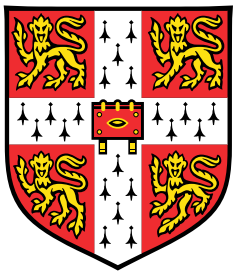
Co-activation detection in ten-finger typing on a virtual keyboard



Conor Foy

Supervisors: Prof. Per Ola Kristensson
Dr. John Dudley

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy in Machine Learning and Machine Intelligence

Hughes Hall August 2020

Declaration

I, Conor Foy of Hughes Hall, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

This report contains 14,961 words, excluding declarations, bibliography, photographs and diagrams, but including tables, footnotes, figure captions and appendices.

The software used for this report was writing by the author except in the following cases;

- The typing renders of the pre-recorder data outlined in section 4.1.2 was produced using software written by Dudley et al. [4]. Permission was gained from the author to use this software.
- The expected calibration error (ECE) values stated throughout chapter 5 were calculated using the software implementation written by M Cerliani [2] based on the work done by Guo et al. [9]. No licence accompanied the software or any other terms for its use.
- The decoder mentioned throughout the report was written by Dudley et al. [4]. The integration of the model developed in this project was also performed by Dudley et al.

Conor Foy
August 2020

Acknowledgements

First and foremost I would like to thank Prof. Per Ola Kristensson and Dr. John Dudley. They met with me every week and provided excellent support and guidance throughout the project.

I would also like to thank my parents and girlfriend Niamh for their help and support throughout the ups and downs over the year.

Lastly, I would like to thank the other members of the course and my friends at Hughes Hall, particularly Peyton and Andrew, for a very enjoyable and exciting year.

Abstract

Developing an effective text input method for virtual and augmented reality is an ongoing research area. Virtual keyboards are one potential solution that leverage users existing typing skill. However, they pose their own set of challenges that must be overcome. One of these challenges is the detection and removal of coactivation typing errors. This error is produced by the unintentional selection of a second character as a result of selecting the desired one. This thesis investigates the identifying characteristics of coactivations and uses them to train a model capable of removing coactivations from a text input stream.

Four identifying characteristics were found, these were: inter-key interval, key depth, key velocity, and inter-key correlation. Three different models were trained using features derived from these characteristics. These were a neural network, a support vector machine and a naive Bayes model and they achieved a mean accuracy and standard deviation across 4-fold cross validation of $98 \pm 1\%$, $97 \pm 1\%$ and $98 \pm 1\%$ respectively. The neural network was chosen as the best performing model as it achieved the highest average accuracy and F1 score. This model was successfully integrated into an existing statistical decoder and passed initial tests. Future work includes a full evaluation of the modified decoder. The neural network can successfully identify $97 \pm 1\%$ of coactivations while only incorrectly labelling $2 \pm 1\%$ of intended touch events as coactivations. As 4% of characters typed were coactivations, this equates to a $2 \pm 1\%$ reduction in errors.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

AR	Augmented Reality
AUC	Area Under the ROC Curve
CER	Character Error Rate
ECE	Expected Calibration Error
HMD	Head Mounted Display
KB	Keyboard
LI	Left Index Finger
LM	Left Middle Finger
LP	Left Pinky Finger
LR	Left Ring Finger
LT	Left Thumb
NLTK	Natural Language Toolkit
ONNX	Open Neural Network Exchange
ReLU	Rectified Linear Unit
RI	Right Index Finger
RM	Right Middle Finger
ROC	Receiver Operating Characteristic

RP Right Pinky Finger

RR Right Ring Finger

RT Right Thumb Finger

SVM Support Vector Machine

VR Virtual Reality

WPM Words Per Minute

Chapter 1

Introduction

Virtual and augmented reality (VR, AR) provide new ways of interacting with the digital world. They allow users to move around in and interact with virtual spaces. VR and AR have many potential applications, including virtual offices, training, and development. There have been considerable advancements in head mounted displays (HMDs), which allow unrestricted movement of the user and improve the immersive experience. However, a text input method for VR that is as effective as a traditional keyboard has yet to be developed.

One emerging input method is to use the well-established design of the traditional keyboard to create a virtual keyboard. However, the lack of a physical surface introduces a new set of errors. One of these errors has been termed a coactivation. This error is produced by the unintentional selection of a second character as a result of selecting the desired one. This error is caused by the lack of passive resistance that is normally produced by the spring in each keyboard key. Without this resistance is it easy to allow the other digits to move unintentionally and select undesired keys.

This project intended to answer two questions:

Research question 1 What are the important characteristics of coactivations?

Research question 2 How can coactivations be effectively detected?

The aim of the project was to develop a system that can be integrated into existing auto-correctors and capable of improving their performance, by effectively identifying coactivations. The following objectives were set out to answer these questions and achieve this aim:

Objective 1 Label and analyse an existing dataset for potential identifying characteristics.

Objective 2 Use these characteristics to motivate potential features.

Objective 3 Train and test these features with different models

Objective 4 Select the best performing features and the best model.

Objective 5 Integrate these into an existing statistical decoder.

Chapter 2

Background

Our hands are capable of extremely wide range of both powerful and light motion. To achieve this, they have a very complicated structure, consisting of many types of tissue, including tendons, ligaments, and muscles [18][11][13]. Due to the variety of connecting tissues between components within the hand, constraints are imposed between these components. For example, the middle and ring digits share ligaments and other connective tissues preventing one from moving through its full range of motion without movement of the other.

As well as being physically constrained, the movement of our muscles is also constrained by our neuromuscular system. These constrains are used to reduce the number of degrees of freedom that need to be independently controlled, making it easier to control a group of muscles at once [17]. This coupling makes it almost impossible to control one muscle without unintentionally moving other ones.

These constraints have the effect of coupling the motion of the digits. This is what contributes to the error known as coactivations while typing. A coactivation is the unwanted selection of an unwanted key by a secondary digit as a result of the motion of the primary digit to select the intended key. Coactivations are divided into two types characterised by the order of the intended and unwanted key selection. If the intended key is selected first, the unwanted key is a post-coactivation. If the unwanted key is selected first is it a pre-coactivation.

2.1 Virtual and Augmented Reality

The area of virtual and augmented reality (VR, AR) has seen major improvement over the last number of years with the release of the HTC Vive and Oculus Rift followed by the Value Index. These technologies are based on overlaying a computer-generated scene on top of our normal world, replacing all or part of what we would normally see. This is achieved using a head mounted display (HMD). The display is tracked so its position and orientation

in the real world are known. This allows the generated world to be aligned with the real one, so as a user moves their head around, the displayed scene moves accordingly. There are different head mounted displays for virtual and augmented reality. In virtual reality the user's view is completely blocked by screens placed in front of the face. These screens then project the generated scene towards the user's eyes. This is different to augmented reality where typically a piece of glass is placed in front of the user and the overlaid scene is reflected off this glass into the user's eyes. This allows the real scene to pass through the glass so both can be seen simultaneously.

This technology has many potential applications but has primarily been used for entertainment. One potential application is to use the technology to create virtual reality offices. Allowing more immersive video conferencing and adaptable office spaces. One factor that has contributed to the slow adaption and general use of VR and AR is the lack of a reliable, effective, and intuitive user interface. Feedback to the user via the display has improved considerably over the last number of years but input from the user to the system is further behind in development. Many studies have been done to evaluate different input techniques but no dominant method has emerged [1][5][7][8]. New technologies are only adapted if they provide some advantage over an existing one. VR has found application in entertainment because it provides a new experience to users. For the virtual reality to be more generally adopted users need to be at least as efficient as they would be without the system. Input via text entry is the dominant way users interact with electronic systems and so VR and AR text entry systems need to be further development if they are going to be adapted into mainstream use. This is not a trivial task as traditional methods such as the keyboard will not work without modification or redesign. The lack of a physical surface on which to type creates a number of problems and is the greatest challenge to the virtual keyboard. One of these problems is the lack of support for the wrists which can lead to arm strain. Another is the lack of resistance or touch feedback from the keys to indicated that a key has been pressed which causes coactivations. This is the issue that is addressed in this project.

2.2 Text Entry in Virtual Reality

Limited results were found in the literature regarding coactivations, though work has been done to investigate text entry in virtual and augmented reality, in general [4][1].

Dudley et al. [4] investigated the potential performance of users, covering four typing strategies using a virtual reality keyboard. This was done to motivate design implications and to further understand the limitations of current text entry in virtual reality. The strategies covered involved the alignment of the virtual keyboard with a physical surface and the

number of fingers used to type. The goal was to determine what effect typing in mid-air, with no wrist support or touch feedback to indicate when key was selected, verses typing with a surface aligned with the virtual keyboard, would have on the user's typing performance. The objective was to find the performance potential of users under the four conditions if limitations due to tracking and word correction were removed. This was done using a precision finger tracking system and a simulated statistical text decoder. This study showed that users achieved significantly higher word entry rates when the keyboard was aligned with a physical surface. Users also struggled to utilise all ten fingers, achieving entry rates similar to two finger typing. This shows that traditional designs fail to support the user in full ten finger typing and that these limitations need to be overcome for VR to compete with standard text entry methods.

Bowman et al. [1] evaluated four text input techniques for immersive virtual environments. The objective of this study was to measure the performance and usability of the following techniques, the Pinch Keyboard, a one hand chord keyboard, a soft keyboard using a pen and tablet, and speech. This was achieved by presenting a series of phrases to the participants, which they relayed back to the system using one of the four techniques. This work showed that there was no clear technique better suited for text entry. Each had its advantages and disadvantages. For example, speech had the highest input rate but was also found to be tedious by the users. The pen and tablet method had a high input rate but also caused arm strain and discomfort. This found no dominant solution and the best choice depended greatly on the specific user. Further research is needed in this area to address the disadvantages and to increase the usability of such techniques.

2.3 Coupled Digit Motion

Coupled motion of human digits is fundamental to the mechanisms that cause coactivations. Understanding these mechanisms may provide useful insights into how best to detect them. There have been several studies that aimed to investigate and quantify this motion, of which a few are outlined here [6] [17] [10].

Fish et al. [6] investigated the correlations between finger lengths and between finger orientations during typing tasks. This was done to quantify similar finger motion while typing. The positions of the metacarpal-phalangeal joint and the middle phalanx of each finger were tracked with reflective dots. This tracking was limited to a plane parallel to the keyboard, so the height of the fingers above the keyboard was not recorded. The correlation coefficient between the rate of change of each finger length and orientation were calculated over a range of typing tasks. Finger length and orientation were equated to the length and orientation of

the line connecting the two reflective dots on each finger. They found in most cases that there was significant correlation between digits and that the highest was between neighbouring digits. They also found that while digits move in coordinated motions, once one was needed to complete a task, it was decoupled from the group and the correlation dropped substantially. This work was however performed on a physical keyboard with passive resistance by the keys. This differs from the setup used in this project, but it is expected that these observations will be relevant because up to the moment a digit contacts a key its motion is unrestricted, which is true for both setups. This is an important insight into the coordination of the digits while typing and may prove useful in understanding coactivations.

Schieber et al. [17] completed a review of the physical and neuromuscular constrains on the motion of the hand. The purpose of this review was to investigate what constraints are imposed on hand function and control, give an overview of what is known about each constraint, and to motivate further study in this area through examples of applications that would benefit from it. Their review outlined kinematic constraints, which are physical restrictions in motion due to mechanical coupling of the soft tissue and tendons of different fingers. Central control constraints were also explored. These are not physical constrains and are imposed by the control system. These arise due to synchronisation of close proximity motor units that control different muscles and due to muscles being influenced by a range of motor neurons with no one to one mapping. They also found that during tasks involving the motion of multiple digits, like typing, when any one digit is being used for a specific task, all the other digits are also in motion. This involuntary motion is caused by the reduction in the number of degrees of freedom needed to simplify the complex control task of moving the hand. These constraints and limitations give rise to coactivations and so understanding them can lead to useful insights into how they might be identified.

H*ä*ger-Ross et al. [10] investigated the individuation and stationarity of human fingers. The objective of this study was to determine whether humans move other digits when attempting the move just one and if so to quantify this movement. This was done by moving one digit back and forth while trying to hold the others stationary. To simplify and standardise participant's motion a digit guide was created that allowed the digits to only move in one dimension. While the wrist was fixed, each finger in turn was extended and flexed repeatedly and the motion of the other digits was recorded. This work showed that there was significant motion of the digits that were intended to be held stationary. These results agree with the findings of Fish et al. [6]. To quantify this motion, two metrics, individuation and stationarity, were calculated for each digit. Individuation measures how much other digits move when the digit of focus moves. Stationarity measures how little the digit of focus moves when each of the other digits move. They showed that the thumb and index had the greatest

individuation and stationarity from the other digits, followed by the pinky, with the middle and ring having the least. This suggest that the middle, ring and pinky digits are more likely to cause coactivations.

2.4 Feature Engineering

Feature engineering is an important step in the process of developing a machine learning model. It is the process of choosing information and the form of this information, that will be feed into the model [19]. This is a critical step as the model's task is to learn the underlying function that transforms the input into the output. If the relevant information is spread too sparsely in the input feature vector the model may not find the correct function. If the complexity of the underlying function is too great it may not find the function. To increase the likelihood of our model finding the correct function two steps can be performed. The first is to narrow the input feature vector to only the most relevant inputs. This reduces the size of the model's initial layers needed to process the inputs. This frees up more resources to be used for feature extraction. The second is to perform any necessary complex computation before passing the inputs to the model, such as differentiation or integration. Passing the velocity of an object, instead of its position and time may achieve higher performance. This can simplify the function the model is trying to learn as it no longer needs to do the feature extraction itself. Effective feature selection simplifies the model's task allowing smaller models to be used, which can be trained with smaller datasets. This almost always improves the performance of a model. Simplifying a task and reducing the size of a model is crucial if training data or computing resources are limited. This is the case in this project, as only a small dataset is available.

2.5 K-Fold Cross Validation

Another important step in developing a model is estimating its out-of-sample performance. Out-of-sample refers to the testing of samples that were not used in the training set. This indicates how the model will perform in a real word situation. This must be accurate as it is easy to over estimate performance. This is typically done using a train-test split, where the model is trained on most of the dataset and tested on a small subset not shown to the model during training.

