UNIVERSITY OF CALIFORNIA

SANTA CRUZ

EVALUATING BINARY CLASSIFICATION NEURAL NETWORKS TO DETERMINE SENSITIVITY TO SLEPTON PRODUCTION AT THE LHC

A thesis submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

PHYSICS

by

Zachary E Dethloff

December 2023

The Thesis of Zachary Dethloff
is approved:

Associate Professor Michael Hance, Chair

Professor Jason Nielsen

Professor Stefano Profumo

Peter Biehl
Vice Provost and Dean of Graduate Studies

Contents

List of Figures

List of Tables

Abstract

Zachary Dethloff - EVALUATING BINARY CLASSIFICATION NEURAL NETWORKS TO DETERMINE SENSITIVITY TO SLEPTON PRODUCTION AT THE LHC

Four different Binary Classification Neural Networks are used to assess the sensitivity of experiments at the CERN Large Hadron Collider related to Supersymmetry. Several methods are used to study the effectiveness of each neural network's ability to separate signal from background by evaluating their performance during and after the training phase. Sensitivities over the slepton's parameter space are graphed using each of the four neural networks. The four neural networks are then evaluated individually and comparatively in order to analyze each of the neural network's respective performance and general trends in sensitivity to the slepton's parameter space.

1 Introduction

Classification Neural Networks

Machine Learning (ML) is a swiftly developing tool in many areas of research. The increased use of computational analysis in nearly every field of industry, the advancement of computer processing components like Graphics Processing Units (GPUs), as well as the collection of extremely large data-sets has opened up new possibilities in understanding the correlation and causation of phenomena [6].

Throughout recent years, nuanced methods of applying neural networks and deep learning algorithms to abstract problems normally performed by humans have become more common as Python libraries like Tensorflow [23] have simplified the process immensely. Libraries like Keras [3] and Sci-kit Learn [9] take the task of designing and implementing statistical algorithms, that make up the backbone of machine learning in Python and condense them into simple commands. This minor revolution in the capabilities of computational intelligence has opened a range of new opportunities in the scientific world as a way to re-approach old or current questions to find answers thought previously impossible to obtain or model.

Binary classification in machine learning is a specific type of neural network that uses a logistic regression algorithm to learn features of signal and background data [2]. Once the algorithm's parameters have been trained to recognize features of the signal and background data, the neural network can be used to separate a given event. Logistic regression aims to produce a prediction of how signal-like incoming data is using the logistic function [2,3,8]. The logistic function, or sigmoid, will return a value between 0 (background) and 1

(signal) based on the binary probability equation given in Eq. (1) [8].

$$p(y = 1|x) = \frac{1}{1 + \exp(-w^T x - b)} \tag{1}$$

Given a data-set, each row can be referred to as x in Eq. (1), which is an n-dimensional vector whose length n is determined by the different column values or features in the data-set [8]. The variable y is then the binary output variable which defines the signal label for a signal event [2,8]. The value calculated in Eq. (1) is the probability that the given event x is a signal event similar to y, giving it a score close to 1 [8]. Eq. (1) is the equation of a sigmoid function that has been augmented by b, the bias, and w, the weight vector, both of which are parameters learned by neural networks in their training and testing phases [2,8]. Using conditional maximum likelihood estimation, values for parameters w and b are chosen that maximize the log probability for the true value of y in the validation set for a given event [26]. Using the equation for cross-entropy loss [26] (Eq. 4), a binary classification function that describes how far off a given prediction of an event is from its true y-value, these same weights and biases can be used to return large values when the neural network is confused, and small when the neural network is close to correct. These weights and biases can be further optimized by minimizing the cross-entropy loss over a sum of all respective events, weights, and biases, which can be solved by gradient descent. A neural network results in an array of sigmoid functions are created with their own weights and biases that are all used to complete a specialized task, and this array of functions is called a Multi-Layer Perceptron (MLP) [2,6]. This can also be referred to as a feed-forward network due to the sequential nature of data processing [2]. The neural network is designed through the use of layers, which represent collections of neurons that each receive input data and apply the respective weight and bias terms to that data. This process continues until

the neural network reaches a final layer, which only contains one neuron for binary classification tasks, which will classify or give a final score to the input data [2]. Each of the layer's weights and biases are developed over the training and testing phase [2,6]. This general approach is then distilled into a specific task which dictates much of the architecture for the MLP [2]. The focus of this project will be on the task of classification, and while designating a task does narrow the scope of a ML project, tasks themselves are still very general and must be specialized further.

