Novel use of machine learning tools for basecalling of Nanopore reads

CS39440 Major Project Report

Author: Ferenc Szpisjak ([fes9@aber.ac.uk](mailto:fes9@aber.ac.uk))

Supervisor: Dr Wayne Aubrey (waa2@aber.ac.uk)

23th February 2021

Version 1.0 (Draft)

This report is submitted as partial fulfilment of a BSc degree in  
Artificial Intelligence and Robotics (GH76)

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Wales, UK

Declaration of originality

I confirm that:

* This submission is my own work, except where clearly indicated.
* I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
* I have read the regulations on Unacceptable Academic Practice from the University’s Academic Registry (AR) and the relevant sections of the current Student Handbook of the Department of Computer Science.
* In submitting this work, I understand and agree to abide by the University’s regulations governing these issues.

Name …………………………………………

Date ……………………………………………

Consent to share this work

By including my name below, I hereby agree to this project's report and technical work being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name …………………………………………

Date ……………………………………………

Acknowledgements

I am grateful to…

I’d like to thank…

Abstract

Include an abstract for your project. This should be approximately 300 words.

The abstract is an overview of the work you have done. Highlight the purpose of the work and the key outcomes of the work.

Contents

1 Background, Analysis & Process 6

1.1 Background 6

1.2 Analysis 6

1.3 Process 6

2 Experiment Methods 7

3 Software Design, Implementation and Testing 8

3.1 Design 8

3.1.1 Overall Architecture 8

3.1.2 Some detailed design 8

3.1.3 User Interface 9

3.1.4 Other relevant sections 9

3.2 Implementation 9

3.3 Testing 9

3.3.1 Overall Approach to Testing 10

3.3.2 Automated Testing 10

3.3.3 Integration Testing 10

3.3.4 User Testing 10

4 Results and Conclusions 11

5 Critical Evaluation 12

6 Annotated Bibliography 13

7 Appendices 14

A. Third-Party Code and Libraries 15

B. Ethics Submission 16

C. Code Samples 17

# Background, Analysis & Process

This section should discuss your preparation for the project, including background reading, your analysis of the problem and the process or method you have followed to help structure your work. It is likely that you will reuse part of your outline project specification, but at the end of the project you should have more to discuss.

**Notes**:

* All of the sections and text in this example are for illustration purposes. The main Chapters are a good starting point, but the content and actual sections that you include are likely to be different.
* Look at the document MMP\_SO8 Project Report and Technical Work [7] for additional guidance.

## Background

**What was your background preparation for the project? What similar systems did you assess? What was your motivation and interest in this project?**

### New generation of basecalling

Nanopore sequencing, is the next generation technology for DNA sequencing. Before its existence, DNA sequencing methods such as Sanger’s made use of a machine that was similar in size to that of an ATM machine. While it had an accuracy of 99.9%, this older technology did not allow a sample to be sequenced in the field, the sample had to be brought back for analysis to for instance a lab. Nanopore sequencing allows the use of devices not much bigger than a USB flash drive used together with a laptop, to sequence DNA. This is made possible not only because of the nanopore technology, but the machine learning algorithms that are behind it. This technology however is not yet perfect, the accuracy rate of state-of-the-art sequencers is 95%.

Researches read pointed out that the most error prone part of basecalling is the segmentation. Segmentation in this context means deciding on parts of the raw data that make up one base. This is a non-trivial task for two main reasons. First problem is that the bases do not pass through the nanopore at a constant speed, under the same time period 5 to 15 bases can pass through. The second problem is that the raw signal is not only based on the base that is in the middle of the nanopore, but bases that come before and after the one in the middle also influence the raw signal.

### Chiron

Researches were done by (…). They made their own basecaller called Chiron. To attempt to improve basecalling accuracy, Chiron introduced the idea of basecalling without segmenting the data. Instead of segmentation, Chiron does basecalling directly from the raw data.

### UR-Net

Another approach to dealing with high segmentation error rate, is to improve segmentation. The research published by BMC bioinformatics, addressed the poor segmentation performance mentioned earlier in a different approach, by proposing a new custom network model for segmentation. This is based on the U-Net neural network, that is commonly used for image recognition. U-Net is a convolutional neural network, that is 2 dimensional, because for image segmentation a two-dimensional approach is used. However, the segmentation task for basecalling is a one-dimensional task, and for this reason, the base U-Net had to be refined. This U-Net optimised for basecalling is called UR-Net, it stands for refined U-Net.

### DeepNano

Another research paper, named DeepNano investigates different problems. There are multiple tasks (such as infectious disease detection) that could be done with the help of Nanopore sequencing, however they cannot be done with the official toolkit. In order to be able to develop applications for such tasks, a good open-source basecaller is needed, as the official tools are not open-source. For this reason, they created an open-source basecaller called DeepNano. It also mentions that a new version of the MinIon sequencer is able to generate 2D reads. This was introduced by the official creators Oxford Nanopore Tech to reduce error rate. It produces 2D reads by reading both side of the strand of DNA. This can reduce error rate, because if one side of the strand is known, its pair can be predicted. Adenine(A) always bonds with Thymine(T), while Cytosine(C) always bonds with Guanine(G). This makes it easier to pinpoint potential errors, if the pairs do not match. It uses a combination of three types of neural networks to deal with such reads. Simple recurrent neural networks are common to be used for basecalling. DeepNano introduces the use of bidirectional recurrent neural networks. This means that the network can use its most recently acquired knowledge first in backwards run, instead of just being able to start with the oldest knowledge. DeepNano also uses Gated recurrent units, which also was not commonly associated with basecalling.

