Systems Engineering for Computational Physics

Engineering a Big Data System for Analysing Radiation Data

Abhishek Shenoy

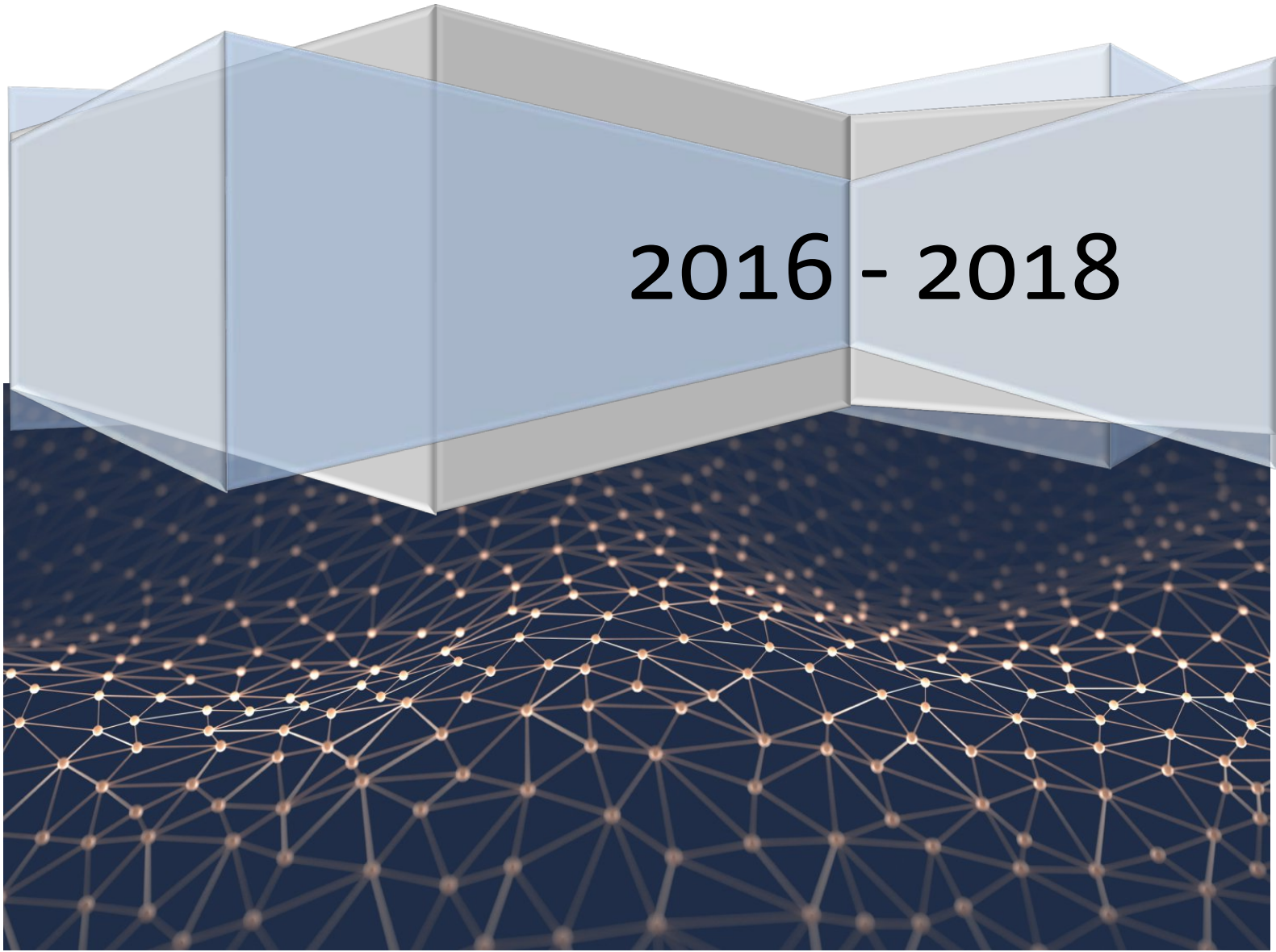


Table of Contents

Analysis

Background

The Institute for Research in Schools is a charity that focuses on bringing high-level research in all schools all around the UK. The institute has small hand-held radiation detectors which records particle tracks as matrices. Several of these detectors have been combined into a single detector device called LUCID and this device has been placed into a low Earth orbit on a satellite. The data from this experiment is collected every two weeks to a server. The institute aims to encourage research in schools by propagating projects such as LUCID all around the country.

Identification of the Problem

1. **Data Collection** - The data that is collected needs to be accessible and easy to use by a student wanting to develop the project further (a student with sufficient knowledge of the data).

    a. All of the data must be accessible to anyone who wants to use it so the data must be in a similar format to the current method of storage.
    b. The schema of the data must be easy to read so that data analysis can be performed easily as a part of the analysis system.

2. **Data Analysis** - The data needs to be analysed so that the particles that are detected are classified based on their properties with a very high accuracy.

    a. As the data is actually advanced in terms of analysis, the final analysis must be the output with the only given inputs being a single datum file.
    b. To make sure that the new analysis system is better than the current system, the new system will have to be an innovative and a very technical solution to achieve a higher accuracy.

3. **Data Visualisation** - The analysed data needs to be presentable to the wider public as well as a member of the institute.

    a. The data must be easy to visualise with the use of simple visualisation techniques such as geolocation on maps and statistical data on graphs.
    b. The software used to present the data must be a web application so that it can be accessible by the general public.
    c. The web application must be interactive and insightful to give a full view of the work that has been done on the project.

Current System

The current system has many difficulties with regards to solving the three problems (explained above):

1. **Data Collection**
    a. The data is currently only accessible by navigating directories on a webpage. However, this is very slow for a user wanting to utilise the data.
    b. The metadata is currently not accessible to the user as it is stored in a private centralised database.

2. **Data Analysis**
    a. Various metrics, describing the particle track, are calculated and these metrics are then used in conditional loops to classify the particle. Therefore, the analysis algorithm is very simplistic (shown in the Documented Design section). This is very inaccurate as particles are not simple categories that can be easily distinguished between. The classes of particle tracks are almost continuous as many of the classes can overlap due to similarities in the shape and energy of the tracks and as the current analysis is very categorical, the analysis is inaccurate.

3. **Data Visualisation**
    a. The data is only viewable after downloading and there is too much data to download.
    b. As the data is also technical, only a programmer is actually capable of converting the downloaded data into a viewable image.
    c. There is no web application allowing the visualisation of the data.

Prospective User

The three parts of this project will form the backbone of the LUCID Project allowing students to build upon the research that has already been done as well as allowing the wider public to view the progress being done in the scientific field. The students will have a good deal of scientific understanding as well as an understanding of big data and data science. The wider public will be the more general audience and will not be able to understand the data by itself or even in a presentable format. Researchers and teachers will have a good knowledge of the project and will be able to guide students to getting started with viewing, using and analysing data.

1. All the data and metadata must be accessible from a single access point. This will mean that all end users will have full access to the data with the same procedures.
2. The data must be in a convenient and usable format to allow all users to use the data on their own local computer.
3. Students must be able to easily view and analyse the data upon request.
4. The analysis must be very accurate to reduce false conclusions in student research.
5. The analysis must be viewable and presentable to the public in a comprehensive manner.

Proposed Objectives

1. **User Need:** All the data and metadata must be accessible from a single access point.
   Specific Objectives:
   The user provides a specific ID of a data file to view the frames from that data file.
   The user can view all the XYC files (data frames) and their metadata such as timestamp, geolocation and detector settings for that frame.
   The data must be accessible from a web browser to provide easy access to the data.

2. **User Need:** The data must be in a convenient and usable format.
   Specific Objectives:
   The binary file downloaded on the server is already converted to the XYC file format and therefore this is the file that will be accessible to the user. This file format is easier to analyse and easily usable with limited knowledge of the data. XYCs are easily downloadable with given download links.

3. **User Need:** Students must be able to easily view and analyse the data upon request.
   Specific Objectives:
   A given XYC file will be analysed when a user submits the file ID, the frame number and the active detector as parameters in a URL.
   A given XYC file will be analysed when a user submits the contents of the file into a form element.
   All XYC files are viewable as images so that the user does not have to process the images. Energy values are converted to coloured pixels based on the amount of energy deposit.
   User interface is easy to navigate with each page having its own respective purpose.

4. **User Need:** The analysis must be very accurate to reduce false conclusions.
   Specific Objectives:
   The analysis algorithm must be innovative and fully accounting to reduce inaccuracies in the classification of particles.
   A lot of example data must be collected to allow the accuracy to be tested.
   The analysis must provide a range of classifications to make the current analysis system better.

5. **User Need:** The analysis must be viewable and presentable to the public.
   Specific Objectives:
   The particle classifications need to be visible on the web application so that users themselves are minimally involved with the programming side making further data analysis easier to do.
   The public must be able to interact with the data to make them more interested in getting involved with the project.
   Interactive features include data search, interactive graphs, energy filters on data frames, geolocation of LUCID frames on a satellite map and documentation of API endpoints.

Proposed System

The purpose of this project is to demonstrate the construction of a more accurate and contemporary analysis algorithm for analysing particle tracks generated by Timepix chips using machine learning. The process of research, investigation and development of the analysis algorithm is described and explained with respect to particle physics, computational heuristics and mathematical statistics. The novel analysis algorithm will be a neural network that can be integrated into a web platform to classify particles from a range of detector chips for a variety of research projects. The main research project that this paper will focus on is the LUCID (Langton Ultimate Cosmic Ray Intensity Detector) project aiming to analyse radiation data collected in the Low Earth Orbit.

Proposed System

1. **Data Collection** - The data that is collected needs to be accessible and easy to use by a student wanting to develop the project further (a student with sufficient knowledge of the data).

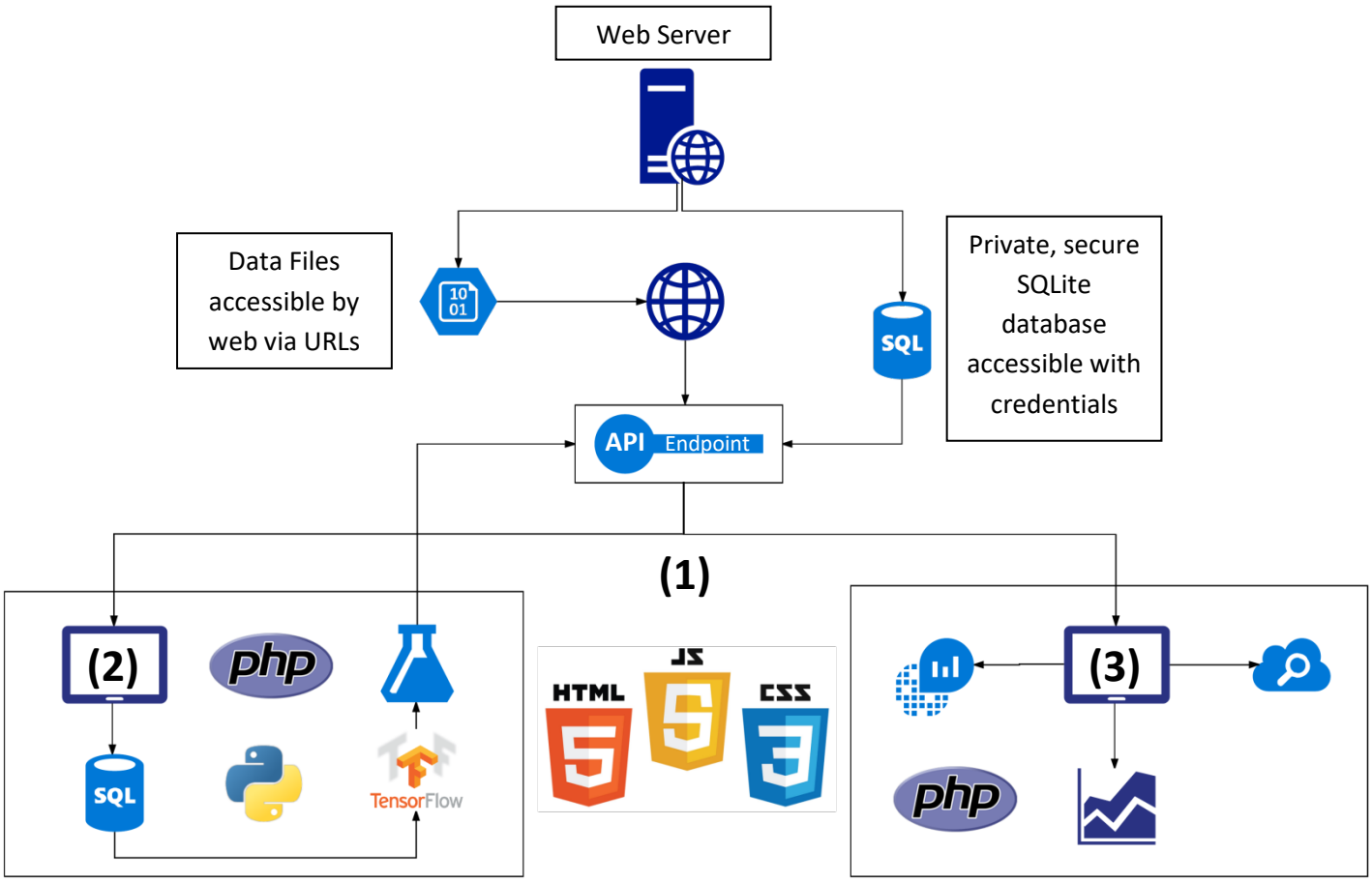
    a. As the student will have sufficient knowledge of the data, instead of simply downloading data from the server, a more preferable method would be an API through which they can create their own projects using the data.

2. **Data Analysis** - The data needs to be analysed so that the particles that are detected are classified based on their properties with a very high accuracy.

    a. The current analysis system is very inaccurate and therefore to improve the analysis algorithm, a machine learning algorithm would be much more accurate as it will be able to account for the inconsistencies in the particles classes.
    b. As choosing a machine algorithm will be a process of experimentation, the primary aim will be to develop multiple classifiers that can compete with the current algorithm. The most accurate classifier can then be used or a multi-vote system could be used by taking the most popular classification from all the classifiers.

3. **Data Visualisation** - The analysed data needs to be presentable to the wider public as well as a member of the institute.

    a. Designing a web application will allow the data to be visualised online and accessible to the wider public allowing research findings to be presented more easily.



As the metadata database is private while the data files are publicly accessible by the web, the best way to allow students, researchers and the public to access the data was to combine the two sources into a public API **(1)**. This would then allow the API to be used as a data access point for creating the web applications for analysing the data **(2)** and visualising the data **(3)**.

(1) will be a public API allowing multiple URL callbacks each performing specific functions relating to the processing of LUCID data.

(2) will include a citizen science platform (created using PHP, HTML, JS, CSS) to generate training data for the machine learning algorithms for future analysis. The results from the platform will be stored in another private SQLite database and then this data will be used to train machine learning algorithms in Python created using the Tensorflow library. Python is more suitable for machine learning as PHP is designed more for backend management than research purposes. The Tensorflow models that have been trained will then be reconnected to the API so that users can analyse data with the created models rather than having to recreate their own. For naming purposes, I will reference this part of the project as the LUCID Trainer (the training application), the LUCID Neural Analysis (the neural network) and the LUCID Classifiers (other machine-learning classifiers) as a part of Data Analysis.

(3) will be a user-friendly dashboard (created using PHP, HTML, JS, CSS) to browse all the data files and frames as well as providing operational insights about the satellite and graphs generated from the collected data files. The user interface will be interactive with many features for data manipulation to make users interested in the project work. For naming purposes, I will reference this part of the project as the LUCID Dashboard as a part of Data Visualisation.

Research – Data Collection

Server-Side Management

The server has 12GB of RAM and 16 Intel Xeon cores clocked at 2.53GHz, and runs Ubuntu 16.04 LTS. It has 146GB of storage for the OS and 21TB for the data. Every 8 days, a cron job is used to check an FTP server hosted by SSTL to download any new LUCID data to the server.

Each data frame is stored as an individual file inside its designated data run folder on the server for the specific data file ID, frame number and the detector number.
(https://starserver.thelangton.org.uk/data/LUCID/xyc/data_run/id/frame_channel.txt)
Example: https://starserver.thelangton.org.uk/data/LUCID/xyc/2016-12-16/0747002227/frame1c0.txt
The above URL returns the XYC file as plaintext in a browser.

The metadata of all the data files is stored in a SQLite database consisting of the following fields:

- ID – The ID of the data file
- Filename – The name of the data file with the storage path
- Latitude, Longitude – The location of where the data file was created
- Timestamp – The time when the data file was created
- Config – The configuration file used for the data file
- Active Detectors – The detectors that were used in the data file
- Number of Frames – The total number of data frames collected for the data file
- Data Run – The date the 8 day data run began (when the data file was created)

Data API

Data Format

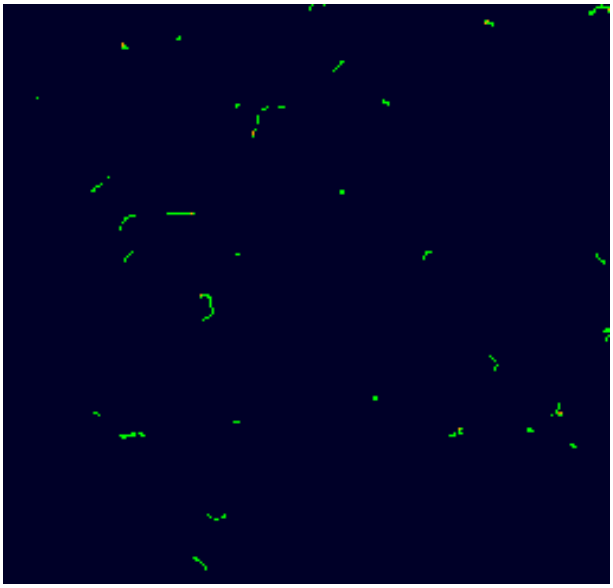
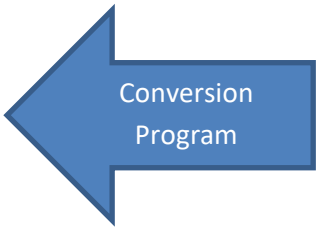


Figure 2 - PNG of a Data Frame



14	41	10.0
37	82	49.0
38	81	22.0
38	82	17.0
38	180	19.0
39	80	5.0
39	180	40.0
40	80	7.0
40	181	19.0
41	79	33.0
44	76	30.0
49	98	31.0
49	99	26.0
49	190	21.0
50	17	135.0
50	18	143.0
50	19	8.0
50	96	16.0
50	190	15.0
50	191	19.0
51	18	22.0
51	19	17.0
51	94	19.0
51	95	44.0
51	112	7.0

Figure 1 - Partial XYC of a Data Frame

The three columns of the XYC file in **Error! Reference source not found.** represent the x-coordinate, the y-coordinate and the energy (Time over Threshold) value, hence the name XYC. The XYC can be converted to an image (**Error! Reference source not found.**) simply by using a conversion program which plots each pixel depending on its energy value to create an image of the data frame.

Research – Data Analysis

Neural Networks

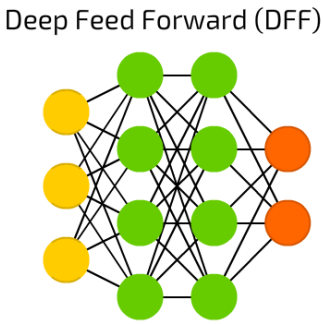
Introduction

Neural networks are a machine learning method used to solve data classification problems. Neural networks learn by creating a training model using a training dataset containing an array of inputs with their expected outputs. This type of machine learning is known as supervised machine learning. The training dataset is passed through a series of interconnected nodes and the features of the training dataset are interpreted and learned by the mathematical modification of the neural weights for each individual node. This neural model can then check a given input to give an accurate representation of the expected output.

Architectures

The fast development in the field of machine learning and artificial intelligence means that different neural architectures are being invented very frequently. This means that there are several variations of names and abbreviations for the same or similar neural architectures [15].

Deep Feed Forward Network



A deep feed forward network is also known as a multi-layer feed forward neural network and it is the basic neural network architecture used to build up other neural networks with far more advanced uses. In a deep feed forward network, there are three types of layers: the input layer, the hidden layers and the output layer. The input layer receives inputs from a training dataset and propagates it to the hidden layers. The neural model is created by adjusting the weights on the connections to the hidden layers by backpropagation until the error of the neural network is minimal for the corresponding outputs of the given dataset. The neural model is stored as a series of weights and can be used again to classify inputs for an unknown output.

Layered Structure

For a neural network with 2 hidden layers with the activation functions being sigmoid function for the hidden layers and then softmax in the output layer generates the following equations:

$$x_i = Input\ at\ Node\ i$$

$$f_j = \sigma\left(w_j \cdot x_i + b_j\right) \quad where \quad \sigma(z) = \frac{1}{1 + e^{-z}}$$

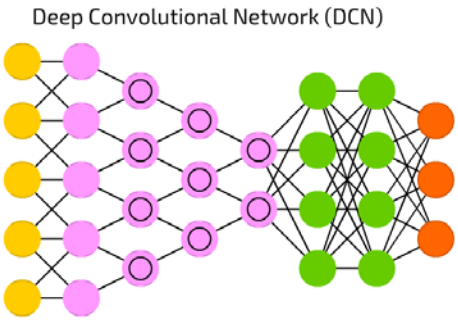
$$f_k = \sigma\left(w_k \cdot f_j + b_k\right) \quad where \quad \sigma(z) = \frac{1}{1 + e^{-z}}$$

$$y_l = \rho(w_l \cdot f_k + b_l) \quad where \quad \rho(z) = \frac{e^{z_k}}{\sum_{p=1}^{N} e^{z_p}} for\ l = 1, \dots, N$$

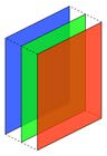
$$y_l = Output\ at\ Node\ l$$

The above equations intuitively explain how a neural network works. Although these equations explain the structure of a neural network they do not demonstrate how they are trained (how the weights are updated). Also these equations only explain the output from one node in a layer to another single node in the next layer. The actual output is based on the sum of the outputs going to each node.

Deep Convolutional Network



A deep convolutional neural network [16] is essentially a neural network for image classification where the input is an image matrix (a 3-dimensional matrix with dimensions $h * w * d$).



Each image input matrix is such that the height and width is the height and width of the image and the depth is the number of colour components. (RGB is 3)

There are two different types of layers for image processing in a deep convolutional network: the convolutional layer and the subsampling layer.

Convolutional Layers

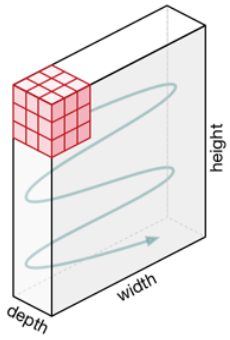


Figure 3 - Convolution

Convolutional layers run a convolution kernel on the 3D tensor where the convolutional kernel is also 3-dimensional with a small value for width and height but the depth being the same as the input image. In Figure 3 [17], the red cube is slid across the entire volume of the input image and at each position, the 27 elements are summed up and written to the output. The weights here belong to the convolution kernel and as there are 27 elements, there will be 27 weights for this kernel. As in a deep neural network, the weights describe the strength of the connection between an input and output. Using only this convolution kernel, the output matrix then has the same width and height as the original but the depth is now 1. However, each convolutional layer has more than one convolution kernel and therefore the output volume is a matrix with the same width and height but the depth being the number of convolution kernels.

Subsampling Layers

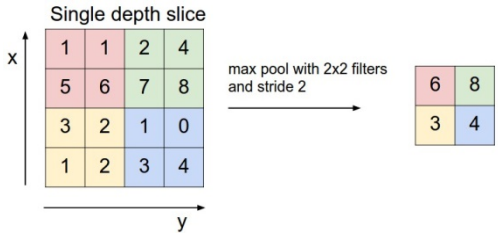


Figure 4 - Subsampling

Subsampling layers are needed to reduce the size of the input to allow focussing on significant features. There are multiple ways to subsample, but the most popular are max pooling, average pooling and stochastic pooling. For subsampling layers, a fixed subsampling region is defined (Figure 4).

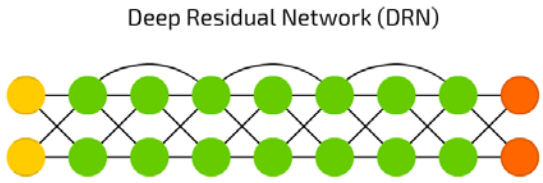
- Max pooling takes the maximum value in the region.
- Average pooling takes the average in the region.
- Stochastic pooling uses the normalisation of an activation function to generate probabilities to weight the values in the region.

Taking an example, if a pooling layer has a subsampling region of 2x2 and the input consists of a 16x16 matrix then the output would be a 8x8 matrix, meaning that 4 pixels (each 2x2 square) of the input matrix are combined into a single output pixel using one of the pooling processes (Figure 4).

Fully-Connected Layers

The last layer of the convolutions and pooling is usually connected to one or more fully connected layers, the last of which is the output layer representing the target data. Training is performed using modified backpropagation that takes the subsampling layers into account and updates the convolutional filter weights based on all values to which that filter is applied. This is essentially a deep forward neural network attached to the end of an image reduction process.

Deep Residual Network



Deep residual networks are neural networks made up of interconnected blocks of neural networks. Therefore, the old input for the first layer can be re-entered into the neural network three layers later so that different features are learned at each layer. This makes this neural network highly efficient at learning patterns. Moreover, the interconnected layers mean that removing a single block simply reduces the accuracy in a proportional manner. However, removing a layer in a deep feed forward neural network can change the accuracy completely. Deep residual networks tend to be very deep and can use convolutional and pooling layers for image classification tasks.

Residual Blocks

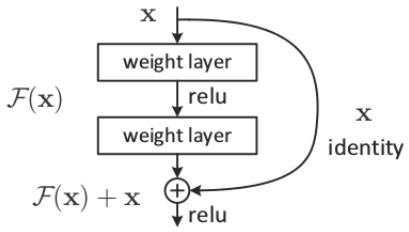


Figure 5 - Residual Block

Residual networks are made up of residual blocks. Each residual block has multiple weighted layers (Figure 5) [18] and as residual networks tend to be used for image classification, these weighted layers tend to be convolutional layers. The greater the number of blocks, the more the network learns and therefore the accuracy is usually relatively proportional to the depth of the network.

Benchmark Classifiers

To evaluate the best method for the classification of particles, it was necessary to research other common methods for machine learning to compare the results with the neural networks. As these classifiers were not the focal point of the project, my research is more concerned about the evaluation of these classifiers rather than how they function.

Decision Tree

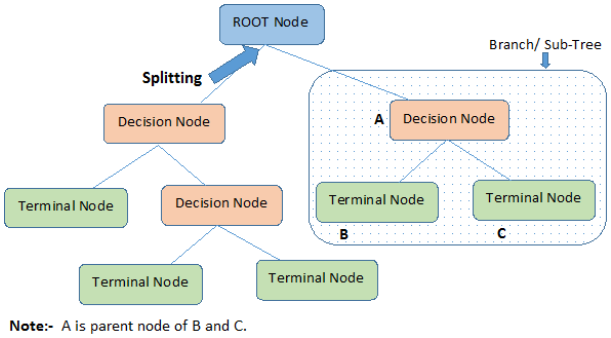


Figure 6 - Decision Tree

Classification and regression trees (CART) use decision trees for machine-learning. The number of features in the training data are used as the number of candidate splits. Using a cost function, the accuracy cost of each split is calculated and the split with the least cost is chosen. So, the primary root node of the decision tree tends to be the discriminating feature for most of the data (Figure 6) [19]. The algorithm is recursive and therefore the groups can be sub-divided in the same manner. The other features are used as the decision nodes to improve the class prediction accuracy.

Advantages:

- ✓ Decision trees quickly identify most significant variables and relation between multiple variables.
- ✓ It requires less data cleaning compared to some other modelling techniques and it is not greatly influenced by outliers or missing values.
- ✓ It can handle both numerical and categorical variables.

- ✓ Decision tree is considered to be a non-parametric method. This means that decision trees have no assumptions about the space distribution and the classifier structure.

Disadvantages:

- Over-fitting is one of the most practical difficulties for decision tree models. However, it can be solved by setting constraints on model parameters and pruning.
- While working with continuous numerical variables, decision tree loses information when it categorizes variables.

Random Forest

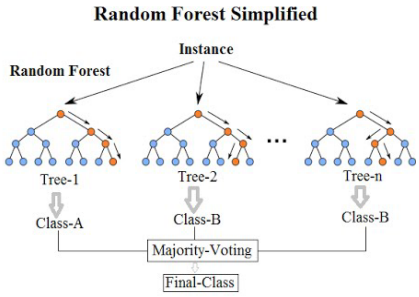


Figure 7 - Random Forest

A problem with decision trees is that they use a "greedy" algorithm to minimise error. Hence, multiple decision trees are often used to validate the accuracy (known as bagging). However even with bagging, the decision trees can have many structural similarities and in turn have high correlation in their predictions and therefore random forest classifiers are used as an ensemble method to improve accuracy. The random forest classifier once again uses multiple decision trees (Figure 7) [20] but it limits the search to a random sample of features so that the learned sub-trees are less similar and hence the resulting predictions have less correlation.

Advantages:

- ✓ Random forests are useful for both classification and regression.
- ✓ They balance errors due to skewed datasets and imbalances in class.
- ✓ They can easily handle high dimensionality data.
- ✓ Bootstrap sampling provides an effective method for estimating missing data.

Disadvantages:

- Overfitting can lead to inaccurate predictions for noisy datasets.
- Regression problems are only resolved within the range of the training dataset.

K-Nearest Neighbour

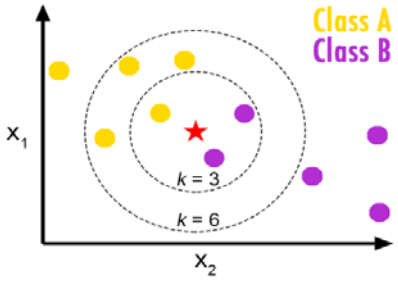


Figure 8 – K-Nearest Neighbour

The k-nearest neighbour classifier is very easy to interpret and to use for classification. The training data is plotted as a series of vectors and when given a test data, the algorithm simply determines k closest data points from the training dataset (Figure 8) [21].This is fundamentally used as the prediction where k is a hyperparameter to be determined. The classifier itself can use various distance measures for finding the nearest neighbour. The most common ones include Euclidean, Hamming, Minkowski and Manhattan distance measures.

Advantages:

- ✓ It is very robust to noisy training data.
- ✓ Highly effective if the training data is large.

Disadvantages:

- It can be very computationally expensive as the distance between each query instance and all the training samples need to be calculated.
- Optimal hyperparameters such as the number of nearest neighbours (K) and the type of distance measure are difficult to determine.
- When the class distribution is skewed "majority voting" classification can be inaccurate as they tend to be common among the k-nearest neighbours due to their large number.

Support Vector Machines

Unlike neural networks, which have multiple output neurons, a SVM always has one single output. However, this does not mean that you cannot do multi-class classification with a SVM. Multi-class classification is the usual reason for having multiple output neurons in a neural network however for a SVM the classes can be encoded as ordinal values between a specific range.

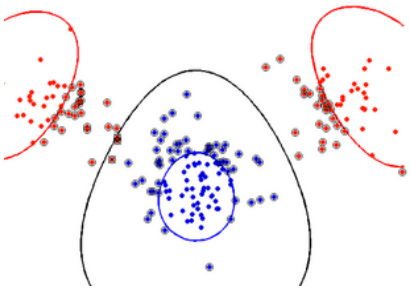


Figure 9 - RBF Kernel

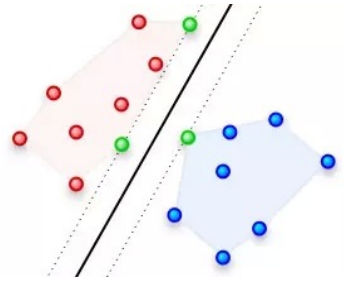


Figure 10 - Linear Kernel

The SVM algorithm tries to find the optimal separating hyperplane (multi-dimensional plane with n-1 dimensions where n is the number of dimensions of the training data) that maximizes the margin of the training data. Instead of an activation function, the SVM uses a kernel, which acts like an activation function. The kernel is usually a linear function (Figure 10) [22] however, for the purpose of particle classification I will be using the radial basis function (Figure 9) [23] assuming that the data is not linearly separable. The higher the value of the kernel coefficient, the SVM will try more to exact fit the training dataset. Adapting this value is used to calibrate the generalization error and prevent over-fitting. When a linear function is used, the gradient descent optimisation method is used to minimise the loss and improve classification accuracy.

Advantages:

- ✓ It works really well with clear margin of separation.
- ✓ It is effective in high dimensional spaces.
- ✓ It uses support vectors to make it more memory efficient.

Disadvantages:

- Performance is weaker when dataset has target classes that overlap.
- SVM does not directly provide probability estimates. These are calculated using an expensive five-fold validation.

Research – Data Visualisation

Visualisation Libraries

To understand more about LUCID data, the simplest way was to use data visualisation software alongside data mining techniques to display practical data in a format that conveys a correlation or relationship between multiple factors. To do this, I intended to create an online web application to allow the presentation of LUCID data in a user-friendly and useful manner. For presenting the data, I had to research different data analysis APIs and software to successfully evaluate and choose the most suitable ones for my web application.

Tableau Software

Tableau is a business intelligence and data-analytics software company based in Seattle. The proprietary Tableau software provides an easy way of visualising data from a range of statistical formats and a range of database management systems (DBMS) namely the popular ones being MySQL, CSV, SQLite and JSON. The software is mainly used for offline data visualisation and most of the analysis features are only provided offline however the graphs that are produced can be exported as an image or can be viewed in a web browser using Tableau Server for premium users.

Plotly Library

The JavaScript Plotly library is entirely open-source and the advantage of using JavaScript over the Python library is that it can be readily viewed on a website. The facility of Plotly to be integrated into various programming languages including Python and JavaScript, gives it much more functionality than Tableau and provides data visualisation with various forms of interactivity. Many more data formats can be easily integrated simply by standardising the data in a programming language. Plotly graphs also provide high amounts of user-interaction making this API highly suitable for an online web application.

ArcGIS Platform

The ESRI (Environmental Systems Research Institute) software company provides ArcGIS for location-based data analysis. Although it is very useful as a location-based analysis software, it lacks many of the features for data analysis. Moreover, the software itself is more designed for geological analysis rather than data visualisation.

Google Maps API

The Google Maps API provides a range of features however as the LUCID data is based on locational comparisons, I will only need to use two of the Google Maps APIs including the Maps JavaScript API and the Maps Geocoding API.

Google Maps Geocoding API
Convert between addresses and geographic coordinates.

Google Maps Javascript API
Customize maps with your own content and imagery.

The Google Maps API is very flexible, is fully apt at presenting geo-locational data, and provides features like place searches, proximity to an address, marker attachments and embedding on a webpage. This makes the API very useful for displaying the LUCID data on a webpage.

Final Solution

Specific Objectives

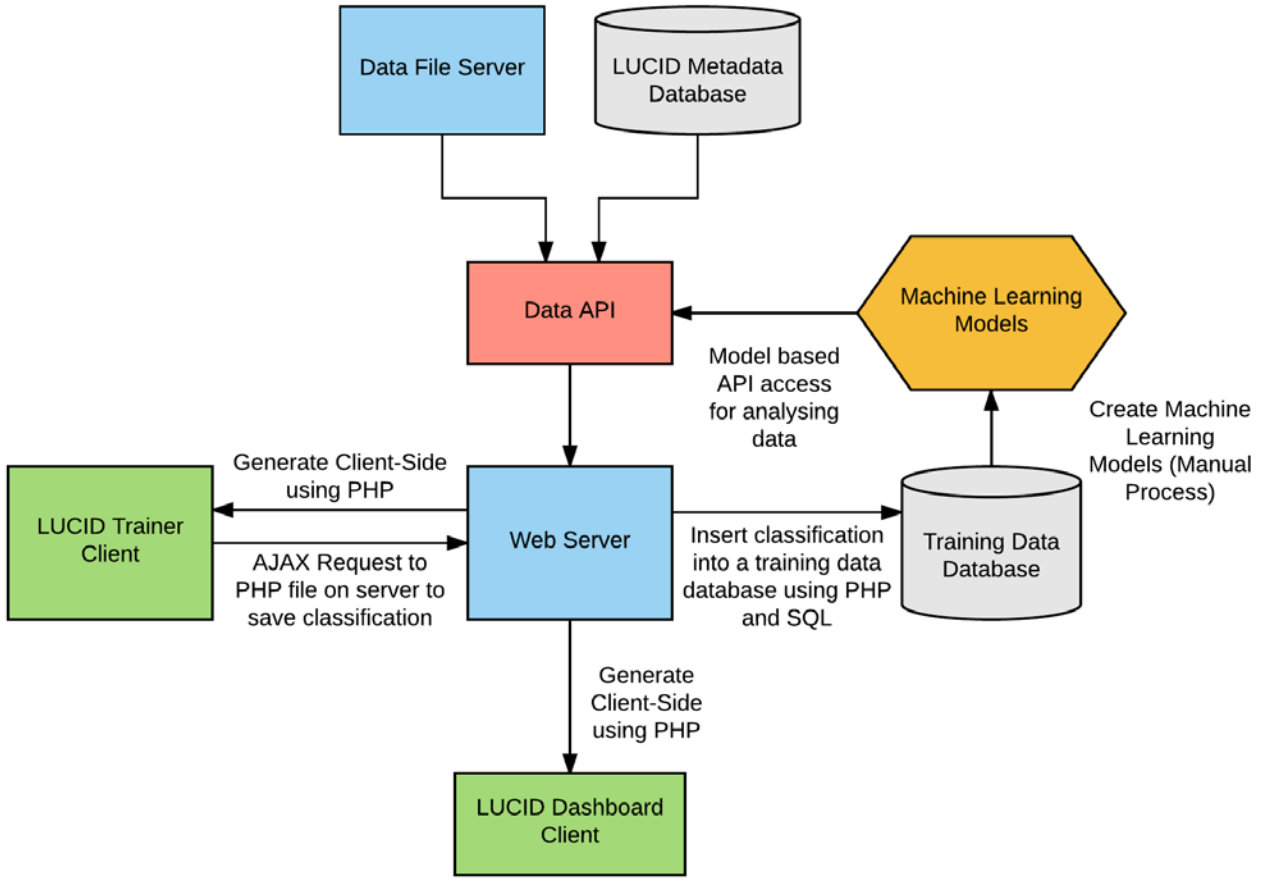
1. Data Collection – All the data and metadata must be accessible from a single access point.
    1.1. Database Model - The data must be in a convenient and usable format.
        1.1.1. Well-designed schema
        1.1.2. Easy to access and use data
    1.2. API Model - Students must be able to easily view and analyse the data upon request.
        1.2.1. Simple URL access
        1.2.2. Organised structural data

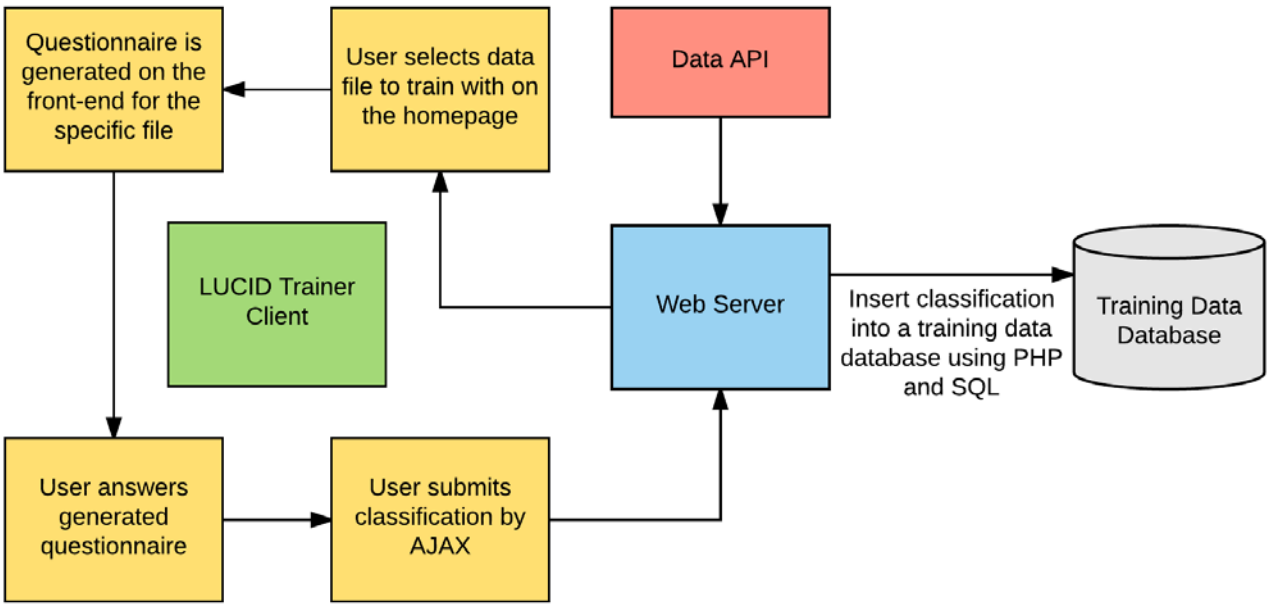
2. Data Analysis – Particles can be analysed by a computer autonomously with minimal human input.
    2.1. Particle Identification Algorithm – Identify all the individual particles in a frame.
    2.2. Feature Extraction from Particle Tracks – Pick out most important characteristics of specific particles.
        2.2.1. Creating Metrics
    2.3. Neural Network for Classification – Machine learning for classifying particles.

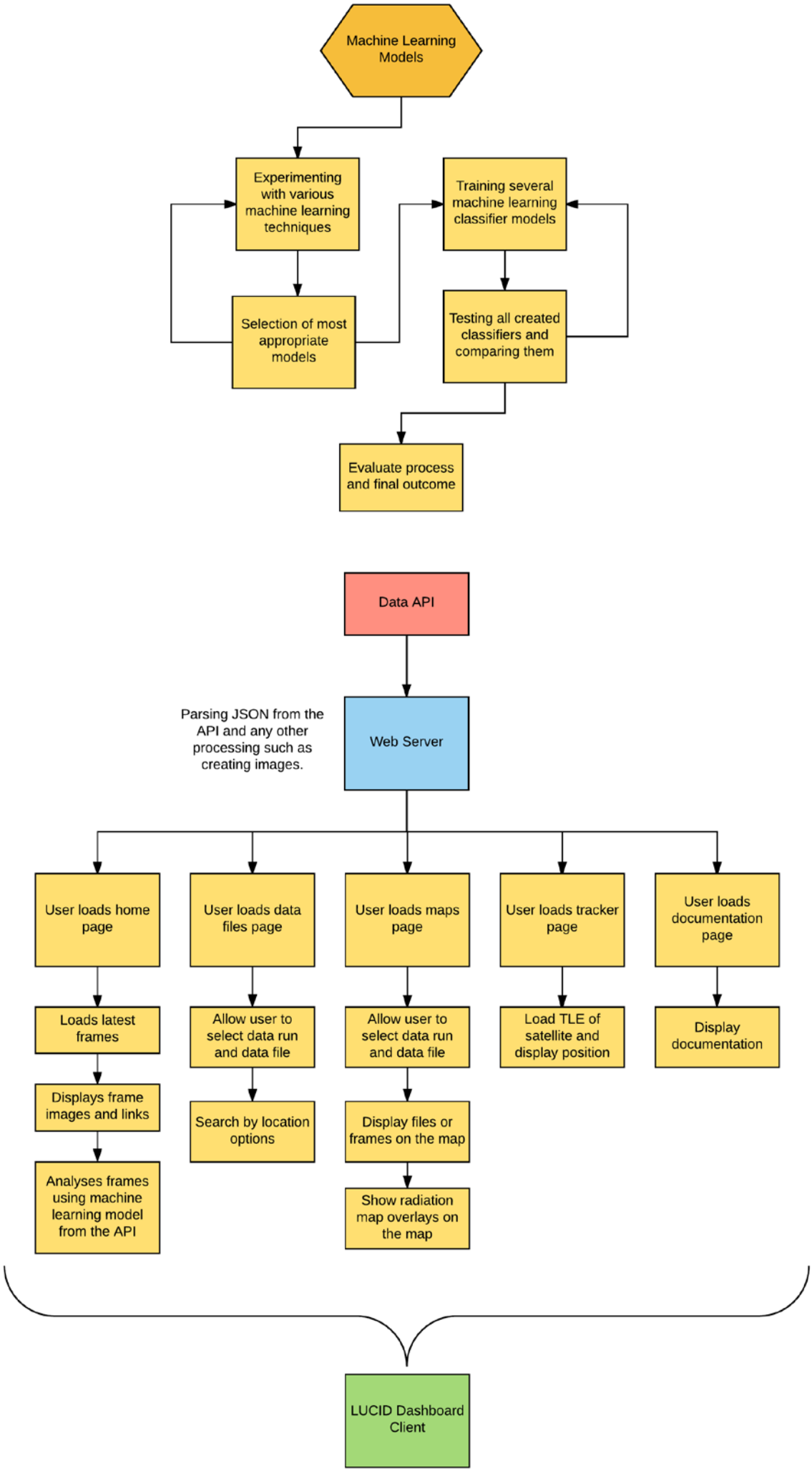
2.3.1. Training Web Application – Generate training data for the machine learning classifiers.
    2.3.1.1. Appropriate Method Selection – Select best method for generating training data.
    2.3.1.2. Efficient Database Model – Store training data in a database to access when needed.
    2.3.1.3. Utility Functions – Present all functions involved in the web application.
2.3.2. Neural Network Creation – Create and experiment with neural networks.
    2.3.2.1. Select Suitable Architecture – Select most appropriate model.
    2.3.2.2. Inputs & Outputs Encoding – Encoding inputs and outputs appropriately.
    2.3.2.3. Hyperparameter Selections – Optimise the neural network for best results.
    2.3.2.4. Final Neural Network
2.3.3. Model Training & Testing – Show training and testing stages of the neural network.
    2.3.3.1. Machine Learning Classifiers Evaluation – Compare neural network with other machine learning classifiers.
    2.3.3.2. South Atlantic Anomaly – Show results relating to the original physics-based investigation.
    2.3.3.3. Model Saving & Loading – Easy-access library for student and programmer analysis.

3. Data Visualisation – Build another web application for the presentation of all results related to the project.
    3.1. Show Latest Data Frames on Index Page
    3.2. Provide Analysis of a Given Frame
    3.3. Plot Data Frames and Files on a map
    3.4. Show Collection Statistics on Graphs
    3.5. Provide Documentation of the API

Solution Diagrams



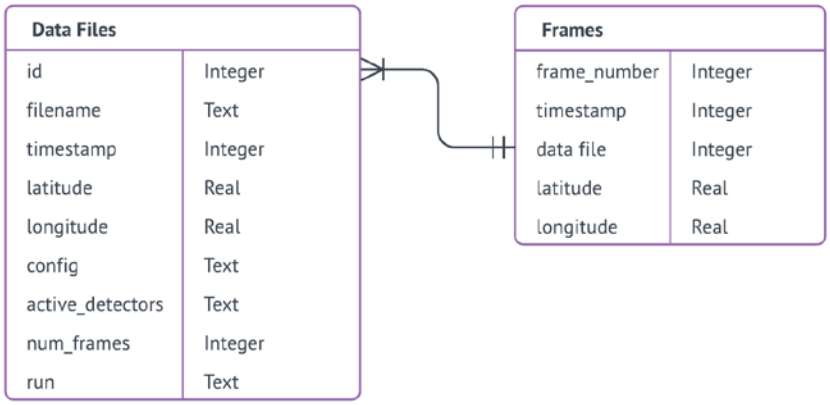



Machine Learning Models
Experimenting with various machine learning techniques
Training several machine learning classifier models
Selection of most appropriate models
Testing all created classifiers and comparing them
Evaluate process and final outcome
Data API
Parsing JSON from the API and any other processing such as creating images.
Web Server
User loads home page
User loads data files page
User loads maps page
User loads tracker page
User loads documentation page
Loads latest frames
Allow user to select data run and data file
Allow user to select data run and data file
Load TLE of satellite and display position
Display documentation
Displays frame images and links
Search by location options
Display files or frames on the map
Analyses frames using machine learning model from the API
Show radiation map overlays on the map
LUCID Dashboard Client

Documented Design

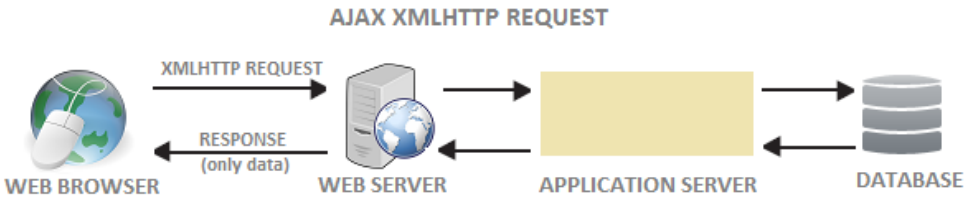
Data Collection

Database Model



This is the SQLite database schema for the storage of LUCID data. The primary key in the Data Files table is a foreign key in the Frames table used to access a specific frame from a given data file. This will allow easy loading of data with any amount of given detail.

API Model



The API endpoint will simply be a server-side file on the application server that will query the database and return data based on the AJAX request sent to it.

The API can be written in either server-side Python, PHP or NodeJS. For the purpose of creating the other web applications, PHP would be most suitable as it can be seamlessly integrated into the other applications. NodeJS requires extra preparation and configuration so it is the least suitable. A Python library for accessing LUCID data called lucid_utils already exists however it is not as efficient over the web.

Get Data Files:

[0]

id	533354612
timestamp	1430842330
latitude	40.3133888888889
longitude	82.0262222222222
config	0129
run	2015-05-04
num_frames	236
active_detectors	0,1,3

[1]

id	533354658
timestamp	1430883730
latitude	71.89675
longitude	-110.008055555556
config	0129
run	2015-05-04
num_frames	216
active_detectors	0,1,3

Get Data Frames by Data File ID:

[0]

timestamp	1430842330
latitude	40.3133888888889
longitude	82.0262222222222
number	1

channels

0	
1	
3	254 0 256.0

[1]

timestamp	1430842331
latitude	40.3740833333333
longitude	82.0065833333333
number	2

channels

0	
1	
3	254 0 256.0

The channels variable holds the serialised version of the XYC frame for each chip detector.

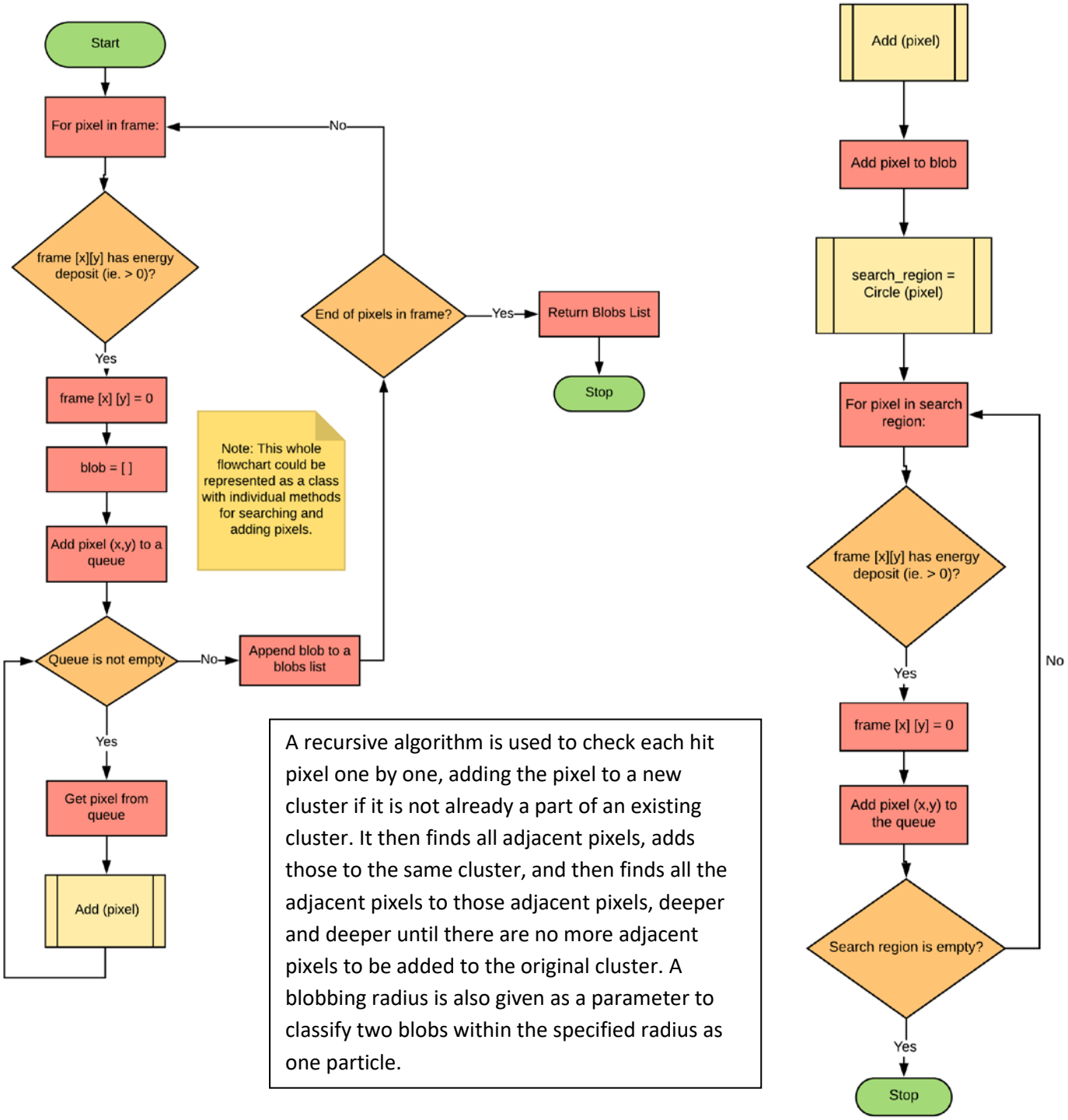
Data Analysis

To gain valuable insight, the data needs to be processed in a way by which useful data can be extracted from the raw format and the vast amounts of data. The main use of LUCID data is for the exploration of particle physics using particle analysis. The existing data analysis algorithm is currently used by several entities including CERN@school and the TAPAS (Timepix Analysis Platform at School) web application by IRIS. The aim of this section is to identify the significant parts of the current analysis algorithm so that the new algorithm is more accurate.

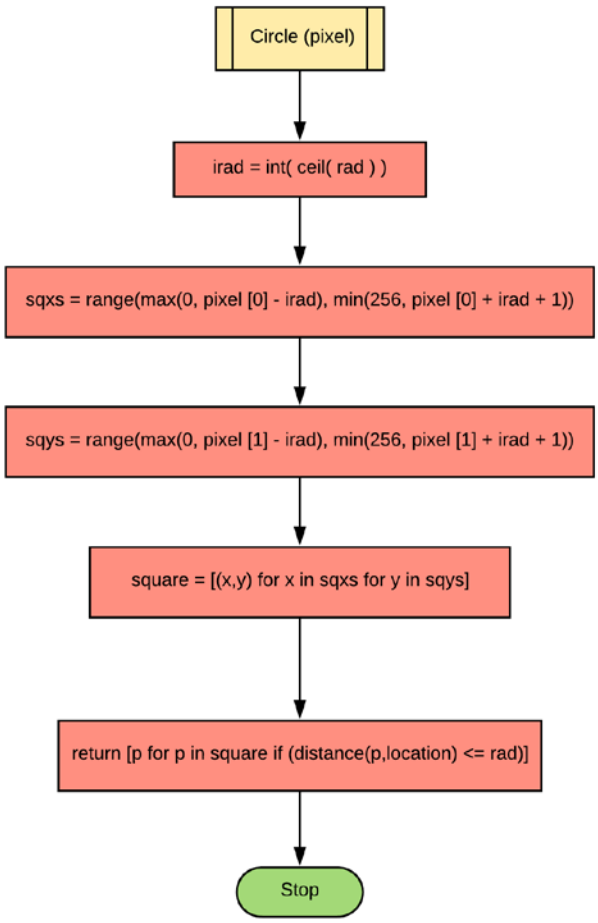
Blobbing (Observation)

Blobbing

To analyse the frames of LUCID data, it is necessary to identify the individual particles in a data frame. To accomplish this, the blobbing program simply takes a 256x256 NumPy array (representing a frame) as a parameter and returns a list of lists of tuples where each sub-list is a blob and the tuples it contains are the coordinates of every single pixel within that blob.



