Deep Learning in Particle Physics

Fatemeh Abbasi Razgaleh

Contents

1 Introduction

1.1 Background

Approximately 13.7 billion years ago a massive explosion – known as the Big Bang - created all the matter in our universe. Initially, the universe was extremely hot, but shortly after, it started to cool down. During this time, the fundamental particles of our universe were formed. At first, when the right conditions were met quarks and electrons were created, and later protons and neutrons. A million years later, the Milky Way was born [5].

Figure 1: The Early Universe [18]

Particle physicists are deeply interested in these first moments of the newly born universe. To study these particles and the forces between them, physicists have tried to recreate the initial conditions of our universe, using large-scale experiments such as the Large Hadron Collider (LHC) at CERN [16] or complex simulations such as Monte Carlo Simulations [17]. So far, we have been able to classify and detect these elementary particles using a theory called the Standard Model (Figure 3) [6]. Physicists face a considerable challenge when it comes to classifying particles in experiments that involve numerous background noises and various experimental factors to consider. This is where the integration of Machine Learning tools becomes crucial in overcoming this obstacle.

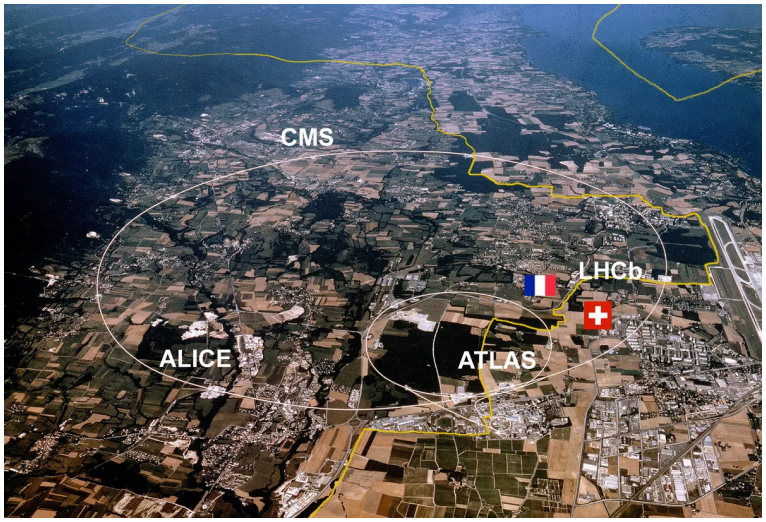

Figure 2: Different CERN Experiments [10]

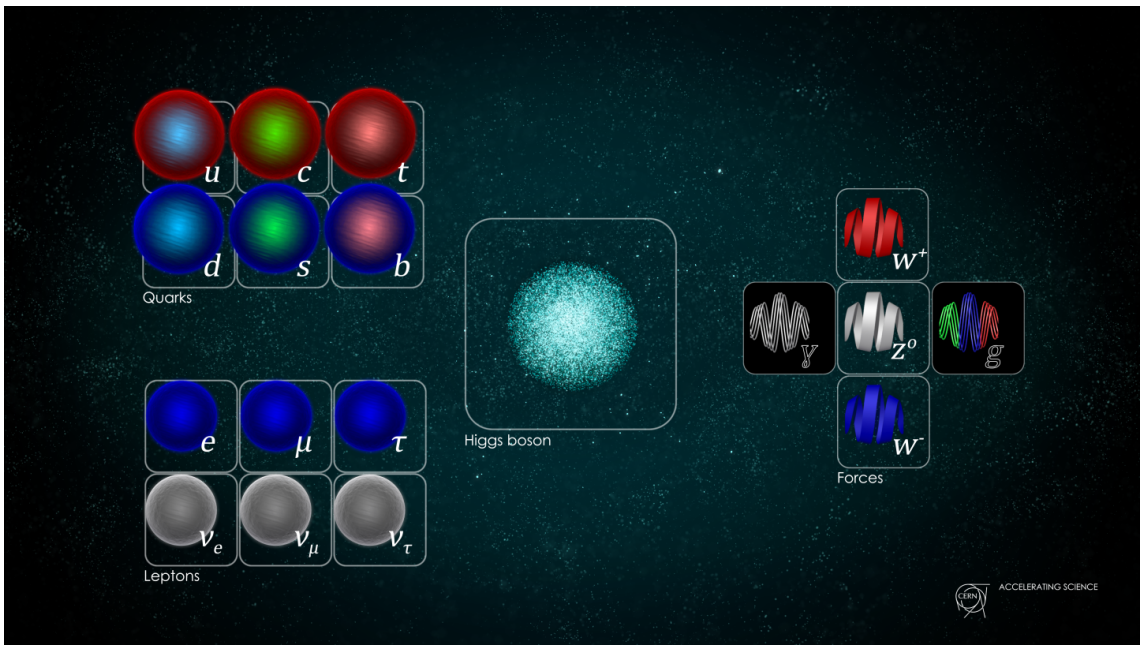

Figure 3: The Standard Model [6]

1.2 Overview

SUSY which is an acronym for supersymmetry is a theory that predicts a space-time symmetry in such a way that there's a partner particle for each particle in the standard model (Figure 3). This project aims to use the latest deep learning tools to classify the SUSY particles from background noise in simulated data from Monte Carlo Simulation - a complex computational algorithmic model used to predict all the possible outcomes of an uncertain event, and in return, however small way, help answer the fundamental questions we have about our universe.

Inspired by the original paper on the dataset, "Searching for Exotic Particles in High-Energy Physics with Deep Learning[1] ", I decided to explore the ways machine learning and deep learning tools can be used in scientific research. In this classification problem, I am eager to delve into creating the best neural network model. This will allow me to identify the most suitable deep learning tool for this specific classification task, considering factors such as efficiency and accuracy.

This report begins by examining multiple papers that discuss the application of machine learning tools in particle physics or physics in general. These papers serve as valuable sources of inspiration and learning, helping me gain a deeper understanding of the topic.

Afterward, the focus shifts to the project itself, where I outline its design and evaluation plan. This section highlights the structure and goals of the project, providing a clear direction for its development.

Next, I present the implementation of the machine learning model, showcasing the project's capabilities.

Later, I thoroughly evaluate the project and the machine learning model's performance.

Finally, I made an encompassing conclusion and summary of the project while highlighting area of further development in the future.

2 Literature Review

In this section, I will provide a brief overview of several papers that explore the application of machine learning in the context of SUSY particles, particle physics, or physics in general.

