflection of the project, the only portion of the work that would be heavily changed is the methodology used to amalgamate the results for the kNN. The method used to obtain optimal k-values and predicted accuracies relied on all the data to be available in the initial run, and thus did not mimic the real-world scenario of classifying unknown data. A more rigorous strategy for estimating the values required would be to use rotation estimation [11] on the known training portion of the dataset, and then evaluate on the blind portion. The reasons given for not implementing this are given below:

1. The already relatively low accuracy of the kNN with DTW to perform multimodal classification in comparison with the DL methods. Further handicapping the algorithm (with less data) would have no material effect on the final outcomes of this study.
2. The computational cost of implementation would reduce the already poor performance of the algorithm.

However, if there were no constraints on time or processing power this would be something that would have been interesting to implement for both completeness of the project, and academic curiosity.

However, overall, this project can be considered a veritable success. As stated, the goals of this project were to explore the Mex dataset and perform both deterministic and deep learning classification on it, evaluating the methods independently and comparing the results. All these primary goals were met.

Throughout this process several different aspects provided difficulty however the primary challenges were technical in nature, in how to combine mixed data models using the Keras library. As can be seen in the code, and the results, this was overcome, highlighting key deep learning principles in the process. Another issue, already discussed, is the size of the dataset, however given the constrains of the project this was unavoidable, and the results speak for themselves.

The project management within the group was minimal, but effective. The plan laid out in the proposed statement of intent was kept to in the most part with only minor delays. The team management revolved around monthly meetings, with others interspersed, along with using online version control software to collaborate effectively on code.

To summarise, the results of the deterministic methods were disappointing in terms of accuracy, however this still demonstrated several points, including to highlight the difficulty in using a scalar kNN in terms of accuracy and computational cost, whilst also breaking old expectations with regard to small datasets. Furthermore, the quantifiable outputs from the deep learning method were sublime, notably with an accuracy of 95.8% with only 10% of the data to train with. In addition to the quantifiable results, this project also highlighted the potential power, and efficiency, in using complex deep learning models.

9. References

[1] A. Wijekoon, N. Wiratunga, and K. Cooper, ‘MEx - Multi-modal Exercise Dataset | IEEE DataPort’, *MEX - MULTI-MODAL EXERCISE DATASET*. https://ieee-dataport.org/open-access/mex-multi-modal-exercise-dataset (accessed Oct. 24, 2019).

[2] D. Wither and M. E. Manca, ‘EE581 Project Statement of Intent’. Dec. 04, 2019.

[3] N. Kumar, ‘The Professionals Point: Advantages and Disadvantages of KNN Algorithm in Machine Learning’, *The Professionals Point*, Feb. 23, 2019. http://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-knn.html (accessed Apr. 15, 2020).

[4] M. Brown and L. Rabiner, ‘An adaptive, ordered, graph search technique for dynamic time warping for isolated word recognition’, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 30, no. 4, pp. 535–544, Aug. 1982, doi: 10.1109/TASSP.1982.1163916.

[5] J. Brownlee, ‘How to Develop a Naive Bayes Classifier from Scratch in Python’, *Machine Learning Mastery*, Oct. 06, 2019. https://machinelearningmastery.com/classification-as-conditional-probability-and-the-naive-bayes-algorithm/ (accessed Apr. 15, 2020).

[6] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, ‘Time Series Classification Using Multi-Channels Deep Convolutional Neural Networks’, in *Web-Age Information Management*, vol. 8485, F. Li, G. Li, S. Hwang, B. Yao, and Z. Zhang, Eds. Cham: Springer International Publishing, 2014, pp. 298–310.

[7] N. Ackermann, ‘Introduction to 1D Convolutional Neural Networks in Keras for Time Sequences’, *Medium*, Sep. 14, 2018. https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf (accessed Apr. 15, 2020).

[8] ‘Regression with Keras’, *PyImageSearch*, Jan. 21, 2019. https://www.pyimagesearch.com/2019/01/21/regression-with-keras/ (accessed Apr. 15, 2020).

[9] ‘Keras, Regression, and CNNs’, *PyImageSearch*, Jan. 28, 2019. https://www.pyimagesearch.com/2019/01/28/keras-regression-and-cnns/ (accessed Apr. 15, 2020).

[10] A. Wijekoon, N. Wiratunga, and K. Cooper, ‘MEx: Multi-modal Exercises Dataset for Human Activity Recognition’, *arXiv:1908.08992 [cs, eess, stat]*, Aug. 2019, Accessed: Apr. 03, 2020. [Online]. Available: http://arxiv.org/abs/1908.08992.

[11] R. Kohavi, ‘A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection’, 1995, pp. 1137–1143.

[12] ‘duncan-wither/IVP-Human-Activity-Recognition’, *GitHub*. https://github.com/duncan-wither/IVP-Human-Activity-Recognition (accessed Apr. 16, 2020).

Appendix A – Simulation Results

Table 6. Complete Simulation Results

Method	Accuracy (%)						Training / Testing (%)
	1	2	3	4	5	Average	
ACT MLP	57.34	62.63	63.31	68.16	61.62	62.61	70 / 30
	56.18	58.65	65.74	54.33	43.09	55.60	50 / 50
	58.75	55.53	50.02	46.29	52.67	52.65	30 / 70
	55.34	47.74	46.22	49.32	49.77	49.68	10 / 90
	56.78	58.26	49.92	43.36	53.82	52.43	5 / 95
	44.34	30.65	31.30	32.60	39.09	35.60	1 / 99
ACW MLP	50.34	44.93	33.87	30.62	51.39	42.23	70 / 30
	35.75	29.90	43.20	47.92	32.86	37.93	50 / 50
	28.21	25.24	28.03	27.41	43.29	30.44	30 / 70
	25.09	27.70	29.00	25.09	28.10	27.00	10 / 90
	25.15	29.08	25.77	25.09	27.27	26.47	5 / 95
	25.05	21.73	26.51	25.88	20.36	23.91	1 / 99
DC CNN	99.06	99.39	98.94	98.90	99.03	99.06	70 / 30
	98.65	98.95	98.17	99.02	98.06	98.57	50 / 50
	97.90	98.25	97.43	96.85	98.34	97.75	30 / 70
	95.46	95.61	93.25	95.56	95.30	95.04	10 / 90
	93.61	93.37	93.39	92.71	93.70	93.36	5 / 95
	76.66	74.42	76.45	78.26	75.07	76.17	1 / 99
PM CNN	97.61	97.56	97.56	97.56	97.57	97.57	70 / 30
	97.43	97.36	97.40	97.38	97.31	97.38	50 / 50
	97.20	97.24	97.18	97.22	97.27	97.22	30 / 70
	95.90	95.24	95.63	96.05	96.04	95.77	10 / 90
	93.54	93.09	93.38	93.21	93.10	93.26	5 / 95
	78.95	77.44	78.18	79.93	77.09	78.32	1 / 99
MC-NN	99.81	99.90	99.60	98.74	99.65	99.54	70 / 30
	99.88	99.28	98.81	97.79	99.44	99.04	50 / 50
	97.52	97.24	99.31	98.87	98.97	98.38	30 / 70
	96.58	96.74	95.15	96.77	97.72	96.59	10 / 90
	85.37	92.29	93.5	93.72	91.89	91.35	5 / 95
	67.28	75.19	71.65	68.25	62.96	69.07	1 / 99