### Bulkvis

BulkVis is a tool to visualise basecalling data. Usually, every basecall is stored in a new file, and they cant be put together with the original kit. BulkVis allows the user to visualize multiple basecall files in bulk. This was developed to investigate two cases: they theorized that switching to a new file can result in data loss, which can impact the overall performance of basecalling. The second concerns unblocking. Unblocking is reversing the movement of the motor protein, thus ejecting bases. This is useful for two reasons: Firstly, if we require a “read until” read, when the goal is reached, the sequencing stops. The other use is removing blockages in the pore. According to this paper, unblocking can cause a change in the electric current for about 2 seconds after it occurs. But using the generic tools this cannot be investigated, because the data after unblocking is not saved.

## Analysis

**Taking into account the problem and what you learned from the background work, what was your analysis of the problem? How did your analysis help to decompose the problem into the main tasks that you would undertake? Were there alternative approaches? Why did you choose one approach compared to the alternatives?**

**There should be a clear statement of the research questions, which you will evaluate at the end of the work.**

**In most cases, the agreed objectives or requirements will be the result of a compromise between what would ideally have been produced and what was felt to be possible in the time available. A discussion of the process of arriving at the final list is usually appropriate.**

The researches previously mentioned defined the problem to some extent. Under time period x sometimes 5, sometimes 15 bases pass through the nanopore. What is more, the signal strength is not only influenced by the one base just passing through, it is also influenced by bases that came before and after. To make basecalling accuracy better, either methods that basecall without segmentation, or methods that do segmentation more accurately are needed.

A good tool to evaluate different machine learning methods for either option, is WEKA. The first challenge is to be able to import basecalled bases and raw signal data pairs produced by the albacore basecaller, into WEKA. This is not straightforward because to store basecalling data a format called fast5 is commonly used that is based on the HDF5 format, while WEKA supports its own, ARFF (Attribute-Relation File Format) file format.

Fast5 files cannot be viewed with common text viewers/editors. They require specialised tools that can export them as commonly used file extensions.

Once the data is in a much more accessible format, it will be much easier to devise steps for the data to be transformed into an ARFF format for WEKA.

After the data can be loaded into WEKA, different approaches and tools can be evaluated. What approach means in this context, is the concept of representing this type of biological data in the WEKA format. ARFF data is made up of instances, each instance having their own attributes and a class. It has to be investigated how the bases translate to classes. Also, all instances in ARFF have to have the same number of attributes. This however is not the case for basecall data, because different reads have different length, partly because of the varying speed of bases. At this point, it is not known for me, how different are the quantity of attributes. This is to be investigated, together with a method that is able to import data with a varying number of attributes.

## Process

**You need to describe briefly the life cycle model or research method that you used. You do not need to write about all of the different process models that you are aware of. Focus on the process model or research method that you have used. It is possible that you needed to adapt an existing method to suit your project; clearly identify what you used and how you adapted it for your needs.**

**For the research-oriented projects, there needs to be a suitable process for the construction of the software elements that support your work.**

# Experiment Methods

**This section should discuss the overall hypothesis being tested and justify the approach selected in the context of the research area. Describe the experiment design that has been selected and how measurements and comparisons of results are to be made.**

**You should concentrate on the more important aspects of the method. Present an overview before going into detail. As well as describing the methods adopted, discuss other approaches that were considered. You might also discuss areas that you had to revise after some investigation.**

**You should also identify any support tools that you used. You should discuss your choice of implementation tools or simulation tools. For any code that you have written, you can talk about languages and related tools. For any simulation and analysis tools, identify the tools and how they are used on the project.**

**For the parts of your project that need some engineering (hardware, software, firmware, or a mixture) to support the experiments, include details in your report about your design and implementation. You should discuss with your supervisor whether it is better to include a different top-level section to describe any engineering work. In this template, Chapter 3 is suggested as a place for that discussion.**

### Custom program concept

A custom tool had to be created to be able to use Albacore basecall data with WEKA. Upon close inspection of the output data of the Albacore basecaller, multiple patterns can be found. Firstly, the process of basecalling is described in detail, each line of data belonging to 5 raw signal values, so 5-time units. Each line here has the same format:

I have 3762 files to use for this project. Each file contains one sequencing run. All together they are about 562 megabytes, in fast5 format. Each file contains on average 9000 basecall events. However, events where a new base is found, are only about 1500 per run. So about 7500 events have moves of zero, the meaning of this will be described later. This totals 3762\*1500=5643000 bases. How many instances this means in machine learning terms depends highly on the model used for representing this data, but even the best-case scenario is one fifth of the number of bases, which is 1128600, so just over a million. This means that if I am to include all files, initially I do not think that non-linear classification algorithms, anything with at least quadratic time complexity can be used. Depending on the model, I will evaluate multiple cases later on.

### Basecall events

This is how one basecall event is described in the Albacore basecaller:



The first float value is the mean value of the raw signals of the period

The second integer value shows the start time point of the line

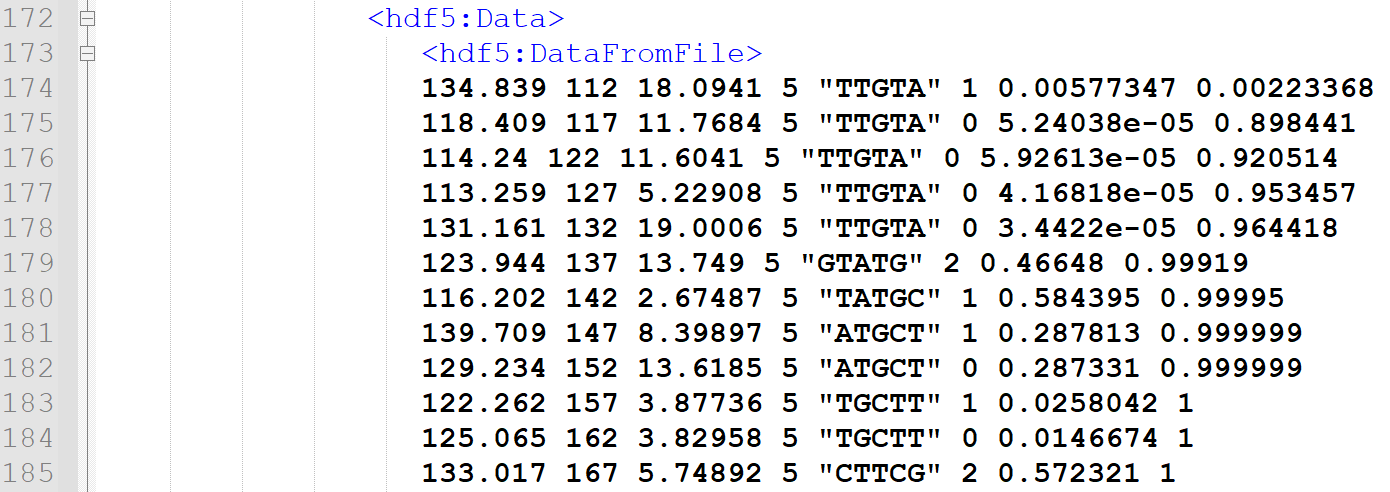
The third float is the standard deviation, so how much the mean differs from the average mean

The fourth integer is the length, this says how many raw signals correspond to the event, this is 5 in all cases here

The fifth is a string, called the model state. This shows the last 5 called bases.

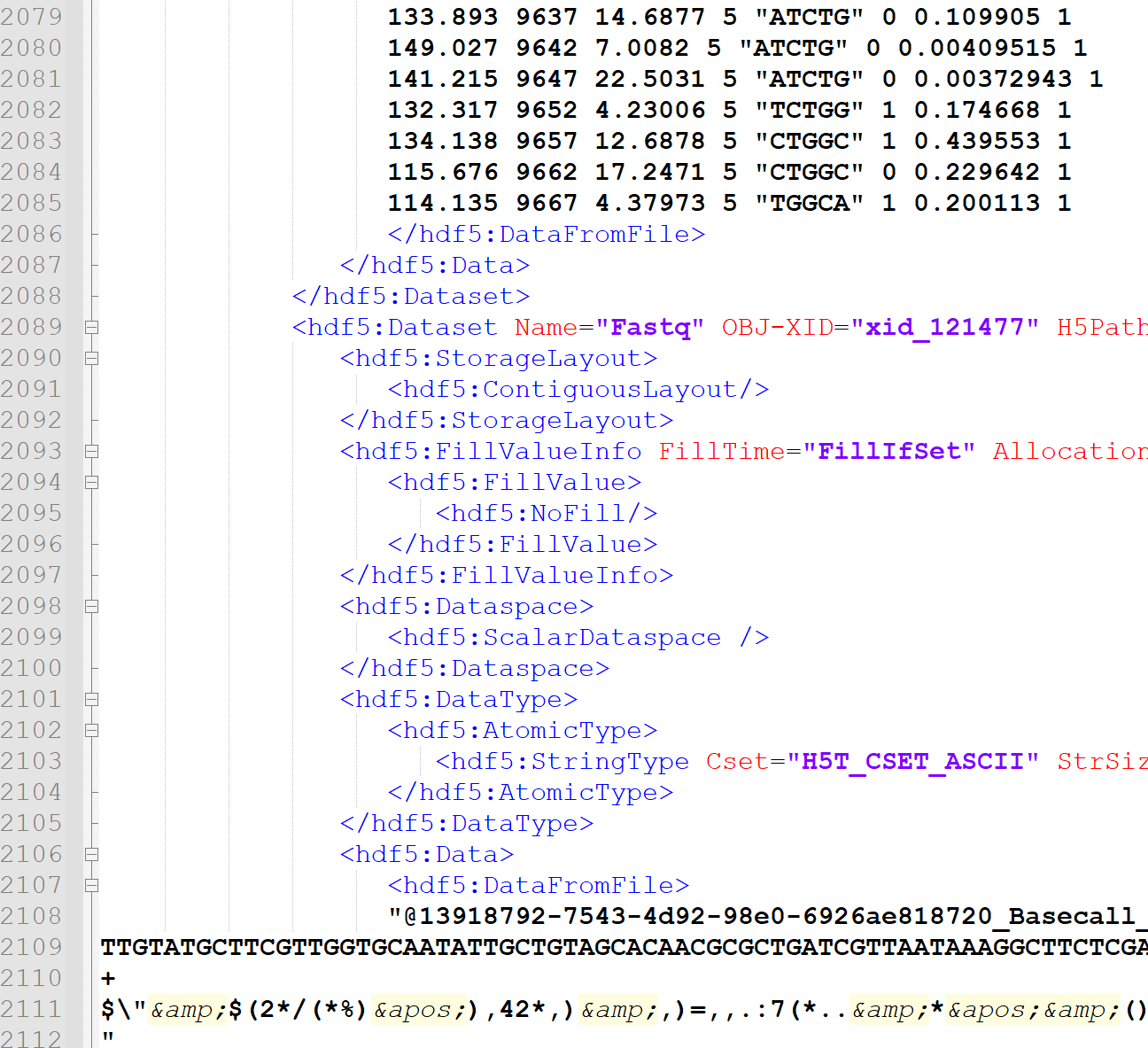
The sixth integer is the move, which can be 0 or 1 or 2. This tells how many bases were added to the model since the previous event.