2.1 Searching for Exotic Particles in High-Energy Physics with Deep Learning

This is the original paper that used the SUSY dataset [14]. In this paper, the authors provide sufficient evidence that using deep learning methods compared to shallow neural networks provides greater accuracy as much as 8% in total for two separate datasets: Higgs and SUSY. They also highlight another advantage of deep learning methods is that there's no need for manually constructed inputs or high-level features such as the 10 high-level features found on the SUSY dataset.

In the case of the SUSY dataset, the authors created a "five-layer neural network with 300 hidden units in each layer, a learning rate of 0.05 and 300 hidden units in each layer and a weight decay coefficient of 1×10^{-5}"[1].

They compared the above method to a more common approach of shallow networks: "a feed-forward neural network with a single hidden layer and boosted decision trees" using a widely-used TMVA package [1] - a CERN library that provides ML tools [7].

The evaluation method chosen by the authors was AUC (Area Under Characteristic curve) and concluded that Deep Networks (DNs) again perform better than shallow networks even though the improvement is less dramatic than the Higgs dataset - however, the difference is statistically important [1]. The authors urged the 'physicists who have reluctantly accepted the limitation of the shallow networks'[1] to use deep learning methods where there is no need to create many high-level features to achieve a decent accuracy.

Finally, as mentioned earlier, since the publication of this paper roughly a decade ago using a highly specialized ML library, I aim to explore the latest machine learning and deep learning models using modern and common libraries like Scikit-Learn, TensorFlow, or PyTorch. Additionally, their comparison was limited to shallow neural networks and Bayesian decision trees. In light of this, I hope to achieve comparable accuracy results, identifying the best model and hyperparameters that suit this classification problem.

2.2 The BSM-AI Project: SUSY-AI

The authors of this paper created a Python program called SUSY-AI (link) that can determine if a specific particle physics theoretical model matches the experimental data gathered from LHC [16] in context of Supersymmetric Standard Model (MSSM) with 93.8% confidence level. This is called the exclusion of a model and it can be done by comparing theoretical model benchmarks to available data. According to the paper they used a random forest classifier trained on 310,000 data points of the phenomenological Minimal Supersymmetric Standard Model (pMSSM) with a reliable classification accuracy of 93.8%. The classifier was tested on many different MSSM models such as "the 19-dimensional pMSSM, the 6-dimensional natural SUSY model, and the 5-dimensional constrained MSSM" [4].

Although this paper does not use the SUSY dataset chosen in this project and it is not a binary classification problem, it provides great insight into the development of machine learning methods used in the field of particle physics.

2.3 Bayesian Neural Networks for Fast SUSY Predictions

In this paper, the authors use Bayesian neural networks (BNNs) to accurately model the mapping from the high dimensional spaces beyond the standard model (BSM) theories such as phenomenological minimal supersymmetric standard model (pMSSM) to their prediction. The pMSSM is a Beyond the Standard Model (BSM) theory with 19 free parameters, and the model generated by machine learning captures its predictions. The machine learning method was used to predict three quantities [11] :

1. "Cross sections for arbitrary pMSSM parameter points"
2. "The mass of the associated lightest neutral Higgs boson"
3. "The theoretical viability of the parameter points"

They were able to achieve an average percent error of 3.34% or lower for the prediction of all these three quantities and a high F1 score of 0.957 and recall score of 0.987. The researchers suggested the modeling process is significantly faster compared to traditional supersymmetry codes, as much as 50,000 to 16.5 million times faster, which further showcases the impact of machine learning tools in particle physics, especially beyond standard model (BSM) theories [11]. This paper alongside the BSM-AI project [4] suggests machine learning tools are getting widely used to quickly verify theoretical model predictions with experimental data.

This recent paper provides an overview of successful applications of machine learning tools in high-energy physics, showcasing their potential in the field of physics. It highlights various instances where machine learning methods have been effectively utilized, demonstrating the promising prospects they offer.

2.4 SCYNet: Testing Supersymmetric Models at the LHC with Neural Networks

The researcher in this paper used neural networks to provide fast evaluations of how much the supersymmetric models align with LHC data. To do this, they calculated the profile likelihood ratio which is a statistical measure used to assess the compatibility of a specific parameter value within a statistical model.

They used two different neural network approaches: The first neural network has been trained on an '11-dimensional phenomenological Minimal Supersymmetric Standard Model (pMSSM-11)'[2]. The second neural network has been trained on 'model-independent signature-related objects, such as energies and particle multiplicities'[2].

The authors were successful in providing a much faster evaluation of theoretical models using neural networks compared to the old methods that used very time-consuming complex simulations. According to the paper, "the mean error of both neural network approaches lies in the range of 1.3 – 1.7" with a relative precision of 2% – 3% [2].

2.5 Machine Learning for Event Selection in High Energy Physics

The authors of this paper primarily focus on using machine learning in event selection in high-energy physics, to differentiate the "particles of interest (signal) from other event-producing particles (background)" [15]. They describe their two-step approach in the paper as the following:

First, they use supervised learning for the event selection expecting minimal success since supervised learning assumes the highest accuracy result is the best result which is not true in this case due to the sensitivity of such analysis to background noise. Second, they showcase a novel method that employs stochastic optimization techniques to directly explore selectors that maximize either the precision of top quark mass measurements or the sensitivity to the existence of the Higgs boson [15].

The paper suggests great improvement using the second approach compare to the first one in both tasks. 29% improvement in lower uncertainty for top quark mass measurement and 37% lower scale factor for the Higgs Boson search.

Finally, this paper uses machine learning in a similar fashion that I intend to use -binary classification of signal to noise and showcases yet another way to use machine learning methods to achieve higher certainty.

2.6 Artificial Intelligence — Applications in High Energy and Nuclear Physics

This paper provides a brief overview of the different artificial intelligence approaches that can be used in high-energy and nuclear physics. The author summarises the different approaches as [13]:

1. "Wavelet Analysis: smoothing spectra in small-angle neutron scattering"
2. "Support Vector Machines: classification"
3. "Neural Networks: event selection, energy reconstruction, tracking, particle identification, trigger"

Finally, this paper offered a concise and insightful review of the various approaches used and showcased their successful applications. It further strengthens the confidence in utilizing artificial intelligence in physics research.

2.7 Jet Substructure at the Large Hadron Collider

This paper is a review of recent advances in theory and machine learning applications in particle physics [12]. The authors of this paper highlight the importance of Jet Substructure in the forefront of particle physics at LHC.

Machine learning techniques such as classification and regression can be used for Jet Tagging and Anomaly detection. Furthermore machine learning techniques can be used in Jet simulation such as "Matrix Element Calculations, Parton Shower Models, Fast Simulations" and even "Beyond Simulations"[12].