An MLP can seemingly be used for anything. From object recognition in images [2,9], language processing, and signal vs. background filtering, as long as the MLP is properly trained these tasks can be completed effectively and efficiently. Even the complicated task of scoring data-sets over 100,000 entries long can be accomplished with a relatively shallow neural network [2,8]. But how do we know if an MLP is working reliably? Each model can be evaluated based on its loss, precision, and recall, metrics developed throughout the training and testing phase that will be discussed in detail later in this paper [3,9]. Based on these values, neural networks may have their architecture changed to better reflect a desired precision through a process known as hyper-parameter tuning [2,6].

The State of SUSY

The world of particle physics is well described by the Standard Model (SM) which breaks up elementary particles into groupings of quarks, leptons, and bosons (force carriers) [1]. While this elegant framework has proven itself by predicting and describing everything we see at particle accelerators like the LHC, it falls short of describing some of the mysteries that we think lie in higher energy physics that are hinted at in the details of symmetry breaking [1,10].

Phenomena like dark matter, neutrino masses, hierarchy of fermion masses, and matter-antimatter asymmetry are so far unexplained by the SM [10]. The discovery of the Higgs Boson proved the accuracy of the SM when symmetry requirements and Gauge theories predicted the W and Z bosons to have zero mass [7,10]. Yet since it falls short of describing the previously mentioned array of phenomena, there must be some extension made to the SM [7,10].

Using the SM to predict new physics indirectly is not a simple process, as the Higgs mass correction is extremely sensitive to new mass scales, as it is both quadratic and divergent, according to Eq. (2), where M is the new physics scale above which the SM is no longer effective [7].

$$\delta m_H^2 = -(2m_W^2 + m_Z^2 + m_H^2 - 4m_t^2)\frac{3M^2}{16\pi^2 v^2} \tag{2}$$

Determining this value M is a tricky task; as of run 2 of the LHC at an energy of 13 Tera-electron Volts (TeV) no sign of new physics has been observed, which already necessitates M be a large value [7,10]. One of the proposed solutions to this problem is super-symmetry (SUSY), an extension to the SM in which every particle currently on the SM has an assigned superpartner [7,10]. This would allow for new terms of the same order of the term M which cancel with the divergent value and keep the Higgs mass from being a divergent value. This also provides a cancellation for the re-normalization scale of new physics making SUSY a UV complete framework [7].

This symmetry between fermions and bosons is the core of SUSY, which states that not only are fermions and bosons related, but each fermion has a bosonic superpartner and vice versa [7]. The challenge in finding these super-partners is that they come from standard interactions of quarks with extremely small cross-sections at current energy levels reached by the LHC. For the slep-ton, the pair production cross-section given by [19] for a 13 TeV collider with

a left and right-handed slepton with masses of about 110 GeV is estimated to be 0.55 pico-barns. The fundamental SUSY partons, squarks and gluinos, are very unstable and instantly decay, but according to [10] have an 100% branching ratio to long lived, weakly interacting Lightest Supersymmetric Partners, or LSPs [10]. According to [7], the cross-section for SUSY products increases as the energy of the collisions increase, encouraging colliders to push to higher energies in order to be in a regime where creation of SUSY particles is more abundant. Even then, the mass parameters of the parent SUSY particle and LSP are not fixed making the observations even more challenging [10]. What is known is that the superpartner would need to be a product of the majority of this energy to be created due to its cross-section scaling with increasing energy, which is unlikely at best since the background process will be leading order [7]. Since detectors only detect the initial interaction and its final state products, scientists must use other variables to infer the existence of the parent superpartners interwoven in SM processes [7]. Not only this, but if LSPs are the most accessible of the superpartners given our current energy restrictions, then their weakly interacting nature would make even this final state's detection extremely difficult. Due to this, it is paramount that experimentalists use what information is detected to the fullest, and analyze the kinematics of all observed final states.

