Project Proposal for the Automated Electron Identification with the LHC ALICE Transition Radiation Detector

JEDDA BOYLE, University of Cape Town
TAPIWA CHADENGA, University of Cape Town
RYAN WONG, University of Cape Town

1. INTRODUCTION

In this project proposal we specify the problems, solutions and methods of a new particle identification framework indented for use at ALICE.

2. PROJECT DESCRIPTION

Not enough is known about the very early universe. Roughly a millionth of a second into the universe's existence the mixture of quarks and gluons, known as quark-gluon plasma, began to cool and form hadron particles such as protons and neutrons - the fundamental building blocks of matter [Trafton. 2010]. A Large Ion Collider Experiment (ALICE) attempts to study this period of the universes existence by studying heavy-ion - the nuclei of heavy particle - collisions provided by CERN's Large Hadron Collider (LHC) to form quark gluon plasma. The Transition Radiation Detector (TRD) is a subdetector of the ALICE detector which measures and identifies particles, emitted from heavy ion collisions, as they pass through it [Wilk 2010]. However, the signature of different particle species is not easily discernible and this means machine learning techniques are required to classify the data. Artificial neural networks (ANN) are the standard method for particle identification at ALICE [Wilk 2010].

Many of the experiments at ALICE have a very high noise to signal ratio. This means that for some experiments more collisions are required than is currently possible. In order to achieve this ALICE is scheduled for an upgrade in 2017 which will improve the frequency of collisions and change the data format. This means that more particles need to be detected using a new data format. This means the particle identification software needs an upgrade.

To make the problem more difficult online classification is desired. This means the machine learning implementation must be computationally efficient and, due to hardware limitations, each particle traversal of the TRD must be compressed into six 8 bit online particle IDs (online PID) which are used as input for the machine learning algorithm. Additionally, accuracy cannot be lost because this would reduce the benefits of the upgrade.

In this project we are going to investigate possible solutions to the problems mentioned above. We are going to look at ways of improving the speed of particle identification using only 8-bit online PIDs without loss of accuracy. For the sake of this project we are interested in the classification of electrons and pions. There are two key problem areas we will consider. Firstly, the construction of a descriptive online PID and secondly, the development of a computationally efficient and accurate machine learning algorithm.

3. PROBLEM STATEMENT

The aim of this project is to investigate possible avenues for an improved particle identification system, that provides online and accurate classification, at ALICE. In order to do this we are going to implement an algorithm which constructs the online PID and a machine learning based particle identification algorithm. During the construction of this system we are going to answer the following research question:

1 *How well does an optimised AdaBoost implementation, using an 8-bit online PID, compare in terms of functionality, speed and accuracy of particle identification at ALICE compared to the current artificial neural network implementation.*

This is a very complex research questions and in order to answer it effectively we have broken it up into 5 smaller questions. Each Group member is allocated a set of these of smaller questions.

The construction of the online PID is crucial because it is 8-bits and is the only input for the particle identification algorithm. The online PID is a compressed value representing the TRD data. The TRD is constructed in segments that are divided up into voxels of gas which record the charge deposited from a particle passing through it. A single traversal of the TRD usually intersects 6 segments of the TRD. The recorded information from each of these segments needs to be converted into a single 8 bit online PID. A particle traversal of a segment produces a 30×4 matrix of 10-bit integers. Each entry in the matrix relates to the deposited charge of 1 voxel. This matrix needs to be converted into a single 8 bit online PID which is descriptive of the entire matrix. A possible solution, called the summation method, simply adds all the integers and then truncates the total to an 8-bit integer. The size of the online PID could be increased to 18 or 40 bits if we can show that this results in accuracy large improvements. Given the above problems we are going to answer the following research questions which will be answered by Ryan:

2 *Does an evolutionary algorithm produce a online PID with a larger difference between the probability distribution of electron and pion than the summation method?*

3 *Does increasing the size of the online PID to 18 or 40 bits increase the difference between the probability distribution of electrons and pions online PID?*

The next major consideration is the machine learning section. The machine learning algorithm must be accurate, even with an 8-bit online PID. It must be be computationally efficient because online classification is needed. Lastly, it must give confidence ratings for its classifications.

Given these requirements an optimised version of AdaBoost is our proposed solution. To increase accuracy and speed we are going to implement evolutionary pruning. Evolutionary pruning results in better weight distributions for the weak learners and can reduce the number of weak learnersJang and Kim [2008]. To provide a measure of the algorithms confidence of a classification we are going to implement Robert Schapires confidence rated-predictions algorithm [Schapire 1999]. These requirement result in the following research questions which will be answered by Jed:

4 *Does evolutionary pruning improve the classification speed and accuracy of AdaBoost with respect to particle identification at ALICE?*

5 *Does confidence rated-predictions provide confidence ratings which are* 90% *statistically similar to the distribution2 of the online PID used to generate the training data?*

The selection of weak learners impacts the accuracy and speed of AdaBoost classification. A weak learner must balance accuracy and efficiency. Generally, boosting is more efficient when given weak learners which are computationally effective rather than accurate. This is because many are combined. This is why decision trees are a common choice. However, given the complexity of the classification task at ALICE and previous success of ANNs other types could perform better. Therefore we need to answer the following research question which will be answered by Tapiwa:

6 *Is the ratio between performance and speed of a decision tree less than that of an ANN or a support vector machine?*

4. PROCEDURES AND METHODS

The project has been broken up into three parts and each one answers a subset of the research questions mentioned above. The evaluation of each part has been constructed in such a way that it can tested and developed separately.

