BOXING PUNCH CLASSIFICATION WITH ACCELEROMETER DATA

DANTE TOLLENAAR

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF SCIENCE IN COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE

DEPARTMENT OF
COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE
SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
TILBURG UNIVERSITY

STUDENT NUMBER

308941

COMMITTEE

dr. S. Ong
dr. G. Nápoles

LOCATION

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science &
Artificial Intelligence
Tilburg, The Netherlands

DATE

July 1, 2023

WORD COUNT

6629

ACKNOWLEDGMENTS

First and foremost, I would like to thank my thesis supervisor, dr. Sharon Ong, for her attentive supervision during the-four month period of writing this thesis. Moreover, I would like to thank my father for his valuable feedback and support.

BOXING PUNCH CLASSIFICATION WITH ACCELEROMETER DATA

DANTE TOLLENAAR

Abstract

This thesis project evaluates the performance of five Machine Learning algorithms on a human movement recognition task. Specifically, these algorithms were compared for their ability to classify boxing punches, captured by accelerometers. The acceleration of 7606 punches in the x, y and z axes, performed by eight professional boxing practioners, were used in a constructed Machine Learning (ML) pipeline and Deep Learning (DL) pipeline to train five supervised Machine Learning models and two supervised Deep Learning models. By extracting a total number of thirty features from the accelerometer data, we were able to train ML algorithms that can accurately classify boxing punches with a minimal error rate. Giving the sequential accelerometer data as input to the DL models yielded similar results. Additionally, we investigate whether standardizing the period length of accelerometer data, influences the performance of these models.

0.1 *Source/Code/Ethics/Technology Statement*

The SmartPunchDataset has been acquired from TheSmartPunchTeam (2019) through `https://www.kaggle.com/`. The obtained data is anonymised. Work on this thesis did not involve collecting data from human participants or animals. The original owner of the data and code used in this thesis retains ownership of the data and code during and after the completion of this thesis. However, the institution was informed about the use of this data for this thesis and potential research publications. All the figures belong to the author. The thesis code can be accessed through the GitHub repository following the link `https://github.com/DTollenaar/BoxingPunchClassification`. Part of the code has been adapted by the author from Wagner (2019). The reused/adapted code fragments are clearly indicated in the notebook.

1 INTRODUCTION

The automatic classification of boxing punches is a technology that holds value in two practical domains. Firstly, the technology is potentially useful in the minimization of damage to the long-term health of competitive fighters.

Miele and Bailes (2007) have researched whether the number and type of landed punches can be used to determine when a fight should be stopped, so that fatal damage is avoided. They found that as the competitiveness of the boxing matches increased, their method became less effective. Therefore, they called for further research of other methods that quantify the acceleration-deceleration of blows to the head. Research on the usability of accelerometers connects with such research.

Secondly, it can be valuable as an assistance technology for performance tracking of martial arts practitioners. Automatic classification of boxing punches can be useful for athletes to track their performance during training and competition. A reliable technique for detecting and recognizing thrown punches, opens the field up to more specific research, such as automatic detection of landed punches, or automatic movement efficiency analysis. In this niche domain of martial art punch classification, varying machine learning (ML) and deep learning (DL) techniques have been employed to solve the problem (Labintsev, Khasanshin, Balashov, Bocharov, & Bublikov, 2021; Wagner, Jäger, Wolff, & Fricke-Neuderth, 2019). In the broader domain of posture and general movements classification (Nunavath et al., 2021), yet more ML and DL approaches are applied to tackle similar problems. In these related studies, measurements of bodily movements are captured with accelerometers, which are a type of electromagnetic sensors, and one of the most commonly used methods of capturing bodily movements (Hindle, Keogh, & Lorimer, 2021). Other techniques that are often used are 3D optics and 2D video, which are able to capture a lot more information, but which both have implications. 3D optics is a robust measurement technique, however it must be used in a controlled environment and is also expensive, which is why it is only applied to restricted problems. 2D video measurements capture a lot of noise and are expensive to annotate (Hindle et al., 2021). A way to overcome the annotation problem of 2D video measurements of movement, would be to let a highly accurate ML model annotate the movements in a video, by processing accelerometer data that was gathered in parallel with the video data.

As of now, there exist few studies which extensively compare the performance of machine learning and deep learning classifiers on a standardized dataset of accelerometer-measured punches. Hence, a study that evaluates

the best approaches of all these studies, contributes to the field of automated movement classification as a whole. With regards to the supplied motivations, this study evaluates multiple machine learning and deep learning techniques on accelerometer data, to classify boxing punches. The movement of the human arm can be measured by a wrist-worn, triaxial accelerometer in for example smart watches. As mentioned before, accelerometers are devices which contain electromechanical sensors that measure acceleration. The specific type of sensors that are used to detect non-uniform movements, like a punch, are called dynamic accelerometers. Some of the meters used in the literature, have three degrees of freedom, which results in measurement of acceleration into the x, y and z direction. There also exist measurement devices with, for example, six degrees of freedom, which can detect the rotation (pitch, yaw, roll) of the device by incorporating a gyroscope. However the literature shows three degrees of freedom sufficiently represents the target movements for classification purposes (Wagner et al., 2019). The time series data that these sensors yields hold representations of the movements made. With the employment of pre-processing and machine learning techniques, features can be extracted from this data and used in classification models. All with the purpose of detecting the three broad types of boxing punches: the straight punch, the hook, and the upper-cut.
This study is based on the following research question.
"To what extent can boxing punches (straight punch, hook, and upper-cut) be classified with temporal data measured by a wrist-worn, tri-axial accelerometer?"
Additionally, the effect of normalizing the period length of the temporal data on classifier performance will be investigated. To this end, a sub question is formulated:
SQ1: "How does normalizing the period length of the time series data influence the performance of the machine learning and deep learning classifiers?"
Lastly, we are interested in what features from the data have the most impact on classifier performance. Therefore, our second sub question is:
SQ2: "What is the impact of the individual features on the performance of the machine learning classifiers?"

2 RELATED WORK

2.1 *Classification of martial art punches*

Wagner et al. (2019) are the creators of the dataset that will be used in this study. Their approach to classifying boxing punches involves extracting a total of 24 features from the time series data, and use these to train ma-

chine learning algorithms. Of the algorithms, their random forest decision tree with 120 estimators proved to be the best performing on the data (98 percent accuracy). Something interesting they have found, is that the period length and the sampling rate of the data prior to pre-processing, can heavily influence the performance of the model.
Similar to Wagner et al. (2019), a study was conducted by Labintsev et al. (2021) to classify karate punches measured by accelerometers. Their approach did not involve feature extraction in the pre-processing phase, yet they were still able to achieve similar performance (96 percent accuracy) by employing a multilayered, 2D convolutional neural network. Additionally, their dataset was three-and-a-half times smaller (1912 annotated instances) than that of Wagner et al. (2019).
On another note, Worsey, Espinosa, Shepherd, and Thiel (2020) evaluated machine learning algorithms and two wearable intertial sensor configurations on the automatic boxing punch classification task. Their sensor devices captured the movement in nine dimensions, by incorporating an accelerometer, a gyroscope and a magnetometer. In their configurations, the devices were worn on both wrists, and in configuration two, and additional device was worn on the back in between the shoulder blades. In their experiments, Worsey et al. (2020) extracted features with principal component analysis (PCA) and evaluated a total of six machine learning classifiers on the features. In their experiment with configuration one, the gaussian support vector machine logistic regression algorithms performed the best (accuracy score of 0.96). In experiment two, their multi-layered perceptron had the best accuracy score (0.98).

2.2 *Classification of general movements*

Wang and Wu (2016) studied the classification of human posture and general movements measured by wrist- and hip-worn accelerometers. With their annotated dataset of 175000 measured movements, they evaluated the performance of Gaussian Discriminant Analysis (GDA), Support Vector Machine (SVM) and K-means classification. This research concluded that the use of unsupervised methods like the K-means classifier were not suitable for this problem. In addition, it was found that the GDA was more efficient with regards to learning from a smaller training set, but that the SVM performed the best on their dataset.
Kerr et al. (2015), Ellis, Kerr, Godbole, Staudenmayer, and Lanckriet (2015), and Rosenberg et al. (2016) all classified one-minute windows of everyday movements measured by an accelerometer, with a decision tree algorithm. Their pre-processing technique involved extracting a total of 41 features from the one minute windows.