A more accurate estimator is K-fold cross validation. This technique goes one step further by applying a train-test split multiple times, training and testing the model multiple times and averaging the results [15]. The way the splits are created is important. First, the data is split

evenly into K groups or folds. One group is kept hidden as the test set while the other $K-1$ groups are combined into the training set. A newly initialised model is trained and tested on these sets and the performance is recorded. This is repeated K times with each group taking a turn as the test set. Averaging the results gives a more accurate estimation of the real-world performance. Using K-fold cross validation also allows data to be used more efficiently as each fold is used in both training and testing. This helps reduce biases that may arise due to the choice of where to split the data.

Certain criteria can also be applied when creating the K folds. This is known as stratified K-fold cross validation. This is where the elements of each group are not randomly assigned but assigned in such a way to meet or minimise desired criteria. For example, the groups could be stratified so that an equal number of each class appears in each group. Another could be to minimise the difference of an averaged continuous variable across each group. This ensures each group is a good representative of the sample dataset and does not provided a biased sample to the model which would affect the estimation of the real-world performance.

2.6 Machine Learning Algorithms

There are a wide range of machine learning algorithms each with their own strengths and weaknesses. Each is best suited for a particular task, like clustering or classification. These algorithms are trained on sample data to produce a desired output, by adjusting their internal parameters. This allows them to learn many complex relationships between the input and output data. Once these relationships are learned they can be used to make predictions on new inputs. Three types of these algorithms were tested in this project and are outlined below.

2.6.1 Artificial Neural Network

Neural networks have seen a major increase in popularity as computing abilities have grown. They are a very adaptable and as a result have been used in a wide range of areas. Their adaptability comes from the way their architecture can be modified to suit a specific task. Neural Networks are comprised of any number of layers, chosen based on the nature of the task. There are several different types of layers that can be used, each with their own strengths and weaknesses. The simplest is the fully connected layer, but there are other more complex layers like convolutional layers or long short-term memory layers.

A neural network is a mapping process that applies a function to an input vector, to produce an output vector. This function is comprised of subfunctions, one for each layer in

the network. The input is transformed by the first layer's subfunction and its output becomes the input to the second layer's subfunction. This is repeated until the last layer where its output is the final output of the overall function. This allows different types of layers to be used in any order. The simplest layer, the fully connected layer consists of several nodes. Each node takes each input element and multiplies it by a weighting factor. It then sums these products and adds a bias. Finally, a non-linear function is applied, and the result is the output of this node. Examples of non-linear functions include sigmoids and rectified linear units. The output of a node is given by the following equation, where y is the output of the node, $\mathbf{x}$ is the input vector, $\mathbf{W}$ is the weight vector, b is the bias term, and ϕ is the non-linear function.

$$y = \phi(\mathbf{W}\mathbf{x} + b) \tag{2.1}$$

This node's output along with the outputs of the other nodes form the output of the layer. Each node has different weighting factors and bias, which produces different outputs.

The training of a neural network involves tuning these weights and biases. This is done by applying the function to an input vector and calculating the difference between the actual output and the desired output. The function is completely differentiable so the effect each weight and bias has on the output is known. The weights and biases are then adjusted slightly so the actual output is slightly closer to the desired output. This is repeated multiple times with several different sample inputs until the weights converge. The network is now trained and can be tested on a new set of sample inputs to evaluate how well the predicted outputs match the desired ones.

2.6.2 Support Vector Machines

Support vector machines (SVMs) are another popular algorithm primarily used for classification. Many systems ranging from spam filters to cancer detectors rely on categorising inputs. SMVs are based on the concept of creating a partition between classes. For binary classification there is only one partition, where everything that falls on one side is labelled class 0 and everything that falls on the other side is labelled class 1.

SVMs try to find the best surface that separates data points. This is done by plotting the input vector of several sample data points on the same graph. Each element of the input is plotted in a separate dimension and extends to any number of dimensions. In the simple two-dimensional case, the objective of the SVM is to fit a curve that divides all the points labelled 0 from all the points labelled 1. The simplest curve is a straight line and the simplest set of points is where a line can completely separate the data. In this case there are in fact multiple lines to choose from, so the one chosen is the one that is furthest away from any

one point. This distance is called the margin and maximising it is the goal of an SVM. For example, in the case of just two points labelled 0 and 1, the dividing line would pass through the midpoint of the line connecting the two points and be perpendicular to it. The points closest to the dividing line are called the support vectors as they define where the dividing line is placed. The rest of the points have no bearing on its position. Once the dividing line or surface is found, new predictions are made by plotting the input and classed according to which side of the surface the input falls.

There are also a number of techniques that allow SVMs to generalise to more complex datasets. Soft margins allow some training points to fall on the wrong side of the dividing surface if doing so achieves a significantly wider margin. Another technique is the use of a kernel. This can transform data that cannot be linearly separated into a form that can, allowing SVMs to achieve non-linear separating curves.

2.6.3 Naive Bayes

The last algorithm used in this project was naive Bayes. This is the only true probabilistic classifier of the three, as the others only output approximate probabilities. The probability of input belonging to a class can be far more useful than a binary decision. This allows the confidence of the model to be determined so more informed actions can be taken.

The naive Bayes algorithm calculates the probability of the input belonging to a class given the values of each input element. This is done by combining the prior probability density functions of each input with the probability of each class using Bayes theorem. The name naive comes from the assumption that the probability density functions are all conditionally independent. The prior probability of each class is calculated from its empirical frequency in the sample training set. The prior density function for a class is found by first filtering out all the samples of that class from the training set. A relative frequency histogram is then plotted from the first element from each sample. Finally, the density function is found by fitting a curve to this histogram. This is repeated for each of the input elements for each class. Each distribution returns the probability that the input element will equal some value given that it belongs to a certain class. Bayes theorem allows this to be rearranged to get the probability that an input element belongs to a certain class given its value. This is also true for multiple inputs which is described by the following equation, where C_i is class i, x_j is input element j with value equal to x_j.

$$
\begin{aligned}
P(C_1|x_1,x_2,\ldots) &= \frac{P(x_1,x_2,\ldots|C_1)P(C_1)}{P(x_1,x_2,\ldots)} \\
&= \frac{P(x_1|C_1)P(x_2|C_1)\ldots P(C_1)}{P(x_1|C_1)P(x_2|C_1)\ldots P(C_1)+P(x_1|C_2)P(x_2|C_2)\ldots P(C_2)+\ldots}
\end{aligned}
\tag{2.2}
$$

This gives the desired prediction, the probability of the input belonging to a class given the values of each input element.

Chapter 3

Approach

The general approach to this problem would be to train a machine learning model to take information from the user as they type and use this to predict the likelihood that a character selection is a coactivation. There are many models that can be used and many potential sources of information to feed into the model. These will both need to be investigated to discover the most suitable.

A typical approach would be to use a fully connected neural network and simply include all the available information from the typing task in the network's feature vector. This vector could include inputs like the position and velocity of the digits, the correlation between each pair of digits or whether a touch event occurred. In the available dataset, the positions of the digits, hands and head are recorded at a rate of 90 Hz or once every 11 ms. Even over a short period this would require passing a lot of information to the network.

To illustrate this, Dhakal et al. [3] found the average interval between keypresses was 239 ms and that slow typists had an interval of over 480 ms. Taking the data from 40 samples before a touch event, an interval of approximately 440 ms, should therefore contain, in almost all cases, all the information back to and including the last touch event. This data would be fed into a neural network and it would predict the likelihood the current keypress is a coactivation.

In this example, the feature vector might contain the following data from each sample; if an activation occurred at that sample, the height of the digits above the keyboard, the velocity of the digits, and the correlation between pairs of digits separated by hand. In this case, each sample would contribute 61 elements to the input vector and so the full feature vector would contain 2,440 inputs and predict 1 output.

A small network with a 2,440 input dimension, 4 fully connected layers with 32, 32, 32 and 1 nodes respectively, would have approximately 80,000 parameters. One rule of thumb

is that at least 10 data points are needed per model parameter to reliably train it without overfitting. Therefore approximately 1 million data points would be needed.

There were 119,890 key presses recorded throughout the dataset of which 4,627 of these were coactivations. Using an equal number of coactivations and intended touch events would result in 9,252 data points available for training and testing. It took Dudley et al. [4] approximately one hour to record the mid-air ten finger typing data for each of the 24 participants. This is the equivalent of three eight hour days. At this rate it would take approximately one year to gather enough data to train the model described above (not including the time needed to label the data).

This small dataset restricts the size of the network that could be used, the complexity of the task that could be learned, and ultimately the approaches that could be taken. To account for these, some computation could be performed using classical methods and then passed to the network. This would condense the information into fewer inputs that would be better at classifying coactivations. Determining the inputs or features to pass to the network is a difficult task and is called feature engineering. For the network to achieve good performance in classifying the keypresses, effective features need to be found.

As a result of the initial analysis of the problem statement and the restrictions on feasible approaches, the following approach was taken. The main steps of this approach are also illustrated in Figure 3.1. The project was split into two parts. The first involved learning about coactivations by studying them during the labelling process and the analysis of their characteristics. The second part used the knowledge gained in the first part and applied it to training a model to detect coactivations.

Firstly, the dataset was inspected in order to identify possible characteristics of coactivations. This also identified the underlying variables that were to be included in the feature vector. While inspecting the dataset, each touch event was also labelled as intended or coactivated to allow for training of the model. Once possible characteristics and their underlying variables were identified, they were analysed to determine if they were in fact identifying characteristics of coactivations. This was done on a sample of the dataset and only identified characteristics of coactivations, not how effective they were as inputs to the models. This identified which characteristics would be used as features in the feature vector and concluded the first part of the project.

The second part began with further testing of the selected features. A neural network was selected for these tests. This involved retraining the same model solely on each feature. From this, the effectiveness of each was evaluated and any underperforming ones were removed. Other suitable models including the neural network were then selected and trained on the best performing characteristic variables. The best model was then selected and some final

tests were performed. Finally, the selected model was incorporated into an existing statistical decoder.

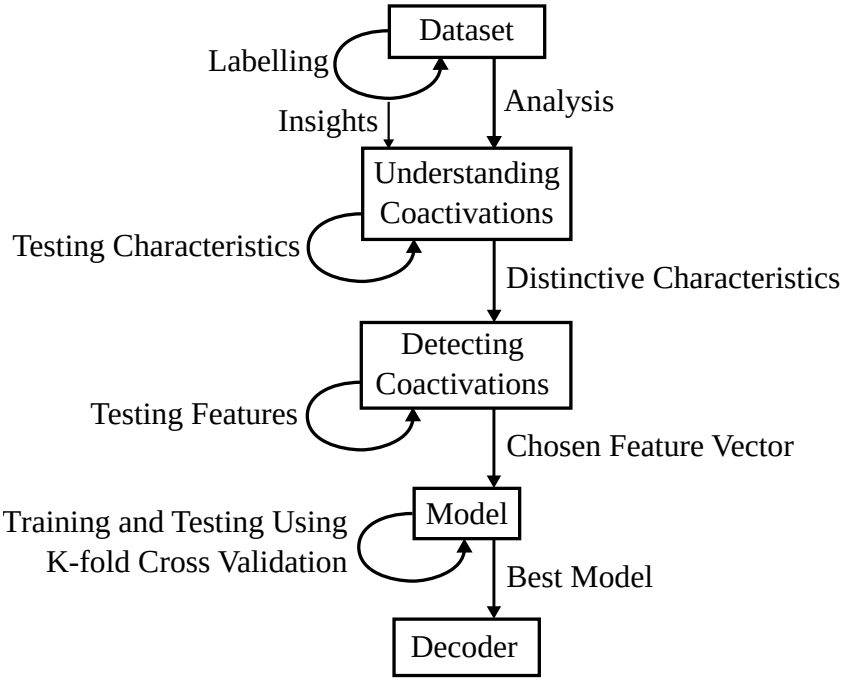


Fig. 3.1 *Flow diagram of the approach used to complete the project*

Chapter 4

Understanding Coactivations

4.1 Coactivation Dataset

4.1.1 Existing Dataset

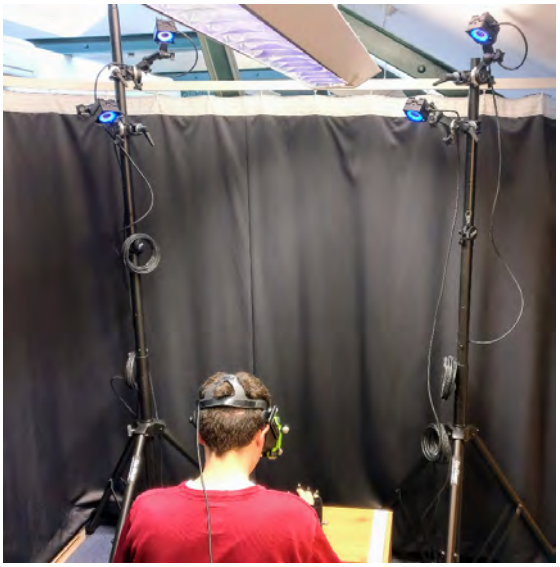
This work builds on the work of Dudley et al. [4] and uses the dataset they collected. This dataset comprises of tracking data for 24 participants. In the work of Dudley et al. [4], each participant typed various sentences while their digit tips, hands and head were tracked as shown in 4.1a. The tracking was performed by a series of OptiTrack infrared cameras [14], that provided sub-millimetre tracking at a 90 Hz sample rate shown in 4.1b. To track the digits, reflective markers were attached to the tips of each participants' digits. The body of the hand was tracked by a rigid set of reflective markers on the back of a glove the participant wore. The head was tracked by attaching reflective markers to the head mounted display the participant wore.

Each participant was presented with a virtual office shown in 4.2a and a virtual keyboard shown in 4.2b on which to type.

Each participant was presented with 170 sentences in total and asked to retype them. The participants had an initial ten sentences to become familiar with the setup and then performed four sets of forty sentences, with breaks in between to make sure they were not getting fatigued.