The evaluation of the final solution will asses speed and accuracy. Accuracy is measured by pion and electron efficiency. Pion efficiency is number of pions incorrectly classified as an electrons. Electron efficiency is the number of correctly classified electron. For speed we are interested in the amount of CPU time taken by the slowest particle in a batch no larger than $250,000$. This is because the hardware at ALICE is unable to process more than $250,000$ particles per batch and it must wait for the slowest particle to be processed before starting on the next batch.

4.1. online PID Generating Algorithm

To answer the first sub-research question we are going to use an evolutionary algorithm to construct the online PID generation algorithm. The evolutionary algorithm's population is a set of online PID generators which take the deposited charge information from the TRD layer and create a online PID for that particle. It does this for all particles in the training data and uses the results to plot a probability density function of the possible online PID values of electrons and pions. This is used to measure the fitness of the online PID generator. The less the distribution for pions and electrons have in common the greater the fitness. In the ideal case the fitness would be measured by how effective the particle algorithm is when it uses the online PID generator. However, to keep the project separate measuring the difference in probability distribution will give us a good idea of how descriptive the online PID generator is.

To explain this in more detail, consider that for every online PID value there is a probability that the particle is an electron or a pion. If for every online PID value we plot the associated probability of it being a pion and an electron then we get a probability density function. The more these two plots differ the better. To measure the difference between the two plots, sum the absolute value of the differences between the two plots for every online PID. The greater this number the better.

This measure of fitness is important because it will be used as the fitness function of the evolutionary algorithm and also to evaluate the solution. A suggested technique to produce the online PID would be to sum the data from the TRD and truncate it into 8-bits. We are going to compare the probability density function given by the summation online PID generator with the evolutionary algorithms best solution and see which one is more different - and therefore better.

Next we need to consider if increasing the size of the online PID improves classification accuracy. To answer this question we are going to implement the same algorithm as above and investigate what impact increasing the size of the online PID has on the difference between the probability density function for the online PID of electrons and pions.

4.2. AdaBoost Optimisations

This section of the project aims to solve the inefficient weighting of weak learners, lack of pruning (speed) and lack of classification confidence provided by the original AdaBoost implementation.

The first stage of the process requires obtaining training data. Training data is going to be simulated using a probability distribution of possible online PIDs. We pick how many pions and electrons we want in our training set and then use the probability distribution function to sample a online PID for each one. The exact shape of the probability density is not yet known. We will try different probability density functions and investigate what performs best. The results of this could be used to help the online PID generation algorithm in the previous section find optimal probability distributions.

The problems this section aim to solve depend on a successful implementation of AdaBoost. AdaBoost is going to be implemented using the original specification given by Schapire [1999]. In this section of the project AdaBoost is going to use decision trees as weak learners. These decision trees dont need to be optimised. This part of the project is concerned with the impacts of possible optimisations and it only requires that the same weak learners be used throughout.

To solve the pruning and weighting problems we are going to implement evolutionary pruning as suggested by Jang and Kim [2008]. It is based upon the assumption that there is redundancy in weak learners resulting from dependencies [Jang and Kim 2008]. The algorithm consists of two parts. Initially an evolutionary approach is used for reweighting weak learners and then weak learners with a weighting below a certain threshold are pruned.

To test the impact of the evolutionary reweighting we are going to investigate its impact on accuracy. This stage of optimisation cannot improve speed. Therefore, we are interested in pion and electron efficiency. If we divide electron efficient by the pion efficiency we will get a single statistics which will measure if the new weighting has improved accuracy.

The next step is to prune weak learners. For this stage we are interested in the number of weak learners which can be pruned because this will relate to an improved classification speed. To test the impact of evolutionary pruning has on performance we will fix an acceptable level of accuracy and then monitor how many weak learners can be pruned without impacting the accuracy. It would also be interesting to see what the payoff is between speed and accuracy.

Finally, we are going to implement Robert Schapires confidence rated-prediction algorithm to see if it provides a good measurement of confidence. To test this we measure the confidence of a prediction for each online PID and then compare this to the probability distribution which generated the online PID. If they are similar then the confidence rating is probably a good measure. We will use statistical techniques measure similarities between the two.

4.3. Weak Learner Selection

Machine learning algorithms will be created using C++. This is the core language that is utilized in ALiRoot, the ALICE simulation and development environment.