One of the methods used to help detect the presence of a SUSY process is by the use of Missing Transverse Energy (MET), or the imbalance of momenta between the colliding protons and detected particles [7]. Since the LSP weakly interacts with the detector, there will be missing energy in a decay channel including sleptons, or any superpartner. If the MET of a given collision matches the expected MET from a undetected neutralino then its likely a slepton was created during the decay. Analysing observables like MET aids in the efforts to

comb through particle data to probe the existence of superpartners in accelerator processes.

By hypothesizing the kinematics of SUSY particles through simulated collider events, and analyzing the kinematics of any detected final state information, physicists can hone in on specific detector conditions that heavily suggest or are only possible if a SUSY particle existed somewhere in the hundreds of processes [7,10]. Not knowing the exact parameter space of SUSY makes this challenging, but different classes of SUSY models can be generated to help with this using programs like MadGraph [18], especially when we are interested in a specific subset of SUSY models. Observables that relate to particle kinematics or combinations of particle kinematics can be used to develop understandings of SUSY by translating the parameter space to collider observables [7].

The Project Focus

The goal of this project is to develop neural networks that can discriminate between SUSY and large SM backgrounds using simulated ATLAS experiment data. This Thesis will be divided up into three main sections.

The first section will cover the data being investigated, what programs are used to create it, as well as why a neural network is so well suited to handle such data. Large amounts of well-curated data are required to build a reliable MLP that can classify signal events.

The second chapter will focus on the architecture of the neural network, the hyper-parameters that were chosen, and how the data was used from training and testing all the way through to the final outputs.

The third section will cover the results given by the MLP, the performance of the MLP, and what the results mean for our ability to test SUSY parameter space using LHC data. Performance values which are stored with each training

and test run, can be used in combinations or viewed separately to discern a number of statistics about the validity of the final results. These metrics are essential in sequential runs focused on optimizing an MLP for a given task.

The main takeaway from this paper will be the performance diagnostics and sensitivity contours that result from each MLP. Determining the most efficient slepton data to train a neural network on that generates reliable predictions of theoretical parameter spaces and defining methods to evaluate these neural networks will make the hunt for SUSY in colliders more efficient.

2 The Data

Signal and Background Generation

Neural networks need large numbers of events to create a reliable logistic function [2]. A combination of programs: MadGraph5-aMC@NLO [17] and Pythia [12] are used to simulate collisions using SUSY models. MadGraph5 is a high energy particle physics process generation toolkit that only requires the user to specify initial and final state particles, and the model to be used to calculate the rate at which the scattering process occurs [24,18]. The specifics of how MadGraph5 works is described in detail in the paper cited as [24]. According to [12] Pythia must also calculate and store values concerning the entire process for the virtual particle showering which cannot just be found by analysing the final state particles. These two programs mainly generate the important features like energy and momentum that act as key discriminators in developing a binary classifier [12].

The signal data generated in this project is generated using the channels shown in Fig. 1 (left), and is categorized by three key quantities for a given sample: slepton mass, neutralino mass, and the mass splitting, the difference

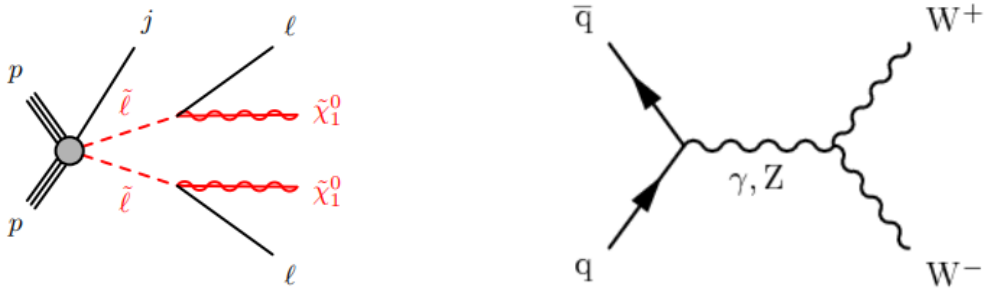

Figure 1: These figures show the decay channels of a pp collision pair producing sleptons (left), and W-bosons (right). Both of these have the same final state signature.