This typing was performed on a virtual keyboard where a letter was typed when a digit passed through a detection plane at the *x*-*y* position of a key on the keyboard. As there is no physical key to press this act of typing a letter is referred to as a touch event. A touch event is then further classified as a coactivation or an intended touch event.

(a) *Arrangement of the tracking dots used to track the participant's digits, hands and head.*



(b) *Positioning of the OptiTrack infrared cameras.*

Fig. 4.1 *Tracking setup.*

(a) *Main office environment.*

(b) *Keyboard layout and interface.*

Fig. 4.2 *Virtual office presented to participants along with virtual keyboard which they used to type sentences. Purple spheres—participants tracked digit tips. Purple cuboids—participants tracked main hand body (wrist to knuckles).*

From this tracking data and the text entered during each task, Dudley et al. [4] labelled each marker connected to each of the participants' digits and calculated the typing rate and error rate of the participant for each task.

4.1.2 Addition of Coactivation Labels to the Dataset

For the purpose if this project this dataset was extended by adding a one-digit indicator to each character typed by the participants. This was done by plotting each frame of the original 3D tracking data overlaid on the virtual keyboard shown in Figure 4.2b. The frames of each task were then combined into a video render of the participant's typing. These videos were inspected and any coactivations that occurred were noted. The touch events were labelled;

0 The character was an intended touch event.

1 The character was a coactivation linked to a preceding character,

2 The character was a coactivation linked to a succeeding character

3 The character was a coactivation cause by some other means.

The dataset contained 115,263 intended touch events, 4,205 post-coactivations, 392 pre-coactivations and 30 other coactivations. A total of 119,890 touch events recorded over the entire dataset. This shows that coactivations account for 3.9% of touch events.

4.1.3 Data Filtering

After the coactivation data was added, the data was filtered to remove any incomplete participant tasks. First each task was checked to ensure the number of characters recorded matched the number of touch events recorded. If these were not equal the task was removed. There were few errors of this kind.

Each task in the dataset was also expected to contain exactly 10 labelled digits corresponding to the 10 reflective markers that were picked up by the camera system. This was not always the case, as it was possible that the cameras could mistakenly register the shine of a reflective surface as a marker, though participants were asked to remove anything that may cause this kind of interference [4]. Dudley et al. [4] performed a post labelling of the markers, where the position and order of the markers relative to the body of the hand were used to identify which marker corresponded to which digit. Possible issues that could arise were, if the tracking of a marker was lost, for example due to obstruction, then the labelling algorithm may fail to label one or more markers resulting in unlabelled markers. When the

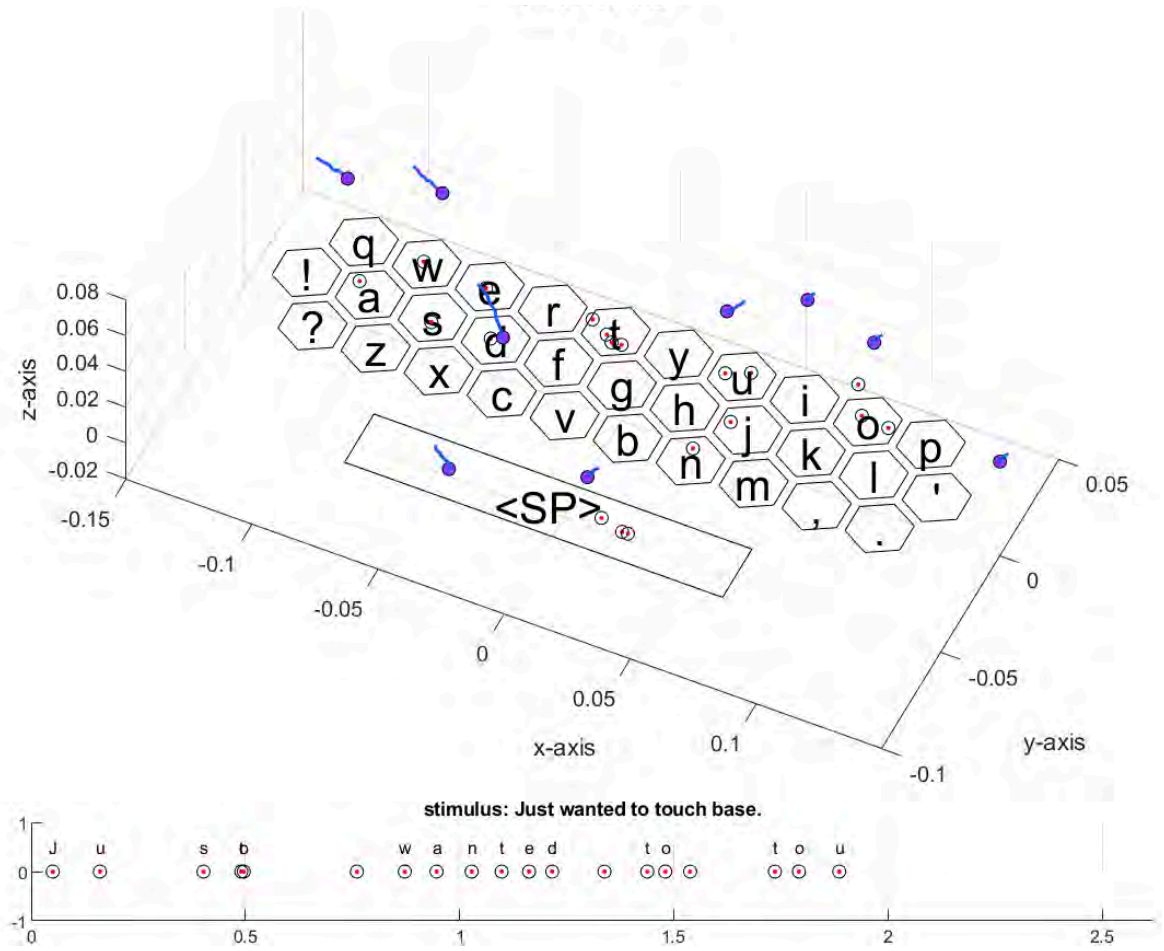


Fig. 4.3 *Frame taken from the typing renders used to identify and label coactivations. The figure shows the digit tips current and past positions, indicating the digits velocity and trajectory. The figure shows the positions of detected touch events on the keyboard, a timeline of these events, and the letter that was typed.*

lost marker reappeared, it could be labelled as a new instance of the same digit resulting in duplicates of the digit.

A significant portion of time was dedicated to investigating, analysing, and fixing the different cases of errors. Tasks were classed according to the following four parameters where any combination of these four could be present; label duplicates present, all ten unique finger labels present, the number of markers present and the number of labelled markers present. It was found that 44% of the tasks had some form of error. The dominant two cases both had more than ten markers present and all ten unique finger labels present. In the first case only ten of the markers were labelled, the extra markers were all unlabelled and in the second case the extra markers were either labelled or unlabelled. These cases accounted for

44% and 41% of the total errors, respectively. The next most dominant case only accounted for 4% of the total errors. It was decided that the all remaining cases, accounting for 15% of the errors, would require too much time to fix and so were removed. As the dominant two cases contained all 10 unique finger labels these were also the two cases that could most likely be fixed.

These cases were fixed by joining the unlabelled and duplicate markers. This was done by first joining any two markers with the same digit ID or any unlabelled marker to a labelled one, that met the following criteria. A join is accepted if the first entry of one marker is sufficiently close, both spatially and temporally, to the last entry of another marker. The thresholds chosen were a 5 cm spatial difference and a 500 ms temporal difference. These are slightly generous and can be reduced if required. This will re-join any markers where tracking was lost for less than 500 ms.

Once the joining process was complete each task was further filtered for missing data. If any of the 10-digit markers contained less than 90% of the median number of entries, it was deleted. This insured that the tasks remaining only contained minor tracking losses. If a task passed the second filter, any unlabelled markers were also removed. These unlabelled markers were usually only tracked for a very short time and were possibly due to reflections. This completed the filtering process.

The resulting dataset contained 22,059 intended touch events, 3,301 post intended touch event coactivation, 89 pre intended touch event coactivations and 12 coactivations cause by any other means.

4.2 Motivation for Potential Features

The dataset was inspected for common trends and features of coactivations. These included, the depth and velocity of the coactivations, as well as the interval and correlation of velocity between touch events.

4.2.1 Digit Depth

The depth of the touch event was noted as a possible feature as many coactivations were seen to only just enter the detection plane. In many cases the coactivating digit trailed behind the intended digit, reaching a shallower depth before both digits were retracted. From this it seemed likely that the depth of the touch event would be of benefit in distinguishing coactivations from intended touch events.

4.2.2 Digit Velocity

The velocity of the touch event was also noted as a possible feature. It takes effort and force to physically press a key with a digit. The natural intuition when pressing a key is to apply force in the movement. Without the usual resistance of a physical key this forceful motion is unopposed and the digit is accelerated downward, resulting in high velocity motion as it enters the detection plane. Coactivations are due of the lack of a natural resistance (physical keys to be pressed) to the downward motion of the digits. As this motion is not intended the movement carries little force and so there is little acceleration and therefore low velocity in the coactivating digits motion. This indicates that the velocity at the touch event would also be beneficial at distinguishing coactivations.

4.2.3 Inter-Key Interval

Another possible feature was the interval between touch events. Touch events often involve motion from the wrist, but wrist motion moves all the digits of the hand, not just the digit intended to be moved. By this intended wrist motion to move the intended digit to select the intended key, the wrist motion can also move another digit to select an unintended key. As both selections were caused by the same motion of the wrist, it is very likely that these events will have a high temporal proximity.

People can only process information at a finite speed and so there are limitations to the speed and accuracy of our motion. This results in a maximum rate at which we can accurately type. Dhakal et al. [3] found the average fast typist had an interval between touch events of around 120 ms and a lower bound of around 60 ms. This indicates that touch events that occur in an interval less than 120 ms are likely to be unintended and touch events with an interval less than 60 ms are almost certainly unintended. This suggests that the interval between touch events is likely to be a useful indicator when distinguishing coactivations.

4.2.4 Averaged Digit Velocity Correlation

The brain does not control each muscle in our bodies independently [17]. Instead muscles are controlled in groups making it almost impossible to control one muscle without unintentionally moving other ones. The digits also share ligaments and other connective tissues which physically limit their independent movement [13].

This feature can be motivated by the individual controllability of digits [10], which shows significant coupling motion between a digit intended to be moved and the others, intended to be held stationary.

This coupled motion can be seen as similar movement between digits. Fish et. al. [6] showed that while typing there is a reasonable degree of correlated digit movement when the digits were not engaged in a keypress task. Once a digit was selected for a keypress task its correlation to the other digits decreased substantially. It seems reasonable to assume that this decoupling of the task focused digit from the others may not always occur. This would lead to coupled digit motion possibly causing a coactivation. This indicates that measuring the correlation of digit motion would allow the detection of these cases and their resulting coactivations.

4.2.5 Finger and Letter Coactivation Distributions

Coactivations are caused by the coupled motion of a pair of digits. Understanding the distribution of the coactivations among these pairs would allow those more prone to coactivations to be identified. This would also allow the prior likelihood of a pair causing a coactivation to be determined.

Sorting the coactivations into pairs would split the data into 40 groups, 20 ordered pairs for each hand. As the dataset is quite limited this may result in too few counts per pair to be statistically significant.

Instead sorting the data solely by the digit that caused the coactivation would allow the prior likelihood of each digit to cause a coactivation to be found. This would reduce the number of groups the data would be split into from 40 down to 10. This could also prove useful and would be more likely to produce statistically significant distributions.

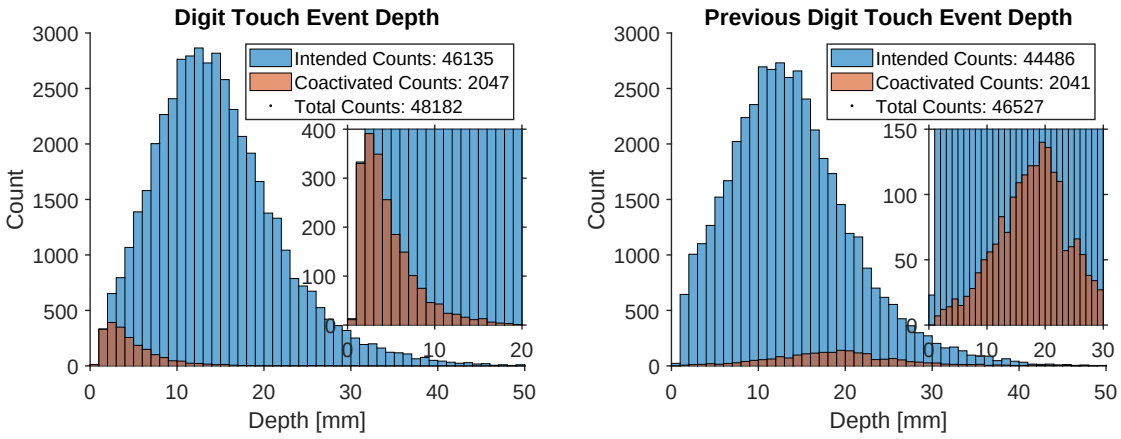
Another prior that could prove useful is the distribution of coactivations over the keyboard keys. Knowing this would identify which letters are more likely to be coactivated. The letter before the coactivated letter may also be of use.

4.3 Feature Evaluation

Each of the proposed features were analysed by calculating the feature for each task and for half the participants. The set was then divided into intended touch events and coactivations and plotted using a histogram. This was done to show how well the feature differentiated between the two distributions.