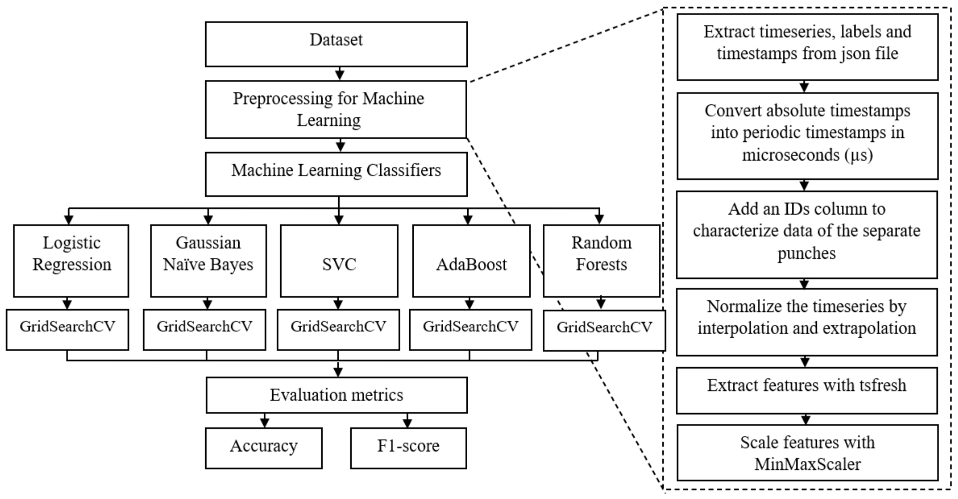
Dataset
Preprocessing for Machine Learning
Machine Learning Classifiers
Logistic Regression
Gaussian Naïve Bayes
SVC
AdaBoost
Random Forests
GridSearchCV
Evaluation metrics
Accuracy
F1-score
Extract timeseries, labels and timestamps from json file
Convert absolute timestamps into periodic timestamps in microseconds (μs)
Add an IDs column to characterize data of the separate punches
Normalize the timeseries by interpolation and extrapolation
Extract features with tsfresh
Scale features with MinMaxScaler

Figure 1: The experimental workflow of the Machine Learning pipeline.

Another study on the classification of general movements conducted by Nunavath et al. (2021), involved deploying a Deep feed forward Neural Network (DNN) and a Recurrent Neural Network (RNN). Their dataset, consisting of hip-worn and wrist-worn measured movements, was pre-processed into time windows of 3, 5 and 10 seconds. They found that a fine-tuned RNN performed better when evaluated on F1-score and accuracy, than a DNN. Also, their networks performed best when trained and evaluated on data that was pre-processed into 10 second windows.
In general, studies surrounding general movement classification use time windows of longer accelerometer recordings. The recordings are long (in the temporal dimension) because much of these studies are conducted for the purpose of automatic patient monitoring, which in many cases means that behaviors of the patient need to be recognized in periods of hours to days. Nevertheless, these studies still provide insights into what methods for automatic movement classification work well. Overall, we learn that universally, SVMs and random forest classifiers are among the best performing classifiers in this domain's literature. Furthermore, multiple studies report promising performance scores by RNNs and CNNs (Labintsev et al., 2021; Nunavath et al., 2021).

3 METHOD

3.1 *Experimental workflow*

The ML experimental workflow is divided into four broad steps (figure 1). To begin, the dataset is pre-processed for machine learning use. In

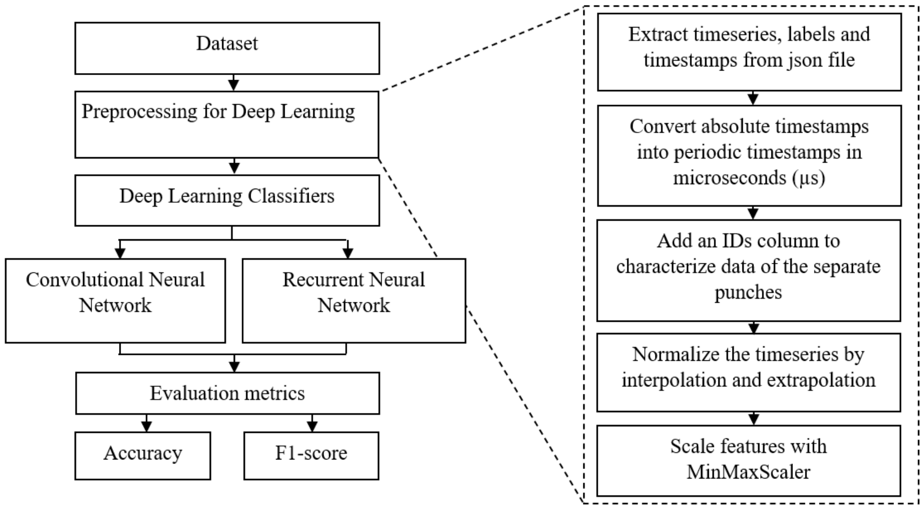
Dataset
Preprocessing for Deep Learning
Deep Learning Classifiers
Convolutional Neural Network
Recurrent Neural Network
Evaluation metrics
Accuracy
F1-score
Extract timeseries, labels and timestamps from json file
Convert absolute timestamps into periodic timestamps in microseconds (μs)
Add an IDs column to characterize data of the separate punches
Normalize the timeseries by interpolation and extrapolation
Scale features with MinMaxScaler

Figure 2: The experimental workflow of the Deep Learning pipeline.

this step, we also normalize the period length and sampling rate of the timeseries. The processed data is then used to train five different machine learning classifiers. We find the best hyperparameters for the individual classifiers by performing grid search with cross validation. And at last, the performance of the tuned classifiers is evaluated on their f1 and accuracy score.
The DL pipeline will be set up on the Google Colaboratory cloud platform (Bisong, 2019), to make use of their freely available GPU resources.The experimental workflow for the deep learning experiments (figure 2) differs in two steps. Firstly, during pre-processing, we do not extract any features with tsfresh. Instead, the x, y, and z time series are appended after one another and padded with zeroes (when that is necessary in the raw data case). Secondly, we do not perform grid search when tuning the hyperparameters. Such optimization techniques can be computationally expensive in the case of deep learning models, and since we have limited GPU resources from Google Colaboratory, we chose to approximate the DL hyperparameters by trial and error.

3.2 *Data*

The dataset used, consists of 7606 instances of thrown punches (figure 3), and was created by Wagner et al. (2019) in their research to classify boxing punches. The movement was measured by a tri-axial accelerometer that is part of a smartwatch, which was worn on both wrists of the participants (see figure 4 for reference). The time series data of the movements is

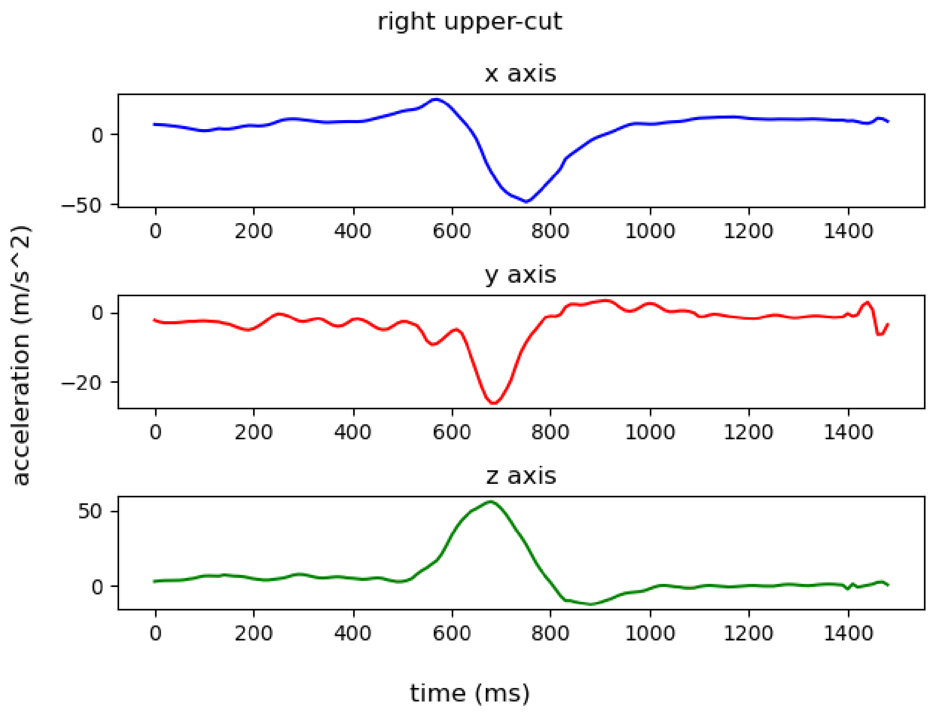

Figure 3: A generic instance of a punch.

annotated with a label of the punch (target values: *frontal, hook, upper-cut or no-action*), the hand (target values: *left or right*), and the annotator (target values: integers 0-7, anonymously representing the participants). The distribution of the punch labels used in this study is almost balanced (see figure 5). The punches were measured with a sampling rate of 100 Hz (1 observations every 10 milliseconds). On average, the punch recordings are 1550 milliseconds long (median length is 1330 ms).