This part of the research answers the question of which machine learning algorithm performs best as a weak learner for AdaBoost. To keep this section of the project separate from the boosting section we are going to test the ratio between speed and performance. This will be used as a proxy measure of how good the algorithm would be when boosted. If time allows we will test the boosted versions of the algorithms. The algorithms we will consider are ANNs, Decision Trees and Support Vector Machines. Implementations will be parameterized so to allow for easy alteration and experimentation.

Decision trees are designed to assign categories based on large sets of examples. A decision tree consists of interconnected sequences of binary splits. In particle identification, data is split into a valid signal or background noise. In particle identification, decision trees attempt to identify which variables in the online PIDs best split data into two categories. What these categories are exactly is determined during learning. We will look at the most widely used algorithm, namely, the ID3. Decision trees are effective but unstable. A small change in the training data can produce a large change in the tree. This is what boosting essentially eliminates. When decision trees are combined sequentially, each tree tries to reduce the bias of the combined group.

We will experiment with both feedforward (FNN) and recurrent (RNN) topologies of artificial neural networks. FNNs are widely used in many diverse applications. Their strength is that performance can be modified significantly by changing the activation function easily, allowing for greater potential for experimentation. In addition to basic FNN and RNN topologies, we will consider a few variations. With regards to RNNs, we will consider the simple recurrent network. It is a three layer ANN that has a back loop from the hidden layer to the input layer which ultimately enables it to store information. We will also look at Long Short Term Memory, which differs from other ANNs in that it is configured to remember important values for unspecified time periods. Lastly we also hope to consider Bi Directional ANNs, which specialise in remembering the past in order to predict the future.

Support vector machines (SVM) are classifiers based on the concept of decision planes. SVMs perform a classification task by constructing hyperplanes in a multidimensional space that separate cases of different class labels. SVMs are stable learners, therefore it is unconventional to use them as base learners in boosting. SVM also have a high computational cost. To overcome these issues we will build lighter local models, not expensive global ones. When boosted appropriately, SVMs can perform better than decision trees.

To test the accuracy of these algorithms we will use the ratio of electron and pion efficiency. To measure speed we will use the amount of time the slowest classification took on the CPU in a batch no bigger than $250,000$. To help make comparisons we'll use a ratio of speed and accuracy statistics to compare the three implementations. Decision trees will be implemented first since they are the standard. ANNs and SVMs will be developed later and compared to decision trees.

5. PROJECT PLAN

5.1. Risks

Risk Condition	**Prob (10)**	**Impact (10)**	**Risk Factor (100)**	**Consequence**	**Mitigation**	**Monitoring**	**Management**
Project member drops out	2	8	16	Increases in workload for other members	Ensure everyone in the group is focused and communicates any potential problems.	Communicate with project members.	Reallocation of workload amongst remaining project members or change the scope of the project.
Scope of project is larger than expected	7	6	42	Project becomes infeasible for time given	Keep referring to objectives, plan and goals and communicate with supervisors about our aim of our project	Have regular meetings with supervisors on what we are aiming to do and still plan to do within the project	Reduce the scope of the project.
Project deadlines not met	2	8	16	The project gets delayed.	Stick to timeline and have regular meetings to ensure everyone is will finish by the desired deadlines.	Arrange regular meetings to ensure project members are making progress with their part of the project.	Reduce scope of the project or reallocate workload amongst members if a member is struggling.
Misunderstanding of project requirements	4	7	28	Need to rethink the design of our proposed system.	Stick to scope of project and verify any confusion with requirements to the supervisors via email or during meetings.	Give feedback on what we plan to do and what we have done to project members and supervisors.	If project members are confused, arrange meetings amongst project members first and then with supervisors to clarify the project requirements.

Lack of communication	3	5	15	Lack of communication between supervisors and team leads to misunderstanding and delays of project.	Arrange regular team meetings as well as having regular meetings with the supervisors.	Check if we have meetings regularly as well as check with each team member on their progress with the project.	Arrange a meeting immediately between team members and then arrange a meeting with the supervisors.
Lack of training / skills	2	8	16	Unable to implement our algorithm / design	Arrange a time to use and test the current systems with supervisors.	If we feel confused about certain aspects of the project we should communicate with supervisor.	Arrange meetings to learn relevant skills
Design is infeasible for current system	5	5	25	Need to start from scratch with the design	Check what system can do before implementation	Have checks involved during implementation to test our system with the current system.	Implement design but give ideas to make the design feasible within current system.

5.2. Ethical, Professional and Legal Issues

The ROOT and AliROOT system is available under GNU Lesser General Public License. Access to the online framework AliEN would require CERN certification. This is not necessary for our project. Our project only uses the offline framework which is subject to copyright and the the following criteria .

(1) The use of the AliRoot Offline system is for non-commercial purposes.
(2) All projects contain the AliRoot Offline systems copyright notice.