2.8 Machine and Deep Learning Applications in Particle Physics

This paper is summary of all the recent machine learning techniques that can be used in particle physics such as "Boosted Decision Tree, Neural Network, Artificial Neural Network, Multilayer Perceptron, Deep Neural Network, Convolutional Neural Network, Recurrent Neural Network, Long Short-Term Memory, Autoencoder, Instance Normalization, and Stochastic Gradient Descent" [3].

This is very extensive report consisted of an overview of many deep learning methods. The author of the paper gives a word of caution about the use of "black box" models and how they are "are good at learning correlations, not necessarily at helping to understand the underlying causations"[3]. The author also give a promising outlook towards the future of machine learning techniques and artificial intelligence used in Particle Physics.

2.9 Deep Learning and Its Application to LHC Physics

Yet another paper highlighting the application of machine learning in high energy physics and discussing their concerns and hopes for its prospects. The summaries a "survey of applications"[9] such as "Event Selection and High-Level Physics Tasks, Jet Classification, Tracking, and Fast Simulation"[9]. One of the concerns mentioned in the paper is that usually these machine learning models are specifically created and optimized for a particular task that is far behind the goal of the physics research at hand. Another concern that

was mentioned was the effects of model's performance on high precision physics problems. Finally, the authors emphasized on the impact of deep learning methods on data analytic at LHC, high energy physics, and particle physics and how it have started a new space of collaboration between machine learning experts and physicists.

2.10 Literature Review Summary

Here's a brief summary of all the papers mentioned here:

Article	Methods	Conclusion
Searching for Exotic Particles in High-Energy Physics with Deep Learning [1]	Deep Network with 5 hidden layers & dropout algorithm	AUC: 0.88
BSM-AI Project: SUSY-AI [4]	Random Forest Classifier	Accuracy: 93.8%
Bayesian Neural Networks for Fast SUSY Predictions [11]	Bayesian Neural Networks	F1 Score: 0.957 Recall Score : 0.987
SCYNet: Testing Supersymmetric Models at the LHC with Neural Networks [2]	Neural Networks	Mean error for the both neural network approaches lies in the range of 1.3 – 1.7
Machine learning for Event Selection in High Energy Physics [15]	Stochastic Optimization Techniques	Top Quark Mass: Uncertainty lowered by 29% Higgs Boson Search: Scale factor lowered by 37%
Artificial Intelligence — Applications in High Energy and Nuclear Physics [13]	Wavelet Analysis Support Vector Machines Neural Networks (NN)	NNs perform better than non-AI methods and Support Vector Machines can sometimes perform even better than NNs
Jet Substructure at the Large Hadron Collider [12]	Classification Regression Techniques	Deep learning techniques are essential for development of Jet Substructure, Jet Tagging, and Simulations
Machine and Deep Learning Applications in Particle Physics [3]	Many differrent deep learning techniques	Black Box Models can be bad at identifying causation Promising outlook towards applications of ML and AI in Particle Physics
Deep Learning and Its Application to LHC Physics [9]	Deep Learning Methods	Discussed concerns and highlighted bright prospects of deep learning in Particle Physics and at LHC

3 Project Design

3.1 Project Overview

This project focuses on the signal-to-background binary classification of SUSY particles which stands for supersymmetry - a theory proposing a space-time symmetry where each particle in the standard model has a corresponding partner particle (Figure 3), as mentioned before in the Introduction Section 1.2.

The dataset was found on UCI Machine Learning Repository [14] which is an open data repository. Inspired by the 'The Deep Learning on a Public Dataset Template', I decided to tackle the binary classification problem on the SUSY dataset.

3.2 Dataset

SUSY dataset was produced using Monte Carlo simulation creating 5 million instances of labeled data (1 for signal, 0 for background) with 18 features. According to UCI, the first 8 features are low–level features that are kinematic properties of the signal and the remaining 10 features are functions of the first 8 features to help physicists discriminate between the signal and background [14].

The dataset is stored in Comma Separated Values (CSV) format and has a total size of 2.22 GB.

3.3 Domain and Users

This project aims to serve as a resource for researchers and experts who employ machine learning techniques within the scientific community. The project focuses on the domain of deep learning in Particle Physics, with Particle physicists, High Energy Physics (HEP) researchers, and scientists utilizing machine learning tools in science as the primary target audience. In a broader sense, the project aims to cater to researchers and experts employing machine learning techniques within the scientific community and it is highlighting the machine learning applications outside the industry. Finally, the domain is deep learning in Physics and the users are physicists or the scientific community in general.

The requirements for this project determined by their users are a great degree of accuracy and lower computational resources and higher performance. These two requirements by themselves would make a significant difference in research in particle physics. I would like to attempt to explore the ways to achieve both in this project.

3.4 Work Plan

This Project consists of an 11-step work plan with a defined timeline. You can view the Gantt Chart in Figure 4. Here is the full work plan, take a look below:

1. Background Research:
    - Research to gain a comprehensive understanding of the topic at hand
    - Making the final decision on the choice of the proper project template
    - Choosing a final project idea
    - Duration: April 16, 2023 - May 1, 2023
2. Data Collection and Preparation:
    - Identifying all the possible open data sources
    - Finding the proper dataset
    - Considering the dataset format and data processing steps required
    - Considering research ethics

- Duration: May 2, 2023 - May 15, 2023

3. Project Deliverable and Deadlines
    - Identifying all the project deliverable: Project proposal, Preliminary Report, Feature Prototype, Final Report
    - Project Proposal: Presentation Video
    - Preliminary Report: Literature Review, Project Design, and Feature Prototype
    - Final Deliverable: Final Presentation Video and Final Report
    - Create a work plan & Gantt Chart
    - Duration: May 2, 2023, May 15, 2023
4. Literature Review
    - Extensive research to find similar projects done in the past
    - Summarizing each academic paper to highlight the tools and results
    - Writing out the literature review
    - Duration: May 16, 2023 - June 15, 2023
5. Project Design
    - Considering Domain and Users
    - Identifying the requirements
    - Identifying the proper evaluation metrics
    - Writing the project design
    - Duration: May 16, 2023 - June 16, 2023
6. Model Selection and Architecture
    - Evaluating different deep learning models suitable for the project
    - Researching the latest deep learning models
    - Considering all the tools and frameworks, version control, and finding possible mentors
    - Duration: June 17, 2023 - July 17, 2023
7. Feature Prototype
    - Implementing the most important feature
    - Extensive Research to find the proper tools to create the feature prototype
    - Creating a feature prototype
    - Duration: June 17, 2023 - July 17, 2023
8. Training and Validation
    - Creating the final deliverable
    - Outlining the training strategy, including hyperparameter tuning, optimization algorithms, and regularization techniques
    - Defining the validation procedure to assess the model's performance during training
    - Considering any specific challenges related to the training process
    - Duration: July 18, 2023 - August 18, 2023