3.3 *Pre-processing for Machine Learning*

The first step of pre-processing is to extract the timeseries, punch labels and timestamps and store them in a pandas (Wes McKinney, 2010) dataframe. The timestamps come represented in microseconds relative to the Unix time. These timestamps are converted into periodic timestamps, so that every punch instance starts at timestamp 0. Thereafter, an ID column is added to the dataframe that characterizes which timestamp observation belongs to which punch instance. Then, the period length of the data is normalized. This is achieved by interpolating and extrapolating the time series. The pre-processing stage is concluded by extracting a set of minimal features (table 1) with the tsfresh library (Christ, Braun, Neuffer,

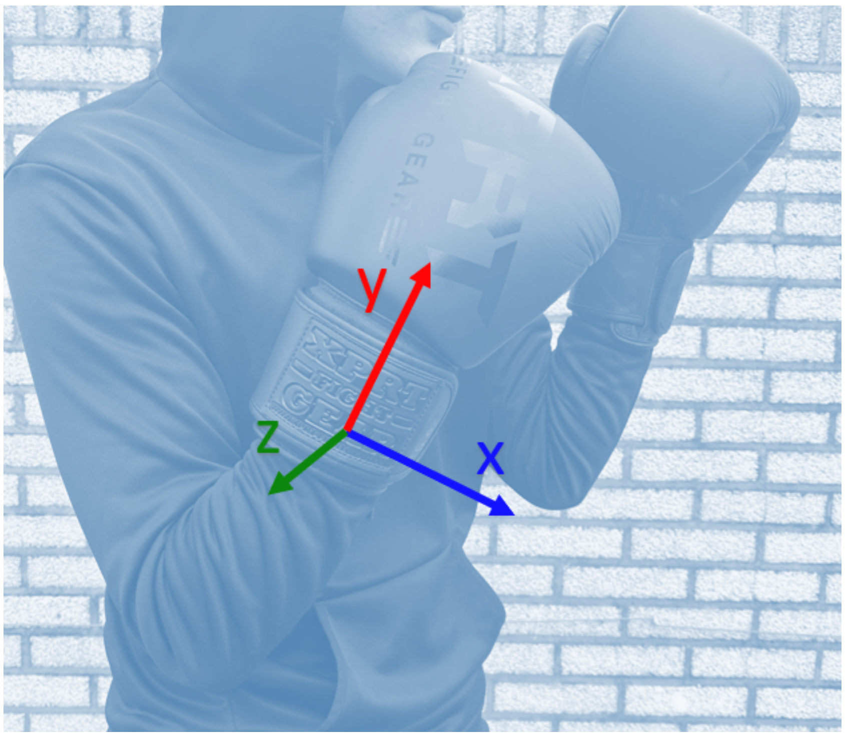

Figure 4: The orientation of the accelerometer axes.

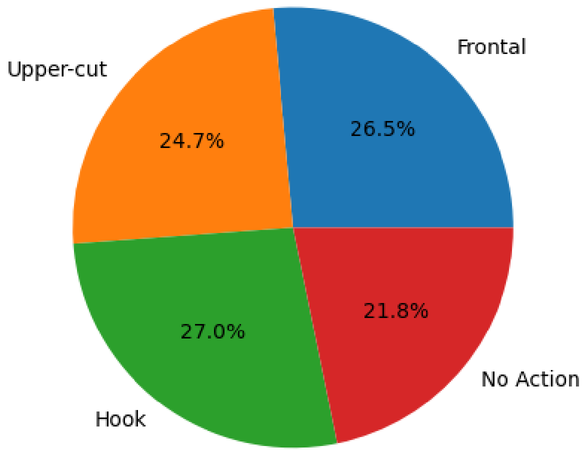

Figure 5: The distribution of the labels.

& Kempa-Liehr, 2018), and scaling these features into the range of 0-1. At this point, we are left with a dataframe that contains 7606 rows (the total number of punches) with 30 features on the columns (10 features per accelerometer axis). This feature set was chosen over other feature sets in the tsfresh library, mainly for its interpretability.

Table 1: The features extracted by tsfresh.

x__sum_values	y__sum_values	z__sum_values
x__median	y__median	z__median
x__mean	y__mean	z__mean
x__length	y__length	z__length
x__standard_deviation	y__standard_deviation	z__standard_deviation
x__variance	y__variance	z__variance
x__root_mean_square	y__root_mean_square	z__root_mean_square
x__maximum	y__maximum	z__maximum
x__absolute_maximum	y__absolute_maximum	z__absolute_maximum
x__minimum	y__minimum	z__minimum

3.4 *Feature Importance*

Feature importance allows an insight into which features have the most predictive power in the classification task. To retrieve feature importance information from different classifiers, this research will utilize the unified prediction interpretation framework SHAP (SHapley Additive exPlanations), presented by Lundberg and Lee (2017). SHAP uses the Shapley value (Rozemberczki et al., 2022), which is a concept originating in cooperative game theory, that aims to capture the contributions made by individual players. In SHAP, a features Shapley value is calculated by generating subsets of features, computing the contribution of the feature by permuting it from every subset, and then averaging the contributions of the feature to each subset. This calculation can become quite expensive when there are many features, therefore various approximation techniques can be utilized by SHAP to reduce computation cost (e.g. model-agnostic methods such as kernel SHAP or model-specific methods like Tree SHAP, (Lundberg & Lee, 2017)). One of the reasons these approximation methods are successful, is because they assume feature independence.

3.5 *Machine Learning Classifiers*

Multiple machine learning classifiers were evaluated, to get a broader sense of how well boxing punches can be recognized. From the literature, the random forest (Breiman, 2001) classifier was picked as it showed to be the best performing. Random forest classifiers tend to be robust to multidimensional data, which is advantageous in our case. The adaptive boosting classifier was chosen as another ensemble technique, to find out whether it performs better than the random forest classifier. We will look at the performance of a support vector machine (Cortes & Vapnik, 1995), as it also responds well to high-dimensional data. To diversify the considered classifiers, Gaussian Naive Bayes was added to the evaluation. And finally, the logistic regression algorithm is included, as it is considered good practice to evaluate simpler models. The linearity of the model is another reason for its inclusion. All machine learning models were implemented using the Scikit-Learn library (Pedregosa et al., 2011).

3.5.1 *Logistic Regression*

We will start with the simplest of the machine learning classifiers. Logistic regression is a supervised classification model that natively is used for binary classification. It works by using the logistic function to transform the output into a probability between 0 and 1. In our case, we want to predict four classes, not just two. To enable multi-class classification, Scikit-Learn has the option to use the one-vs-rest heuristic. Here, Scikit-Learn trains a separate logistic regression model for each class, that predicts whether a data point belongs to that class or not (hence, one-versus-rest). Additionally, we let Scikit-Learn apply the L2 regularization term, with the purpose of discouraging the model to overfit. L2 regularization (or ridge regularization) adds a penalty term to the loss function of the model during training, that leads the model to find the most efficient parameters for the coefficients and discourages it to have high magnitude coefficients. Large coefficients tend to cause a high reliance on certain features or noise, which is indicative of overfitting. In training, we will use gradient search to find the best regularization parameter C for our logistic regression model.

3.5.2 *Gaussian Naive Bayes*

The Gaussian Naive Bayes classifier works by calculating the probabilities of the input features, to predict an instances class. The classifier makes two assumptions. Firstly, it assumes the feature values are normally (Gaussian) distributed within each class. In practice, this assumption results in a preference for feature values that are closer to the mean. Scikit-Learn offers the option to perform variance smoothing, which is a technique that smooths the Gaussian curve by adding a pre-defined variance. This way, we even the playing field for features that are further from the mean. The optimal variance smoothing parameter will be sought for during gradient search. The model also assumes independence between variables, which covers the naive part of the classifiers name. However the model tends to yield good performance scores, regardless of whether the variables are independent.