A recursive algorithm is used to check each hit pixel one by one, adding the pixel to a new cluster if it is not already a part of an existing cluster. It then finds all adjacent pixels, adds those to the same cluster, and then finds all the adjacent pixels to those adjacent pixels, deeper and deeper until there are no more adjacent pixels to be added to the original cluster. A blobbing radius is also given as a parameter to classify two blobs within the specified radius as one particle.



Creates a bounding square to search pixels around a given pixel.

Generates list of pixels to search within the given square

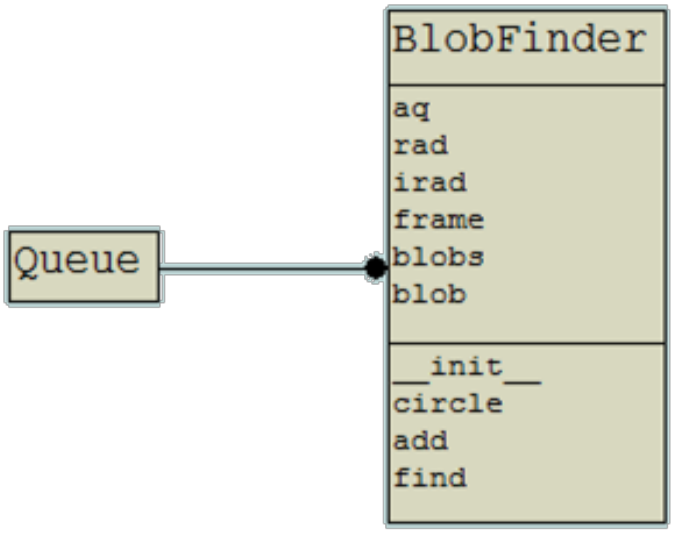
Check if each generated pixel is within a given radius (making this a circular search). (For calculating the Euclidean distance - use Pythagoras' Formula) .

The radius attribute of the class is the radius within which to allow a pixel to be considered a single particle.

The possible x values to search are enumerated and the possible y values to search are enumerated. These values are based upon the single pixel parameter provided to the circle method.

The enumerated x and y values are then used to generate a list of tuples that hold the possible pixel coordinates that are within that given bounding square.

The final part is to only allow coordinates that are within a radial distance instead of within a bounding box.



The **aq** will be the queue of pixels that will be handled by the class. The **rad** variable will be the radius parameter where **irad** is the integer ceiling of the radius. The **frame** stores the whole frame that is being analysed. The **blobs** list is an empty array that will later be filled with calculated blobs. **Blob** is simply a variable to reference the blob in the class.

The methods within the class include all the ones shown in the flowchart and all of them function exactly as mentioned in the flowchart. The main function which finds the blobs in a frame is **find**. Calling this method on a frame will result in a list of blobs that can be iterated through.

Algorithm Pseudocode

```
## Matrix of size 256 x 256
frame = matrix[256][256]
blobs = []
## Queue Object
queue = Queue()

def circle(pixel):
    return list of pixels within circle with centre(pixel)
for x -> 0 to 255:
     for y -> 0 to 255:
          ## if pixel has energy
          if frame[x][y] > 0:
              ## set pixel value to 0
              frame[x][y] = 0
              blob = []
              ## add pixel to queue
              queue.put((x,y))

              ## repeat till queue is empty
              while not queue.empty():
                  ## get pixel from queue
                  pixel = queue.get()
                  ## add pixel to cluster array
                  blob.append(pixel)

                  ## add pixels in the surrounding region to the queue
                  close_region = circle(pixel)
                  for x1,y1 in close_region:
                      if frame[x1][y1] > 0:
                          frame[x1][y1] = 0
                          queue.put((x1,y1))

              ## once queue is empty, a cluster has been fully defined
              ## add cluster to list of clusters
              blobs.append(blob)
```

Feature Extraction (Interpretation)

Common

To interpret and extract the key features of the clusters, it was necessary to calculate some metrics for the particle tracks in a way that would help to determine the properties of the particles.

The following metrics are calculated from the cluster that is provided as a list of pixels to the program:

Number of Pixels → This is calculated as the length of the pixels list.

Centroid → This is calculated using the **find centroid** function.

Radius → This is calculated using the **calculate radius** function.

Diameter → This is twice the calculated radius.

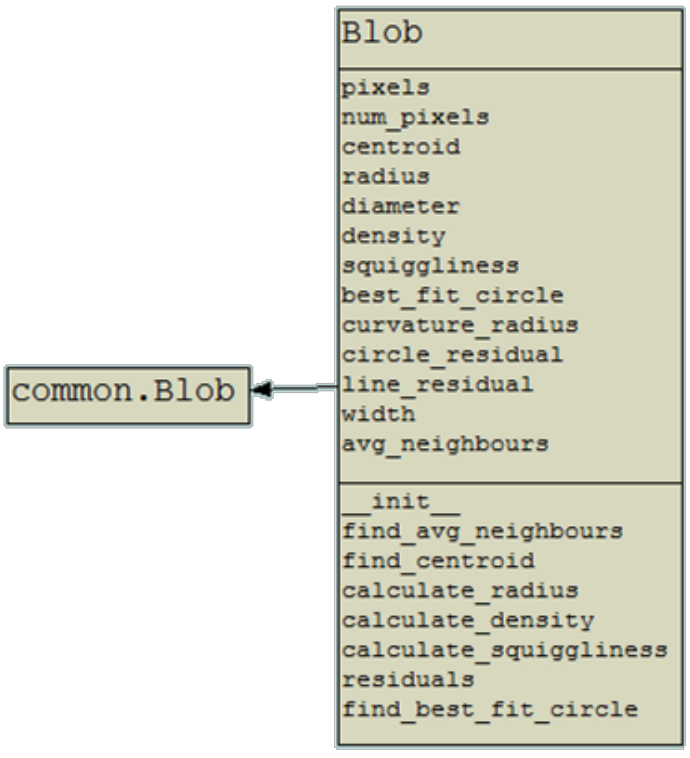
Density → This is calculated using the **calculate density** function.

Line Residual, Best Fit Theta → This is calculated using the **calculate non-linearity ("squiggliness")** function.

Centre of Circle, Curvature Radius, Circle Residual → This is calculated using the **find best fit circle** function.

Average Neighbours → This is calculated using the **find average neighbours** function.

Width → This is the number of pixels divided by the diameter provided that the number of pixels is not 1. For single pixels, the width is set to 0.



This class diagram shows the methods and attributes of the Blob class written in Python. The class is a part of common.py.

The different variables that have been shown here are each individual metrics that will form an input into the neural network.

These are the methods that are a part of the blob class and the mathematics behind them will be explained further on.

Find Average Neighbours

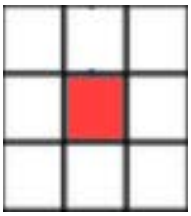


Figure 11 - Average Neighbours

For each individual pixel (red pixel in Figure) in the cluster, the number of neighbouring pixels is counted by iterating through the surrounding pixel coordinates (all the white pixels in Figure 11) and checking if they exist in the cluster. The pixels that exist are then stored in a list as these are the ones that have an energy deposit. The mean of the list is then calculated and the function returns the average number of neighbours.

Find Centroid – (Centroid)

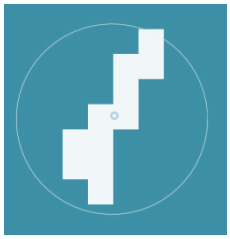


Figure 12 - Centroid

The centroid of the blob is the centre of the particle track. This centroid is found by calculating the mean of the x values and the mean of the y values of the pixels in the blob. The mean x-value is the presumed to be the x-coordinate of the centre point and the mean y-value is presumed to be the y-coordinate of the centre point. This coordinate is not the centre of the smallest enclosing circle but in fact, it is the centre of mass with no weighting on energy values (shown in Figure 12).

Calculate Radius – (Radius & Diameter)

$$Radius = \sqrt{(x_c - x_n)^2 + (y_c - y_n)^2}$$

The Euclidean distance between the coordinates of each pixel to the centroid is calculated. The radius is set to the highest distance from the centroid. The diameter is then simply twice the calculated radius.

Calculate Density – (Density)

$$Density = \frac{Pixel\ Count}{\pi r^2}$$

The density can be greater than 1 as the blob's radius passes through the centre of the outer pixels rather than around them. If the blob is only one pixel in size, then the density is by default set to 1.

Calculate Non-Linearity – (Line Residual & Best Fit Theta)

This function returns angle theta anticlockwise from the x-axis, with the line passing through the cluster centroid as well as the line residual. First, all the coordinates of the pixels within the cluster are split into separate lists of X and Y values. Single pixel gammas are by default given an angle and line residual of 0 as these are completely linear.

$$b = \frac{\sum XY - \frac{\sum X \sum Y}{n}}{\sum X^2 - \frac{(\sum X)^2}{n}}, a = \bar{Y} - b\bar{X}\ where\ \bar{X} = \frac{\sum X}{n}\ \&\ \bar{Y} = \frac{\sum Y}{n}$$

For all other clusters, the above least squares regression function is used to calculate the line of best fit in the form $y = a + bx$. From the line of best fit, the best-fit angle θ (anticlockwise from the x-axis) of the line-of-best-fit can be calculated using simple trigonometry.

$$Line\ Residual = \sum_{n=0}^{Pixel\ Count} (x_n sin\theta - y_n cos\theta - x_c sin\theta + y_c cos\theta)^2$$

This best fit angle is used to calculate the line residual value (also known as the "squiggliness" of the particle track) using the equation above. The line residual value is calculated by the sum of the square of the distance from each pixel coordinate to the line travelling through the centroid where the line is calculated using the best fit angle θ. The equation above shows the simplest form of the line residual where (x_c, y_c) is the coordinate of the centroid and $(x_n, y_n)$is the coordinate of the pixels in the cluster where n is the index of the pixel in the cluster.

Find Best Fit Circle – (Curvature Radius & Circle Residual)

This function is used to perform circle regression however for single pixel gamma particles, this cannot be done and therefore single pixels are by default given values of zero.

As the cluster is a very poor initial guess for the centre of the circle, multiple test circles are generated using the calculated best fit angle from the regression line calculation.

For each of these test circles, the program performs circle regression on the cluster using least squares. The Euclidean distance between the data points and the mean circle centred at (x_c, y_c) is calculated. The mean distance is then subtracted from all of the calculated distances. This value is optimised using the least squares function. The optimisation process returns an optimised centre point.

This optimised centre point is used to calculate the distance of all the points from the centre of the circle once again. The mean distance is set as the new radius and the residual is calculated by the summation of the squares of the difference between the distance from each point to the optimised centre point and the mean radius.

$$(Mean\ Radius)\ R_\mu = \frac{\sqrt{(x_c - x_n)^2 + (y_c - y_n)^2}}{n}$$

$$Circle\ Residual = \sum_{n=0}^{Pixel\ Count} (\sqrt{(x_c - x_n)^2 + (y_c - y_n)^2} - R_\mu)^2$$

The optimised test circles are then organised in order of the magnitude of the residual. The aim of the optimisation is to reduce the circle residual as much as possible and therefore the circle with the least residual is chosen as the best fit circle.

Classification (Decision)

Method 1 – Current Analysis System

```
if num_pixels < 4:
    return 'gamma'
if num_pixels < 8:
    return 'beta'
if (density > 0.75 and num_pixels > 11) or (avg_neighbours > 6 and curvature_radius > 8):
    return 'alpha'
if (line_residual / radius) < 0.1 and radius > 40:
    return 'muon'
if ((curvature_radius > 50 or line_residual < circle_residual) and width > 1.5) or avg_neighbours
> 3.5:
    return 'proton'
else:
    return 'beta'
```

The simplicity of the **Method 1** allows it to define alpha, beta and gamma particles with a high degree of accuracy. However, it also means that occasionally protons and MIPs are misclassified as beta particles. Moreover, as there is no option for unknown particles, any unrecognised particle is automatically classified as a beta particle. The limitations of the current analysis algorithm are clearly outlined above hence suggesting the need for a more robust flexible analysis algorithm that will be future-proof and will account for any new particle classes. This algorithm however is highly inaccurate and does not account for many of the factors involved in producing the specific particle tracks.

Method 2 – Proposed Analysis System

Method 2 was to use machine learning for data classification. So when given another dataset, the program will be able to deduce the type of particles within the test data. There are many possible improved algorithms therefore before choosing a superior algorithm, it is important to understand the basics of machine learning.

There are two types of machine learning: supervised and unsupervised learning. Supervised learning uses labelled data for classification and regression whereas unsupervised learning uses unlabelled data for clustering, dimensionality reduction and association rule learning.

As unsupervised learning is not the most appropriate for classification, it is best to use supervised learning to ensure maximum accuracy for the data that has already been collected. For supervised learning, the metric calculations are rather useful as these provide useful data about the geometrics of the particle track. These extracted features can be used to determine the class of the particles in the test dataset. It would also be possible to use the raw images of each blob with their corresponding labels to train a classifier to accurately predict the class.

The most suitable machine learning method to solve this task was a neural network as the hyperparameters provide the flexibility for continual modification and the various architectures will allow me to explore the field of artificial intelligence. However, I will also be using other supervised learning techniques as a method of evaluating the outcome of the development of my neural networks.

Method 2 has many more advantages over **Method 1** and they have been clearly explained above. Therefore, the next step of design was to generate the training data for the machine learning classifiers.

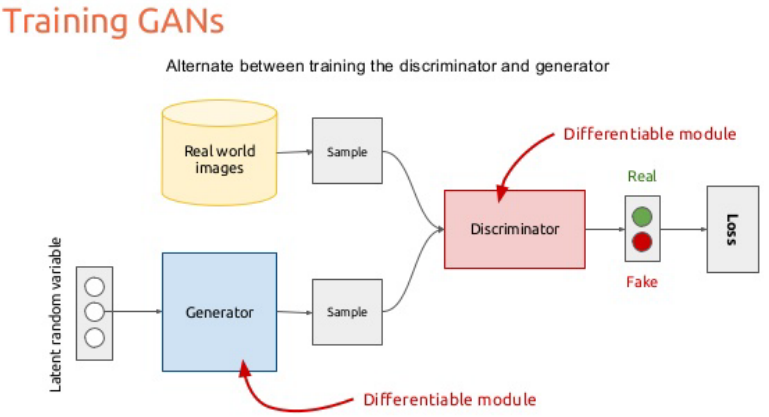
LUCID Neural Trainer

Method Selection

To produce the training data for the neural network, I had several options:

1. Use LUCID data with a platform (like Zooniverse [24]) for user input classification ("crowd" science or citizen science) to create a database matching particle tracks to the classification provided by users. The advantage of this is that it would use real-life data with human predictions for classification. The disadvantages are that human opinions can be diverse in debatable situations and therefore the classifications made from multiple people can lead to inaccuracies in the training data. Moreover, this method requires a lot of time as it has to be done manually and cannot be done computationally.

2. Generative modelling of different particle tracks using Generative Adversarial Networks (GANs) (Figure 13) [25]. So providing the GAN with several alpha particle tracks would allow us to generate numerous alpha particle tracks with slight variances resembling the real life distribution. This can then be used as the training data for our classification neural network. The disadvantages to this method are that the GAN can require a lot of training to provide the reliable training data needed for the classification neural network and that the generated distribution of data is not always fully accurate to the real-life distribution.



The generator generates data that the discriminator validates. The generator improves itself using the loss output via backpropagation. The real life data is entered into the discriminator to train the discriminator to learn the features that make the data valid.

Figure 13 - Generative Adversarial Network

3. Using data frames created by a human as training data would provide the classification neural network with accurate but minimal data. The limitations of this also includes the fact that human-made particle tracks tend to be almost perfect to the expected tracks and therefore the neural network may fail in recognising more complex tracks. This method takes the longest time by far to generate any substantial or useful training data.

For this part of the project, I decided to use a custom-built platform to collect user classifications for a large quantity of particle tracks since this was the quickest method and required the least data processing. I have recognised and considered the drawbacks of this method as well as the other methods and my conclusion was to develop my neural classification model in the most efficient and quickest way possible.
A platform like Zooniverse is not suitable as there is simply too much LUCID data for it to be manually uploaded and edited into a questionnaire for users to fill in. Instead, the simplest method was to create a custom web application that would dynamically generate the questionnaire and ask users for their classification response. Moreover, the Zooniverse project builder was far more complex than necessary for generating some training data. A custom platform provided the flexibility and the mobility needed to generate the training data quickly and efficiently.

Database Design

To store the training data, I had to consider the type of storage and the format of the data. The simplest method was to use a SQLite database as it required the least preliminaries in setting up.

The training data mainly needs to contain the blob data with the corresponding user classification, which will be used for training the neural network. There were several methods for storing the data in an SQLite database. The following schematics are shown below:

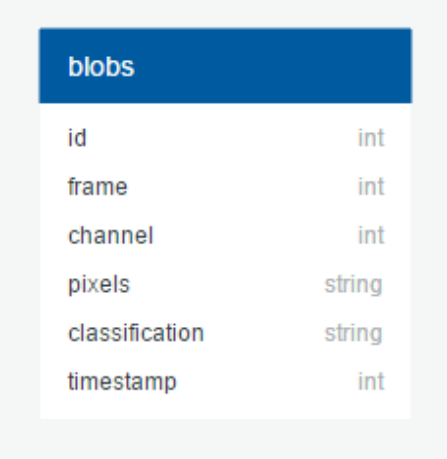


Figure 14 - DB Schema 1

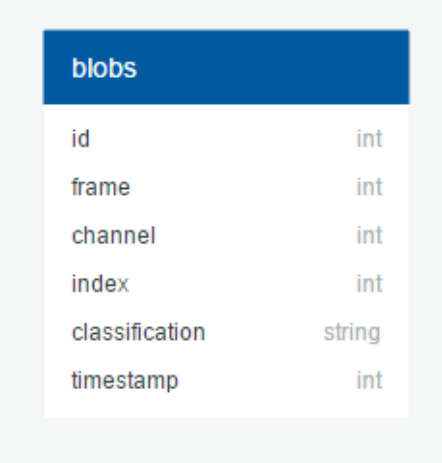


Figure 15 - DB Schema 2

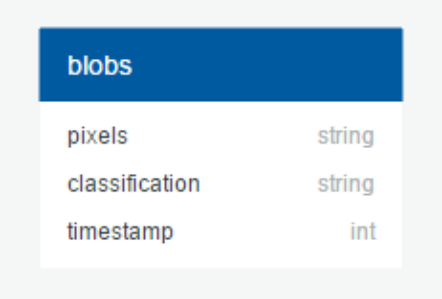


Figure 16 - DB Schema 3

DB Schema 1

Advantages

- Data can be reaccessed and checked for redundancies using id, frame and channel.
- The data will never be redundant as the pixels themselves are stored.
- The pixels can be updated when the blobbing algorithm is changed.
- The data can be checked for duplicates.

Disadvantages

- It uses the most storage of all the schematics.

DB Schema 2

Advantages

- Data can be reaccessed and checked for redundancies using id, frame and channel.
- It uses the least storage as the index of the blob is stored not the blob itself.
- The data can be checked for duplicates.

Disadvantages

- The database becomes redundant if the blobbing algorithm is changed.
- Accessing the blob during the training process would be very slow.

DB Schema 3

Advantages

- Uses less storage than DB Schema 1

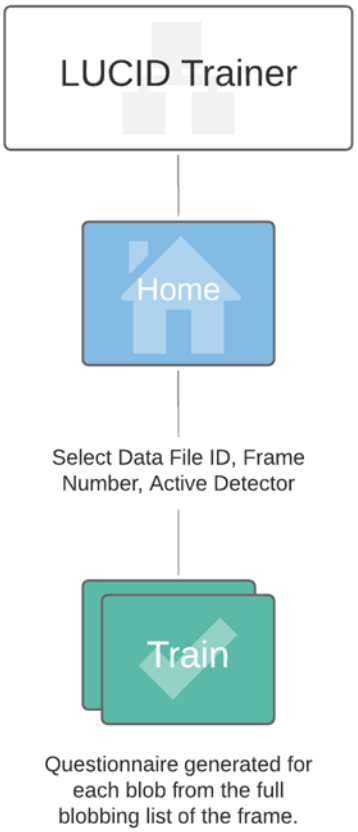
Disadvantages

- The data cannot be rechecked in the original frame and therefore cannot be updated or checked for duplicates.
- It still uses a lot of storage for busy frames.

Conclusion

From the above evaluations, I decided to use DB Schema 1 as the only disadvantage it had was that the storage space would be too large for a large amount of data. Given that a large storage space does not necessarily affect my investigation in any way, this method is most appropriate for storing the training data.

Sitemap



The home page should ask the user which data file to analyse and then upon selection, it will request the frame number and the active detector chip. Once the user has selected which frame to analyse, the user will be taken to the training page.

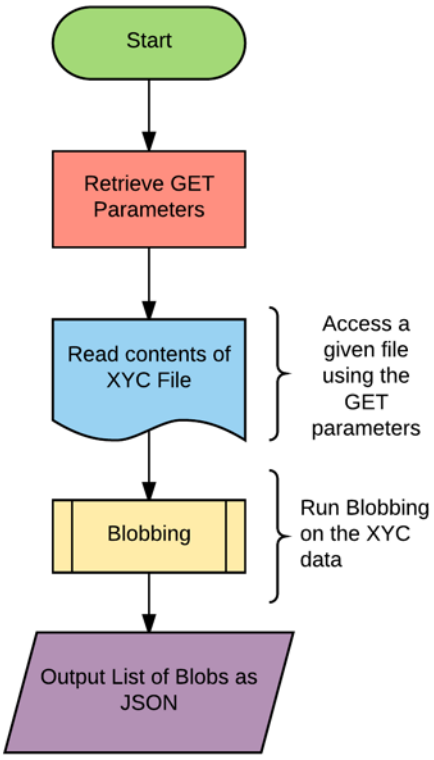
The training page will run the blobbing algorithm on the frame using the blobbing API generating a list of blobs. The user is then presented with the first questionnaire for the first blob in the list.
This page will have the following GET parameters:
`{"id":0123456789, "frame":0, "channel":0, "index":0}`

All of the data for this software will be accessed using the data API to ease development as well as allowing stability to future revisions.

Utility Functions

These are all the fundamental functions that will be needed to create the training web application.

Blobbing API (Python)



As the original python code for the blobbing algorithm is written in Python for data analysis using scientific libraries, the best way to allow a web interface was to create a REST API in Python with CGI (Common Gateway Interface) enabled for sending the data to a web browser or an application.

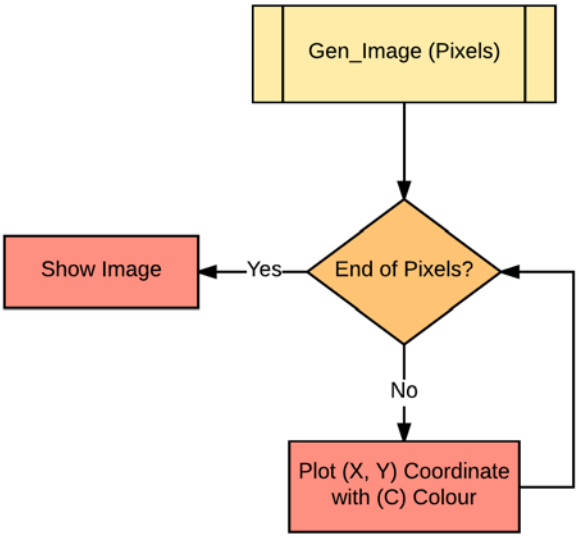
The list of blobs generated from the given GET parameters can then be iterated through in PHP and each one can be outputted as an image that the user can classify in a questionnaire.

The blobbing API should return a 2D array: An array of blobs each containing arrays of coordinates (X,Y,C) that make up that blob:

Blobs -> Blob -> [(x,y,c), (x,y,c), (x,y,c)]
Example: **`[[[7, 58]], [[126, 153]], [[128, 151], [129, 151], [130, 150], [130, 149]]]`**

The example above shows three blobs: two blobs with only one pixel and one blob with four pixels.

XYC to Image (PHP)

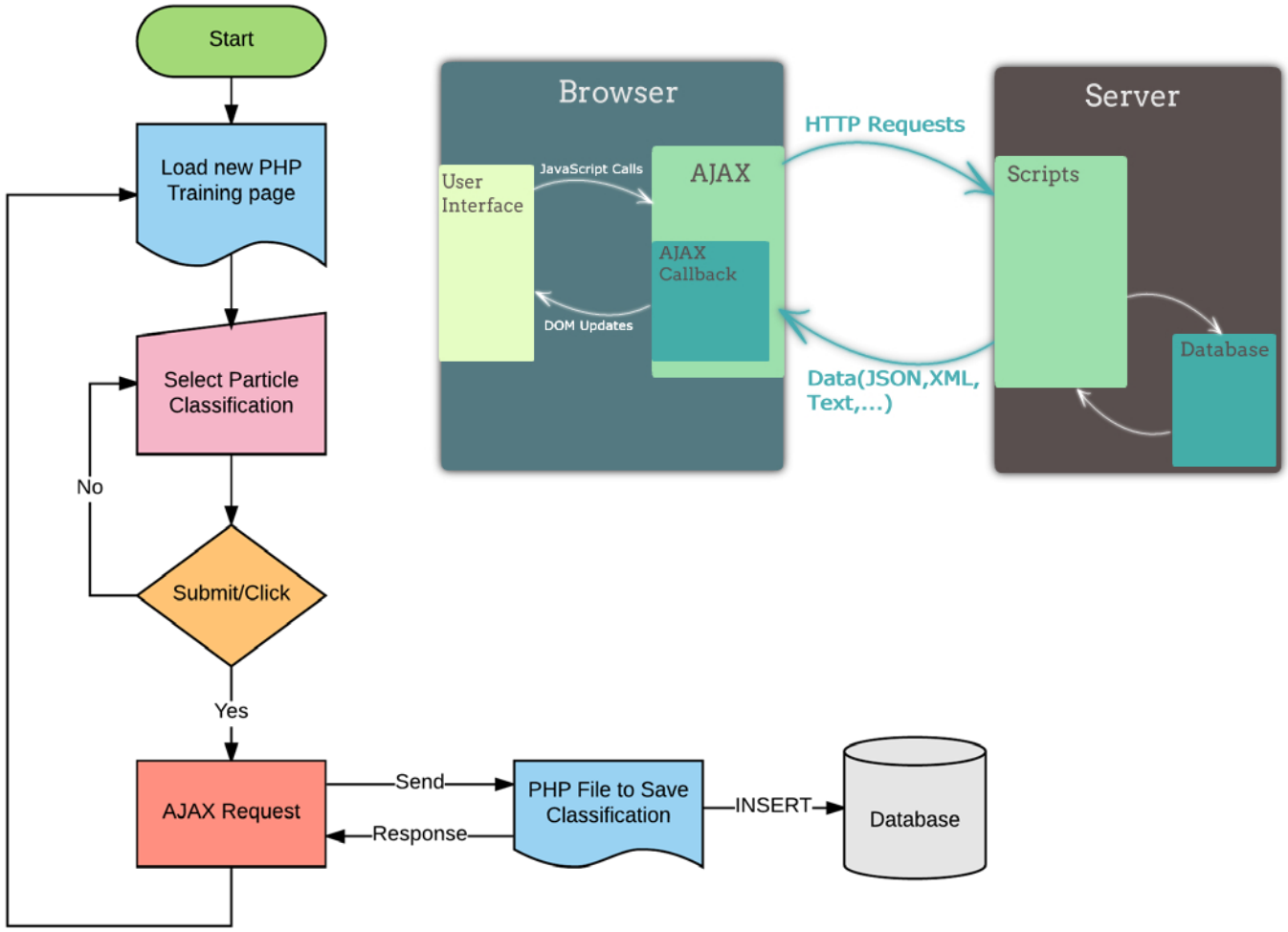


To display the blobs conveniently to the user, the XYC format had to be converted into an image which the user could view to classify the particle in the frame.

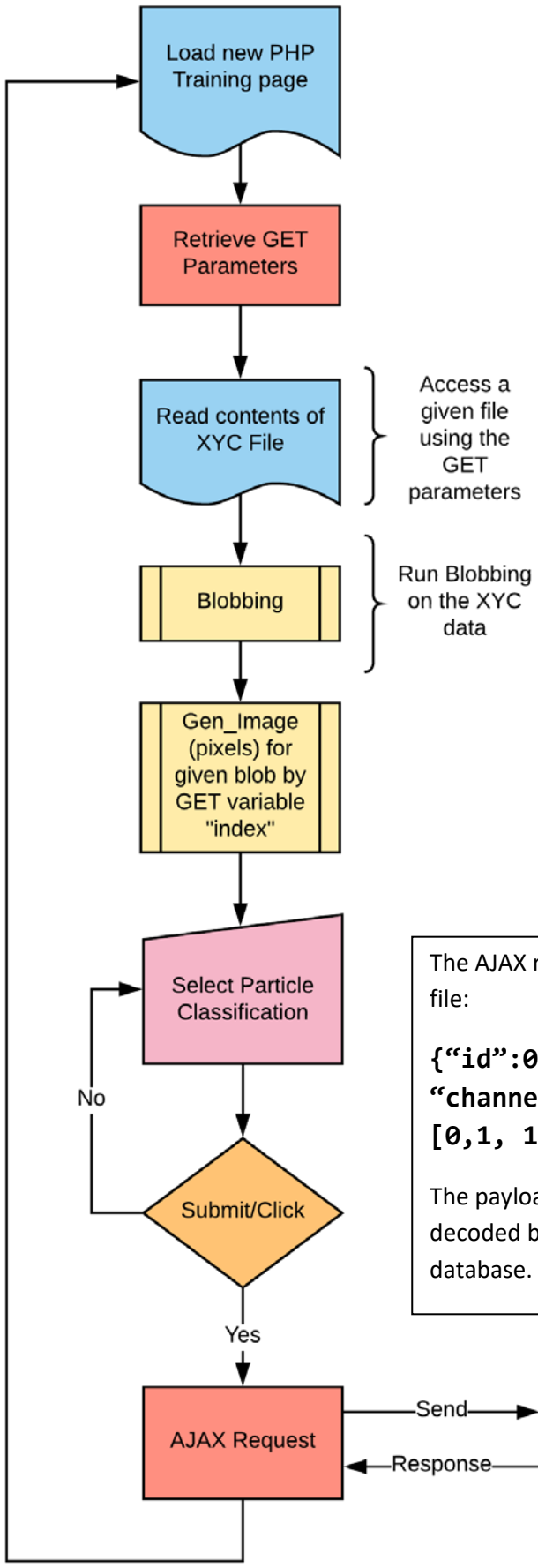
This function could have been performed using client-side JavaScript or server-side Python or PHP. However, it was much more efficient to generate the image in PHP as JavaScript image computation would slow down the web application and server-side Python would require more configuration to integrate it into the PHP web application.

Classification Submission (JavaScript)

To submit the classification from the user, the page will need to use AJAX for sending a request to a PHP file that saves the user's classification to the LUCID Trainer database. After the request is sent, a new training page with the next blob will be created.



LUCID Trainer Flowchart



Each training page instance will require four GET parameters that will take default values if user generated ones are not provided. The parameters include id, frame, channel and index. The id, frame and channel are used to access the XYC file while the index is used to access a specific blob in the blobs list generated by the blobbing API.

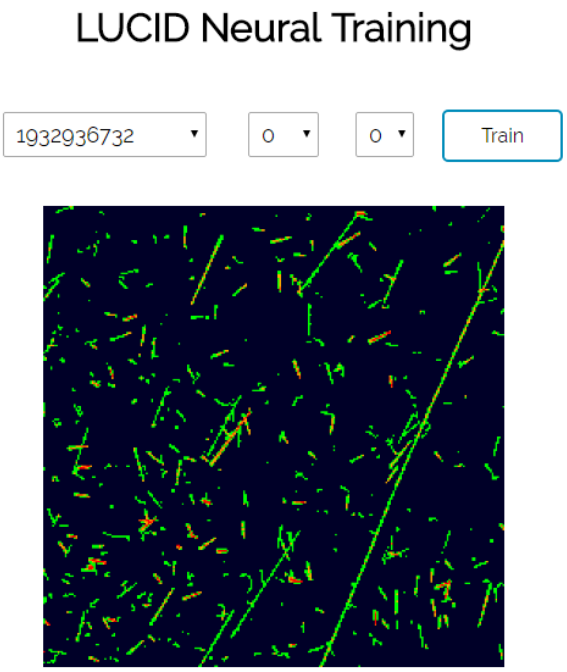
The AJAX request sends the following payload to the PHP file:

```
{“id”:0123456789, “frame”:0,
“channel”:0, “pixels”: [[0,0, 100],
[0,1, 120]], “classification”: “beta”}
```

The payload will be sent as JSON so it will have to be decoded by the PHP file and then stored into the database.

UI Design

As per my design requirements, it was simpler and easier to design the UI first in HTML and CSS and then add in the functions later. Here are screenshots of what the UI will look like:



The home page will be a simple select dropdown which will allow the user to select the data file they want to analyse. The train button once clicked will then go to the training page with the GET parameters being set to the chosen parameters.

The training page will be generated based upon the GET parameters (id, frame, channel, index). The UI is overall very simple to prevent any user confusions and to accomplish the task of generating training data as easily as possible. Here the submit button simply calls a JavaScript function which in turn runs an AJAX request to the server to store the data to a database via a PHP file.

Neural Network Design

Architecture Selection

As a part of my experimentation process, I had constructed all the various neural architectures that I had researched. However, from my preliminary constructions, the most suitable neural architecture to construct was the deep neural network as the data that I had collected was in the most suitable format for training it. The convolutional neural network required the collected data to be pre-processed into images leading to a large single file containing all the image data. This became rather computationally expensive to process during the training of the neural network and the deep residual network simply lacked computing power. Therefore, the main neural architecture I decided to focus on was the deep neural network.

Inputs

Selection

The metrics I selected for the inputs of the neural network included the following:

➢ Number of Pixels ➢ Density ➢ Radius	➢ Curvature Radius ➢ Line Residual ➢ Circle Residual	➢ Width ➢ Average Neighbours

These values are the ones that were synthesised in the feature extraction process. Metrics specific to each individual particle such as the best-fit angle, centroid and centre of circle were not used, as they would not help generalising the class of the particle.

Normalisation

Once the metrics have been calculated from the data, they usually need to be normalised. Normalisation is the process by which the all the data are made to be within a specific range usually between 0 and 1. However, during my development process, I found that it was not necessary depending on the type of activation function that was used as it only resulted in poorer accuracy. It was also very difficult to normalise some of the metric values as they could vary from zero to a very large number.

Output Encoding

Single Output

If we were to have a single output then we would encode the data using single nominal values usually between 0 and 1. However, this nominal value may be interpreted as an ordinal value by a machine-learning algorithm.

```
particle_outputs = {
    "gamma": [0.0],
    "beta": [0.2],
    "muon": [0.4],
    "proton": [0.6],
    "alpha": [0.8],
    "others": [1.0]
}
```

This means that the classes may be treated with respect to the magnitude of their assigned value even though the values are being used as labels. This can alter the output value as the different particles are not spectrally distributed. The advantage of this method is that it is very simple to interpret a single output neuron.

Multiple Outputs

```
particle_outputs = {
    "gamma":    [1, 0, 0, 0, 0, 0],
    "beta":     [0, 1, 0, 0, 0, 0],
    "muon":     [0, 0, 1, 0, 0, 0],
    "proton":   [0, 0, 0, 1, 0, 0],
    "alpha":    [0, 0, 0, 0, 1, 0],
    "others":   [0, 0, 0, 0, 0, 1]
}
```

As single outputs can cause inaccuracies, it is sometimes necessary to use a method called one hot encoding to provide multiple outputs from the neural network. One-hot encoding is used to transform categorical features into a numerical format that works better with classification and regression. It is essentially an array of 0s with one 1 in the array where its position in the array denotes the class. This allows for multi-class classification where each of the neurons in the output layer gives each value in the array.

Hyperparameter Selections

Dropout

Dropout is when one or more neural network nodes will switch off at every training step depending on a provided probability so that it will not interact with the network (its weights cannot be updated, nor affect the learning of the other network nodes). With dropout, the learned weights of the nodes become less sensitive to the weights of the other nodes and become less dependent on the other nodes they are connected to. In general, dropout helps the network to generalize better. It also increases accuracy because dropout decreases the influence of a single node.

Regularisation

Regularisation limits the magnitude of the neural weights by adding an extra cost for weights to the error function. There are two types of regularisation: L1 and L2 regularisation. L1 regularization uses the sum of the absolute values of the weights whereas L2 regularization uses the sum of the squared values of the weights.

Dropout and regularisation are used to prevent overfitting to the training dataset. Although I had implemented dropout rates and regularisation in my program, my program was not overfitting and the test accuracy was not much lower than the training accuracy. This meant that it was not necessary to use dropouts or regularisation for my neural network. Several attempts at these two techniques resulted in a much lower training and test accuracy and led to greater loss.

Training Algorithm

The best training algorithms were the various forms of gradient descent as these were the fastest to compute while providing a high degree of accuracy. After several iterations, the accuracy of the neural network was found to be more dependent on the learning rate than the type of gradient descent algorithm used.
After numerous combinations, the best algorithm was found to be the Adam optimiser with a learning rate of 0.001.

Error Function

The numerous error functions have different derivatives and different purposes. Probabilistically, the cross-entropy (CEE) arises as the natural cost function to use if the sigmoid or softmax activation function is used in the output layer of the neural network. This is used to maximise the likelihood of classifying the input data correctly. If the target is continuous and normally distributed, then using mean-squared (MSE) usually maximises the accuracy.
For classification, cross-entropy tends to be more suitable than mean-squared however, for regression problems mean-squared should be used. As the particle analysis problem is a classification problem, the cross-entropy cost function worked the best. This was evaluated through the process of trial-and-improvement.

Layers & Nodes

There are only one input and output layers in any neural network. The input layer has the same number of nodes as the number of input metrics selected and therefore there are overall 8 inputs into the neural network. The output layer has the same number of nodes as the number of output classes. Since one-hot encoding is being used for the training data of the neural network, there will be multiple nodes for the output layer. From the dataset created using LUCID Trainer, there are 6 different output classes for the different particles the neural network will be classifying.

Activation Functions

The activation functions that were tested were the hyperbolic tangent function, the sigmoid function and the rectified linear function for the hidden layers. After several iterations, the RELUs failed to provide accuracy higher than 80%. This was most likely because the inputs were not normalised and as some of the metric may have been less useful than others when the values were large. The hyperbolic tangent function has values between -1 and 1

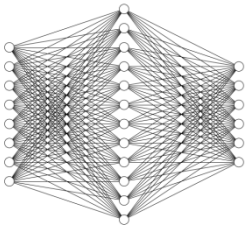
and therefore it also seemed to lack in accuracy. Therefore, the sigmoid function was used as it was able to compensate for the large values due to the "vanishing gradient" and it was in the range 0 to 1.
As the analysis problem is a multi-class classification problem, the softmax function was used for the output layer.

Hidden Layers

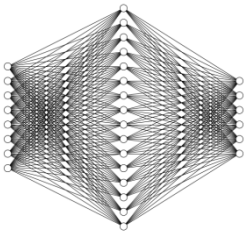
To construct the neural network, it was essential to understand how many hidden layers to have in the neural network. A large number of layers is used for greater learning capacity however this can also result in overfitting as the neural network can end up memorising the training data in the form of the neural weights. Experimenting with the number of hidden layers, showed that a single hidden layer was too shallow for any learning at all and any more than 2 layers resulted in high overfitting.

Number of Nodes

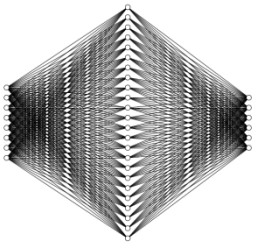
This process was purely trial-and-improvement however, there were some preliminary factors that allowed the decision to be made clearer. As there were more inputs than outputs, it was more appropriate to have a decreasing number of nodes in the hidden layers. This was checked during development as primary tests.
Here are the main neural structures used to form the basis of the final selection. There were many other structures that were tested however due to the iterative nature of the modification process, only the key structures that affected the final decision are shown.



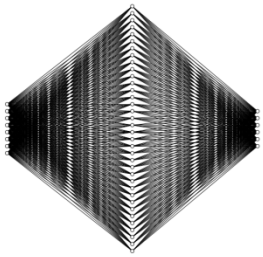
[8, 12, 6]



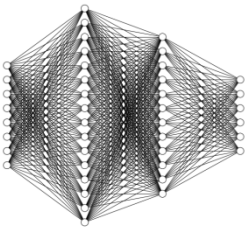
[8, 16, 6]



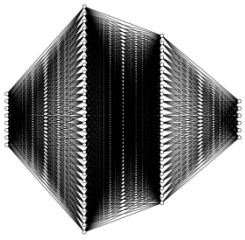
[8, 24, 6]



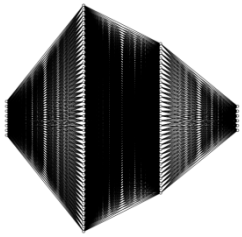
[8, 32, 6]



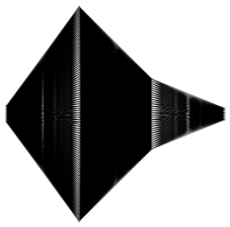
[8, 16, 12, 6]



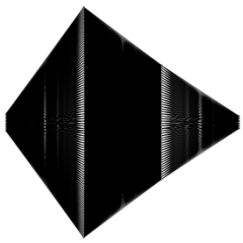
[8, 32, 24, 6]



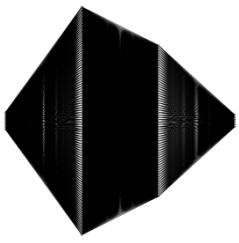
[8, 48, 32, 6]



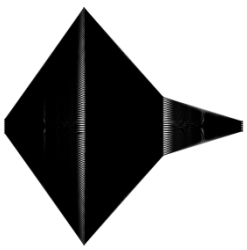
[8, 96, 32, 6]



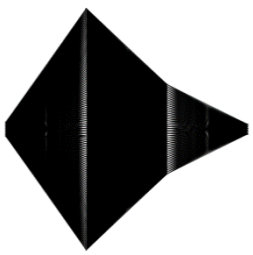
[8, 96, 48, 6]



[8, 96, 64, 6]



[8, 128, 32, 6]



[8, 128, 48, 6]

From the development process, it was noticed that adding any more than 128 nodes in any of the layers resulted in overfitting. Moreover, only 2 hidden layers provided good enough accuracy regardless of the number of nodes. For a greater depth, a low number of nodes simply resulted in the inability to learn while a high number of nodes resulted in overfitting.

Final Neural Network

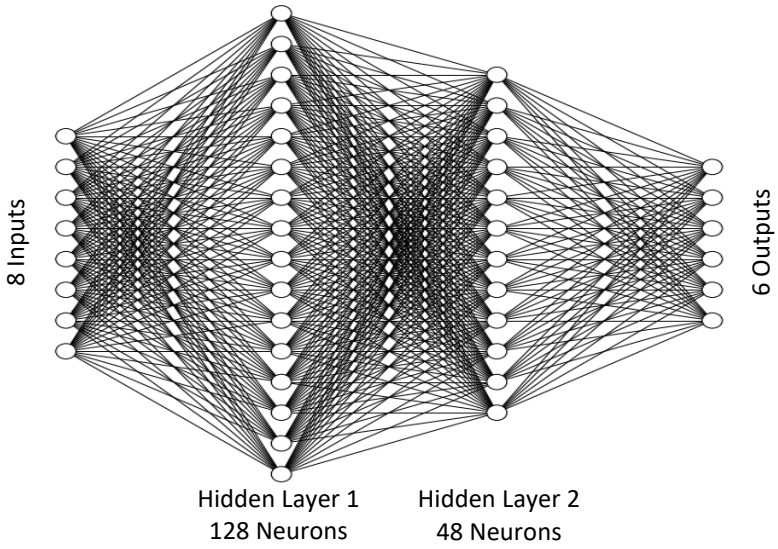
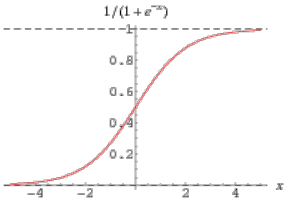


Figure 17 - Final Neural Network

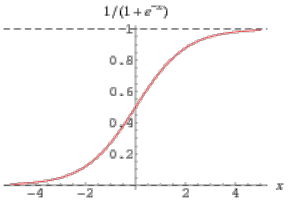
Error! Reference source not found. shows the designed structure of the final neural architecture. **Error! Reference source not found.** shows how the neural network algorithm works. Layers 1 and 2 multiply the weights matrix by the output from the previous layer, the biases are added and then propagated through the sigmoid function. The output layer is the same however, the sigmoid function is replaced with the softmax function.

The following activation functions were used in the final neural network:

Hidden Layer 1 – Sigmoid



Hidden Layer 2 – Sigmoid



Output Layer – Softmax

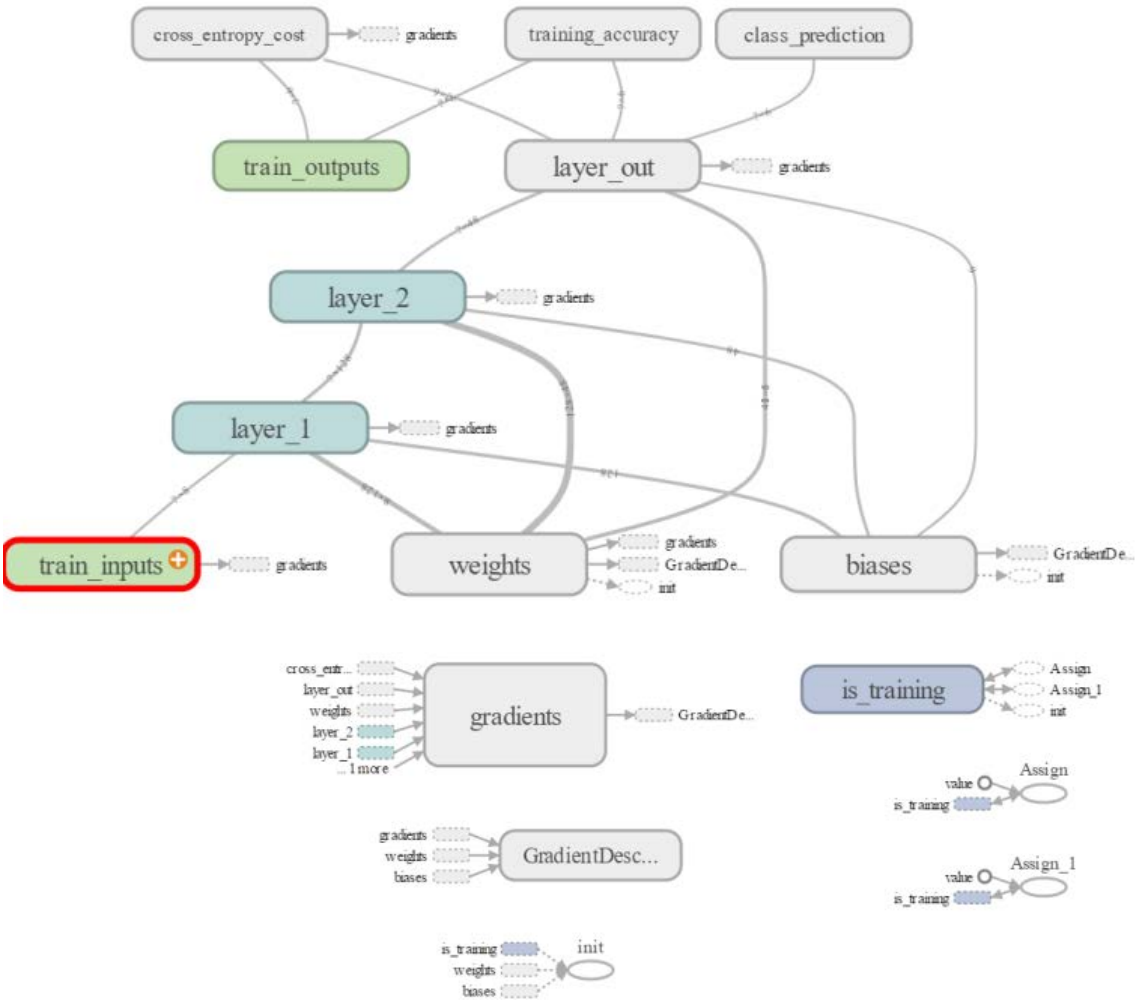
$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{k} e^{z_k}} \; for \; j = 1, \ldots, k$$

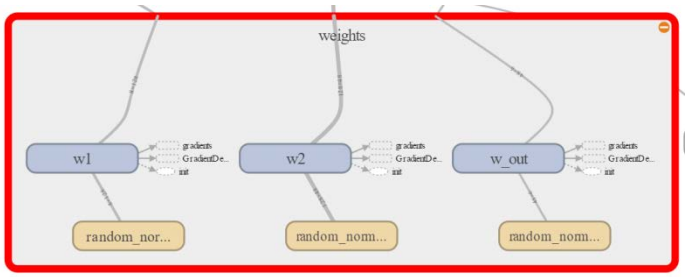
Generalised version of the sigmoid function.

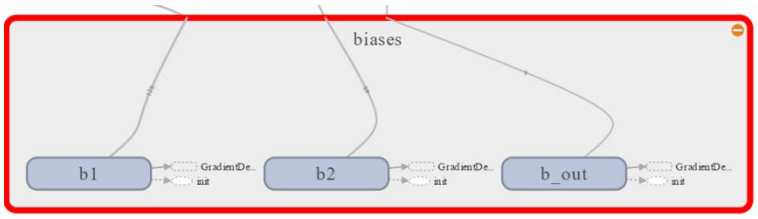
Figure 18 - Meta Graph of Layers

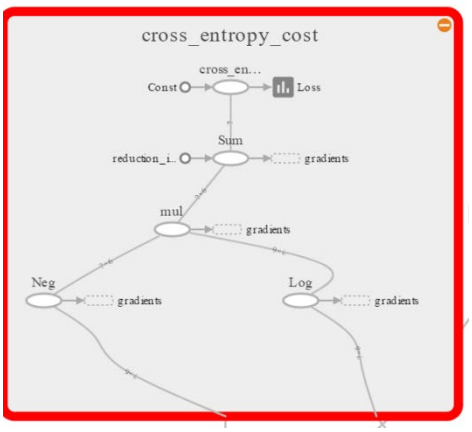
Tensorflow Meta Graphs

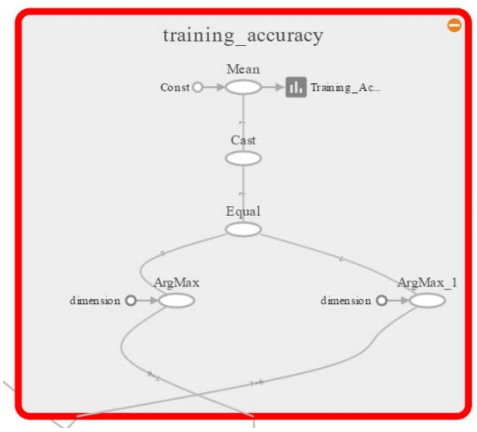
The following meta-graph shows the structure of the whole neural network and how the neural network works in the simplest manner.

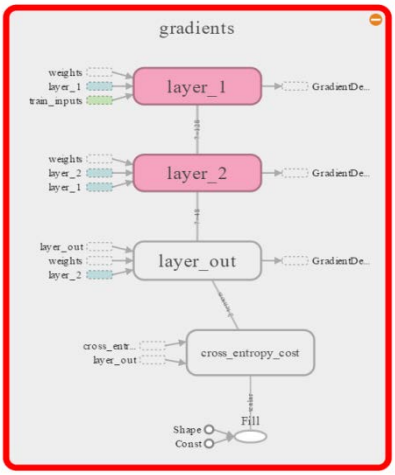


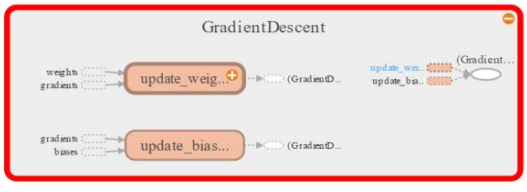




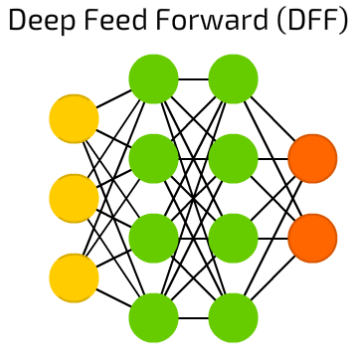








Explanation of the Algorithm



A deep feed forward network is also known as a multi-layer feed forward neural network and it is the basic neural network architecture used to build up other neural networks with far more advanced uses. In a deep feed forward network, there are three types of layers: the input layer, the hidden layers and the output layer. The input layer receives inputs from a training dataset and propagates it to the hidden layers. The neural model is created by adjusting the weights on the connections to the hidden layers by backpropagation until the error of the neural network is minimal for the corresponding outputs of the given dataset. The neural model is stored as a series of weights and can be used again to classify inputs for an unknown output.

Layered Structure

For a neural network with 2 hidden layers with the activation functions being sigmoid function for the hidden layers and then softmax in the output layer generates the following equations:

$$x_i = Input\ at\ Node\ i$$

$$f_j = \sigma\left(w_j \cdot x_i + b_j\right) \quad where \quad \sigma(z) = \frac{1}{1 + e^{-z}}$$

$$f_k = \sigma\left(w_k \cdot f_j + b_k\right) \quad where \quad \sigma(z) = \frac{1}{1 + e^{-z}}$$

$$y_l = \rho(w_l \cdot f_k + b_l) \quad where \quad \rho(z) = \frac{e^{z_k}}{\sum_{p=1}^{N} e^{z_p}} for\ l = 1, \dots, N$$

$$y_l = Output\ at\ Node\ l$$

The above equations intuitively explain how a neural network works.
Every epoch (one run of the training dataset through the neural network), the weights are updated using the error between the actual output and the expected output using an algorithm known as Gradient Descent.

Gradient Descent (GD)

The gradient descent is the simplest algorithm for iteratively finding the minimum point of a function. It is a first-order method meaning that it only uses the first derivative of the function.

One-Dimensional Method

$$x_{n+1} = x_n - \alpha\, f'(x_n)$$

A point x_0 is picked as a starting value and the gradient is calculated for the current value. $g_n = f'(x_n)$ To improve the conjecture, the point needs to move by α (the training rate) in the direction of that gradient:

$$x_{n+1} = x_n - \alpha\, g_n$$

This iteration is repeated until the function is close enough to zero (until $f(x_n) < t$ for some small threshold).

Multi-Dimensional Method

$$x_{n+1} = x_n - \alpha\, \nabla f(x_n)$$

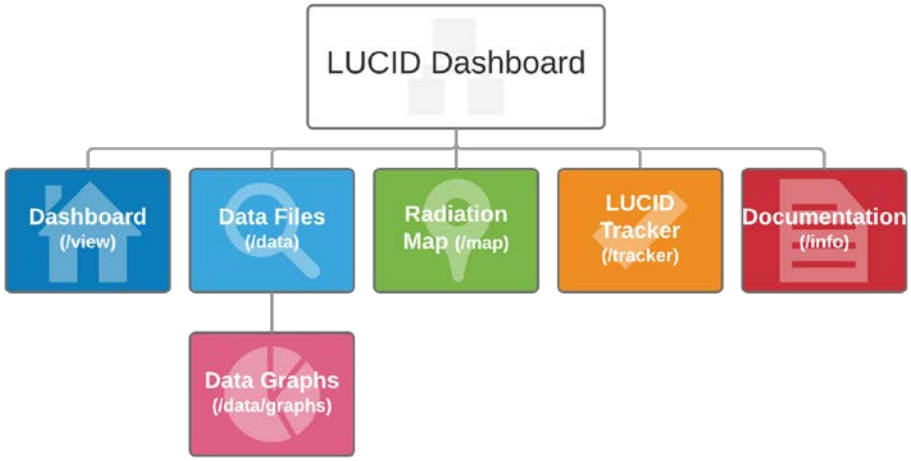
The multi-dimensional method is derived from the one-dimensional method where the first derivative is replaced with the gradient of the original function $f'(x) \to \nabla f(x)$. For both methods, ϵ is a fixed training rate with a small value greater than 0. A point x_0 is picked as a starting value and the gradient is calculated for the current value. $g_n = \nabla f(x_n)$ To improve the conjecture, the point needs to move by α (the learning rate) in the direction of that gradient:

$$x_{n+1} = x_n - \alpha\, g_n$$

This iteration is repeated until the function is close enough to zero (until $f(x_n) < t$ for some small threshold).