9. Evaluation and Result Analysis

    - Finishing the final deliverable
    - Outlining the evaluation method to assess the model's performance
    - Analysing the results in detail, discussing weaknesses and strengths of the models used
    - Deciding on the proper result visual presentations
    - Duration: August 19, 2023 - August 26, 2023

10. Revising, Future Work, and Conclusion

    - Revising the full work done
    - Considering future improvements or extensions to the project
    - Writing a final conclusion and summarizing key results
    - Reflecting on the lessons learned and future work recommendations
    - Duration: August 27, 2023 - September 15, 2023

11. Final Presentation & Report

    - Writing on the final report
    - Creating the final presentation
    - Finalizing the project and submitting
    - Duration: August 27, 2023 - September 15, 2023

3.5 Evaluation

This project can be called successful, if I was able to create a neural networks model that can predict the binary classification of SUSY particles with high accuracy and AUC. This will be one of the evaluation metrics of this project. Another metric, is creating the optimized neural network that would have not be as computationally expensive and would not take hours to compile. In that case, the model's performance would not be worthwhile its computational resources. To summarize, the best model would have short computational compile time and high AUC and accuracy.

The AUC benchmark results presented in dataset's the original paper, 'Searching for Exotic Particles in High-Energy Physics with Deep Learning' by Baldi et al. [1], served as the evaluation metric. AUC refers to the area under the receiver operating characteristic curve, which graphs the true positive rate against the false positive rate. Higher AUC values indicate improved classification accuracy. In their study, the authors employed three primary methods: Bayesian Decision Tree, a 5-layer neural network, and deep networks with a dropout algorithm. They achieved an impressive final AUC of 0.88. I will use both AUC and accuracy as the evaluation metrics since they both provide important insights into the model's performance.

However, considering that this paper was published approximately a decade ago, it remains uncertain whether the reported accuracy was achieved with an optimized performance of the deep learning model. Moreover, the comparison was solely limited to shallow neural networks and Bayesian decision trees. In this project, I want to take advantage of the latest tools available, like Scikit-Learn, TensorFlow, and Keras. By doing this exploration many years later, I hope to achieve even better or similar accuracy results using neural networks and deep learning methods and find the best model and hyperparameters for this classification problem.

Deep Learning in Particle Physics Project Gantt Chart

TASK ID	TASK TITLE	START DATE	DUE DATE	DURATION IN DAYS	PCT OF TASK COMPLETE
1	Background Research				
1.1	Research topic	04/16/23	05/01/23	15	100%
1.2	Final decision on project template	04/16/23	05/01/23	15	100%
1.3	Final project idea	04/16/23	05/01/23	15	100%
2	Data Collection and Preparation				
2.1	Identifying open data sources	05/02/23	05/15/23	13	100%
2.2	Finding the proper dataset	05/02/23	05/15/23	13	100%
2.3	Data processing	05/02/23	05/15/23	13	100%
2.4	Considering research ethics	05/02/23	05/15/23	13	100%
3	Project Deliverables and Deadlines				
3.1	Identifying all the project deliverables	05/16/23	06/16/23	31	100%
3.1	Create a work plan & Gantt Chart	05/16/23	06/16/23	31	100%
4	Literature Review				
4.1	Extensive research	05/16/23	06/16/23	31	100%
4.2	Summarizing each academic paper	05/16/23	06/16/23	31	100%
4.3	Writing out the literature review	05/16/23	06/16/23	31	100%
5	Project Design				
5.1	Considering Domain and Users	05/16/23	06/16/23	31	100%
5.2	Identifying the requirements	05/16/23	06/16/23	31	100%
5.3	Identifying the proper evaluation metrics	05/16/23	06/16/23	31	100%
5.4	Writing the project design	05/16/23	06/16/23	31	100%
6	Model Selection and Architecture				
6.1	Evaluating different deep learning models	06/17/23	07/17/23	30	100%
6.2	Researching the latest deep learning models	06/17/23	07/17/23	30	100%
6.3	Considering all the tools and frameworks	06/17/23	07/17/23	30	100%
7	Feature Prototype				
7.1	Implementing the most important feature	06/17/23	07/17/23	30	100%
7.2	Extensive Research to find the proper tools	06/17/23	07/17/23	30	100%
7.3	Creating a feature prototype	06/17/23	07/17/23	30	100%
8	Training and Validation				
8.1	Creating the final deliverable	07/18/23	08/18/23	31	0%
8.2	Outlining the training strategy	07/18/23	08/18/23	31	0%
8.3	Defining the validation procedure	07/18/23	08/18/23	31	0%
8.4	Considering any specific challenges	07/18/23	08/18/23	31	0%
9	Evaluation and Result Analysis				
9.1	Finishing the final deliverable	08/19/23	08/26/23	7	0%
9.2	Outlining the evaluation method	08/19/23	08/26/23	7	0%
9.3	Analysing the results	08/19/23	08/26/23	7	0%
9.4	Result visual presentations	08/19/23	08/26/23	7	0%
10	Revising, Future Work, and Conclusion				
10.1	Revising the full work done	08/27/23	09/15/23	19	0%
10.2	Considering future improvements	08/27/23	09/15/23	19	0%
10.3	Writing final conclusion	08/27/23	09/15/23	19	0%
10.4	Reflecting on the lessons learned	08/27/23	09/15/23	19	0%
11	Final Presentation & Report				
11.1	Writing on the final report	08/27/23	09/15/23	19	0%
11.2	Creating the final presentation	08/27/23	09/15/23	19	0%
11.3	Finalizing the project & submitting	08/27/23	09/15/23	19	0%
11.4	Project Performance	08/27/23	09/15/23	19	0%