3.5.3 *Support Vector Machine*

The third in our line of Machine Learning classifiers is the support vector machine (SVM). This classifier attempts to draw a decision boundary (hyperplane) between features belonging to different classes. When this is not possible, it utilizes a kernel to map the features into a higher dimension, where they are potentially separable. Different kernels can be used for this mapping, such as a linear kernel, a radial basis function (rbf), or a sigmoid kernel. For cases where the classes are still not separable by a hyperplane, SVMs employ a soft margin, which allows for a few of the closest features to cross the hyperplane up to a certain margin. These features form the support vectors and are used to determine the optimal hyperplane. How far the margin stretches and how many errors are allowed within it, is regulated by the regularization parameter C. During grid search, both the kernel and the regularization parameter will be evaluated. For the SVM, we also use the one-vs-rest heuristic to shape the decision function.

3.5.4 *Adaptive Boosting*

Now we arrive at the ensemble techniques, first of which is the Adaptive Boosting classifier. The algorithm functions by iteratively training a set of weak learners on the data, and assigning a weight to each learner based on its performance. Every iteration, a new weak learner is trained on the data samples that the others failed to classify. Also at the end of a iteration, the weights of the learners are recalculated. When evaluated on new data, the ensemble model will perform weighted voting to determine its prediction. During hyperparameter tuning, we will evaluate three hyperparameters. Those are the base estimator, which in our case are decision trees with either one, two or three splits. The learning rate, which controls how much the weak learners contribute. And the number of estimators that AdaBoost will create before the training process is terminated.

3.5.5 *Random Forest*

And the last model to be evaluated is the Random Forest classifier, which is also an ensemble technique. The models base estimators are decision trees. When decision trees are trained very deep (e.g. they have many splits), they will overfit, which leaves them with a low bias and a high variance towards their training data. Random forests takes advantage of this by combining many of these deep decision trees, which increases the bias of the trees by a bit, but yields much greater performance. If all the trees would be trained on the same training data, they would be highly correlated, and the ensemble model would still have a high variance. Therefore random forests uses the technique bootstrap aggregating, or bagging. Hereby, the algorithm takes sub samples of the training data with replacement, and assigns these to separate trees to be trained on. And lastly, to avoid that the trees become correlated because of a small set of stronger predictors within the feature set, random forests performs a technique called feature bagging, which takes random sub samples of the features and assigns these to each tree. When the ensemble is fully trained, it makes predictions by taking the majority vote of the predictions of all trees. The hyperparameters that will be tuned for random forests, are the number of estimators, the maximal depth that the estimators may grow, and the function used to determine the quality of a tree split.

3.6 *Deep Learning classifiers*

One advantage of deep learning models, is that they are strong representation learners. DL models possess the ability to learn what features to extract from the raw data. And by employing multiple connected layers, they learn to extract abstract features in the continuing layers. There are different types of deep learning layers, with varying applications. Because of the temporal component of the data in this study, preference is given to layers that capture contextual information during feature extraction. Both convolutional layers and recurrent layers meet this preference, each in their own way. Labintsev et al. (2021) and Nunavath et al. (2021) showed that these deep learning layers perform well in the classification task of human movements measured by inertial accelerometers. In the second experiment of Worsey et al. (2020), promising results are reported for the multi-layered perceptron, which is a simple type of deep neural network. We are not expecting to yield the same results with a DNN, because their data dimensions and pre-processing steps differ greatly from ours. Even so we might be able to harness some of the power of DNNs, by incorporating fully connected layers into our neural networks, as is often done in neural network development. We will be developing two DL models with PyTorch (Paszke et al., 2019). In the two following sections, we cover the general functioning of the convolutional and recurrent layers in neural networks.

3.6.1 *Convolutional Neural Network*

The foundation of a convolutional neural network (CNN), is the convolution operation. When the convolutional layer receives the input sequence, it convolves the sequence with a 'kernel', which is another value sequence of size k. In general terms, what convolving the sequence with the kernel does, is flipping the kernel in all its dimensions, overlapping the kernel at each input sequence point, and calculating the overlapping area. As such, the convolution operation takes the neighboring values of a data point into account during processing (local receptive field). In a CNN, the size of the kernel is chosen beforehand, but the values in the kernel are trained. Therefore a CNN can learn that for example, high transitions (large difference) in neighboring values, are discriminative features. In a convolutional layer, we typically train multiple kernels in parallel, so that we can extract varying features per layer. The layer outputs a feature map for each kernel, which are called channels. CNNs can have multiple convolution layers, which can increase in both the number of channels, and the size of the receptive field (increasing kernel size).
A mathematical property of convolution is shift invariance, which is useful as the punches in the accelerometer data are not necessarily aligned.

3.6.2 *Recurrent Neural Network*

Recurrent layers process the input tensor sequentially, and are able to remember previous elements in the sequence by utilizing a 'hidden state'. Every step, information from the current input is extracted and used to update the hidden state. Such a hidden state holds information about sequence elements and their preceding neighbors, essentially capturing temporal dependencies. The hidden state is passed onto fully connected layers in the RNN, where its contents are used to generate an output.
A frequent problem with RNNs as described before, is the vanishing gradient problem. Broadly speaking, this problem occurs when long sequences are used to train a RNN. As the network stores more and more information of long-distance points in the hidden layer, the actual information becomes very small and the gradient disappears. Hochreiter and Schmidhuber (1997) presented the LSTM (Long Short-Term Memory)

model as a solution to the problem. The main novel concept of the LSTM is the memory cell, which manages what information to remember and what information to discard (or forget) over time. It does so by combining four components. One of which is the cell state, that functions as an information carrier for previous time steps and that can be updated and modified. Information flow through the layer is managed by gates, which come in three variants, namely the forget gate, that determines what information from the earlier cell state to discard; the input gate, that determines what information from the input combined with the earlier hidden state to pass; and the output gate, that functions as a final filter on the cell state, as it is passed onto the new iteration. What solves the vanishing gradient problem is the LSTM's selectivity of information history.
As this study's raw punch data has a temporal dimension, a LSTM can use that aspect to its advantage.

3.7 *Hyperparameter tuning*

The GridSearchCV algorithm from Scikit-Learn is used to find the best hyperparameters per Machine Learning model. This algorithm splits the training data into five folds, and uses one of the folds to validate the models, while it fits the models with the other four. It does this for each fold, then chooses the hyperparameters that scored best out five, based on the accuracy performance of the models.

3.8 *Evaluation*

The tuned models' performances will be evaluated on the accuracy score and the F1 score. Every model will be trained on several versions of the dataset, which are normalized to different period lengths. The raw dataset will be used as a baseline version.
To determine feature importance, the best performing model of each machine learning technique will be used to extract the SHAP values. These model-specific SHAP values will then be compared to each other, using the mean SHAP value of each feature.

4 RESULTS

To evaluate how well Machine Learning models can recognize boxing punches measured by an accelerometer, we trained and tested seven classifiers on the dataset. Out of the seven, the Random Forest classifier performed best, with an accuracy score of 0.995 and a F1 score of 0.995 (see

Table 2: The best score of each model after hyperparameter tuning. 'Raw' indicates the data was not normalized with respect to the period length.

Period length (ms)	Models	Accuracy	F1-score
Raw	Logistic Regression	0.984	0.984
Raw	Gaussian NB	0.886	0.887
Raw	SVM	0.993	0.994
1500	AdaBoosting	0.962	0.963
Raw	Random Forest	0.995	0.995
2000	CNN	0.983	0.979
2500	RNN	0.911	0.891

table 2). Closely followed by the Support Vector Machine classifier with an accuracy of 0.994 and a F1 score of 0.993. Apart from the Gaussian Naive Bayes and the RNN classifier, all the classifiers were able to yield scores of above 95%.
By analyzing the confusion matrices of the classifiers (figures 6 and 7), we observe that the hook and upper-cut classes are most often confused with one another, by all classifiers except for the recurrent neural network.