LUCID Dashboard

Sitemap



Dashboard:

GET Params: ID, Frame

Function:

This will be the main page where the default page will show the latest frames captured by LUCID.

Data Files:

GET Params: Run, (Optional: Location Search)

Function:

This will be the main method for searching for data files. The main method to search will be by filtering the data run however a more advanced search such as location search might be more preferable for analysis purposes.

Radiation Map:

GET Params: Run (Displays all files in that run on the map) or Run & ID (Display all frames in that file on the map)

Function:

The map will display either frames or files either specific geo-location allowing students to find and analyse frames more easily.

LUCID Tracker:

GET Params: N/A

Function:

LUCID can be tracked via the TLE file and this page will be used for giving lie information on the speed, altitude and location of the satellite.

Documentation:

GET Params: N/A

Function:

This will be a full write-up of all URL formats of the LUCID Dashboard and the API used to create the whole system.

Data Graphs:

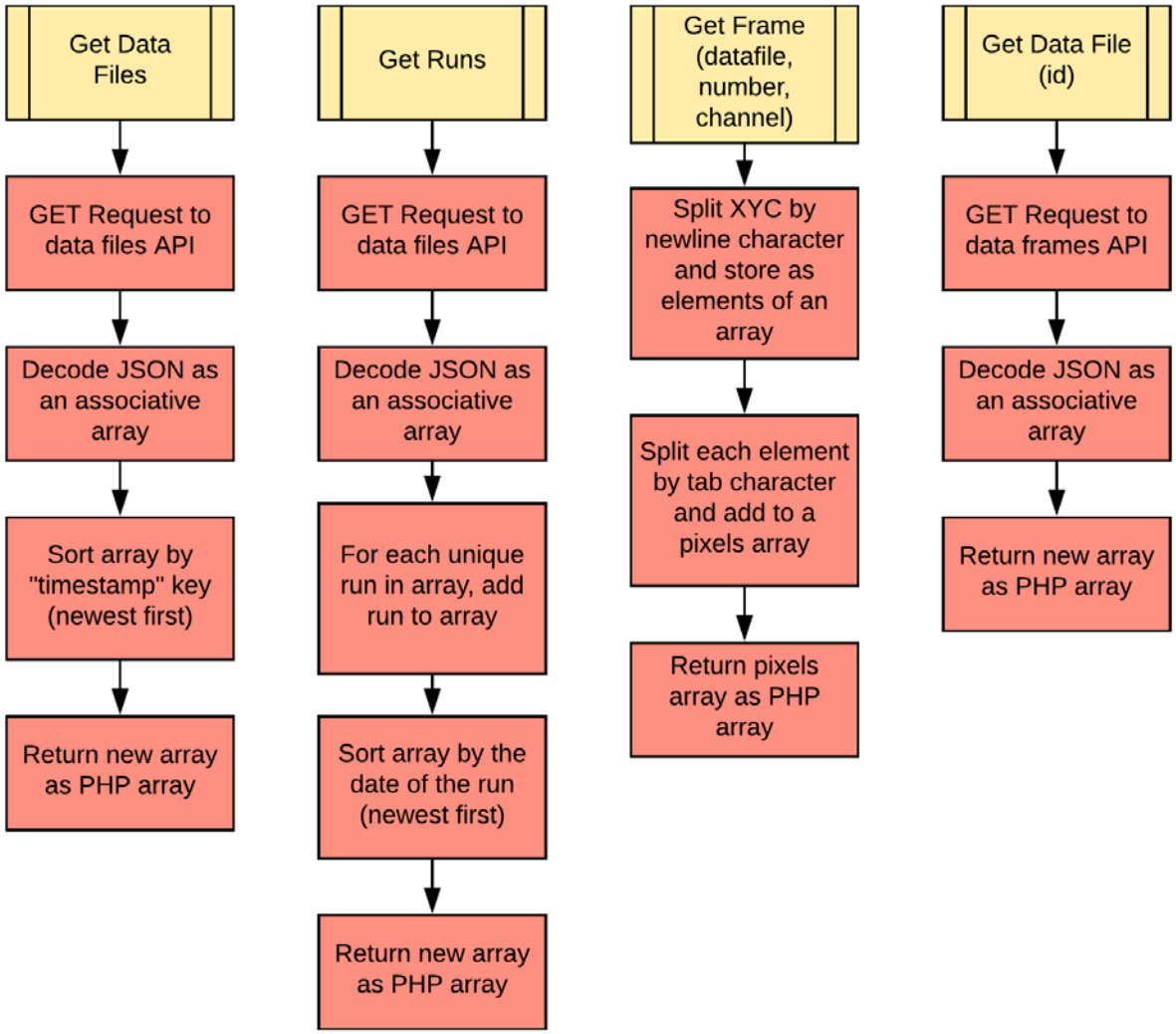
GET Params: Graph ID

Function:

This will show statistics on the amount of LUCID data collected while providing an insight into the quality of the data.

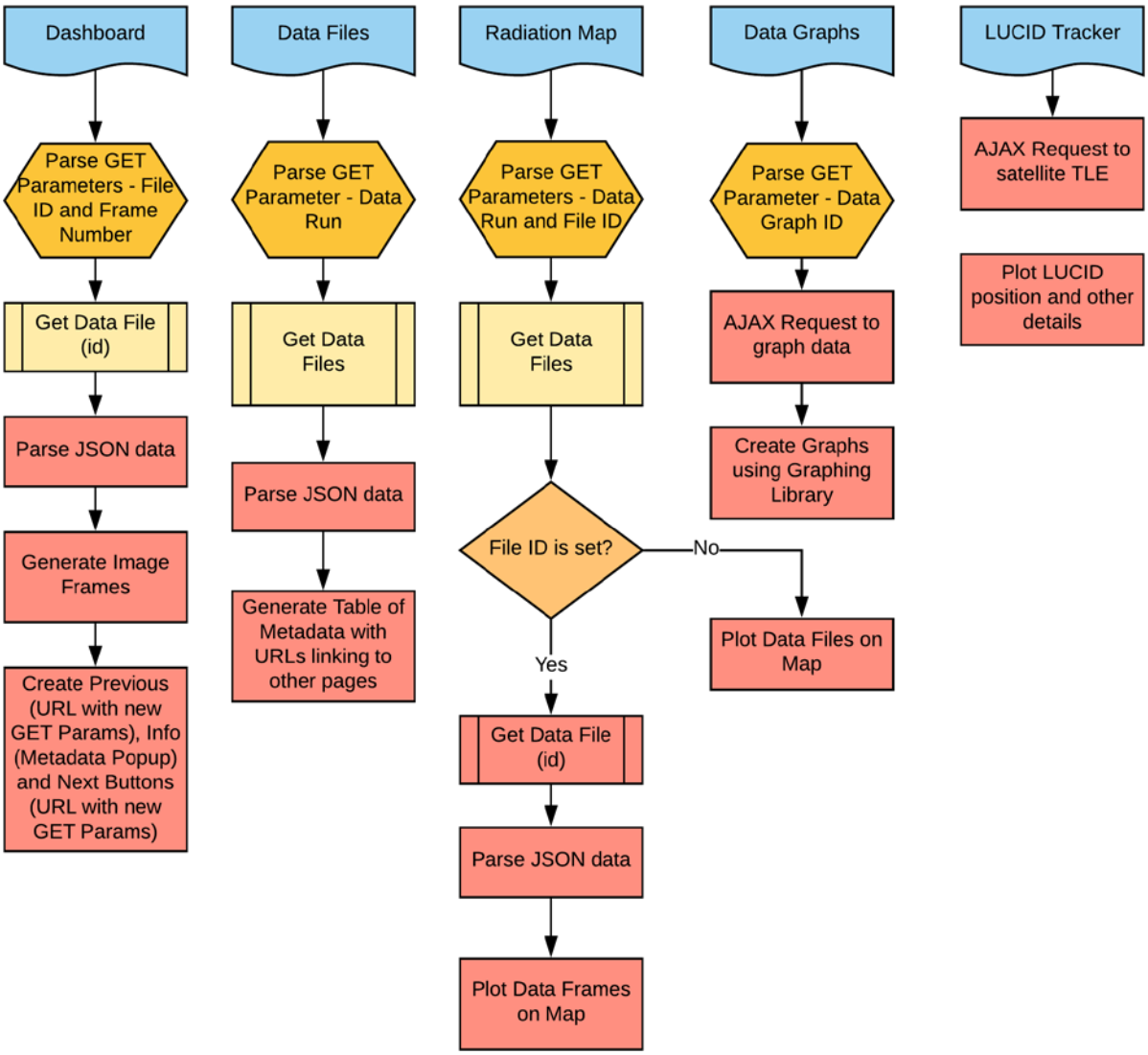
Utility Functions

These are all the fundamental functions that will be needed to create the dashboard web application. However, to improve the functionality of the dashboard many more functions will be added during the development of the technical solution.



These are simply the functions to access and parse the data from the data API. Other functions also include generating the image from the XYC file (shown and explained in the LUCID Trainer section) and geocoding a latitude and longitude into an address. These will form the basic part for creating the dashboard and the entire UI will be built while considering the implementation of these functions to create the web application.

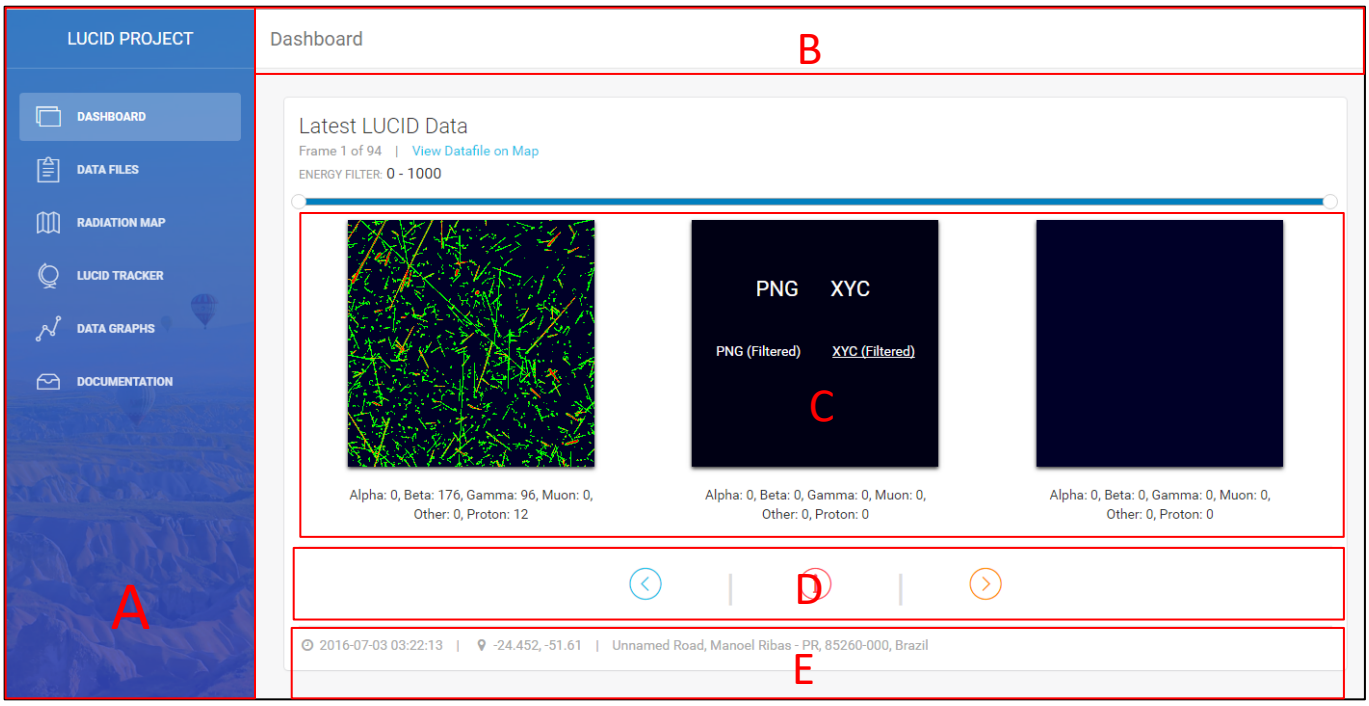
Web Application Flow



Each page of the web application has a different aim relating to fulfilling the project. The idea of the dashboard page is simply a page where frames can be viewed but more importantly new frames are to be shown first. A data file will be picked by the ID and the first few frames in the file will be shown and the user will be able to navigate through the frames. As I will be using PHP, the majority of the processing of the page will be done synchronously on the backend as it will be simpler to program and easier to blend in all the different tools created for LUCID. Any user interface processing such as complex mouse events will be handled by JavaScript while simple clicks for navigation will be best handled by page reloads with different GET parameters. The data files page will again follow a similar processing style but this aims to be as least dynamic as possible as page loading will be slow anyway due to the high amount of metadata download on each page reload. The radiation map page is similar to the dashboard page in the sense that it should be about function as well as aesthetics focussing to improve user experience and function over communicating information. The page parameters on this page will have multiple functions such as filtering and searching and this makes this page have overall a highly complex viewing model. The data graphs page will use client-side processing for displaying the graphs however the data will be loaded via the server side to prevent page crashes due to the high amount of data processing. As the tracker page is a part of the LUCID dashboard, it will therefore be a PHP page however it will be mostly static except loading the modular parts of the page and loading the TLE so that an AJAX request does not have to be performed on every page reload.

UI Design

Dashboard



A) As the web application will be designed in PHP, the side navigation bar will be integrated by using the include function in PHP. The modular-page design will allow easy debugging and a cleaner and neater design. The links to the other pages can also be edited more easily as each page does not have its own sidebar and a single common page allows uniformity during editing.

B) For the same reason as having a modular sidebar, a title bar that can have added features will also be more useful as parameters can be provided to the included page which will customise the bar based on which page it is being added to.

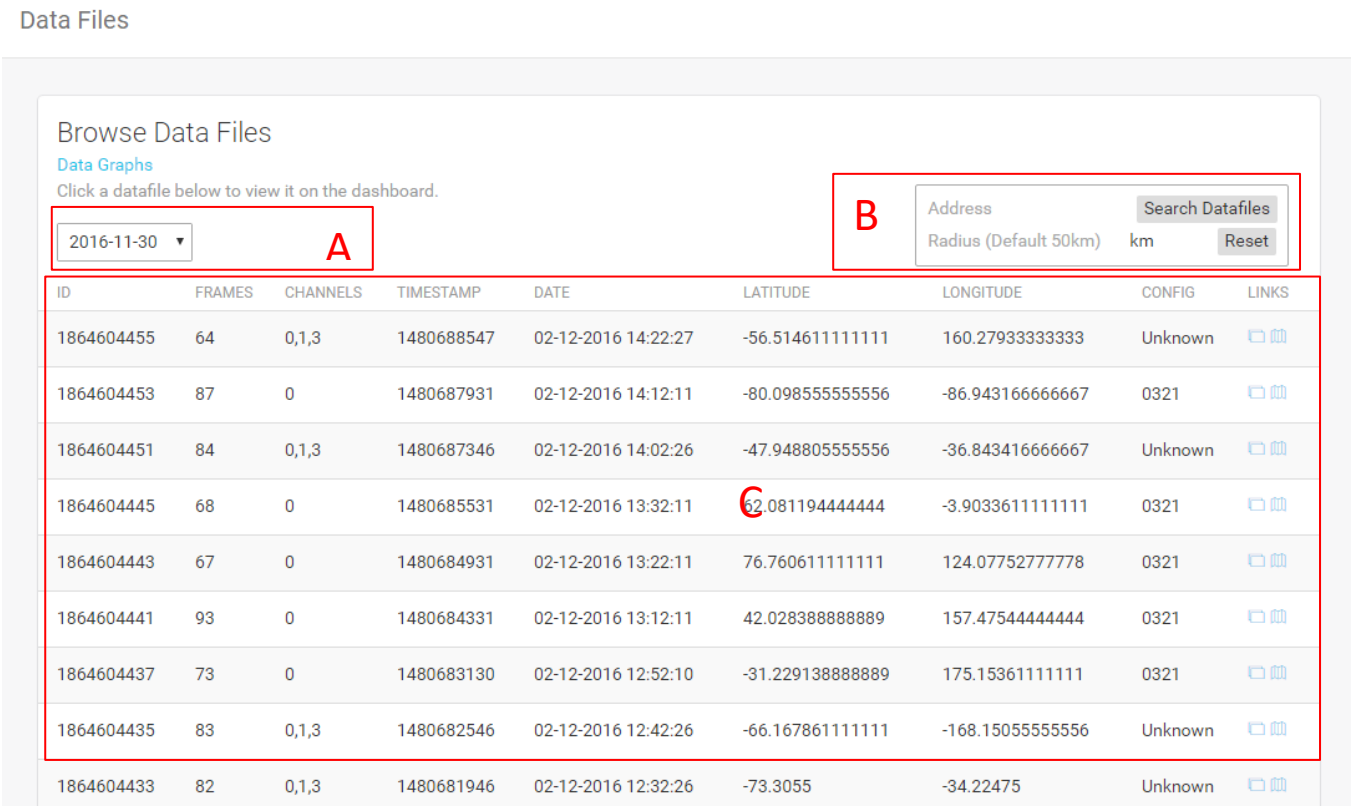
C) The frames that are to be displayed will be embedded here with the links to the image and the XYC file overlaid upon hover using either JavaScript or CSS. The analysis of the frame will be displayed below the image and this will be done by simply running the analysis algorithm on the frame via a PHP CURL request.

D) This bar will be used to hold the navigation buttons for flicking through the frames of the chosen data file. By clicking next, the browser will simply load a new URL with the GET parameters needed for getting the next frame and this will be similar for the previous button. The info button in the centre will load up a popup showing the location of the frame on a map as well as other metadata.

E) This will be a for showing the timestamp, the geolocation and the geocoded address using the Google Maps API via a CURL request.

Data Files

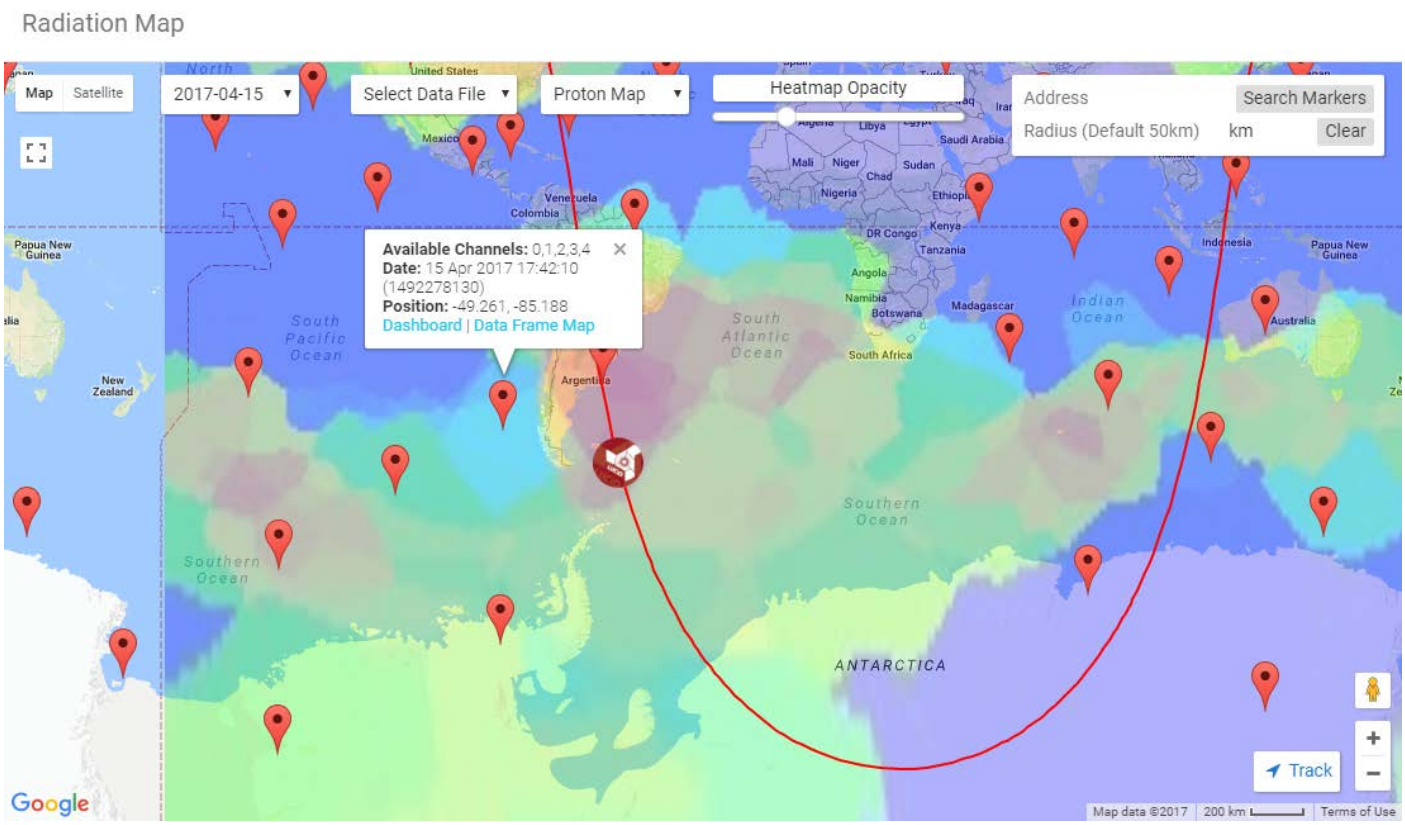
The page will have a sidebar but for explanation purposes, it has been removed from this screenshot as I have already explained it above.



This page allows users to search for data files by their data run and also by the form of radial distance from a specific location. By clicking on the links in the link column, the user can be redirected to the dashboard or the radiation map to view their selected data file in further detail. All of the metadata about the data file that is stored in the server database is presented in a neat table for users to scroll through.

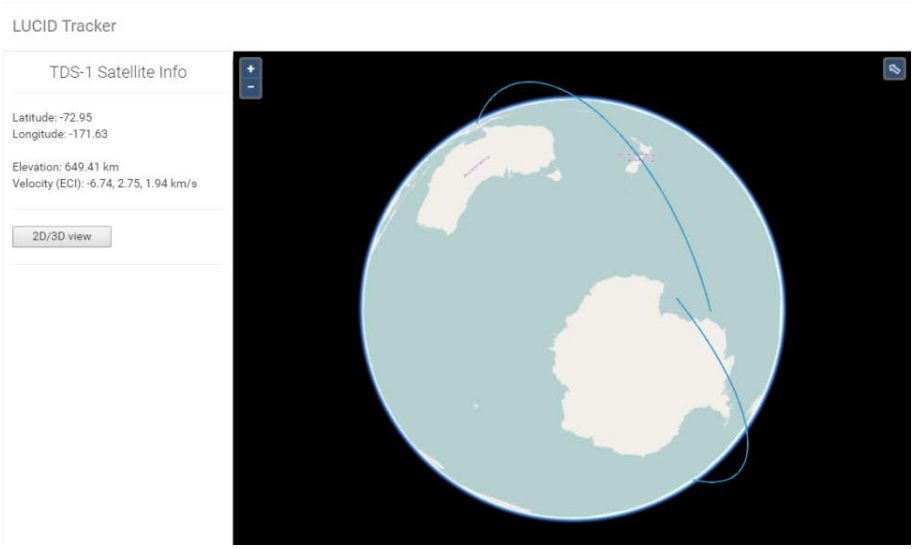
A) This feature is the main part of this page as it allows the user to filter data files by the data run. Upon selection, the GET parameter will be changed as the page reloads and the table will be updated with the new data files.
B) This is a search feature to filter by location and it works by parsing through the data files JSON data received via the data API and filtering files out if their location is not within a circular radius of a given distance of a given location.
C) The following details will be shown in the table: File ID, Number of Frames, Channels (Active Detectors), Timestamp, Date, Latitude, Longitude, Configuration File Name and links to the file on the dashboard and the map.

Radiation Map



This page shows a pre-processed radiation map overlaid on the map of the Earth using the Google Maps API. Each marker is a data file in the selected run. Upon clicking the marker, the user will be shown a small information window that shows the active detectors, the timestamp of when the file was captured and the geolocation. It will also provide a link to the same data file on the dashboard and a link to all the frames of the data file on the same radiation map. LUCID will be tracked by default however it can also be followed on the map by simply clicking the track button. The most important part of this page will be its look and its ease of use to an ordinary student and therefore most of the real time render will be done via client-side JavaScript.

LUCID Tracker



The LUCID tracker will mainly be used for demonstration purposes to show the location and trajectory of LUCID in 3D space. The user interface has been made such that the key details are presented on a side bar and the user if left to explore on the 3D globe.

Data Graphs

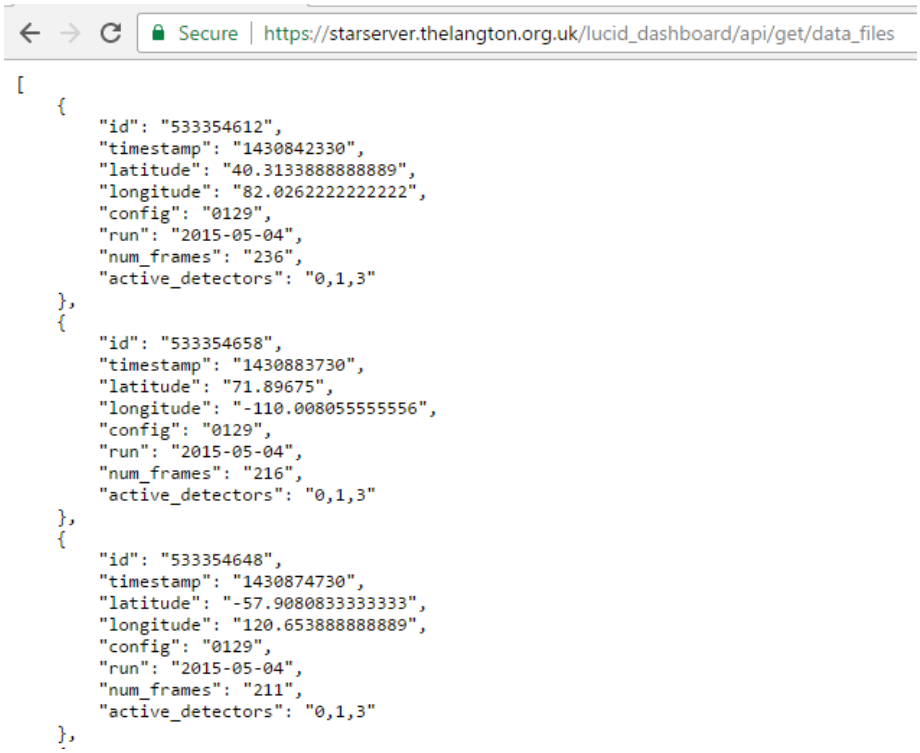
This page will provide an overview on the amount of LUCID data that has been collected. To generate graphs, the metadata will be loaded from the database and the different properties will be filtered based on the chart that is to be created. It was important to be able to keep track of an overview of the data that was collected. The metadata of all the collected data files will be processed on runtime using PHP and then converted to JSON so that it can be served into the Plotly JavaScript API for plotting charts.

Technical Solution

LUCID Data API

To access the LUCID data, I created a REST (Representational State Transfer) API (Application Programming Interface) which returns the data in JSON (JavaScript Object Notation) format when a specific URL is typed. This is done by using PHP and PDO (PHP Data Objects) to access the SQLite database where all of the metadata is stored. The base URL for the API: http://starserver.thelangton.org.uk/lucid_dashboard/api/

get/data_files



This returns all the information about all the LUCID data files ever collected using the GET request via the HTTP protocol.

Each single block of text in the curly braces refers to a single data file returning the location, the timestamp, the total number of frames taken, the configuration file used, the active chips out of the 5 detectors and the data run (the 2-day time period during which the data file was created).

get/frames?data_file={id}

The ID of the data file from the previous URL can then be used to view the frames of the data file by using the above GET request.

Each single block of text in the curly braces refers to a data frame recorded by the chips at the given timestamp and location.

The channels array refers to the data frame from each individual chip, out of the active detectors. The key of the dictionary is the chip number and the value is the XYC frame in a serialised format.

Filename: api.php

```php
<?php
header('Access-Control-Allow-Origin: *');
// Initialise Global Variables and Functions
require("init.php");
function get_datafiles(){
    // Specify sqlite database name and path
    $dir = 'sqlite:'.$GLOBALS["sqlite_db"];
    // Instantiate connection to SQLITE database
    $dbh = new PDO($dir) or die("cannot open database");
    // Run SQL Query to get all Data Files
    $query = "SELECT Id as id, Timestamp as timestamp, Latitude as latitude,
Longitude as longitude, Config as config, Run as run, num_frames, Active_detectors
as active_detectors FROM DATA_FILES";
    // Print out the metadata in JSON format
    echo json_encode($dbh->query($query)->fetchAll(PDO::FETCH_ASSOC));
}

function get_datafile($id){
    //10 digit ID of the Data File
    $id = str_pad($id, 10, "0", STR_PAD_LEFT);
    $dir = 'sqlite:'.$GLOBALS["sqlite_db"];
    // Instantiate PDO connection object and failure msg
    $dbh = new PDO($dir) or die("cannot open database");
    // Get the active detectors and the date of the run
    $query = "SELECT Active_detectors as active_detectors, Run as run FROM
DATA_FILES WHERE Id='$id'";
    $info = $dbh->query($query)->fetchAll(PDO::FETCH_ASSOC)[0];
    $run = $info["run"];
    $active_detectors = $info["active_detectors"];
    $active_detectors = explode(',', $active_detectors);

    // Run SQL Query to get a single Data File by ID
    $query = "SELECT Timestamp as timestamp, Latitude as latitude, Longitude as
longitude, Frame_number as number FROM frames WHERE Data_file='$id'";
    // Read all the metadata of the specified Data File
    $data = $dbh->query($query)->fetchAll(PDO::FETCH_ASSOC);
    $new_data = array();
    foreach ($data as $dat){
        // Get the frame number
        $num = $dat["number"];
        foreach ($active_detectors as $detector){
            // The url variable is the url of each individual XYC file
            $url =
'../data/LUCID/xyc/'.$run.'/'.$id.'/frame'.$num.'c'.$detector.'.txt';
            // Get the contents of the XYC file
            @$text = file_get_contents($url);
            // Append XYC data to PHP array for each detector
            $dat["channels"][$detector] = "$text";
        }
        //Append each edited data frame array to a new array
        $new_data[] = $dat;
    }
    // Print out the full data file (containing all the frames) in JSON format
    echo json_encode($new_data);
}

// If GET Request, then run function and print output
if (isset($_GET["data_files"])){
    // Print all data files
    echo get_datafiles();
}else if (isset($_GET["frames"]) and isset($_GET["data_file"])){
    //Print all the frames of a datafile
    echo get_datafile($_GET["data_file"]);
}
?>
```

Data Analysis

The aim of this section will be to convert the mathematics in the documented design section into a pure data analysing program class.

Blobbing (Observation)

Blobbing

To analyse the frames of LUCID data, it is necessary to identify the individual particles in a data frame. To accomplish this, the blobbing program simply takes a 256x256 NumPy array (representing a frame) as a parameter and returns a list of lists of tuples where each sub-list is a blob and the tuples it contains are the coordinates of every single pixel within that blob.

A recursive algorithm is used to check each hit pixel one by one, adding the pixel to a new cluster if it is not already a part of an existing cluster. It then finds all adjacent pixels, adds those to the same cluster, and then finds all the adjacent pixels to those adjacent pixels, deeper and deeper until there are no more adjacent pixels to be added to the original cluster. A blobbing radius is also given as a parameter to classify two blobs within the specified radius as one particle.

Filename: blobbing.py

```
1.  import numpy as np
2.  from copy import deepcopy
3.  try:
4.      from Queue import Queue
5.  except ImportError:
6.      from queue import Queue
7.
8.  def distance(point1, point2):
9.      # Simple 2D distance function using Pythagoras:
10.     # Calculates the distance between point1 (x, y) and point2 (x, y)
11.     return np.sqrt(((point2[0] - point1[0])**2) + ((point2[1] - point1[1])**2))
12.
13. class BlobFinder:
14.
15.     def __init__(self, frame, rad):
16.         self.aq = Queue()
17.         self.rad = rad
18.         self.irad = int(np.ceil(rad))
19.         self.frame = deepcopy(frame)
20.         self.blobs = []
21.         self.blob = None # Holds currently active blob
22.
23.     def circle(self, location):
24.         sqxs = range(max(0, location[0] - self.irad), min(256, location[0] + self.irad + 1))
25.         sqys = range(max(0, location[1] - self.irad), min(256, location[1] + self.irad + 1))
26.         square = [(x,y) for x in sqxs for y in sqys]
27.         return [p for p in square if (distance(p,location) <= self.rad)]
28.
29.     def add(self, xy):
30.         x,y = xy
31.         self.blob.append((x,y))
32.         close_region = self.circle((x,y))
33.         for x1,y1 in close_region:
34.             if self.frame[x1][y1] > 0:
35.                 self.frame[x1][y1] = 0
36.                 self.aq.put((x1,y1))
37.
38.     def find(self):
39.         for x in range(256):
40.             for y in range(256):
41.                 if self.frame[x][y] > 0:
42.                     self.frame[x][y] = 0
43.                     self.blob = []
44.                     self.aq.put((x,y))
```

```
45.
46.                     while not self.aq.empty():
47.                         pixel = self.aq.get()
48.                         self.add(pixel)
49.
50.                     self.blobs.append(self.blob)
51.
52.         return self.blobs
53.
54.
55. def find(frame, rad=2.9):
56.     return BlobFinder(frame,rad).find()
```

Filename: blobbing_energy.py

This is a newer version of the blobbing algorithm that includes the energy values of the pixels in the list of clusters.

```
1.  import numpy as np
2.  from scipy.spatial.distance import euclidean as dist
3.  from copy import deepcopy
4.  try:
5.      from Queue import Queue
6.  except ImportError:
7.      from queue import Queue
8.
9.
10. class BlobFinder:
11.
12.     def __init__(self, frame, rad):
13.         self.aq = Queue()
14.         self.rad = rad
15.         self.irad = int(np.ceil(rad))
16.         self.frame = deepcopy(frame)
17.         self.blobs = []
18.         self.blob = None  # Holds currently active blob
19.
20.     def circle(self, location):
21.         sqxs = range(max(0, location[0] - self.irad), min(256,
22.                      location[0] + self.irad + 1))
23.         sqys = range(max(0, location[1] - self.irad), min(256,
24.                      location[1] + self.irad + 1))
25.         square = [(x, y) for x in sqxs for y in sqys]
26.         return [p for p in square if dist(p, location) <= self.rad]
27.
28.     def add(self, xy):
29.         (x, y, c) = xy
30.         self.blob.append((x, y, c))
31.         close_region = self.circle((x, y))
32.         for (x1, y1) in close_region:
33.             if self.frame[x1][y1] > 0:
34.                 c1 = self.frame[x1][y1]
35.                 self.frame[x1][y1] = 0
36.                 self.aq.put((x1, y1, c1))
37.
38.     def find(self):
39.         for x in range(256):
40.             for y in range(256):
41.                 if self.frame[x][y] > 0:
42.                     ##added energy value
43.                     c = self.frame[x][y]
44.                     self.frame[x][y] = 0
45.                     self.blob = []
46.                     self.aq.put((x, y, c))
47.                     #print(x,y,c)
48.                     while not self.aq.empty():
49.                         pixel = self.aq.get()
50.                         #print(pixel)
```

```
                        self.add(pixel)
                    self.blobs.append(self.blob)

        return self.blobs


def find(frame, rad=2.9):
    return BlobFinder(frame, rad).find()
```

Filename: blobbing_api.py

This is the API that will allow users to perform particle clusterin directly with just a simple GET request instead of having to program the entire algorithm.

```
#!/usr/bin/env python3

import cgi
import urllib.request
import urllib.parse

import json
import sys
sys.stderr = sys.stdout
import cgitb; cgitb.enable()

import blobbing_energy as blobbing
import numpy as np
##from lucid_utils.classification.lucid_algorithm import classify
##from lucid_utils.classification.old_algorithm import classify

args = cgi.FieldStorage()
get_vars = {}
for key in args.keys():
        variable = str(key)
        value = str(args.getvalue(variable))
        get_vars[variable] = value

def read(filename):
    frame = np.zeros((256, 256))
    f = (urllib.request.urlopen(filename).read()).decode("utf-8")
    lines = f.split("\n")
    for line in lines:
        if line != "":
                vals = line.split("\t")
                x = int(float(vals[0].strip()))
                y = int(float(vals[1].strip()))
                c = int(float(vals[2].strip()))
                frame[x][y] = c
    return frame

try:
        x = int(get_vars["id"])
except:
        get_vars["id"] = 1932936732

try:
        x = int(get_vars["frame"])
except:
        get_vars["frame"] = 0

try:
        x = int(get_vars["channel"])
except:
        get_vars["channel"] = 0


filename = "http://starserver.thelangton.org.uk/lucid_dashboard/"+str(get_vars["id"])+"_"+str(get_
vars["frame"])+"_"+str(get_vars["channel"])+".txt"
```

```
54. frame = read(filename)
55.
56. clusters = blobbing.find(frame, 1.5)
57.
58. if __name__ == "__main__":
59.         print("Content-Type: application/json")    # JSON is following
60.         print()                                    # blank line, end of headers
61.         print (json.dumps(clusters))
```

Feature Extraction (Interpretation)

Common

To interpret and extract the key features of the clusters, it was necessary to calculate some metrics for the particle tracks in a way that would help to determine the properties of the particles.

The following metrics are calculated from the cluster that is provided as a list of pixels to the program:

Number of Pixels → This is calculated as the length of the pixels list.
Centroid → This is calculated using the **find centroid** function.
Radius → This is calculated using the **calculate radius** function.
Diameter → This is twice the calculated radius.
Density → This is calculated using the **calculate density** function.
Line Residual, Best Fit Theta → This is calculated using the **calculate non-linearity ("squiggliness")** function.
Centre of Circle, Curvature Radius, Circle Residual → This is calculated using the **find best fit circle** function.
Average Neighbours → This is calculated using the **find average neighbours** function.
Width → This is the number of pixels divided by the diameter provided that the number of pixels is not 1. For single pixels, the width is set to 0.

Filename: common.py

```
1.  # A set of helpful classes and functions which are common to most classification algorithms
2.  # This includes a 'Blob' class which will automatically determine the properties of a cluster (yay
    !!)
3.
4.  import numpy as np
5.  from scipy.optimize import leastsq
6.  import os
7.  from PIL import Image
8.  try:
9.      import least_squares_circle
10. except ImportError:
11.     from . import least_squares_circle
12.
13. def chunks(l, n):
14.     """Yield successive n-sized chunks from l."""
15.     for i in range(0, len(l), n):
16.         yield l[i:i + n]
17.
18. def distance(point1, point2):
19.     # Simple 2D distance function using Pythagoras:
20.     # Calculates the distance between point1 (x, y) and point2 (x, y)
21.     return np.sqrt(((point2[0] - point1[0])**2) + ((point2[1] - point1[1])**2))
22.
23. def point_line_distance(point, centroid, theta):
24.     x1, y1 = centroid
25.     x2, y2 = (centroid[0] + np.cos(theta), centroid[1] + np.sin(theta))
26.     x0, y0 = point
27.     # cheers wikipedia
28.     return np.fabs( (y2-y1)*x0 - (x2-x1)*y0 + x2*y1 - y2*x1 ) / np.sqrt( (y2-y1)**2 + (x2-
    x1)**2 )
29.
30. # Stores and calculates the attributes of a single cluster ('blob') of pixels
31. class Blob:
32.
33.     def __init__(self, pixels):
34.         self.pixels = pixels
35.         self.num_pixels = len(pixels)
36.         if not self.num_pixels:
37.             raise Exception("Cannot work on a blank cluster!")
38.         # Calculate attributes
39.         self.centroid = self.find_centroid()
40.         self.radius = self.calculate_radius()
41.         self.diameter = 2 * self.radius
42.         self.density = self.calculate_density()
43.         self.squiggliness, self.best_fit_theta = self.calculate_squiggliness()
44.         self.best_fit_circle = self.find_best_fit_circle() # x, y, radius, residuals
45.         self.curvature_radius = self.best_fit_circle[2]
46.         self.circle_residual = self.best_fit_circle[3]
47.         self.line_residual = self.squiggliness # For silly people who like words which actually ex
    ist
48.         self.width = self.num_pixels / (2 * self.radius) if not self.num_pixels == 1 else 0
49.         self.avg_neighbours = self.find_avg_neighbours()
50.
51.     def find_avg_neighbours(self):
52.         n_ns = []
53.         for x,y in self.pixels:
54.             z = [(x-1,y-1), (x-1, y), (x-1, y+1), (x, y-1), (x,y+1), (x+1, y-
    1), (x+1, y), (x+1, y+1)]
55.             n_ns.append(len(set(z).intersection(self.pixels)))
56.         return np.mean(n_ns)
57.
58.     def find_centroid(self):
59.         # Firstly, compute the centroid of the blob
60.         x_vals, y_vals = [], []
61.         for pixel in self.pixels:
62.             x_vals.append(pixel[0])
63.             y_vals.append(pixel[1])
64.         centroid = (np.mean(x_vals), np.mean(y_vals))
65.         return centroid
66.
```

```
    def calculate_radius(self):
        # Loop through each pixel and check its distance from the centroid; set the radius to the highest of these
        radius = 0.0
        for pixel in self.pixels:
            dist = distance(self.centroid, pixel)
            if dist > radius:
                radius = dist
        return radius

    def calculate_density(self):
        # Calculate the fill by hit pixels of a circle of the blob's radius around the centroid]
        # This can be >1 as the blob's radius passes through the centre of outer pixels rather than around them
        # Firstly, compute the area of the enclosing circle
        circle_area = np.pi*((self.radius)**2)
        if circle_area == 0:
            # If the blob is only one pixel in size, and so has a radius of 0, it is fully dense
            return 1
        else:
            # Divide the number of pixels in the blob by this
            return self.num_pixels / circle_area

    def calculate_squiggliness(self):
        # return angle theta anticlockwise from x axis, with the line passing through the cluster centroid
        # Split up into x and y value lists
        x_vals, y_vals = [], []
        for pixel in self.pixels:
            x_vals.append(pixel[0])
            y_vals.append(pixel[1])
        # Special case for single pixel: horizontal line, completely linear!
        if len(self.pixels) == 1:
            return (0, 0)
        # Otherwise, use leastsq to estimate a line of best fit
        # x axis as inital guess
             first_guess_theta = 0.1
             # Use scipy's regression function to magic this into a good LoBF
             best_fit_theta = leastsq(self.residuals, first_guess_theta, args = (np.array(y_vals), np.array(x_vals)))[0] % (np.pi)
             #print np.degrees(best_fit_theta)
             squiggliness = np.sum([point_line_distance(p, self.centroid, best_fit_theta)**2 for p in self.pixels])
             return squiggliness, best_fit_theta[0]

         # For the regression in squiggliness calculations...
         def residuals(self, theta, y, x):
             return point_line_distance((x,y), self.centroid, theta)

         def find_best_fit_circle(self):
             if self.num_pixels == 1:
                 return 0,0,0,0
             # Circle regression will break if only one pixel is given
             if self.num_pixels == 1:
                 return 0, 0, 0, 0 # We love special cases
             x_vals, y_vals = [], []
             for pixel in self.pixels:
                 x_vals.append(pixel[0])
                 y_vals.append(pixel[1])
             # The cluster centroid is often a very bad first guess for the circle centre, so try with a couple of others...
             x, y, = self.centroid
             d = self.diameter
             th = np.radians(self.best_fit_theta)
             p1 = (x + d*np.cos(th - (np.pi/2)), y + d*np.sin(th - (np.pi/2)))
             p2 = (x + d*np.cos(th + (np.pi/2)), y + d*np.sin(th + (np.pi/2)))
             test_circles = [least_squares_circle.leastsq_circle(x_vals, y_vals, test_point) for test_point in [self.centroid, p1, p2]]
             # circle[3] is being minimised
             test_circles.sort(key = lambda circle: circle[3])
             return test_circles[0]
```

Classification (Decision)

This part of the project focuses on combining the common.py file into a single easy-to-use analysis module. The file below returns a list of metrics when analysing a given blob.

Filename: lucid_algorithm_data.py

```
try:
    import common
except ImportError:
    from . import common

class Blob(common.Blob):
    def classify(self):
        return [self.num_pixels, self.density, self.radius, self.curvature_radius, self.line_residual, self.circle_residual, self.width, self.avg_neighbours]

def classify(blob):
    # A quick wrapper method for ease of use
    b = Blob(blob)
    return b.classify()
```

To access this module and return the metrics, simply do the following:

```
from lucid_utils.classification.lucid_algorithm_data import classify
```

```
metrics = []
clusters = blobbing.find(frame, 1.5)
for cluster in clusters:
    metric = classify(cluster)
    metrics.append(metric)
```

The metrics list will be a 2 dimensional array containing the metrics for each cluster (blob) in the frame.

To classify the particle, the metrics of each particle will be inputted into the neural network and it will return the predicted class of the particle.

As per the design, the neural network analysis requires two programs to be created:

1. A program to train the neural network with a given dataset, test its accuracy, efficiency and performance and then store the model as a file
2. A program to use the stored model as an easy-to-use prediction library

1) Neural Network Training

```
import os
## USED FOR MATRIX CALCULATIONS AND NETWORK GRAPH CREATION
import tensorflow as tf
from tensorflow.contrib.tensorboard.plugins import projector
## ONLY FOR BASIC ARRAY MANIPULATION
import numpy as np
## import class that calculates and returns the particle metrics
from lucid_utils_custom.classification.lucid_algorithm_data import classify

###############################################
## ONLY NEED THIS LIBRARY FOR SHUFFLING DATA
import tflearn
from tflearn.data_utils import shuffle
###############################################

import sqlite3
## CONNECT TO LUCID TRAINER DATABASE
conn = sqlite3.connect('./classified.db')
c = conn.cursor()

# Neural network expected outputs
```

```
particle_outputs = {
    "gamma":    [1, 0, 0, 0, 0, 0],
    "beta":     [0, 1, 0, 0, 0, 0],
    "muon":     [0, 0, 1, 0, 0, 0],
    "proton":   [0, 0, 0, 1, 0, 0],
    "alpha":    [0, 0, 0, 0, 1, 0],
    "others":   [0, 0, 0, 0, 0, 1]
}

# Lists to be used as training data
inputs = []
targets = []

## read and preprocess data from database
for row in c.execute('SELECT * FROM blobs'):
    r_id = row[0]
    r_frame = row[1]
    r_channel = row[2]
    r_blob = eval(row[3])
    r_class = row[4]

    r_data = classify(r_blob)
    r_tar = particle_outputs[r_class]
    inputs.append(r_data)
    targets.append(r_tar)

## NEURAL NETWORK ARCHITECTURE
def multilayer_perceptron(x, weights, biases):
    x = tf.nn.dropout(x, 1.0)

    with tf.name_scope('layer_1') as scope:
      # Hidden layer with sigmoid activation
      layer_1 = tf.add(tf.matmul(x, weights['w1']), biases['b1'])
      layer_1 = tf.nn.sigmoid(layer_1, name="sigmoid")
      layer_1 = tf.nn.dropout(layer_1, 1.0, name="dropout")

    with tf.name_scope('layer_2') as scope:
      # Hidden layer with sigmoid activation
      layer_2 = tf.add(tf.matmul(layer_1, weights['w2']), biases['b2'])
      layer_2 = tf.nn.sigmoid(layer_2, name="sigmoid")
      layer_2 = tf.nn.dropout(layer_2, 1.0, name="dropout")

    with tf.name_scope('layer_out') as scope:
      # Output layer with softmax activation
      out_layer = tf.nn.softmax(tf.matmul(layer_2, weights['w_out']) + biases['b_out'], name="soft
max")

    return out_layer

# Training and Test Data Splitter
from sklearn.model_selection import train_test_split

X = inputs
Y = targets
# Shuffle the data
X, Y = shuffle(X, Y)
X = np.array(X)
Y = np.array(Y)

##SPLIT SINGLE DATASET INTO A SPECIFIC RATIO
trX, teX, trY, teY = train_test_split(X, Y, test_size=0.1, random_state=1304)

# Hyperparameters
learning_rate = 0.01/2
epochs = 100
# Architecture Parameters
n_hidden_1 = 128 # 1st layer number of nodes
n_hidden_2 = 48 # 2nd layer number of nodes
n_input = 8 # data input
n_classes = 6 # total classes

# Setup Tensorflow name scopes
with tf.name_scope('train_inputs') as scope:
```

```
  X = tf.placeholder("float", [None, n_input])
with tf.name_scope('train_outputs') as scope:
  Y = tf.placeholder("float", [None, n_classes])

tf.add_to_collection("X", X)
tf.add_to_collection("Y", Y)

## INITIALISE WEIGHTS
with tf.name_scope('weights') as scope:
  weights = {
      'w1': tf.Variable(tf.random_normal([n_input, n_hidden_1]), name="w1"),
      'w2': tf.Variable(tf.random_normal([n_hidden_1, n_hidden_2]), name="w2"),
      'w_out': tf.Variable(tf.random_normal([n_hidden_2, n_classes]), name="w_out")
  }
  tf.summary.histogram("w1", weights["w1"])
  tf.summary.histogram("w2", weights["w2"])
  tf.summary.histogram("w_out", weights["w_out"])

## INITIALISE BIASES
with tf.name_scope('biases') as scope:
  biases = {
      'b1': tf.Variable(tf.zeros(n_hidden_1), name="b1"),
      'b2': tf.Variable(tf.zeros(n_hidden_2), name="b2"),
      'b_out': tf.Variable(tf.zeros(n_classes), name="b_out")
  }
  tf.summary.histogram("b1", biases["b1"])
  tf.summary.histogram("b2", biases["b2"])
  tf.summary.histogram("b_out", biases["b_out"])

## CREATE NEURAL NETWORK USING WEIGHTS AND BIASES
py_x = multilayer_perceptron(X, weights, biases)

## CALCULATE CROSS ENTROPY COST SCOPE
with tf.name_scope('cross_entropy_cost') as scope:
  cross_entropy = tf.reduce_sum(- Y * tf.log(py_x), 1)
  cost = tf.reduce_mean(cross_entropy, name="cross_entropy_mean")

## GRADIENT DESCENT COST OPTIMISER
train_op = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
tf.add_to_collection("train_op", train_op)

## TRAINING ACCURACY SCOPE
with tf.name_scope('training_accuracy') as scope:
  correct_prediction = tf.equal(tf.argmax(py_x, 1), tf.argmax(Y, 1))
  accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

## CLASS PROBABILITY SCOPE
with tf.name_scope('class_probability') as scope:
  full_op = py_x
  tf.add_to_collection("full_op", full_op)

## CLASS PREDICTION SCOPE
with tf.name_scope('class_prediction') as scope:
  predict_op = tf.argmax(py_x, 1)
  tf.add_to_collection("predict_op", predict_op)

# Create a summary for our cost and accuracy
tf.summary.scalar("Loss", cost)
tf.summary.scalar("Training Accuracy", accuracy)
# Merge all summaries into a single "operation" which we can execute in a session
summary_op = tf.summary.merge_all()

# Log directory where all working files are saved
LOG_DIR = "./tf_models/"

# Launch the graph in a session
with tf.Session() as sess:
    # Initialize all variables
    sess.run(tf.global_variables_initializer())
    writer = tf.summary.FileWriter(LOG_DIR, graph=tf.get_default_graph())
    ################################################################
    ## RESTORE MODEL
    saver = tf.train.Saver()
```

```
167.      checkpoint = tf.train.latest_checkpoint(LOG_DIR)
168.      if checkpoint:
169.          saver.restore(sess, checkpoint)
170.      ###############################################################
171.      ## TRAINING FOR A NUMBER OF EPOCHS
172.      for i in range(epochs):
173.          ## TRAIN ON TRAINING DATA
174.          for start, end in zip(range(0, len(trX), 128), range(128, len(trX)+1, 128)):
175.              _, train_summary = sess.run([train_op, summary_op], feed_dict={X: trX[start:end], Y:
    trY[start:end]})
176.          # Run prediction on training data
177.          train_predict = sess.run(predict_op, feed_dict={X: trX})
178.          # Calculate training accuracy
179.          train_acc = np.mean(np.argmax(trY, axis=1) == train_predict)
180.          # Run prediction on test data
181.          predictor = sess.run(predict_op, feed_dict={X: teX})
182.          # Calculate test accuracy
183.          predict_acc = np.mean(np.argmax(teY, axis=1) == predictor)
184.          # Prints results for each epoch
185.          print(i, train_acc, predict_acc)
186.          # Save summary data for visualisation
187.          test_summary = tf.Summary(value=[tf.Summary.Value(tag="Test Accuracy", simple_value=predi
    ct_acc)])
188.          writer.add_summary(test_summary, i)
189.          writer.add_summary(train_summary, i)
190.
191.      ## RUNNING PREDICTION ON TEST DATA
192.      print(sess.run(predict_op, feed_dict={X: teX}))
193.      accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
194.      print("Accuracy:", accuracy.eval({X:teX, Y:teY}))
195.
196.      # Save all variables (weights and biases)
197.      names_to_vars = {v.op.name: v for v in tf.global_variables()}
198.      saver = tf.train.Saver(names_to_vars)
199.      # This is the model file which contains the weights
200.      save_path = saver.save(sess, LOG_DIR+"model5.ckpt")
201.
202.  # This is the tensorflow graph
203.  meta_graph_def = tf.train.export_meta_graph(filename=LOG_DIR+'model5.meta')
```

This is the simplified training file although the proper code shown in the neural network section is the full version which includes the Tensorboard training visualisations as well as plots of the training data. The entire algorithm is explained in the research section of this project. The neural network is a rather advanced piece of machine learning that can be used to approximate any function given an appropriately sized neural network.

2) Neural Model Prediction

```
#!/usr/bin/env python
import os
import tensorflow as tf
import numpy as np
from lucid_utils.classification.lucid_algorithm_data import classify

# Classes can be renamed but they must stay in the same order
classes = ["gamma", "beta", "muon", "proton", "alpha", "others"]

def predict(cluster):
    tf.reset_default_graph()
    # Model directory and path initialisation
    this_dir, this_filename = os.path.split(__file__)
    model_name = '\model5.meta'
    LOG_DIR = os.path.join(this_dir, "Model")

    checkpoint = tf.train.latest_checkpoint(LOG_DIR)
    # Launch the graph in a session
    with tf.Session() as sess:
        # Initialise all variables
        sess.run(tf.global_variables_initializer())
        # Import model
        saver = tf.train.import_meta_graph(LOG_DIR+model_name)
        # Restore model weights from checkpoint
        saver.restore(sess, checkpoint)
        X = tf.get_collection("X")[0]
        # Get model operations
        predict_op = tf.get_collection('predict_op')[0]
        full_op = tf.get_collection('full_op')[0]
          # Run prediction operation on cluster metrics and return class index
        index = sess.run(predict_op, feed_dict={X: [classify(cluster)]})[0]
        return classes[index]
```

To use the model, the basics of this library is very simple. It loads up a neural model in a Tensorflow session allowing the graph to be used with any provided data. The function takes a cluster/blob as a parameter and then it returns the matching class by running the neural network on the metrics calculated for that blob (metrics calculated using the **classify** function in the **lucid_algorithm_data** file).