between slepton and neutralino masses. Slepton masses range from 100 to 300 GeV, neutralino masses range from 25 to 290 GeV, and mass splittings range from 10 GeV to 75 GeV [19]. Mass splittings are important in addressing the different possible LSP mass configurations. Depending on the mass splittings that define the data passed into a neural network during training and testing stages, the resulting MLP will be sensitive to that region. This is due to kinematics of the processes changing in response to different masses of the LSPs in the final states. Background data is generated using the channel shown in Fig. 1b (right), and is a leptonically decaying diboson. This can either be a WW-boson or WZ-boson pair, and these events are confined to the same kinematic regions as the sleptons are.

Training sets can be comprised of one SUSY model with many events making the resulting MLP sensitive to the kinematics of sleptons with similar mass and mass splittings. Training sets can also be comprised of many slepton models with fewer events, all with varying mass splitting and slepton masses making the MLP less sensitive but having a wider range. Finally some combination of these two methods could be employed. Utilizing methods such as mixing events from two large sample files is usually done to make the MLP sensitive to

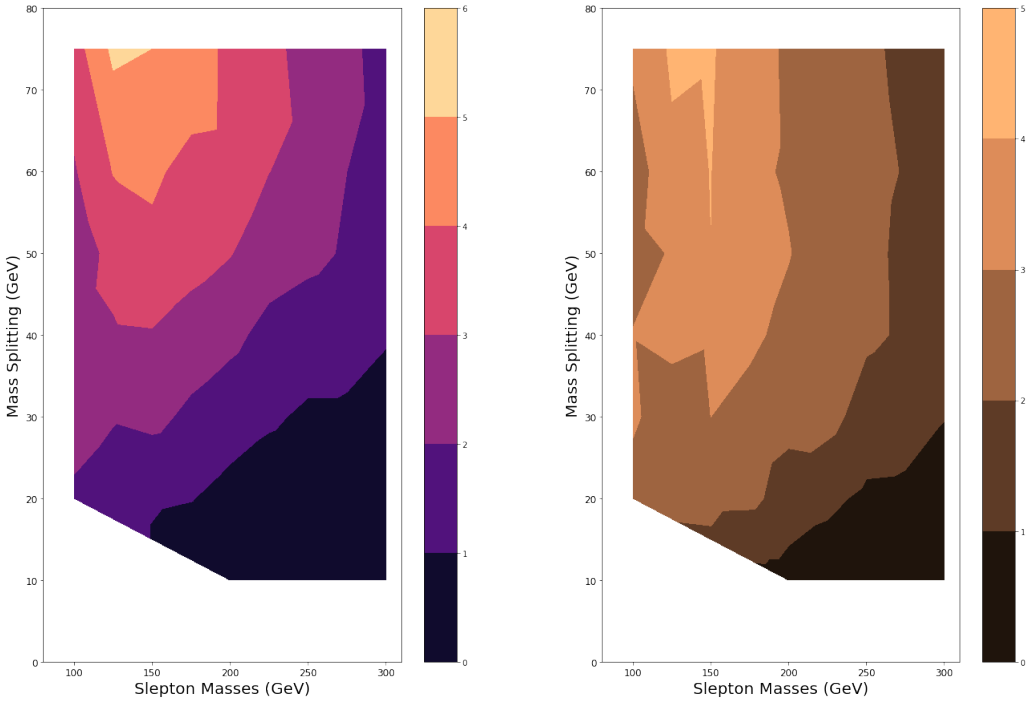

Figure 2: These two images display the confidence intervals in the mass spectrum of an MLP trained on a single sample file (left) vs. random events taken from multiple sample files (right). Sample files dictate the slepton mass and mass splitting of a given set of simulated slepton events. Slepton masses and mass splittings are measured in GeV, and the colors on the color bar indicate the confidence level.

a range of SUSY models and should only be employed to attain better statistics at different regions of the SUSY parameter space compared to the narrow focus of a single large sample file of similar quantities.