4.1 *Hyperparameter tuning results*

During grid search, multiple values for the hyperparameters of the classifiers were evaluated (table 3). The hyperparameters of the best classifiers are displayed in table 3. Note that the best regularization parameter C of Logistic Regression and the Support Vector Machine, and the best number of estimators of AdaBoost are the maximally evaluated hyperparameters. Because of the before mentioned reason, the structure and hyperparameters of the DL models were chosen by trial and error. The CNN consists of three convolutional layers, followed up by three fully connected layers (figure 4). The sequences were padded with zeroes on both sides to minimize information loss. The best kernel sizes were found to be five, five and seven in layer order. Neither increasing stride nor dilation of the kernel had a positive effect on the CNN's performance. In between the convolutional layers, batch normalization and the ReLU activation function are applied. We also use ReLU for the fully connected layers, as well as dropout with a probability of 0.01. Dropout sets a part (here 1%) of the neurons to zero every forward pass, to avoid overfitting. L2 regularization is also applied in the form of weight decay, albeit a small amount (probability of 0.0001). The RNN consists of a single LSTM layer with a hidden state of size

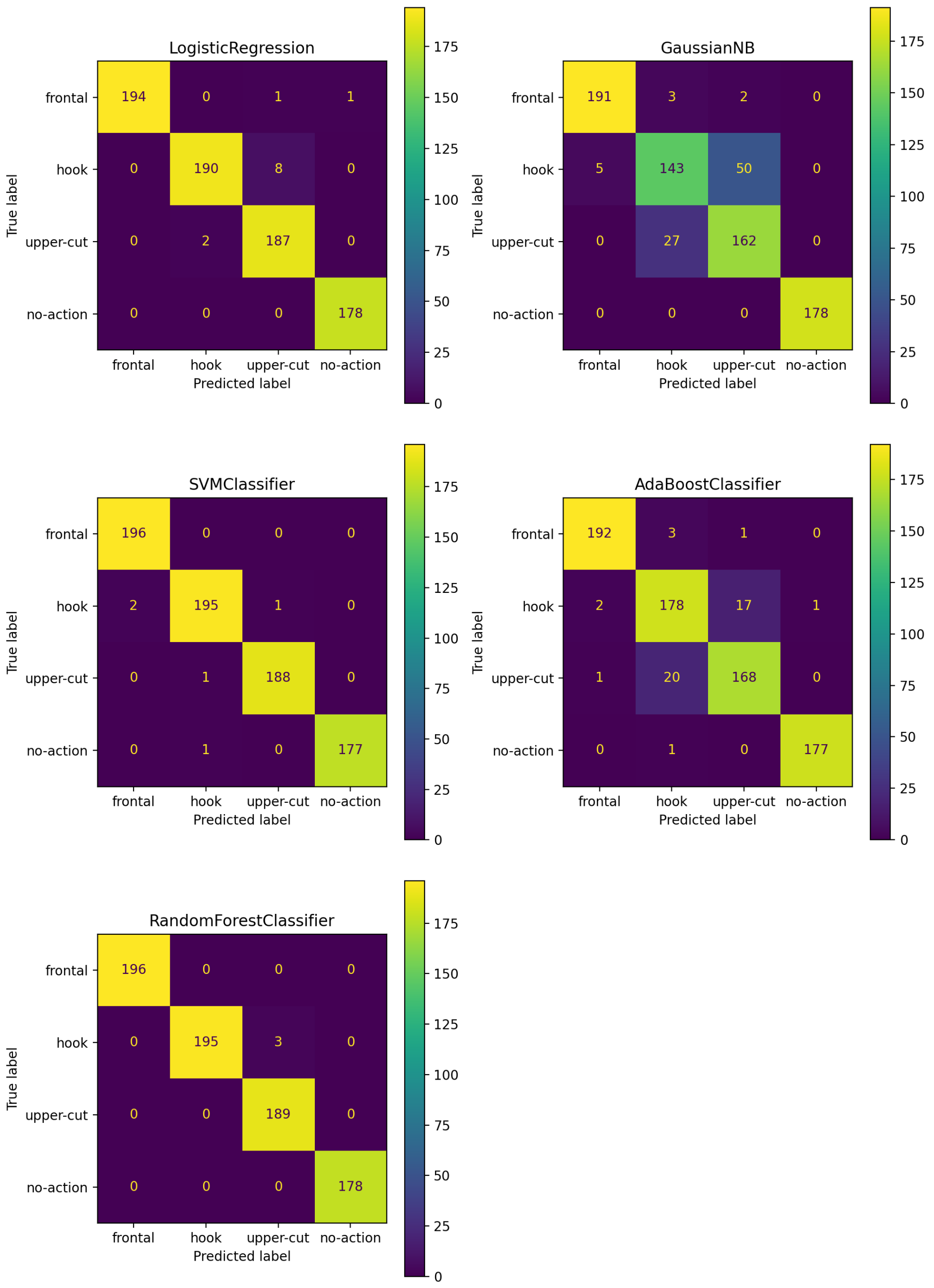

Figure 6: Confusion matrices for the highest performing, tuned version of each machine learning classifier.

Table 3: The best hyperparameters for each machine learning model. Determined by performing grid search with cross validation.

Models	Evaluated hyperparameters	Best hyperparameters
Logistic Regression	C : 0.001, 0.01, 0.1, 1, 10, 100	C = 100
Gaussian NB	var smoothing : 1, 1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6, 1e-7, 1e-8, 1e-9	var smoothing = 1.e-4

SVM	C : 0.001, 0.01, 0.1, 1, 10, 100
	kernel : linear, rbf, sigmoid
		C = 100
		kernel = rbf

AdaBoost	estimator : tree(max depth=1)*, tree(max depth=2), tree(max depth=3) learning rate : 0.1, 1, 10 n estimators : 20, 65, 110, 155, 200	estimator = tree(max depth=3) learning rate = 1 n estimators = 200
Random Forest	criterion : entropy, gini, log loss max depth : 20, 40, 60, 80, 100 n estimators : 20, 50, 80, 120, 200, 500, 800	criterion = entropy max depth = 80 n estimators = 120

**tree is an abbreviation for the decision tree classifier.*

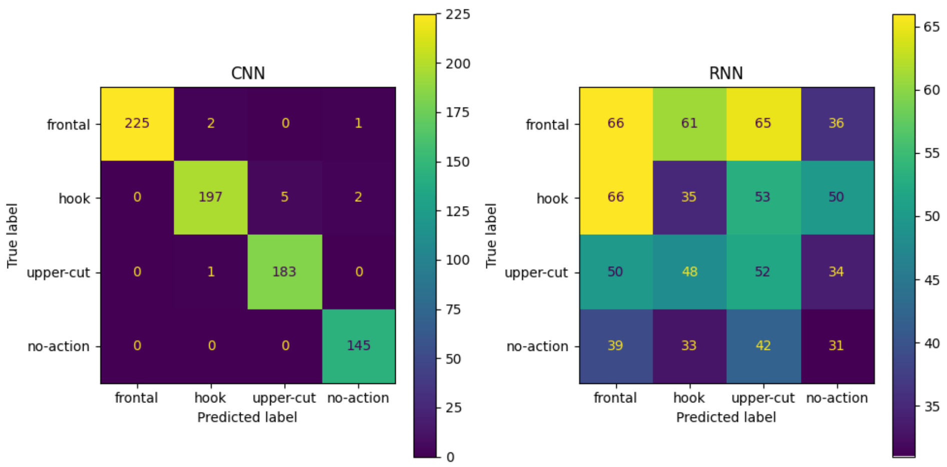

Figure 7: Confusion matrices for the highest performing deep learning classifiers.

128 (figure 5). Increasing the amount of recurrent layers did not seem to improve performance. After The LSTM layer, dropout is applied with a probability of 0.01. Thereafter come two fully connected layers, one of size 256 and one of size 128. In between the layers, ReLU activation is applied, however no dropout. The output is passed through the Softmax activation function. The same amount of weight decay (0.0001) is used during the training process of both DL models.
In training the CNN and the RNN, we used cross entropy loss and the Adam optimizer. All DL models were trained for one-hundred epochs, with early stopping set to five epochs. A learning rate of 0.0001 and a batch size of 16 were found to work the best.

Table 4: CNN model architecture.

Layer	Kernel	Padding	Channels	Parameters
Conv1D	5	3	8	48
MaxPool1D(2)	-	-	-	0
ReLU	-	-	-	0
BatchNorm	-	-	-	16
Conv1D	5	3	16	656
MaxPool1D(2)	-	-	-	0
ReLU	-	-	-	0
BatchNorm	-	-	-	32
Conv1D	7	4	32	3616
MaxPool1D(4)	-	-	-	0
ReLU	-	-	-	0
BatchNorm	-	-	-	64
Linear	-	-	-	369968
ReLU	-	-	-	0
Dropout(0.01)	-	-	-	0
Linear	-	-	-	11590
ReLU	-	-	-	0
Dropout(0.01)	-	-	-	0
Linear	-	-	-	156
Softmax	-	-	-	0

4.2 *Period length experiment*

In pursuit of answering the first research sub-question regarding normalization of the period length, we normalized the data to various period lengths. As such, all classifiers were evaluated on each of the normalization cases, and on the raw data. The results of these experiments are displayed

Table 5: RNN model architecture.