Our product is not for commercial purposes. We plan to licence our work under the same GNU Lesser General Public License. Therefore, if we are careful to put the offline systems copyright notice on our project there are no other legal issues with using AliRoot.

5.3. Related Work

Currently there has been no implementation of using a online PID for the input of the particle identification algorithms. The online PID is created by using particles deposited charge information which is obtained from the TRD. The data structure is explained by Wilk [2010]. Beyer and Schwefel [2002] highlights suitable approaches to solving problems similar to finding a online PID Generator.

The original description of AdaBoost was written by Robert Schapire and can be found at Schapire [1999]. The literature relating to evolutionary pruning and confidence based ratings can be found at Jang and Kim [2008] and Schapire and Singer [1999] respectively.

Not much has been published on the use of various machine learning techniques on the ALICE detector, but several notable papers have been released with regards to the use of soft computing at the Large Hadron Collider in general.Dong and Han [2005] used rule ensembles to successfully identify supersymmetric particles at the LHCs ATLAS detector. Ackermann et al. [2007] showed that genetic algorithms performed better in off line particle identification by the Ring imaging Cherenkov (RICH) also at CERN. Kuusela et al. [2010] demonstrated that gene expression programming, neural networks and support vector machines proved very effective in diffractions detection within large samples of simulated proton-proton collisions. As concluded at a CERNs Proceedings of Science Session 2 in 2007, there needs to be an ongoing effort towards developing algorithms, statistical methods and tools for data analysis in particle physics.

5.4. Anticipated Outcomes

We imagine that our proposed solutions will work and help solve some of the problems of particle identification at ALICE. Hopefully this will impact implementation choices in future version of AliRoot. However, we do not expect that our solution will be able to compete with the current ANN implementation in terms of speed or accuracy. This is because we don't have the same time or resources. If our system does compare it would fantastic and a sign of success.

However, it is more likely that we will show that our solutions work and should be considered in later version of AliRoot. Our project will be considered a success if we can prove that our proposed solutions could solve some of the particle identification problems at ALICE. This is effectively the same as answering all the research questions.

5.5. Timeline

See the Gantt chart appended to the end of this document.

5.6. Resources Required

AliRoot has an API and libraries necessary for critical events such as producing training data. We will also have access to some of the algorithms currently being utilized. This will be useful for comparison purposes.

5.7. Deliverables and Milestones

Task:	**Due Date:**
• Project Proposal and Project Plan	– 19 May
• Presentation of Project Proposals	– 27 May
• Review of staff feedback from presentations	– 11 June
• Revised Proposal Finalized	– 11 June
• Project Web Presence	– 12 June
• Initial Feasibility Demonstration	– 20 July to 24 July
• Background/Theory Section of final paper	– 24 July
• Design section	– 21 August
• First Implementation and Writeup	– 11 September
• Final Prototype/Experiment/Performance Test + Writeup	– 21 September
• Sections on Implementation and Testing	– 25 September
• Outline of complete report	– 2 October
• Final Complete Draft of Report	– 16 October
• Weighting for project marking decided	– 16 October
• Project Report Final	– 26 October
• Poster	– 2 November
• Website	– 9 November
• Reflection Paper	– 13 November
• Demonstration	– 18 November

5.8. Work Allocation

The work is going to be split into 3 sections which are the three mentioned in the procedures and methods section. Ryan is going to do the online PID generation Algorithm, Jed is going to do the AdaBoost optimisations and Tapiwa is going to do the weak learner selection. The sections have been designed such that there are no interdependencies and each can be evaluated separately. If the project is successful and there is time there must be an integration of the entire system which will be done as a group. During the development stage of the project we must keep integration in mind because this may affect some development decisions.

REFERENCES

Wet al Ackermann, G Asova, V Ayvazyan, A Azima, N Baboi, J Bähr, V Balandin, B Beutner, A Brandt, A Bolzmann, and others. 2007. Operation of a free-electron laser from the extreme ultraviolet to the water window. *Nature photonics* 1, 6 (2007), 336–342.

Hans-Georg Beyer and Hans-Paul Schwefel. 2002. Evolution strategies–A comprehensive introduction. *Natural computing* 1, 1 (2002), 3–52.

Yan-Shi Dong and Ke-Song Han. 2005. Boosting SVM classifiers by ensemble. In *Special interest tracks and posters of the 14th international conference on World Wide Web*. ACM, 1072–1073.