These descriptions of the basecall events always start at line 174 of the file. This kind of information extraction is safe to use as the source code of this type of file was not meant for viewing, and there are no line breaks because of long lines at all, all line breaks in the file are dictated by its format.



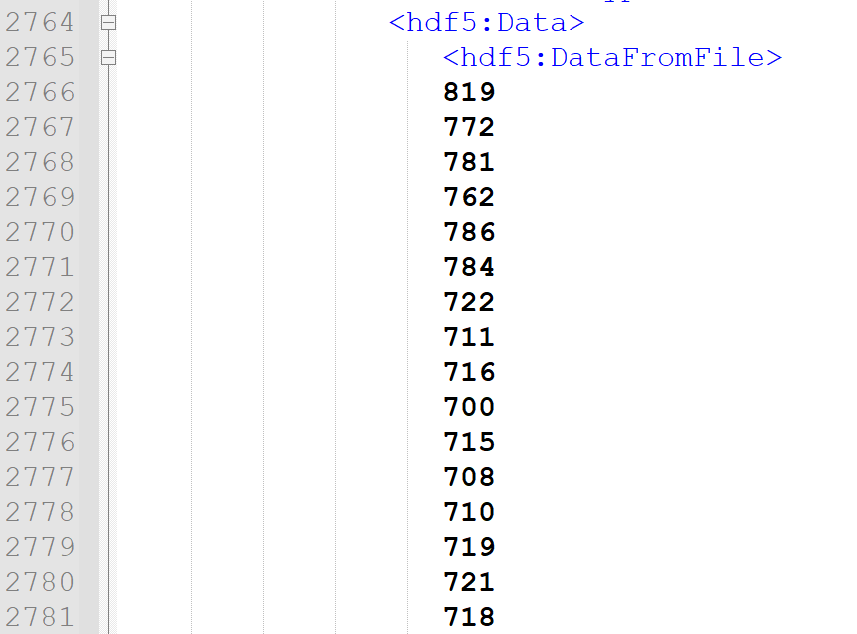
StartOfEvents 1

After the last line of the events, always exactly 24 lines below is the long string of the complete basecalled bases. This is about 2000 characters, but stored in one line. The events always start in a number, and when the list of events end, this is marked by a xml style end of element. Because different files contain a different number of events, the line number of the end of events differ from file to file, but the string of bases is always 24 lines below.



endOfEventsAndBasecalls 1

Then always exactly 657 lines below is where the raw signal data are stored, one value in one line.



These are all very useful information to be able to extract the information, allowing the transformation to WEKA’s data type, arff.

### Data Processing

Some adjustments are to be made to this data: This picture: StartOfEvents 1 shows that the events start describing from time point 112 onwards. This means that in the raw data, the first 111 value are to be deleted, as they do not belong to any of the bases.

Picture endOfEventsAndBasecalls 1 shows that the last event starting timepoint is 9667, so the last time looked at for basecalling is 9667+5=9672. This means that if these are any raw data starting from 9673 timepoint, they will have to be deleted as well, because again they do not belong to any of our bases. To find out if there are data after the events end, I did the following: In every file events start at line 174, and in this particular file, it ends at line 2085. There are 2085-174=1911 event lines. Each lines describes 5 raw signal data, so 5 time point. This means there are 1911\*5=9555 raw signal values that the basecalling describes. For raw signals, they start at line 2766 and finish at line 12441. So there are 12441-2766=9675 number of raw signal data. The first 113 needs to be deleted as described above, There are 9675-113=9562 values now. However basecalling described only 9555 as earlier mentioned, so the answer is yes, there is still data after the events stop. These have to be deleted as well as again they do not belong to any bases here. 9562-9555=7, so the last 7 raw signal is to be deleted.

Making a WEKA-style dataset out of this is possible by making a program that consists of three steps: first extracting the information, then processing it, and then outputting it in the required format. Read-in: Using Java’s Scanner class, a jump to line 174 is made. The first float is ignored, and the second integer is saved, as this marks the time where basecalling events start, as described above. This is important because all of the raw data before this point has to be deleted, because the basecalling events do not describe them, this deems them useless for machine learning training purposes. Then from each line the number of moves of the DNA bases are saved in a list. These are to be able to tell how many of the raw signal data is used to determinate each base, for as mentioned earlier, this is not constant. Example: if starting from 227 there is 0 move, then the next event gets checked, starting from 232 it is also 0 so moving on to the next, starting from 237 there is 1 move, an ‘A’ base was added in the model state, this means that for this ‘A’ base belongs the raw data from time 227 until time 236. It is also possible to have a move of 2 at once, as mentioned earlier, this means that from time X until time Y belongs to not one but two bases.

The end of the lines of events are marked by the fact that the line no longer begins with a float. When this happens, the Scanner is scroller down 23 lines, where the complete result of the basecalling, the bases can be found in one line. This is saved.

After this, 657 lines below is the beginning of the raw data, so the Scanner is scrolled there. Each line here contains one integer value. All of these are saved in a list. The end of these lines is marked by the line not beginning in an integer anymore.

As for processing the data, it was saved that the data starts at time 112, so the first 111 instances of raw data are deleted here. It is also saved that the last event starts at 9667, because events don’t always cover the raw signals until the end, and any raw signal data after the end of events have to be deleted, because the base calling events do not cover these. In this case as they cover 5 units of time, any data after 9671 is deleted. For the string of bases, the first 5 bases are deleted. This is because the data starts with the model state being full with 5 bases, these were formed from the raw data before the start of the events in the data, and as previously described, their raw data counterparts were also deleted.