Fig. 2 shows an example of the sensitivity contours of a completed run of a MLP trained on one large sample file versus a MLP trained on many smaller sample files. While training on one sample file yields a higher maximum confidence level, the average confidence interval shown by the standard deviations is smaller than that of the multi-model. Using different combinations of slepton mass, mass splitting, and neutralino mass gives insight into any trends that

binary-classification MLPs have. Looking for similarities in these contours not only implies MLP trends, but also regions of consistent high confidence intervals as a means to better constrain the SUSY parameter space.

Pre-Processing

After the signal and background files are successfully created, the next step is to apply a pre-selection to the data outlined by the study in [19]. Pre-selection allows us to pick out specific kinematic variables that the neural network will learn to distinguish between background and signal. These variables are chosen not only to constrain our data to SUSY sensitive regions, but to act as possible discriminators of background events.

Kinematic Variables	MET	Lepton 1,2 Flavor	Lepton 1,2 Charge
Pre-Selection Cuts	> 200 GeV	Same	Opposite
Kinematic Variables	MT Lepton 1,2	Min Azimuthal Jet Seperation	Jet 1 Azimuthal Seperation MET
Pre-Selection Cuts	MT > 10 GeV	DPhi > 0.4 radians	DPhi > 2 radians
Kinematic Variables	Lepton 1,2 Psuedo-Tapidity	Lepton 1,2 MET Azimuthal Seperation	Lepton 1,2 MET
Pre-Selection Cuts	NA	NA	NA
Kinematic Variables	# B-Tagged JET PT > 20 GeV	Lepton 1,2 PT	Dilepton Distance Parameter
Pre-Selection Cuts	0	> 10 GeV	Rll > 0.75
Kinematic Variables	# Jet PT > 30 GeV	MET Phi	Lepton 1,2 Phi
Pre-Selection Cuts	# < 3	NA	NA
Kinematic Variables	Dilepton Invariant Mass	MET/J1PT	Number of Leptons
Pre-Selection Cuts	NA	NA	2

Table 0: Shows all of the kinematic variables that are used for pre-selection and passed into the neural network. This table does not include the 6 different weights associated with each event.

There are 26 kinematic variables chosen to represent each slepton event when passed into the neural network. Some of these variables are also used for pre-selection cuts to ensure the proper slepton model's kinematics are respected. Lepton counts, charge, flavor, and number of b-tagged jets are used solely by

the pre-selection cuts to ensure that the events given to the neural network are consistent with the decay channels shown in Fig 1. While these variables all have the same restrictions for all events passing the pre-selection requirements, they can still be kept as training variables as the neural network quickly learns that they are not discriminators between signal and background. Individual final state lepton information includes missing transverse mass (MT), momenta (PT), azimuthal angle (ϕ) (angles around the beam axis in the x,y plane, where x-axis points towards the center of the LHC ring, and the y-axis points directly upwards [25]), and pseudo-rapidity (η) are important in both the pre-selection and discrimination power as shown in Table 0. Overall process features can also be used to teach the neural network how to classify as well as constrain the events like individual jet momenta, lepton separation (R_{ll}), dilepton invariant mass (m_{ll}) and total MET, since they are direct results of the slepton mass and mass splitting. Other observables have been created as combinations of these kinematics but have proven to be significantly less useful in developing an effective MLP, but do provide some use in pre-selection, which can be seen in Table 0.

Each event also has an associated weight used to compute a realistic version of the Slepton's yield in a process at a collider. These weights concern corrections for particle pile up and tagging efficiency as well as statistical weights from the generator. The proper weight can be found using Eq. (3), where L is the luminosity for a 13 TeV process which is L = 139 fb^{-1}.

$$P_w = W_{event} * W_{gen} * W_{pu} * W_l * W_{bT} * W_{jvt} * L \tag{3}$$

The working data with the pre-selection applied there exist close to 200,000 events between background and signal to be used by the neural network for training and testing, and up to 300,000 for certain signal files with larger event

counts.