4.3.1 Digit Depth

The touch event depth was measured using the Z component of the digit position in the keyboard frame. The depth of the touch event was taken as the deepest point of a continuous



(a) *Current digit depth.* (b) *Previous digit depth.*

Fig. 4.4 *Depth feature results for the combined dataset of participant 0 to participant 11. Showing how depth of the current and previous key can be use to differentiate between intended touch events and coactivations.*

downward movement from the detection plane, in other words, the first local minimum after a touch event. The Z axis was orientated so that a more negative Z value equated to a deeper touch event. The markers were attached to the participant's fingernails, which introduced an offset between the marker and bottom of the finger. To account for this offset the keyboard was rendered at $Z = 0$ mm in the keyboard frame while the detection plane was positioned at approximately $Z = 10$ mm and the depth was measured from this plane. The resulting distributions of intended and coactivated touch event depths are shown in Figure 4.4a. This figure shows the depth of each touch event and whether that touch event was intended or coactivated.

This is a promising result with the means of the two distributions occurring at different depths. The mean intended touch event depth and standard deviation was 15 ± 7 mm ($Z = -5 \pm 7$ mm) whereas the average coactivation depth was much shallower, 5 ± 3 mm ($Z = 5 \pm 3$ mm). This shows that this feature will be beneficial when classifying coactivations.

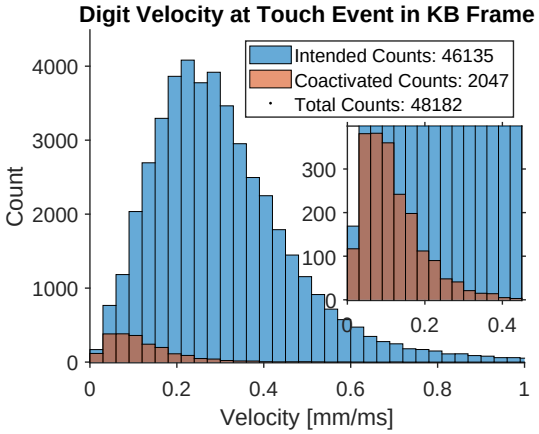
Due to the coupled motion of the digits it is possible that deeper intended touch events would be more likely to pull a neighbouring finger down to the detecting plane producing a coactivation on the next touch event. To illustrate this the depth data was also plotted to show the depth of the previous touch event while being classed according to the current touch event. This is shown in Figure 4.4b. This shows only a slight difference between the distributions but as it only relies on data from the previous touch event, it is a unique feature.

4.3.2 Digit Velocity

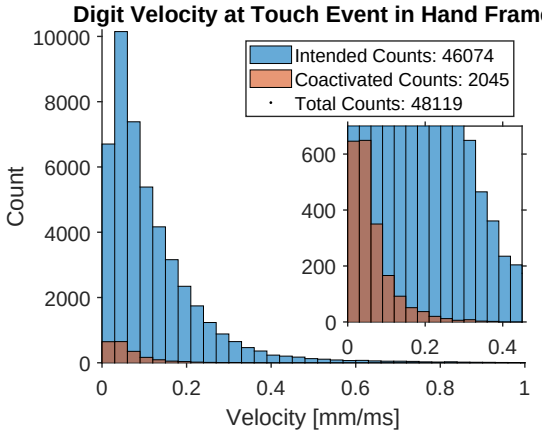
The touch event velocity was chosen to be the average velocity of the activating digit from 3 timesteps before the touch event to 3 timesteps after the touch event.

This range was chosen using the following logic. Dhakal et al. [3] found the average interval between keypresses was 239 ms. In this time four movements need to take place. The activating digit has to move from its rest height to the detection plane, from the detection plane to its max depth, from there back to the detection plane, and then back to the rest height. If each of these actions take approximately the same amount of time, then the full downwards motion will occur from about 60 ms before a keypress to about 60 ms after it. Taking the centre half of this motion to remove the portions where the digit was accelerating and decelerating, resulted in a 60 ms window over which to averaged. With each time step approximately equal to 11 ms, this equated to averaging the velocity over three time steps before the touch event to three time steps after the touch event.

As before, this feature was calculated for the first 12 participants and plotting as histograms shown in Figures 4.5a and 4.5b. This feature was evaluated both in the keyboard and the hand frame to determine if one separated the intended and coactivated distributions more than the other.



(a) Results performed in the keyboard frame.



(b) Results performed in the hand frame.

Fig. 4.5 *Velocity feature results for the combined dataset of participant 0 to participant 11. The histograms show how the velocity feature can differentiate between intended touch events and coactivations.*

This feature shows promising results. The keyboard frame shows more potential and shows that the wrist movement is a key factor in distinguishing coactivations according to digit velocity. In the keyboard frame there is a clear separation between the peaks of the distributions but in the hand frame the peaks lie almost on top of each other. This is reflected

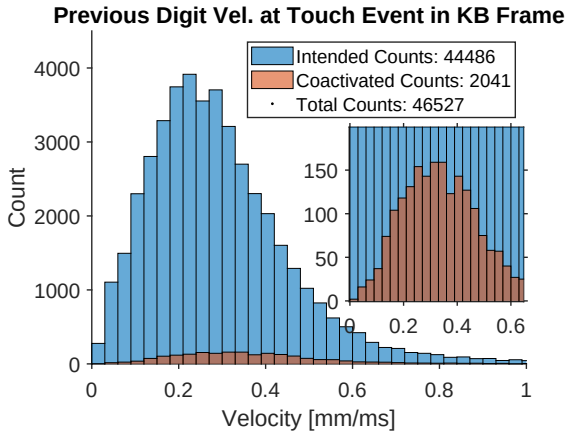


Fig. 4.6 *Velocity feature results for the combined dataset of participant 0 to participant 11. The histograms show how the velocity of the previous key can differentiate between intended touch events and coactivations.*

by the means of the distributions. In the keyboards frame the average intended and coactivated touch event velocities are 0.31 ± 0.17 mm/ms and 0.12 ± 0.08 mm/ms respectively, while in the hand frame they are 0.12 ± 0.12 mm/ms and 0.06 ± 0.06 mm/ms. Velocity of the digit in the keyboard frame may be of use in the feature vector but in the hand frame it is unlikely.

Due to the depth of the previous touch event showing some promise, the velocity of the previous touch event was also investigated. The results are shown in Figure 4.6. This shows almost no difference between the distributions and is unlikely to be very useful in the final model.

4.3.3 Inter-key Interval

The calculation of the inter-key interval feature differed slightly between an intended-intended touch event pair and an intended-coactivation pair. For the pair of intended events, the interval was between two consecutive intended touch events. For the coactivations, is was between a coactivation and its associated intended touch event, but these did not necessarily need to be consecutive. This was due to cases where one intended touch event caused two or more coactivations. In these cases, the interval was taken from the intended touch event to the second coactivation. The resulting distributions are shown in Figure 4.7

This is the most promising result with only a small overlap between the two distributions. This feature does not however indicate which of the two touch events, the interval is calculated between, is the intended touch event and which is the coactivation. This feature will have to be combined with the others in order to distinguish the coactivation.

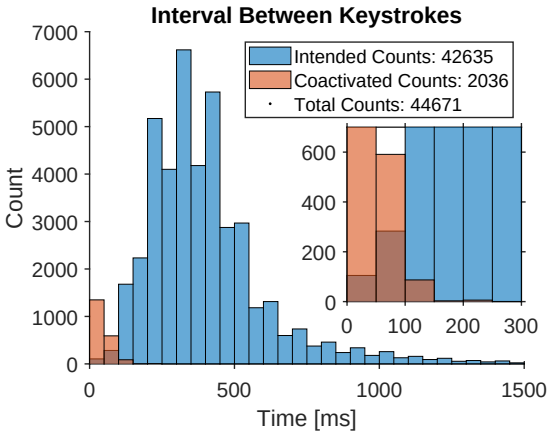


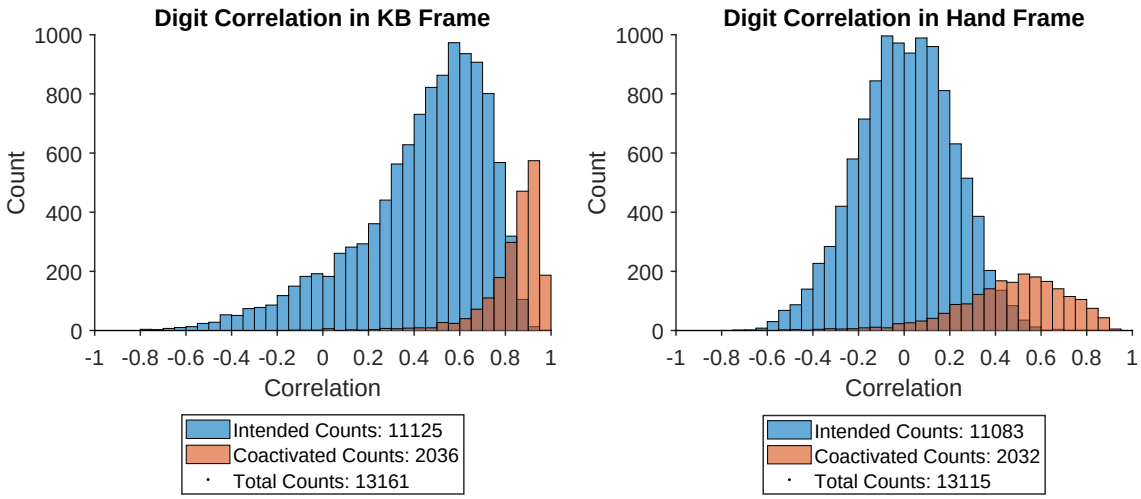
Fig. 4.7 *Interval feature results for the combined dataset of participant 0 to participant 11. The histograms show how the interval feature differentiates between intended touch events and coactivations.*

4.3.4 Averaged Digit Velocity Correlation

The calculation of the average correlation feature differed between intended-intended touch event pairs and intended-coactivation pairs in the same way as for the inter-key interval feature. Intended-intended touch event pair correlation was calculated between consecutive intended touch events and intended-coactivation pairs, between a coactivation and its associated intended touch event, respectively.

The correlation of two digits was calculated as the averaged dot product of the velocity vectors of the digits, normalised by the magnitude of the velocity vectors. The average was taken from 50 ms before the first touch event to 50 ms after the second touch event. The times were chosen using a similar logic to the velocity feature. The motion to activate a key begins approximately 60 ms before the key is activated and ends approximately 60 ms after it. The first and last 10 ms of the motion was removed to reduce the part of the motion where there is little to no movement. Using a window that extends from before the first touch event to after the second, ensures the motion of the two touch events was captured.

The correlation was calculated using the velocity of each digit, which was calculated from consecutive time and position samples. Due to tracking losses, the digits may be missing some of these samples. To ensure the velocities of the digits are aligned temporally and calculated over the same interval, if an entry at a certain timestamp was missing from one digit, the entry for the other digit was skipped until two entries with the same timestamp were found. The correlation was then calculated using the new temporally aligned entries.



(a) Results performed in the keyboard frame.

(b) Results performed in the hand frame.

Fig. 4.8 *Averaged correlation feature results for the combined data set of participant 0 to participant 11. The histograms show how the averaged correlation feature differentiates between touch events and coactivations.*

The equation used to calculate the correlation, the normalised dot product of velocity vectors, is given below, where $\mathbf{X}_{T_i}$ is the three dimensional position of digit i at time T_i.

$$Corr = \frac{\mathbf{V}_{D_1} \cdot \mathbf{V}_{D_2}}{||\mathbf{V}_{D_1}|| \, ||\mathbf{V}_{D_2}||} \qquad \text{Where} \quad \mathbf{V}_{D_x} = \frac{\mathbf{X}_{T_2} - \mathbf{X}_{T_1}}{T_2 - T_1} \tag{4.1}$$

This feature was also evaluated both in the keyboard and the hand frames to determine if one performed better at separating the distributions. The resulting data is shown in Figures 4.8a and 4.8b respectively.

Both frames show promising results. The hand frame has removed a bias towards higher correlations, spreading the data out over a wider range and moving it away from the upper limit of 1. The mean and standard deviation of the intended and coactivated distributions for the keyboards frame are 0.43 ± 0.29 and 0.84 ± 0.13 respectively. For the hand frame they are 0.01 ± 0.21 and 0.47 ± 0.24 respectively.

The distributions in the hand frame match more closely to intuitive expectation than the keyboard frame. For example, on average pairs of intended fingers have no correlation. Pairs containing a coactivation show significant correlation. The hand frame is more expensive as it requires tracking of the hand body and transforming of the digit coordinates into the hand frame. Therefore, it may be more practical to use the keyboard frame feature, which has less overhead.

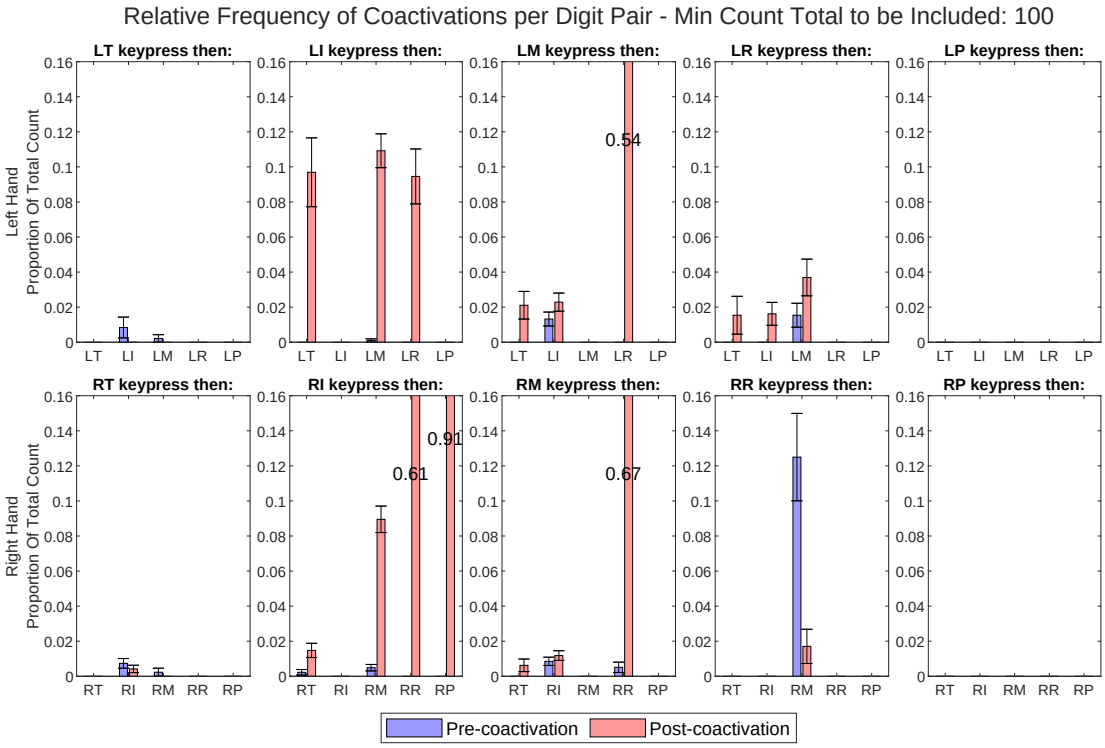


Fig. 4.9 *Digit-pair feature results for the combined data set of participant 0 to participant 11. This shows the ratio of coactivations cause by each pair of digits, of each hand compared to the total number of times that pair of digits is used. Where the total is less than 100 counts the set is removed.*

4.3.5 Digit Distribution Analysis

The digit-pairs feature shows the proportion of times an ordered pair of digits contained a pre-coactivation or a post-coactivation. Figure 4.9 shows normalised proportions and standard errors on those proportions. There are 40 pairs in total, each digit followed by one of the other digits on the same hand. Although a coactivation cannot be caused by the same digit, the case of each digit followed by the same digit is included in the graph for consistency across subplots.