LUCID Neural Trainer

Homepage

Filename: home.php (Stylistic Home Page)

```
<?php
// Get metadata of all data files
$get_files =
json_decode(file_get_contents("https://starserver.thelangton.org.uk/lucid_dashboard/
api/get/data_files"), true);
// Reverse array to get latest files first
$get_files = array_reverse($get_files);
$_GET["msg"] = isset($_GET["msg"]) ? $_GET["msg"]:"";
?>
<html>
<head>
    <title>LUCID Trainer Home</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
    <link href="https://fonts.googleapis.com/css?family=Raleway|Space+Mono"
rel="stylesheet">
    <link rel="shortcut icon" type="image/png" href="./img/LN.png"/>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
    <script>
    $( document ).ready(function() {
        //Selection dropdown logic for file and frame selection
        $( "#selected_file" ).change(function() {
            if ($('#selected_frame').length && $('#selected_channel').length &&
$('#submit').length){
                $( '#selected_frame' ).remove();
                $( '#selected_channel' ).remove();
                $( '#submit' ).remove();
            }
            var sel = $('<select id="selected_frame">').appendTo('#selectors');
            index_max = this.options[this.selectedIndex].getAttribute("data-
frames");
            for (i=0; i < index_max; i += 1) {
                sel.append($("<option>").attr('value',i).text(i));
            }
            jQuery("#selected_frame option:first-child").attr("selected", true);
            var sel2 = $('<select id="selected_channel">').appendTo('#selectors');
            channels = this.options[this.selectedIndex].getAttribute("data-
channels").split(",");
            for (var i = 0; i < channels.length; i++) {

sel2.append($("<option>").attr('value',channels[i]).text(channels[i]));
            }
            jQuery("#selected_channel option:first-child").attr("selected", true);
            $("#selectors").append('<button id="submit" class="button submit-
button">Train</button>');
        });
        // Load new frame image when new file clicked
        $(document).on( "change", "select", function(){
            var url =
"https://starserver.thelangton.org.uk/lucid_dashboard/gen_image.php?id="+$("#selecte
d_file").val()+"&frame="+$("#selected_frame").val()+"&channel="+$("#selected_channel
").val()+"&scale=2";
            $('#frame_image').attr("src",url);
        });
    });
    // Redirect to training page when user clicks train
    $(document).on('click', '#submit', function() {
```

```
        var url =
"./train.php?id="+$("#selected_file").val()+"&frame="+$("#selected_frame").val()+"&c
hannel="+$("#selected_channel").val()+"&index=0";
        window.location.href = url;
    });
    </script>
    <style>
    *{
        font-family: 'Raleway', sans-serif;
        /*font-family: 'Space Mono', monospace;*/
    }
    body {
        margin: 30px auto;
        text-align: center;
    }
    select{
        margin: 30px 15px 5px 15px;
        padding: 6px;
        position: relative;
        border-radius: 2px;
        border: 1px solid #aaa;
        /*box-shadow: 1px 1px 3px rgba(0,0,0,.3);*/
        font-size: 18px;
    }
    .button {
        -webkit-transition-duration: 0.4s; /* Safari */
        transition-duration: 0.4s;
        padding: 10px 30px;
        font-size: 16px;
        border-radius: 5px;
    }
    .button:hover {
        background-color: #008CBA;
        color: white;
    }
    .submit-button {
        background-color: white;
        color: black;
        border: 2px solid #008CBA;
    }
    iframe {
        margin: 70px 20px 0px;
        width: calc(100% - 40px);
    }
    </style>
</head>
<body>
    <h1>LUCID Neural Training</h1>
    <div id="selectors">
        <label for="selected_file">Select Data File</label>
        <select id="selected_file" name="selected_file">
        <?php
        //Add each file as dropdown option
        foreach ($get_files as $file){
            echo "<option value='".$file["id"]."' data-
frames='".$file["num_frames"]."' data-
channels='".$file["active_detectors"]."'>".$file["id"]."</option>";
        }
        ?>
        </select>
    </div>
    <br>
    <p>
    <?php
        if ($_GET["msg"] == "complete"){
            echo "<h3>All blobs complete in this frame. Thank you!</h3>";
        }elseif($_GET["msg"] == "empty"){
```

```
            echo "<h3>This frame is empty! Please try another id, frame or
channel...</h3>";
        }else{
            //do nothing
        }
    ?>
    </p>
    <img id="frame_image" width="450px" height="450px"/>
</body>
</html>
```

AJAX Responder

Filename: classify.php (AJAX Responder)

```php
<?php
ini_set('max_execution_time', 0);
$db_storage = "./db/classified.db";
$db = new SQLite3($GLOBALS["db_storage"]) or die('Unable to open database');
$query = "
CREATE TABLE IF NOT EXISTS blobs (
    id INTEGER,
    frame INTEGER,
    channel INTEGER,
    pixels STRING,
    classification STRING,
    timestamp INTEGER
);";
$db->exec($query) or die('');
$db->close();

function mempty(){
    foreach(func_get_args() as $arg)
        if(empty($arg))
            continue;
        else
            return false;
    return true;
}

$id = isset($_POST["id"]) ? $_POST["id"]:"";
$frame = isset($_POST["frame"]) ? $_POST["frame"]:"";
$channel = isset($_POST["channel"]) ? $_POST["channel"]:"";
$pixels = isset($_POST["pixels"]) ? $_POST["pixels"]:"";
$classification = isset($_POST["classification"]) ? $_POST["classification"]:"";

// check if the variables are empty
if (!mempty($id, $frame, $channel, $pixels, $classification)){
    //connect to DB
    $db = new SQLite3($GLOBALS["db_storage"]) or die('Unable to open database');
    //check if cluster already exists
    $result = $db->query("SELECT COUNT(*) as count FROM blobs WHERE id='$id' AND
frame='$frame' AND channel='$channel' and pixels='$pixels';");
    $row = $result->fetchArray();
    $numRows = $row['count'];

    //create time signature for DB entry
    $timestamp = time();
    // If it doesn't already exist, INSERT data in database
    if ($numRows == 0){
        $db->exec("INSERT INTO blobs (id, frame, channel, pixels,
classification, timestamp) VALUES ('$id', '$frame', '$channel', '$pixels',
'$classification', '$timestamp');");

    // else UPDATE data in database
    }else{
       $db->exec("UPDATE blobs SET classification='$classification',
timestamp='$timestamp' WHERE id='$id' AND frame='$frame' AND channel='$channel'
AND pixels='$pixels';");
    }
    $db->close();
    //return response for client-side JS
    echo (json_encode(array("status"=>"valid")));
}else{
    //return response for client-side JS
    echo (json_encode(array("status"=>"invalid")));
}
?>
```

Filename: train.php (Auto-Generate Questionnaire)

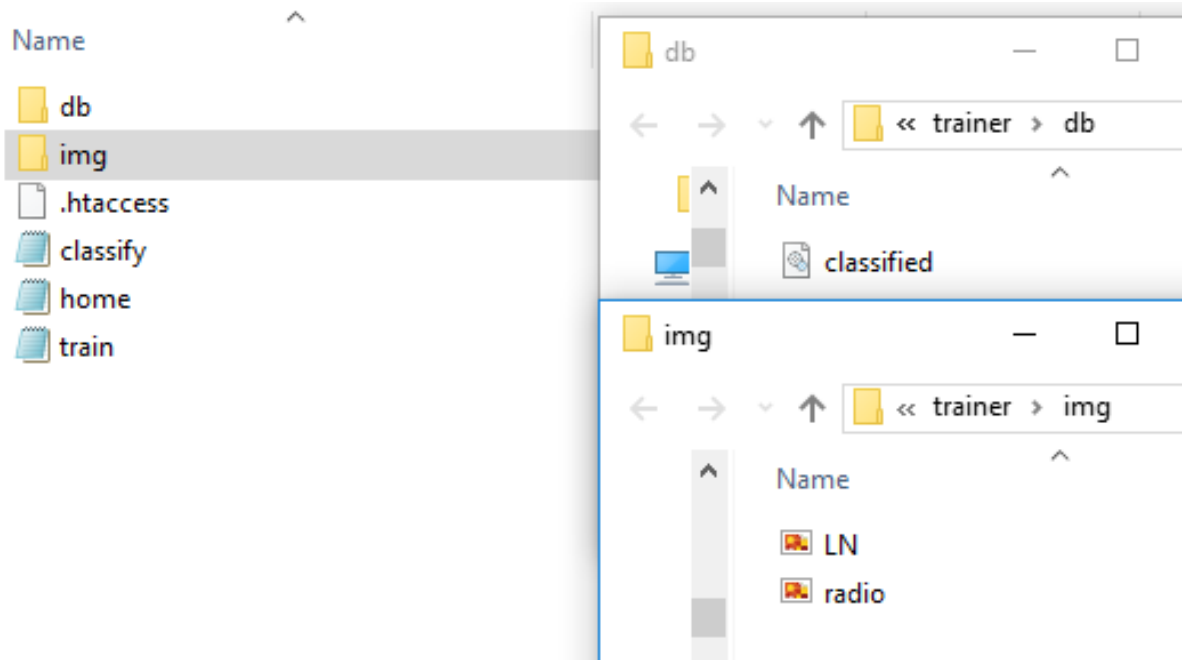
```php
<?php
ini_set('max_execution_time', 0);

function is_int_val($data) {
    if (is_int($data) === true){
        return true;
    }elseif (is_string($data) === true && is_numeric($data) === true) {
        return (strpos($data, '.') === false);
    }else{
        return false;
    }
}

function getProperColour($image, $energy_val){
    if ($energy_val > 0 && $energy_val < 100){
        $colour = "green";
    }elseif ($energy_val >= 100 && $energy_val < 200){
        $colour = "yellow";
    }elseif ($energy_val >= 200 && $energy_val < 300){
        $colour = "orange";
    }elseif ($energy_val >= 300){
        $colour = "red";
    }
    return $colour;
}

function gen_image($pixels, $scale){
    $width = 256;
    $height = 256;
    $image = imagecreate( $width, $height );

    // $colour_opt = isset($_GET["simple"]);
    $colour_opt = True;
    $dark_blue_background = imagecolorallocate ( $image, 0, 0, 40 );
    //$blue_background = imagecolorallocate ( $image, 31, 119, 208 );
    $green = imagecolorallocate ($image, 0, 255, 0);
    $yellow = imagecolorallocate ($image, 255, 170, 0);
    $orange = imagecolorallocate ($image, 255, 85, 0);
    $red = imagecolorallocate ($image, 255, 0, 0);
    $white = imagecolorallocate ($image, 255, 255, 255);

    foreach ($pixels as $pixel){
        $x = $pixel[0];
        $y = $pixel[1];
        if ( getProperColour($image, $pixel[2]) == "red" ){
            $colour = $red;
        }elseif( getProperColour($image, $pixel[2]) == "orange" ){
            $colour = $orange;
        }elseif( getProperColour($image, $pixel[2]) == "yellow" ){
            $colour = $yellow;
        }elseif( getProperColour($image, $pixel[2]) == "green" ){
            $colour = $green;
        }
        imagesetpixel ( $image, $x, $y, $colour );
    }
    if ($scale >= 1 and $scale < 21){
        $image = imagescale ($image, $scale*256, $scale*256, IMG_NEAREST_NEIGHBOUR);
    }

    ob_start();
    imagepng($image);
    $imageData = ob_get_contents();
    ob_clean();
    return base64_encode($imageData);
```

```
}

if (isset($_GET["rand"])){
    $get_files =
json_decode(file_get_contents("https://starserver.thelangton.org.uk/lucid_dashboard/
api/get/data_files"), true);
    $item = $get_files[rand(0, count($get_files)-1)];
    // print_r($item);
    $id = $item["id"];
    $frame = rand(0, $item["num_frames"] - 1);
    $channels = explode(",", $item["active_detectors"]);
    // echo $channels;
    $channel = $channels[rand(0, count($channels)-1)];
    //$id = str_pad($id, 10, "0", STR_PAD_LEFT);
    $text =
file_get_contents("https://starserver.thelangton.org.uk/lucid_dashboard/pyxan/blobbi
ng/".$id."_".$frame."_".$channel."");
    $blobs = json_decode($text);
    $index = rand(0, count($blobs) - 1);
}else{
    $id = isset($_GET["id"]) ? $_GET["id"]:"1932936732";
    $frame = isset($_GET["frame"]) ? $_GET["frame"]:"0";
    $channel = isset($_GET["channel"]) ? $_GET["channel"]:"0";
    $index = isset($_GET["index"]) ? $_GET["index"]:"0";

    if (is_int_val($id) and is_int_val($frame) and is_int_val($channel) and
is_int_val($index)){
        //$id = str_pad($id, 10, "0", STR_PAD_LEFT);
        $text =
file_get_contents("https://starserver.thelangton.org.uk/lucid_dashboard/pyxan/blobbi
ng/".$id."_".$frame."_".$channel."");
        $blobs = json_decode($text);
    }else{
        die(header("Location: home.php"));
    }
}

if (count($blobs) == 0){
    die(header("Location: home.php?msg=empty"));
}else{
    //image data , scale
    $image_data = gen_image($blobs[$index], 2);
}

?>
<!DOCTYPE html>
<html>
<head>
<title>LUCID Trainer</title>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<link href="https://fonts.googleapis.com/css?family=Raleway|Space+Mono"
rel="stylesheet">
<link rel="shortcut icon" type="image/png" href="./img/LN.png"/>
<style type="text/css">
*{
    /*font-family: 'Raleway', sans-serif;*/
    font-family: 'Space Mono', monospace;
}
body {
    width: 100%;
    margin: 0;
}
#frame_container {
    padding: 20px;
    width: auto;
    display: inline-block;
    float:left;
```

```
}
#train_container {
    padding: 20px;
    width: 45%;
    display: inline-block;
}
#train_container p{
    font-size: 20px;
    margin: 0;
}
#train_container small{
    font-size: 14px;
    margin: 0;
    display: inline;
}
input[type=radio].css-checkbox {
    position:absolute; z-index:-1000; left:-1000px; overflow: hidden; clip: rect(0 0
0 0); height:1px; width:1px; margin:-1px; padding:0; border:0;
}
input[type=radio].css-checkbox + label.css-label {
    padding-left:26px;
    height:21px;
    display:inline-block;
    line-height:21px;
    background-repeat:no-repeat;
    background-position: 0 0;
    font-size:18px;
    vertical-align:middle;
    cursor:pointer;
}
input[type=radio].css-checkbox:checked + label.css-label {
    background-position: 0 -21px;
}
label.css-label {
    background-image:url(./img/radio.png);
    -webkit-touch-callout: none;
    -webkit-user-select: none;
    -khtml-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
}
.button {
    -webkit-transition-duration: 0.4s; /* Safari */
    transition-duration: 0.4s;
    padding: 10px 30px;
    font-size: 18px;
    border-radius: 5px;
}
.button:hover {
    background-color: #008CBA;
    color: white;
}
.submit-button {
    background-color: white;
    color: black;
    border: 2px solid #008CBA;
}
</style>
</head>
<body>
    <div id="frame_container">
        <?php
        echo '<img src="'.$image_data.'">';
        ?>
    </div>
    <div id="train_container">
        <?php
```

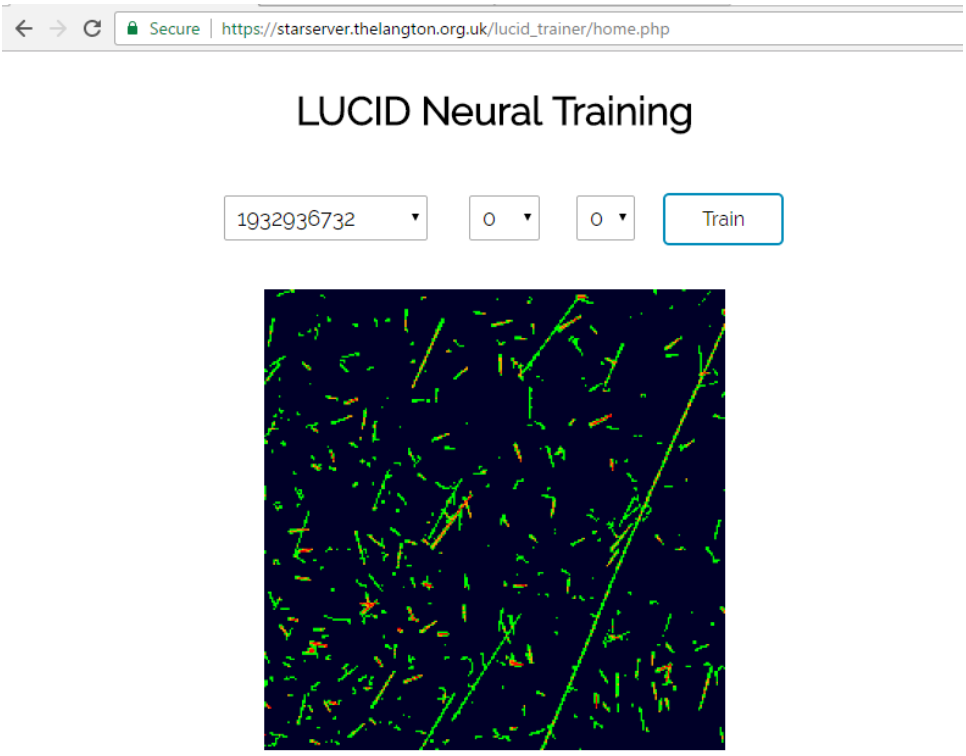
```
        echo '<p>Data Frame: '.$id.'_'.$frame.'_'.$channel.'</p>';
        echo '   -   <small>Maximum Blob Index: '.(count($blobs)-
1).'</small>';
        echo '<p>Blob Index: '.$index.'</p>';
        echo '  -   <small>Pixels:
'.json_encode($blobs[$index]).'</small></br>';
        ?>
        <script>
        $( document ).ready(function() {
            $( "#submit" ).click(function() {
                var selected =
document.querySelector('input[name="classification"]:checked').value;
                $.ajax({
                  method: "POST",
                  url: "classify.php",
                  data: { id: "<?php echo $id; ?>", frame: "<?php echo $frame; ?>",
channel: "<?php echo $channel; ?>", pixels: "<?php echo json_encode($blobs[$index]);
?>", classification: selected }
                }).done(function( msg ) {
                  //reload page
                  //console.log(msg);
                  <?php
                  if (isset($_GET['rand'])){
                    echo 'location.reload();';
                  }else{
                    $new_index = $index + 1;

                    if ($new_index == count($blobs)){
                      echo 'window.location.href = "./home.php?msg=complete";';
                    }else{
                        echo 'window.location.href =
"./train.php?id='.$id.'&frame='.$frame.'&channel='.$channel.'&index='.$new_index.'";
';
                    }
                  }
                  ?>
                });
            });
        });
        </script>
        <br>
        <div>
          <input type='radio' class='css-checkbox' name='classification' id='alpha'
value='alpha' checked> <label for='alpha' class='css-label'>Alpha</label><br><br>
          <input type='radio' class='css-checkbox' name='classification' id='beta'
value='beta'> <label for='beta' class='css-label'>Beta</label><br><br>
          <input type='radio' class='css-checkbox' name='classification' id='gamma'
value='gamma'> <label for='gamma' class='css-label'>Gamma</label><br><br>
          <input type='radio' class='css-checkbox' name='classification' id='proton'
value='proton'> <label for='proton' class='css-label'>Proton</label><br><br>
          <input type='radio' class='css-checkbox' name='classification' id='muon'
value='muon'> <label for='muon' class='css-label'>Muon</label><br><br>
          <input type='radio' class='css-checkbox' name='classification' id='others'
value='others'> <label for='others' class='css-label'>Others</label><br><br>
           </div>
        <br>
        <button id="submit" class="button submit-button">Submit</button>
    </div>
    </br></br>
    <div id="frame_container">
        <?php
        echo '<img
src="https://starserver.thelangton.org.uk/lucid_dashboard/gen_image.php?id='.$id.'&f
rame='.$frame.'&channel='.$channel.'&scale=2">';
        ?>
    </div>
</body>
</html>
```

Full File Directory



This web application is very minimal as it fulfils the single purpose of generating training data. Its simplicity also makes it very flexible and allows for improvements or additions to the training data. The images which are shown include only the logo and the stylish radio button design. All other assets such as CSS and JS are fully accessed from Content Delivery Networks (CDNs). The database stores the training data as the data is sequentially analysed. The htaccess file simply restricts users from accessing the database folder or the database and prevents the data from being stolen. It is also used for routing URLs to specific pages with their respective parameters although for this web application, this was not necessary as URL rewrites were not used. Directory listing was also disabled for the file directory in the htaccess file. The whole set of APIs that have been used to created this application will be held in the LUCID Dashboard directory as this will be the main access point for the data.

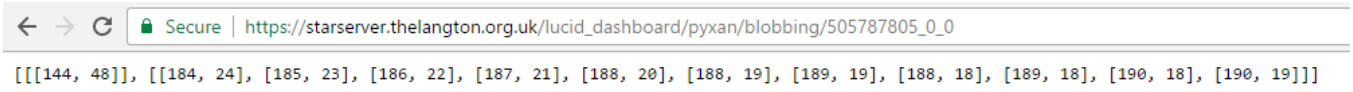
Final Web Application



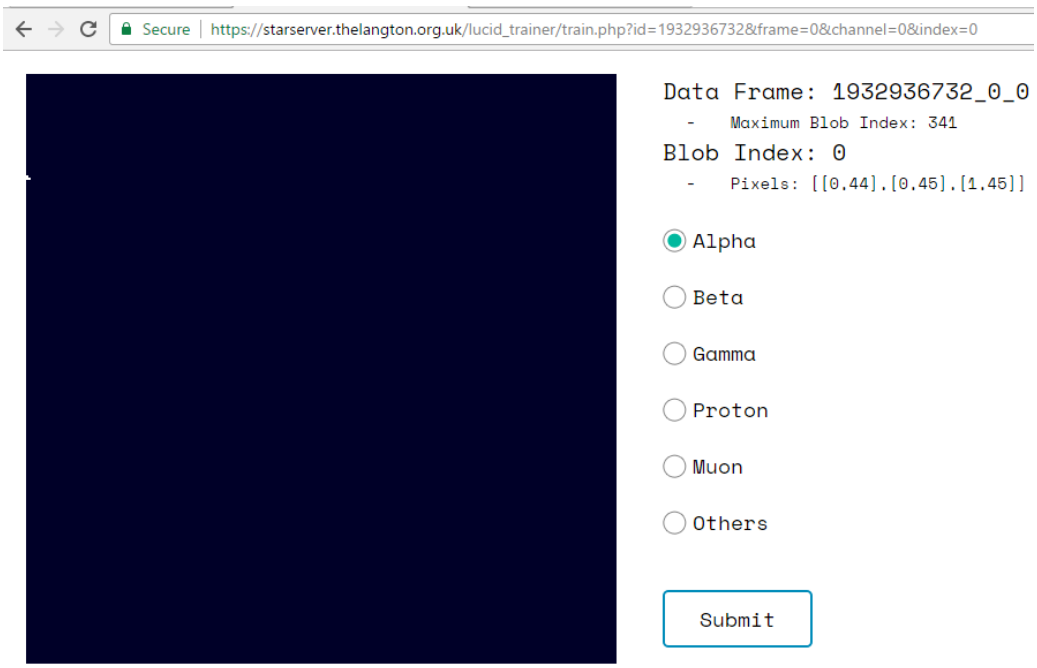
In the home page, the user can select a data file ID. After selecting the data file, the user will then be able to choose a frame number and a channel.

When the user clicks the train button (after having selected the parameters), the user will be redirected to the training questionnaire for the specific frame.

If the frame is empty, the user will be redirected back to the home page.



To display the blob images in the questionnaire, I needed a REST API for the blobbing algorithm as all the scientific programming could only be done in Python and not in PHP. The blobbing API takes the id, frame and channel as the GET parameters and returns a JSON array containing the blobs as arrays of coordinates. This enabled me to use the JSON blobbing data in the PHP questionnaire.



The questionnaire page loops through all the blobs in a specified frame asking the user for the classification. When all the blobs in the specific frame are classified, the user is redirected to the home page to train a new data file.

All the classifications are stored in a SQLite database via AJAX (Asynchronous JavaScript and XML) requests to another PHP file passing the id, frame, channel, pixels and the classification as the POST variables.

Neural Model

Using the flowcharts created in the Documented Design section, it was possible to create the neural network in Python with ease.

Training & Testing

The data used to test the neural network was simply generated by dividing the database from the LUCID Neural Trainer randomly into two sets of data: a training dataset and a test dataset; where the test dataset is smaller than the training dataset. Then using the metrics program, the metrics of the blobs were calculated. The training dataset was used to train the neural network and then to calculate the accuracy after training the neural network, the test data was inputted into the neural network and the output values were computed and stored. Using the output value, the representative class was determined for each measured particle track and these were then compared with the original classes stored in the database.

The test data had to be different from the training data just so that the neural network did not overfit to the training data. Overfitting means that the accuracy of the neural network is only valid for the training data inputted into the neural network and therefore the output would not be of the same accuracy for another dataset. This is mainly because the model may have been made too complex and therefore accounts for idiosyncrasies in the training data set. To prevent overfitting, the test data had to be chosen randomly and had to represent a real-life dataset.

(Testing Process & Results shown in Testing Section)

Model Training

```
#!/usr/bin/env python
import os

## USED FOR MATRIX CALCULATIONS AND NETWORK GRAPH CREATION
import tensorflow as tf
from tensorflow.contrib.tensorboard.plugins import projector

## ONLY FOR BASIC ARRAY MANIPULATION
import numpy as np

## import class that calculates and returns the particle metrics
from lucid_utils_custom.classification.lucid_algorithm_data import classify
##from lucid_utils.classification.lucid_algorithm_data import classify

################################################
## ONLY NEED THIS LIBRARY FOR SHUFFLING DATA
import tflearn
from tflearn.data_utils import shuffle
################################################

import sqlite3
## CONNECT TO LUCID TRAINER DATABASE
conn = sqlite3.connect('./classified.db')
c = conn.cursor()

# Neural network expected outputs
particle_outputs = {
    "gamma":    [1, 0, 0, 0, 0, 0],
    "beta":     [0, 1, 0, 0, 0, 0],
    "muon":     [0, 0, 1, 0, 0, 0],
    "proton":   [0, 0, 0, 1, 0, 0],
    "alpha":    [0, 0, 0, 0, 1, 0],
    "others":   [0, 0, 0, 0, 0, 1]
}
# List of particle classes
vis_classes = ["gamma", "beta", "muon", "proton", "alpha", "others"]

# Lists to be used as training data
inputs = []
```

```
targets = []

## read and preprocess data from database
for row in c.execute('SELECT * FROM blobs'):
    r_id = row[0]
    r_frame = row[1]
    r_channel = row[2]
    r_blob = eval(row[3])
    r_class = row[4]
    #print(r_blob)
    r_data = classify(r_blob)
    print(r_data)
    ## NORMALISE LIST - not required (tested during design)
##    props_normal = list(map(lambda x: normalize_input(x, min_vals[r_data.index(x)], max_vals[r_data.index(x)]), r_data))
##    print(props_normal)
##    inputs.append(props_normal)
    inputs.append(r_data)

    tar = particle_outputs[r_class]
    targets.append(tar)


## NEURAL NETWORK ARCHITECTURE
def multilayer_perceptron(x, weights, biases):
    x = tf.nn.dropout(x, 1.0)

    with tf.name_scope('layer_1') as scope:
      # Hidden layer with sigmoid activation
      layer_1 = tf.add(tf.matmul(x, weights['w1']), biases['b1'])
      ##layer_1 = tf.nn.relu(layer_1)
      layer_1 = tf.nn.sigmoid(layer_1, name="sigmoid")
      layer_1 = tf.nn.dropout(layer_1, 1.0, name="dropout")

    ##tf.add_to_collection("layers", layer_1)

    with tf.name_scope('layer_2') as scope:
      # Hidden layer with sigmoid activation
      layer_2 = tf.add(tf.matmul(layer_1, weights['w2']), biases['b2'])
      ##layer_2 = tf.nn.relu(layer_2)
      layer_2 = tf.nn.sigmoid(layer_2, name="sigmoid")
      layer_2 = tf.nn.dropout(layer_2, 1.0, name="dropout")

    ##tf.add_to_collection("layers", layer_2)

    with tf.name_scope('layer_out') as scope:
      # Output layer with softmax activation
      ##out_layer = tf.matmul(layer_2, weights['out']) + biases['out']
      out_layer = tf.nn.softmax(tf.matmul(layer_2, weights['w_out']) + biases['b_out'], name="softmax")

    ##tf.add_to_collection("layers", out_layer)

    return out_layer

# Training and Test Data Splitter
from sklearn.model_selection import train_test_split

X = inputs
Y = targets
# Shuffle the data
X, Y = shuffle(X, Y)
X = np.array(X)
Y = np.array(Y)

##SPLIT SINGLE DATASET INTO A SPECIFIC RATIO
trX, teX, trY, teY = train_test_split(X, Y, test_size=0.1, random_state=1304)
# Used for visualisation
visX = X
visY = Y

# Parameters
learning_rate = 0.01/2
```

```
epochs = 100
##0.01 for 2000, 0.01/2 for 5000

##128, 48 - Worked nicely during design
# Network Parameters
n_hidden_1 = 128 # 1st layer number of nodes
n_hidden_2 = 48 # 2nd layer number of nodes
n_input = 8 # data input
n_classes = 6 # total classes

# Setup Tensorflow name scopes
with tf.name_scope('train_inputs') as scope:
  X = tf.placeholder("float", [None, n_input])
with tf.name_scope('train_outputs') as scope:
  Y = tf.placeholder("float", [None, n_classes])

tf.add_to_collection("X", X)
tf.add_to_collection("Y", Y)

## INITIALISE WEIGHTS
with tf.name_scope('weights') as scope:
  weights = {
      'w1': tf.Variable(tf.random_normal([n_input, n_hidden_1]), name="w1"),
      'w2': tf.Variable(tf.random_normal([n_hidden_1, n_hidden_2]), name="w2"),
      'w_out': tf.Variable(tf.random_normal([n_hidden_2, n_classes]), name="w_out")
  }
  tf.summary.histogram("w1", weights["w1"])
  tf.summary.histogram("w2", weights["w2"])
  tf.summary.histogram("w_out", weights["w_out"])

## INITIALISE BIASES
with tf.name_scope('biases') as scope:
  biases = {
      'b1': tf.Variable(tf.zeros(n_hidden_1), name="b1"),
      'b2': tf.Variable(tf.zeros(n_hidden_2), name="b2"),
      'b_out': tf.Variable(tf.zeros(n_classes), name="b_out")
  }
  tf.summary.histogram("b1", biases["b1"])
  tf.summary.histogram("b2", biases["b2"])
  tf.summary.histogram("b_out", biases["b_out"])

## CREATE NEURAL NETWORK USING WEIGHTS AND BIASES
py_x = multilayer_perceptron(X, weights, biases)

## CALCULATE CROSS ENTROPY COST SCOPE
with tf.name_scope('cross_entropy_cost') as scope:
  cross_entropy = tf.reduce_sum(- Y * tf.log(py_x), 1)
  cost = tf.reduce_mean(cross_entropy, name="cross_entropy_mean")

## GRADIENT DESCENT COST OPTIMISER
train_op = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
##train_op = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)
##train_op = tf.train.RMSPropOptimizer(0.001, 0.9).minimize(cost)
tf.add_to_collection("train_op", train_op)

## TRAINING ACCURACY SCOPE
with tf.name_scope('training_accuracy') as scope:
  correct_prediction = tf.equal(tf.argmax(py_x, 1), tf.argmax(Y, 1))
  accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

## CLASS PROBABILITY SCOPE
with tf.name_scope('class_probability') as scope:
  full_op = py_x
  tf.add_to_collection("full_op", full_op)

## CLASS PREDICTION SCOPE
with tf.name_scope('class_prediction') as scope:
  predict_op = tf.argmax(py_x, 1)
  tf.add_to_collection("predict_op", predict_op)

# Create a summary for our cost and accuracy
tf.summary.scalar("Loss", cost)
tf.summary.scalar("Training Accuracy", accuracy)
```

```
# Merge all summaries into a single "operation" which we can execute in a session
summary_op = tf.summary.merge_all()

# Log directory where all working files are saved
LOG_DIR = "./tf_models/"

# Launch the graph in a session
with tf.Session() as sess:
    # Initialize all variables
    sess.run(tf.global_variables_initializer())
    # Iraining data embedding
    EMB = np.zeros((len(visX), len(vis_classes)), dtype='float32')
    writer = tf.summary.FileWriter(LOG_DIR, graph=tf.get_default_graph())
    ###############################################################
    ## RESTORE MODEL
    saver = tf.train.Saver()
    checkpoint = tf.train.latest_checkpoint(LOG_DIR)
    if checkpoint:
        print ("Restoring from checkpoint", checkpoint)
        saver.restore(sess, checkpoint)
    else:
        print ("No checkpoint found. Starting over.")
    ###############################################################
    ## TRAINING FOR A NUMBER OF EPOCHS
    for i in range(epochs):
        ## TRAIN ON TRAINING DATA
        for start, end in zip(range(0, len(trX), 128), range(128, len(trX)+1, 128)):
            _, train_summary = sess.run([train_op, summary_op], feed_dict={X: trX[start:end], Y: t
rY[start:end]})

        # Run prediction on training data
        train_predict = sess.run(predict_op, feed_dict={X: trX})
        # Calculate training accuracy
        train_acc = np.mean(np.argmax(trY, axis=1) == train_predict)
        # Run prediction on test data
        predictor = sess.run(predict_op, feed_dict={X: teX})
        # Calculate test accuracy
        predict_acc = np.mean(np.argmax(teY, axis=1) == predictor)
        # Prints results for each epoch
        print(i, train_acc, predict_acc)
        # Save summary data for visualisation
        test_summary = tf.Summary(value=[tf.Summary.Value(tag="Test Accuracy", simple_value=predic
t_acc)])
        writer.add_summary(test_summary, i)
        writer.add_summary(train_summary, i)

    ## GETTING PROBABILITIES OF EACH CLASS FOR EACH PARTICLE IN TEST DATA
    ## Used for visualising the results on the test data
    for i in range(len(teX)):
        EMB[i] = sess.run(full_op, feed_dict={X: [visX[i]]})
        if (i % 20 == 0):
            print(i)
    ## Testing embedding variable
    #print(EMB)
    ## RUNNING PREDICTION ON TEST DATA
    print(sess.run(predict_op, feed_dict={X: teX}))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
    print("Accuracy:", accuracy.eval({X:teX, Y:teY}))

    ######################  TENSORBOARD VISUALISATION  ########################
    # Create metadata file for Tensorboard visualisation
    metadata_file = open(os.path.join(LOG_DIR, 'metadata.tsv'), 'w')
    metadata_file.write('Name\tClass\n')
    # Write visualisation data to the file
    for i in range(len(visX)):
        ind = np.where(visY[i] == 1)[0][0]
        metadata_file.write('%06d\t%s\n' % (i, vis_classes[ind]))
    metadata_file.close()

    # The embedding variable needs to be stored
    # This will allow all of the training data to be visualised
    embedding_var = tf.Variable(EMB,  name='particle_map')
    sess.run(embedding_var.initializer)
```

```
    config = projector.ProjectorConfig()
    embedding = config.embeddings.add()
    embedding.tensor_name = embedding_var.name

    # Save metadata for Tensorboard visualisation
    embedding.metadata_path = os.path.join('metadata.tsv')
    projector.visualize_embeddings(writer, config)
    ################################################################################

    # Save all variables (weights and biases)
    names_to_vars = {v.op.name: v for v in tf.global_variables()}
    saver = tf.train.Saver(names_to_vars)
    # This is the model file which contains the weights
    save_path = saver.save(sess, LOG_DIR+"model5.ckpt")

# This is the tensorflow graph
meta_graph_def = tf.train.export_meta_graph(filename=LOG_DIR+'model5.meta')
```

Model Saving

Instead of training the neural network every time, the neural network was saved as a model which could be used to classify a particle track without having to retrain it.

Neural Model Prediction

```
#!/usr/bin/env python
import os
import tensorflow as tf
import numpy as np
from lucid_utils.classification.lucid_algorithm_data import classify

# Classes can be renamed but they must stay in the same order
classes = ["gamma", "beta", "muon", "proton", "alpha", "others"]

def predict(cluster):
    tf.reset_default_graph()
    # Model directory and path initialisation
    this_dir, this_filename = os.path.split(__file__)
    model_name = '\model5.meta'
    LOG_DIR = os.path.join(this_dir, "Model")

    checkpoint = tf.train.latest_checkpoint(LOG_DIR)
    # Launch the graph in a session
    with tf.Session() as sess:
        # Initialise all variables
        sess.run(tf.global_variables_initializer())
        # Import model
        saver = tf.train.import_meta_graph(LOG_DIR+model_name)
        # Restore model weights from checkpoint
        saver.restore(sess, checkpoint)
        X = tf.get_collection("X")[0]
        # Get model operations
        predict_op = tf.get_collection('predict_op')[0]
        full_op = tf.get_collection('full_op')[0]
        # Run prediction operation on cluster metrics and return class index
        index = sess.run(predict_op, feed_dict={X: [classify(cluster)]})[0]
        return classes[index]
```

Model Evaluation

To compare the predictions of the neural network with other machine learning classifiers, I used a library to simply generate a set of classifiers which were created only to evaluate the neural network. The theory behind these classifiers however was researched but only to understand their applications and not to develop them from scratch. The classifiers that I tested were: Support Vector Machine, K-Nearest Neighbours, Random Forest and Decision Tree. Below is the code for creating the classifiers.

Other Machine Learning Classifiers

```
1.  import pickle
2.  import numpy as np
3.  from lucid_utils.classification.lucid_algorithm_data import classify
4.  ##returns [self.num_pixels, self.density, self.radius, self.curvature_radius, self.line_residual,
    self.circle_residual, self.width, self.avg_neighbours]
5.  from sklearn import metrics
6.  from sklearn.neighbors import KNeighborsClassifier
7.  from sklearn.svm import SVC
8.  from sklearn.tree import DecisionTreeClassifier
9.  from sklearn.ensemble import RandomForestClassifier
10.
11. import sqlite3
12. conn = sqlite3.connect('./classified.db')
13. c = conn.cursor()
14.
15. ## EXPECTED PARTICLE OUTPUTS
16. particle_outputs = {
17.     "gamma": [0.0],
18.     "beta": [0.2],
19.     "muon": [0.4],
20.     "proton": [0.6],
21.     "alpha": [0.8],
22.     "others": [1.0]
23. }
24.
25. inputs = []
26. targets = []
27. ## READ DATABASE INTO ARRAYS OF INPUTS AND OUTPUTS
28. for row in c.execute('SELECT * FROM blobs'):
29.     r_id = row[0]
30.     r_frame = row[1]
31.     r_channel = row[2]
32.     r_blob = eval(row[3])
33.     r_class = row[4]
34.     ##print(r_blob)
35.     r_data = classify(r_blob)
36.     r_data = [ int(x) for x in r_data ]
37.
38.     inputs.append(r_data)
39.     tar = particle_outputs[r_class][0]*10
40.     targets.append(tar)
41.
42. inputs = np.array(inputs)
43. targets = np.array(targets)
44. train_size = 1200
45.
46. ## SPLIT DATA INTO TRAINING AND TEST DATA
47. trX = inputs[:train_size]
48. trY = targets[:train_size]
49. teX = inputs[train_size:]
50. teY = targets[train_size:]
51.
52. coords = [[104,238],[104,239],[105,237],[105,238],[105,236],[106,236],[106,237],[106,235],[107,235
    ],[107,234],[108,234],[108,235],[108,233],[108,232],[109,232],[109,231],[109,230],[110,230],[110,2
    29],[111,229],[111,228],[111,227],[112,227],[112,226],[113,225],[113,224],[114,224],[114,223],[115
    ,222],[115,221],[116,221],[116,222],[114,220],[115,220],[116,220],[113,219],[113,220],[116,219],[1
    17,219],[117,218],[118,218],[118,217],[118,216],[119,216],[119,215],[120,214],[120,213],[121,213],
    [121,212],[122,211],[122,210],[123,210],[122,209],[123,209],[121,208],[122,208],[124,208],[121,207
    ],[124,207],[125,207],[121,206],[125,206],[121,205],[126,205],[121,204],[126,204],[127,204],[121,2
    03],[127,203],[120,202],[121,202],[127,202],[128,202],[128,201],[129,200],[129,199],[130,199],[130
    ,198],[131,197],[131,196],[132,196],[132,195],[133,194],[133,193],[134,193],[134,192],[135,193],[1
    35,194],[134,191],[135,191],[135,195],[136,195],[133,190],[134,190],[135,190],[136,190],[136,196],
    [133,189],[136,189],[132,188],[133,188],[137,188],[132,187],[133,187],[137,187],[138,187],[138,188
    ],[138,189],[132,186],[137,186],[138,186],[138,190],[131,185],[132,185],[136,185],[137,185],[138,1
    85],[139,185],[138,191],[130,184],[130,185],[130,186],[131,184],[136,184],[137,184]]
53. test = classify(coords)
54. test = [ int(x) for x in test ]
55. print(test)
56.
57. log_dir = "./benchmark_classifiers/"
```

```
58.
59. ############### TRAIN BENCHMARK CLASSIFIERS ###############
60. svm = SVC()
61. svm.fit(trX, trY)
62. with open(log_dir+'svm.pkl', 'wb') as f:
63.     pickle.dump(svm, f)
64.
65. knn = KNeighborsClassifier(n_neighbors=6)
66. knn.fit(trX, trY)
67. with open(log_dir+'knn.pkl', 'wb') as f:
68.     pickle.dump(knn, f)
69.
70. dt = DecisionTreeClassifier()
71. dt.fit(trX, trY)
72. with open(log_dir+'dt.pkl', 'wb') as f:
73.     pickle.dump(dt, f)
74.
75. rf = RandomForestClassifier()
76. rf.fit(trX, trY)
77. with open(log_dir+'rf.pkl', 'wb') as f:
78.     pickle.dump(rf, f)
79. ############################################################
80.
81. ## QUICK PREDICTION TESTS
82. print(svm.predict(test)/10)
83. print(knn.predict(test)/10)
84. ##print(knn.predict_proba(test))
85. print(dt.predict(test)/10)
86. print(rf.predict(test)/10)
87.
88. ## TESTING ON TRAINING DATA DURING DEVELOPMENT TO CHECK IT WAS EVEN WORKING
89. ##predicted = [svm.predict(x)[0] for x in trX]
90. ##print(metrics.classification_report(trY, predicted))
91. ##print(metrics.confusion_matrix(trY, predicted))
92. ##
93. ##predicted2 = [knn.predict(x)[0] for x in trX]
94. ##print(metrics.classification_report(trY, predicted2))
95. ##print(metrics.confusion_matrix(trY, predicted2))
96. ##
97. ##predicted3 = [dt.predict(x)[0] for x in trX]
98. ##print(metrics.classification_report(trY, predicted3))
99. ##print(metrics.confusion_matrix(trY, predicted3))
100. ##
101. ##predicted4 = [rf.predict(x)[0] for x in trX]
102. ##print(metrics.classification_report(trY, predicted4))
103. ##print(metrics.confusion_matrix(trY, predicted4))
104.
105. ## FULL TEST REPORT ON TEST DATA FOR ALL CLASSIFIERS
106. predicted = [svm.predict(x)[0] for x in teX]
107. print(metrics.classification_report(teY, predicted))
108. print(metrics.confusion_matrix(teY, predicted))
109.
110. predicted2 = [knn.predict(x)[0] for x in teX]
111. print(metrics.classification_report(teY, predicted2))
112. print(metrics.confusion_matrix(teY, predicted2))
113.
114. predicted3 = [dt.predict(x)[0] for x in teX]
115. print(metrics.classification_report(teY, predicted3))
116. print(metrics.confusion_matrix(teY, predicted3))
117.
118. predicted4 = [rf.predict(x)[0] for x in teX]
119. print(metrics.classification_report(teY, predicted4))
120. print(metrics.confusion_matrix(teY, predicted4))
```

LUCID Dashboard

Dashboard

The LUCID Dashboard is an online web application designed to display LUCID data in a viewable, easy-to-analyse format, providing various methods of looking through all of the collected data files. I made the application using PHP as the backend scripting language and, HTML (Hypertext Mark-up Language), JS (JavaScript) and CSS (Cascading Style Sheet) for serving it as a user-friendly website in a web browser. The fundamental structure of the web application was based on the user interface of an existing admin dashboard [29] complying with the MIT license.

The front-end of the dashboard needed to fulfil the following criteria:

- Display the latest LUCID data frames from the most recent data file
- Display the metadata of the data file
- Provide features that allow data analysis – (XYC for Energy Values, Energy Filter, Particle Type and Counts)

The details of the frame can also be viewed by clicking the info button and this shows the geo-location of the capture on a Google Maps popup. The XYC file of each frame is analysed using the saved neural network model and the counts of each particle are then displayed under the PNG frame. For fast page-loading, this analysis feature was disabled however the implementation clearly shows the broad uses of the neural network.

Filename: dashboard.php

```
<?php include("init.php"); ?>
<!doctype html>
<html lang="en">
<head>
    <base href="<?php echo $base_url; ?>">
    <meta charset="utf-8" />
    <link rel="icon" type="image/png" href="<?php echo $favicon; ?>">
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
    <title>LUCID Dashboard</title>
    <meta content='width=device-width, initial-scale=1.0, maximum-scale=1.0, user-
scalable=0' name='viewport' />
    <meta name="viewport" content="width=device-width" />
    <meta name="theme-color" content="<?php echo $nav_theme_colour; ?>">
```

```
    <link href="assets/css/bootstrap.min.css" rel="stylesheet" />
    <link href="assets/css/animate.min.css" rel="stylesheet"/>
    <link href="assets/css/light-bootstrap-dashboard.css" rel="stylesheet"/>
    <link href="assets/css/demo.css" rel="stylesheet" />
    <link href="assets/css/pe-icon-7-stroke.css" rel="stylesheet" />
    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-
awesome.min.css" rel="stylesheet">
    <link href='https://fonts.googleapis.com/css?family=Roboto:400,700,300'
rel='stylesheet' type='text/css'>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
    <link rel="stylesheet"
href="https://code.jquery.com/ui/1.11.4/themes/smoothness/jquery-ui.css">
    <script src="https://code.jquery.com/ui/1.11.4/jquery-ui.js"></script>
    <script src="assets/js/bootstrap-checkbox-radio-switch.js"></script>
    <script src="assets/js/light-bootstrap-dashboard.js"></script>
    <script type="text/javascript"
src="https://maps.googleapis.com/maps/api/js?sensor=false&key=<?php echo
$GLOBALS['google_maps_key']; ?>"></script>

    <?php
    if (isset($_GET['iframe'])){
        $form_action = "view/iframe";
        ?>
        <style>
        .main-panel{
            width: 100%;
            height: 100%;
        }
        </style>
        <?php
    }else{
        $form_action = "view";
    }
    ?>
</head>

<?php include("utils.php"); ?>
<?php
if (isset($_GET["id"])){
    $datafile_id = $_GET["id"];
}else{
    $datafile_id = get_datafiles()[0]["id"];
}

$datafile = get_datafile($datafile_id);

//MAX INDEX is TOTAL - 1 and first index is 0
$datafile_framecount = count($datafile);

$frame = isset($_GET["frame"]) ? $_GET["frame"] : 0;

if ($frame == 0){
    $next_frame = $frame+1;
    $previous_frame = $frame;
}elseif($frame == ($datafile_framecount - 1)){
    $next_frame = $datafile_framecount - 1;
    $previous_frame = $datafile_framecount - 2;
}else{
    $next_frame = $frame+1;
    $previous_frame = $frame - 1;
}

$datafile_frame_channels = array_keys($datafile[$frame]["channels"]);
$datafile_frame_channels_pixels = $datafile[$frame]["channels"];
$datafile_frame_timestamp = $datafile[$frame]["timestamp"];
```

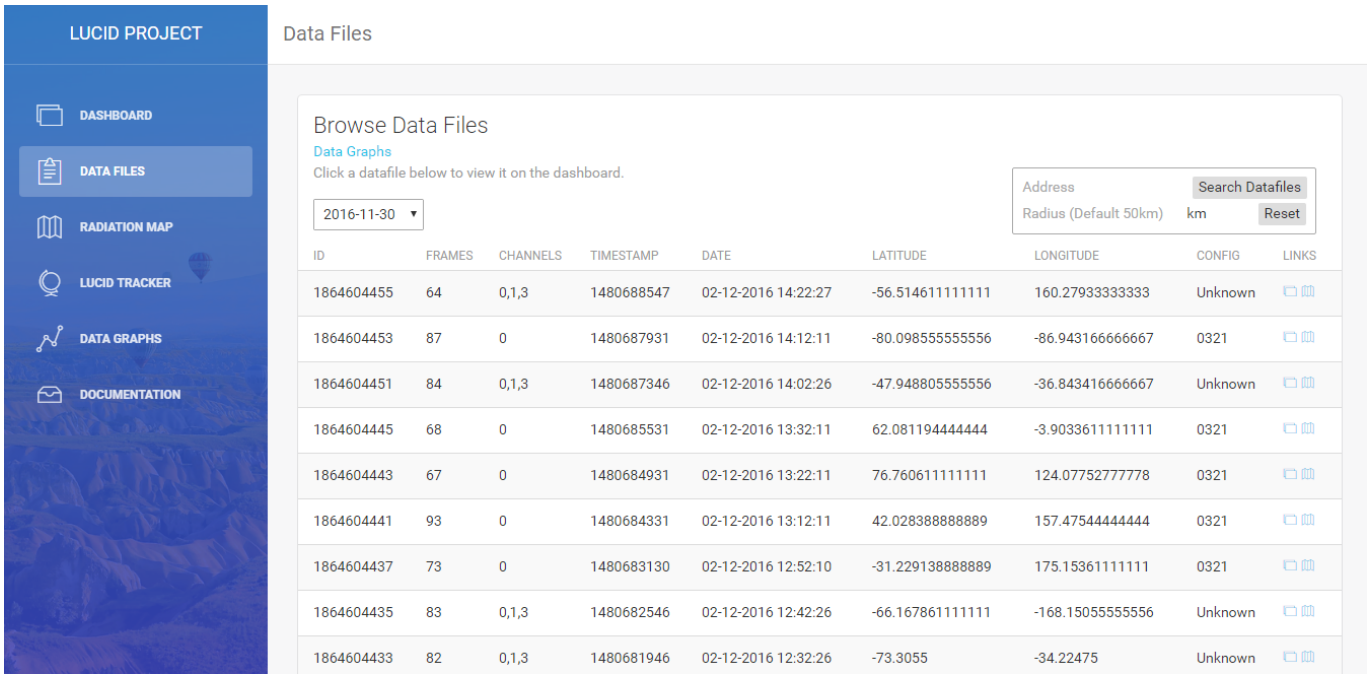
```
$datafile_frame_lat = $datafile[$frame]["latitude"];
$datafile_frame_lng = $datafile[$frame]["longitude"];

?>

<script>
var dataMarker;
var map;
function initialize() {
    var mapOptions = {
      center: { lat: <?php echo $datafile_frame_lat; ?>, lng: <?php echo
$datafile_frame_lng; ?> },
      zoom: 3,
      minZoom: 1,
      mapTypeControl: true,
      disableDefaultUI: false
    };
    map = new google.maps.Map(document.getElementById('map-canvas'),
        mapOptions);
    dataMarker = new google.maps.Marker({
        map: map,
        position: {lat: <?php echo $datafile_frame_lat; ?>, lng: <?php echo
$datafile_frame_lng; ?>}
    });
    dataMarker.info =  new google.maps.InfoWindow({
        content: "<strong>File ID -> Frame:</strong> <?php echo $datafile_id; ?> ->
<?php echo $frame; ?> </br> <strong>Available Channels:</strong> <?php echo
implode(",",$datafile_frame_channels); ?> </br> <strong>Date:</strong> <?php echo
date("d-m-Y H:i:s",$datafile_frame_timestamp); ?> </br> ( <?php echo
$datafile_frame_timestamp; ?> )</br> <strong>Position:</strong> <?php echo
round($datafile_frame_lat, 3); ?>, <?php echo round($datafile_frame_lng, 3); ?>",
        map: map,
        position: dataMarker.getPosition(),
    });
    dataMarker.info.close();
    google.maps.event.addListener(dataMarker, 'click', function() {
        this.info.open(map, this);
    });
}

$( document ).ready(function() {
    $('#dataModal').on('shown.bs.modal', function () {
        google.maps.event.trigger(map, "resize");
        initialize();
    });
    initialize();

    $( "#slider-range" ).slider({
    // orientation: "vertical",
      range: true,
        step: 10,
      min: 0,
      max: 1000,
      values: [ 0, 1000 ],
      slide: function( event, ui ) {
        $( "#energy_range" ).val( ui.values[ 0 ] + " - " + ui.values[ 1 ] );
        d = new Date();
        $('.frame').each(function() {
            var frame = $(this);
            var url_params = "?id=" + frame.attr('data-id') + "&frame=" +
frame.attr('data-frame') + "&channel=" + frame.attr('data-channel') +
'&min_e='+ui.values[ 0 ] + "&max_e=" + ui.values[ 1 ];

            <?php
            if (!isset($_GET['iframe'])){
            ?>
                frame.find('.dl-link-png-e').attr('href',
"gen_image.php"+url_params);
```

```
                frame.find('.dl-link-xyc-e').attr('href', "gen_xyc.php"+url_params);
            <?php
            }
            ?>
            frame.find('.frame_tpx').attr('src', "gen_image.php"+url_params);
        });
    }
});
    $( "#energy_range" ).val( $( "#slider-range" ).slider( "values", 0 ) + " - " +
$( "#slider-range" ).slider( "values", 1 ) );
});
</script>

<style>
.ui-slider-horizontal .ui-state-default {background: white; border-radius: 50%;
height: 16px; width: 16px;}
.ui-slider-range {background: #007FBF;}
.ui-slider {height: 8px;}
</style>

<body>
<div class="modal fade" id="dataModal" role="dialog">
<div class="modal-dialog">
  <div class="modal-content">
    <div class="modal-header">
      <button type="button" class="close" data-dismiss="modal">&times;</button>
      <h4 class="modal-title">Frame Details</h4>
    </div>
    <div class="modal-body">
      <div id="map-canvas" style="width: 100%; height: 450px;"></div>
    </div>
  </div>
</div>
</div>

<div class="wrapper">
    <?php $page = "Dashboard"; ?>
    <?php if (!isset($_GET["iframe"])){ include("sidebar.php"); } ?>

    <div class="main-panel">
        <?php if (!isset($_GET["iframe"])){ include("navbar.php"); } ?>
        <div class="content">
            <div class="container-fluid">
                <div class="row" style="margin-bottom:-50%;">
                    <div class="col-md-12">
                        <div class="card">
                            <div class="header">
                                <h4 class="title">Latest LUCID Data</h4>
                                <p class="category"><?php echo 'Frame '.($frame+1).'
of '.$datafile_framecount; ?>
                                <?php
                                $runs = get_runs();
                                if (!isset($_GET['iframe'])){?>
                                      |  
                                    <a href="map/<?php echo get_datarun($runs,
$datafile_frame_timestamp); ?>/<?php echo $datafile_id; ?>">View Datafile on Map</a>
                                <?php } ?>
                                </p>
                                <p>
                                    <label for="energy_range">Energy Filter:</label>
                                    <input type="text" id="energy_range" readonly
style="border:0;">
                                </p>
                                <div id="slider-range"></div>
                            </div>
                            <div class="content">
                            <!--<div id="chartPreferences" class="ct-chart ct-
perfect-fourth">-->
```

```
                                <div id="data">
                                    <div id="data-inner">
                                    <?php
                                    foreach ($datafile_frame_channels as $channel){
                                        echo '<div class="frame"
id="tpx'.$channel.'" data-id="'.$datafile_id.'" data-frame="'.$frame.'" data-
channel="'.$channel.'">
                                        <img class="frame_tpx"
src="gen_image.php?id='.$datafile_id.'&frame='.$frame.'&channel='.$channel.'">';

                                        if (!isset($_GET['iframe'])){
                                            echo '
                                                <div class="download">
                                                <div class="download-mask"></div>
                                                <div class="download-content">
                                                <a
href="'.$datafile_id.'_'.$frame.'_'.$channel.'.png" class="dl-link dl-link-
png">PNG</a>
                                                    
                                                <a
href="'.$datafile_id.'_'.$frame.'_'.$channel.'.txt" class="dl-link dl-link-
xyc">XYC</a>
                                                </br></br>
                                                <a class="dl-link-e dl-link-png-e"
href="gen_image.php?id='.$datafile_id.'&frame='.$frame.'&channel='.$channel.'">PNG
(Filtered)</a>
                                                    
                                                <a class="dl-link-e dl-link-xyc-e"
href="gen_xyc.php?id='.$datafile_id.'&frame='.$frame.'&channel='.$channel.'">XYC
(Filtered)</a>
                                                </div>
                                                </div>
                                                </div>';
                                        }else{
                                            echo '</div>';
                                        }
                                    }
                                    ?>
                                    <div id="pyxan-wrapper">
                                    <?php
                                    //Python XYC Analysis by calling an API a from
PHP function
                                    // foreach ($datafile_frame_channels as
$channel){
                                    // $pyxan = pyxan($datafile_id, $frame,
$channel);
                                    // $pyxan = str_replace('"', "", $pyxan);
                                    // $pyxan = str_replace('{', "", $pyxan);
                                    // $pyxan = str_replace('}', "", $pyxan);
                                    // // print_r($pyxan);
                                    // echo '<div class="pyxan">'.$pyxan.'</div>';
                                    // }
                                    ?>
                                    </div>
                                </div>

                                <hr style="width: 100%; height: 1px; display:inline-
block; margin: 30px 0px 0px;">
                            <div class="footer">
                                <div class="legend">
                                    <a href="<?php echo $form_action ?>/<?php echo
$datafile_id; ?>/<?php echo $previous_frame; ?>"><i class="pe-7s-angle-left-circle
text-info"></i></a>
                                    <p style="color:#DDD; display:inline; font-
size:40px; padding: 0px;"> | </p>
                                    <a data-toggle="modal" href="#dataModal"><i
class="pe-7s-info text-danger"></i></a>
```

```
                                    <p style="color:#DDD; display:inline; font-
size:40px; padding: 0px;"> | </p>
                                    <a href="<?php echo $form_action ?>/<?php echo
$datafile_id; ?>/<?php echo $next_frame; ?>"><i class="pe-7s-angle-right-circle
text-warning"></i></a>
                                </div>
                                <hr>
                                <div class="stats">
                                    <i class="fa fa-clock-o"></i> <?php echo
date('Y-m-d H:i:s', $datafile_frame_timestamp); ?>
                                      |  
                                    <i class="fa fa-map-marker"></i> <?php echo
round($datafile_frame_lat,3).', '.round($datafile_frame_lng,3); ?>
                                      |  
                                    <?php echo geocode(
round($datafile_frame_lat,3), round($datafile_frame_lng,3) ); ?>
                                </div>
                            </div>
                        <!-- </div>-->
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <?php if (!isset($_GET["iframe"])){ include("footer.php"); } ?>
</div>
</body>
</html>
```

Data Files



This page allows users to search for data files by their data run and also by the form of radial distance from a specific location. By clicking on the links in the link column, the user can be redirected to the dashboard or the radiation map to view their selected data file in further detail. All of the metadata about the data file that is stored in the server database is presented in a neat table for users to scroll through.