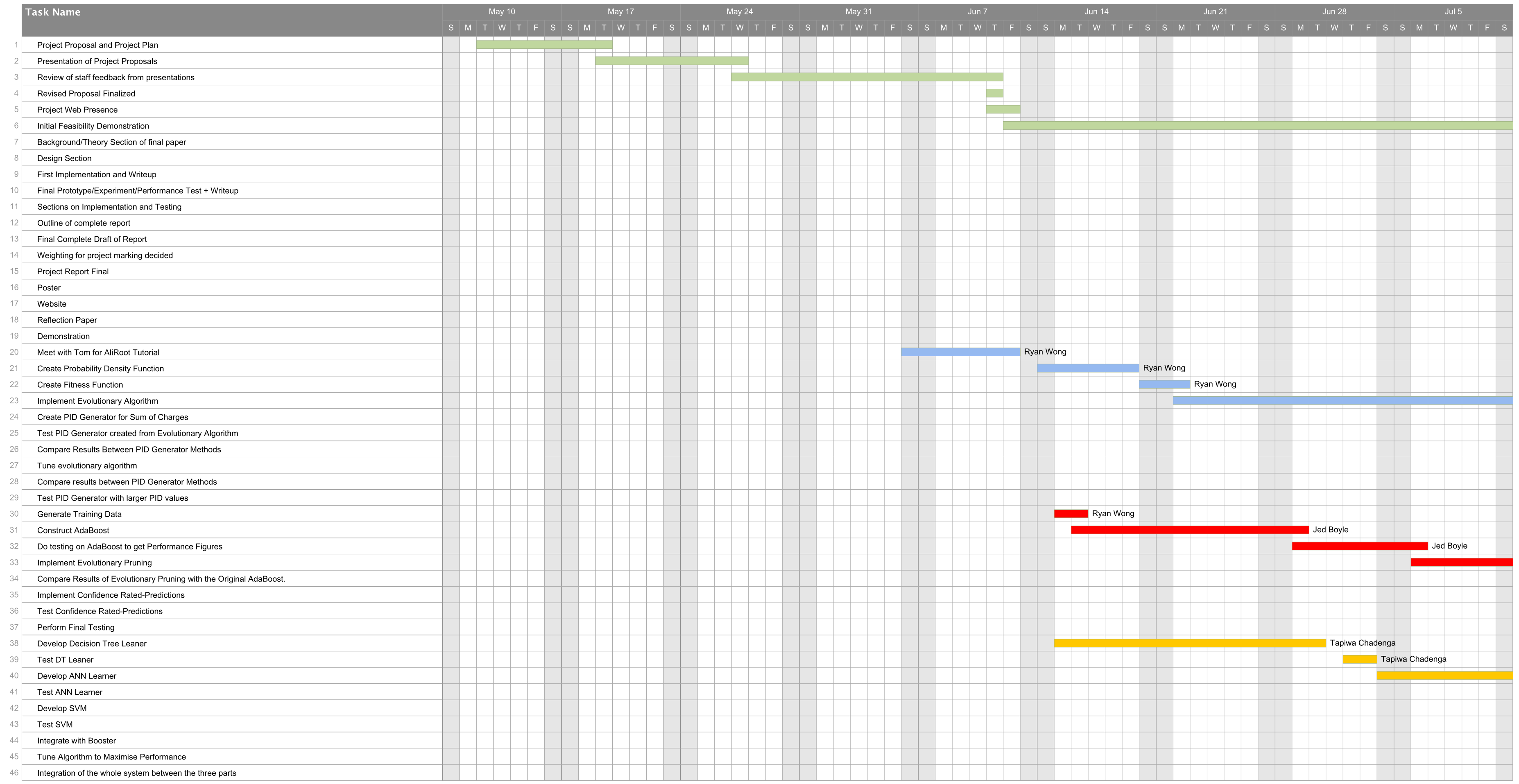
Jun-Su Jang and Jong-Hwan Kim. 2008. Fast and robust face detection using evolutionary pruning. *Evolutionary Computation, IEEE Transactions on* 12, 5 (2008), 562–571.