I have experimented with representing the data in WEKA in multiple ways.

### WEKA models

The raw base data belonging to each instance of bases, are considered to be attributes. The bases are considered to be the classes.

#### First approach

My first program builds a model that considers 4 classes in total – one class for each base A,C,G,T. As WEKA requires all instances to have the same number of attributes, I used the method that saves the maximum number of attributes among all instances, and for every occurrence where the instance has less attributes, they are being filled with questions marks, representing missing data. There are classification algorithms that just ignore missing data, and there are some that do not do well with missing data. This is something that has to be taken into consideration. At this point what I think will be a bigger problem is that feature-selection wise, the emphasis will naturally shift towards the attributes that more instances have, the first X number of attributes. This will in my opinion cause the classifiers to do much more poorly on instances that are long reads, so the ones that have a high number of attributes.

I started experimenting with an initial amount of 300 files, this is about 270000 instances. This has an average accuracy of about 26%, which considering the fact that there are 4 classes, is purely the percent at which we could predict classes if we were to start blindly guessing. I tried the accuracy with just 1 file, and it is only on average half percent lower. This tells me that this model approach is not good enough, because even if this was to work, it would need too much data.

#### Second Approach

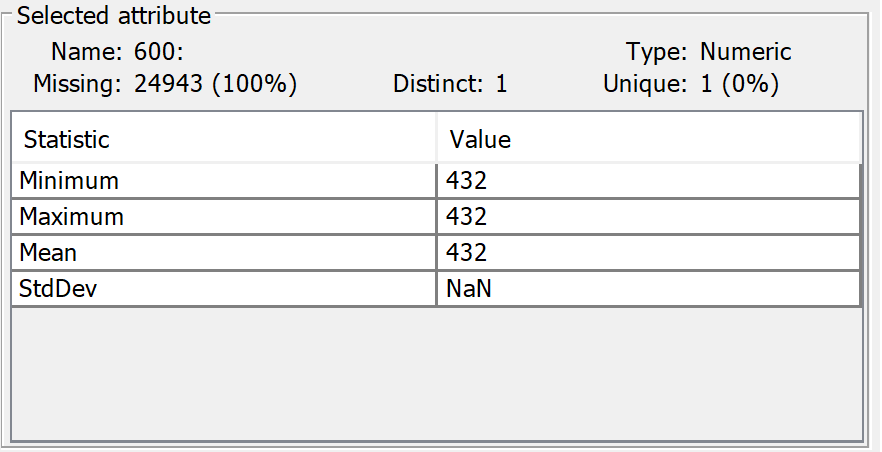
The second concept that I have tested is to have classes made up of 5 bases. So any permutation of length 5 of the bases A,C,G,T. This results in a number of classes equal to 4 on the power 5, which is equal to 2 on the power of 10, which is 1024. This is a lot of classes, and to be able to see this working, I predicted that an enormous amount of training data will be required. The concept of raw data as attributes is the same, however as now they of length equal to 5 bases instead of just one, the average number of attributes should be about 5 times higher per instance, however the number of instances for the same amount of data should be lineally reduced as well. In theory, this should result in reduced training time compared to classes of one base, as typically a higher number of instances have a much higher effect on training time then the number of attributes each instance has.

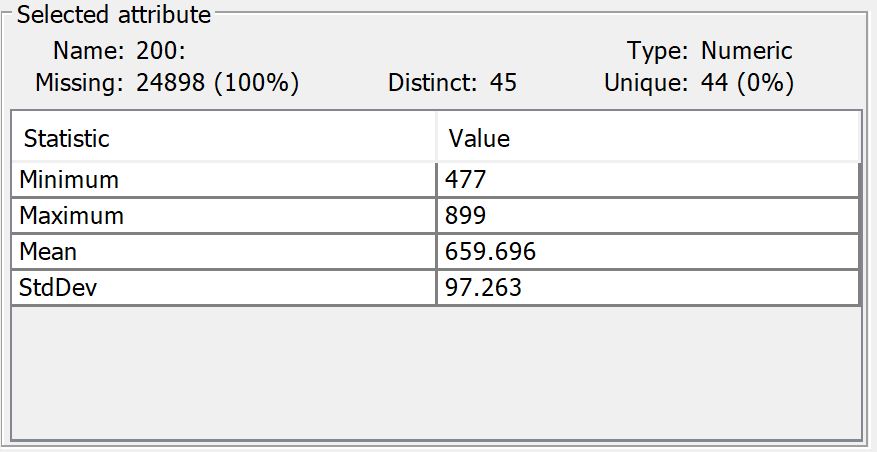
#### Third approach

Models in the research papers I read were mostly using k-mers. This in a way is a hybrid approach of my previous two approaches, so this is what I went with for my third concept. Based on k-mers I have tested representing the data in 5-mers. This is a sliding-window concept. For instance, the seven bases ACGTACG with moves of one would be represented as three 5-mers: ACGTA, CGTAC, GTACG. If there is a move of two, this would mean only two 5-mers: ACGTA, GTACG. If moves of one and two are mixed, the length of the bases of the previous example have to be extended by one, this looks like the following example: ACGTACGT – ACGTA – first two moves: GTACG – then one: TACGT. It is also possible to have the move of first, and the move of two second – ACGTA – CGTAC – TACGT.