Layer	Hidden dim	Parameters
LSTM	128	452096
Dropout(0.01)	-	0
Linear	-	8256
ReLU	-	0
Linear	-	260
Softmax	-	0

in tables 6-12. The only machine learning model which seems to benefit from the normalization technique, is the adaptive boosting classifier (table 9). Either a period length of 1300 or 1500 milliseconds achieve results of a 0.96 accuracy and F1 score, 0.02 points higher than for the raw data case. As for the other machine learning classifiers, the raw data experiments outperform the normalized period length experiments. The differences between the highest scores and second highest scores for logistic regression, Naive Bayes, Support Vector Machine, and Random Forest, are 0.05, 0.09, 0.03, and 0.01, respectively. Contrarily, the deep learning models do perform better when the input data is normalized to a period length. The difference between scores on the raw data and on the normalized data, is more than 0.01 for both the convolutional neural network and the recurrent neural network.

4.3 *Feature importance*

The second component we investigated, is the importance of the extracted features. As mentioned before, the importance is measured with the SHAP method. SHAP has been implemented as a python package by Lundberg and Lee (2017), as such this package is used in the experiment. We set the algorithm setting to 'auto', which lets the package decide which Shapley value approximation algorithm to use for which model. The SHAP value of a feature is approximated at each sample. We will be looking at the absolute mean SHAP value of each feature, for each classifier (see figure 9). The correlation matrix for the features is plotted in figure 8.
On first sight, it is observed that 'z_mean' is the most important feature for all machine learning classifiers, except the Gaussian Naive Bayes, for which it is 'z__sum_values'. The second placed features differs for the classifiers. For Random Forest and Support Vector Machine, it is 'z__sum_values' while for Gaussian Naive Bayes it is 'z_mean', for Logistic Regression it is 'x__root_mean_square', and for AdaBoost it is 'y__sum_values'. Going further down the feature ranking, there are no features which rank the

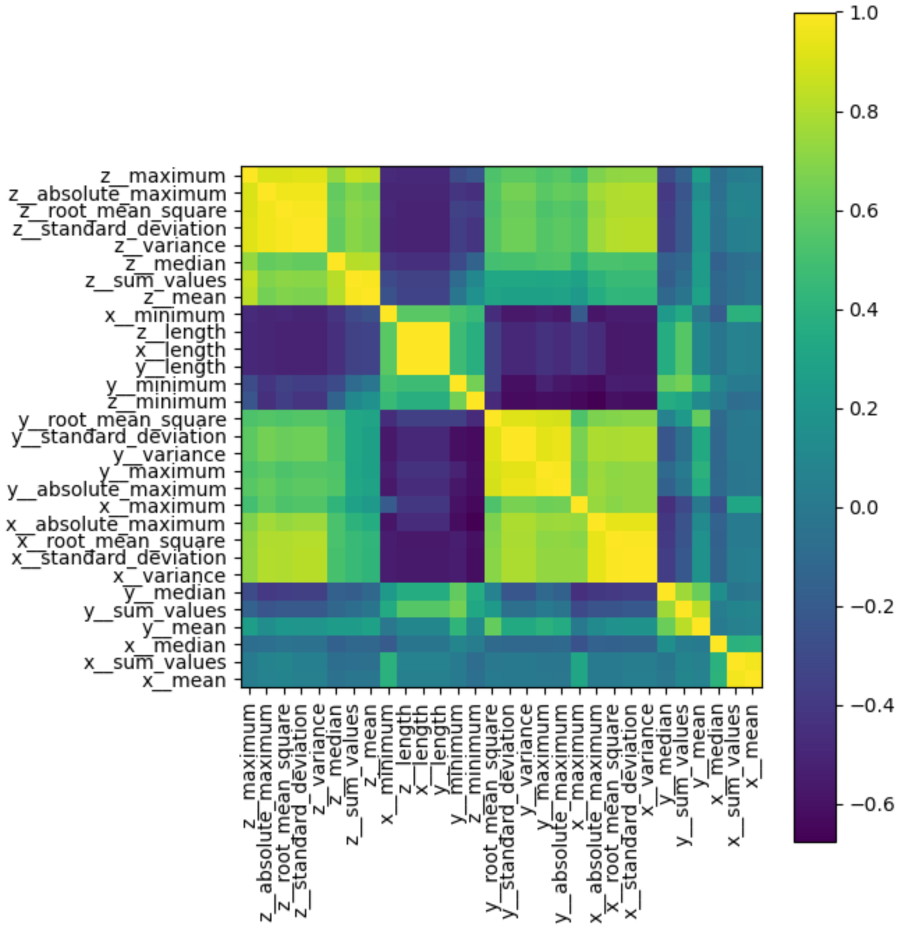

Figure 8: Correlation matrix of the features extracted by tsfresh.

same for each classifiers. However, there is similarity between the feature preference of the ML classifiers. 'z__mean' and 'z__sum_values' (which have the highest and second highest absolute mean SHAP value averaged over all classifiers) occur in the top five features of every classifier.

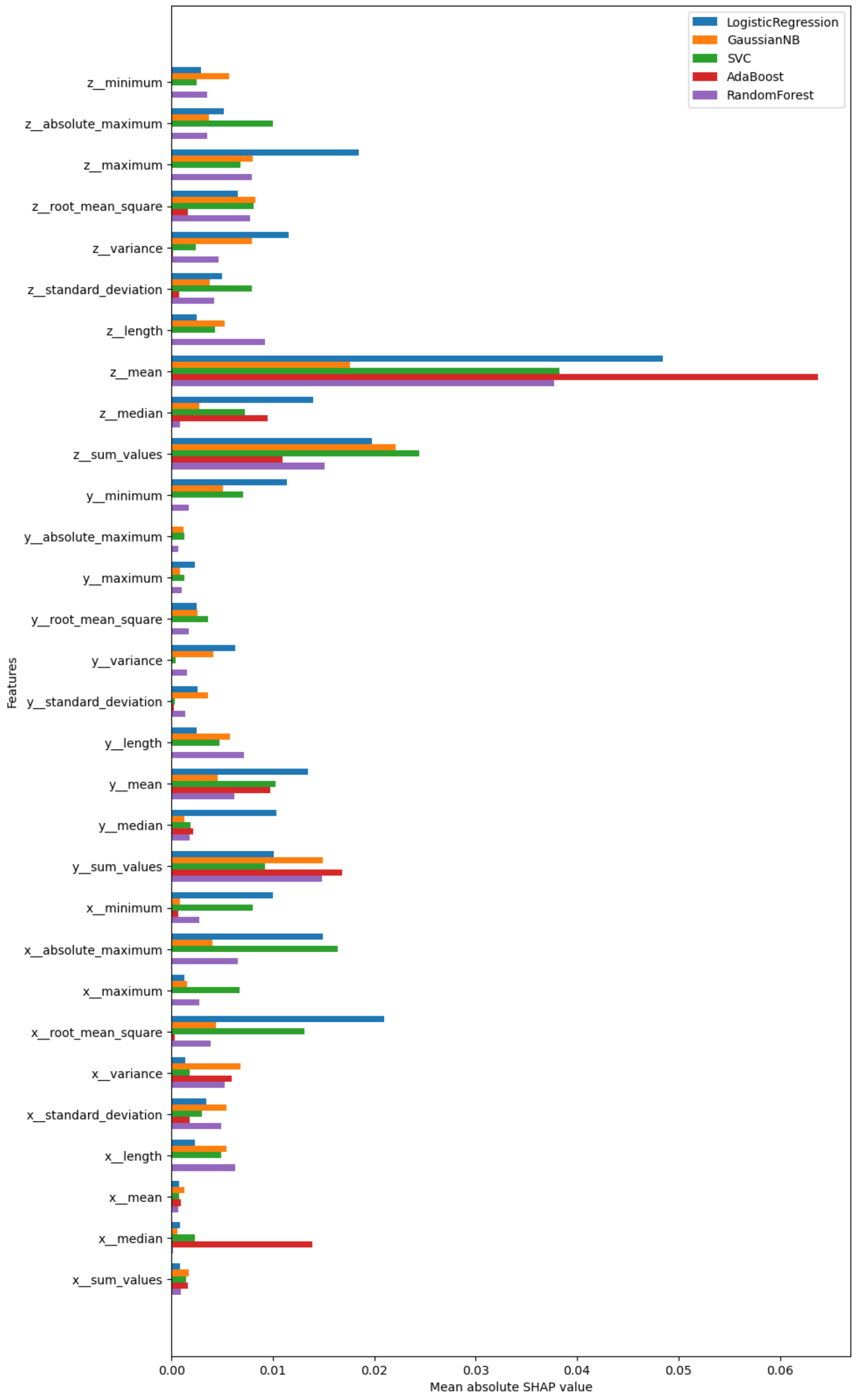

Figure 9: The feature importance of each classifier.