All of the selected data must be reformed into specified training, testing, and validation dataframes [3,9]. This can be done simply using the Scikit-Learn [9] library which randomizes what events go in either set while keeping track of signal and background for validation. For this project the split was 80 percent to training and 20 percent to testing. The final step that must be applied to the data before the MLP can begin construction is the standardization of the data. Using the Scikit-Learn [9] Python library the data to be passed into training is centered and scaled from -1 to 1, and the same scaling is applied to the testing data. This makes it easier for the MLP to be built on top of this data as it means the neural network is essentially learning a standard distribution. This scaling is determined first by the larger training data and the same fit is then applied to the testing data [9].

3 Building a Multi-Layer Perceptron

While it is possible to develop a neural network from scratch in vanilla Python, the use of libraries like Keras [3] and Scikit-Learn [9] not only make the work easier but also promote vastly larger coding efficiency. Libraries like Numpy [20], Pandas [22], and Matplotlib [21] are a staple of any data science project in Python, but machine learning libraries allow for much more readable and efficient code. The prior libraries will be used extensively throughout this project, but they aim to define and control the neural network while the latter play the role of manipulating data for analysis.

Evolving Algorithms

The Keras library [3] has three different ways to implement models to design a neural network's structure. The sequential model is what has been used for this project, which is a shallow, feed forward model that takes a number of inputs and gives one output [3], also shown in Fig. 3.

In a sequential model the motion of information through the neural network is acyclic, not allowing for any back-propagation or cross talk between neurons of the same layer [2]. Despite this, it is still more than enough for the task at hand. MLPs also need an optimizer with an associated learning rate. The optimizer class in this project is known as Adam [4], a stochastic gradient decent procedure which is essential for updating network weights. The stochastic gradient decent procedure deals with the minimization of a stochastic scalar function, the binary cross-entropy, through the use of gradient descent applied to the function [4]. The loss function, shown in Eq. (4), must be minimized and the Bernoulli Distribution, Eq. (5), must be maximized in order to determine the proper weights for each neuron shown as Eq. (1), and Adam handles that [4].

$$L_{CE} = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})] \tag{4}$$

$$p(y|x) = \hat{y}^y (1 - \hat{y})^{1-y} \tag{5}$$

The loss function used in Eq. (4) is known as Binary Cross Entropy, and determines how far the predicted value, something between 0 and 1, is from the real value, either 0 or 1. The Bernoulli Distribution is simplified to either $\hat{y}$ for y = 1, or $1 - \hat{y}$ for y = 0, which allows for minimizing the Binary Cross Entropy to determine the best possible weights and bias. Eq. (5) is specific to binary classification tasks, and predicts the likelihood that the input vector represents

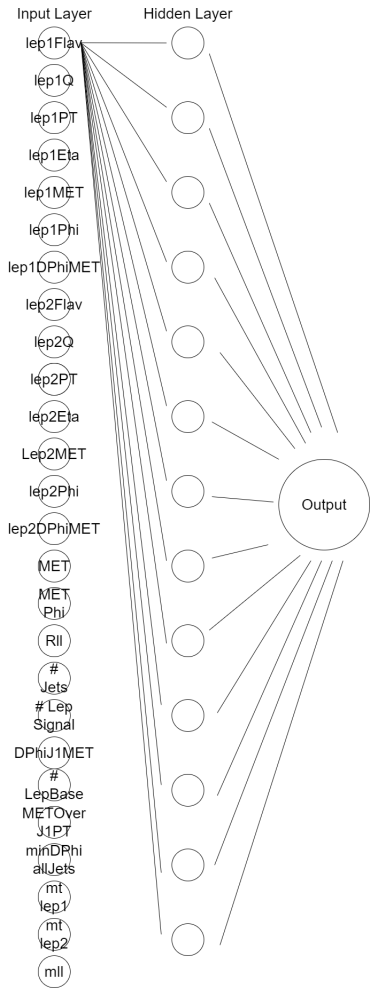

Figure 3: This figure shows the general outline of the sequential neural network architecture used in this project. Each of the nodes in the input layer is connected to each of the nodes in the hidden layer. Gaps in the hidden layer represent dropped out neurons.