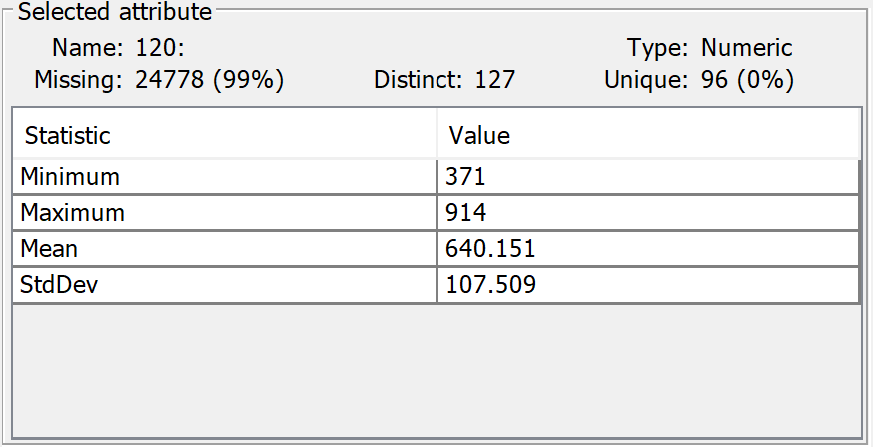
This method is what most researches I have looked at used, however they the length of k-mers varies, DeepNano uses 6-mers for instance. This method is a refinement of my previous attempt. However, this method will result in a much larger number of instances, something that will significantly increase training time. The reason for the growth of the number of instances is that the number is equal to the number of moves that are not 0(So moves that are 1 or 2). This takes the number towards the first model, while the previous instance had a number of instances equal to the number of events divided by five.

Taking 200 files into the program, I loaded the output file into WEKA. First, I had to see under how many time periods a 5-mer is created. The program run allowed reads to be as long as 600 time periods, however it is clearly visible that these are some rare outliers. This file contains 24943 instances, and but when looking at how many instances have at most 200 this number is 24898.





This means that out of about 25000 instances only about 100 has more than 200 attributes. Looking at 100, about 500 has more than 100. Naturally I would like to work without data loss, but the nature of WEKA is that every instance has to have the same number of attributes. This means that I am to have 1 instance with 600 attributes, and 24999 instances with just 100, for every one of said 24999 I would have to mark the remaining 500 attributes as missing by filling it with a question mark. This skyrockets the size of the data. To significantly reduce my training time, I had to find a sweet spot, loosing as little data as possible while keeping training time reasonable. I have made this spot 120 for testing.



Filtering out any instances created over a time period of 120, and then running this in WEKA using a basic neural network with default settings, using a 10-layer cross-validation resulted in 1.554% accuracy. Instances from 200 files are nowhere near enough training data, however considering that there are 1024 classes, mathematically speaking a generic guessing result would be 1/1024≈0.0009766%. A result slightly over one and a half percent, with this amount of training data used, seem to suggest that the model works, but to what extent, it has to be further evaluated.

Because there is no tool currently to my knowledge that allows for importing fast5 files to WEKA, I decided to take this project towards creating a tool that acts as a bridge for this gap. A GUI tool is always good, but it is much simpler for now to create a tool that can be executed through the command line. Because tools that allow the extraction of information from fast5 files are widely available on Linux, but are not common on Windows, my tool currently is only for the Linux command line. It allows the user to train basic WEKA algorithms for basecalling and evaluate their performance. This tool is not mainly aimed towards these basic classification algorithms, and if I were to continue this project outside of the time limit, the next step would be to integrate Eclipse’s DeepLearning4J into the algorithms my tool works with. For now, it can only use the base WEKA classification algorithms. My program makes use of the Linux command line tool called h5dump, my java code, and the java command line version of WEKA, all controlled with a bash script.

The CDImpute class follows my final model described earlier.

First it takes the filepath of the file that is to be read in.

Then it creates the output file, and writes the necessary text for the attribute relation file format. It also sets all possible classes, so all possible permutations of length 5 using the 4 DNA bases, (repetitiveness is allowed) which is 4 on the power of 5, which is 1024.

Next is a for loop to print a report on the process, so it prints how much of the work is done in percentages.

Then I create two integers, one that says start, and one that says end. These will store start and end time point of the events in the current file. ArrayList steps will contain the the complete basecall string. ArrayList signals will contain all of the signals from the file. ArrayList save is an array that will be temporary placeholder that is required for the functioning of the program. More on this later.

A scanner is created to read in text from the file passed in the beginning. The scanner will jump over the first 173 lines to end up at line 174 where the basecalling event descriptions begin.

I am interested in saving the number of steps (0 or 1 or 2) from each event. Also, from the first event the timepoint, as this is the timepoint where basecalling events start. And also from the last event line the timepoint, as this is where description of basecalling events end.

To go ahead in the code, it is now going to skip over the first float, and then save the timepoint of start to the integer I created earlier for this purpose. It is also going to save the number of steps the model has moved, this is the 6th data. Then it saves just the number of moves from every line, until it realises that the line it is currently on is not a basecall event anymore. This can be realised because the lines would no longer start in a float, rather in text.

When it reaches this point, it goes back to the last line to save the end time point integer as well.

Then it skips down 23 lines, where the entire basecalled bases are found in one string. It saves it into the string basecalls.

Then it skips 657 lines below, where the first raw signal data start. These are just one integer in one line. So it keeps adding each integer into the signals array list, until it reaches a point where the line does not start with an integer anymore, thus there is no more raw signal data.

After this comes processing the data.

First step here is to recognize that the first line in the events already has 5 bases and a move filled out. We do not know anything about how these were created so they have to be deleted. So the first 5 bases and the first move is deleted.

As mentioned earlier, basecall events usually do not cover all of the raw data. Some data in the beginning and in the end are not covered, so these raw signals that are not covered by events have to be found, and deleted.