Filename: datafiles.php

```
<?php include("init.php"); ?>
<!doctype html>
<html lang="en">
<head>
    <base href="<?php echo $base_url; ?>">
    <meta charset="utf-8" />
    <link rel="icon" type="image/png" href="<?php echo $favicon; ?>">
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />

    <title>Data Files</title>

    <meta content='width=device-width, initial-scale=1.0, maximum-scale=1.0, user-
scalable=0' name='viewport' />
    <meta name="viewport" content="width=device-width" />
    <meta name="theme-color" content="<?php echo $nav_theme_colour; ?>">

    <!-- Bootstrap core CSS     -->
    <link href="assets/css/bootstrap.min.css" rel="stylesheet" />
    <!-- Animation library for notifications   -->
    <link href="assets/css/animate.min.css" rel="stylesheet"/>
    <!--  Light Bootstrap Table core CSS    -->
    <link href="assets/css/light-bootstrap-dashboard.css" rel="stylesheet"/>
        <link href="assets/css/pe-icon-7-stroke.css" rel="stylesheet" />

    <!--     Fonts and icons     -->
    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-
awesome.min.css" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css?family=Roboto:400,700,300"
rel='stylesheet' type='text/css'>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.2/jquery.min.js"></script>

    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
```

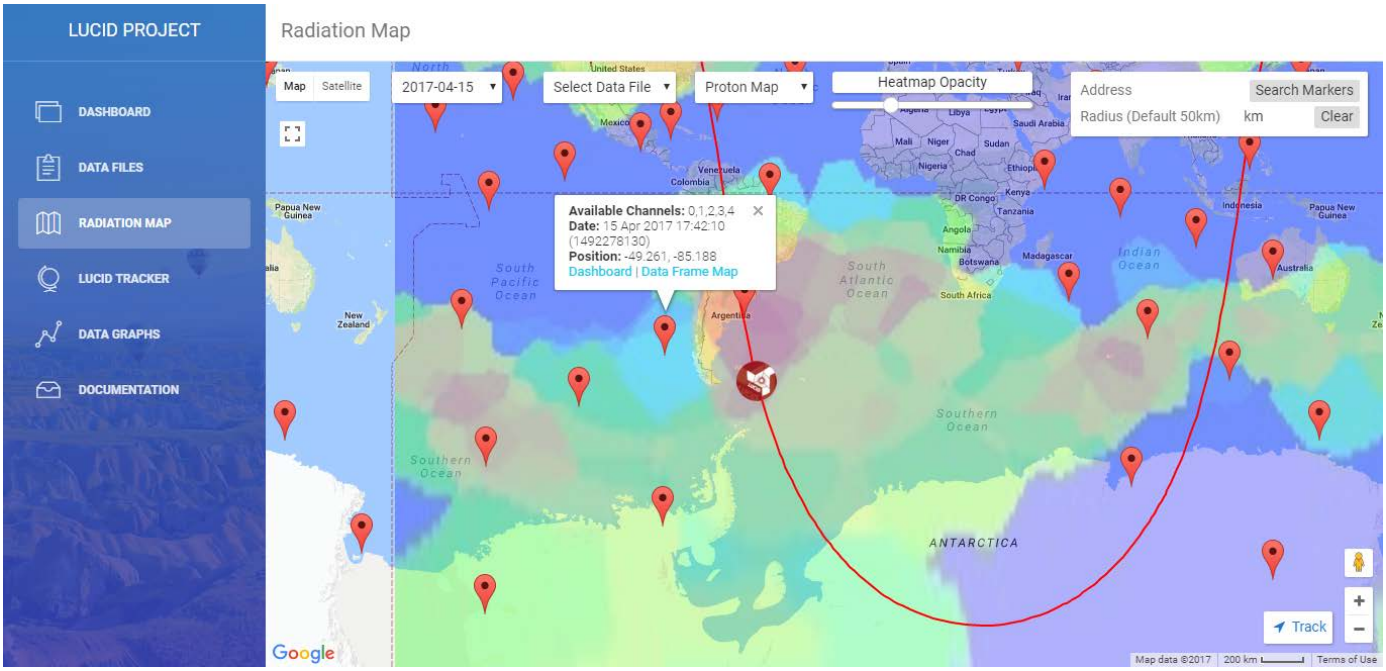
```
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>

</head>


<?php include("utils.php"); ?>
<?php
$init_radius = 50000;
$data_files = get_datafiles();
$runs = get_runs();
?>
<script>
    $( document ).ready(function() {
        $("tr").on('click', function(){
            link = $(this).attr('data-id');
            window.location.href = "view/"+link;
        });
        var form = document.getElementById('select_run'),
        input = document.getElementById('selected_run');

        input.onchange = function() {
            // navigate to the desired page
            window.location = form.action +'/'+ input.value;
        };

    });

    function updateInput(id,inp){
        document.getElementById(id).value = inp;
    }

    function clearAll(){
        document.getElementById('address').value = '';
        document.getElementById('radius').value = '';
        //redirect to selected run
        var form = document.getElementById('select_run');
        var url_action = form.action;
        var selected_run = document.getElementById('selected_run').value;
        window.location = url_action +'/'+ selected_run;
    }

    function codeAddress() {
      var address = document.getElementById('address').value;
      var rad_val = document.getElementById('radius').value;

      if ((/^[+]?([1-9][0-9]*(?:[\.][0-9]*)?|0*\.0*[1-9][0-9]*)(?:[eE][+-][0-
9]+)?$/).test(rad_val)){
        var radius = rad_val*1000;
      }else{
        var radius = <?php echo $init_radius; ?>;
      }

      var form = document.getElementById('select_run');
      var url_action = form.action;
      var selected_run = document.getElementById('selected_run').value;
      var address_encoded = encodeURIComponent(address);
      var radius_encoded = encodeURIComponent(radius);

      window.location = url_action +'/'+
selected_run+'/'+address_encoded+'/'+radius_encoded;
    }

</script>

<body>
<div class="wrapper">
    <?php $page = "Data Files"; ?>
```

```
    <?php include("sidebar.php"); ?>

    <div class="main-panel">
        <?php include("navbar.php"); ?>

        <div class="content">
            <div class="container-fluid">
                <div class="row">
                    <div class="col-md-12">
                        <div class="card">
                            <div class="header">
                                <h4 class="title">Browse Data Files</h4>
                                <p class="category">
                                    <a href="data/graph">Data Graphs</a>
                                    </br>
                                    Click a datafile below to view it on the
dashboard.
                                </p>
                                <!--<p class="category"><a
href="data/collection">Data Collection Graph</a>Select a filter ( Timestamp order,
Config, Available Channels / Number of channels ) COMING SOON</p>-->
                            </div>
                            <div class="content table-responsive table-full-width">
                                <form id="select_run" action="data" method="get">
                                    <select id="selected_run" name="selected_run"
data-role="data-run">
                                    <?php
                                    foreach ($runs as $run){
                                        if (!isset($_GET['selected_run']) and
array_values($runs)[0] == $run){
                                            $_GET['selected_run'] = $run;
                                        }
                                        if ($_GET['selected_run'] == $run){
                                        ?>
                                            <option value="<?php echo $run; ?>"
selected="selected"><?php echo $run; ?></option>
                                        <?php
                                        }else{
                                        ?>
                                            <option value="<?php echo $run;
?>"><?php echo $run; ?></option>
                                        <?php
                                        }
                                    }
                                    if ($_GET['selected_run']=="all"){
                                        // $_GET['selected_run'] = "none";
                                        $run_all = $_GET['selected_run'];
                                        ?>
                                        <option value="<?php echo $run_all; ?>"
selected="selected"><?php echo "All Datafiles"; ?></option>
                                        <?php
                                    }else{
                                        $run_all = "all";
                                        ?>
                                        <option value="<?php echo $run_all;
?>"><?php echo "All Datafiles"; ?></option>
                                        <?php
                                    }
                                    ?>
                                    </select>
                                    <div id="searchRadiusData">
                                      <input id="address" type="textbox"
placeholder="Address" onchange="updateInput(this.id, this.value)">
                                      <input type="button" value="Search Datafiles"
onclick="codeAddress()"><br>
                                      <input id="radius" type="textbox"
placeholder="Radius (Default 50km)" onchange="updateInput(this.id, this.value)">km
   
```

```
                                    <input type="button" value="Reset"
onclick="clearAll()">
                                </div>
                            </form>
                            <table class="table table-hover table-striped">
                                <thead>
                                    <th>ID</th>
                                    <th>Frames</th>
                                    <th>Channels</th>
                                    <th>Timestamp</th>
                                    <th>Date</th>
                                    <th>Latitude</th>
                                    <th>Longitude</th>
                                    <th>Config</th>
                                    <?php
                                    if (isset($_GET["address"])){
                                        ?>
                                        <th>Distance</th>
                                        <?php
                                    }
                                    ?>
                                    <th>Links</th>
                                </thead>
                                <tbody>
                                <?php
                                if (isset($_GET["address"])){
                                    $latlng_t =
geocode_address($_GET["address"]);
                                }
                                $radius = isset($_GET["radius"]) ?
urldecode($_GET["radius"]) : $init_radius;
                                foreach ($data_files as $datafile){
                                    if ($datafile["timestamp"] != 0){
                                        if ($_GET['selected_run'] == "all" or
$_GET['selected_run'] == $datafile["run"]){
                                            if (isset($_GET["address"])){
                                                if ($latlng_t != ''){
                                                    $dist =
distance($latlng_t['lat'], $latlng_t['lng'], $datafile['latitude'],
$datafile['longitude']);
                                                    if (($radius/1000) >=
$dist){
                                                    ?>
                                                        <tr data-id="<?php echo
$datafile['id']; ?>">
                                                            <td><?php echo
$datafile['id']; ?></td>
                                                            <td><?php echo
$datafile['num_frames']; ?></td>
                                                            <td><?php echo
$datafile['active_detectors']; ?></td>
                                                            <td><?php echo
$datafile['timestamp']; ?></td>
                                                            <td><?php echo
date("d-m-Y H:i:s",$datafile['timestamp']); ?></td>
                                                            <td><?php echo
$datafile['latitude']; ?></td>
                                                            <td><?php echo
$datafile['longitude']; ?></td>
                                                            <td><?php echo
$datafile['config']; ?></td>
                                                            <td><?php echo
round($dist,3); ?></td>
                                                            <td>
                                                            <a href="view/<?php
echo $datafile['id']; ?>"><i class="pe-7s-photo-gallery"></i></a>
```

```
                                                    <a href="map/<?php
echo get_datarun($runs, $datafile['timestamp']); ?>/<?php echo $datafile['id'];
?>"><i class="pe-7s-map"></i></a>
                                                    </td>
                                                </tr>
                                            <?php
                                            }
                                        }
                                    }else{
                                    ?>
                                        <tr data-id="<?php echo
$datafile['id']; ?>">
                                            <td><?php echo
$datafile['id']; ?></td>
                                            <td><?php echo
$datafile['num_frames']; ?></td>
                                            <td><?php echo
$datafile['active_detectors']; ?></td>
                                            <td><?php echo
$datafile['timestamp']; ?></td>
                                            <td><?php echo date("d-m-Y
H:i:s",$datafile['timestamp']); ?></td>
                                            <td><?php echo
$datafile['latitude']; ?></td>
                                            <td><?php echo
$datafile['longitude']; ?></td>
                                            <td><?php echo
$datafile['config']; ?></td>
                                            <td>
                                            <a href="view/<?php echo
$datafile['id']; ?>"><i class="pe-7s-photo-gallery"></i></a>
                                            <a href="map/<?php echo
get_datarun($runs, $datafile['timestamp']); ?>/<?php echo $datafile['id']; ?>"><i
class="pe-7s-map"></i></a>
                                            </td>
                                        </tr>
                                    <?php
                                    }
                                }
                            }
                        }

                        ?>
                                </tbody>
                            </table>

                        </div>
                    </div>
                </div>
            </div>
        </div>
        <?php include("footer.php"); ?>
    </div>
</div>
</body>
    <!--  Checkbox, Radio & Switch Plugins -->
    <script src="assets/js/bootstrap-checkbox-radio-switch.js"></script>
    <!-- Light Bootstrap Table Core javascript and methods -->
    <script src="assets/js/light-bootstrap-dashboard.js"></script>
</html>
```

Radiation Map



This page shows a pre-processed radiation map overlaid on the map of the Earth using the Google Maps API. Each marker is a data file in the selected run. Upon clicking the marker, the user is shown a small information window that shows the active detectors, the timestamp of when the file was captured and the geolocation. It also provides a link to the same data file on the dashboard and a link to all the frames of the data file on the same radiation map. LUCID is tracked by default however it can also be followed on the map by simply clicking the track button.

Filename: radiation_map.php

```
<?php include("init.php");?>
<!doctype html>
<html lang="en">
<head>
    <base href="<?php echo $base_url; ?>">
    <meta charset="utf-8" />
    <link rel="icon" type="image/png" href="<?php echo $favicon; ?>">
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />

    <title>Radiation Map</title>

    <meta content='width=device-width, initial-scale=1.0, maximum-scale=1.0, user-
scalable=0' name='viewport' />
    <meta name="viewport" content="width=device-width" />
    <meta name="theme-color" content="<?php echo $nav_theme_colour; ?>">

    <!-- Bootstrap core CSS     -->
    <link href="assets/css/bootstrap.min.css" rel="stylesheet" />
    <!-- Animation library for notifications   -->
    <link href="assets/css/animate.min.css" rel="stylesheet"/>
    <!--  Light Bootstrap Table core CSS    -->
    <link href="assets/css/light-bootstrap-dashboard.css" rel="stylesheet"/>
    <link href="assets/css/pe-icon-7-stroke.css" rel="stylesheet" />
    <!--     Fonts and icons     -->
    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-
awesome.min.css" rel="stylesheet">
    <link href='https://fonts.googleapis.com/css?family=Roboto:400,700,300'
rel='stylesheet' type='text/css'>

    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
```

```
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>

    <link rel="stylesheet"
href="https://code.jquery.com/ui/1.11.4/themes/smoothness/jquery-ui.css">
    <script src="https://code.jquery.com/ui/1.11.4/jquery-ui.js"></script>

    <!--  Google Maps Plugin    -->
    <script type="text/javascript"
src="https://maps.googleapis.com/maps/api/js?sensor=false&key=<?php echo
$GLOBALS['google_maps_key']; ?>"></script>

    <script src="./satellite-js-master/dist/satellite.min.js"></script>
    <script src="./satellite-js-master/dist/satellite.js"></script>
    <script src="./satellite-js-master/dist/orbits.js"></script>
    <script src="./satellite-js-master/dist/sidereal.js"></script>

    <?php
    if (isset($_GET['iframe'])){
        $form_action = "map/iframe";
        ?>
        <style>
        .main-panel{
            width: 100%;
            height: 100%;
        }
        select[data-role="rad-map-run"] {
            top: 10px;
        }
        select[data-role="rad-map-file"] {
            top: 10px;
        }
        select[data-role="rad-map-select"] {
            top: 10px;
        }
        #searchRadiusMap {
            top: 10px;
        }
        #heatmap_slider{
            top: 40px;
        }
        #map-canvas {width: 100%; height:100%;}
        </style>
        <?php
    }else{
        $form_action = "map";
        ?>
        <style>
        #map-canvas {width: 100%; height:90.5%;}
        </style>
        <?php
    }
    ?>

    <script>
      var track_lucid = false;
      var lucidMarker;
      var map;
      var circle;
      var geocoder = new google.maps.Geocoder();
      var gmarkers = [];
      var orbitLine;
      var lineDrawn = false;
      var overlay;
      var init_opacity = 0.3;

    function initialize() {
        var mapOptions = {
```

```
          center: { lat: 0, lng: 0},
          zoom: 2,
          minZoom: 2,
          disableDefaultUI: false,
          mapTypeControl: true,
          rotateControl: true,
          scaleControl: true,
          fullscreenControl: true,
          fullscreenControlOptions: {
                position: google.maps.ControlPosition.LEFT_TOP
            },
        };
        map = new google.maps.Map(document.getElementById('map-canvas'),
            mapOptions);

        var image = {
          url: "assets/img/lucidthumb.png",
          size: new google.maps.Size(40, 40),
          origin: new google.maps.Point(0, 0),
          anchor: new google.maps.Point(20, 20),
        };

        lucidMarker = new google.maps.Marker({
            map: map,
            icon: image,
            anchor: new google.maps.Point(20, 20)
        });

        lucidMarker.info =  new google.maps.InfoWindow({
            content: 'TDS-1 - LUCID',
            map: map,
            position: lucidMarker.getPosition(),
        });

        lucidMarker.info.close();

        // google.maps.event.addListener(lucidMarker, 'mouseover', function(event) {
              // // this.setIcon('./assets/img/lucidthumb.png');
              // this.info.open(map, this);
          // });
          // google.maps.event.addListener(lucidMarker, 'mouseout', function(event)
{
              // // this.setIcon('./assets/img/lucidthumb.png');
              // this.info.close();
          // });

        google.maps.event.addListener(lucidMarker, 'click', function() {
            this.info.open(map, this);
        });

        var bounds = new google.maps.LatLngBounds(
              new google.maps.LatLng(-85.0, -179.0),
              new google.maps.LatLng(85.0, 179.0));

        var srcImage = './maps/proton_map.png';
        var overlayOpts = {
            opacity: init_opacity
          }

        overlay = new google.maps.GroundOverlay(srcImage, bounds, overlayOpts);
        overlay.setMap(map);
    }

    function toggle_track_lucid(){
        if (track_lucid == true){
            track_lucid = false;
            $("#track_lucid").css("background","white");
            $("#track_lucid").css("color","#1F77D0");
```

```
        }else{
            track_lucid = true;
            $("#track_lucid").css("background","#1F77D0");
            $("#track_lucid").css("color","white");
        }
    }

    function setPosition(lat, lng) {
        // console.log(lat);
        // console.log(lng);
        newLatLng = new google.maps.LatLng(lat, lng);
        lucidMarker.setPosition(newLatLng);
        if (track_lucid == true){
            map.setCenter(newLatLng);
        }
    }

    function add_file_marker(lat, lng, timestamp, channels, id, run){
        marker = new google.maps.Marker({
          position: {lat: lat, lng: lng},
          map: map,
          title: id+' -> '+timestamp,
          // url: url
        });
        marker.info =  new google.maps.InfoWindow({
            content: "<strong>Available Channels:</strong> "+channels+" </br>
<strong>Date:</strong> "+timeConverter(timestamp)+" </br> ("+timestamp+") </br>
<strong>Position:</strong> "+ lat.toFixed(3) +", "+ lng.toFixed(3) +" </br> <?php if
(!isset($_GET['iframe'])) { echo "<a href='view/\"+id+\"'>Dashboard</a> | "; } ?><a
href='<?php echo $form_action; ?>/"+run+"/"+id+"'>Data Frame Map</a>",
            map: map,
            maxWidth: 500,
            position: marker.getPosition(),
        });
        marker.info.close();
        google.maps.event.addListener(marker, 'click', function() {
            this.info.open(map, this);
        });
        // google.maps.event.addListener(marker, 'click', function() {
            // window.location.href = this.url;
        // });
        return marker;
    }

    function add_frame_marker(lat, lng, timestamp, channels, datafile_id, frame_num,
frame_total){
        marker = new google.maps.Marker({
          position: {lat: lat, lng: lng},
          map: map,
          title: timestamp
        });
        var channels_array = channels.split(',');
        var arrayLength = channels_array.length;
        var img_cont = "";
        for (var i = 0; i < arrayLength; i++) {
            img_cont += "<img class='marker_frame'
src='"+datafile_id+"_"+(frame_num-1)+"_"+channels_array[i]+".png'></img></br></br>";
        }
        marker.info =  new google.maps.InfoWindow({
            content: "<strong>Frame:</strong> "+frame_num+"/"+frame_total+" </br>
<strong>Available Channels:</strong> "+channels+" </br> <strong>Date:</strong>
"+timeConverter(timestamp)+" </br> ("+timestamp+") </br> <strong>Position:</strong>
"+ lat.toFixed(3) +", "+ lng.toFixed(3) +" </br> <?php if (!isset($_GET['iframe']))
{ echo "<a href='view/\"+datafile_id+\"/\"+(frame_num-1)+\"'>View on Dashboard</a>
</br>";} ?> <div style='padding: 20px 60px 10px 20px;'>"+img_cont+"</div>",
            map: map,
            position: marker.getPosition(),
        });
```

```
        marker.info.close();
        google.maps.event.addListener(marker, 'click', function() {
            this.info.open(map, this);
        });
        gmarkers.push(marker);
        return marker;
    }

    function timeConverter(UNIX_timestamp){
          var a = new Date(UNIX_timestamp * 1000);
          var months =
['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec'];
          var year = a.getUTCFullYear();
          var month = months[a.getUTCMonth()];
          var date = a.getUTCDate();
          var hour = a.getUTCHours();
          var min = a.getUTCMinutes();
          var sec = a.getUTCSeconds();
          var time = date + ' ' + month + ' ' + year + ' ' + hour + ':' + min + ':'
+ sec ;
          return time;
        }
    //http://www.geocodezip.com/v3_MW_example_map3_radiusSearch.html
    </script>

    <!--  Checkbox, Radio & Switch Plugins -->
    <script src="assets/js/bootstrap-checkbox-radio-switch.js"></script>
    <!-- Light Bootstrap Table Core javascript and methods for Demo purpose -->
    <script src="assets/js/light-bootstrap-dashboard.js"></script>

</head>
<?php include("utils.php"); ?>

<style>
.ui-slider-horizontal .ui-state-default {background: white; border-radius: 50%;
height: 16px; width: 16px;}
.ui-slider {height: 8px; width: 200px;}

#map_canvas label { width: auto; display:inline; }
#map_canvas img { max-width: none; max-height: none; background-color: transparent;}
</style>


<body>
<?php
get_tle();
?>
<script>
    $( document ).ready(function() {
        var form = document.getElementById('select_data'),
        input = document.getElementById('selected_run');
        input2 = document.getElementById('selected_file');

        input.onchange = function() {
            // navigate to the desired page
            window.location = form.action +'/'+ input.value;
        };
        input2.onchange = function() {
            // navigate to the desired page
            window.location = form.action +'/'+ input.value +'/'+ input2.value;
        };

        $( "#heatmap_slider" ).slider({
          step: 0.01,
          min: 0,
          max: 1,
          value: init_opacity,
          slide: function( event, ui ) {
```

```
            overlay.setOpacity(ui.value);
          }
        });
    });

    function set_map_image(){
        map_select = document.getElementById("map_select");
        image = map_select.options[map_select.selectedIndex].value;
        overlay.set('url','./maps/'+image);
        overlay.setMap(map);
    }

    function updateInput(id,inp){
        document.getElementById(id).value = inp;
    }

    function clearCircle(){
        if (typeof circle !== 'undefined') {
            circle.setMap(null);
          }
    }

    function clearAll(){
        clearCircle();
        document.getElementById('address').value = '';
        document.getElementById('radius').value = '';
    }

    function codeAddress() {
      var address = document.getElementById('address').value;
      var rad_val = document.getElementById('radius').value;

      //regex to ensure only numbers are accepted (decimals allowed)
       if ((/^[+]?([1-9][0-9]*(?:[\.][0-9]*)?|0*\.0*[1-9][0-9]*)(?:[eE][+-][0-
9]+)?$/).test(rad_val)){
        var radius = rad_val*1000;
      }else{
        var radius = 50*1000;
      }
      geocoder.geocode( { 'address': address}, function(results, status) {
        if (status == google.maps.GeocoderStatus.OK) {
          var lat = results[0].geometry.location.lat();
          var lng = results[0].geometry.location.lng();

          var searchCenter = new google.maps.LatLng(lat, lng);

          // google.maps.event.trigger(map, 'resize');
          // map.setCenter(new google.maps.LatLng(lat, lng));

          // console.log(lat,lng);
          // var marker = new google.maps.Marker({
            // map: map,
            // position: results[0].geometry.location
          // });

          clearCircle();
          circle = new google.maps.Circle({center: searchCenter,
                                           radius: radius,
                                           strokeColor: '#FF0000',
                                           strokeOpacity: 0,
                                           strokeWeight: 0,
                                           fillOpacity: 0.35,
                                           fillColor: "#FF0000",
                                           map: map});
          var bounds = new google.maps.LatLngBounds();
          var foundMarkers;
          for (var i=0; i<gmarkers.length; i++) {
```

```
            if
((google.maps.geometry.spherical.computeDistanceBetween(gmarkers[i].getPosition(),se
archCenter) / 1000) < radius) {
              bounds.extend(gmarkers[i].getPosition());
              gmarkers[i].setMap(map);
              foundMarkers++;
            } else {
              gmarkers[i].setMap(null);
            }
          }
          if (foundMarkers > 0) {
              map.fitBounds(bounds);
            } else {
              map.fitBounds(circle.getBounds());
            }

            setTimeout(function() {
                map.setCenter(searchCenter);
            }, 100)

        } else {
          //alert('Geocode was not successful for the following reason: ' + status);
          console.log('Geocode was not successful for the following reason: ' +
status);
        }
      });
    }

  function getLucidPosition(nowdate){
        //console.log("Grabbing coordinates")
    var dat = document.getElementsByTagName("PRE")[0];
    var dat = dat.innerHTML;
    var dat = dat.split('\n');

    // Initialize a satellite record
    var satrec = satellite.twoline2satrec (dat[1], dat[2]);
    //  Propagate satellite using time since epoch (in sec).
    var position_and_velocity = satellite.propagate (satrec,
            nowdate.getUTCFullYear(),
            nowdate.getUTCMonth() + 1, // Note, this function requires months in
range 1-12.
            nowdate.getUTCDate(),
            nowdate.getUTCHours(),
            nowdate.getUTCMinutes(),
            nowdate.getUTCSeconds()
        );

    // The position_velocity result is a key-value pair of ECI coordinates.
    // These are the base results from which all other coordinates are derived.
    var position_eci = position_and_velocity["position"];
    // var velocity_eci = position_and_velocity["velocity"];

     var now = new Date();
    // You will need GMST for some of the coordinate transforms
    //var gmst = getGMST( now );
    var gmst = satellite.gstimeFromDate(
            now.getUTCFullYear(),
            now.getUTCMonth() + 1, // Note, this function requires months in range
1-12.
            now.getUTCDate(),
            now.getUTCHours(),
            now.getUTCMinutes(),
            now.getUTCSeconds()
        );

    // You can get ECF, Geodetic, Look Angles, and Doppler Factor.
    // var position_ecf   = satellite.eciToEcf (position_eci, gmst);
    // var observer_ecf   = satellite.geodeticToEcf (observer_gd);
```

```
    var position_gd    = satellite.eciToGeodetic (position_eci, gmst);
    // var look_angles    = satellite.ecfToLookAngles (observer_gd, position_ecf);
    //var doppler_factor = satellite.dopplerFactor (observer_ecf, position_ecf,
velocity_ecf);

    // The coordinates are all stored in key-value pairs.
    // ECI and ECF are accessed by "x", "y", "z".
    // var satellite_x = position_eci["x"];
    // var satellite_y = position_eci["y"];
    // var satellite_z = position_eci["z"];

    // Look Angles may be accessed by "azimuth", "elevation", "range_sat".
    // var azimuth   = look_angles["azimuth"];
    // var elevation = look_angles["elevation"];
    // var rangeSat  = look_angles["rangeSat"];

    // Geodetic coords are accessed via "longitude", "latitude", "height".
    var longitude = position_gd["longitude"];
    var latitude  = position_gd["latitude"];
    var height    = position_gd["height"];

    //  Convert the RADIANS to DEGREES for pretty printing (appends "N", "S", "E",
"W". etc).
    var lng = satellite.degreesLong (longitude);
    var lat = satellite.degreesLat  (latitude);
    return [lat, lng];
  }

  function grabData() {
      latlng = getLucidPosition(new Date());
      setPosition(latlng[0], latlng[1]);
      // $("#lat").text(lat);
      // $("#lng").text(lng);
      setTimeout('grabData()', 1000); // schedule for 1 sec...
  }

  function grabLine() {
    //console.log("Grabbing orbital line") // for testing
    var return_text = '';
    var ts = Math.floor(new Date().getTime() / 1000);

    var sub = 50 * 60;
    ts = ts - sub;

    for (i = 0; i < 100; i++) {
        // console.log(ts);
        var orig_date = new Date(ts*1000);
        latlng = getLucidPosition(orig_date);

        return_text += latlng[0].toString() + "," + latlng[1].toString();
        if (i < 99){
            return_text += ";";
        }
        ts += 60;
    }

          points = return_text.split(";");

          if (lineDrawn) {
            orbitLine.setMap(null);
          }

          var orbitLinepoints = [];
          $(points).each(function(index, point) {
            point = point.split(",");
            point[0] = Math.round(point[0] * 100) / 100;
            point[1] = Math.round(point[1] * 100) / 100; //rounding...
            orbitLinepoints.push(new google.maps.LatLng(point[0], point[1]));
```

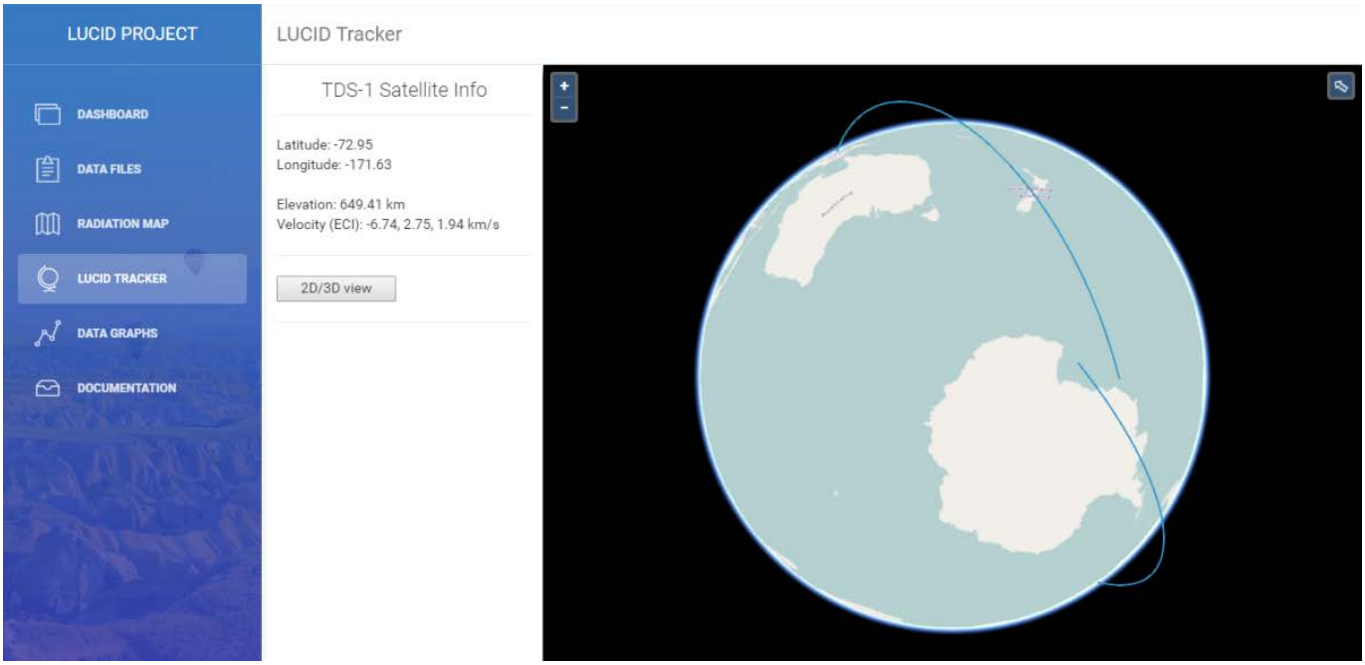
```
          });

          orbitLine = new google.maps.Polyline({
            path: orbitLinepoints,
            geodesic: true,
            strokeColor: '#FF0000',
            strokeOpacity: 1.0,
            strokeWeight: 2
          });

          orbitLine.setMap(map);
          lineDrawn = true;
          setTimeout('grabLine()', 1000); // schedule for 10 secs...
  }

  $(document).ready(function() {
    initialize();
    grabData();
    grabLine();
    latlng = getLucidPosition(new Date());
    newLatLng = new google.maps.LatLng(latlng[0], latlng[1]);
    map.setCenter(newLatLng);
  });
</script>
<?php
$runs = get_runs();
$files = get_datafiles();
?>
<div class="wrapper">
    <?php $page = "Radiation Map"; ?>
    <?php if (!isset($_GET["iframe"])){ include("sidebar.php"); } ?>

    <div class="main-panel" style="overflow: hidden;">
        <?php if (!isset($_GET["iframe"])){ include("navbar.php"); } ?>

        <div id="heatmap_slider">
        <p class="category">Heatmap Opacity</p>
        </div>

        <form id="select_data" action="<?php echo $form_action; ?>" method="get">
            <select id="selected_run" name="selected_run" data-role="rad-map-run">
            <?php
            if (!isset($_GET['selected_run'])){
                $_GET['selected_run'] = "none";
                $run_none = $_GET['selected_run'];
                ?>
                <option value="<?php echo $run_none; ?>" selected="selected"><?php
echo "Select Data Run"; ?></option>
                <?php
            }else{
                $run_none = "none";
                ?>
                <option value="<?php echo $run_none; ?>"><?php echo "None";
?></option>
                <?php
            }

            // INIT at start
            foreach ($runs as $run){
                if ($_GET['selected_run'] == $run){
                ?>
                    <option value="<?php echo $run; ?>" selected="selected"><?php
echo $run; ?></option>
                <?php
                }else{
                ?>
                    <option value="<?php echo $run; ?>"><?php echo $run; ?></option>
                <?php
```

```
                }
            }

            if ($_GET['selected_run']=="all"){
                // $_GET['selected_run'] = "none";
                $run_all = $_GET['selected_run'];
                ?>
                <option value="<?php echo $run_all; ?>" selected="selected"><?php
echo "All Datafiles"; ?></option>
                <?php
            }else{
                $run_all = "all";
                ?>
                <option value="<?php echo $run_all; ?>"><?php echo "All Datafiles";
?></option>
                <?php
            }

            ?>
            </select>

            <select id="selected_file" name="selected_file" data-role="rad-map-
file">
            <?php
            if (isset($_GET['selected_run'])){
                if (!isset($_GET['selected_file'])){
                    $_GET['selected_file'] = "none";
                    $file_none = $_GET['selected_file'];
                    ?>
                    <option value="<?php echo $file_none; ?>"
selected="selected"><?php echo "Select Data File"; ?></option>
                    <?php
                }else{
                    $file_none = "none";
                    ?>
                    <option value="<?php echo $file_none; ?>"><?php echo "None";
?></option>
                    <?php
                }

                // INIT at start
                foreach ($files as $file){
                    if ($file["run"] == $_GET["selected_run"]){
                        $file_id = $file['id'];
                        if ($_GET['selected_file'] == $file_id){
                        ?>
                            <option value="<?php echo $file_id; ?>"
selected="selected"><?php echo $file_id; ?></option>
                        <?php
                        }else{
                        ?>
                            <option value="<?php echo $file_id; ?>"><?php echo
$file_id; ?></option>
                        <?php
                        }
                    }elseif($_GET["selected_run"] == "all"){
                        $file_id = $file['id'];
                        if ($_GET['selected_file'] == $file_id){
                        ?>
                            <option value="<?php echo $file_id; ?>"
selected="selected"><?php echo $file_id; ?></option>
                        <?php
                        }else{
                        ?>
                            <option value="<?php echo $file_id; ?>"><?php echo
$file_id; ?></option>
                        <?php
                        }
```

```
                    }
                }
            }else{
                $file_none = "none";
                ?>
                <option value="<?php echo $file_none; ?>"><?php echo "None";
?></option>
                <?php
            }
            ?>
            </select>
        </form>

        <div id="searchRadiusMap">
          <input id="address" type="textbox" placeholder="Address"
onchange="updateInput(this.id, this.value)">
          <input type="button" value="Search Markers" onclick="codeAddress()"><br>
          <input id="radius" type="textbox" placeholder="Radius (Default 50km)"
onchange="updateInput(this.id, this.value)">km    
          <input type="button" value="Clear" onclick="clearAll()">
        </div>

        <select id="map_select" data-role="rad-map-select"
onchange="set_map_image()">
            <option value="proton_map.png" selected="selected">Proton Map</option>
            <option value="electron_map.png">Electron Map</option>
        </select>

        <!-- pe-7s-look pe-7s-compass pe-7s-paper-plane -->
        <div id="track_lucid" onclick="toggle_track_lucid()">
            <i class="fa fa-location-arrow" aria-hidden="true"></i> Track
        </div>

        <?php
        if ($_GET['selected_run'] != "none"){
            if ($_GET['selected_run'] != "all"){
                if ($_GET['selected_file'] == "none"){
                    //INIT at start
                    $data_files = $files;
                    foreach ($data_files as $datafile){
                        if ($datafile["timestamp"] != 0 and $datafile["run"] ==
$_GET['selected_run']){
                            $channels = $datafile["active_detectors"];
                            $timestamp = $datafile["timestamp"];
                            $dat_id = $datafile["id"];
                            echo "<script>
                            $(document).ready(function() {
                                add_file_marker(".$datafile["latitude"].",
".$datafile["longitude"].", '".$timestamp."', '".$channels."', '".$dat_id."',
'".$_GET['selected_run']."');
                            });
                            </script>";
                        }
                    }
                }else{
                    //INIT at start
                    $datafile = get_datafile($_GET['selected_file']);
                    $total_frames = count($datafile);
                    foreach ($datafile as $frame){
                        $channels = implode(',',array_keys($frame["channels"]));
                        $timestamp = $frame["timestamp"];
                        $frame_lat = $frame["latitude"];
                        $frame_lng = $frame["longitude"];
                        $frame_num = $frame["number"];
                        echo "<script>
                        $(document).ready(function() {
```

```
                            add_frame_marker(".$frame_lat.", ".$frame_lng.",
'".$timestamp."', '".$channels."', '".$_GET['selected_file']."', '".$frame_num."',
'".$total_frames."');
                        });
                        </script>";
                    }
                }
            }else{
                if ($_GET['selected_file'] == "none"){
                    //INIT at start
                    $data_files = $files;
                    foreach ($data_files as $datafile){
                        if ($datafile["timestamp"] != 0){
                            $channels = $datafile["active_detectors"];
                            $timestamp = $datafile["timestamp"];
                            $dat_id = $datafile["id"];
                            echo "<script>
                            $(document).ready(function() {
                                add_file_marker(".$datafile["latitude"].",
".$datafile["longitude"].", '".$timestamp."', '".$channels."', '".$dat_id."',
'".$_GET['selected_run']."');
                            });
                            </script>";
                        }
                    }
                }else{
                    //INIT at start
                    $datafile = get_datafile($_GET['selected_file']);
                    $total_frames = count($datafile);
                    foreach ($datafile as $frame){
                        $channels = implode(',',array_keys($frame["channels"]));
                        $timestamp = $frame["timestamp"];
                        $frame_lat = $frame["latitude"];
                        $frame_lng = $frame["longitude"];
                        $frame_num = $frame["number"];
                        echo "<script>
                        $(document).ready(function() {
                            add_frame_marker(".$frame_lat.", ".$frame_lng.",
'".$timestamp."', '".$channels."', '".$_GET['selected_file']."', '".$frame_num."',
'".$total_frames."');
                        });
                        </script>";
                    }
                }
            }
        }
        ?>
        <div id="map-canvas"></div>
    </div>
</div>
</body>
 </html>
```

LUCID Tracker



The LUCID tracker will mainly be used for demonstration purposes to show the location and trajectory of LUCID in 3D space. The user interface has been made such that the key details are presented on a side bar and the user if left to explore on the 3D globe.

Filename: track.php

```
<?php include("init.php"); ?>
<!doctype html>
<html lang="en">
<head>
    <base href="<?php echo $base_url; ?>">
    <meta charset="utf-8" />
    <link rel="icon" type="image/png" href="<?php echo $favicon; ?>">
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
    <title>LUCID Tracker</title>
    <meta content='width=device-width, initial-scale=1.0, maximum-scale=1.0, user-
scalable=0' name='viewport' />
  <meta name="viewport" content="width=device-width" />
    <meta name="theme-color" content="<?php echo $nav_theme_colour; ?>">
  <!-- Bootstrap core CSS     -->
  <link href="assets/css/bootstrap.min.css" rel="stylesheet" />
  <!-- Animation library     -->
  <link href="assets/css/animate.min.css" rel="stylesheet"/>
  <!--  Light Bootstrap Table core CSS    -->
  <link href="assets/css/light-bootstrap-dashboard.css" rel="stylesheet"/>
    <link href="assets/css/pe-icon-7-stroke.css" rel="stylesheet" />
    <!--     Fonts and icons     -->
    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-
awesome.min.css" rel="stylesheet">
    <link href='https://fonts.googleapis.com/css?family=Roboto:400,700,300'
rel='stylesheet' type='text/css'>
</head>
<body>
<div class="wrapper">
    <?php $page = "LUCID Tracker"; ?>
      <?php include("sidebar.php"); ?>
    <div class="main-panel" style="overflow:hidden;">
        <?php include("navbar.php"); ?>
        <?php include("lucid_track2.php"); ?>
    </div>
</div>
</body>
```

```
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
    <!--  Checkbox, Radio & Switch Plugins -->
    <script src="assets/js/bootstrap-checkbox-radio-switch.js"></script>
    <!--  Google Maps Plugin    -->
    <script type="text/javascript"
src="https://maps.googleapis.com/maps/api/js?sensor=false&key=AIzaSyDJlcv7E3JJl6ejN1
09-wFHm5E26HkCH_k"></script>
    <!-- Light Bootstrap Table Core javascript and methods for Demo purpose -->
    <script src="assets/js/light-bootstrap-dashboard.js"></script>
 </html>
```

Filename: lucid_track2.php (Included in track.php)

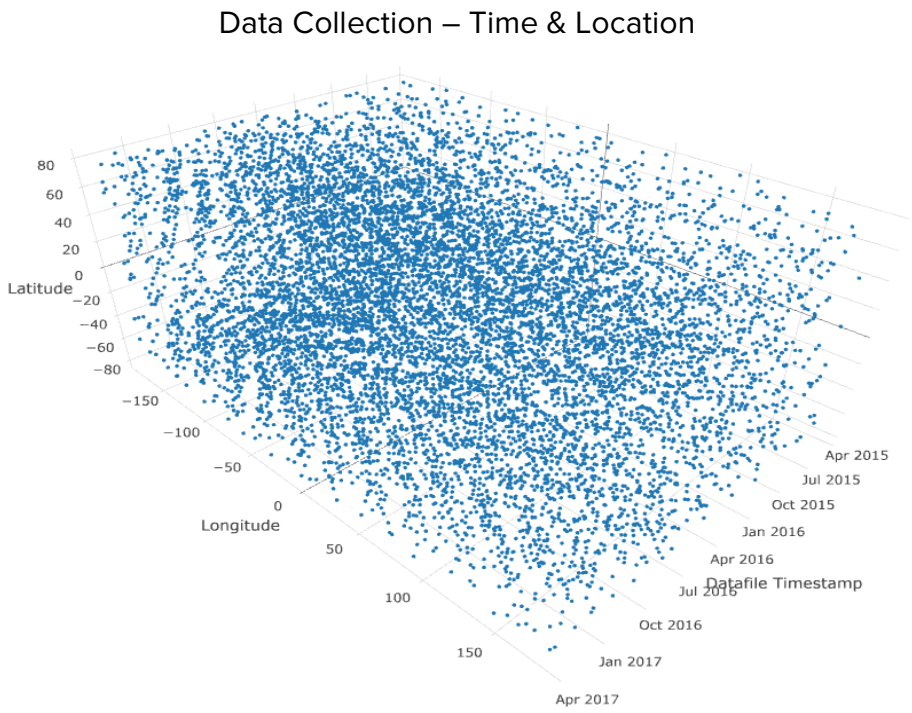
```
<?php
include("utils.php");
get_tle();
?>
<link href="https://openlayers.org/ol3-cesium/ol.css" rel="stylesheet"/>
<script src="https://openlayers.org/ol3-cesium/Cesium/Cesium.js"></script>
<script src="https://openlayers.org/ol3-cesium/ol3cesium.js"></script>
<script src="./satellite-js-master/dist/satellite.min.js"></script>
<script src="./satellite-js-master/dist/satellite.js"></script>
<script src="./satellite-js-master/dist/orbits.js"></script>
<script src="./satellite-js-master/dist/sidereal.js"></script>
<script src="https://code.jquery.com/jquery-1.9.1.min.js"></script>
<style>
.we-pm-icon{
    border-radius: 50%;
    background-image: url('http://starserver.thelangton.org.uk/lucid-data-
browser/static/img/lucid.jpg');
    background-size: contain;
    background-repeat: no-repeat;
}
#body {
    overflow: hidden;
}
#info{
    float:left;
    width:25%;
    height:100%;
    position:absolute;
    margin-left: 5px;
    padding: 10px;
    box-shadow: 7px 0 10px -7px #000;
    z-index:2;
    background: white;
}
#info input{
    width: 120px;
    display: inline;
}
#earth_div{
}
</style>
<script>

function update_loc(){
    var dat = document.getElementsByTagName("PRE")[0];
    var dat = dat.innerHTML;
    var dat = dat.split('\n');

    var satrec = satellite.twoline2satrec (dat[1], dat[2]);
```

```
    var nowdate = new Date();
    var position_and_velocity = satellite.propagate (satrec,
            nowdate.getUTCFullYear(),
            nowdate.getUTCMonth() + 1, // Note, this function requires months in
range 1-12.
            nowdate.getUTCDate(),
            nowdate.getUTCHours(),
            nowdate.getUTCMinutes(),
            nowdate.getUTCSeconds()
        );

    var position_eci = position_and_velocity["position"];
    var velocity_eci = position_and_velocity["velocity"];

     var now = new Date();
    // You will need GMST for some of the coordinate transforms
    //var gmst = getGMST( now );
    var gmst = satellite.gstimeFromDate(
            now.getUTCFullYear(),
            now.getUTCMonth() + 1, // Note, this function requires months in range
1-12.
            now.getUTCDate(),
            now.getUTCHours(),
            now.getUTCMinutes(),
            now.getUTCSeconds()
        );

    var position_gd    = satellite.eciToGeodetic (position_eci, gmst);
    var vel_ecf = satellite.eciToEcf(velocity_eci, gmst);

    // Geodetic coords are accessed via "longitude", "latitude", "height".
    var longitude = position_gd["longitude"];
    var latitude  = position_gd["latitude"];
    var height    = position_gd["height"];

    var lng = satellite.degreesLong (longitude);
    var lat = satellite.degreesLat  (latitude);
    return [[lat,lng],[height,vel_ecf]];
}


var satelliteTrackSource;
var satelliteTrackLayer;
var satelliteSource;
var satelliteLayer;
var ol2d;
var ol3d;

function update_marker(){
    // satelliteSource.clear();
    // satelliteLayer.setVisible(true);
    satelliteSource = new ol.source.Vector({
      format: new ol.format.GeoJSON(),
      // url: '//satellite.mediagis.com/?s=25544|28931'
      url: '//satellite.mediagis.com/?s=40076'
    });
    satelliteLayer.setSource(satelliteSource);
    // satelliteTrackSource.clear();
}

function initialize(){
    satelliteSource = new ol.source.Vector({
      format: new ol.format.GeoJSON(),
      // url: '//satellite.mediagis.com/?s=25544|28931'
      url: '//satellite.mediagis.com/?s=40076'
    });
    satelliteLayer = new ol.layer.Vector({
      source: satelliteSource
```

```
    });
    satelliteTrackSource = new ol.source.Vector({
      format: new ol.format.GeoJSON(),
      // url: '//satellite.mediagis.com/orbit/?s=25544|28931'
      url: '//satellite.mediagis.com/orbit/?s=40076'
    });
    satelliteTrackLayer = new ol.layer.Vector({
      source: satelliteTrackSource
    });
    // Init 2D map
    ol2d = new ol.Map({
      layers: [
        new ol.layer.Tile({
          source: new ol.source.OSM()
        }),
        satelliteLayer,
        satelliteTrackLayer
      ],
      controls: ol.control.defaults({
        attributionOptions: /** @type {olx.control.AttributionOptions} */ ({
          collapsible: false
        })
      }),
      target: 'map',
      view: new ol.View({
        // center: ol.proj.transform([0, 50], 'EPSG:4326', 'EPSG:3857'),
        center: ol.proj.transform([0, 50], 'EPSG:4326', 'EPSG:3857'),
        zoom: 1.5,
        minZoom: 1
      })
    });

    // Init 3D map
    ol3d = new olcs.OLCesium({map: ol2d});
    var scene = ol3d.getCesiumScene();
    var terrainProvider = new Cesium.CesiumTerrainProvider({
        url : '//assets.agi.com/stk-terrain/world'
    });
    scene.terrainProvider = terrainProvider;

    setInterval(function(){
        update_marker();
    }, 2000);

}
</script>
<body onload="initialize()">
<div id="map" style="float:right; width:74.4%; height:94%;"></div>
<div id='info'>
    <script>
    setInterval(function(){
        var pos = update_loc();
        var lat = Math.round(pos[0][0] * 100) / 100;
        var lng = Math.round(pos[0][1] * 100) / 100;
        var height = pos[1][0];
        var velocity = pos[1][1];
        document.getElementById("latlng").innerHTML = '<br>Latitude: '+lat+'
<br>Longitude: '+lng;
        document.getElementById("ev_data").innerHTML = '<br>Elevation:
'+Math.round(height * 100) / 100+' km<br>Velocity (ECI): '+Math.round(velocity.x *
100) / 100+', '+Math.round(velocity.y * 100) / 100+', '+Math.round(velocity.z * 100)
/ 100+' km/s';
    }, 500);
    // setInterval();
    </script>
    <h4 style="font-size:20px; margin-bottom: -10px; margin-top: 0px; text-align:
center;">TDS-1 Satellite Info</h4>
    <hr>
```

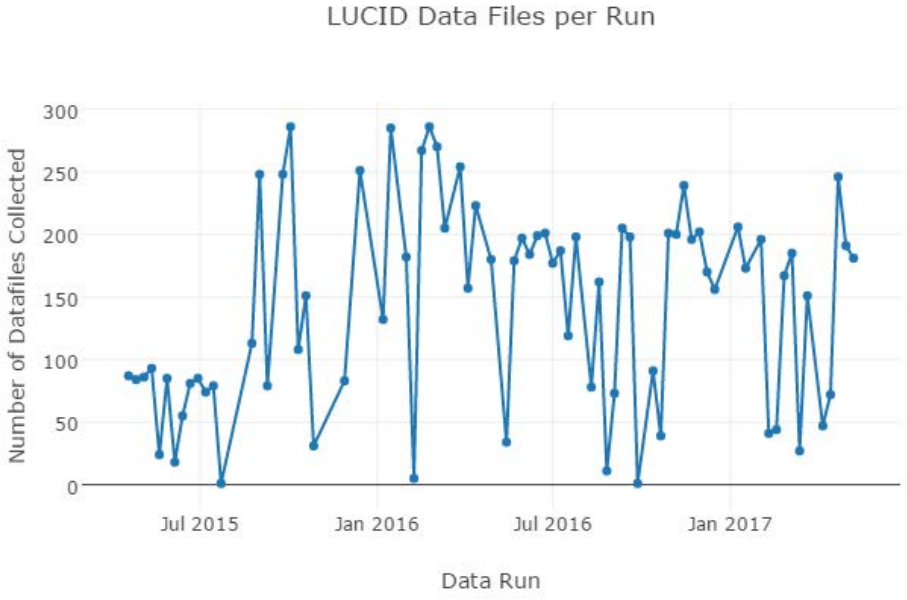
```
    <div id='latlng' style="margin-top: -20px;"></div>
    <div id='ev_data'></div>
    <hr>
    <input id="mapview" type="button" value="2D/3D view"
onclick="javascript:ol3d.setEnabled(!ol3d.getEnabled())" />
    <hr>
</div>
</body>
```

Data Graphs



It was important to be able to keep track of an overview of the data that was collected. To do this, the aim was to generate graphs for different aspects of the LUCID data so that users of the LUCID Dashboard can view these features easily.

The metadata of all the collected data files is processed on runtime using PHP and then converted to JSON so that it can be served into the Plotly JavaScript API.



Another example of a data graph is the plot of the data files collected per run as a time series graph.

All of these graphs are intended to provide real-time information about LUCID data collection.