Dividing the limited data into the 40 categories results in few entries per category in some cases. The proportions shown in the graph's subplots also give no indication to the number of elements used to calculate the proportion. It is therefore difficult to extract reliable information from the figures as some proportions rely on few counts. To mitigate this any categories with less than a threshold of 100 total counts were removed.

The graph shows that coactivations of the thumb, middle, ring and pinky digits all occur after the intended use of the index digit. It also shows that the intended use of the middle digit followed by a coactivation of the ring digit makes up a significant proportion of the

times that pair is used. This agrees with one of the observations made by H*ä*ger-Ross et al. [10].

It is difficult to draw reliable insights from this set of subplots as the majority of the coactivations lie in only a few groups. To reduce the number of groups the data was also divided solely by the digit performing the touch event. This is shown in Figure 4.10.

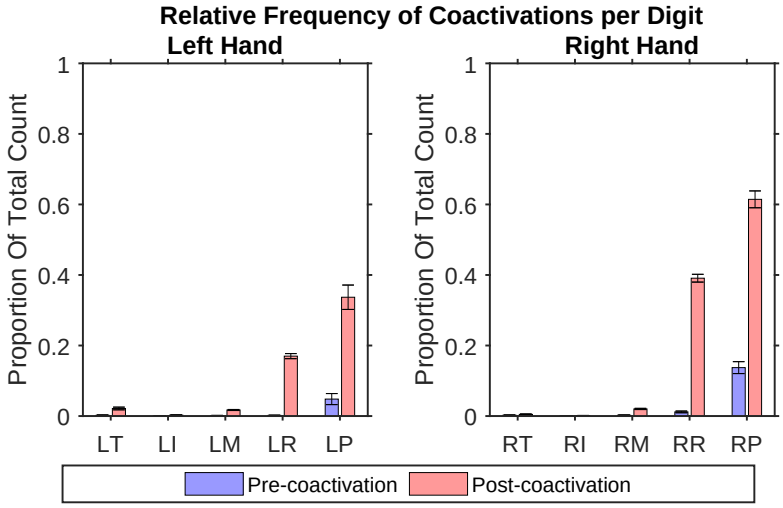


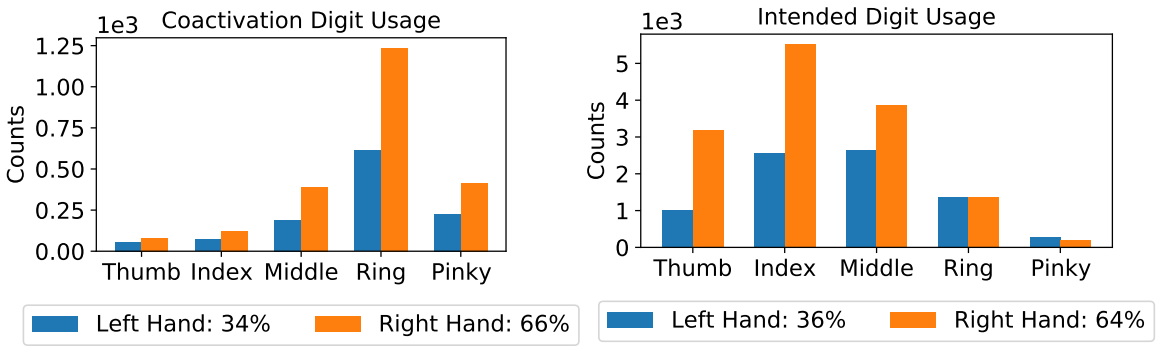
Fig. 4.10 *Proportion of coactivations caused by each digit for the combined data set of participant 0 to participant 11. This shows the ratio of coactivations cause by each digit of each hand compared to the total number of times that digits was used.*

Figure 4.10 shows the ratio of the coactivations to the total number of times that digit was used. This graph shows that of the times the ring and pinky fingers are used they produce a much higher proportion of coactivations than the other digits, which agrees with what was intuitively expected. These are useful results as they show the prior probabilities of each digit causing a coactivation.

To keep the feature vector simple, it did not rely on statistical inference, instead it relies on features mentioned in the previous sections. As this digit data was not to be incorporated in the final model, there was no need to keep an unseen test set. The remaining digit and letter analysis could therefore be calculated across the entire dataset.

To illustrate the distribution of coactivations caused by each digit the coactivating digit data was graphed without normalising by the total counts per digit. This is shown in Figure 4.11a

This figure shows that the majority of the coactivations observed were caused by the ring digit. This was not expected, as it was thought that the pinky digit would contribute the most.



(a) *Count of coactivations*

(b) *Count of intended touch events*

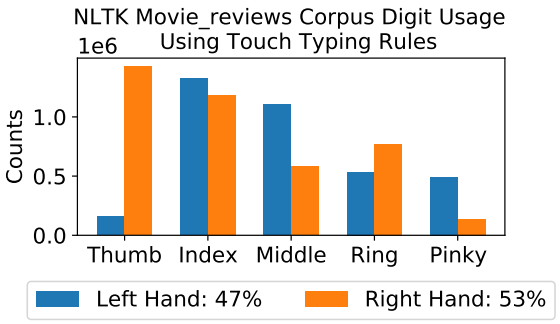
Fig. 4.11 Digit count results for the entire dataset cause by each digit of each hand.

This figure also shows that on average the right hand was responsible for two thirds of the coactivations. This suggests that the right hand may be more prone to coactivations.

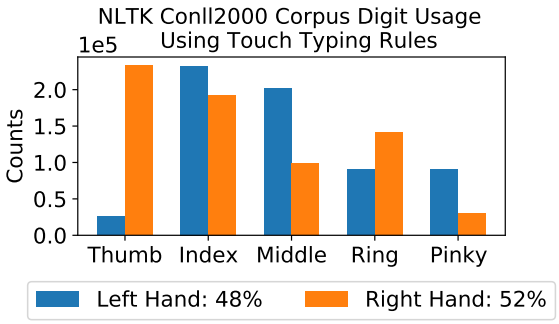
To determine if the proportion of coactivations by hand usage was not simply caused by the right hand being used more than the left, the intended counts were also graphed in the same way. The graph is shown in Figure 4.11b. This shows that indeed the right hand was used on average approximately two thirds of the time and explains why there is a two thirds to one third split in coactivations by hand. This indicates that neither hand is more prone to coactivations.

This poses the question of whether the dataset being used is a good sample of general text with no biases. This was tested by comparing the hand usage of the dataset to two sample corpora, namely the NLTK Movie-reviews and NLTK Conll2000. The digit usage for these corpora were graphed in the same way as before. These corpora do not contain any finger data, so letters were assigned to fingers according to common touch-typing rules. The space bar can be typed with either hand and as the handedness of the participants was not recorded in the original dataset, it is assumed that the space bar counts would be split 90:10 following the commonly accepted split of right handedness to left handedness. The resulting digit usage graphs are shown in Figure 4.12.

The graphs agree very well with one another, showing very similar distributions. They also show that there is approximately a 10% difference in the hand usage compared to the coactivation dataset. As this dataset is dependent on the participants' style of typing, and not all the participants followed the touch-typing style, it is expected that there would be some difference. This is still a reasonable agreement and does not suggest that there are any biases in the dataset.



(a) *Digit usage for the NLTK Movie-reviews corporus.*



(b) *Digit usage for the NLTK Conll2000 corporus.*

Fig. 4.12 Digit usage according to touch-typing rules.

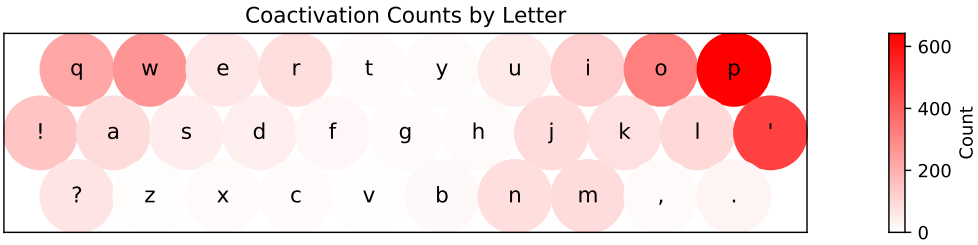
4.3.6 Letter Distribution Analysis

The distribution of coactivations over the letters on the keyboard was analysed to determine which letters or areas of the keyboard had a higher concentration of coactivations. The raw coactivation counts and the normalised coactivation counts by total counts, per letter are shown as heat-maps in Figures 4.13a and 4.13b, respectively.

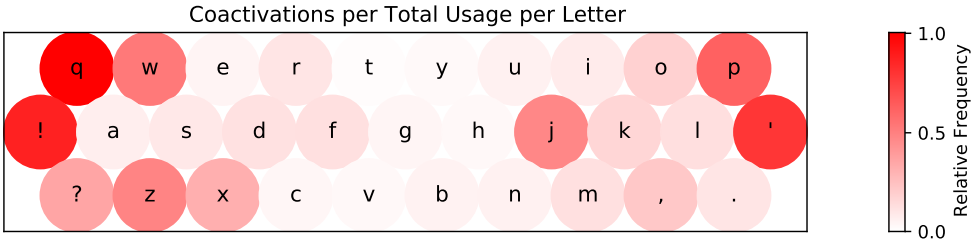
These graphs show that coactivations occur most frequently at the sides of the keyboard around the areas closest to the ring and pinky digits. This agrees with the results of the distribution of coactivations over the digits shown in the last section. It also shows higher counts on the right side, which agrees with the right-left hand usage split seen in the last section. Normalising by total count per letter shows that the majority of the touch events of the edge letters, including “q”, “!”, “p” and “ ’ ”, were coactivations. These are useful results showing the prior probabilities of each letter being coactivated.

As a further check to make sure there were no biases in the dataset, the distribution of intended letter touch events were also graphed using a heat-map. The letter distributions for the two sample corpora, NLTK Movie-reviews and NLTK Conll2000, were also graphed in this way and compared to the coactivation dataset. These are shown in Figures 4.14a, 4.14b and 4.14c respectively.

The letter distribution of the two corpora agree well with each other and agree with the intended touch event distribution. As these letter distributions are not participant dependent, they are better indicator of bias in the dataset. This result shows that the sentences presented to the participants were a good representative sample of English text with no bias towards letters that are more or less likely to be coactivated.

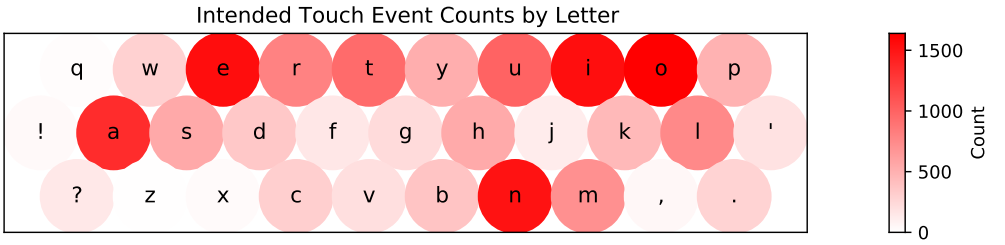


(a) *Total coactivation counts.*

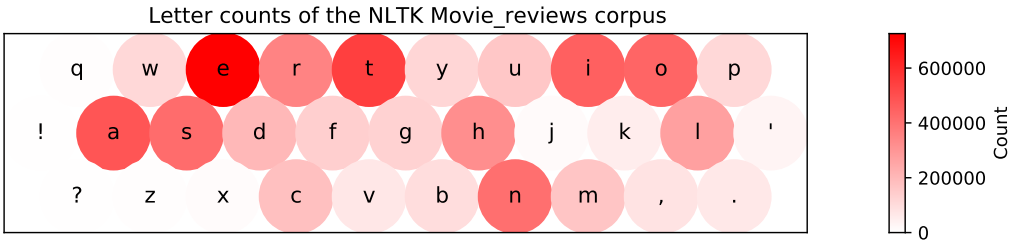


(b) *Normalised coactivation counts.*

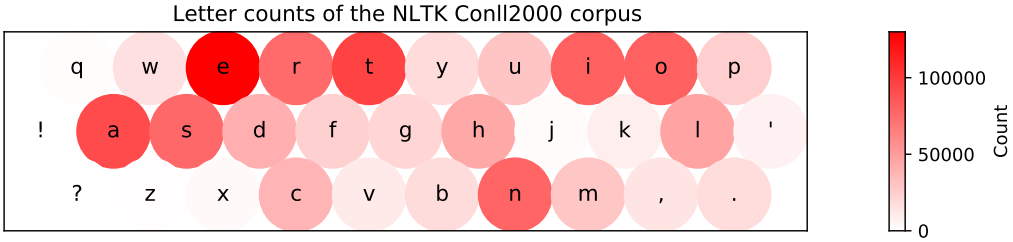
Fig. 4.13 *Coactivation count distribution over the letters of the keyboard.*



(a) *Intended touch event letter frequency distribution.*



(b) *NLTK Movie-reviews corpora letter frequency distribution.*



(c) *NLTK Conll2000 corpora letter frequency distribution.*

Fig. 4.14 *Letter frequency distributions.*

Chapter 5

Detecting Coactivations

5.1 Feature Selection

As mentioned in the Typical Approach and Coactivation Dataset sections, using a large feature vector with all the data for every time step is not feasible. The feature vector therefore needs to be trimmed to only use the most relevant pieces of information.