This is done by having the length of all raw signal data as the integer sinlen, start is where the events start, and end where the events end. So from the beginning of the raw signal data, I delete start-1 elements, as these are the ones not covered. Then going backwards from the last item, I have to delete sinlen-(end-start) number of items, because I this is how many more items are at the end. Sinlen again is the current length of the raw data array, and end-start is how many elements are covered by the events.

Then comes a while loop which is set to go on forever, it has a break condition that comes active when all the data has been used.

How the contents of this huge loop works, is that each integer in the moves array gets checked one by one. There are different steps to take for each of their 3 possible values. It was earlier described that one basecall event corresponds to 5 raw signal data. If the first move is 0, then the first 5 raw signal data gets stored into an array. Then integer x gets incremented by one, this is the integer that tells the program which element to check in the steps array. So it moves on to the next step value. At the same time, c gets incremented by 5, as c is the value that tells us which 5 raw signal data to handle. If the next move is zero again, this is what will happen again, save the 5 raw signal data in the array on top of the ones that were previously saved, and increment move index by one, and raw signal index by 5.

If the move is 1, then the current raw signal data is added to the array as before, the difference is that the first base gets written into our current model. Then the correct increments are made for indexes, +1 for moves, +5 for signal, +1 for bases.

If the move is 2, then the only difference from those that happened in move 1 is that 2 bases get written into our model instead of one. And then as a result the bases index will be incremented by 2 instead of 1.

These moves go on until our initial model gets filled with 5 bases.

The moment our model has 5 bases, the model is written out in the output file along with all the raw data that has been save up so far. Then the program empties its save array. Now, in the beginning the model has zero elements, it gets loaded up to 5, and from there on out it always consists of 5 elements. If by any chance that were to occur that the model is in state 4, and a move of 2 comes, then the program treats this as two separate 1 moves, without deleting the save array after the first move of 1, to make sure that there is no data loss. So what happens is the first base of the 2 move gets filled into the model, and the five bases are written out along with all the saved signals. Then in a sliding windows concept, the first base in our model gets deleted, and the other base from the move of 2 gets added to the end, and this model is written to the file along with all saved signals. Then the signal array is emptied, and the move is incremented by 1, the signals by 5, and the bases by 2.

Now that the initial filling up of the model is done and it will always stay at length 5, from here on out, the concept will be the same:

If the move is 0, save the 5 signals to array, increment move by one and signals by 5.

If the move is 1, delete first base in model, insert new base to last place in model, add 5 signals to signal array, write signal array along with model to file, delete signal array, increment signal index by 5, move index by 1, basecall index by 1.

If the move is 2, delete first and second bases in model, add 2 new bases to last 2 places. Save 5 signals to array, then write out model and signal array to file, then clear signal array, increment signal index by 5, move index by 1, basecall index by 2.

# Software Design, Implementation and Testing

This could be one chapter or a few chapters. It should define and discuss the software that is developed to support the research that is being conducted. For example, if your research involves running experiments, what software are you creating to support that work? What functionality is required? What design will be used? What implementation issues are there and what testing is used?

Even though a research project is investigating specific research questions, it is still necessary for you to discuss the software that you develop. Research has a habit of generating bits of software that can exist for several years and need future modification. Therefore, you need to be able to discuss the technical issues as well as the research approach.

## Design

You should concentrate on the more important aspects of the design. It is essential that an overview is presented before going into detail. As well as describing the design adopted it must also explain what other designs were considered and why they were rejected.

The design should describe what you expected to do and might also explain areas that you had to revise after some investigation.

Typically, for an object-oriented design, the discussion will focus on the choice of objects and classes and the allocation of methods to classes. The use made of reusable components should be described and their source referenced. Particularly important decisions concerning data structures usually affect the architecture of a system and so should be described here.

How much material you include on detailed design and implementation will depend very much on the nature of the project. It should not be padded out. Think about the significant aspects of your system. For example, describe the design of the user interface if it is a critical aspect of your system, or provide detail about methods and data structures that are not trivial. Do not spend time on long lists of trivial items and repetitive descriptions. If in doubt about what is appropriate, speak to your supervisor.

You should also identify any support tools that you used. You should discuss your choice of implementation tools - programming language, compilers, database management system, program development environment, etc.

Some example sub-sections may be as follows, but the specific sections are for you to define.

### Overall Architecture

### Some detailed design

#### Even more detail

### User Interface

### Other relevant sections

## Implementation

This section should discuss issues you encountered as you tried to implement your experiments. What were the results of running the experiments? What conclusions can you draw from these results?

During the work, you might have found that elements of your experiments were unnecessary or overly complex; perhaps third-party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

If you had multiple experiments to run, it may be sensible to discuss each experiment in separate sections.

## Testing

Detailed descriptions of every test case are definitely not what is required in this section; the place for detailed lists of tests cases is in an appendix. In this section, it is more important to show that you adopted a sensible strategy that was, in principle, capable of testing the system adequately even if you did not have the time to test the system fully.

Provide information in the body of your report and the appendix to explain the testing that has been performed. How does this testing address the requirements and design for the project?

How comprehensive is the testing within the constraints of the project? Are you testing the normal working behaviour? Are you testing the exceptional behaviour, e.g. error conditions? Are you testing security issues if they are relevant for your project?

Have you tested your system on “real users”? For example, if your system is supposed to solve a problem for a business, then it would be appropriate to present your approach to involve the users in the testing process and to record the results that you obtained. Depending on the level of detail, it is likely that you would put any detailed results in an appendix.