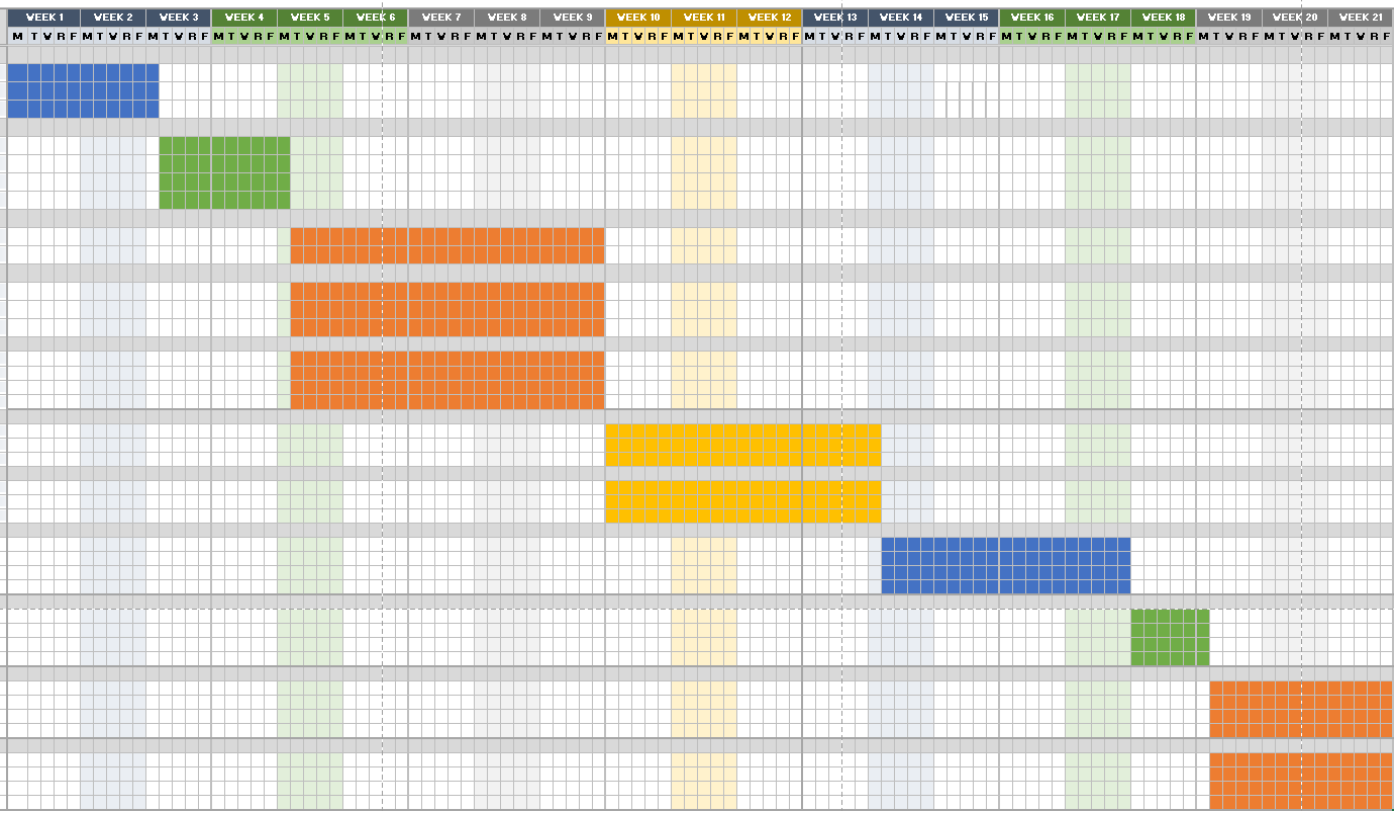

Figure 4: The Gantt Chart

4 Implementation

I created a deep learning model by following the "The universal workflow of machine learning" guidelines from François Chollett's book "Deep Learning with Python"[8]. Additionally, I closely followed the recommendations and requirements provided by the "Deep Learning on a Public Dataset" Template. Here is a step-by-step summary of how I implemented the machine learning model:

4.1 Defining the Problem

As previously mentioned in section 1.2, this project is a signal-to-background binary classification of SUSY particles which stands for supersymmetry - a theory proposing a space-time symmetry where each particle in the standard model has a corresponding partner particle. The dataset was found on UCI Machine Learning Repository which is an open data repository.

4.2 Measure of Success

As previously discussed in section 3.5, the AUC and accuracy serve as the measure of success and evaluation of this machine learning model. That is because the model's performance is compared to the results of the original paper [1] by Baldi et al. inspired by this project. So naturally I optimized the model to achieve higher or similar values of AUC and ACC (accuracy).

4.3 Evaluation Protocol

Using simple hold-out evaluation in this project should work well since we have a large dataset available to evaluate the model. Hold-out evaluation is used to test the performance of the model against unseen data and make small adjustments to hyperparameters and model architecture along the way until we create the optimized model.

This evaluation method divides the training dataset into two splits of training and validation datasets. As you can see in Figure 5, knowing that I have a million dataset, I used the shape of the original dataset to calculate 20% of the 80% of the 5 million rows of data ($20\% \cdot 5000000 \cdot 80\% = 800,000$) to create the validation set. After each training session of the model, we evaluate the model on the validation set. I used an 80/20 split to create the training and validation sets and subsequently created a line plot to monitor the change in accuracy and AUC in each epoch of the training session. To do this, I will use the following code: `model.evaluate(x_val,y_val)` to explicitly evaluate the model on the held-out validation data set as stated in "Deep Learning with Python" [8].

```
# Create validation set - 80/20 split on training dataset

X_val = X_train[:800000]
partial_X_train = X_train[800000:]
y_val = y_train[:800000]
partial_y_train = y_train[800000:]
```

Figure 5: Create Validation and Training Sets

4.4 Preparing Data

In this section, I used the following steps to prepare the dataset for creating the neural network model:

1. Importing the Dataset: Imported all the required libraries in addition to the data from its original CSV format as a Pandas data frame

2. Exploring the Dataset: Explored the shape of the dataset, its statistical summary, its column header names, its data types, and its features and finally selected only low-level features to base the model on. In this step, I also realized there was no need for feature engineering since each data type is already in float format.

3. Cleaning the Dataset: I checked for null and duplicate values. The dataset in its original form was clean and needed no further changes.

4. Visualizing the Dataset: Plotted the features to become further familiar with the dataset and assess which features can be used better for a classification problem.

5. Splitting the Dataset: In this section, I created the training, test, and validation dataset using Scikit-Learn's `train_test_split()` method and also set the classifications labels column (1 for signal and 0 for background) as the target feature values.