From comparing the results of the features tested, the touch event depth, touch event velocity, inter-key interval, and average correlation features were selected to be used in the feature vector of the model. These features combine data from multiple time steps extracting useful information and reducing the size of the feature vector.

For the best chance of detecting both pre and post coactivations the features mentioned above would need to be calculated for the current touch points as well as both the previous and next touch points. The features to be used in the feature vector would then be: the depth and velocity of the three key presses in the keyboard frame, the time intervals between the touch points and averaged correlations between the associated digits in the hand frame.

One problem with the above feature vector is its dependence on information occurring after the touch point to being classified. Although this feature vector is likely to produce the best performance, for a model to run easily online, it should run in real time with no such dependencies. Removing information about the post touch event will reduce the model's ability to classify pre-touch-event coactivations. Pre-touch-event coactivations contribute to 8% (392 of 4,627) of the coactivations recorded in the dataset and after filtering, only 89 examples were available for training. As there is so few training examples it is unlikely a model would accurately learn to identify them even with the extra features. Due to their relatively low contribution, the model should still achieve a good performance even if the model is poor at classifying these coactivations.

The feature vector would then include: the velocity of the two touch events in the keyboard frame, the depth of the previous touch event, the time interval, and the averaged velocity correlation in the hand frame. A total of 5 input elements. The depth of the current touch event relied on the current digit's Z position several time steps after the event. For this reason, it was not included.

The averaged correlation results shown in section 4.3.4 also relied on data from a few time steps recorded after the touch event. The extra 50 ms of data included after the second touch event was used to achieve a more accurate average. The correlation calculation was altered to remove these extra time steps. The extra time steps included before the first touch event were also removed to reduce the complexity for implementation. It is not believed that this alteration will affect the features ability to distinguish coactivations.

These alterations limit the features and therefore feature vector to only data prior to the current touch event and post the previous touch event, allowing the model to be more easily incorporated into an online system.

5.2 Data Partitioning into Train and Test Sets

As the size of the data set is limited, how the data is broken into training and test sets will determine how much data is available for training. If multiple models are to be trained, then a portion of the data will also need to be put aside as a validation set. The validation set functions as an unseen dataset used to test various models and parameters so the best model with the best parameters can be selected. This is to ensure the final model can be tested on completely unseen test data so an unbiased estimation of its performance can be determined. Reducing the proportion of data assigned to the test set would allow more data to be used for training and likely result in a better preforming model. However, a reduced test set may result in an unreliable estimation of the model's true performance. Another problem that arises from using a small dataset is that the performance of the model can often depend on which samples are included in the training and test sets. Overfitting can also occur when using such a small dataset.

One way of increasing the accuracy of the estimation and reducing the effect of the other issues is to use K-fold cross validation.

5.2.1 K-fold Cross Validation

As described in the theory section, this technique trains the model multiple times using different portions of the data as the test set and then averages the results. This ensures that the results are not dependent on the choice of test set.

The data could either be split by participant, so all the data from one participant was included in one and only one group, or split so that data from each participant is included in each group. There are advantages and disadvantages in using either technique. In a real-world scenario, the system will encounter users that were never seen before. Keeping each user's data together so that the test set contains unseen users, would more closely match real-world conditions. This would provide a better evaluation of how well the system is likely to preform so this technique was chosen. As there are 24 participants, it was decided to split the data into four groups of six.

5.2.2 Participants Group Assignment

The next step was assign participants to suitable groups. There was high variation between the users in the dataset. Two such variations are coactivations count and typing speed. It was decided to stratify the groups as mentioned in section 2.5. This was done to ensure each group was evenly matched and no group contained an unfair number of training samples. The participants were first grouped to minimise the difference in total coactivation counts per group. This was done by recording the number of coactivations per participant after filtering and searching exhaustively to find the minimum. This search found a few hundred groupings where the maximum different between totals was 1. This was the optimal solution for the total number of coactivations and the number of groups. A secondary criterion was then used to select one set of groupings from those found by the first search. This was chosen to be the grouping that minimised the average words per minute (WPM) across groups. This was chosen as it pairs fast and slow typists together ensuring an even mix of typists in each group. The group statistics of the final groupings are shown in table 5.1.

5.3 Model Selection and Initial Tests

Three models were selected to test. These were a neural network, a support vector machine, and a naive Bayes model. Each of these models was trained and tested using cross validation described in section 2.5. A calibration curve and an ROC curve were graphed for each model and various metrics were calculated to compare the performance of the models. These results are shown in table 5.3.

	Coactivation Count	Average Words Per Minute	Average Error Rate
Group 1 Participants: 0 10 13 14 15 18	850	34.5	7.6
Group 2 Participants: 1 2 3 16 17 22	851	34.1	7.4
Group 3 Participants: 4 8 11 12 21 23	850	33.8	7.1
Group 4 Participants: 5 6 7 9 19 20	851	33.8	7.1
Max - Min	1	0.7	0.5

Table 5.1 *Statistics of the final groupings used for K-fold cross validation. The Max - Min row contains the maximum element minus the minimum element of each column.*

5.3.1 Neural Network

The neural network was the first model selected because it is widely used and has been shown to be very effective in machine learning tasks. Their architecture is also very adaptable and can therefore be tailored to specific tasks. Though the task of detecting coactivations is inherently a time dependent problem, it was decided to address the problem with a time independent solution. This was done because time dependent solutions are often more complex and more difficult to train. The time independent features investigated also showed effective identification of coactivations and so a time independent solution had the potential to offer a reliable solution.

Using the features described, the task of identifying coactivations was reduced to a much simpler problem. Due to this simplification, a small fully connected neural network consisting of only a few layers was chosen, as it was likely to achieve good performance without over fitting. A few preliminary tests were performed on a few variations of the base architecture outlined below. These were not extensive as to avoid data leakage into the chosen model. These variations included increasing the number of layers and the number of nodes in these layers. No substantial improvements were observed and as these variations had more parameters to train, the simpler base model was selected. These tests did show that using batch normalisation and dropout layers had a significant impact on the performance

of the model and so they were kept. This impact was expected and why these layers were included in the base model. Batch normalisation has been shown to reduce training time [12] and both batch normalisation and dropout can also help reduce overfitting. This is important when training on small datasets.

The architecture chosen comprised of the following layers; 32 node dense with batch normalisation, ReLU activation and 20% dropout, followed by another 32 node dense with batch normalisation, ReLU activation and 20% dropout, then a 16 node dense with batch normalisation, ReLU activation and 20% dropout, and finally a 1 node dense with a sigmoid activation function. This network had just less than 2,000 trainable parameters. The model was trained on three of the groups and used the fourth for testing and validation. For training, all the coactivations and an equal number of intended touch events were selected from each participant in the training group. This ensured the model was trained on an equal number of positive and negative cases. This was done to remove bias towards one class in the training set. As a result, about 7500 data points were used in the training of the model's 2,000 parameters, 3.75 data points per parameter. This ratio could have been increased by reducing the size of the network but as it was already quite small it was decided not to alter the model and that this low ratio was acceptable. For testing, the entire test set was used, including all the intended touch events from each participant. In a real world test the number of coactivations and intended touch events are not going to be evenly split so the test group should also reflect this. The loss function chosen was binary cross-entropy as it was the most suitable for this type if problem.

Calibration of the output of the model was also an important consideration. As this model was intended to be incorporated into a statistical decoder the output was interpreted as the probability of a touch event being a coactivation. The cross-entropy loss function is a probabilistic loss and is minimised when the model's predictions approximates the ground truth conditional distribution [9]. As desired, this loss favours model configurations whose predictions match the underlying probability.

To test the effectiveness of each feature, the feature vector was reduced to just the feature being tested. The model was trained and tested using cross validation. A confusion matrix was created for each fold of the validation and used to calculate the mean and standard deviation of the accuracy, precision, recall and F1 score. These evaluation metrics are given for each feature in table 5.2.

The correlation feature can be calculated using the velocity of the digits in the keyboard or hand frame. As both frames showed promising results in earlier analysis, both were tested and the results are shown in 5.2.

	Accuracy	Precision	Recall	ECE	AUC
Previous Digit Depth	0.58 ± 0.07	0.20 ± 0.07	0.64 ± 0.11	0.38 ± 0.05	0.64 ± 0.05
Current Digit Velocity	0.61 ± 0.03	0.22 ± 0.03	0.72 ± 0.01	0.39 ± 0.02	0.70 ± 0.02
Previous Digit Velocity	0.54 ± 0.16	0.14 ± 0.04	0.47 ± 0.21	0.35 ± 0.03	0.50 ± 0.03
Interval	0.96 ± 0.02	0.80 ± 0.11	0.96 ± 0.01	0.03 ± 0.02	0.98 ± 0.01
Correlation Keyboard Frame	0.85 ± 0.02	0.48 ± 0.07	0.76 ± 0.05	0.21 ± 0.04	0.88 ± 0.01
Correlation Hand Frame	0.86 ± 0.04	0.52 ± 0.12	0.81 ± 0.05	0.20 ± 0.05	0.90 ± 0.02

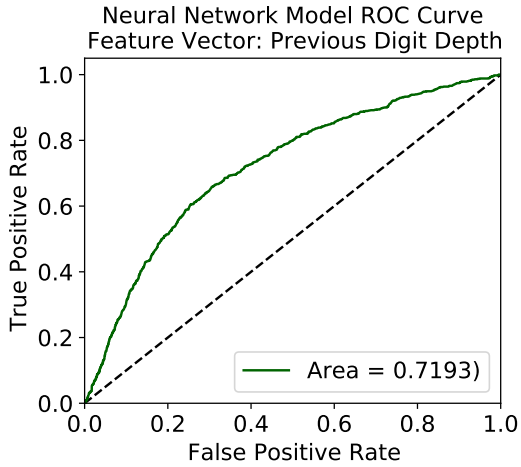
Table 5.2 *Table of metrics showing the performance of various features.*

The ROC curves for the first fold of the cross validation, for the features: previous digit depth, interval, current and previous digit velocities in the keyboard frame, and correlation in the keyboard and hand frame, are shown in Figures 5.1a, 5.1b, 5.1c, 5.1d, 5.1e, and 5.1f respectively.

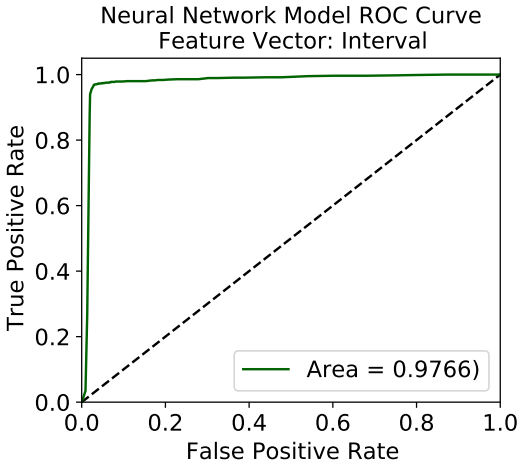
From examining the evaluation metrics shown in table 5.2 and the ROC curves, the interval feature showed the best performance and the previous key velocity feature performed the worst, only marginally better than random guessing. The remaining features also showed good performance. As a result, it was decided to remove the previous key velocity feature from the feature vector. This was done as its ROC curve showed it contributed very little to the classification prediction and reducing the feature vector would increase the data points per parameter potentially improving the fit of the other features. It was also decided to use the keyboard frame correlation over the hand frame. This was due to the keyboard frame achieving slightly higher performance metrics and area under the ROC curve. The keyboard frame also has a lower computational overhead and does not require the tracking of the hands. The final feature vector composed of the following features; previous depth, current velocity in the keyboard frame, interval, and average velocity correlation also in the keyboard frame.