Table 6: Results for the Logistic Regression classifier.

Period length (ms)	Accuracy	F1-score	C
raw	**0.984**	**0.984**	100
1000	0.926	0.927	100
1300	0.921	0.922	100
1500	0.932	0.933	100
1800	0.922	0.923	100
2000	0.907	0.908	100
2200	0.921	0.922	100
2500	0.916	0.917	100

Table 7: Results for the Gaussian Naive Bayes classifier.

Period length (ms)	Accuracy	F1-score	Var smoothing
raw	**0.886**	**0.887**	0.0001
1000	0.796	0.771	0.0001
1300	0.773	0.737	0.0001
1500	0.761	0.719	0.001
1800	0.752	0.704	0.001
2000	0.745	0.694	0.001
2200	0.736	0.681	0.001
2500	0.712	0.652	0.001

Table 8: Results for the Support Vector Machine classifier.

Period length (ms)	Accuracy	F1-score	C	kernel
raw	**0.993**	**0.994**	100	rbf
1000	0.947	0.948	100	rbf
1300	0.963	0.964	100	linear
1500	0.961	0.961	100	rbf
1800	0.957	0.957	100	rbf
2000	0.958	0.959	100	rbf
2200	0.955	0.956	100	rbf
2500	0.955	0.956	100	rbf

5 DISCUSSION

This study aimed to uncover to what extent boxing punches, measured by wrist-worn tri-axial accelerometers, could be classified. In support of that research question, we investigated how normalizing the period length of the time series data affected classifier performance, and we sought to find which features extracted for machine learning use, contributed the most in

Table 9: Results for the Adaptive Boosting classifier.

Period length (ms)	Accuracy	F1-score	Estimator	Learning rate	n estimators
raw	0.941	0.942	tree(3)	0.1	20
1000	0.933	0.934	tree(3)	1	200
1300	0.961	0.961	tree(3)	1	200
1500	**0.962**	**0.963**	tree(3)	1	200
1800	0.951	0.952	tree(3)	1	200
2000	0.943	0.945	tree(3)	1	200
2200	0.93	0.932	tree(3)	1	200
2500	0.933	0.934	tree(3)	1	155

**tree(3) is an abbreviation for the decision tree classifier with the max depth parameter set to three.*

Table 10: Results for the Random Forest classifier.

Period length (ms)	Accuracy	F1-score	Criterion	Max depth	n estimators
raw	**0.995**	**0.995**	entropy	80	120
1000	0.979	0.979	gini	40	800
1300	0.986	0.986	gini	100	500
1500	0.986	0.986	entropy	20	80
1800	0.988	0.988	gini	60	200
2000	0.986	0.986	entropy	60	120
2200	0.986	0.986	log_loss	80	200
2500	0.984	0.984	entropy	80	800

Table 11: Results for the CNN classifier.

Period length (ms)	Accuracy	F1-score
raw	0.97	0.968
1000	0.968	0.962
1300	0.975	0.974
1500	0.974	0.972
1800	0.965	0.963
2000	0.**983**	**0.979**
2200	0.966	0.965
2500	0.966	0.966

the classification task.
The main results, displayed in table 3, show that boxing punches can be classified by machine learning models with an accuracy of up to 99.5%. This is over one percent higher than previous benchmarks on boxing punch

Table 12: Results for the RNN classifier.

Period length (ms)	Accuracy	F1-score
raw	0.876	0.869
1000	0.893	0.879
1300	0.803	0.785
1500	0.893	0.879
1800	0.855	0.835
2000	0.906	0.889
2200	0.858	0.848
2500	**0.911**	**0.891**

classifications ((Labintsev et al., 2021; Wagner et al., 2019; Worsey et al., 2020). Our best classifier overall is the random forest classifier, with 120 base estimators, entropy as the criterion, a max depth of 80 and trained on the raw data. Wagner et al. (2019), who's best performing classifier was also a random forest classifier with 120 base estimators (criterion and max depth specifications were not mentioned in their paper), trained their model on the same dataset, but found it performed best on data normalized to a period length of 2200 ms and a sampling rate of 200 Hz (opposed to our sampling rate of 100 Hz). This comparison leads to a few hypotheses. Firstly, it is possible that upsampling the data to a sampling rate of 200 Hz negatively influences the performance. By interpolating data points, we might introduce noise to the data, and too much noise in the data can negatively impact model performance. On the other hand, it can be that the hyperparameter tuning stage of Wagner et al. (2019) was not thorough enough, whereby they missed important parameters in the models (e.g. max depth in random forest), and as a result that caused the difference in performance.

That our second and third best classifiers, are the support vector machine and logistic regression, is in line with the reports of experiment one in Worsey et al. (2020). Our SVMs best performing kernel was a Gaussian radial basis function, which is the same kernel that was used by Worsey et al. (2020).

Comparing our CNN results to that of Labintsev et al. (2021), who experimented with both one-dimensional and two-dimensional convolutional neural networks, we score 0.02 higher in accuracy. Their one-dimensional CNN used three convolutional layers, but used each layer on a different axis of the data (x, y or z). Our one-dimensional CNN outperformed this model by 0.36 in accuracy. The best performing CNN of Labintsev et al. (2021) was a CNN with three two-dimensional convolutional layers, which yielded an accuracy score of 0.96. Notably, all CNNs in Labintsev et

al. (2021) have only a single fully connected layer after the convolutional layers. This could be a reason that their performance is lower. The other reason could very well be that their dataset was too small for their models to perform competitively, as their dataset has 5 classes compared to our 4 and consists of 1912 punches in total compared to our 7606 punch dataset (TheSmartPunchTeam, 2019). The disadvantage of deep learning models is that they often require a lot of data to learn accurate representations.
Our RNN can be lightly compared with that of Nunavath et al. (2021), which achieved 98.75% accuracy in classifying basic daily movements measured by wrist-worn, tri-axial accelerometers. Big differences between our model and theirs (apart from the differing classification task), is that theirs was trained for 2000 epochs and with different optimizers (stochastic gradient decent, AdaGrad, and Adam). Also, their RNN consisted of four standard recurrent layers of size 32, combined with a fully connected layer. This way, they were able to score 0.08 higher in accuracy than our RNN with a single LSTM layer. We experimented with multiple recurrent layers, but did not evaluate hidden states lower than 64. And as we observe in the confusion matrix for the RNN (figure 7), the model makes mistakes in all possible ways. This can be interpreted as a sign that the capacity of the RNN was not large enough, or that it lacked the additional layers that would have allowed it to learn to extract more abstract features. Yet again, the difference in dataset size (around 175,000 instances for their wrist-worn dataset) might also be a cause for the difference in performance.
The results of the period length normalization experiment produced different results for the classifiers. For one, the DL models seem to benefit from the normalization method, especially the RNN (+0.035 accuracy score and +0.022 F1 score with respect to the baseline raw period lengths). For the CNN, there is one percent increase in accuracy and F1 score. Of the ML classifiers, adaptive boosting was the only algorithm which profited from the normalized period length (+0.021 accuracy score and + 0.021 F1 score with respect to the baseline data). The optimal period lengths differed across classifiers. Regarding the overall influence that the suggested normalization method has on model performance, it can be stated that the effect is classifier specific. From what we observe, some classifiers experience small increases in performance, given that they already perform well on the raw data. What period length works best, is also unique for every classifier.
Contribution of the individual tsfresh features were investigated by calculating the Shapley Additive Explanations (SHAP) values (figure 9). We found that the mean of the z axis and the sum of value of the z axis were the most important features, averaged over all classifiers. Furthermore, we found that the top ranking features were similar between classifiers. It is

important to note that the feature correlation matrix in figure 8 displays a high correlation between features. As a result, the SHAP values are at risk of being misleading, because the SHAP method assumes feature independence. As such, it is hard to say with certainty what features are most important in the classifiers' predictions. Further error analysis of all the classifiers, showed that hooks and upper-cuts are commonly confused with one another (figures 6 and 7). A possible explanation for this, is that the arm of a boxer is similarly oriented during upper-cuts and hooks. For both punches, the arm is bent at the elbow, and moves forward in the direction of the front of the fist. The main difference between the punches, is caused by the position of the shoulder joint. For the upper-cut, the upper arm is oriented vertically, while for the hook, the upper arm is oriented horizontally. It is reasonable to hypothesize that this difference is not captured by the accelerometer, as it does not register which way the force of gravity is directed.