Filename: data_graph.php

```php
<?php
include("init.php");
include("utils.php");

$graph_num = isset($_GET["g_num"]) ? $_GET["g_num"] : 0;

$graph_list = [0,1,2,3,4,5];
$graph_titles = ['LUCID Data Files per Run','LUCID Data Frames Per File','LUCID Data Frames Per File','Data Collection - Time & Location','Density of Datafiles in a Location','Data Location vs Number of Frames'];
//NOTE
//PLOTLY axes are wrong. y-axis is called z. x-axis is called y. z-axis is called x. (For 3d plots)

switch ($graph_num) {
//-----------------------------------------------------------------
    default:
        $graph_num = 0;

        $plotly_script = "
        <script>
            Plotly.d3.json('data_json.php?g_num=$graph_num', function(figure){
              var trace = {
                x: figure[0], y: figure[1],
                type: 'scatter', mode: 'lines+markers' //lines+markers is best (lines is 2nd)
              };
              var layout = {
                title: '$graph_titles[$graph_num]',
                yaxis:  {title : 'Number of Datafiles Collected'},
                xaxis: {title : 'Data Run'}
                };
              Plotly.plot(document.getElementById('line-plot'), [trace], layout, {showLink: false});
            });
            </script>
        ";
        break;

//-----------------------------------------------------------------
    case 1:

        $plotly_script = "
        <script>
            Plotly.d3.json('data_json.php?g_num=$graph_num', function(figure){
              var trace = {
                x: figure[0], y: figure[1],
                type: 'scatter', mode: 'markers' // lines(2nd) or markers (1st) are best
              };
              var layout = {
                title: '$graph_titles[$graph_num]',
                yaxis:  {title : 'Number of Frames Collected'},
                xaxis: { title : 'Data File Timestamps',
                    rangeselector: {
                        buttons: [
                            {   count:1,
                                label:'1m',
                                step:'month',
                                stepmode:'backward'},
                            {   count:6,
                                label:'6m',
                                step:'month',
                                stepmode:'backward'},
                            {   count:1,
```

```
                                label:'1y',
                                step:'year',
                                stepmode:'backward'},
                            {   step:'all'}
                        ],

                    },
                    rangeslider: {},
                    type: 'date'
                }
                };
              Plotly.plot(document.getElementById('line-plot'), [trace], layout,
{showLink: false});
            });
        </script>";
        break;

//---------------------------------------------------------------------
    case 2:

        $plotly_script = "
        <script>
            Plotly.d3.json('data_json.php?g_num=$graph_num', function(figure){
              var trace = {
                x: figure[0], y: figure[1],
                type: 'bar'
                // barmode: group, stack, relative
              };
              var layout = {
                title: '$graph_titles[$graph_num]',
                yaxis:  {title : 'Number of Frames Collected'},
                xaxis: {title : 'Datafile Timestamp',
                    rangeselector: {
                        buttons: [
                            {   count:1,
                                label:'1m',
                                step:'month',
                                stepmode:'backward'},
                            {   count:6,
                                label:'6m',
                                step:'month',
                                stepmode:'backward'},
                            {   count:1,
                                label:'1y',
                                step:'year',
                                stepmode:'backward'},
                            {   step:'all'}
                        ]
                    }
                }
              };
              Plotly.plot(document.getElementById('line-plot'), [trace], layout,
{showLink: false});
            });
        </script>";
        break;

//---------------------------------------------------------------------
    case 3:

        //3d plot xyz -- DATAFILE TIMESTAMPS, LAT, LNG
        $plotly_script = "
        <script>
            Plotly.d3.json('data_json.php?g_num=$graph_num', function(figure){
              var trace = {
                x: figure[0], y: figure[1], z: figure[2], text: figure[3],
                type: 'scatter3d', mode: 'markers',
                marker: {
```

```
                    //color: 'rgb(127, 127, 127)',
                    size: 2,
                    symbol: 'circle',
                    // line: {
                      // color: 'rgb(204, 204, 204)',
                      // width: 1
                    // },
                    opacity: 1
                  },
              };
              var layout = {
                title: '$graph_titles[$graph_num]',
                scene: {
                    xaxis: { title : 'Datafile Timestamp'},
                    yaxis: { title : 'Longitude'},
                    zaxis: { title : 'Latitude'},
                }
              };
              Plotly.plot(document.getElementById('line-plot'), [trace], layout,
{showLink: false});
            });
        </script>";
        break;

//-----------------------------------------------------------------
    case 4:

        //2d density plot -- LAT, LNG, Density of Datafiles collected
        $plotly_script = "
        <script>
        Plotly.d3.json('data_json.php?g_num=$graph_num', function(figure){
            var trace1 = {
              x: figure[1],
              y: figure[2],
              mode: 'markers',
              text: figure[3],
              //Datafiles as Points
              name: 'Datafile',
              marker: {
                color: 'rgb(100,0,0)',
                size: 2,
                opacity: 0.4
              },
              type: 'scatter'
            };
            var trace2 = {
              x: figure[1],
              y: figure[2],
              //Density of points as heatmap
              name: 'Datafile Density',
              ncontours: 14,
              colorscale: 'Hot',
              reversescale: true,
              showscale: true,
              type: 'histogram2dcontour'
            };
            var trace3 = {
              x: figure[1],
              //x is longitude, y is number of datafiles on hover
              name: '(Lng, Dat)',
              marker: {color: 'rgb(102,0,0)'},
              yaxis: 'y2',
              type: 'histogram'
            };
            var trace4 = {
              y: figure[2],
              //x is number of datafiles, y is latitude
              name: '(Dat, Lat)',
```

```
              marker: {color: 'rgb(102,0,0)'},
              xaxis: 'x2',
              type: 'histogram'
            };
            var data = [trace1, trace2, trace3, trace4];
            var layout = {
              title: '$graph_titles[$graph_num]',
              showlegend: false,
              hovermode: 'closest',
              bargap: 0,
              xaxis: {
                title: 'Longitude',
                domain: [0, 0.85],
                showgrid: false,
                zeroline: false
              },
              yaxis: {
                title: 'Latitude',
                domain: [0, 0.85],
                showgrid: false,
                zeroline: false
              },
              xaxis2: {
                domain: [0.85, 1],
                showgrid: false,
                zeroline: false
              },
              yaxis2: {
                domain: [0.85, 1],
                showgrid: false,
                zeroline: false
              }
            };
            Plotly.plot(document.getElementById('line-plot'), data, layout,
{showLink: false});
        });
        </script>";
        break;
//-----------------------------------------------------------------------
    case 5:

        //3d plot xyz -- NUMBER OF FRAMES, LAT, LNG -- TRACES
        $plotly_script = "
        <script>
            Plotly.d3.json('data_json.php?g_num=$graph_num', function(figure){
                var data = [];

                for (i = 0; i < figure[0][Object.keys(figure[0])[0]].length; i++) {
                    data.push({
                        x: figure[0][Object.keys(figure[0])[i]],
                        y: figure[1][Object.keys(figure[1])[i]],
                        z: figure[2][Object.keys(figure[2])[i]],
                        text: figure[3][Object.keys(figure[3])[i]],
                        name: Object.keys(figure[0])[i],
                        type: 'scatter3d', mode: 'markers',
                        //change type to lines+markers to view connections
                        marker: {
                            //color: 'rgb(127, 127, 127)',
                            size: 2,
                            symbol: 'circle',
                            // line: {
                              // color: 'rgb(204, 204, 204)',
                              // width: 1
                            // },
                            opacity: 1
                          },
                        // hoverinfo: 'x+y+z+text+name'
                    });
```

```
                    // console.log(data);
                }

                var layout = {
                  title: '$graph_titles[$graph_num]',
                  scene: {
                      //set tickformat and hoverformat
                      xaxis: { title : 'Number of Frames'},
                      yaxis: { title : 'Longitude'},
                      zaxis: { title : 'Latitude'},
                  },
                  // showlegend: true,
                  legend: {'orientation': 'v', 'traceorder': 'reversed'}
                };
                Plotly.plot(document.getElementById('line-plot'), data, layout,
{showLink: false});


                //ANNOTATIONS
              var divid = document.getElementById('line-plot');
              divid.on('plotly_click', function(data){
                    var point = data.points[0];
                  var newAnnotation = {
                      // x: point.xaxis,
                      // y: point.yaxis,
                      // z: point.zaxis,
                      x: 0,
                      y: 0,
                      ax: 40,
                      ay: 0,
                      arrowhead: 0,
                      // arrowhead: 6,
                      bgcolor: 'rgba(255, 255, 255, 0.9)',
                      // arrowcolor: point.fullData.marker.color,
                      font: {size:12},
                      bordercolor: point.fullData.marker.color,
                      borderwidth: 3,
                      borderpad: 4,
                      uniqid: point.data.text[point.pointNumber],
                      text: '<b>Data Run:</b> ' +(point.data.name) + '</br><b>Lat:</b>
'+ (point.z) + ' <b>Lng:</b> ' + (point.y) + '</br> <b>No. of Frames:</b>
'+(point.x) + '</br><a href=\"view/'+point.data.text[point.pointNumber]+'\">View on
Dashboard</a>'
                  };

                  newIndex = (divid.layout.annotations || []).length;
                  // console.log(point.pointNumber);

                  if(newIndex) {
                     var foundCopy = false;
                     divid.layout.annotations.forEach(function(ann, sameIndex) {
                         // console.log(ann);
                       if(ann.uniqid == newAnnotation.uniqid) {
                         Plotly.relayout(divid, 'annotations[' + sameIndex + ']',
'remove');
                         var foundCopy = true;
                       }else{
                          // Plotly.relayout(divid, 'annotations[' + newIndex + ']',
newAnnotation);
                          var foundCopy = false;
                       }
                     });
                     if(foundCopy) return;
                   }
                    Plotly.relayout(divid, 'annotations[' + newIndex + ']',
newAnnotation);
              });
        });
```

```
        </script>";
        break;
}

// }elseif ($graph_num == 6){
    // //scattergeo -- Maybe not needed due to radiation map -- Number of Frames
(Colour Gradient), LAT, LNG

// }elseif ($graph_num == 7){
    // //2d density plot -- NUMBER OF FRAMES, LAT, LNG -- DATA TIMESTAMP SLIDER
CONTROL

// }elseif ($graph_num == 8){
    // //heatmap

// }elseif ($graph_num == 9){
    // //histogram

// }elseif ($graph_num == 10){


//IDEAS:
// ADD up all C values in xycs of specific datafile and divide by 256*256 to get
average energy per pixel

//individual scatter lines (different colours) for each run (chart 1, 2)
//add multiple traces

//add name: "{run}" when displaying run on hover

//add dropdown to select the run or to select all

//onclick of data marker, shows text

?>

<html lang="en">
<head>
    <base href="<?php echo $base_url; ?>">
    <meta charset="utf-8" />
    <link rel="icon" type="image/png" href="<?php echo $favicon; ?>">
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />

    <title>Data Graphs</title>

    <meta content='width=device-width, initial-scale=1.0, maximum-scale=1.0, user-
scalable=0' name='viewport' />
    <meta name="viewport" content="width=device-width" />
    <meta name="theme-color" content="<?php echo $nav_theme_colour; ?>">

    <!-- Bootstrap core CSS     -->
    <link href="assets/css/bootstrap.min.css" rel="stylesheet" />
    <!-- Animation library for notifications   -->
    <link href="assets/css/animate.min.css" rel="stylesheet"/>
    <!--  Light Bootstrap Table core CSS    -->
    <link href="assets/css/light-bootstrap-dashboard.css" rel="stylesheet"/>

    <!--     Fonts and icons     -->
    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-
awesome.min.css" rel="stylesheet">
    <link href='https://fonts.googleapis.com/css?family=Roboto:400,700,300'
rel='stylesheet' type='text/css'>
    <link href="assets/css/pe-icon-7-stroke.css" rel="stylesheet" />

    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
```

```
    <link rel="stylesheet"
href="https://code.jquery.com/ui/1.11.4/themes/smoothness/jquery-ui.css">
    <script src="https://code.jquery.com/ui/1.11.4/jquery-ui.js"></script>

    <!--  Checkbox, Radio & Switch Plugins -->
    <script src="assets/js/bootstrap-checkbox-radio-switch.js"></script>

    <!-- Light Bootstrap Table Core javascript -->
    <script src="assets/js/light-bootstrap-dashboard.js"></script>

    <!-- Latest compiled and minified plotly.js JavaScript -->
    <script type="text/javascript" src="https://cdn.plot.ly/plotly-
latest.min.js"></script>

    <?php
    if (isset($_GET['iframe'])){
        $form_action = "data/graph/iframe";
        ?>
        <style>
        .main-panel{
            width: 100%;
            height: 100%;
        }
        select[data-role="data-graph"] {top: 10px;}
        #line-plot{position: absolute; top:0; width:100%; height:100%;}
        </style>
        <?php
    }else{
        $form_action = "data/graph";
        ?>
        <style>
        #line-plot{position: absolute; margin-top:-3px; top:10%; width:100%;
height:94%;}
        </style>
        <?php
    }
    ?>

    <script>
    $( document ).ready(function() {
        var form = document.getElementById('select_graph'),
        input = document.getElementById('selected_graph');
        // input2 = document.getElementById('selected_file');

        input.onchange = function() {
            // navigate to the desired page
            window.location = form.action +'/'+ input.value;
        };
        // input2.onchange = function() {
            // // navigate to the desired page
            // window.location = form.action +'/'+ input.value +'/'+ input2.value;
        // };
    });
    </script>
</head>
<body>
    <div class="wrapper">
        <?php $page = "Data Graphs"; ?>
        <?php if (!isset($_GET["iframe"])){ include("sidebar.php"); } ?>

        <div class="main-panel" style="overflow: hidden;">
            <?php if (!isset($_GET["iframe"])){ include("navbar.php"); } ?>
            <form id="select_graph" action="<?php echo $form_action; ?>"
method="get">
                <select id="selected_graph" name="g_num" data-role="data-graph">
                    <?php
                    // INIT at start
```

```
                    foreach ($graph_list as $graph){
                        if ($graph_num == $graph){
                        ?>
                            <option value="<?php echo $graph_num; ?>"
selected="selected"><?php echo "Graph ".$graph_num; ?></option>
                        <?php
                        }else{
                        ?>
                            <option value="<?php echo $graph; ?>"><?php echo "Graph
".$graph; ?></option>
                        <?php
                        }
                    }
                    ?>
                </select>
            </form>
            <!--<div class="uil-ripple-css"
style="transform:scale(2);"><div></div><div></div></div>-->
            <div class="uil-squares-css"
style="transform:scale(2);"><div><div></div></div><div><div></div></div><div><div></
div></div><div><div></div></div><div><div></div></div><div><div></div></div><div><di
v></div></div><div><div></div></div></div>
            <?php
            echo $plotly_script;
            ?>
            <div id="line-plot"></div>
        </div>
    </div>
</body>
</html>
```

Filename: data_json.php

This file supplies the data graphs page with the data in JSON format after having collected, parsed and processed the data.

```
<?php
include("init.php");
include("utils.php");

$graph_num = isset($_GET["g_num"]) ? $_GET["g_num"] : 0;

if ($graph_num == 0){

    $runs = get_runs();
    $runs = array_reverse($runs);
    $datafiles = get_datafiles();
    $runs_all = [];
    foreach ($datafiles as $datafile){
        if ($datafile['timestamp'] != 0){
            $runs_all[] = $datafile['run'];
        }
    }
    usort($runs_all, 'compareTime');
    $datafiles_per_run = array_values(array_count_values($runs_all));
    $average_datafile_per_run = array_sum($datafiles_per_run) / count($runs);

    //$average_datafile_per_run is extra info
    print_r (json_encode(array($runs, $datafiles_per_run,
$average_datafile_per_run)));

}elseif (in_array($graph_num, array(1,2))){

    $data = get_datafiles();
    foreach ($data as $datum){
```

```
        if ($datum["timestamp"] != 0){
            $timestamps[] = date("Y-m-d H:i:s",$datum['timestamp']);
            $nums[] = $datum["num_frames"];
        }
    }
    $average_dataframes_per_file = array_sum(array_values($nums)) /
count(array_values($timestamps));

    //$average_dataframes_per_file is extra info
    print_r (json_encode(array($timestamps, $nums, $average_dataframes_per_file)));

}elseif (in_array($graph_num, array(3,4))){

    $data = get_datafiles();
    foreach ($data as $datum){
        if ($datum["timestamp"] != 0){
            $timestamps[] = date("Y-m-d H:i:s",$datum['timestamp']);
            $lat[] = $datum["latitude"];
            $lng[] = $datum["longitude"];
            $ids[] = $datum["id"];
            $runs[] = $datum["run"];
        }
    }
    //ids is extra info
    print_r (json_encode(array($timestamps, $lng, $lat, $ids, $runs)));


}elseif (in_array($graph_num, array(5))){

    $data = get_datafiles();

    //multiple traces split up by the data run
    foreach ($data as $datum){
        if ($datum["timestamp"] != 0){
            $num_frames[$datum['run']][] = $datum['num_frames'];
            $lat[$datum['run']][] = $datum["latitude"];
            $lng[$datum['run']][] = $datum["longitude"];
            $ids[$datum['run']][] = $datum["id"];
            // $runs[$datum['run']][] = $datum["run"];
            $timestamps[$datum['run']][] = $datum["timestamp"];
        }
    }
    //$ids and $runs is extra info
    print_r (json_encode(array($num_frames, $lng, $lat, $ids, $timestamps)));

// // USE BELOW for 2d density plot
// }elseif (in_array($graph_num, array(5))){

    // $data = get_datafiles();
    // foreach ($data as $datum){
        // if ($datum["timestamp"] != 0){
            // $num_frames[] = $datum['num_frames'];
            // $lat[] = $datum["latitude"];
            // $lng[] = $datum["longitude"];
            // $ids[] = $datum["id"];
            // $runs[] = $datum["run"];
            // $timestamps[] = $datum["timestamp"];
        // }
    // }
    // //$ids and $runs is extra info
    // print_r (json_encode(array($num_frames, $lng, $lat, $ids, $runs,
$timestamps)));

}

?>
```

Documentation

XYC File URL

https://starserver.thelangton.org.uk/lucid_dashboard/id_frame_channel.txt
https://starserver.thelangton.org.uk/lucid_dashboard/747002715_0_0.txt

XYC Analysis

https://starserver.thelangton.org.uk/lucid_dashboard/pyxan/id_frame_channel.txt
https://starserver.thelangton.org.uk/lucid_dashboard/pyxan/747002715_0_0.txt

{"Alpha": 0, "Beta": 8, "Gamma": 5, "Muon": 0, "Other": 0, "Proton": 0}

https://starserver.thelangton.org.uk/lucid_dashboard/pyxan/xyc_input.html

Type your xyc here...

Submit

Dashboard Usage & API

Brief Outline of all the Features ([Optional] {Parameter} (Range) **Optimal)

TYPE	DATA	DESCRIPTION	SIMPLE USAGE	ADVANCED USAGE	EXAMPLE
API	PNG	Display the PNG of a given frame	*{id}* _ *{frame}* _ *{channel}* [_x *{scale(1-20)}*] [_s].png ** frame / *{id}* / *{frame}* / *{channel}* / [*{scale(1-20)}*] / [s] /	gen_image.php?id=*{id}*&frame=*{frame}*&channel=*{channel}* [&scale=*{scale(1-20)}*] [&min_e=*{min_e(0-11810)}*] [&max_e=*{max_e(0-11810)}*] [&simple]	1175485581_0_0_x4_s.png frame/1175485581/0/0/4/s gen_image.php? id=1175485581&frame=0&channel=0 &scale=4&min_e=10&max_e=100&simple
API	XYC	Display the XYC of a given frame	*{id}* _ *{frame}* _ *{channel}*.txt ** xyc / *{id}* / *{frame}* / *{channel}* /	gen_xyc.php?id=*{id}*&frame=*{frame}*&channel=*{channel}* [&min_e=*{min_e(0-11810)}*] [&max_e=*{max_e(0-11810)}*]	1175485581_0_0.txt xyc/1175485581/0/0 gen_xyc.php? id=1175485581&frame=0&channel=0 &min_e=10&max_e=100
IFrame	Map	Embed Radiation Map on another website - If the optional parameters are not provided then no markers are shown unless selected by dropdown.	/map/iframe/ **	/map/iframe / [*{selected_run}*] / [*{selected_file}*]	<iframe src='/map/iframe' width="100%" height="100%"></iframe>
IFrame	Dashboard	Embed Dashboard on another website - If the optional parameters are not provided then the latest data is displayed in the iframe.	/view/iframe/ **	/view/iframe / [*{id}*] / [*{frame}*] / [*{channel}*]	<iframe src='/view/iframe' width="100%" height="100%"></iframe>
IFrame	Data Graphs	Embed Data Graphs on another website - If the optional parameters are not provided then the first graph is displayed.	/data/graph/iframe/ **	/data/graph/iframe / [*{g_num}*] /	<iframe src='/data/graph/iframe' width="100%" height="100%"></iframe>

Radiation Map	Address & Radius	The map is zoomed to a translucent circle which shows the data files within the radius of the specified location. Note* The latitude and longitude of a datafile is the latitude and longitude of the first frame.	Any place name recognisable by Google Maps can be entered in the address box. The radius can be any positive numerical value including decimals.	If an invalid address then the circle will not be created. If the radius is not set or is invalid then the default value of 50km is used.	Address: Canterbury, Kent | Radius: 10 km
Data Files	Address & Radius	All the data files within the specified radius of a location from the selected run are returned.	Any place name recognisable by Google Maps can be entered in the address box. The radius can be any positive numerical value including decimals.	If an invalid address then no files will be returned. If the radius is not set or is invalid then the default value of 50km is used. / data / *{selected_run}* / *{address}* / [*{radius 'metres'}*] /	Address: Canterbury, Kent | Radius: 10 km /data/all/Canterbury%2C%20Kent/10000/

Filename: docs.php

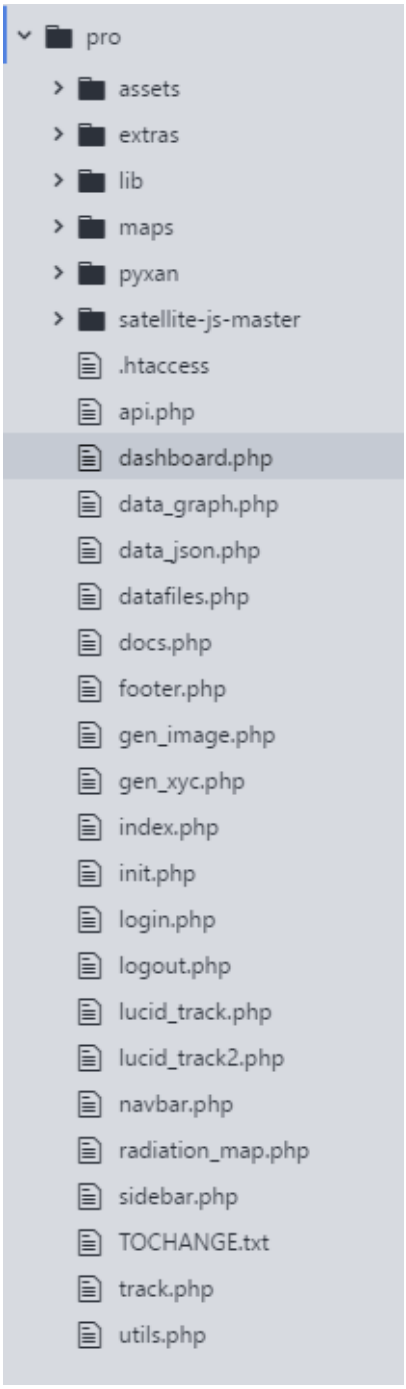
```
<?php include("init.php"); ?>
<!doctype html>
<html lang="en">
<head>
    <base href="<?php echo $base_url; ?>">
    <meta charset="utf-8" />
    <link rel="icon" type="image/png" href="<?php echo $favicon; ?>">
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />

    <title>Documentation</title>

    <meta content='width=device-width, initial-scale=1.0, maximum-scale=1.0, user-
scalable=0' name='viewport' />
    <meta name="viewport" content="width=device-width" />
    <meta name="theme-color" content="<?php echo $nav_theme_colour; ?>">

    <!-- Bootstrap core CSS     -->
    <link href="assets/css/bootstrap.min.css" rel="stylesheet" />
    <!-- Animation library for notifications   -->
    <link href="assets/css/animate.min.css" rel="stylesheet"/>
    <!--  Light Bootstrap Table core CSS    -->
    <link href="assets/css/light-bootstrap-dashboard.css" rel="stylesheet"/>

    <!--     Fonts and icons     -->
    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-
awesome.min.css" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css?family=Roboto:400,700,300"
rel='stylesheet' type='text/css'>
    <link href="assets/css/pe-icon-7-stroke.css" rel="stylesheet" />

    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.2/jquery.min.js"></script>

    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
</head>

<?php include("utils.php"); ?>
<body>
<div class="wrapper">
    <?php $page = "Documentation"; ?>
    <?php include("sidebar.php"); ?>
    <div class="main-panel">
```

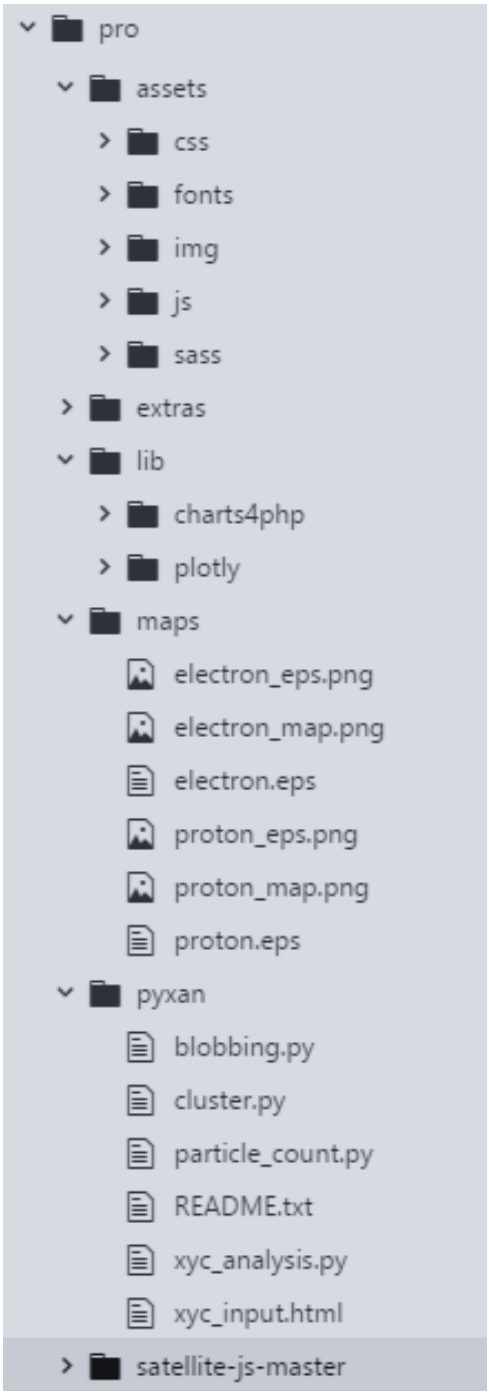
```
        <?php include("navbar.php"); ?>
        <div class="content">
            <div class="container-fluid">
                <div class="row">
                    <div class="col-md-12">
                        <div class="card">
                            <div class="header">
                                <h4 class="title">Dashboard Usage & API</h4>
                                <p class="category">
                                    Brief Outline of all the Features ( [Optional]
{Parameter} (Range) **Optimal )
                                </p>
                            </div>
                            <div class="content table-responsive table-full-width">
                                <table class="table table-hover table-striped">
                                    <thead>
                                        <th>Type</th>
                                        <th>Data</th>
                                        <th>Description</th>
                                        <th>Simple Usage</th>
                                        <th>Advanced Usage</th>
                                        <th>Example</th>
                                    </thead>
                                    <tbody style="font-size:12px;">
                                        <tr>
                                            <td>API</td>
                                            <td>PNG</td>
                                            <td>Display the PNG of a given
frame</td>
                                            <td><i>{id}</i> _ <i>{frame}</i> _
<i>{channel}</i> [_x <i>{scale(1-20)}</i> ] [_s].png ** </br></br> frame /
<i>{id}</i> / <i>{frame}</i> / <i>{channel}</i> / [ <i>{scale(1-20)}</i> ] / [s]
/</td>
                                            <td>

gen_image.php?id=<i>{id}</i>&frame=<i>{frame}</i>&channel=<i>{channel}</i>[&scale=<i
>{scale(1-20)}</i>]</br>[&min_e=<i>{min_e(0-11810)}</i>]</br>[&max_e=<i>{max_e(0-
11810)}</i>]</br>[&simple]
                                            </td>
                                            <td>1175485581_0_0_x4_s.png </br></br>
frame/1175485581/0/0/4/s </br></br>
gen_image.php?id=1175485581&frame=0&channel=0</br>&scale=4&min_e=10&max_e=100&simple
</td>
                                        </tr>
                                        <tr>
                                            <td>API</td>
                                            <td>XYC</td>
                                            <td>Display the XYC of a given
frame</td>
                                            <td><i>{id}</i> _ <i>{frame}</i> _
<i>{channel}</i>.txt ** </br></br> xyc / <i>{id}</i> / <i>{frame}</i> /
<i>{channel}</i> /</td>
                                            <td>

gen_xyc.php?id=<i>{id}</i>&frame=<i>{frame}</i>&channel=<i>{channel}</i></br>[&min_e
=<i>{min_e(0-11810)}</i>]</br>[&max_e=<i>{max_e(0-11810)}</i>]
                                            </td>
                                            <td>1175485581_0_0.txt </br></br>
xyc/1175485581/0/0 </br></br>
gen_xyc.php?id=1175485581&frame=0&channel=0</br>&min_e=10&max_e=100</td>
                                        </tr>
                                        <tr>
                                            <td>IFrame</td>
                                            <td>Map</td>
                                            <td>Embed Radiation Map on another
website - If the optional parameters are not provided then no markers are shown
unless selected by dropdown.</td>
                                            <td>/map/iframe/ **</td>
```

```
                                        <td>/map/iframe / [
<i>{selected_run}</i> ] / [ <i>{selected_file}</i> ] </td>
                                        <td> &lt;iframe src='/map/iframe'
width="100%" height="100%"&gt;&lt;/iframe&gt; </td>
                                    </tr>
                                    <tr>
                                        <td>IFrame</td>
                                        <td>Dashboard</td>
                                        <td>Embed Dashboard on another website -
If the optional parameters are not provided then the latest data is displayed in the
iframe.</td>
                                        <td>/view/iframe/ **</td>
                                        <td>/view/iframe / [ <i>{id}</i> ] / [
<i>{frame}</i> ] / [ <i>{channel}</i> ] </td>
                                        <td> &lt;iframe src='/view/iframe'
width="100%" height="100%"&gt;&lt;/iframe&gt; </td>
                                    </tr>
                                    <tr>
                                        <td>IFrame</td>
                                        <td>Data Graphs</td>
                                        <td>Embed Data Graphs on another website
- If the optional parameters are not provided then the first graph is
displayed.</td>
                                        <td>/data/graph/iframe/ **</td>
                                        <td>/data/graph/iframe / [
<i>{g_num}</i> ] / </td>
                                        <td> &lt;iframe src='/data/graph/iframe'
width="100%" height="100%"&gt;&lt;/iframe&gt; </td>
                                    </tr>
                                    <tr>
                                        <td>Radiation Map</td>
                                        <td>Address & Radius</td>
                                        <td>The map is zoomed to a translucent
circle which shows the data files within the radius of the specified location. Note*
The latitude and longitude of a datafile is the latitude and longitude of the first
frame.</td>
                                        <td>Any place name recognisable by
Google Maps can be entered in the address box. The radius can be any positive
numerical value including decimals.</td>
                                        <td>If an invalid address then the
circle will not be created. If the radius is not set or is invalid then the default
value of 50km is used.</td>
                                        <td>Address: Canterbury, Kent  |
Radius: 10 km</td>
                                    </tr>
                                    <tr>
                                        <td>Data Files</td>
                                        <td>Address & Radius</td>
                                        <td>All the data files within the
specified radius of a location from the selected run are returned.</td>
                                        <td>Any place name recognisable by
Google Maps can be entered in the address box. The radius can be any positive
numerical value including decimals.</td>
                                        <td>If an invalid address then no files
will be returned. If the radius is not set or is invalid then the default value of
50km is used.</br>/ data / <i>{selected_run}</i> / <i>{address}</i> / [ <i>{radius
'metres'}</i> ] /</td>
                                        <td>Address: Canterbury, Kent  |
Radius: 10 km </br>/data/all/Canterbury%2C%20Kent/10000/</td>
                                    </tr>
                                </tbody>
                            </table>

                        </div>
                    </div>
                </div>
            </div>
        </div>
```

```
        </div>
        <?php include("footer.php"); ?>
    </div>
</div>
</body>
    <!--  Checkbox, Radio & Switch Plugins -->
    <script src="assets/js/bootstrap-checkbox-radio-switch.js"></script>
    <!--  Google Maps Plugin    -->
    <script type="text/javascript"
src="https://maps.googleapis.com/maps/api/js?sensor=false"></script>
    <!-- Light Bootstrap Table Core javascript and methods for Demo purpose -->
    <script src="assets/js/light-bootstrap-dashboard.js"></script>
</html>
```

Full File Directory





The **pro** folder is the root of the project. The **assets** folder contains the CSS, JS, SASS, images and font files used to create the web application. The **extras** folder was simply a folder that was used during development to store old pages that are no longer used in the web application alongside ideas that I decided against implementing. The **lib** folder contains the PHP libraries that I was planning on using. However, during development, I found that I did not need the libraries as all the graphs could be created client-side using JavaScript with a simple file added to the html code. The **maps** folder contains the radiation map vectors and images and I created a separate folder for these as in the future I could automate the creation of these maps without affecting the other folders. The **pyxan** folder contains all the Python APIs for analysing the data. The **satellite-js-master** was a JavaScript library that was downloaded from GitHub to perform satellite tracking and other orbital functions.

Included PHP Files

Filename: init.php

```php
<?php
ini_set('memory_limit', -1);
set_time_limit(0);
date_default_timezone_set ( "UTC" );

$base_url = "/LUCID/pro/";
$favicon = "assets/img/favicon-96x96.png";

//works outside server
$lucid_api_root = "https://starserver.thelangton.org.uk/lucid_dashboard";
//alternative url
// $lucid_api_root = "https://starserver.thelangton.org.uk/lucid-data-browser";

$pyxan_root = "https://starserver.thelangton.org.uk/lucid_dashboard/pyxan";

//this will not work externally - only needed for making the api
$sqlite_db = "./db/data_browser.db";
// $sqlite_db = "../lucid-data-browser-python3/data_browser.db";

$google_maps_key = "AIzaSyDJlcv7E3JJl6ejN109-wFHm5E26HkCH_k";
$nav_theme_colour = "#1F77D0";
//AIzaSyCI4ZHMQCdRj7Z3tOgNhLVsW2baymkb0k4 -- Mine
// AIzaSyDJlcv7E3JJl6ejN109-wFHm5E26HkCH_k -- Cal


function curl_installed(){
    //return function_exists('curl_version');
    return false;
}

function file_get_contents_curl($url) {
    $ch = curl_init();

    curl_setopt($ch, CURLOPT_AUTOREFERER, TRUE);
    curl_setopt($ch, CURLOPT_HEADER, 0);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_FOLLOWLOCATION, TRUE);

    $data = curl_exec($ch);
    curl_close($ch);

    return $data;
}

?>
```

Filename: navbar.php

```php
<?php
?>
<nav class="navbar navbar-default navbar-fixed">
    <div class="container-fluid">
        <div class="navbar-header">
            <a class="navbar-brand" href="#"><?php echo $page; ?></a>
        </div>
        <div class="collapse navbar-collapse"></div>
    </div>
</nav>
<?php
?>
```

This navbar file is just an html file without any PHP however including it as a PHP file means that any changes in the navbar with PHP backend logic will automatically be update on every page.

Filename: sidebar.php

```
<?php
?>
<div class="sidebar" data-color="blue" data-image="assets/img/sidebar-2.jpg">
    <!--
        Tip 1: you can change the color of the sidebar using: data-color="blue |
azure | green | orange | red | purple"
        Tip 2: you can also add an image using data-image tag (2,5,4)
    -->
        <div class="sidebar-wrapper">
            <div class="logo">
                <a href="view" class="simple-text">
                    LUCID Project
                </a>
            </div>
            <ul class="nav">
                <li class="<?php echo ($page == "Dashboard" ? "active" : "")?>">
                    <a href="view">
                        <i class="pe-7s-photo-gallery"></i>
                        <!-- pe-7s-albums pe-7s-graph pe-7s-cloud pe-7s-photo-
gallery pe-7s-search-->
                        <p>Dashboard</p>
                    </a>
                </li>
                <li class="<?php echo ($page == "Data Files" ? "active" : "")?>">
                    <a href="data">
                        <i class="pe-7s-note2"></i>
                        <p>Data Files</p>
                    </a>
                </li>
                <li class="<?php echo ($page == "Radiation Map" ? "active" : "")?>">
                    <a href="map">
                        <i class="pe-7s-map"></i>
                        <!-- fa fa-map-o -->
                        <p>Radiation Map</p>
                    </a>
                </li>
                <li class="<?php echo ($page == "LUCID Tracker" ? "active" : "")?>">
                    <a href="tracker">
                        <i class="pe-7s-world"></i>
                        <!-- pe-7s-map-marker pe-7s-paper-plane pe-7s-global -->
                        <p>LUCID Tracker</p>
                    </a>
                </li>
                <li class="<?php echo ($page == "Data Graphs" ? "active" : "")?>">
                    <a href="data/graph">
                        <i class="pe-7s-graph1"></i>
                        <!-- pe-7s-graph pe-7s-graph1 pe-7s-graph2 pe-7s-graph3  -->
                        <p>Data Graphs</p>
                    </a>
                </li>
                <li class="<?php echo ($page == "Documentation" ? "active" : "")?>">
                    <a href="info">
                        <i class="pe-7s-drawer"></i>
                        <!-- pe-7s-notebook pe-7s-drawer  -->
                        <p>Documentation</p>
                    </a>
                </li>
            </ul>
        </div>
    </div>
<?php
?>
```

Filename: utils.php

```php
<?php
//ORDER BY TIMESTAMP
function compareOrder($a, $b){
  return $a['timestamp'] - $b['timestamp'];
}

// normal file_get_contents works too however curl has been used in case
file_get_contents is not allowed
function get_datafiles(){
    if (curl_installed()){
        $request =
file_get_contents_curl($GLOBALS['lucid_api_root']."/api/get/data_files");
    }else{
        $request =
file_get_contents($GLOBALS['lucid_api_root']."/api/get/data_files");
    }
    $data_files = json_decode($request, true);
    usort($data_files, 'compareOrder');
    //LATEST DATA FIRST
    $data_files = array_reverse($data_files);
    return ($data_files);
}

function compareTime($a, $b){
  return strtotime($a) - strtotime($b);
}

function get_runs(){
    if (curl_installed()){
        $request =
file_get_contents_curl($GLOBALS['lucid_api_root']."/api/get/data_files");
    }else{
        $request =
file_get_contents($GLOBALS['lucid_api_root']."/api/get/data_files");
    }
    $data_files = json_decode($request, true);
    $runs = [];
    foreach ($data_files as $datafile){
        if (!in_array($datafile['run'], $runs)){
            $runs[] = $datafile['run'];
        }
    }
    usort($runs, 'compareTime');
    $runs = array_reverse($runs);
    return $runs;
}

function get_frame($datafile, $number, $channel){
    $pixels_array = explode("\n",($datafile[$number]["channels"][$channel]));
    foreach($pixels_array as $pixels){
        $pixel_array = explode("\t",$pixels);
        $pixel_list[] = $pixel_array;
    }
    return ($pixel_list);
}

function get_datafile($id){
    if (curl_installed()){
        $request =
file_get_contents_curl($GLOBALS['lucid_api_root']."/api/get/frames?data_file=".$id);
    }else{
        $request =
file_get_contents($GLOBALS['lucid_api_root']."/api/get/frames?data_file=".$id);
    }
    $datafile = json_decode($request, true);
```

```
    return $datafile;
}


function get_datarun($runs, $timestamp){
    foreach($runs as $run){
        $interval[] = abs(strtotime($run) - $timestamp);
    }
    asort($interval);
    $closest = key($interval);
    return $runs[$closest];
}

function geocode($lat, $lng){

    $request =
file_get_contents("https://maps.googleapis.com/maps/api/geocode/json?key=".$GLOBALS[
'google_maps_key']."&latlng=".$lat.",".$lng);

    $data = json_decode($request);
    $data = array($data);

    $object = new StdClass();
    foreach ( $data as $key => $value ){
        $object -> $key = $value;
    }
    $arr = json_decode(json_encode($object), true);
    // print_r($arr);
    if ( @$arr[0]["results"][0]["formatted_address"] != ""){
        $data = $arr[0]["results"][0]["formatted_address"];
    }else{
        $data = "";
    }
    return ($data);

}

function geocode_address($address){
    $request =
file_get_contents("https://maps.googleapis.com/maps/api/geocode/json?key=".$GLOBALS[
'google_maps_key']."&address=".urlencode($address));
    $data = json_decode($request);
    $data = array($data);
    $object = new StdClass();
    foreach ( $data as $key => $value ){
        $object -> $key = $value;
    }
    $arr = json_decode(json_encode($object), true);
    // print_r($arr);
    if ( @$arr[0]["results"][0]["geometry"]["location"] != ""){
        $data = $arr[0]["results"][0]["geometry"]["location"];
    }else{
        $data = "";
    }
    return ($data);
}

function distance($lat1, $lng1, $lat2, $lng2){
    $pi80 = M_PI / 180;
    $lat1 *= $pi80;
    $lng1 *= $pi80;
    $lat2 *= $pi80;
    $lng2 *= $pi80;
    // mean radius of Earth in km
    $r = 6371.0088;
    $dlat = $lat2 - $lat1;
    $dlng = $lng2 - $lng1;
```

```
    $a = sin($dlat / 2) * sin($dlat / 2) + cos($lat1) * cos($lat2) * sin($dlng / 2)
* sin($dlng / 2);
    $c = 2 * atan2(sqrt($a), sqrt(1 - $a));
    $km = $r * $c;
    // return floor($km * 0.621371192);
    return $km;
}

function get_tle(){
    echo '<div style="display:none;">';
    $url = 'http://www.celestrak.com/cgi-bin/TLE.pl?CATNR=40076';
    if (curl_installed()){
        $doc = file_get_contents_curl($url);
    }else{
        $doc = file_get_contents($url);
    }
    // $array = json_decode($location, true);
    // SET TITLE OF PAGE
    $doc = str_replace("NORAD Catalog Number 40076","LUCID Tracker", $doc);
    echo $doc;
    $doc = explode($doc, '<PRE>');
    //in case get_tle fails -- uses old tle default automatically in js
    echo "<pre>TDS 1
    1 40076U 14037H   16241.07882549  .00000095  00000-0  19483-4 0  9992
    2 40076  98.3245 334.8261 0005052 303.4095  56.6614 14.81243408115632
    </pre>";
    echo '</div>';
}

function pyxan($id, $frame, $channel){
    if (curl_installed()){
        $request =
file_get_contents_curl($GLOBALS['pyxan_root']."/".$id."_".$frame."_".$channel.".txt"
);
    }else{
        $request =
file_get_contents($GLOBALS['pyxan_root']."/".$id."_".$frame."_".$channel.".txt");
    }
    // $request = json_decode($request, true);
    return $request;
}

?>
```

Filename: gen_image.php

```
<?php
include("init.php");

function get_frame($datafile, $number, $channel, $min_e, $max_e){
    $pixels_array = explode("\n",($datafile[$number]["channels"][$channel]));
    foreach($pixels_array as $pixels){
        $pixel_array = explode("\t",$pixels);
        if (count($pixel_array) == 3){
            if ($pixel_array[2] >= $min_e and $pixel_array[2] <= $max_e){
                $pixel_list[] = $pixel_array;
            }
        }
    }
    if (isset($pixel_list)){
        return ($pixel_list);
    }else{
        return [];
    }
}

function get_datafile($id){
    if (curl_installed()){
        $request =
file_get_contents_curl($GLOBALS['lucid_api_root']."/api/get/frames?data_file=".$id);
    }else{
        $request =
file_get_contents($GLOBALS['lucid_api_root']."/api/get/frames?data_file=".$id);
    }
    $data = json_decode($request);
    $data = array($data);
    $object = new StdClass();
    foreach ( $data as $key => $value ){
        $object -> $key = $value;
    }
    $arr = json_decode(json_encode($object), true);
    $datafile = $arr[0];
    return $datafile;
}

function getProperColour($image, $energy_val){
    if ($energy_val > 0 && $energy_val < 100){
        $colour = "green";
    }elseif ($energy_val >= 100 && $energy_val < 200){
        $colour = "yellow";
    }elseif ($energy_val >= 200 && $energy_val < 300){
        $colour = "orange";
    }elseif ($energy_val >= 300){
        $colour = "red";
    }
    return $colour;
}

function gen_image($pixels, $scale){
    header('Content-type:image/png');
    $width = 256;
    $height = 256;
    $image = imagecreate( $width, $height );

    $colour_opt = isset($_GET["simple"]);
    if ($colour_opt == False){
        // $black_background = imagecolorallocate ( $image, 0, 0, 0 );
        // $grey_background = imagecolorallocate ( $image, 100, 100, 100 );
        $dark_blue_background = imagecolorallocate ( $image, 0, 0, 40 );
    }else{
        $blue_background = imagecolorallocate ( $image, 31, 119, 208 );
```

```
        // $dark_blue_background = imagecolorallocate ( $image, 0, 0, 40 );
    }

    $green = imagecolorallocate ($image, 0, 255, 0);
    $yellow = imagecolorallocate ($image, 255, 170, 0);
    $orange = imagecolorallocate ($image, 255, 85, 0);
    $red = imagecolorallocate ($image, 255, 0, 0);
    $white = imagecolorallocate ($image, 255, 255, 255);

    foreach ($pixels as $pixel){
        $x = $pixel[0];
        $y = $pixel[1];
        if ($colour_opt == False){
            if ( getProperColour($image, $pixel[2]) == "red" ){
                $colour = $red;
            }elseif( getProperColour($image, $pixel[2]) == "orange" ){
                $colour = $orange;
            }elseif( getProperColour($image, $pixel[2]) == "yellow" ){
                $colour = $yellow;
            }elseif( getProperColour($image, $pixel[2]) == "green" ){
                $colour = $green;
            }
        }else{
            $colour = $white;
        }
        imagesetpixel ( $image, $x, $y, $colour );
    }

    if ($scale > 1 and $scale < 21){
        $image = imagescale ($image, $scale*256, $scale*256, IMG_NEAREST_NEIGHBOUR);
    }
    imagepng($image);
}

function get_datafile_image($id, $frame, $channel, $scale, $min_e, $max_e){
    $datafile = get_datafile($id);
    $pixels = get_frame($datafile, $frame, $channel, $min_e, $max_e);
    // print_r($pixels);
    gen_image($pixels, $scale);
}

$id = isset($get_id) ? $get_id : $_GET["id"];
$frame = isset($_GET["frame"]) ? $_GET["frame"] : 0;
$channel = isset($_GET["channel"]) ? $_GET["channel"] : 0;
$scale = isset($_GET["scale"]) ? $_GET["scale"] : 1;
$min_e = isset($_GET["min_e"]) ? $_GET["min_e"] : 0;
$max_e = isset($_GET["max_e"]) ? $_GET["max_e"] : 11810;

get_datafile_image($id, $frame, $channel, $scale, $min_e, $max_e);

?>
```

Filename: gen_xyc.php

```
<?php
include("init.php");

function gen_xyc($datafile, $number, $channel, $min_e, $max_e){
    if ($min_e == 0 and $max_e == 11810){
        header("Content-Type: text/plain");
        print_r ($datafile[$number]["channels"][$channel]);
    }else{
        $pixels_array = explode("\n",($datafile[$number]["channels"][$channel]));
        foreach($pixels_array as $pixels){
            $pixel_array = explode("\t",$pixels);
            if (count($pixel_array) == 3){
                if ($pixel_array[2] >= $min_e and $pixel_array[2] <= $max_e){
                    $pixel_list[] = $pixel_array;
                }
            }
        }
        if (!isset($pixel_list)){
            $pixel_list = [];
        }
        foreach($pixel_list as $pixel){
            $new_pix_array[] = implode("\t", $pixel);
        }
        $new_xyc = implode("\n", $new_pix_array);
        header("Content-Type: text/plain");
        print_r($new_xyc);
    }
}


function get_datafile($id){
    if (curl_installed()){
        $request =
file_get_contents_curl($GLOBALS['lucid_api_root']."/api/get/frames?data_file=".$id);
    }else{
        $request =
file_get_contents($GLOBALS['lucid_api_root']."/api/get/frames?data_file=".$id);
    }
    $data = json_decode($request);

    $data = array($data);
    $object = new StdClass();
    foreach ( $data as $key => $value ){
        $object -> $key = $value;
    }
    $arr = json_decode(json_encode($object), true);
    $datafile = $arr[0];
    return $datafile;
}

function get_datafile_xyc($id, $frame, $channel, $min_e, $max_e){
    $datafile = get_datafile($id);
    gen_xyc($datafile, $frame, $channel, $min_e, $max_e);
}

$id = isset($get_id) ? $get_id : $_GET["id"];
$frame = isset($_GET["frame"]) ? $_GET["frame"] : 0;
$channel = isset($_GET["channel"]) ? $_GET["channel"] : 0;
$min_e = isset($_GET["min_e"]) ? $_GET["min_e"] : 0;
$max_e = isset($_GET["max_e"]) ? $_GET["max_e"] : 11810;
get_datafile_xyc($id, $frame, $channel, $min_e, $max_e);
?>
```

Filename: footer.php

```
<?php
?>
<footer class="footer">
    <div class="container-fluid">
        <a href="http://www.researchinschools.org">
        <img class="pull-left" src="assets/img/iris.png" height="70px"></img>
        </a>
        <p class="copyright pull-right">
            <a href="http://www.amshenoy.com">Abhishek Shenoy</a> - 2017
        </p>
    </div>
</footer>
<?php
?>
```

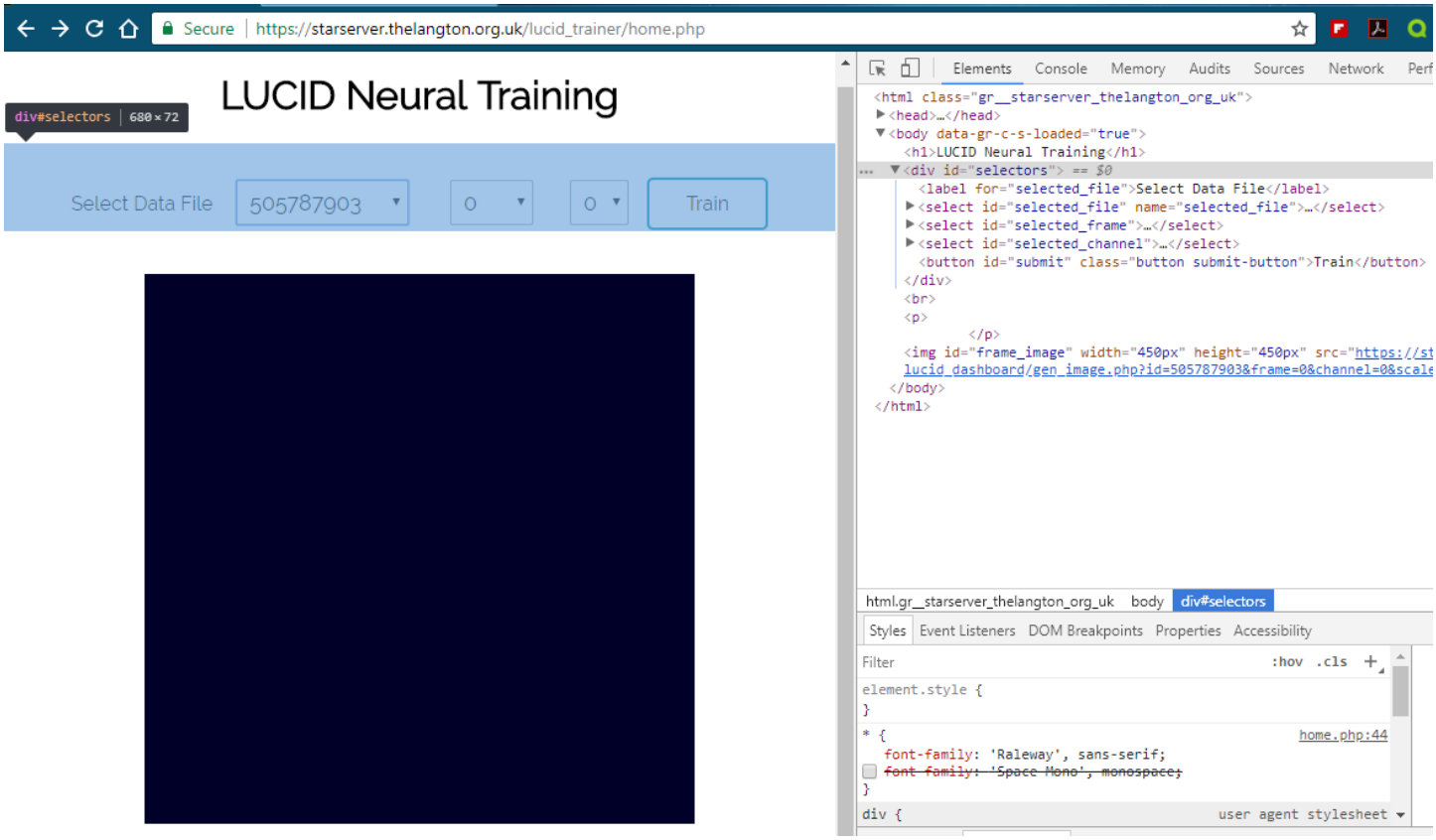
Testing

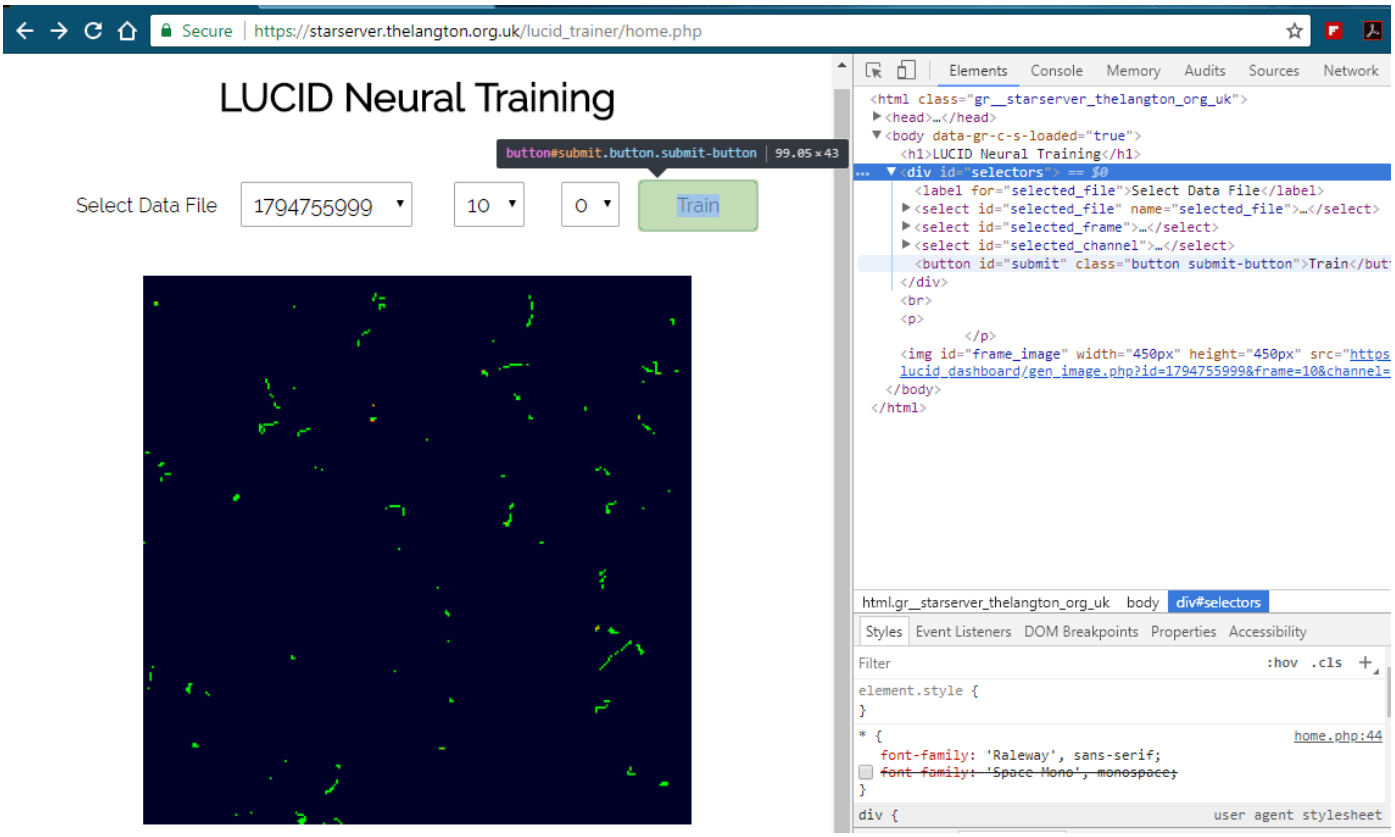
LUCID Trainer Tests

Test No.	Element / Form	Purpose	Test Data/Action	Expected Result	Result
1	Homepage – Dropdown Selectors	Allow users to select which file to train on home page and load image of the first frame of the selected file.	Data File, Frame and Channel are selected	Image of selected frame appears in image box	Image successfully appears in image box.
2	Homepage – Train Button	Check user is redirected to the correct page based on the selected dropdown.	Train button is clicked	User is redirected to training page based on selected inputs.	Page redirects to training page for the selected parameters.
3	Train Page – Radio Buttons Test	Radio buttons change upon selection. Only one button can be selected.	Radio button is selected	Radio buttons switches to a single radio selection when user clicks on an unselected radio button.	Test successful as radio buttons work as expected.
4	Train Page – AJAX Submission	AJAX request is performed successfully upon submit button click.	Submit button is clicked.	AJAX request successfully stores particle classification in database after successful request.	AJAX request is sent by the application and received by the server without any errors and the returns a successful response.
5	Train Page – Redirection after submission	Check user is correctly redirected to appropriate page after submission	Submit button is clicked	Correct redirection of the page which is either a new particle on training page or the home page.	User is redirected to home page after submission if frame is empty, else it continues to the next particle in the frame.
6	Train Page – Request Parameter Testing	Request parameters are only accepted and page is generated, if parameters are valid.	New page is loaded with custom parameters	Page generation is fully automatic based on provided GET parameters	Correct page is generated for a given id, frame, channel and index

Selectors and Train Button Test

Test 1 and 2

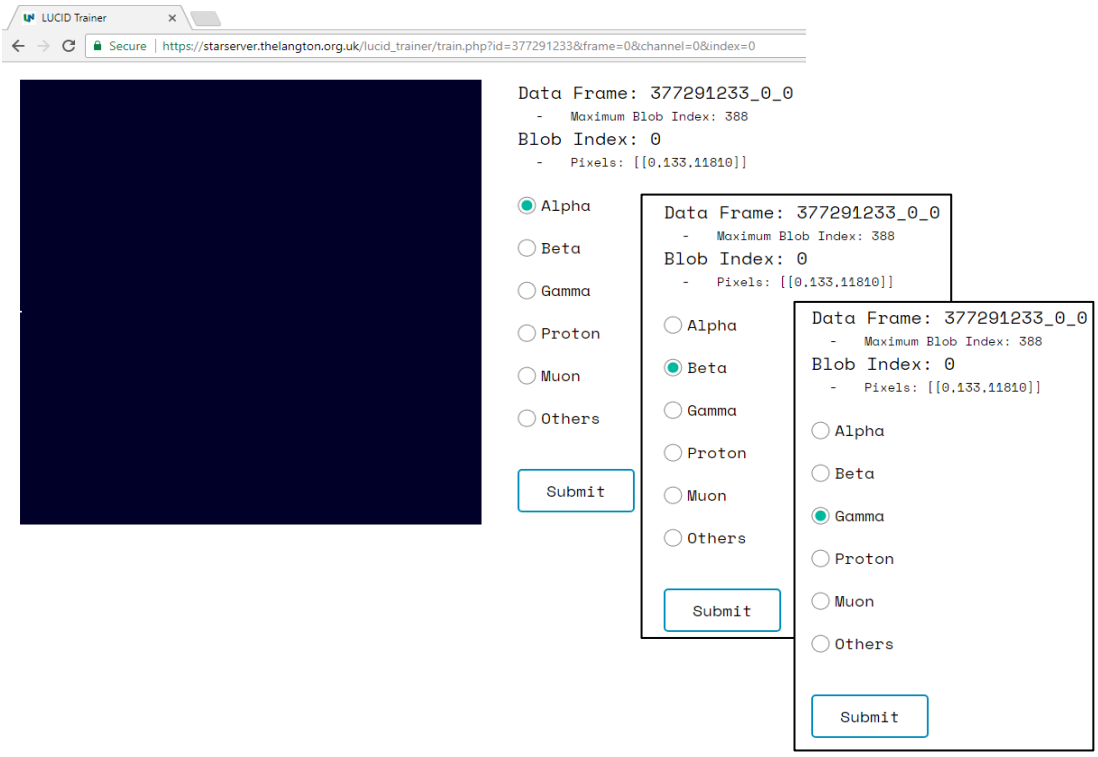




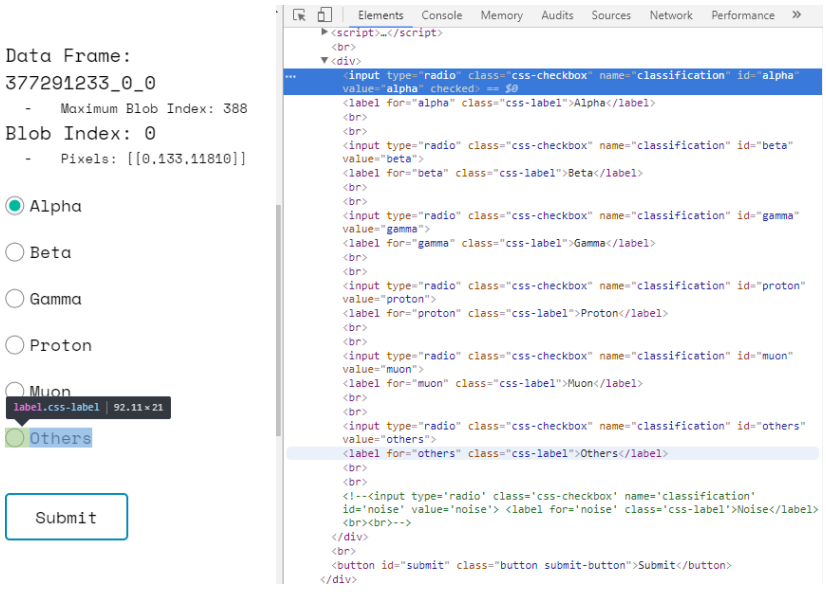
The dropdown selectors worked as expected. When any of the selections were changed the image was updated aaccordingly. The train button click event listener worked too and this was all tested using Google Chrome breakpoints in the script part of the document in the Chrome console “Sources” tab.