5.3.2 Support Vector Machine

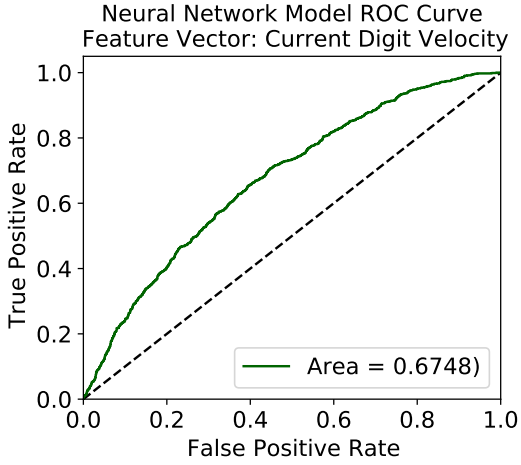
The second model selected was a support vector machine (SVM). This is a commonly used classifier that is well understood. It serves as a good baseline to which the neural network



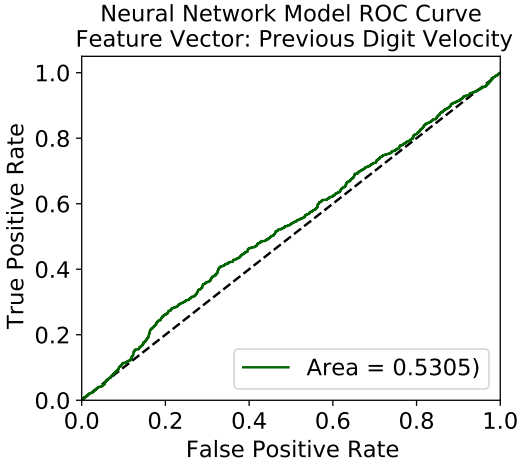
(a) *Trained on only the previous depth feature.*



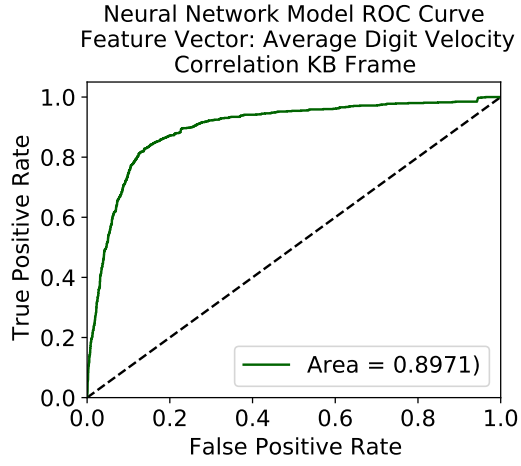
(b) *Trained on only the interval feature.*



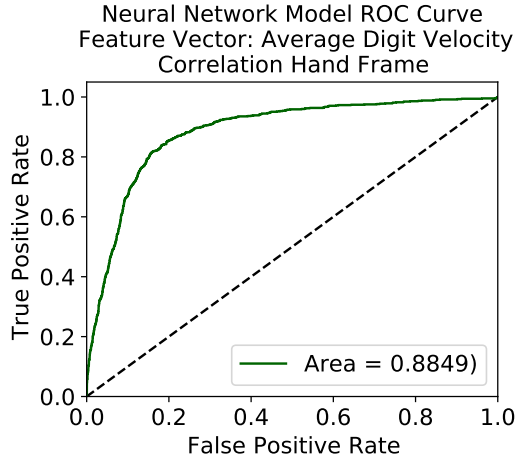
(c) *Trained on only the current velocity feature.*



(d) *Trained on only the previous velocity feature.*



(e) *Trained on only the average velocity correlation feature in the keyboard frame.*



(f) *Trained on only the average velocity correlation feature in the hand frame.*

Fig. 5.1 *ROC curves for potential feature vector elements.*

can be compared. A linear kernel was chosen for simplicity and was the only kernel tested. More complex kernels may have led to a better preforming model but as the main purpose of this model was as a comparison for the neural network, this was unnecessary. The SVM was trained on the same final feature vector containing the four chosen features.

Calibration of the SVM model was also considered as they are non-probabilistic classifiers and so do not necessarily output calibrated probabilities. Platt scaling was used for the calibration as it has been shown to produce comparable or reduced error rates compared to an uncalibrated SVM [16].

5.3.3 Naive Bayes

The third and final model was a naive Bayes model. This is another commonly used well understood classifier. It differs from the SVM and neural network in that it is a probabilistic classifier, outputting well calibrated probabilities. It also provides a good baseline to which the other models can be compared.

The model was constructed using Bayes rule, the empirical probability of a coactivation occurring and the conditional distributions of the four features selected for the final feature vector. The probability of a coactivation occurring was equated to the proportion of coactivations that occurred in the dataset. The conditional distributions of the features were found by first normalising the histograms shown in Figures 4.4a, 4.5a, 4.7 and 4.8a, and then fitting curves to each distribution. This was done from both the coactivation and intended touch event distribution. The normalising and fitting were done using MATLAB's inbuilt function fitter. The functions chosen were the ones that produced the highest likelihood. In all but three cases the generalised extreme value distribution produced the best fit. The first exception was the coactivation distribution of the previous digit depth, for which a student's t distribution with offset and scaling was used. The other two were the intended and coactivation distributions of the keyboard frame averaged velocity correlation, in which extreme value distributions were fitted. These were combined using Bayes theorem to calculate the probability of a coactivation given the values of the four features.

5.4 Model Evaluation

5.4.1 Final Tests

To evaluate the final performance, the selected models were trained on the chosen features and tested using the same cross validation splits to insure consistency. This was done using cross validation in the same way the individual features were tested using the neural network

	Neural Net	SVM	Naïve Bayes
Accuracy	0.978 ± 0.011	0.973 ± 0.011	0.976 ± 0.005
Precision	0.882 ± 0.072	0.854 ± 0.065	0.922 ± 0.028
Recall	0.968 ± 0.009	0.971 ± 0.010	0.902 ± 0.021
F1 Score	0.921 ± 0.044	0.907 ± 0.042	0.912 ± 0.023
ECE	0.027 ± 0.005	0.046 ± 0.007	0.011 ± 0.004
AUC	0.990 ± 0.006	0.985 ± 0.006	0.984 ± 0.008

Table 5.3 *Metrics showing the performance of the three models tested.*

described above. A confusion matrix was created for each fold of the validation and used to calculate the mean and standard deviation of the accuracy, precision, recall and F1 score. This was repeated for each model and the resulting evaluation metrics are given in table 5.3.

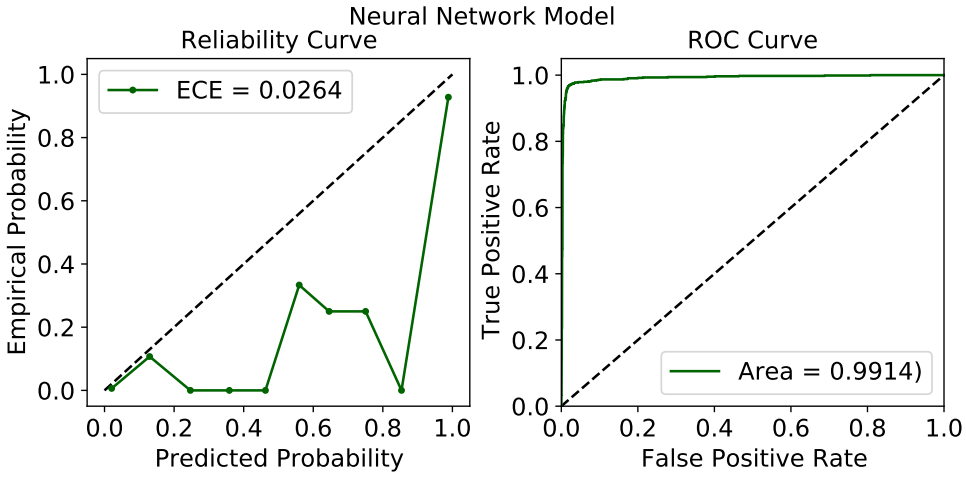
The ROC curves for each fold of the cross validation were plotted and the area under the curve was calculated for each. The mean and standard deviation was calculated and is shown in table 5.3. This was repeated for each model. The ROC curves for each fold of each model were very similar so only the first fold of each model is shown. These are shown on the right of Figures 5.2a, 5.2b and 5.2c, for the neural network, SMV and naive Bayes, respectively.

To evaluate the over or under confidence of each model's predictions, a calibration curve was plotted and expected calibration error (ECE) for each fold was calculated. The mean and standard deviation of the ECE was calculated for each model and also shown in table 5.3. Like the ROC curves the calibration curves for each model were very similar so again only the first fold of each model is shown. These are shown on the left of Figures 5.2a, 5.2b and 5.2c, for the neural network, SMV and naive Bayes, respectively.

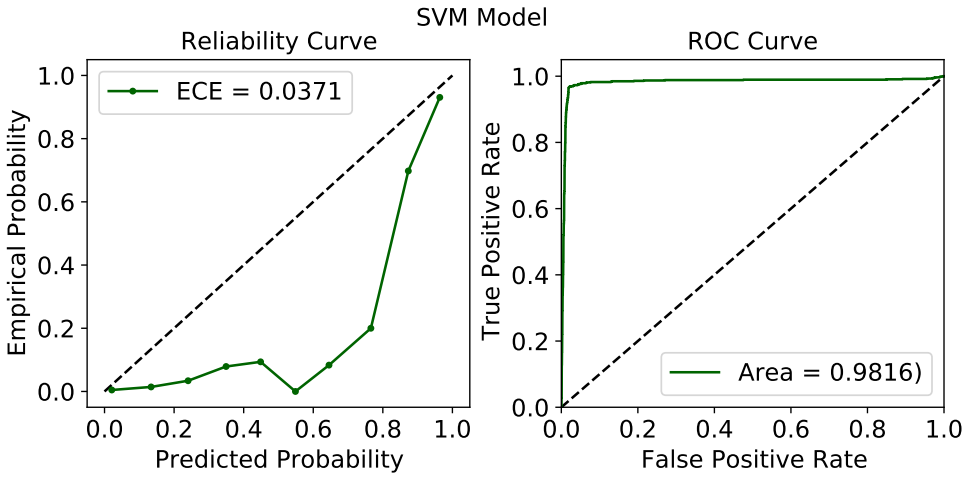
5.4.2 Model Comparison

From comparing the calibration and ROC curves as well as the performance metrics, all models show very good results. None of the models have significantly higher accuracies than any of the others. The naive Bayes model's precision is significantly higher than the SVM, but its recall is significantly lower. This shows the SVM and naive Bayes trading off between false positives and false negatives. The neural network's precision and recall sits between these two. Combining the precision and recall into the F1 scores also shows no significant difference between any of the models and neither does the area under the ROC curves.

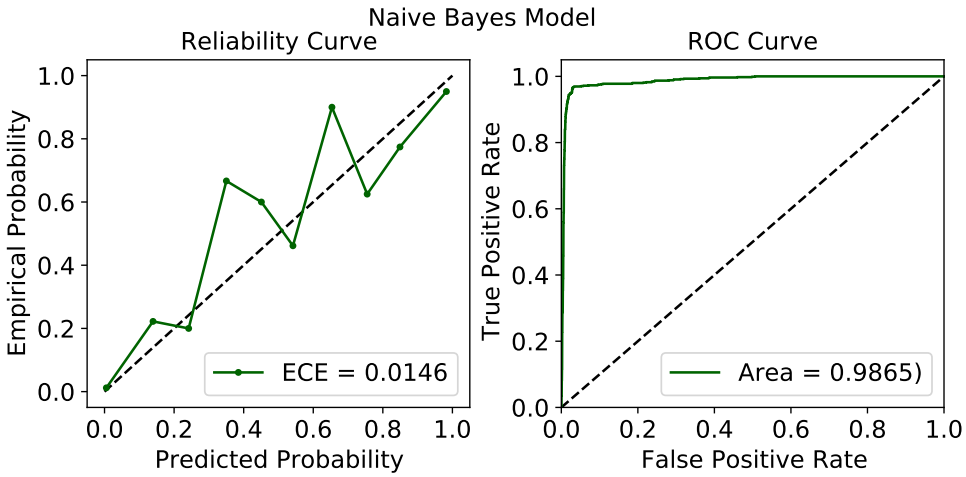
Even though the accuracy, F1 score and area under the ROC curve of the neural network were not significantly higher than the other models, they were still the highest in all three



(a) *Neural network calibration and ROC curves.*



(b) *Support vector machine calibration and ROC curves.*



(c) *Naive Bayes calibration and ROC curves.*

Fig. 5.2 *Calibration and ROC curves of the three models trained on the final feature vector consisting of the previous depth, current velocity, interval, and averaged velocity correlation, all in the keyboard frame.*

categories. The neural network was therefore chosen as the model to implement into the decoder.

Lastly the calibration curves and the ECE show that the naive Bayes model produced the most calibrated probabilities, following the ideal line reasonably well. This was expected as the naive Bayes was the only probabilistic model. The next best was the neural network and finally the SVM. Both calibration curves show that they overpredict across the full range of probabilities between zero and one. It was expected that calibrating the SVM would produce better calibrated predictions but as Platt scaling is better suited to scaling S-shaped calibration curves it is not surprising that the calibration did not perform well in this case.

5.5 Real-Time System Simulation

The final step was to implement the chosen model into an existing decoder. As a first test, a simulation was constructed incorporating the prediction model into coactivation predictor system. This showed that the model could run in real time and on a stream of input data. To best approximate an input stream, the system iterated over the recorded data only once and used only the current sample and what was saved from previous samples to predict the probability of each touch point being a coactivation. This was successfully done and showed the model can run in real time.

To show the system working the Z position of the right-hand digits were plotted against time for a section of one task (Figure 5.3). Overlaid on these plots was the time of each touch event, given as black vertical lines with the letter that was activated at the top. To indicate which digit activated the letter a green or red marker is included. Where a vertical line is missing a marker, the letter was activated by the left hand. The markers indicate the ground truth of whether a touch event was intended (green), or a coactivation (red). Beside each marker is the systems coactivation prediction for the touch event. Figure 5.3 shows the original text typed, “tomoorrokw wipth” and shows the system correctly identifying the coactivations “o”, “k” and “p”. Removing these letters returns the correct phrase "tomorrow with". This figure also show that the system can correctly identify coactivations by different digits, as the features are digit independent.

The system performs poorly on pre-coactivations as expected (Figure 5.4). It correctly identifies the pair containing the coactivation but wrongly predicts which is the intended and which the coactivation.