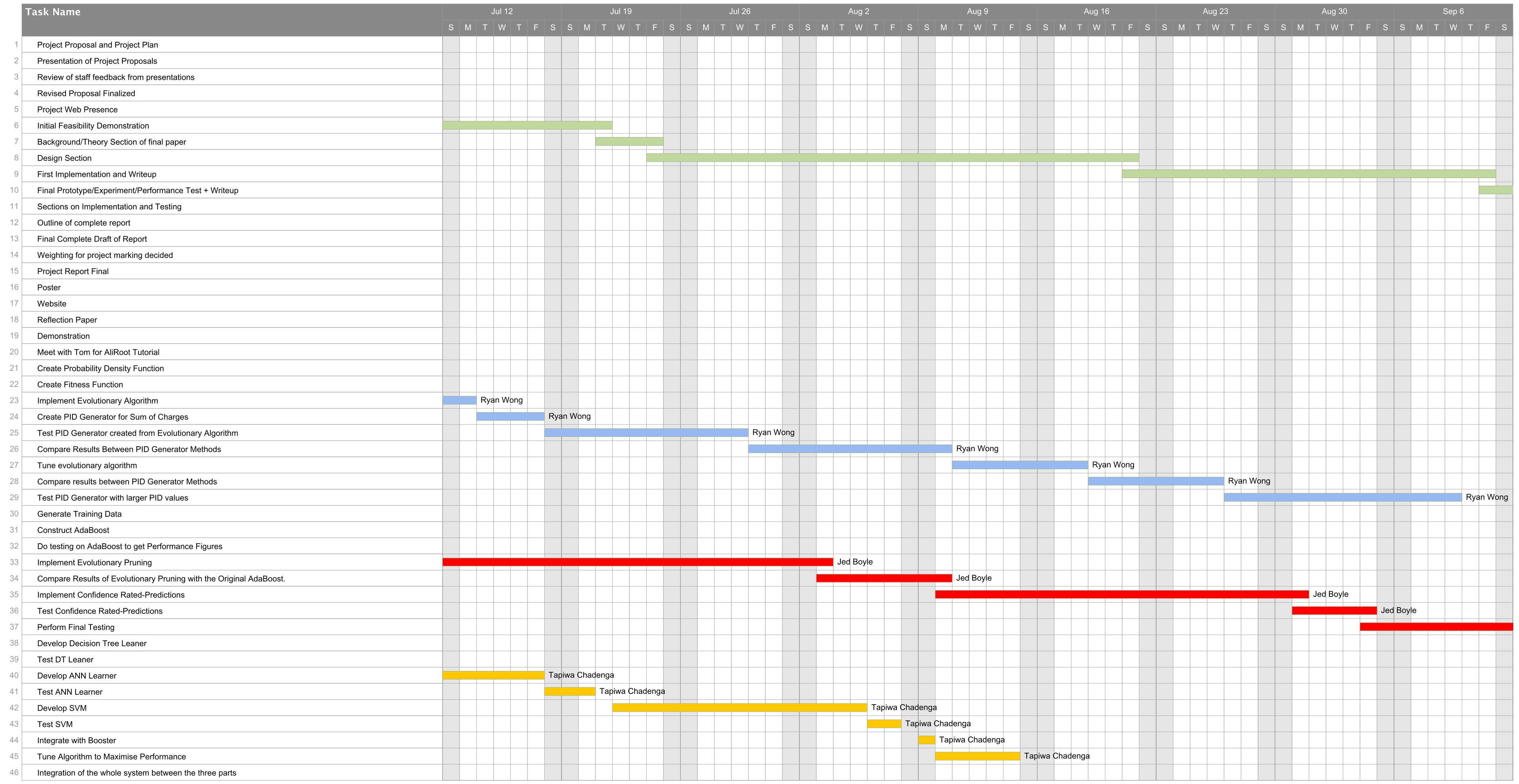
Mikael Kuusela, Jerry W Lämsä, Eric Malmi, Petteri Mehtälä, and Risto Orava. 2010. Multivariate techniques for identifying diffractive interactions at the LHC. *International Journal of Modern Physics A* 25, 08 (2010), 1615–1647.

Robert E Schapire. 1999. A brief introduction to boosting. In *Ijcai*, Vol. 99. 1401–1406.

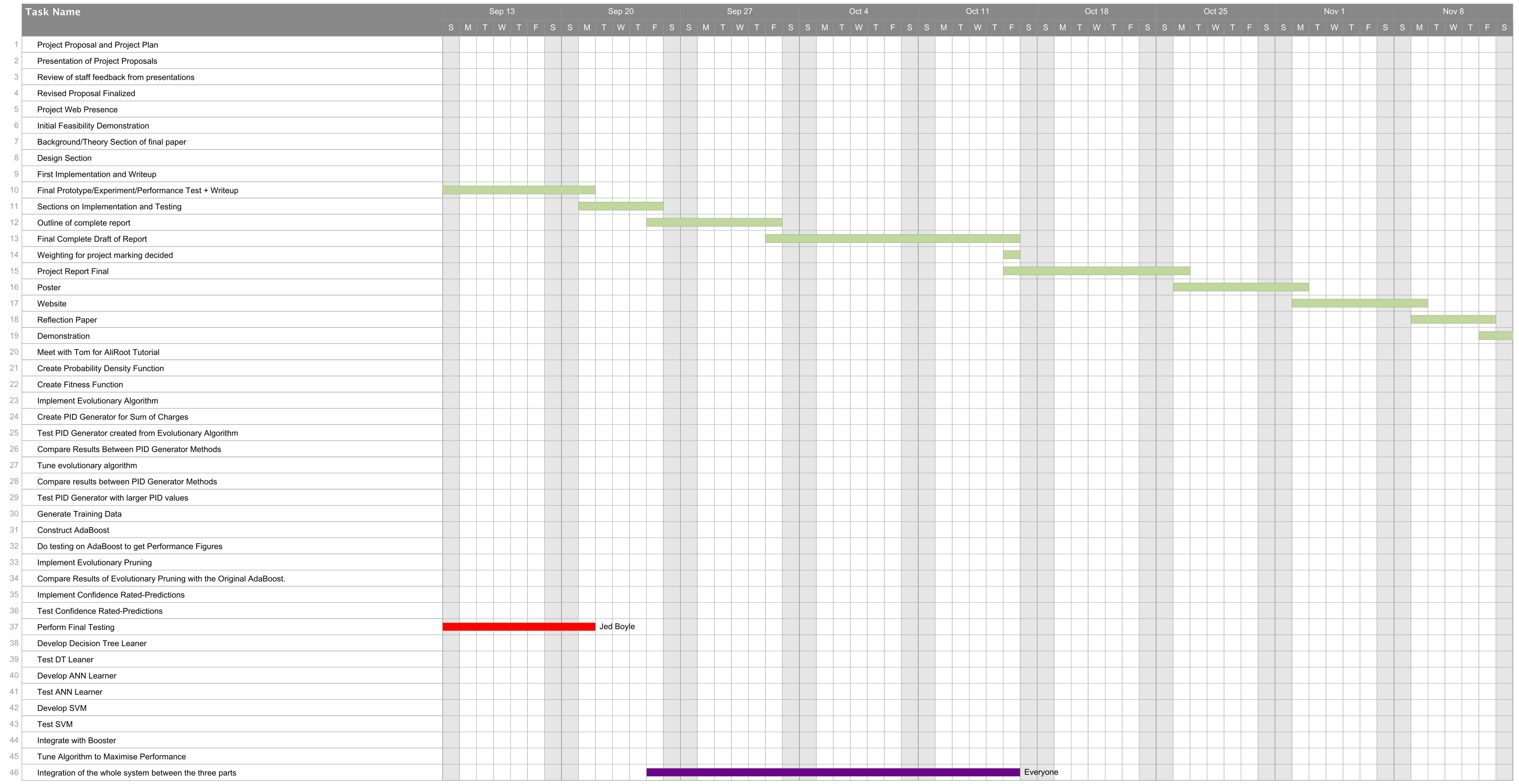
Robert E Schapire and Yoram Singer. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine learning* 37, 3 (1999), 297–336.