5.1 *Limitations and Future Work*

Results show that the RNN and - to a lesser extent - the CNN were outperformed by the machine learning models. We suggested this might be (partly) a consequence of a rather small dataset. Future work should work out whether CNNs and RNNs might actually be more successful in this classification task with tri-axial accelerometer data, by utilizing a substantially larger dataset.

We found no valid evidence that could determine which features contributed the most to the ML classifiers' predictions. The SHAP values that we calculated are at a high risk of being misleading, because SHAP assumes feature independence (Lundberg & Lee, 2017), and some of our features are highly correlated. Hence, to truly find out what features are the most important, future experiments could employ the adjusted method of SHAP by Aas, Jullum, and Løland (2020), which does not assume feature independence. Alternatively, the effect on model performance of dimensionality reduction techniques like PCA or feature clustering techniques could be studied. With regards to the deep learning models, their performance evaluation as of right now, could be enriched by utilizing the Deep SHAP method (Lundberg & Lee, 2017), which is a specilized SHAP method for deep learning models. Deep SHAP could give insights into what parts of the input sequences contribute most and least to the final prediction of the classifiers.

What this study lacked was a broader evaluation of what features can be used. As mentioned, the current features were chosen for their simplicity and interpretability. There may be features that are more discriminative

than the features in the current set. Experiments with clustering and feature permutations could allude whether more or less features can be used to acquire similar or even better performance scores for the classifiers.
The data normalization and feature extraction methods used in this study are not designed for real time classification tasks. If one were to develop a machine learning classification pipeline for applications where real time predictions are essential, an effort should be made to study faster/more efficient pre-processing techniques. Otherwise, one should develop the pipeline with one of the suggested neural networks, which require minimal pre-processing.

6 CONCLUSION

This thesis project aimed to discover to what extent boxing punches, measured by wrist-worn tri-axial accelerometers, can be classified. Inspired by all the previous successful models in the literature, a total of five machine learning models, and two deep learning models were evaluated on their performance of the classification task. We found that five of our models were able to deliver accuracy and F1 scores of above 95%. And our best and second best machine learning models, the random forest classifier and the rbf support vector machine, were able to beat all previous models in accuracy score (99% accuracy and F1 score).
In support of the research, two sub questions were formulated. To answer our first sub question: *"How does normalizing the period length of the time series data influence the performance of the machine learning and deep learning classifiers?"*, we found that normalizing the period length of sequential data can positively influence the model's performance, given that it already performed well on the raw data. Both the effect and the optimal period length are model specific. For some of our models, the period length normalization did not yield any positive effect. The second sub question is: *"What is the impact of the individual features on the performance of the machine learning classifiers?"* We were unable to acquire valid results that could indicate what the accurate contribution of each feature was. We have proposed alternative techniques, that could circumvent the multicollinearity in our extracted features.
To conclude, we have provided evidence that boxing punches (the frontal punch, the hook, and the upper-cut) measured by wrist-worn, tri-axial accelerometers, can be classified with an accuracy and F1 score of up to 99.5%. Next steps would be to investigate pre-processing methods that enhance the pipeline's efficiency, so that it is viable to use in real time applications. Another direction would be to develop another module which can automatically detect punches in a continuous accelerometer

signal. Combining such a module with the current pipeline could prove useful in automatic performance tracking or automatic scoring in martial art competitions. Lastly, as we mentioned in the introduction section, the pipeline holds potential in the automatic annotation of boxing punches in video recordings.

REFERENCES

Aas, K., Jullum, M., & Løland, A. (2020). *Explaining individual predictions when features are dependent: More accurate approximations to shapley values.* doi: 10.48550/arXiv.1903.10464

Bisong, E. (2019). Google colaboratory. In *Building machine learning and deep learning models on google cloud platform: A comprehensive guide for beginners* (pp. 59–64). Berkeley, CA: Apress. doi: 10.1007/978-1-4842-4470-8_7

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5-32. Retrieved from http://dx.doi.org/10.1023/A%3A1010933404324 doi: 10.1023/A:1010933404324

Christ, M., Braun, N., Neuffer, J., & Kempa-Liehr, A. (2018, 05). Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package). *Neurocomputing*, *307*. doi: 10.1016/j.neucom.2018.03.067

Cortes, C., & Vapnik, V. (1995, sep). Support-vector networks. *Mach. Learn.*, *20*(3), 273–297. Retrieved from https://doi.org/10.1023/A:1022627411411 doi: 10.1023/A:1022627411411

Ellis, K., Kerr, J., Godbole, S., Staudenmayer, J., & Lanckriet, G. (2015, 12). Hip and wrist accelerometer algorithms for free-living behavior classification. *Medicine and science in sports and exercise*, *48*. doi: 10.1249/MSS.0000000000000840

Hindle, B. R., Keogh, J. W., & Lorimer, A. V. (2021). Inertial-based human motion capture: A technical summary of current processing methodologies for spatiotemporal and kinematic measures. *Applied Bionics and Biomechanics*, *2021*, 1–14.

Hochreiter, S., & Schmidhuber, J. (1997, 12). Long short-term memory. *Neural computation*, *9*, 1735-1780. doi: 10.1162/neco.1997.9.8.1735

Kerr, J., Patterson, R., Ellis, K., Godbole, S., Johnson, E., Lanckriet, G., & Staudenmayer, J. (2015, 12). Objective assessment of physical activity. *Medicine Science in Sports Exercise*, *48*, 1. doi: 10.1249/MSS.0000000000000841

Labintsev, A., Khasanshin, I., Balashov, D., Bocharov, M., & Bublikov, K. (2021). Recognition punches in karate using acceleration sensors and convolution neural networks. *IEEE Access*, *9*, 138106-138119. doi: 10.1109/ACCESS.2021.3118038

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In I. Guyon et al. (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

Miele, V. J., & Bailes, J. E. (2007, February). Objectifying when to halt a boxing match: A video analysis of fatalities. *Neurosurgery 60(2)*, 307-316. doi: 10.1227/01.NEU.0000249247.48299.5B

Nunavath, V., Johansen, S., Johannessen, T. S., Jiao, L., Hansen, B. H., Berntsen, S., & Goodwin, M. (2021). Deep learning for classifying physical activities from accelerometer data. *Sensors*, *21*(16). Retrieved from https://www.mdpi.com/1424-8220/21/16/5564 doi: 10.3390/s21165564

Paszke, A., Gross, S., Massa, F., Lerer, J., A.and Bradbury, Chanan, G. & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, *32*, 8024-8035. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011, November). Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, *12*(null), 2825–2830.

Rosenberg, D., Godbole, S., Ellis, K., Di, C., Lacroix, A., Natarajan, L., & Kerr, J. (2016, October). Classifiers for accelerometer-measured behaviors in older women. *Medicine Science in Sports Exercise*, *49*, 1. doi: 10.1249/MSS.0000000000001121

Rozemberczki, B., Watson, L., Bayer, P., Yang, H.-T., Kiss, O., Nilsson, S., & Sarkar, R. (2022). *The shapley value in machine learning.*

TheSmartPunchTeam. (2019). *Boxpunch dataset.* Kaggle. Retrieved from https://www.kaggle.com/datasets/smartpunchteam/boxpunch-dataset

Wagner, T. (2019). *standardizer.py.* GitHub. Retrieved from https://github.com/smartpunch/timeseries_helpers/blob/master/standardizer.py

Wagner, T., Jäger, J., Wolff, V., & Fricke-Neuderth, K. (2019, October). A machine learning driven approach for multivariate timeseries classification of box punches using smartwatch accelerometer sensordata. In (p. 1-6). doi: 10.1109/ASYU48272.2019.8946422

Wang, H., & Wu, J. (2016). Classification of human posture and movement using accelerometer data. In *International conference on innovations in computing networking* (pp. 0975–0282).

Wes McKinney. (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt & Jarrod Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (p. 56 - 61). doi: 10.25080/Majora-92bf1922-00a

Worsey, M. T. O., Espinosa, H. G., Shepherd, J. B., & Thiel, D. V. (2020). An evaluation of wearable inertial sensor configuration and supervised machine learning models for automatic punch classification in boxing. *IoT*, *1*(2), 360–381. Retrieved from `https://www.mdpi.com/2624-831X/1/2/21` doi: 10.3390/iot1020021