Radio Button Tests

Test 3



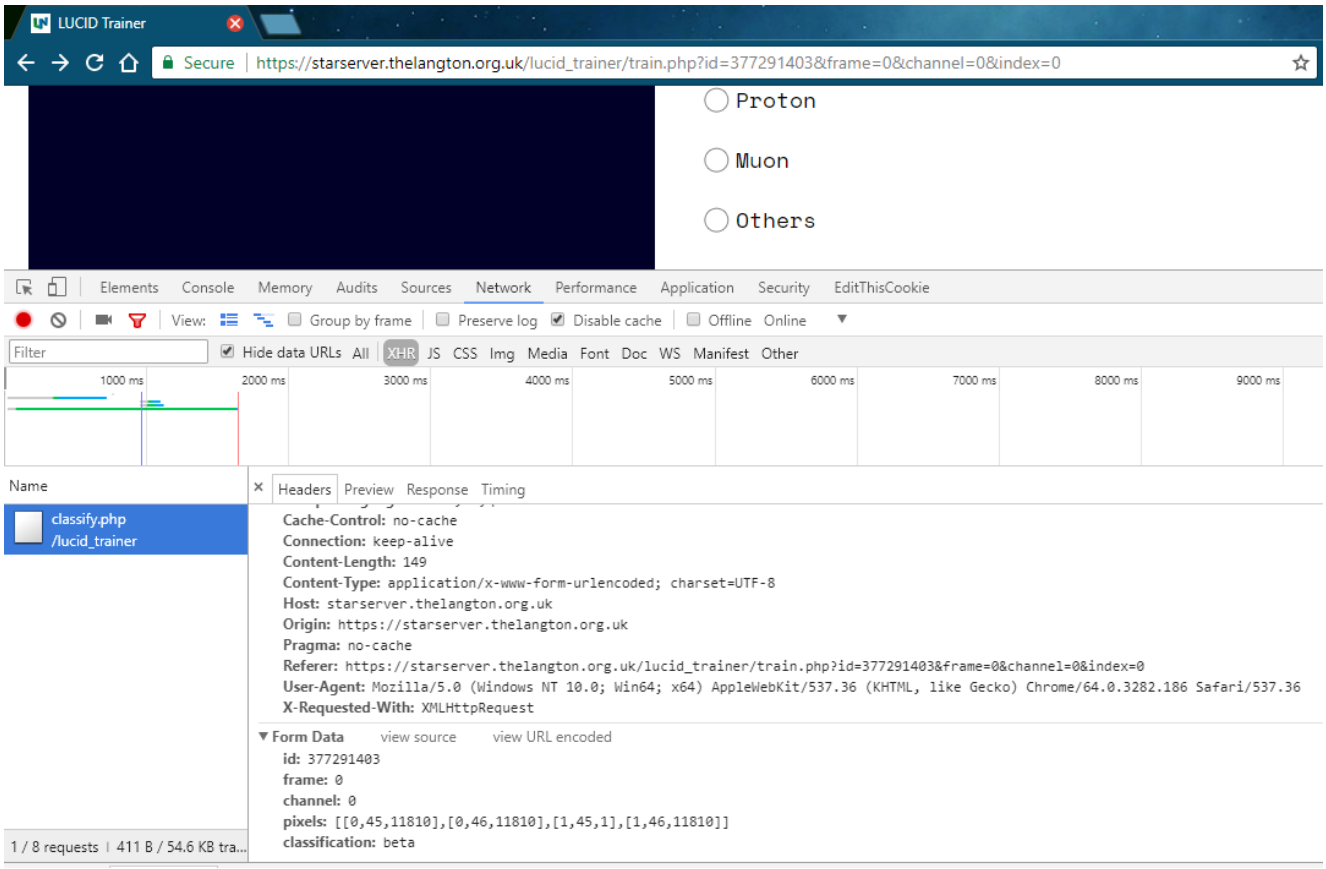
The screenshots show the evidence for the working radio buttons in the questionnaire. As the radio buttons are selected, the data sent to the server using AJAX also changes.

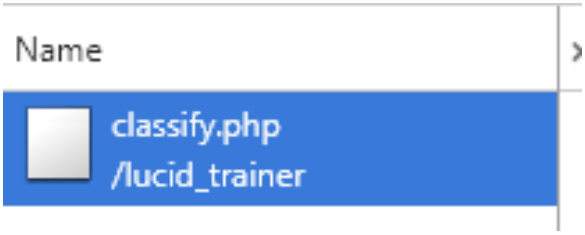


The radio buttons and the corresponding JavaScript event listeners were tested as an individual file as well as using breakpoints when testing the final code. Any errors were fixed and the training page is now fully functional.

AJAX Submission Test

Test 4





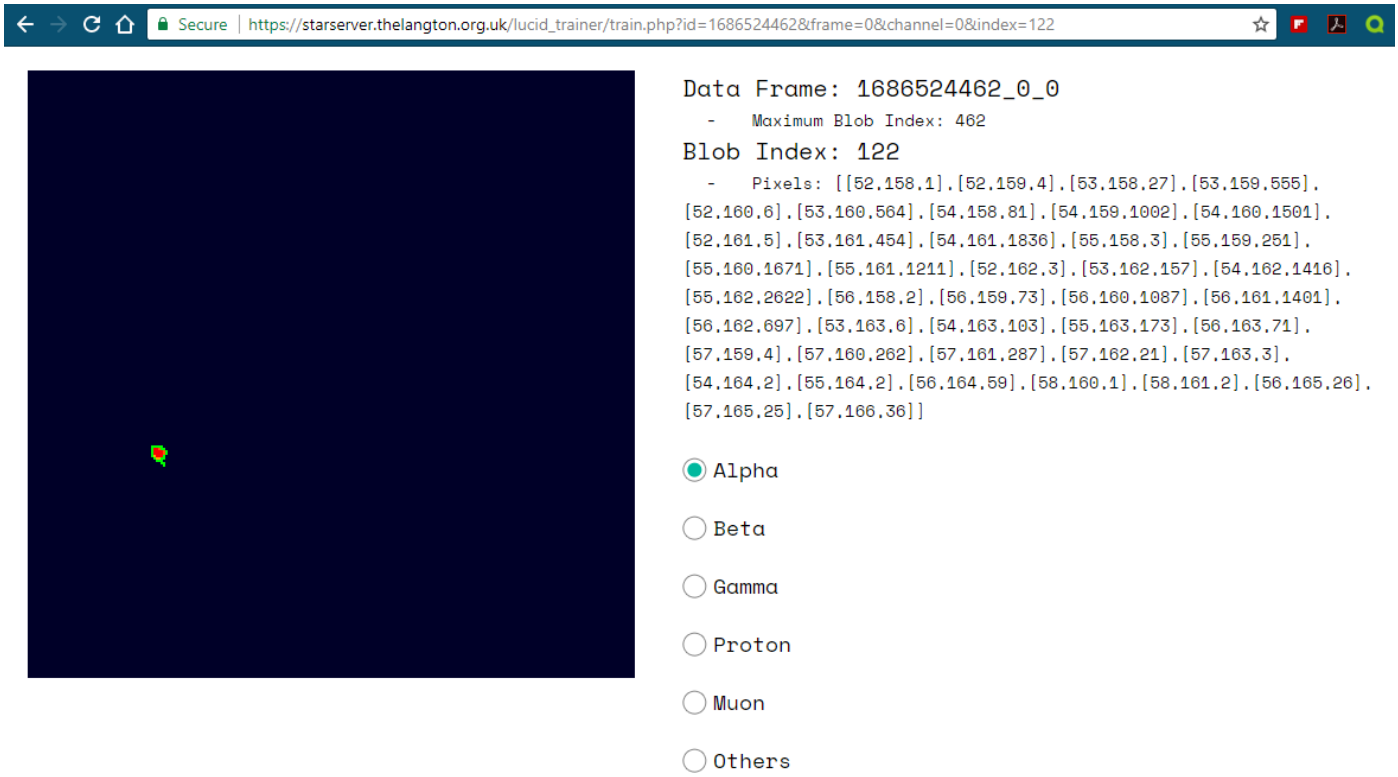
The AJAX request is made to the classify.php file which inserts the form data into the training database, returns {“status”: “valid”} if the request has succeeded or returns {“status”: “invalid”} if it has failed. The user is then redirected to the next particle training page.

The form submission was again tested using Google Chrome breakpoints in the script part of the document in the Chrome console “Sources” tab.

Redirection and Parameter Test

Test 5 and 6

Redirection is done using client-side JavaScript but the redirection link is created using server-side PHP. These both were checked by running the individual parts of code by themselves.



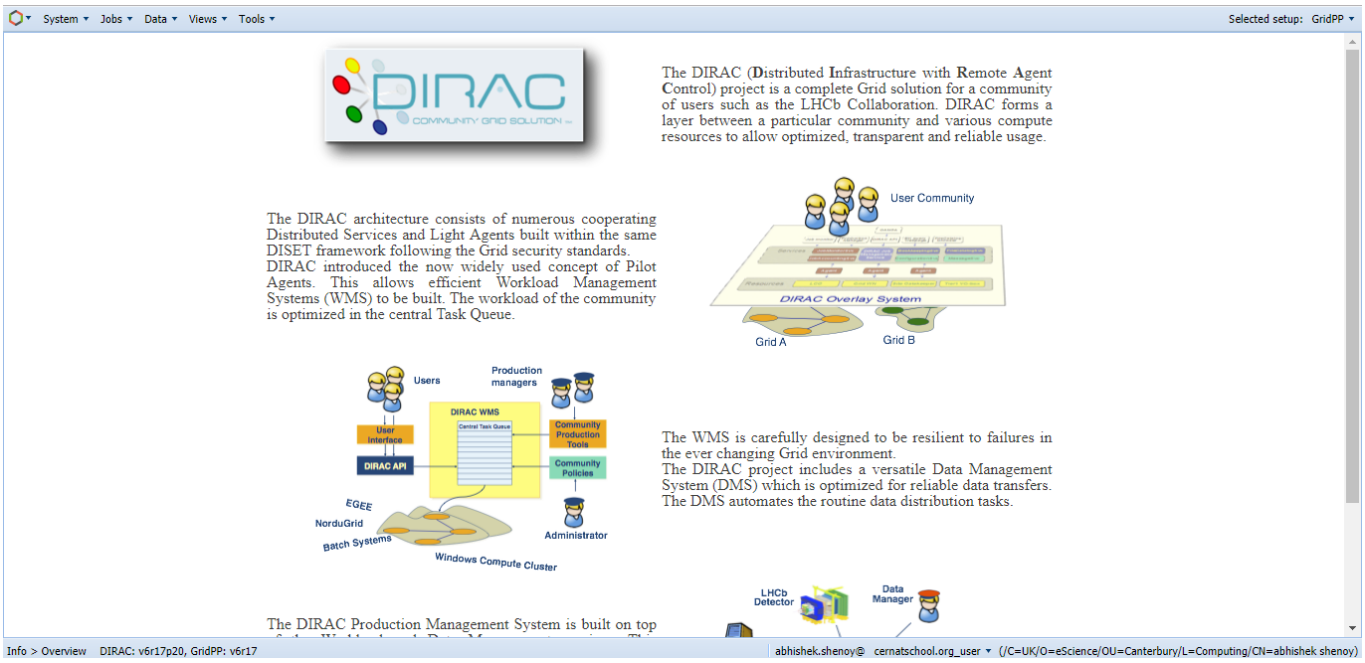
The parameters are validated so that only integers are accepted. Integers with decimal points are also to be rejected as none of the parameters should contain any other characters apart from the numeric digits. Strings in the parameters are invalid as they are not valid data file IDs, frame numbers or particle indices.

If the parameters are invalid, then the page redirects to the hompage. If the frame is empty then the user is also redirected to the homepage but this time with a “msg” GET parameter which is used to output an empty error.

GridPP

“The GridPP Collaboration is a community of particle physicists and computer scientists based in the United Kingdom and at CERN, supporting tens of thousands of CPU cores and petabytes of data storage across the UK and it was used to discover the Higgs boson at CERN's Large Hadron Collider. Such a large-scale distributed computing grid is used to solve data-intensive problems.” [26]

Doing research as a part of the LUCID project, I was able to get an e-science certificate to be able to access the grid and as of 2016 and 2017, I am the youngest member of the Particle Physics Computing Grid. Being a member of the GridPP has enabled me to train and test several of these neural networks concurrently as individual jobs on the computing grid.



Neural Network Tests

Test No.	Purpose	Test Data/Action	Expected Result	Result
1	Neural network can successfully access the training database	Run training database access code.	All particle clusters in the database are printed out correctly.	Particle clusters are printed out as expected.
2	Particle clusters are correctly converted into corresponding metrics.	Test metric conversion code from lucid_utils library as well as performing checks for any internal errors.	User is redirected to training page based on selected inputs.	Page redirects to training page for the selected parameters.
3	Neural network is able to be trained on the training dataset. This was a crucial test because if there was not enough data or the architecture was not good enough then there would be no point in running the neural network on the whole dataset or for a large number of epochs.	Demo train the neural network by running the code. Debugged by printing out variables every few lines and checking outputs.	The training accuracy and test accuracy is printed out every epoch.	The output is as expected.
4	Neural network is successfully able to learn after training. The training accuracy should be high but the neural network should not overfit. Therefore both training and test accuracies must be measured during training.	Train the neural network for a large amount of epochs and run it on the test data to check the training accuracy as well as the test accuracy.	Little overfitting, high training accuracy and high test accuracy.	Overall the result is as expected but this part was judged after having created and tested all the other algorithms.
5	The neural network should overall outperform all the other classifiers that have been produced and tested.	Train and test all the algorithms.	The metric-based neural network should outperform the other algorithms.	After having tested all of the algorithms, the network was not necessarily the most accurate but it was most able to classify uncommon particles.
6	The metric-based neural network should have a high classification accuracy. Apart from not overfitting, the neural network should be better than the current algorithm at classifying more uncommon particles and it must also have an overall higher training and test accuracy.	Generate confusion matrices for showing where the neural network succeeds and where it may falter.	Classification should more accurate than the current algorithm.	The confusion matrices showed that the metric-based network was much better than the current analytic algorithm.

Database Access Test

```
[6, 0.5500394833255956, 1.8633899812498158, 9.6715386336415374, 0.78136350032563995, 0.76273235616700241, 1.6099689437998563, 0.0]
[1, 1, 0.0, 0, 0, 0, 0, 0.0]
[11, 0.41495637464312018, 2.9048264197582641, 10.086199985245901, 2.8077629228700594, 2.6978688692542847, 1.8934005703713281, 0.0]
[4, 1.0185916357881299, 1.1180339887498949, 0.80901699437494745, 0.38196601125263624, 0.38196601125010521, 1.7888543819998317, 0.0]
[16, 0.58995352661031963, 2.9381648183857898, 1.8119641594959046, 13.964171630801125, 8.1420411172896898, 2.7227880307937089, 0.0]
[11, 0.35159374150259332, 3.1557372650381437, 2.446008426960621, 5.9291826137702941, 4.5269927251434572, 1.7428573857949232, 0.0]
[2, 2.5464790894703255, 0.5, 0.5, 1.3372316027246018e-10, 0.0, 2.0, 0.0]
[13, 0.38851489996988264, 3.2635697593225252, 15.656919181286561, 3.1702772562265564, 3.0656752684567166, 1.99168410034211, 0.0]
[8, 0.23551251694523243, 3.2882366094914763, 4.17958892293486, 1.9661405580278508, 0.64011032944392932, 1.2164574740315288, 0.0]
[1, 1, 0.0, 0, 0, 0, 0, 0.0]
[2, 2.5464790894703255, 0.5, 0.5, 4.3858967783808329e-10, 0.0, 2.0, 0.0]
[2, 2.5464790894703255, 0.5, 0.5, 0.0, 0.0, 2.0, 0.0]
[3, 0.66110514822787314, 1.2018504251546629, 1.5811388300841898, 0.13148290817919267, 0.0, 1.2480754415067656, 0.0]
[3, 1.7188733853924452, 0.74535599249993523, 0.70710678118654757, 0.33333333355182804, 0.0, 2.0124611797497964, 0.0]
[6, 0.55003948332558794, 1.8633899812498287, 4.5253734563751236, 1.0271465303366176, 0.99788815829025401, 1.6099689437998452, 0.0]
[18, 0.082855757921172279, 8.3157221314748924, 65.778697203795744, 4.4446299130709619, 4.1287647933846436, 1.0822872454979136, 0.0]
[27, 0.26632490923509244, 5.680689500065915, 3.2749687394967149, 62.637790517745366, 36.1061106273267, 2.3764720813984561, 0.0]
[10, 0.30489452699596792, 3.2310988842807036, 75.763232415381168, 1.5952835436226565, 1.5911458019264515, 1.5474611514754317, 0.0]
[5, 0.44706444688734587, 1.8867962264113221, 1.9256907122731246, 1.0387503062687351, 0.54695582496502337, 1.3249973500079949, 0.0]
[7, 0.96619726514195259, 1.5185922589620897, 1.1456566104074644, 3.3406038564347469, 0.9076475362888301, 2.3047661275398181, 0.0]
[1, 1, 0.0, 0, 0, 0, 0, 0.0]
[5, 0.99471839432434761, 1.2649110640673507, 0.84894854079334492, 1.0000000001849856, 0.40286356089734809, 1.9764235376052388, 0.0]
[1, 1, 0.0, 0, 0, 0, 0, 0.0]
[12, 0.31216769768762148, 3.4980153103025637, 11.833938837676769, 2.9191869817205105, 2.6842211335304214, 1.715258358740867, 0.0]
[7, 0.59016373492454399, 1.9430672155336308, 2.3095112875282267, 1.4369820719141637, 1.2000799601830219, 1.8012758241298326, 0.0]
[18, 0.13931581660216627, 6.412997989230667, 53.278827784061654, 3.3720511781816103, 3.1868675941039006, 1.4033997851104396, 0.0]
[4, 2.5464790894703246, 0.70710678118654757, 0.70710678118654757, 0.99999999999951061, 0.0, 2.8284271247461898, 0.0]
[2, 2.5464790894703255, 0.5, 0.5, 0.0, 0.0, 2.0, 0.0]
[3, 1.71887338539246, 0.74535599249993201, 0.70710678118654757, 0.33333334136976617, 0.0, 2.0124611797498049, 0.0]
[4, 1.0185916357881299, 1.1180339887498949, 0.80901699437494745, 0.38196601134767866, 0.38196601125010521, 1.7888543819998317, 0.0]
[1, 1, 0.0, 0, 0, 0, 0, 0.0]
[25, 0.18436416101203745, 6.5698706227748467, 3.5918940620800748, 72.712104499835249, 63.649341095109165, 1.9026249857444693, 0.0]
[2, 2.5464790894703255, 0.5, 0.5, 0.0, 0.0, 2.0, 0.0]
[4, 0.78353202752933093, 1.2747548783981961, 1.228921966844378, 0.50000000001390899, 0.14520820347780106, 1.5689290811054724, 0.0]
[9, 0.35373156559143815, 2.8458329944145992, 14.36251574583293, 1.1111111111123222, 1.0271970456214072, 1.5812593391221366, 0.0]
[12, 0.13872370323470132, 5.2473538304770528, 97.340764286670733, 1.9349007068954065, 1.9195657627429823, 1.1434334702477118, 0.0]
[17, 0.35214061491127302, 3.9200487250696794, 14.885824222209783, 5.786044376352617, 5.3985353449146301, 2.1683403947610143, 0.0]
[2, 2.5464790894703255, 0.5, 0.5, 1.5163265470804051e-10, 0.0, 2.0, 0.0]
[20, 0.1706755421896998, 6.107372593840994, 70.1499840358413, 4.5677982070983028, 4.4694822103518401, 1.6373653066597809, 0.0]
[8, 0.48360433746617459, 2.2946949688357274, 7.8397806262463563, 1.4595772253476453, 1.365935066346029, 1.7431510742490985, 0.0]
```

Test 1 and 2

Each line in the above screenshot is a particle for which the list of metrics has been calculated. This shows that the metrics library works as expected. The **lucid_utils** Python library for generating the metrics had already been thoroughly tested before being used. As the cluster metrics have been correctly been printed out, this means that the program is correctly loading the particle clusters into memory from the database.

This entire test is the first section of the neural network training code where the database is loaded into runtime variables and parsed by calculating metrics for each particle in the database.

Demo Training Test

Test 3

```
No checkpoint found. Starting over.
0 0.193127962085 0.239361702128
1 0.232819905213 0.244680851064
2 0.250592417062 0.265957446809
3 0.313388625592 0.340425531915
4 0.383886255924 0.43085106383
5 0.59182464455 0.622340425532
6 0.612559241706 0.643617021277
7 0.638033175355 0.675531914894
8 0.681279620853 0.718085106383
9 0.710308056872 0.739361702128
10 0.776066350711 0.81914893617
11 0.781990521327 0.81914893617
12 0.785545023697 0.829787234043
13 0.79028436019 0.829787234043
14 0.796208530806 0.829787234043
15 0.799763033175 0.829787234043
16 0.802132701422 0.829787234043
17 0.804502369668 0.829787234043
18 0.806872037915 0.829787234043
19 0.806872037915 0.829787234043
20 0.806872037915 0.829787234043
21 0.808056872038 0.829787234043
22 0.809834123223 0.829787234043
23 0.809834123223 0.829787234043
24 0.810426540284 0.829787234043
25 0.812203791469 0.835106382979
26 0.812796208531 0.840425531915
27 0.813388625592 0.840425531915
28 0.813388625592 0.840425531915
29 0.813388625592 0.840425531915
30 0.813388625592 0.840425531915
31 0.813981042654 0.840425531915
32 0.813981042654 0.840425531915
33 0.814573459716 0.840425531915
34 0.815165876777 0.840425531915
35 0.815758293839 0.840425531915
36 0.815758293839 0.840425531915
37 0.815165876777 0.840425531915
38 0.815165876777 0.840425531915
39 0.815165876777 0.840425531915
```

This screenshot shows the first few epochs when the neural network was trained for the first time. It is possible to see that the neural network reaches a high accuracy pretty quickly with not only the training data but also the test data. This could be assumed to be due to a lack of training data however, I had created a relatively large dataset of approximately 1800 particles. Although this is not an extremely large dataset, it was surprisingly enough. The main thing to consider was the probability distribution of the different particle classes and due to the unavailability of certain particle types, it was very difficult to even the amount of each particle type. This may have been a contributing factor to the high accuracy however to ensure that it could classify uncommon particles with a relatively good accuracy, I had tested the neural network separately on unique particle tracks and the outputs for the test dataset was really close to the results shown here.

```
97 0.826421800948 0.851063829787
98 0.826421800948 0.851063829787
99 0.827606635071 0.851063829787
0
20
40
60
80
100
120
140
160
180
[0 1 0 1 1 1 1 0 0 1 1 1 1 0 0 1 0 1 1 1 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 1 1
 0 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 5 1 1 1 1 0 0 1 0 0 1 0 1 1 1 0
 0 0 1 0 0 1 1 0 0 1 0 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 1 1 0 1 1 1 1 0 5 1 0
 0 5 1 1 1 1 1 1 1 1 1 3 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 1 1 0
 1 1 0 1 1 1 0 0 1 0 1 4 0 0 1 0 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1
 1 1 1]
Accuracy: 0.851064
```

Training Graphs

Test 4

To check the accuracy of the current algorithm, I ran the current analysis algorithm on all of the blobs in the database created by the LUCID Neural Trainer. The total number of correct classifications was divided by the total number of particles to calculate the accuracy of the algorithm. This was very important, as it would allow me to evaluate the success of my neural network and the improvements necessary to further develop the classifier.

During the training process of the deep neural network, the training accuracy and the loss of the error function was recorded. After the whole training process, this data was converted to a CSV file to allow easy data visualisation. The graphs below were created using the Tableau software from the CSV files generated by Tensorboard.

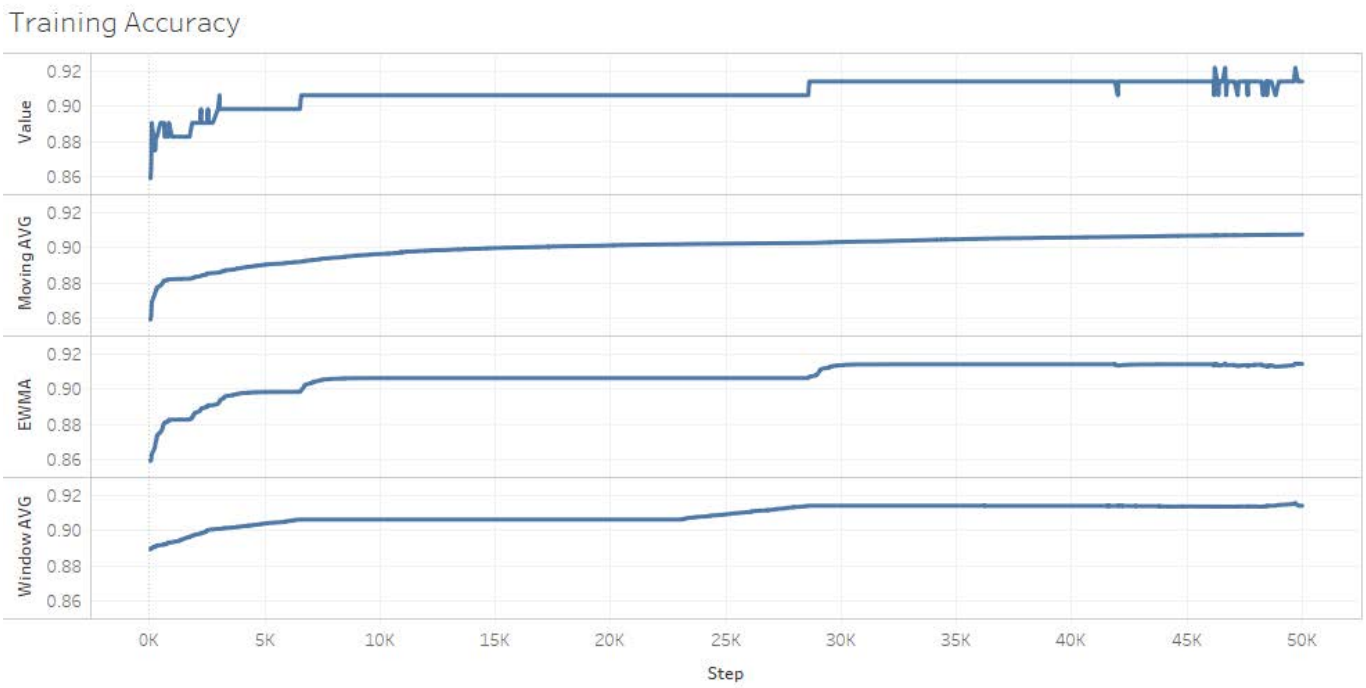


Figure 19 - Training Accuracy Graph

Figure 19 shows the increase in accuracy of the neural model as the model was trained over 50,000 epochs. The moving average, exponentially weighted moving average and window average show the smoothened version of the accuracy graph representing the approximate average training accuracy rather than the absolute value for a specific training instance.

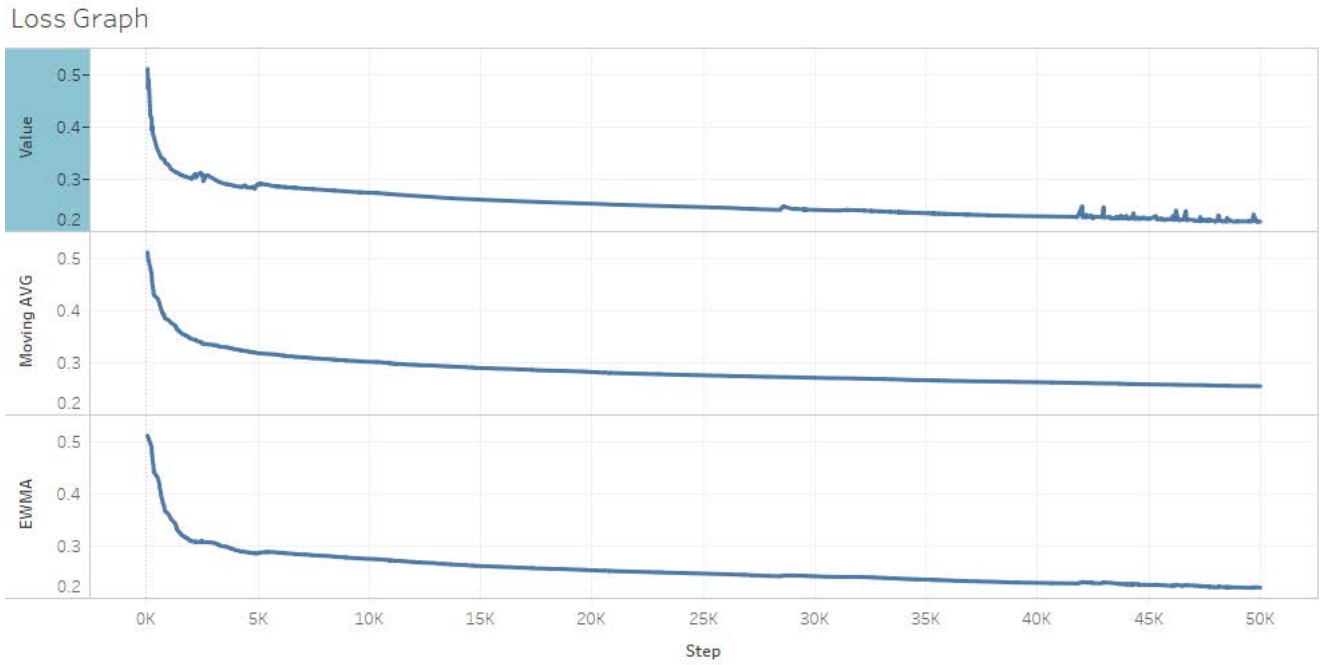


Figure 20 - Training Loss Graph

Figure 20 shows the error from the error function as the model was trained over 50,000 epochs. The moving average and the exponentially weighted moving average show the smoothened version of the loss graph representing the approximate average training loss rather than the absolute value for a specific training instance.

To compare the accuracy of the neural network with the accuracy of the benchmark classifiers, I simply ran the entire database through each of the classifiers and printed out the accuracy tests. Figure 21 shows a summary of the results in the form of a bar graph.

Classifiers Accuracy Graph

Test 5

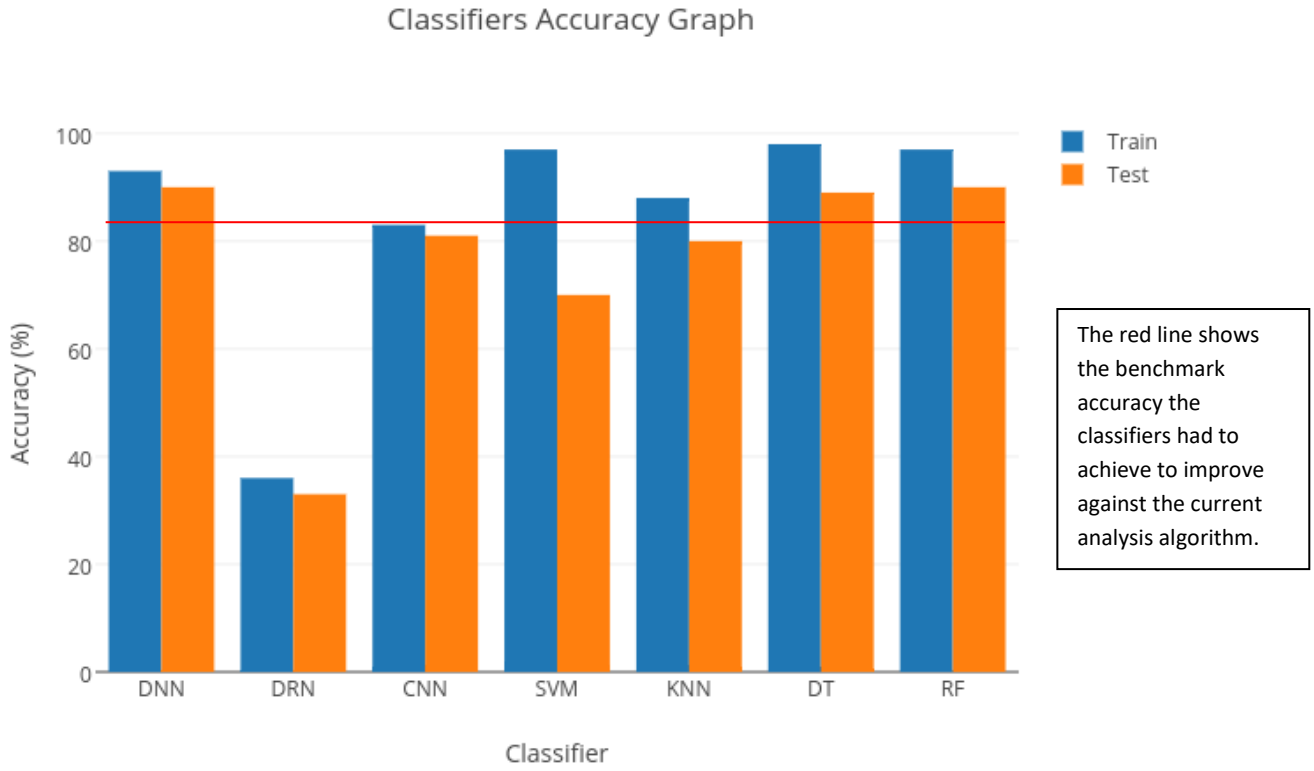
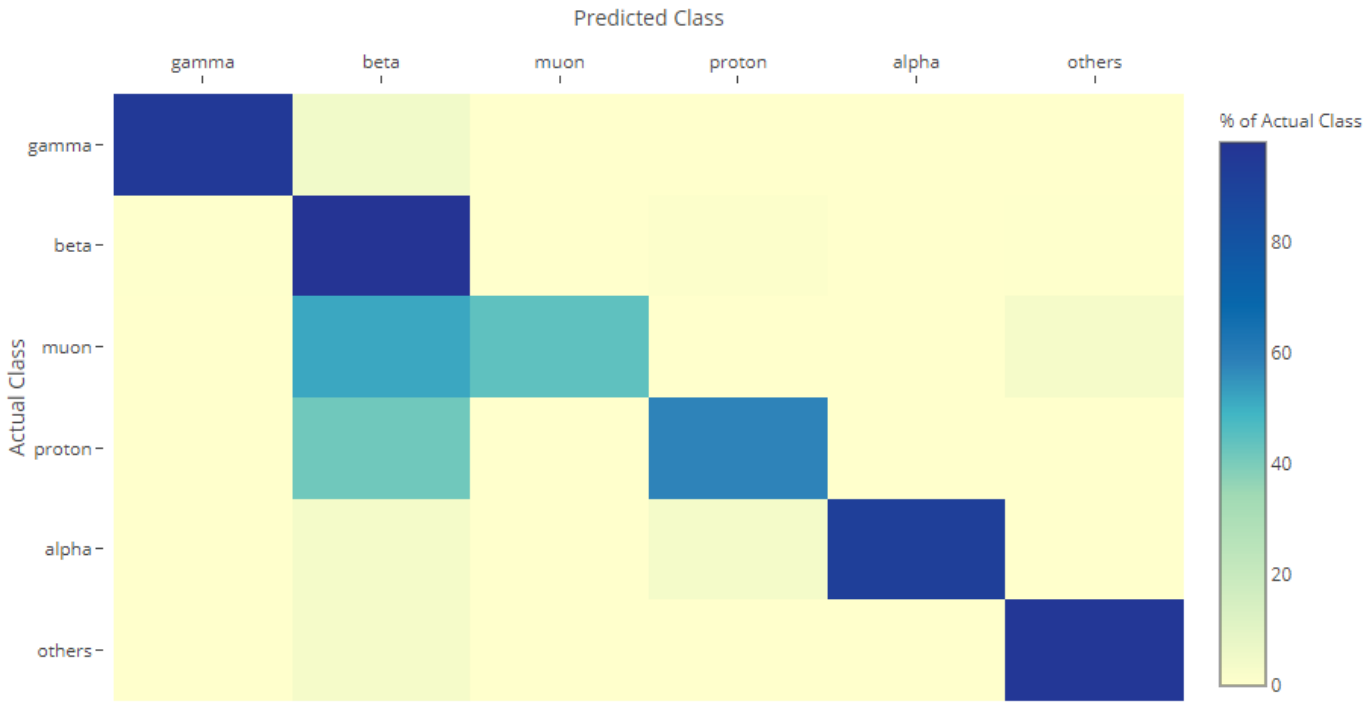


Figure 21 - Classifiers Accuracy Graph

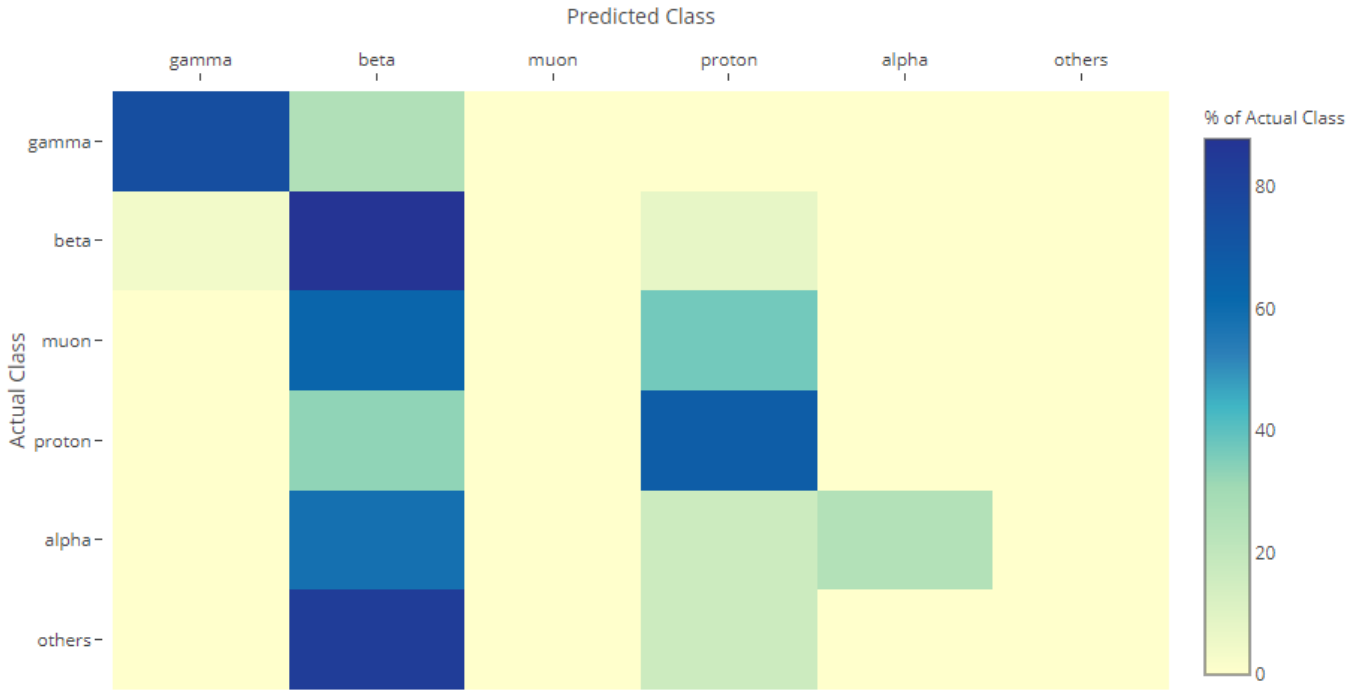
Confusion Matrices

Test 6

Neural Analysis



Old Algorithm



The above confusion matrices clearly show that the new algorithm is a start improvement over the old algorithm. The matrices data has been normalised by converting it to a percentage of the particle type rather than the number of particles given a particular classification. This was because the amount of data for each particle was different and therefore the default matrices do not provide a clear insight into the classified data.

Mini Investigation – South Atlantic Anomaly

To test the program for a real life phenomenon, I devised an experiment using LUCID to analyse the type of radiation in space from the data files that have already been collected as well as new ones that have been configured with specific configuration files for specific regions over the Earth.

Hypothesis

It is possible to prove the existence of the South Atlantic Anomaly using the neural network on LUCID data. On greater analysis of the data collected, it is possible to understand more details about the correlation between the location and the type and intensity of radiation over the Earth.

Experiment

There were two methods for observing the South Atlantic Anomaly:

1) Collect new data files with new configuration files to map the particle counts
2) Use existing data files to map the particle counts

New Data Files

For the first method, to test for the South Atlantic Anomaly, the shutter exposure time had to be quite small to prevent the chip from receiving too many particle hits and whiting out the chip. The other settings of the chip were kept constant as they were not necessary to change.

Existing Data Files

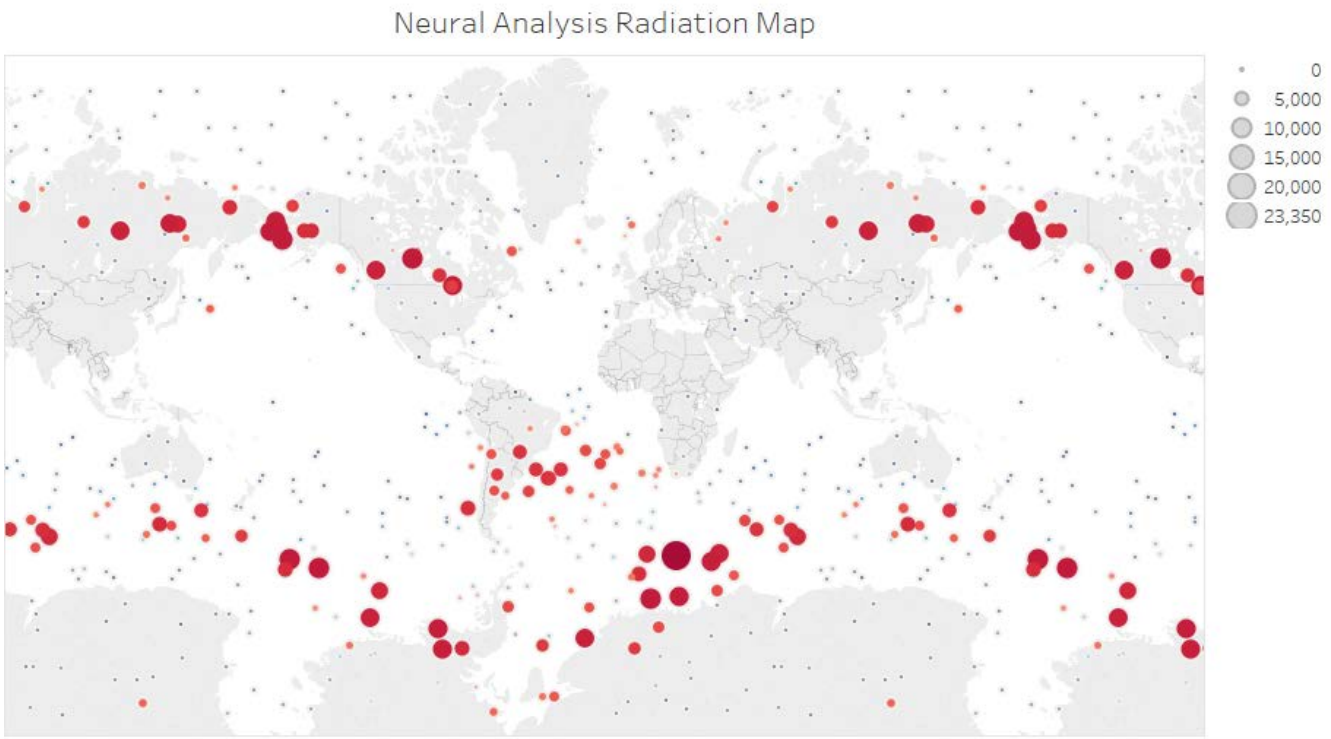
The second method was to analyse the existing data files using the data API in Python to get the selected data files. The latitude, longitude, number of frames and the counts of each particle in each data file will be written to a text file. By collecting the particle counts for all of the selected data files, a radiation map can be plotted which can then be used to observe the high radiation zones and the average levels of radiation over the Earth.

For plotting these maps, I decided to use a mixture of programming and Tableau. Tableau provided me with flexibility in data handling while the programmatic map generator allowed me to make continual revisions to the display of the standard radiation map.

Results

Due to the vast amount of data collected from LUCID, it was only possible to analyse approximately 10 frames from 10% of all the data files to prove this hypothesis. Analysing these frames using the neural network was much slower than using the old analysis algorithm, as the neural model was computationally much more expensive.

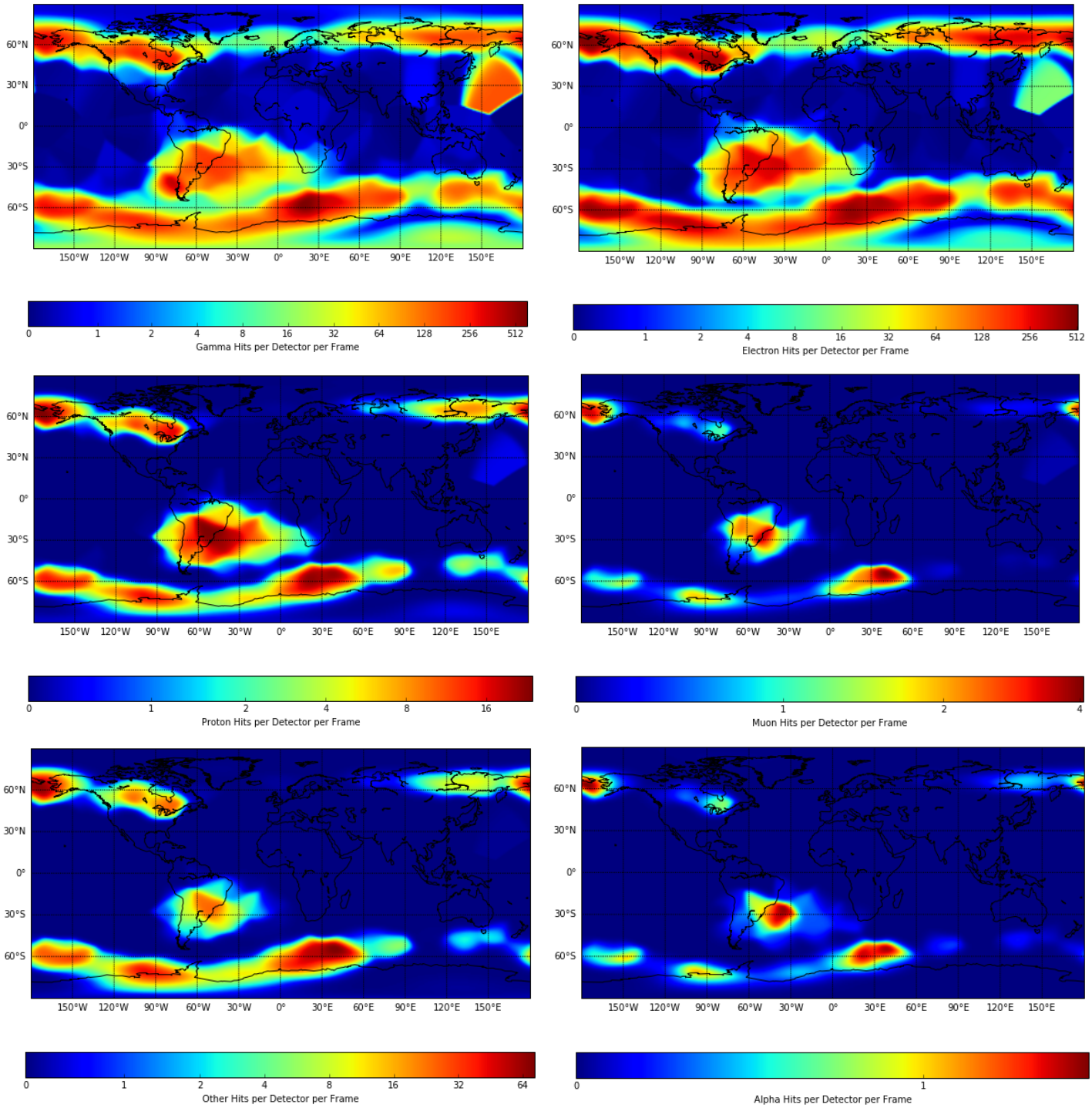
Initial Counts Map



This map shows the total particle counts from the selected data files. This initial map was created mainly to see the distribution of the data files. By looking at the map, it is possible to see that there is an even distribution of data files, yet the most prominent ones are the ones near the poles and in the South Atlantic. This map suggests that the neural network is successful in identifying general particles and that it performs as well as the old analysis algorithm for general count purposes.

However, this does not yet suggest that the neural network is successful at identifying the different particles that are expected from these high radiation zones. To test the model, the counts of each individual particle classes were added and stored in a CSV file. These were then plotted as interpolated logarithmic colour graphs to show the distribution of the various particles.