Alexander Wilk. 2010. *Particle identification using artificial neural networks with the ALICE transition radiation detector*. Ph.D. Dissertation. Mnster (Westfalen), Univ., Diss., 2010.


Task Name
May 10
May 17
May 24
May 31
Jun 7
Jun 14
Jun 21
Jun 28
Jul 5
S M T W T F S
1 Project Proposal and Project Plan
2 Presentation of Project Proposals
3 Review of staff feedback from presentations
4 Revised Proposal Finalized
5 Project Web Presence
6 Initial Feasibility Demonstration
7 Background/Theory Section of final paper
8 Design Section
9 First Implementation and Writeup
10 Final Prototype/Experiment/Performance Test + Writeup
11 Sections on Implementation and Testing
12 Outline of complete report
13 Final Complete Draft of Report
14 Weighting for project marking decided
15 Project Report Final
16 Poster
17 Website
18 Reflection Paper
19 Demonstration
20 Meet with Tom for AliRoot Tutorial
21 Create Probability Density Function
22 Create Fitness Function
23 Implement Evolutionary Algorithm
24 Create PID Generator for Sum of Charges
25 Test PID Generator created from Evolutionary Algorithm
26 Compare Results Between PID Generator Methods
27 Tune evolutionary algorithm
28 Compare results between PID Generator Methods
29 Test PID Generator with larger PID values
30 Generate Training Data
31 Construct AdaBoost
32 Do testing on AdaBoost to get Performance Figures
33 Implement Evolutionary Pruning
34 Compare Results of Evolutionary Pruning with the Original AdaBoost.
35 Implement Confidence Rated-Predictions
36 Test Confidence Rated-Predictions
37 Perform Final Testing
38 Develop Decision Tree Leaner
39 Test DT Leaner
40 Develop ANN Learner
41 Test ANN Learner
42 Develop SVM
43 Test SVM
44 Integrate with Booster
45 Tune Algorithm to Maximise Performance
46 Integration of the whole system between the three parts
Ryan Wong
Ryan Wong
Ryan Wong
Ryan Wong
Jed Boyle
Jed Boyle
Tapiwa Chadenga
Tapiwa Chadenga