After the simulated run was completed, the model was incorporated into an existing statistical decoder. This was done by converting and exporting the Keras model as an ONNX model, a format supported across a wide range of platforms. To make the model as easy as

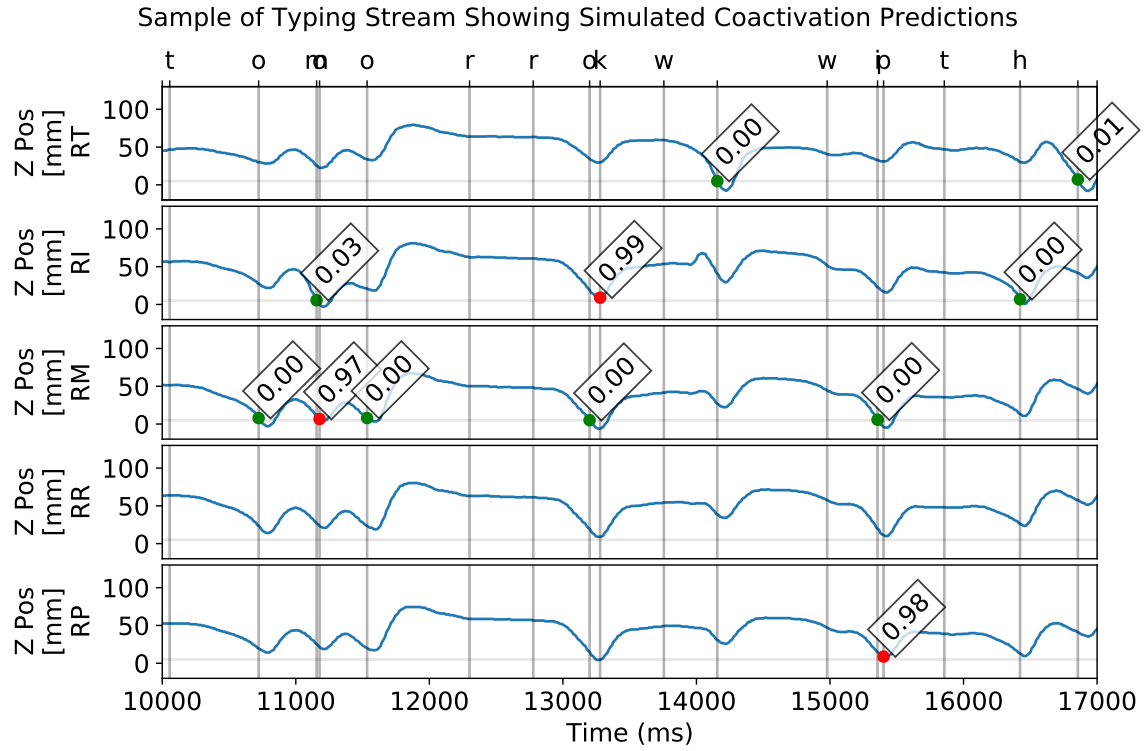


Fig. 5.3 *Z position of the right hand digits plotted against time, showing model coactivation predictions (0: intended, 1: coactivated) and ground truths (green: intended, red: coactivated).*

possible to incorporate into existing applications, all scaling and conversions were built into the model itself using custom scaling layers before being converted to the ONNX format. This format allowed the model to be loaded into a C# class library where predictions could easily be made by passing the unscaled features to the class.

The decoder operates as follows; at each touch event the probability of that touch event being a coactivation is calculated, if the probability is greater than a certain threshold the character is removed from the textbox displayed to the user, but could still be passed to the decoder. If the system has been configured to do so, the character is removed from the text that will be passed to the decoder. If the character passes the threshold or is not removed, the probability of it being a coactivation is combined with a weighting factor and combined with the likelihood that the character is an insertion error. This is done for each character until a space or other word ending character is activated. The text is then passed to the decoder, along with the deletion probabilities, where is it corrected to the word that is most likely. Insertions and deletions have an associated penalty which is set in the decoder's configuration. The most likely word is the one that has the lowest tallying of insertion and

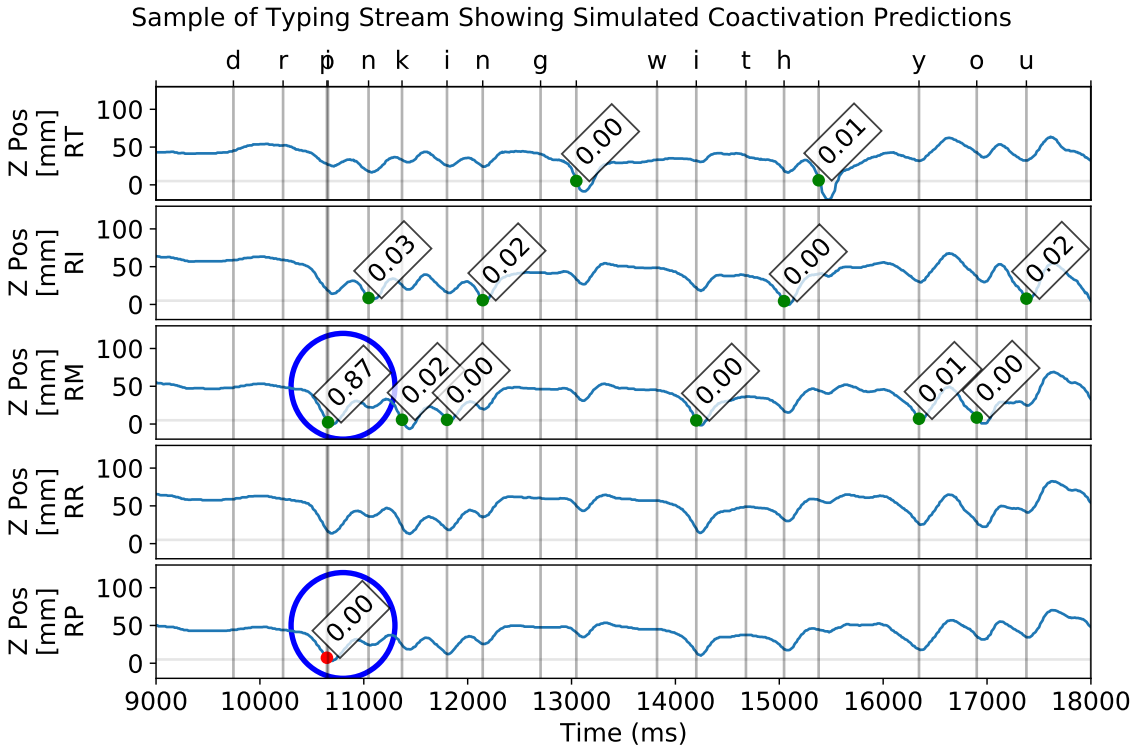


Fig. 5.4 *Z position of the right hand digits plotted against time, showing model coactivation predictions (0: intended, 1: coactivated) and ground truths (green: intended, red: coactivated).*

deletion penalties needed to convert the input text to the corrected word. The decoder's configuration hyperparameters are: a character removal threshold, a flag which indicates whether characters over this threshold are passed to the decoder, a coactivation weighting factor, an insertion penalty, and a deletion penalty. The character error rate (CER) is then calculated to determine how many further insertion, deletion or substitution corrections are needed to convert the final text into the stimulus text the user was trying to replicate. These five hyperparameters had to be chosen and optimised to achieve the lowest character error rate.

The coactivation detection portion of the decoder was then evaluated by replaying a test set of pre-recorded data samples and comparing the character error rate with and without the coactivation detection enabled. These tests showed promising results but due to time constraints further evaluations could not be completed. Only a small test set was tested and optimal settings of the hyperparameters were not found. The positive results of the individual models indicate that the modified decoder is likely to achieve lower error rates, but this will have to be verified and quantified in a future study.

Chapter 6

Discussion

6.1 Limitations

There are a few limitations of the chosen final model and its features, which were a result of training data constraints.

The model is limited by the simplicity and relatively small number of features. This was a design choice based on the size of the dataset. As a result, most of the raw digits' motion is not available to the model or is only available in a simple condensed version. The model could have benefited from more of this information if there was enough data to train a larger model with a larger input vector. The small model used also limits the complexity of the relationship between the features that can be learned. Only a few chosen features could be tested and so there may be other useful features that were not found. The model did not incorporate any prior knowledge about the likelihood of certain digits to produce more coactivations or certain letters to be more heavily coactivated. The model also did not take advantage of the time-series nature of the problem which could have led to more accurate predictions.

Another limitation is the ability to accurately evaluate the model's performance on coactivations preceding the intended touch event. This was due to the limited number of sample pre-coactivations in the data set, 389. The number was further reduced to 89 by filtering, though it is not known why filtering removed so many samples. With so few examples of this type to test, a statistically significant accuracy could not be calculated. It is noted that over the course of the final K-fold training and testing, across the 4-fold models, a total of 16 of the 89 examples were correctly labelled as coactivations. This shows that qualitatively the model can identify pre-coactivations, even if a quantitative accuracy cannot be reliably found.

6.2 Design Implications

This project has shown that identification and removal of coactivations from a text input stream can be completed with simple lightweight features. It was shown that the removal of coactivations is a processing step that can be implemented into new and existing systems. Four features have been identified that can aid the detection of coactivations. The simplest and most effective is the interval between touch events. This feature also indicates that the simplest step that can be taken to prevent coactivations is to simply add a minimum time before the next letter can be selected of 50–100 ms.

Another simple step that can be taken is to limit the digits or pairs of digits that can be typed with. For example, restrict typing to only the index, middle and ring digit. As a significant portion of the touch events produced by the pinky digit are coactivations it may be beneficial to limit a user by not allowing the use of their pinky digit. Another example would be to prevent the ring digit from selecting a key if the middle finger of the same hand was just used. It was found that the majority of the times the ring digit was used after the middle digit on the same hand, a coactivation was produced. Eliminating these inputs would produce lower error rates but would also be unintuitive to a user and may cause other problems.

The best results would be obtained by constructing a model or using the model created in this project, that uses all four features and incorporating it into the system. These features are easily calculated and do not require an exact formula. Implementing the model presented here should be easy to implement as the model is saved in the cross platform ONNX format.

6.3 Future Work

There are many improvements and extensions that could be made to this project given more time and data resources. The most significant would be to perform an in-depth search for the optimal setting of the decoder's hyperparameters. This would allow a real-world test of the coactivation detection model and determine the contribution it would have to an autocorrect system.

The model could also be extended to include prior likelihoods of digits to cause coactivations or characters to be coactivated. This would add a statistical element to the model potentially increasing the accuracy or reliability of the model. Increasing the dataset would also produce a more accurate model and would potentially capture enough pre-coactivations to thoroughly evaluate the model's ability to detect pre-coactivations. Choosing a model that took advantage of the time-dependent nature of the problem could prove more accurate. It

may also allow digit positions to be passed directly into the model reducing the overhead computation and increasing the usability of the model.

Finally, once the optimal setting of the decoder's hyperparameters were found a real-world test of the modified autocorrect system could be done to test accuracy, usability, and intuitiveness as well as to gain any other user feedback.

Chapter 7

Conclusion

This project successfully answered the two proposed research questions. In reference to Research Question 1, the important characteristics of coactivations were found to be the following: inter-key interval, key depth, key velocity, and inter-key correlation. It was also found that the highest proportion of coactivations were caused by the ring digits followed by the pinky digits. In addition, it was found that the upper edge letters of the keyboard were the most likely to be coactivated. The characteristics deemed mostly likely to effectively identify coactivations were tested though there may be others that are more complex or have a higher computational cost. The ones found had a low computational cost and were shown to be effective at detecting coactivations. They therefore met the requirements of this project.

In reference to Research Question 2, coactivations can indeed be effectively detected. The final model developed successfully identified $97 \pm 1\%$ of coactivations—mean and standard deviation across the 4-fold cross validation—while only incorrectly labelling $2 \pm 1\%$ of intended touch points as coactivations. As 4% of characters typed were coactivations, this equates to a $2 \pm 1\%$ reduction in errors. This result is based on data collected from a group of 24 participants. The approach taken was chosen due to the size of the dataset available. A larger dataset would have supported other approaches using more advanced models that may further improve the results.

The project achieved its aim of integrating the final model into an existing statistical decoder. It has demonstrated that simple lightweight features can be used to identify coactivations and that this removal step can be implemented into new and existing auto-correctors, like the decoder. Future work includes a full evaluation of the modified statistical decoder, which is intended to be completed in the near future.

References

[1] Bowman, D. A., Rhoton, C. J., and Pinho, M. S. (2002). Text input techniques for immersive virtual environments: An empirical comparison. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 46(26):2154–2158.

[2] Cerliani, M. (2020). Neural network calibration with keras.

[3] Dhakal, V., Feit, A. M., Kristensson, P. O., and Oulasvirta, A. (2018). Observations on typing from 136 million keystrokes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–12, New York, NY, USA. Association for Computing Machinery.

[4] Dudley, J., Benko, H., Wigdor, D., and Kristensson, P. O. (2019). Performance envelopes of virtual keyboard text input strategies in virtual reality. In *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 289–300.

[5] Dudley, J., Vertanen, K., and Kristensson, P. (2018). Fast and precise touch-based text entry for head-mounted augmented reality with variable occlusion. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 25(6):1–40.

[6] Fish, J. T. and Soechting, J. F. (1992). Synergistic finger movements in a skilled motor task. *Experimental Brain Research*, 91:327–334.

[7] Grubert, J., Witzani, L., Ofek, E., Pahud, M., Kranz, M., and Kristensson, P. O. (2018a). Effects of hand representations for typing in virtual reality. pages 151–158. IEEE.

[8] Grubert, J., Witzani, L., Ofek, E., Pahud, M., Kranz, M., and Kristensson, P. O. (2018b). Text entry in immersive head-mounted display-based virtual reality using standard keyboards. pages 159–166. IEEE.

[9] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks.

[10] Häger-Ross, C. and Schieber, M. H. (2000). Quantifying the independence of human finger movements: Comparisons of digits, hands, and movement frequencies. *Journal of Neuroscience*, 20(22):8542–8550.

[11] InformedHealth.org [Internet]. Cologne, Germany: Institute for Quality and Efficiency in Health Care (IQWiG) (2018). How do hands work?

[12] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift.

[13] Moore, K., Dalley, A., and Agur, A. (2013). *Clinically Oriented Anatomy*. Clinically Oriented Anatomy. Wolters Kluwer Health/Lippincott Williams & Wilkins.

[14] NaturalPoint inc. (2020). Optitrack prime 13.

[15] Ozdemir, S. (2016). *Principles of Data Science.* Packt Publishing.

[16] Platt, J. (2000). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Adv. Large Margin Classif.*, 10.

[17] Schieber, M. H. and Santello, M. (2004). Hand function: peripheral and central constraints on performance. *Journal of Applied Physiology*, 96(6):2293–2300. PMID: 15133016.

[18] Taylor, C. L. and Schwarz, R. J. (1955). The anatomy and mechanics of the human hand. *Artificial limbs*, 2(2):22–35.

[19] Zheng, A. (2018). *Feature engineering for machine learning : principles and techniques for data scientists / Alice Zheng and Amanda Casari.* First edition. edition.