4.5 Model Development

In this section, I created the baseline sequential model using Keras with 4 hidden layers with `relu` as their activation function and an output layer with `sigmoid` as its activation function to create a scalar prediction of the model. As you can see in Figure 6, each layer has a capacity of 16 and the input shape of the dataset in a vector shape of 8×1. Later, I set the optimizer to `rmsprop` and the loss function to `binary_crossentropy` and set the evaluation metrics to ACC and AUC.

```
# Baseline Model
model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(8,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop',
              loss = 'binary_crossentropy',
              metrics = ['accuracy',tf.keras.metrics.AUC()])

model.summary()
```

Figure 6: Baseline Model

4.6 Scaling Up: Overfit Model Development

In this section, I follow these steps to overfit the model. Here is a summary of the results:

1. Adding Layers: Increasing the number of hidden layers from 2 to 4, made a large statistical difference in ACC and AUC compared to the baseline model: ACC increased by 0.23% and AUC by 0.28%. So I continued using 4 hidden layers.

2. Increasing Layer Capacity: Increasing layer capacity from 16 to 128 increased the values for both ACC and AUC by 0.59 % and 0.40% respectively.

3. Train for More Epochs: Increasing the number of training epochs from 10 to 20 using 5 hidden layers and a layer capacity of 128, increased the ACC and AUC by 0.56% and 0.41% respectively.

4.7 Regularizing the Model & Tuning Hyperparameters

In this section, after over-fitting the model, I adjust the hyperparameters and the model's architecture to find the optimized version of the model. Here are the steps I followed:

1. Add L1 and/or L2 Regularization: Adding L1 and L2 both decreased the model's performance significantly and adding them separately had a similar effect as well. So there was no need to use them.

2. Add Dropout Layers: Adding a small dropout layer of 0.025 kept the AUC value the same as the last best performance of the model however increased the ACC by 0.0001. Compared to the base model ACC was increased by 0.56% and AUC increased by 0.78%.

3. Try Different Hyperparameters: Increasing the layer capacity from 128 to 512, only improved the AUC of the model by 0.01% compare to the last best performance of the model and it took 46 minutes for the model to compile. This deemed very inefficient for the performance it would provide. Decreasing the layer capacity made such an insignificant difference that I decided to stick to 128 as the model capacity.

4. Trying Different Architectures: Removing one layer gave an identical result to the best training result of ACC of 87.41% and ACC of 80.14 %. Increasing the numbers of layer by takes lower computational resources while outputing exact same performance, I decided to go back to a 3-layer architecture for the final model.

4.8 Final Model

The final model consists of 3 hidden layers with 128 as their capacity and the model trained for 20 epochs as you can see in figure 7. I finally trained this model one last time and used it to predict the test set and achieve the best performance yet: AUC = 87.46% and ACC = 80.16%. There was a 0.0005 and 0.0002 increase in AUC and ACC respectively. These results concluded there was no need to change the validation method or change the model any further.

```
# Final Model
model = models.Sequential()
model.add(layers.Dense(128, activation='relu', input_shape=(8,)))
model.add(layers.Dropout(0.025))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dropout(0.025))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dropout(0.025))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop',
              loss = 'binary_crossentropy',
              metrics = ['accuracy',tf.keras.metrics.AUC()])

model.summary()

history = model.fit(partial_X_train, partial_y_train, epochs = 20, batch_size = 512, validation_data= (X_val, y_val))
results = model.evaluate(X_test,y_test)
```

Figure 7: Final Deep Learning Model

4.9 Code

You can access the code alongside the data visualizations, graphs, and plots on https://github.com/Razgaleh/SUSY-Classification. Git was used extensively for version control in this project. The deep network model was computed on a GTX 1080 GPU and a Ryzen 5 5600X CPU.

5 Evaluation

There were multiple steps taken to evaluate the deep learning model throughout its construction. Firstly, I defined the evaluation metrics thoroughly, as briefly mentioned in section 3.5 and 4.3. Here I will go into more detail.

5.1 ACC and AUC

ACC or accuracy is the ratio of correctly predicted instances against the total predictions or the total number of instances in the dataset. It is important to note accuracy is not always a good metric, especially if the dataset is imbalanced and there are more values of one class compared to the other. Other metrics such as AUC, Recall, Precision, and F1-Score can be better metrics depending on the classification problem.

In our case, AUC can be a better metric compared to ACC. AUC is the Area Under the Area Under the Receiver Operating Characteristic (ROC) Curve. ROC plots the true positive rate against the false positive rate. The area under this curve or AUC allows us to investigate the model's performance to classify positive instances higher than negative instances. AUC is widely used in binary classification problems.

As mentioned before in sections 3.5 and 4.3, I chose to monitor both these values for the deep learning model. However, I decided to prioritize AUC over ACC since it is a better metric for imbalanced datasets and there is no information on how balanced this dataset is and assumed that it is not balanced. In addition, the original paper [1] that inspired this project, used AUC as their evaluation metric, and to compare the results fairly it is best to use the same metrics.

5.2 Testing and Plots

To keep track of the fluctuations in ACC, AUC, and loss function, I tested the model on validation set and created plots at almost every step of the construction of the deep learning model. The baseline model serves as the comparison metric to evaluate the deep learning model. In figures 8, 9, and 10, you can observe the change in AUC, ACC, and loss values over epochs on both training and validation datasets.

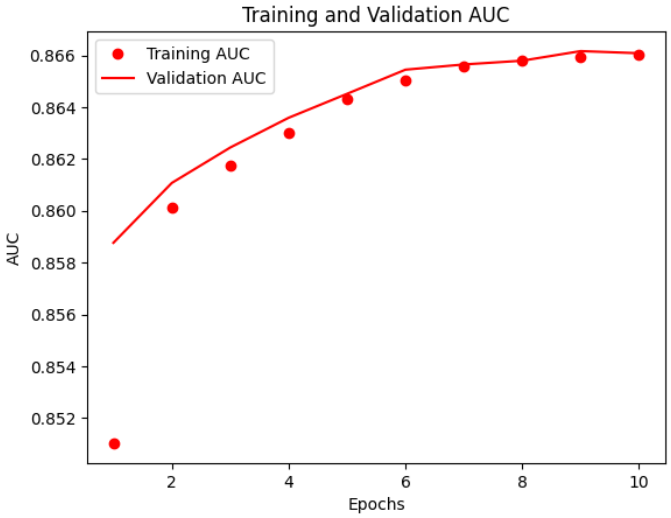

Figure 8: Baseline Model's AUC in Training and Validation Set

In the baseline model, both AUC and ACC increase as expected and the loss decreases also as expected. The model achieves an average AUC of 0.8680 and ACC of 0.7951. These values are used to calculate the percentage increase or decrease in the model's performance. You can compare the final model's plots to these baseline plots in the figures 11, 12, 13.