Interpolated Particle Counts Map



From the newly analysed frames, the results clearly suggest that the neural network is very successful at correctly identifying various particles in the South Atlantic Anomaly. The proportion of each particle type in the various radiation zones is highly accurate and is as predicted by several research papers and institutes. [27] [28]

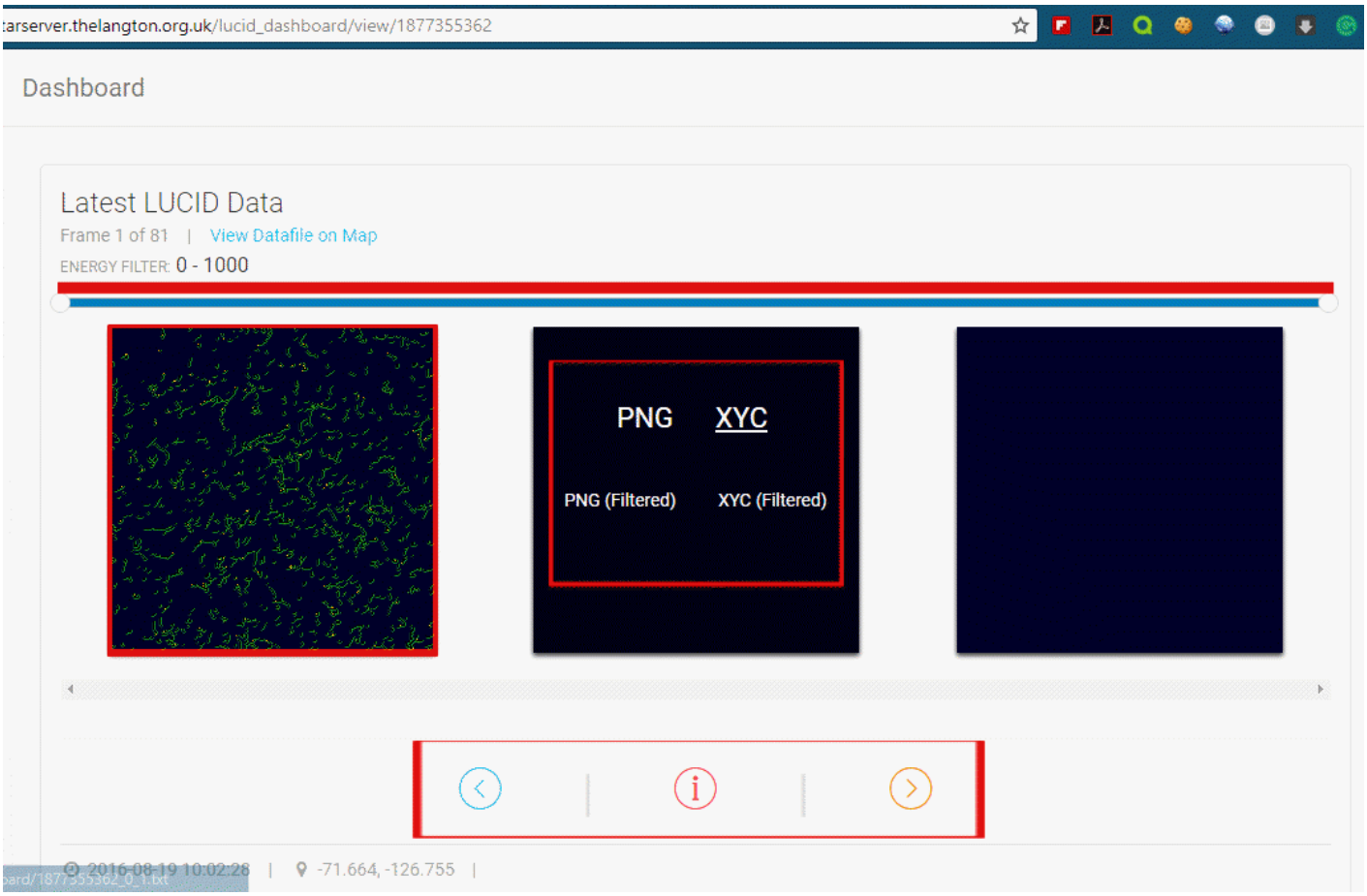
Conclusion

The results generated from the map generator strongly support the hypothesis of this investigation. Apart from providing evidence for the South Atlantic Anomaly, it also provides confidence in the LUCID data and the neural network designed for analysis. This whole endeavour opens scope for new research projects using the collected data and the analysis algorithm.

LUCID Dashboard Tests

Test No.	Page of Application	Actions	Expected Result	Result
1	Homepage	Hover on Frame Images	Display correct functional links and dark translucent overlay	Links were functional as expected and few debugs were required to ensure that the dark translucent overlay was working.
2	Homepage	Change Energy Slider	Image should be correctly updated upon the change of the slider.	Gen Image API is called correctly and energy values are filtered accordingly.
3	Homepage	Frame Navigation Buttons	Previous and next button change the frame number by reloading the page with new GET parameters and frames are updated in the viewing box.	The page is correctly reloaded and updated.
4	Homepage	Frame Metadata Button	The metadata is displayed in a Google Maps popup window.	The result is as expected. The frames are displayed on the map at its respective geolocation.
5	Data Files	Data Run Select	The dropdown allows the user to select the data run.	The whole list of runs is correctly dynamically generated in PHP.
6	Data Files	Location Search	Files in a specific radial region can be searched using an address as a centre point with a given value for the radius.	Files in the specified radial region are listed in the table.
7	Data Files	Data Files List	Files are listed correctly depending on GET parameters.	Files are listed correctly with links and metadata.
8	Radiation Map	Dropdown Selectors	Dropdown selectors are used to select a specific run and/or file which will allow the user to see files or frames on the map.	These work correctly as expected. Map markers are dynamically generated in JS depending on the user input.
9	Radiation Map	Location Search	Files in a specific radial region can be searched using an address as a centre point with a given value for the radius.	A red translucent circle surrounding the specified region is created and zoomed upon, allowing users to see files in that region.
10	Radiation Map	LUCID Tracker	The TDS-1 satellite is tracked based on the TLE of the satellite.	The satellite is tracked correctly as confirmed using NASA's website.

Homepage



All the key features of the homepage were thoroughly tested and reviewed. This included the following:

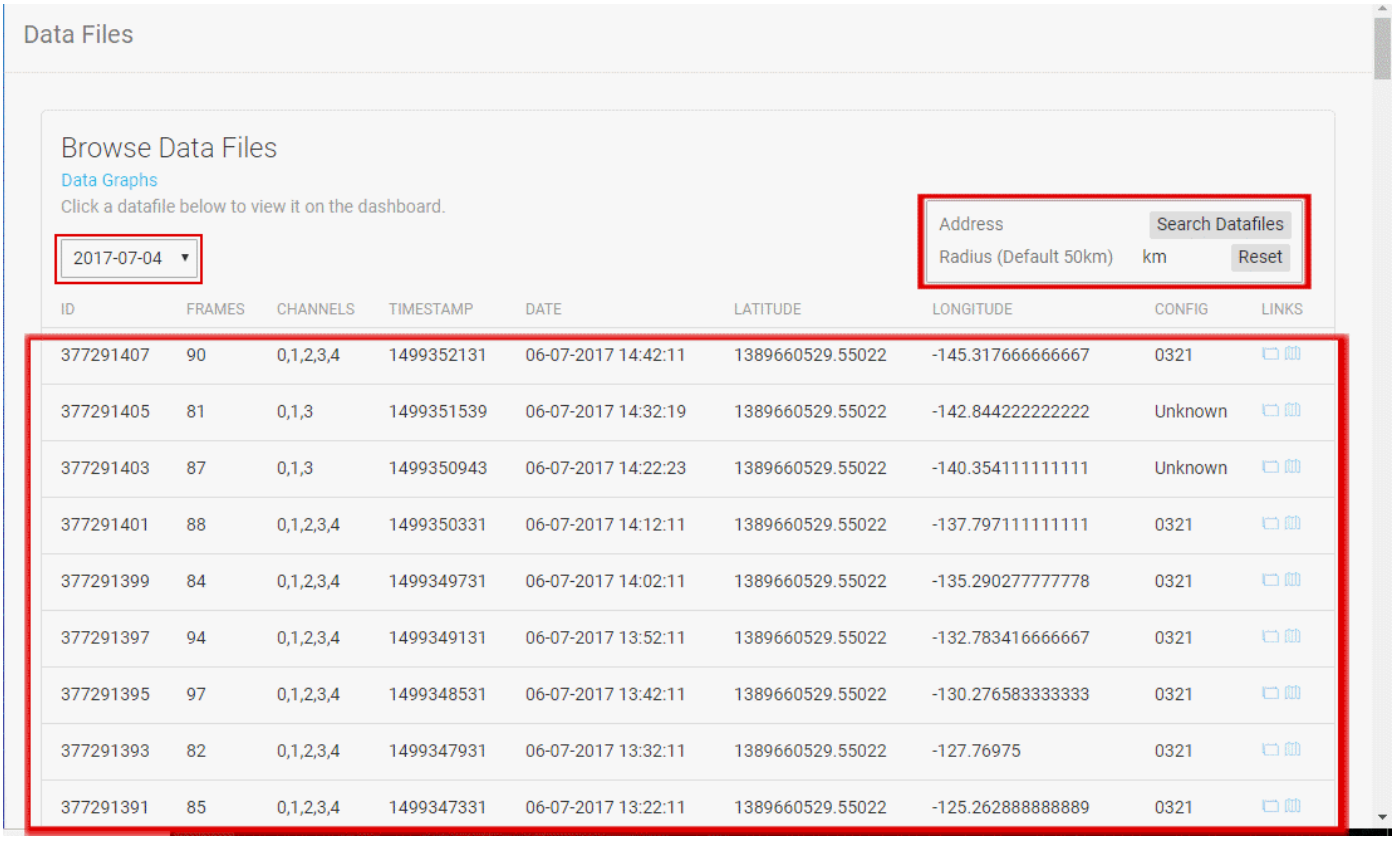
Display Links on Frame Image Hover
When the mouse hovers over a frame image, a translucent dark overlay is layered over the image and the links are then displayed on top of this. As the links are PHP generated, they needed to be tested. Algorithmically they should have worked and therefore after a few trials, it was confirmed that it was working as expected.

Image should change based on Energy Slider
The API call for generating a filtered image is called when the energy slider is changed and the image is updated. The energy slider allows users to hide either low energy or high energy pixels to show key features of the frame. After having tested the image generation API (gen_image.php), the only thing that needed to be tested was the client-side JavaScript using the Google Chrome console.

Frame Navigation
The frame navigation links were PHP generated. It works simply reloading the PHP with new GET parameters. The next button adds 1 to the frame number and the previous button subtracts 1 to the frame number unless the current frame number is the maximum number of frames in the file or the current fram number is 0. This was a simple conditional statement that was tested by outputting the frame number for different frames in PHP.

Viewing the Metadata of the Frames by clicking info
Using the Google Maps API, a popup window is opened that shows the frame location on a map with all of its metadata.

Data Files



Dropdown Selector

The selector for the data run is the same as the one used in the trainer application but instead of loading a list of file IDs, a list of data runs are loaded by calling the data API from a php function.

Location Search

The PHP page takes location GET parameters “address” and “radius”. These are optional parameters and therefore do not need to be set but if they are set then the files within the desired radius centred at the given address are returned. The radius is validated to ensure that it is an integer number of metres. The address cannot be validated as such and is therefore allowed to be any arbitrary string. If the address is invalid, the circle is not displayed.

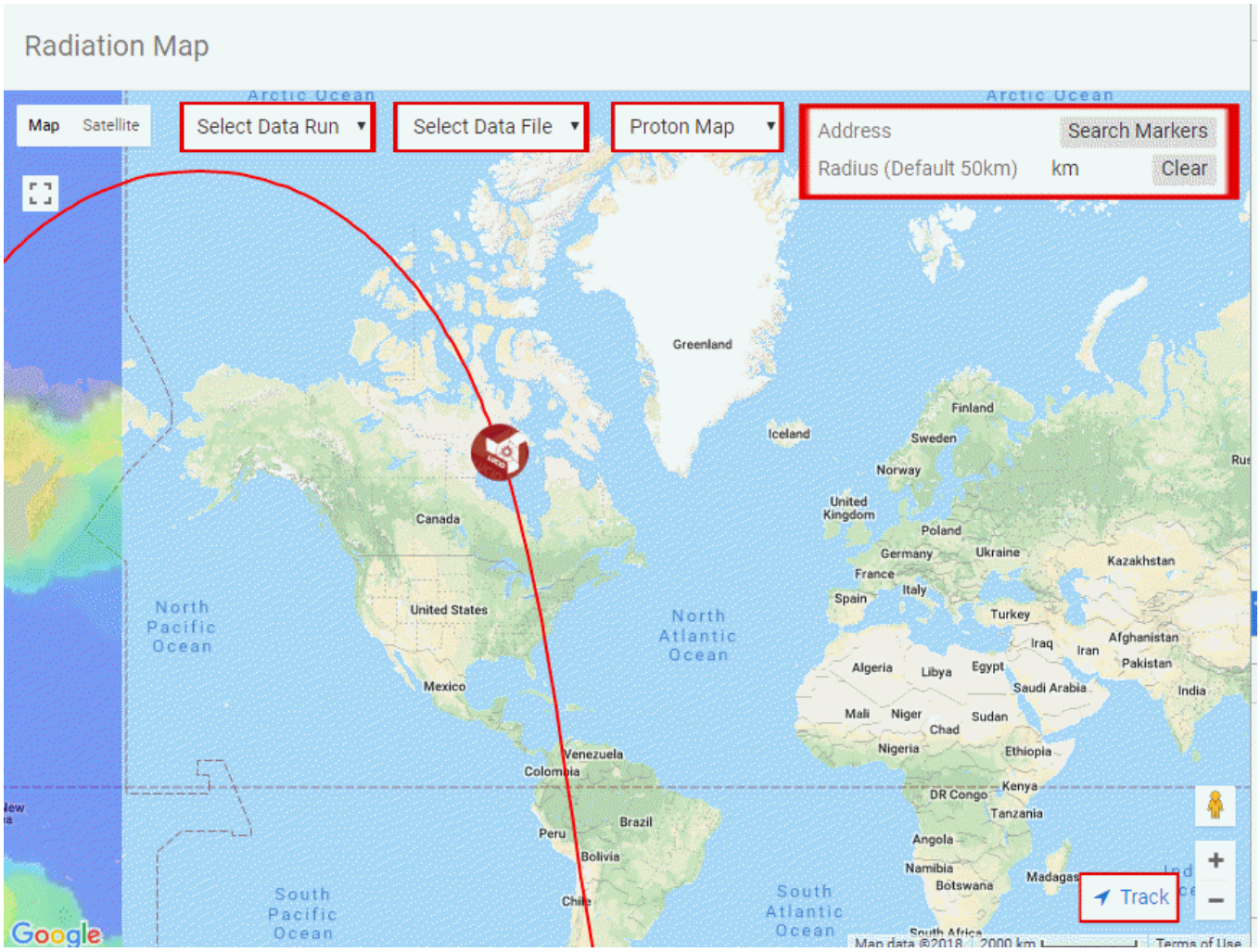
Data Files List

Depending on the GET parameters, the corresponding data files are returned. The files are either from the location search or from the selected data run depending on which one the user has selected. There are no user inputs for this part but it had to be tested to ensure that the correct files were being displayed.

Data Graphs and Docs

The data graphs and documentation page are dynamically generated but they are relatively static pages and therefore any testing was done using echo statements in server-side PHP during development. The documentation page did not require any testing as there are no user action calls on this page. The graphs were created using an API although the data processing for generating these graphs, had to be tested. This was relatively easy as the data is read from the database, formatted and processed into the required columns for each graph. This was much more simpler than the rest of the application as the data was static and no user input was required for doing any of this.

Radiation Map



All the key features of the radiation map were thoroughly tested and reviewed. This included the following:

Dropdown Selectors

The dropdowns “Select Data Run”, “Select Data File” and “Proton Map” were easy to test as the code was very similar to the dropdowns used in the LUCID Trainer. The dropdown selection for the data run was also used in the data files page of the dashboard . The dropdowns worked as expected and are shown in the testing of the Trainer application. The data that dynamically generated the dropdowns is loaded from the php function that calls the data API.

Location Search

The PHP page takes location GET parameters “address” and “radius”. These are optional parameters and therefore do not need to be set but if they are set then the map displays a red circle on the map with the desired radius centred at the given address. All the markers within this circle are the files that are within the specified region. This is similar to the one used in the data files page and the backend function for calculating the latitude and longitude and then filtering the files based on the geolocation is the same.

LUCID Tracker

The page uses the TLE (Two Line Element Set -> satellite details) file to get the satellite location every few seconds. The track button in the right hand corner fixates the screen such that mouse control is not allowed and the screen will follow the LUCID icon around its orbit. An external javascript library is used to get the location details of the satellite.

Evaluation

Classifiers Overview

The hyperparameters of the deep feed-forward neural network (DNN) were fully optimised through the process of trial and improvement. The only way of increasing the accuracy is to provide more labelled data to continue training the neural model.

Deep residual networks (DRNs) require a lot of computational power. Even training a relatively shallow residual network of 20 layers for 10 epochs, for 5 days on a multi-core CPU server with a 2.53GHz clock speed and 12GB RAM would not outperform a GPU training for 12 hours. A high-end GPU such as the Nvidia Tesla or the Nvidia Quadro GP100 can speed up training by almost 15 times and provide a high degree of accuracy. Not only that, they can also train much deeper residual networks with a low training time due to the high amounts of memory and the ability to parallel process. However, these GPUs are very expensive to buy and relatively expensive for electricity usage. The inaccessibility to a GPU meant that I could not explore this neural architecture far enough to exploit its full potential.

The problem of the lack of computational power also applied to training the convolutional neural networks (CNNs) however, the problem was much less difficult as CNNs are not deep in comparison to DRNs and therefore required less computational power to train. The training time was still quite long, approximately 4 days continuously running on the server for a mere 300 epochs. However, the high accuracy showed that the CNN has much greater potential as it uses the raw images for processing rather than the metrics calculated. The newly trained CNN model almost performs as well as the current analysis algorithm making this classifier highly suitable for further development.

SVM and KNN hyperparameters were not optimised and therefore they did not perform as well as the other neural networks yet they still managed to provide a moderate degree of accuracy. The results from the SVM show that the SVM had clearly overfit to the training data and therefore its overall predictive power was lost. By optimising the hyperparameters, this classifier could be thoroughly improved. The KNN, on the other hand did not overfit extensively and still provided a high degree of accuracy. Therefore, optimising the K parameter and increasing the training dataset would largely improve the accuracy of the classifier.

Decision trees (DT) and random forests (RF) had a surprisingly high accuracy. These tend to overfit to the training data but the high test accuracy suggests that these two classifiers can be plausibly used for analysing particles. It was more intriguing to notice that although the random forest classifier (the ensemble of decision trees) had a higher accuracy than a single decision tree, it was not greatly more accurate than the single decision tree. This also led to the conclusion that the generated training data was not sufficient or fully accounting and that more data would definitely increase the accuracy of all the classifiers.

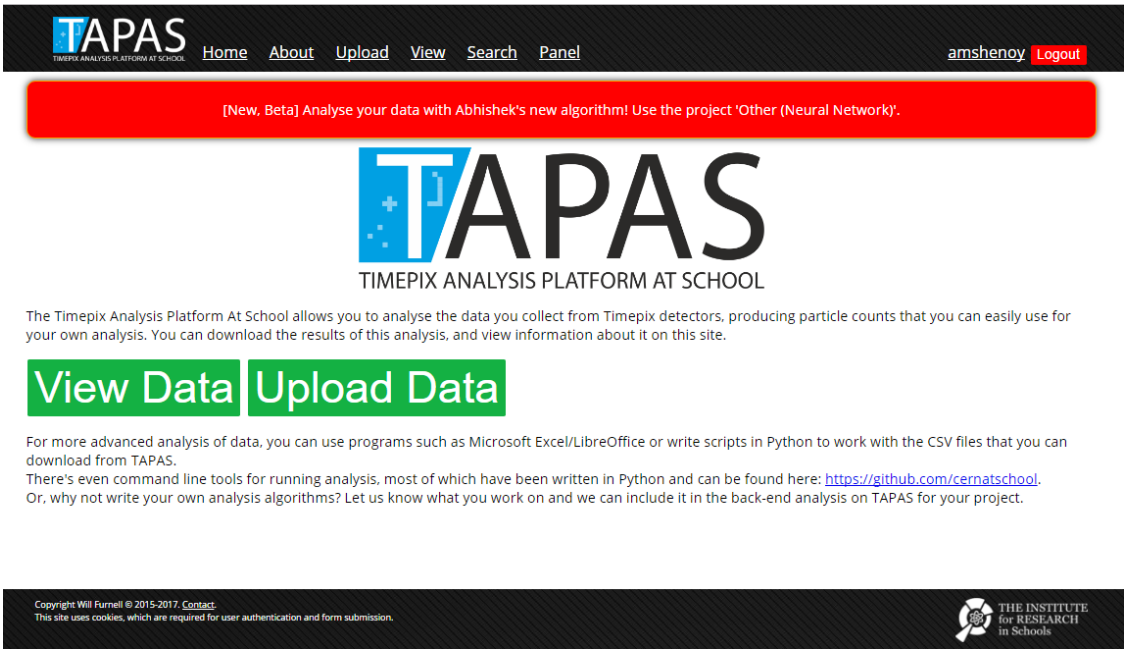
Neural Analysis

The deep neural network accurately classifies all of the common particles very easily and even uncommon ones are recognised to a high degree of accuracy. The advantage of this technique is that the model can be retrained on a new dataset from its current checkpoint allowing gradual improvement in accuracy over time. As the outputs are one-hot encoded, the decimal outputs are used as confidence values for each class that the particle could belong to.

The convolutional network that had been created and tested has the most potential out of all the machine learning techniques as it allows for the addition of more features and has the ability to increase drastically in accuracy for this classification problem.

To develop on the newly created deep neural network and to improve the classification accuracy, I combined the benchmark classifiers with high test accuracies (above 90%) alongside the neural network and the current algorithm so that the new system can be used to provide multiple predictions. The most common prediction from this set of prediction will be the most accurate prediction for the particle cluster.

The neural model and the benchmark classifiers with their respective APIs have been made available on GitHub for use by any institute:
https://amshenoy.github.io/lucid_neural_analysis
https://amshenoy.github.io/lucid_classifiers



Having successfully built the neural network and ensured its accuracy, it was integrated into TAPAS (Timepix Analysis Platform At School) to analyse particle tracks collected by students all around the UK for particle physics research projects (Figure 22).

Figure 22 - TAPAS

The integration of the neural network into TAPAS clearly shows the need for a good analysis algorithm to analyse particle tracks from radiation detectors in various different fields.
Due to the vast amounts of LUCID data, the particle data collected has many applications. The LUCID data could potentially be used to find out the source of radiation in space as well as the dose of radiation that astronauts have to experience in space. Apart from this, it may even be possible to create a computer-simulated model that could predict future radiation in space.

Analysis Improvements

To improve the overall accuracy, the amount of labelled data used to train the neural network needs to be increased. Since a human generates the labelled data, the process of training data production requires a lot of time and is not very efficient. This was the main problem in trying to achieve high accuracies with all the machine learning classifiers. Below I have outlined analysis methods that can be used to improve the analysis accuracy.

Energy Analysis

The deep neural network does not account for the distribution in energy values. This is because analysis using energy values is highly difficult as the pre-processing is dependent on shape and energy deposit and it is harder to describe parametrically. Although this problem could have been solved by using a convolutional neural network for image classification like the one I had tested, it would have been very slow to train and run for analysis on a web platform. Moreover, the LUCID Trainer was mainly developed for the deep neural network using metric analysis and therefore the energy values had been removed during the process of clustering. Hence, the convolutional neural network I developed did not analyse energy values.

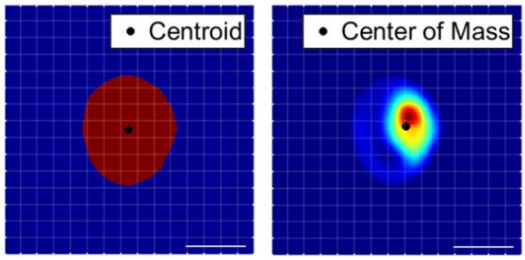


Figure 23 - Centre of Energy Deposit

For a metric-based system, the "centre of mass" (Figure 23) can be calculated along with the centroid and this can be used as another one of the inputs in the neural network. The current centroid calculation method only uses equally weighted pixels without energy values however by adding energy values as the weighting, the collision point of the particle can be determined and this may help a neural network to perform better classification.

Sub-System Analysis

The neural network can be further extended into a system of neural networks or a system of general machine learning classifiers whereby which the particles can even be given sub-classes for a greater discrepancy between alike particles.

Protons → High Energy, Low Energy

Electrons →High Energy, Low Energy

MIPs →Muons, Electrons (Misclassified as MIPs)

Others → Pions, Alpha, Heavy Ions, Particle Crossovers

The above sub-classes are only examples however a full sub-system analysis could almost implement a broad particle analysis spectrum.

Convolutional Analysis

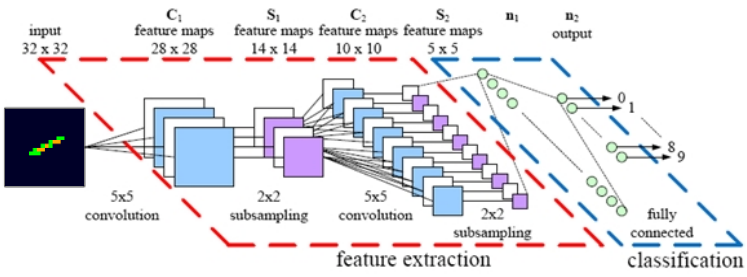


Figure 24 - Convolutional Analysis

Convolutional networks (Figure 24) are very useful as they can not only analyse energy values but they can also provide a deeper insight into what make particles different to each other. This can be visualised by viewing the filters learned by the network after training.

Summary

For a reasonably simple neural network, I was very successful at solving this classification problem. The main limiting factor was the computing power and therefore it was difficult to iteratively edit the design of the network. As an improvement, I would definitely like to use a GPU for training simply to provide more training iterations and therefore allowing the possibility for increased accuracy.

LUCID Dashboard Evaluation

This part of the project evaluates the LUCID Dashboard as a separate piece of software by itself whilst considering its role in the project as a whole.

Selection of Programming Language

After having created the LUCID Dashboard, I learnt NodeJS and found that the dashboard could have been better built with NodeJS. This was because NodeJS is asynchronous and therefore the web application would load much faster. PHP did make development much quicker and simpler as the processes could all be siphoned synchronously using PHP includes without duplicating code.

Method of Data Access

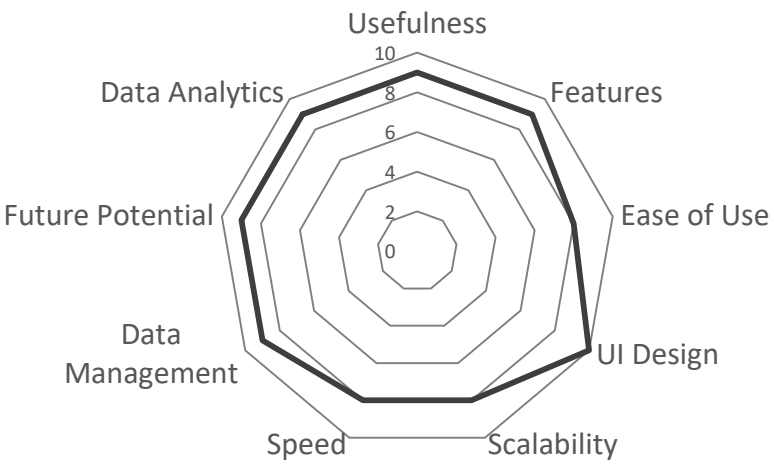
Also the dashboard was built around the public LUCID API which I had created and this was not the most efficient way. Directly accessing the data via file rerouting and connecting to the database directly would have been much faster but this could have created security issues. The current modular system that was designed allows each part of the project to be individually tested and overall improvements can be progressively increased. The API significantly simplified the development process and although the speed was not optimised, the application was still usable with ease.

User Interface Development

The base structure was an open-source HTML template designed with the idea of creating a social network. The integration of LUCID data analysis features was rather seamless and the additional features (energy filtering on images and XYC files, whole frame analysis, radiation map with data markers) that were added, extremely stretch the application's future potential.

The dashboard is available at: https://starserver.thelangton.org.uk/lucid_dashboard

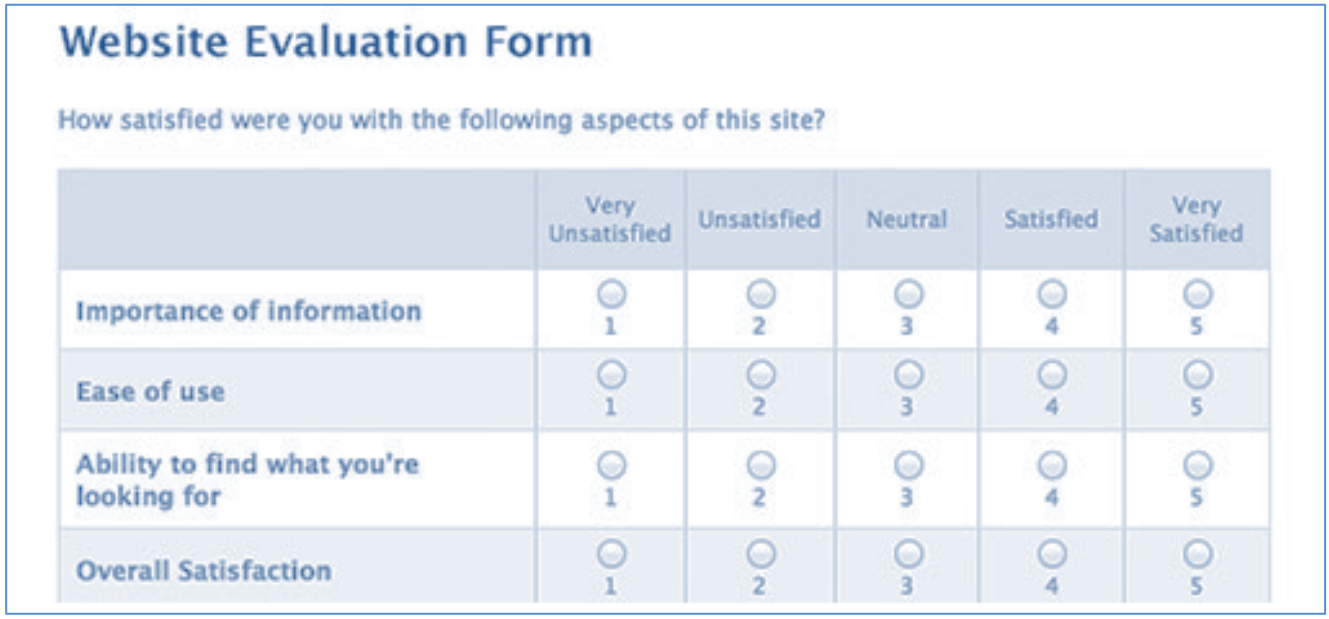
User Feedback



"This is a highly professional website and an excellent platform for the distribution of advanced scientific research and data. Navigation is clear and the data is readily accessible, with impressive data graphs already available." – Student Review

"This is rather cool! Well, for students who don't have a background in coding this provides an easy way to look through the LUCID data with all the fancy analysis features." – Teacher Review

The integrated neural network allows school students to analyse data without having to explicitly understand code making this a useful tool for students exploring the LUCID dataset.



Website Evaluation Form

How satisfied were you with the following aspects of this site?

	Very Unsatisfied	Unsatisfied	Neutral	Satisfied	Very Satisfied
Importance of information	1	2	3	4	5
Ease of use	1	2	3	4	5
Ability to find what you're looking for	1	2	3	4	5
Overall Satisfaction	1	2	3	4	5

To find out if I had accomplished my objective with regards to my users, I conducted a survey that asked 20 users about their experience with the LUCID Dashboard.

The results were as follows: (Score: Number of People)

Importance of Information: {1: 0 | 2: 0 | 3: 0 | 4: 11 | 5: 9}
Ease of Use: {1: 0 | 2: 0 | 3: 4 | 4: 8 | 5: 8}
Ability to Find Things: {1: 0 | 2: 2 | 3: 1 | 4: 9 | 5: 8}
Overall Satisfaction: {1: 0 | 2: 0 | 3: 2 | 4: 10 | 5: 8}

This clearly shows that overall users were very happy with my endeavour and in general although the dashboard had weaknesses, it was the best tool created for looking at LUCID data as it provided insights that could not be seen before without explicit programming.

Evaluation of Objectives

1. Data Collection – All the data and metadata must be accessible from a single access point.

The single access point was a REST API created in PHP to transmit JSON to the web browser. The API endpoint is fully developer-friendly. The metadata is still securely stored in a database with the only access being the API endpoint. This objective has been fully achieved as this primary stage is a fundamental part of the rest of the project.

1.1. Database Model - The data must be in a convenient and usable format.

The database model to store the LUCID data is in the third normal form (3NF) meaning that it is fully normalised. No data is unnecessarily duplicated and no field has more than one value. File references are saved as direct URLs in the database instead of file blobs to create a lightweight database to ensure that data transfer is fast and the data format is convenient.

1.1.1. Well-designed schema

The schema prevents the storage of unnecessary data such as the XYC files and there are no inconsistencies related to the database as no data is duplicated. No data is ever modified after creation as LUCID data is a static scientific record of data collection.

1.1.2. Easy to access and use data

The data is accessible for the general public and students and requires no special authentication or authorisation. This essentially open sources the whole LUCID dataset. As the database is simply an SQLite database, all programming languages are able to query the database and so this allows students of all abilities to be able to use and interpret the dataset.

1.2. API Model - Students must be able to easily view and analyse the data upon request.

The data can be easily accessed via the API and the request parameters allow the user to filter their query with ease without giving explicit access to the database for security reasons.

1.2.1. Simple URL access

All users have direct access to the REST API over the browser and allow them to choose which data attributes they wish to access. The data parsing is very easy in any programming language due to the the normalised format and it is even human-friendly without requiring any parsing.

1.2.2. Organised structural data

The schema is fully based on the database while ensuring encapsulation of any private data that should not be accessible. Overall, the objective was very well achieved. The whole API was designed having considered the order of different attributes, the data format, the method of callback and query parameters after student consultation.

2. Data Analysis – Particles can be analysed by a computer autonomously with minimal human input.

This section was essentially the main objective of the project as it adds value to the LUCID data. Being able to interpret or classify the data allows the data to be used for various research purposes.

2.1. Particle Identification Algorithm – Identify all the individual particles in a frame.

The process of identifying each particle (also known as blobbing or clustering) was crucial for even classifying a particle. If an individual particle cannot be picked out computationally then the whole process of classification cannot be done autonomously. This objective was achieved very early on in the project and lead to the eventual development of classification algorithms.

2.2. Feature Extraction from Particle Tracks – Pick out most important characteristics of specific particles.

This objective was created so that students can fully understand any particle track based on simple A-Level maths knowledge and how the physics concepts link to the mathematical features and hence the computational theory.

2.2.1. Creating Metrics

Various metrics were considered and the ones that differentiated particles the most were selected. These metrics also allow analytic classification through the use of if/else selection as an algorithm. This makes the generated metrics very versatile and useful for generally looking through LUCID data.

2.3. Neural Network for Classification – Machine learning for classifying particles.

After a lot of experimentation and small alterations, perceivably the most optimal network was created that allowed the particles to be classified with a higher accuracy than the current analytic algorithm. Machine learning in general was also more suitable for this classification problem although it was difficult to generate a large quantity of the labelled data.

2.3.1. Training Web Application – Generate training data for the machine learning classifiers.

The training web application was designed with lightweightedness and flexibility in mind. The application had to be simple and straightforward to not only use but also implement to allow the aim of classifying particles to progress quickly.

2.3.1.1. Appropriate Method Selection – Select best method for generating training data.

The best method was decided after a lot of thought before the development of the web application. The aim was to replicate a platform like Zooniverse solely for the LUCID project to build a training dataset from user classifications with a simple questionnaire. The chosen

method was the best method as the custom-design would allow it to readily connect to the LUCID database.

2.3.1.2. Efficient Database Model – Store training data in a database to access when needed.

After considering various schemas, the three best ones were evaluated and the overall best schematic was chosen for storing the training data. The database needed to be self-sufficient while storing metadata about which file the particle came from.

2.3.1.3. Utility Functions – Present all functions involved in the web application.

All the important functions such as converting XYCs into images and other fundamental functions that the web applications were built upon were made in multiple programming languages including PHP and Python to allow for future developments in either languages.

2.3.2. Neural Network Creation – Create and experiment with neural networks.

Before the development of particle classifiers, I had to research and implement rudimentary versions of the final architecture to ensure that the task was even plausible and to ensure that the size of the dataset that I could create would be enough for such algorithms. After having tested several preliminary networks, I was able to narrow down to the three I really wanted to develop and test out. These then became thee basis for the objective of classifying particle from the LUCID dataset.

2.3.2.1. Select Suitable Architecture – Select most appropriate model.

Having experimented and developed three very different neural networks, the most appropriate architecture had to be selected. The different architectures had been researched with innovation, accuracy and appropriateness in mind. The convolutional model was appropriate as it provided a pure image based method and the residual model was appropriate as residual models are very good for image classification. The simple multi-layer metric-based network was appropriate due to the lmited amount of training data, the relatively quick training speed and relatively fast for classifying particles..

2.3.2.2. Inputs & Outputs Encoding – Encoding inputs and outputs appropriately.

For all the other machine classifiers other than the neural networks, the outputs had to be encoded as values on a linear scale. For all the neural networks, the outputs could be one-hot encoded preventing any misinterpretations with the output and providing a probability-based prediction. A human could then check the prediction in case it was too close to 50%.

2.3.2.3. Hyperparameter Selections – Optimise the neural network for best results.

This objective was the hardest challenge as there were not limits to how much you could improve the neural network and yet it was very difficult to optimise the architecture while increasing accuracy while not overfitting to the training set. This objective was overall very well met as shown by the final accuracy graphs and confusion matrices.

2.3.2.4. Final Neural Network

The final neural network was a multi-layer metric-based feedforward network that uses the metrics directly for classifying the particles. The neural architecture with the most potential was the convolutional network due to its relative simplicity to the residual network and due to

its relative complexity to the multi-layer feedforward network. The convolutional network intrinsically analyses the energy of the particle tracks while considering geometry without any presumptions. The convolutional network is able to pick out higher and lower level features that may not have been described by the metrics used by the metric-based network. Due to limited computing power, the metric-based network was able to perform much better than the convolutional network.

2.3.3. Model Training & Testing – Show training and testing stages of the neural network.

While experimenting, I performed this several times to ensure that the neural networks were not only functional but also suitably accurate and appropriate. The original dataset was split into a training dataset and a test dataset with an appropriate ratio to prevent overfitting while maximising the size of the training data. This was crucial as it ultimately determined the accuracy of the network. The only way to improve on this objective was to generate more labelled data.

2.3.3.1. Machine Learning Classifiers Evaluation – Compare neural network with other machine learning classifiers.

To ensure that the limiting factor was the dataset and not the architecture, I had to compare the neural network with other machine learning classifiers. After having run the test data through the different algorithms, the summarised accuracy graph shows the accuracies of the newly created algorithms of which the metric-based neural network performed sufficiently well.

2.3.3.2. South Atlantic Anomaly – Show results relating to the original physics-based investigation.

To test the different algorithms on a real-life phenomenon, I generated a radiation map of the whole Earth for each of the different algorithms. The most significant map created by the neural network is present in this project. This proves that the algorithm is sufficiently able to apply the analysis for various different particles.

2.3.3.3. Model Saving & Loading – Easy-access library for student and programmer analysis.

To allow students to access the most accurate analysis algorithm, I saved the weights and the neural graph of the most accurate neural model produced to a file. Then I coded a program that would access this model and classify a given particle. This allows users to analyse data without having to train their own model. It also allows the metric-based network to encapsulate the interiors of the network and prevent unauthorised access to the code. The whole library is available to all students so that they can easily explore the LUCID dataset fulfilling the last objective of the analysis part of this project.

3. Data Visualisation – Build another web application for the presentation of all results related to the project.

Overall the dashboard compiles all of the LUCID research into one web application which any student can visit to explore the dataset. The various features of the web application allow users to perform functions that would otherwise be very difficult to implement such as the radiation map, exploring the dataset live on the map, getting statistics of the dataset and analysis of any given frame in any given file.

3.1. Show Latest Data Frames on Index Page

All the new frames that come in every two weeks are shown on the index page (also the dashboard homepage). There were many ways in which the data files could be ordered as well as presented. The best method was the chosen method as the latest frames were most preferred to be seen.This is because when the satellite was running, the data files collected would update every two weeks. The best method of showing the file was to display the first frames of the file and the user would then be able to scroll through the rest of the frames easily.

3.2. Provide Analysis of a Given Frame

Each frame, upon page load, is splitting into its individual clusters using the blobbing function. The analysis function from the neural analysis library is then called on each of the clusters and a summary count of the different type of particles is printed out for each frame.

This feature however was later removed as the analysis was being carried out on every page load and therefore the server load was increased. The new method for analysing a specific frame is an explicit analysis API call with a given ID, frame, channel and index. The API then returns the analysis output and this was the best method as it only analyses required frames and analysis on any busy frames can be given plenty of time to run.

3.3. Plot Data Frames and Files on a map

The data frames and files are plotted on a Google map depending on user selection. This allows the user to easily search for files in a specific location such as the South Atlantic Anomaly. This was probably one of the most important features requested and implemented in the LUCID Dashboard. The whole user interface was neatened up to the style of Google material design to make it fit seamlessly into the map. This made the design streamlined and more stylish in comparison to previous applications.

3.4. Show Collection Statistics on Graphs

The graphs were very useful as they provide visual analytics about the data without any processing requiring to be done by the user. This was another highly requested feature that completely obeyed the users requirements and was key to impressing the users in the user survey.

3.5. Provide Documentation of the API

All of the documentation is provided and any new updates to the API will be updated in the LUCID documentation page. The documentation is easy to use and very neat presented in a full table. No API key is needed as the APIs are meant to be fully public and allow anyone to use the application. The REST API standards were fully recognised and accepted while designing these APIs.

Appendix

Figure 25 - TechDemo Sat-1 (TDS-1)

LUCID

LUCID (Langton Ultimate Cosmic Ray Intensity Detector) [1] is a device created by students from Simon Langton to detect radiation in space. The payload is a part of the Space Environment Suite [2] on the satellite TechDemoSat-1 (TDS-1). The satellite was launched on the 8th of July 2014 from the Baikonur Cosmodrome in Kazakhstan. Currently the project is funded by the Institute for Research in Schools (IRIS) with the courtesy of Dr Parker (Director of IRIS). Surrey Satellite Technology Limited has developed the student design in collaboration with successive years of students and scientists at CERN.

There are 5 Timepix chips on the device created by the Medipix collaboration; 4 of them on the outer sides of the open cube and the 5th one at the bottom of the open cube as shown in Figure 2.

Figure 26 - LUCID - 5 x Timepix Chips

Timepix Chip

To understand the basis of particle analysis, it is essential to understand how the detector actually works. This will help to understand the properties and the limitations of the particle tracks.

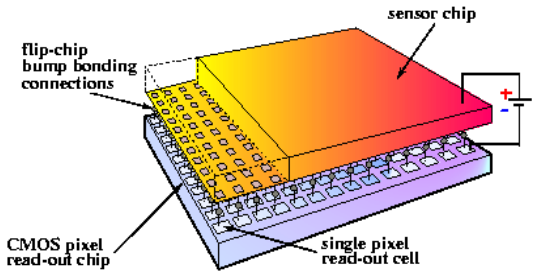


Figure 27 - Timepix Chip

Timepix chips (Figure 27) [3] have a pixel matrix of 256 x 256 (65,536 pixels) with each of the square pixels having only a side length of *55 μm*. The entire active area of the silicon sensor chip is $2cm^2$. Upon collision of an ionising particle onto the surface of the chip, a charge is deposited into the silicon layer. The charge is gradually removed by the potential difference supplied across the detector called the threshold voltage. This process takes place by using CMOS circuits (complementary metal-oxide semiconductor) which are built into the chip.

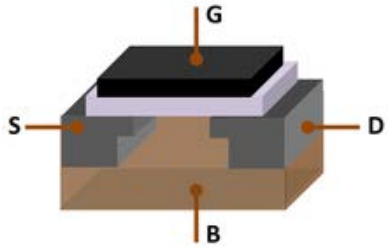


Figure 28 - MOSFET

The CMOS circuits consist of billions of p-type and n-type MOSFETs (metal–oxide–semiconductor field-effect transistors) to create logic gates and other circuitry. The three contacts on a MOSFET (shown in Figure 28) [4] allow it to be used like a switch. The source (S) supplies the electricity and depending on the electrical voltage on the gate terminal (G), the voltage through the drain terminal (D) can be altered. The gate is separated from the body by an insulating layer (white). Increasing the potential difference on the gate allows electrons to flow between the source and the drain terminals. Using this mechanism, each of the pixels can then record the time it takes for the charge to be reduced below a certain threshold and the output is known as Time over Threshold (ToT).

Higher energy particles deposit a greater charge than lower energy particles, so the Time over Threshold is much greater as it takes longer for the charge to be reduced below the configured threshold. Therefore, Time-over-Threshold (ToT) is known as the energy value and is measured by counting the clock ticks. The Timepix chip has a single energy threshold and is stored using 4 bits for the whole pixel matrix. All of the data is accumulated in a 13-bit counter per pixel (1kB per pixel). Each pixel overflows at 11810 counts and the chip data is read out only after exposure to prevent dead time and the loss of data.

Particle Physics

Particle Tracks

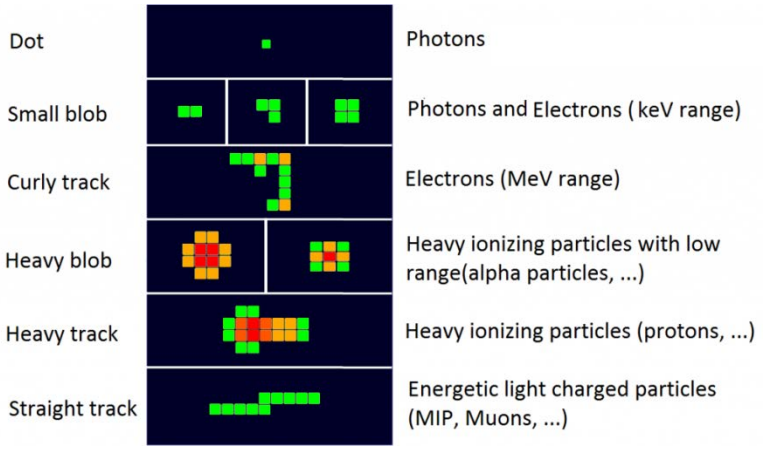


Figure 29 - Particle Tracks

There are several variations in the particle tracks received by LUCID however, the main properties of the tracks are shown in (Figure 29). These are very useful in identifying common particles however they are not very help to judge unrecognised particles.

The energy values are very important in understanding the direction of travel before collision and the point of collision of the particle. These can be used to identify unfamiliar particles. However, these properties are difficult to describe abstractly or mathematically. Abstraction can also lead to incorrect inferences about the type of particle and therefore it is simpler and more accurate to determine a particle by only its track shape rather than the energy values. The energy value may be useful upon deeper analysis, since the properties of the particle are directly correlated with the energy deposit from the particle onto the detector chip.

Energy Values

The mean energy loss of a moving particle is also known as the stopping power of the material it is travelling through. This energy loss is calculated using the Bethe Formula (Figure 30Figure 30 - Bethe Formula). For electrons, the energy loss requires relativistic corrections due to their small mass. Fast charged particles moving through matter interact with the electrons of the atoms in the material. The interaction excites or ionizes the atoms, leading to an energy loss of the travelling particle.

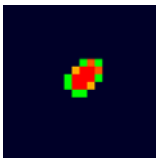
$$-\left\langle \frac{dE}{dx} \right\rangle = \frac{4\pi}{m_e c^2} \cdot \frac{nz^2}{\beta^2} \cdot \left(\frac{e^2}{4\pi\varepsilon_0} \right)^2 \cdot \left[\ln\left(\frac{2m_e c^2 \beta^2}{I \cdot (1-\beta^2)} \right) - \beta^2 \right]$$

Figure 30 - Bethe Formula

$v = Velocity$	$c = Speed\ of\ Light$
$z = Relative\ Charge$	$e = Electron\ Charge$
$E = Energy$	$x = Distance$
$n = Electron\ Density\ of\ Material$	$I = Mean\ Excitation\ Potential$
$c = Speed\ of\ Light$	$\beta = v/c$
$\mathrm{m}_0 = Rest\ Mass$	$\varepsilon_0 = Vacuum\ Permittivity$

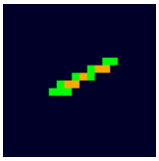
Common Particles

Alpha



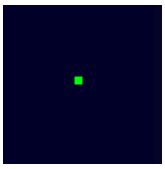
Alpha particles have one of the most distinct particle tracks as they form round circular blobs. They are heavy particles made up of 2 protons and 2 neutrons and therefore have a relative mass of 4. The +2 relative charge means that upon collision, more of the kinetic energy of the particle is deposited into the silicon layer. The velocity of alpha particles also tends to be much lower as the particle has to have more kinetic energy for its mass. From the Bethe Formula, as the velocity increases the amount of energy deposited decreases.

Electron



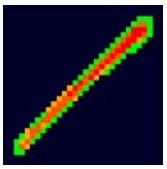
The relativistic Bethe formula is required to calculate the energy deposit of electrons as the equation needs to account for the fact that electrons have a much smaller mass than heavy particles like protons and that the electron is interacting with identical particles in the atoms of the material it is travelling through. The interaction between the incoming electron and the atomic electron allows for the possibility of exchange of the two particles.

Gamma



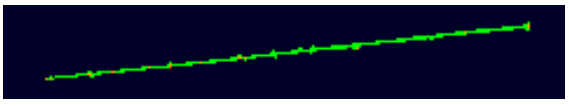
Since gamma particles are a part of the electromagnetic spectrum, these particles travel at the speed of light and therefore calculating the energy deposit for gammas requires the relativistic version of the formula. Since these particles are relativistic, the energy deposit increases logarithmically due to the transversal component of the electric field.

Proton



Protons have a relative charge of +1 and a relative mass of 1. The high charge means that they deposit a lot of energy. However due to the lower mass, they tend to have a high velocity for the same amount of kinetic energy and therefore do not deposit as much energy as a heavy ion or an alpha particle.

Muon



These are minimum ionising particles that have approximately 200 times as much mass as an electron however they are still 10 times lighter than a proton. Muons travel at high velocities and this means that they undergo time dilation as predicted by special relativity allowing them to travel long distances even though their decay time is much smaller. Due to their high velocity, they tend to deposit very little energy into the chip but at the same time their high mass makes the particle tracks thicker than those of beta particles. Due to the high velocity of muons, the particle tracks they produce are highly linear.

Other

These particles are unrecognised or cannot be classified as any of the above particles due to its irregular particle track. These are often heavy ions, multiple particles, particle decay or just unknown particles. Misclassifications are relatively easy for this category due to the varied range of possibilities and therefore this category will mainly be used as a discriminator between recognised and unrecognised particles.

References

[1] Surrey Satellite Technology Limited, “LUCID,” SSTL, [Online]. Available: http://www.sstl.co.uk/Blog/February-2013/TechDemoSat-1-s-LUCID--a-novel-cosmic-ray-detector. [Accessed 30 November 2016].

[2] Surrey Satellite Technology Limited, “TechDemoSat-1 Payloads,” [Online]. Available: http://www.sstl.co.uk/Missions/TechDemoSat-1--Launched-2014/TechDemoSat-1/TechDemoSat-1--Payloads. [Accessed 30 November 2016].

[3] X. Llopart, R. Ballabriga, M. Campbell, L. Tlustos and W. Wong, “Timepix, a 65k programmable pixel readout chip for arrival time, energy and/or photon counting measurements,” in *Nuclear Instruments and Methods in Physics Research Section A*, Elsevier, 2007.

[4] “MOSFET as a Switch,” ELPROCUS, [Online]. Available: https://www.elprocus.com/mosfet-as-a-switch-circuit-diagram-free-circuits/. [Accessed 5 July 2017].

[5] C. Hewitt, “PTR Generator,” 2017. [Online]. Available: https://github.com/amshenoy/ptrgen.

[6] C. Hewitt, W. Furnell and A. Shenoy, “LUCID Utils,” 2017. [Online]. Available: https://github.com/amshenoy/lucid-utils.

[7] R. Pierce, “Vectors Dot Product,” Math Is Fun, [Online]. Available: https://www.mathsisfun.com/algebra/vectors-dot-product.html. [Accessed 2 March 2017].

[8] P. Roelants, “Cross-Entropy Cost Function,” [Online]. Available: http://peterroelants.github.io/posts/neural_network_implementation_intermezzo02/. [Accessed 6 March 2017].

[9] Artelnics, “5 Algorithms to train a Neural Network,” [Online]. Available: https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network. [Accessed 11 March 2017].

[10] S. Ruder, “Optimizing Gradient Descent,” 19 January 2016. [Online]. Available: http://sebastianruder.com/optimizing-gradient-descent/. [Accessed 15 March 2017].

[11] Wolfram MathWorld, “Taylor Series,” Wolfram, [Online]. Available: http://mathworld.wolfram.com/TaylorSeries.html. [Accessed 16 March 2017].

[12] A. Gibiansky, “Conjugate Gradient,” 2 February 2014. [Online]. Available: http://andrew.gibiansky.com/blog/machine-learning/conjugate-gradient/. [Accessed 18 March 2017].

[13] A. Gibiansky, “Hessian Free Optimization,” 13 February 2014. [Online]. Available: http://andrew.gibiansky.com/blog/machine-learning/hessian-free-optimization/. [Accessed 22 March 2017].

[14] SDSU Library, “Conjugate Gradient Algorithms,” [Online]. Available: http://edoras.sdsu.edu/doc/matlab/toolbox/nnet/backpr58.html. [Accessed 25 March 2017].

[15] Asimov Institute, “Neural Network Zoo,” Asimov Institute, 14 September 2016. [Online]. Available:

http://www.asimovinstitute.org/neural-network-zoo/. [Accessed 29 March 2017].

[16] A. Karpathy, “Convolutional Networks,” Stanford University, [Online]. Available: http://cs231n.github.io/convolutional-networks/. [Accessed 4 April 2017].

[17] M. Hollemans, “Convolutional Networks on the iPhone with VGGNet,” 30 August 2016. [Online]. Available: http://machinethink.net/blog/convolutional-neural-networks-on-the-iphone-with-vggnet/. [Accessed 7 April 2017].

[18] L. A. Santos, “Residual Net,” [Online]. Available: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/residual_net.html. [Accessed 10 April 2017].

[19] Analytics Vidhya, “Complete Tutorial on Tree Based Modeling,” 12 April 2016. [Online]. Available: https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/. [Accessed 14 April 2017].

[20] “Random Forest Template,” TIBCO Software Incorporated, [Online]. Available: https://community.tibco.com/modules/random-forest-template-tibco-spotfirer. [Accessed 15 April 2017].

[21] B. DeWilde, “Classification of Hand-written Digits,” 26 August 2012. [Online]. Available: http://bdewilde.github.io/blog/blogger/2012/10/26/classification-of-hand-written-digits-3/. [Accessed 18 April 2017].

[22] M. Antonelli, “Support Vector Machine,” [Online]. Available: http://www.marioantonelli.it/svm. [Accessed 19 April 2017].

[23] “SVMDA - Eigenvector Documentation Wiki,” 16 May 2016. [Online]. Available: http://wiki.eigenvector.com/index.php?title=Svmda. [Accessed 20 April 2017].

[24] “Zooniverse,” [Online]. Available: https://www.zooniverse.org/about. [Accessed 22 April 2017].

[25] A. Karpathy, P. Abbeel, G. Brockman, P. Chen, V. Cheung, R. Duan, I. Goodfellow, D. Kingma, J. Ho, R. Houthooft, T. Salimans, J. Schulman, I. Sutskever and W. Zaremba, “Generative Models,” Open AI, 16 June 2016. [Online]. Available: https://openai.com/blog/generative-models/. [Accessed 22 April 2017].

[26] GridPP Collaboration, “GridPP,” [Online]. Available: https://www.gridpp.ac.uk/. [Accessed 18 May 2017].

[27] A. F. R. Laboratory, “Energetic Proton Maps for South Atlantic Anomaly,” 27 July 2008. [Online]. Available: http://www.dtic.mil/dtic/tr/fulltext/u2/a485155.pdf. [Accessed 13 June 2017].

[28] Vernov, S. N., Gorchakov, E. V., Shavrin, P. I., & Sharvina, K. N.;, “Radiation Belts in the Region of the South-Atlantic Magnetic Anomaly,” [Online]. Available: http://articles.adsabs.harvard.edu//full/1967SSRv....7..490V/0000491.000.html. [Accessed 16 June 2017].

[29] CreativeTim, “Light Bootstrap Dashboard Design,” [Online]. Available: https://github.com/amshenoy/light-bootstrap-dashboard.

Figure Table