	Task Name	
1	Project Proposal and Project Plan	
2	Presentation of Project Proposals	
3	Review of staff feedback from presentations	
4	Revised Proposal Finalized	
5	Project Web Presence	
6	Initial Feasibility Demonstration	
7	Background/Theory Section of final paper	
8	Design Section	
9	First Implementation and Writeup	
10	Final Prototype/Experiment/Performance Test + Writeup	
11	Sections on Implementation and Testing	
12	Outline of complete report	
13	Final Complete Draft of Report	
14	Weighting for project marking decided	
15	Project Report Final	
16	Poster	
17	Website	
18	Reflection Paper	
19	Demonstration	
20	Meet with Tom for AliRoot Tutorial	
21	Create Probability Density Function	
22	Create Fitness Function	
23	Implement Evolutionary Algorithm	Ryan Wong
24	Create PID Generator for Sum of Charges	Ryan Wong
25	Test PID Generator created from Evolutionary Algorithm	Ryan Wong
26	Compare Results Between PID Generator Methods	Ryan Wong
27	Tune evolutionary algorithm	Ryan Wong
28	Compare results between PID Generator Methods	Ryan Wong
29	Test PID Generator with larger PID values	Ryan Wong
30	Generate Training Data	
31	Construct AdaBoost	
32	Do testing on AdaBoost to get Performance Figures	
33	Implement Evolutionary Pruning	Jed Boyle
34	Compare Results of Evolutionary Pruning with the Original AdaBoost.	Jed Boyle
35	Implement Confidence Rated-Predictions	Jed Boyle
36	Test Confidence Rated-Predictions	Jed Boyle
37	Perform Final Testing	
38	Develop Decision Tree Leaner	
39	Test DT Leaner	
40	Develop ANN Learner	Tapiwa Chadenga
41	Test ANN Learner	Tapiwa Chadenga
42	Develop SVM	Tapiwa Chadenga
43	Test SVM	Tapiwa Chadenga
44	Integrate with Booster	Tapiwa Chadenga
45	Tune Algorithm to Maximise Performance	Tapiwa Chadenga
46	Integration of the whole system between the three parts	



	Task Name
1	Project Proposal and Project Plan
2	Presentation of Project Proposals
3	Review of staff feedback from presentations
4	Revised Proposal Finalized
5	Project Web Presence
6	Initial Feasibility Demonstration
7	Background/Theory Section of final paper
8	Design Section
9	First Implementation and Writeup
10	Final Prototype/Experiment/Performance Test + Writeup
11	Sections on Implementation and Testing
12	Outline of complete report
13	Final Complete Draft of Report
14	Weighting for project marking decided
15	Project Report Final
16	Poster
17	Website
18	Reflection Paper
19	Demonstration
20	Meet with Tom for AliRoot Tutorial
21	Create Probability Density Function
22	Create Fitness Function
23	Implement Evolutionary Algorithm
24	Create PID Generator for Sum of Charges
25	Test PID Generator created from Evolutionary Algorithm
26	Compare Results Between PID Generator Methods
27	Tune evolutionary algorithm
28	Compare results between PID Generator Methods
29	Test PID Generator with larger PID values
30	Generate Training Data
31	Construct AdaBoost
32	Do testing on AdaBoost to get Performance Figures
33	Implement Evolutionary Pruning
34	Compare Results of Evolutionary Pruning with the Original AdaBoost.
35	Implement Confidence Rated-Predictions
36	Test Confidence Rated-Predictions
37	Perform Final Testing
38	Develop Decision Tree Leaner
39	Test DT Leaner
40	Develop ANN Learner
41	Test ANN Learner
42	Develop SVM
43	Test SVM
44	Integrate with Booster
45	Tune Algorithm to Maximise Performance
46	Integration of the whole system between the three parts

	Task Name	Nov 8							Nov 15							Nov 22							Nov 29						
		S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S
1	Project Proposal and Project Plan																												
2	Presentation of Project Proposals																												
3	Review of staff feedback from presentations																												
4	Revised Proposal Finalized																												
5	Project Web Presence																												
6	Initial Feasibility Demonstration																												
7	Background/Theory Section of final paper																												
8	Design Section																												
9	First Implementation and Writeup																												
10	Final Prototype/Experiment/Performance Test + Writeup																												
11	Sections on Implementation and Testing																												
12	Outline of complete report																												
13	Final Complete Draft of Report																												
14	Weighting for project marking decided																												
15	Project Report Final																												
16	Poster																												
17	Website	🟩	🟩																										
18	Reflection Paper		🟩	🟩	🟩	🟩	🟩																						
19	Demonstration						🟩	🟩	🟩	🟩	🟩	🟩																	
20	Meet with Tom for AliRoot Tutorial																												
21	Create Probability Density Function																												
22	Create Fitness Function																												
23	Implement Evolutionary Algorithm																												
24	Create PID Generator for Sum of Charges																												
25	Test PID Generator created from Evolutionary Algorithm																												
26	Compare Results Between PID Generator Methods																												
27	Tune evolutionary algorithm																												
28	Compare results between PID Generator Methods																												
29	Test PID Generator with larger PID values																												
30	Generate Training Data																												
31	Construct AdaBoost																												
32	Do testing on AdaBoost to get Performance Figures																												
33	Implement Evolutionary Pruning																												
34	Compare Results of Evolutionary Pruning with the Original AdaBoost.																												
35	Implement Confidence Rated-Predictions																												
36	Test Confidence Rated-Predictions																												
37	Perform Final Testing																												
38	Develop Decision Tree Leaner																												
39	Test DT Leaner																												
40	Develop ANN Learner																												
41	Test ANN Learner																												
42	Develop SVM																												
43	Test SVM																												
44	Integrate with Booster																												
45	Tune Algorithm to Maximise Performance																												
46	Integration of the whole system between the three parts																												