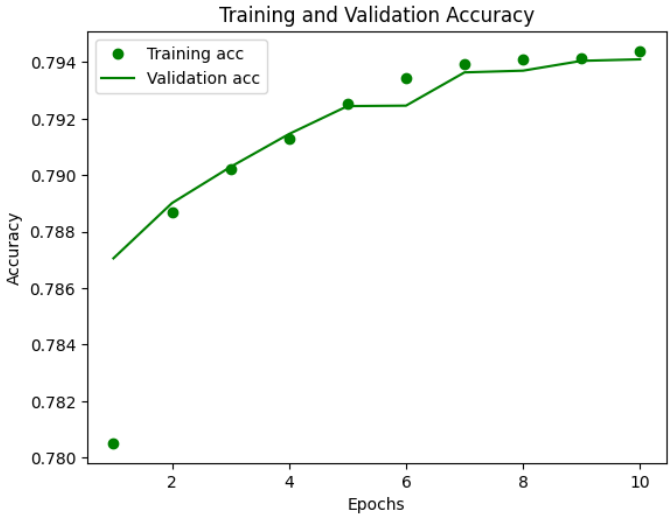

Figure 9: Baseline Model's ACC in Training and Validation Set

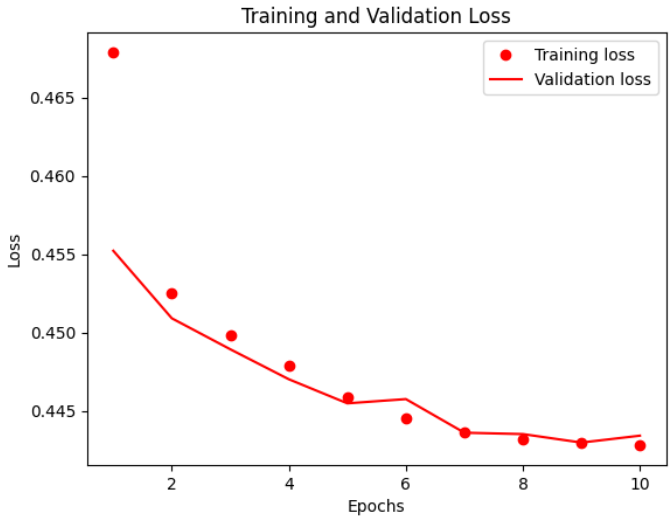

Figure 10: Baseline Model's Loss in Training and Validation Set

The final model's AUC was 87.46% and ACC of 80.16%. As you can see in figure 11, the AUC is consistently increasing at each epoch and converging to a value of 0.874 while from observing the plot 8 we can see it only converges to 0.866. This displays the improvement in model's performance. We can observe similar effect in ACC plots in figures 9 and 12. Finally, the loss function values are decreasing to a lower convergence point in the final model than the baseline model as expected (figures 13 and 10).

5.3 Successes and Limitations

This project has had many successes, as well as failures or limitations. Although the final model's AUC value is 87.46% and is different by 0.61%, I successful to achieve similar AUC value of 88% as the original paper[1]. However depending on the use case of the application of this model, 0.61% discrepancy could be of significance and would make this model not useful, precise enough, and limited. Furthermore, even though I prioritized the AUC performance over ACC, the model was still able to achieve a high ACC of 80.16%. Another limitation of this model would be the choice of AUC as an evaluation metric over other metrics such as recall and precision. AUC can ignore the cost of misclassification which could be costly in real world applications for the users of this model.