Whilst testing with “real users” can be useful, don't see it as a way to shortcut detailed testing of your own. Think about issues discussed in the lectures about until testing, integration testing, etc. User testing without sensible testing of your own is not a useful activity.

The following sections indicate some areas you might include. Other sections may be more appropriate to your project.

### Overall Approach to Testing

### Automated Testing

#### Unit Tests

#### User Interface Testing

#### Stress Testing

#### Other Types of Testing

### Integration Testing

### User Testing

# Results and Conclusions

This section should discuss issues you encountered as you tried to implement your experiments. What were the results of running the experiments? What conclusions can you draw from these results? What graphs or other information have you assessed regarding your experiments? Discuss those.

During the work, you might have found that elements of your experiments were unnecessary or overly complex; perhaps third-party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

If you had multiple experiments to run, it may be sensible to discuss each experiment in separate sections.

# Critical Evaluation

Examiners expect to find a section addressing questions such as:

* Were the requirements correctly identified?
* Were the design decisions correct?
* Could a more suitable set of tools have been chosen?
* How well did the software meet the needs of those who were expecting to use it?
* How well were any other project aims achieved?
* If you were starting again, what would you do differently?

Other questions can be addressed as appropriate for a project.

The questions are an indication of issues you should consider. They are not intended as a specification of a list of sections.

The evaluation is regarded as an important part of the project report; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things in the work and aspects of the work that could be improved. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

In the latter stages of the module, we will discuss the evaluation. That will probably be around week 9, although that differs each year.

# Annotated Bibliography

This final section should list all relevant resources that you have consulted in researching your project. Each reference should also include a brief annotation.

1. Sylvia Duckworth. A picture of a kitten at Hellifield Peel. <http://www.geograph.org.uk/photo/640959>, 2007. Copyright Sylvia Duckworth and licensed for reuse under a Creative Commons Attribution-Share Alike 2.0 Generic Licence. Accessed August 2011.

This is my annotation. I should add in a description here.

1. Mark Neal, Jan Feyereisl, Rosario Rascunà, and Xiaolei Wang. Don’t touch me, I’m fine: Robot autonomy using an artificial innate immune system. In *Proceedings of the 5th International Conference on Artificial Immune Systems*, pages 349–361. Springer, 2006.

This paper…

1. W.H. Press et al. *Numerical recipes in C*. Cambridge University Press Cambridge, 1992.

This is my annotation. I can add in comments that are in **bold** and *italics*and then further content.

1. Various. Fail blog. <http://www.failblog.org/>, August 2011. Accessed August 2011.  
     
   This is my annotation. I should add in a description here.
2. Apache Software Foundation (2014) “*Apache POI - the Java API for Microsoft Documents*” (Online) Available at: <http://poi.apache.org> Accessed: 14th March 2014.

This is my annotation. I should add in a description here.

1. Apache Software Foundation (2004) “Apache License, Version 2.0” (Online) Available at: <http://www.apache.org/licenses/LICENSE-2.0> Accessed: 14th March 2014.

This is my annotation. I should add in a description here.

1. Neil Taylor, “MMP\_S08 Project Report and Technical Work”, 2019 (Online) Available at: <http://blackboard.aber.ac.uk/> Accessed 19th February 2019.

A document that outlines information about the marking guide for the Project Report and Technical Work. This is published in the Resources folder on Blackboard.

# Appendices

The appendices are for additional content that is useful to support the discussion in the report. It is material that is not necessarily needed in the body of the report, but its inclusion in the appendices makes it easy to access.

For example, if you have developed a Design Specification document as part of a plan-driven approach for the project, then it would be appropriate to include that document as an appendix. In the body of your report you would highlight the most interesting aspects of the design, referring your reader to the full specification for further detail.

If you have taken an agile approach to developing the project, then you may be less likely to have developed a full requirements specification. Perhaps you use stories to keep track of the functionality and the ’future conversations’. It might not be relevant to include all of those in the body of your report. Instead, you might include those in an appendix.

There is a balance to be struck between what is relevant to include in the body of your report and whether additional supporting evidence is appropriate in the appendices. Speak to your supervisor or the module coordinator if you have questions about this.

* 1. Third-Party Code and Libraries

If you have made use of any third-party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. If third-party code or libraries are used, your work will build on that to produce notable new work. The key requirement is that we understand what your original work is and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

The following is an example of what you might say.

**Apache POI library** – The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the client’s existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation [5]. The library is released using the Apache License [6]. This library was used without modification.

Include as many declarations as appropriate for your work. The specific wording is less important than the fact that you are declaring the relevant work.

* 1. Ethics Submission

This appendix includes a copy of the ethics submission for the project. After you have completed your Ethics submission, you will receive a PDF with a summary of the comments. That document should be embedded in this report, either as images, an embedded PDF or as copied text. The content should also include the Ethics Application Number that you receive.

* 1. Code Samples

This is an example appendix. Include as many appendices as you need. The appendices do not count towards the overall word count for the report.

For some projects, it might be relevant to include some code extracts in an appendix. You are not expected to put all of your code here - the correct place for all of your code is in the technical submission that is made in addition to the Project Report. However, if there are some notable aspects of the code that you discuss, including that in an appendix might be useful to make it easier for your readers to access.

As a general guide, if you are discussing short extracts of code then you are advised to include such code in the body of the report. If there is a longer extract that is relevant, then you might include it as shown in the following section.

Only include code in the appendix if that code is discussed and referred to in the body of the report.

Random Number Generator

The Bayes Durham Shuffle ensures that the pseudo random numbers used in the simulation are further shuffled, ensuring minimal correlation between subsequent random outputs.

// Some example code here…