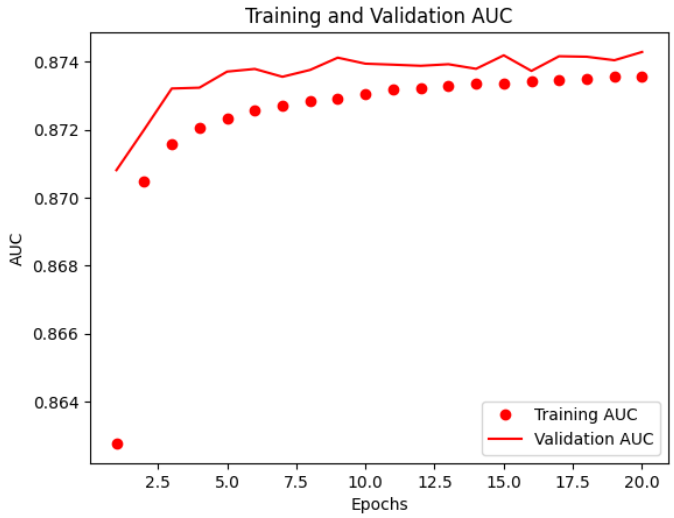

Figure 11: Final Model's AUC in Training and Validation Set

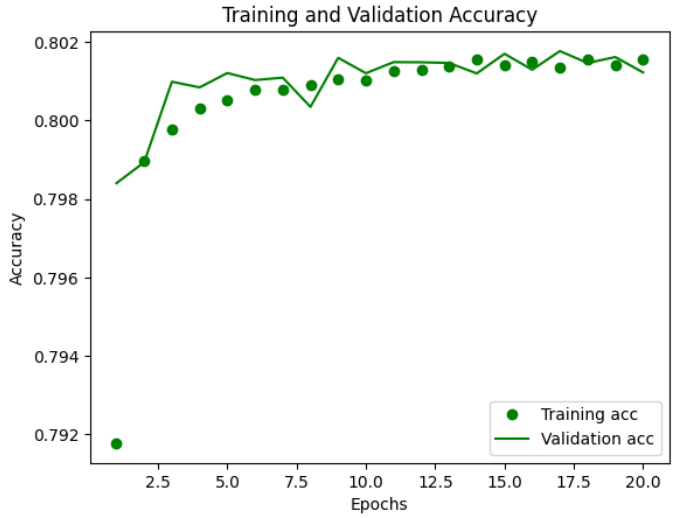

Figure 12: Final Model's ACC in Training and Validation Set

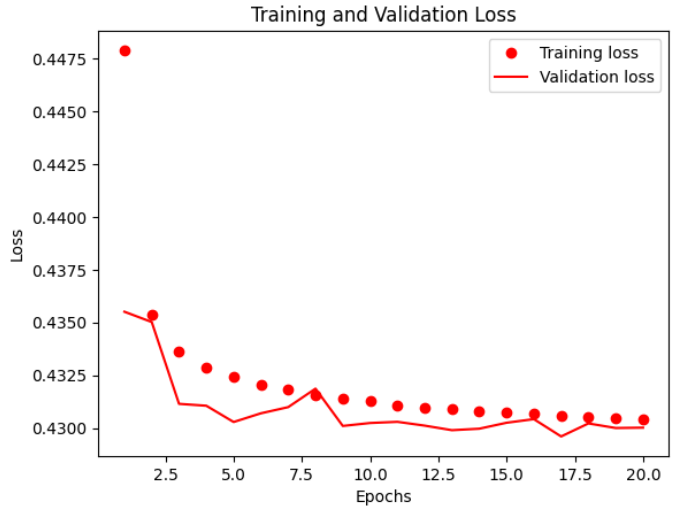

Figure 13: Final Model's Loss in Training and Validation Set

6 Conclusion

6.1 Summary

To begin this project, I did extensive research and literature review to understand and familiarize myself to the topic at hand. I studied multiple research paper on different deep learning and machine learning approaches and methods used in field of Physics. Later I summarized these findings and chose "Searching for Exotic Particles in High-Energy Physics with Deep Learning" by Baldi et al. [1], as a reference and inspiration for this project. This paper used deep learning methods on SUSY particle dataset with 8 low-level features found on UCI [14].

After finding the proper dataset, I began designing the project and created a work plan using Gantt Chart. In this step, I familiarized myself more with the dataset and also contemplated on the project's users and domains and set different evaluation metrics. Creating a work plan was essential to the success of this project.

To create the model I followed the guidelines from "The universal workflow of machine learning" by Francois Chollett. First, I defined the problem as binary classification later I set the measure of success to be AUC and ACC. Next, I set hold-out evaluation method as the evaluation protocol. After preparing and exploring the dataset, I split it into a validation, training, and test set. Next, I developed a baseline model of 2 hidden layer with layer capacity of 16 and one additional output layer. I used the optimizer to `rmsprop` and the loss function to `binary_crossentropy`.

Soon after, I scaled up the model to find overfit the model. I did this step by adding layers, increasing layer capacity and training for more epochs. I finally settled for a model using 4 hidden `relu` layers with capacity of 128 and training the model over 20 epochs.

Next step was regularizing the model and tuning the hyperparameters. To complete this step, I investigated the effects of adding L1 and/or L2 regularization parameters, adding drop out layers, increasing or decreasing the layer capacities again, and adding or removing a hidden layer. I concluded adding L1 or L2 regularization values make no significant difference at all. Adding a small dropout layer of 0.025 increased the performance of the model. Increasing the layer capacity only increased the computational time and decreasing the layer capacity made a very insignificant difference. Next, I explored adding a layer, which made the performance worst but unexpectedly removing a layer results in similar output.

Finally, I successfully created a deep learning model for the binary classification of the SUSY particles. The model performance is an ACC of 80.16% and AUC of 87.46%.

6.2 Further Developments

Given the chance to further develop this project, I would consider doing the following:

1. Try different validation technique
2. Explore different Keras optimizer function and loss functions
3. Using an entirely different Python machine learning library such as PyTorch and comparing the results
4. Integration of Cloud Computing Resources: To speed up the compile time for higher layer capacity models (≥ 512)

6.3 Thank you

Thank you for being a part of this milestone in my academic journey. This project has been a tremendous learning experience, and I am grateful for the opportunity to work on it. I would like to thank my family, professors, and peers at University of London for their unwavering support.

7 References

[1] Baldi, P., Sadowski, P., and Whiteson, D. Searching for exotic particles in high-energy physics with deep learning. *Nature Communications 5*, 1 (2014).

[2] Bechtle, P., Belkner, S., Dercks, D., Hamer, M., Keller, T., Krämer, M., Sarrazin, B., Schütte-Engel, J., and Tattersall, J. SCYNet: testing supersymmetric models at the LHC with neural networks. *The European Physical Journal C 77*, 10 (oct 2017).

[3] Bourilkov, D. Machine and deep learning applications in particle physics. *International Journal of Modern Physics A 34*, 35 (dec 2019), 1930019.

[4] Caron, S., Kim, J. S., Rolbiecki, K., de Austri, R. R., and Stienen, B. The BSM-AI project: SUSY-AI–generalizing LHC limits on supersymmetry with machine learning. *The European Physical Journal C 77*, 4 (April 2017).

[5] CERN. The Early Universe — CERN, Conseil Européen pour la Recherche Nucléaire. https://home.cern/science/physics/early-universe. [Online; accessed 13-April-2023].

[6] CERN. The Standard Model — CERN, Conseil Européen pour la Recherche Nucléaire. https://home.cern/science/physics/standard-model. [Online; accessed 15-June-2023].

[7] CERN. TMVA — CERN, Conseil Européen pour la Recherche Nucléaire. https://root.cern/manual/tmva/. [Online; accessed 1-June-2023].

[8] Chollet, F. *Deep Learning with Python*. Manning, Nov. 2017.

[9] Guest, D., Cranmer, K., and Whiteson, D. Deep learning and its application to lhc physics. *Annual Review of Nuclear and Particle Science 68*, 1 (2018), 161–181.

[10] IEEE. Analyzing the LHC Magnet Quenches — IEEE Spectrum, for the technology insider. https://spectrum.ieee.org/analyzing-the-lhc-magnet-quenches, 2017. [Online; accessed 14-July-2023].

[11] Kronheim, B., Kuchera, M., Prosper, H., and Karbo, A. Bayesian neural networks for fast SUSY predictions. *Physics Letters B 813* (February 2021), 136041.

[12] Larkoski, A. J., Moult, I., and Nachman, B. Jet substructure at the large hadron collider: A review of recent advances in theory and machine learning. *Physics Reports 841* (jan 2020), 1–63.

[13] Müller, U. Artificial intelligence—applications in high energy and nuclear physics. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 502*, 2 (2003), 811–814. Proceedings of the VIII International Workshop on Advanced Computing and Analysis Techniques in Physics Research.

[14] UCI. SUSY — UCI, University of California Irvine Machine Learning Repository. https://archive.ics.uci.edu/dataset/279/susy, 2014. [Online; accessed 15-April-2023].

[15] Whiteson, S., and Whiteson, D. Machine learning for event selection in high energy physics. *Engineering Applications of Artificial Intelligence 22*, 8 (2009), 1203–1217.

[16] Wikipedia. Large Hadron Collider (LHC) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Large_Hadron_Collider. [Online; accessed 13-May-2023].

[17] Wikipedia. Monte Carlo Method — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Monte_Carlo_method. [Online; accessed 25-May-2023].

[18] Wikipedia. The Big Bang — Hubble Site, NASA. https://hubblesite.org/contents/articles/the-big-bang, 2022. [Online; accessed 13-April-2023].