a signal event [26]. Adam does this through applying gradient descent to Eq. (4) at each of the subfunctions as opposed to the entire function, evaluating their first derivatives on their own step sizes. This prioritizes efficiency and speed while still retaining a high degree of accuracy in optimizing the weights and biases. The balance of maximizing this likelihood and minimizing the loss by changing the weights and bias terms present in Eq. (1) is what defines machine learning.

With a model chosen, the next step is to define the layers and neurons within them. In the input layer the neuron count is the same as the number of features being passed into the neural network. For this project, the input layer is 26 neurons wide. Only one hidden layer was needed to perform a meaningful analysis, and anything over one leads to poorer predictive performance. Any neural network that has architecture too large for the task at hand will run into decreases in performance. Due to the relatively simple nature of the task, a shallow network is best [8]. The following hidden layer has the same amount of neurons as the input layer. The difference between the input and hidden layers is that the hidden layer will make use of the dropout function, a way to simultaneously turn off a given number of the neurons in the layer corresponding to the dropout rate. The dropout rate for this project is set to 50%. Dropout significantly helps the neural network's ability to generalize predictions allowing for even the most specialized MLP a better ability to predict events in different regions of the SUSY parameter space [8]. The layer after this is the output layer with one node and outputs the signal probability for a given event.

Each neuron needs a method to introduce non-linearity into the model which comes in the form of activation functions [2,6]. Each neuron in the input and hidden layers use the rectified linear-unit (ReLU) activation function, which maps negative values to 0 and keeps positive values linear [2]. This is essential

in developing the MLP's ability to distinguish between signal and background, as well as scoring events labelled as signal. The final output layer then maps all inputs from the neurons into a sigmoid activation function which is necessary for the classification ability of the MLP [2]. The sigmoid function will output a score indicating how signal-like an event is, given its features [8], which can be compared to the validation set in the training to improve the MLP, and allows for the calculation of significance scores in the final analysis [2]. Without these activation functions, there would be no true machine learning taking place so they are essential in defining how the MLP will function.

Back-Testing

In order to understand the results and their accuracy it is important to employ multiple tests after the MLP has been developed. An MLP can be evaluated on a number of different statistics that Keras keeps track of over the training and testing in order to asses how accurate a certain MLP is.

The first source of un-reliability in an MLP comes from the data passed into it initially, before any training and testing is ever done. An unbalanced proportion of signal and background data will lead to the MLP over-training on whichever data has the majority of the events. Over-training a MLP on large amounts of signal data compared to background data will lead to an excess in confidence of its performance. Over-training is the lack of generalization in an MLP [8]. To check if an MLP is over-trained, an analysis of the loss values, the error in a neural network's training and testing predictions [11], can be used. There are multiple types of loss, but for most binary classification MLPs, including this project, binary-cross entropy will be used [11]. If the difference between validation loss and training loss is greater than 0, then there is at least some over-training taking place [8]. Finding the gap in these two losses to be

0 is a perfect case scenario, and in real applications is more of a target value than mandate. The best way to check if a MLP is over-training is by analyzing loss curves describing training and testing epochs [11], an epoch being a neural networks full pass through the training data, as well as two other methods that will be discussed later.

This leads to the standard convention of including the same count of signal data as background, thus balancing the training/testing data before the MLP is built. Balancing signal and background input data is followed in the project, but it is interesting to note that even with large imbalances, upwards of a 60,000 event count difference between signal and background, the MLP still returns performance diagnostics similar to a balanced data set. Repeat trials of both excess signal and excess background proved this stability, and each time the MLP was able to differentiate the two in the same manner. It can be concluded that this property is due to the simplicity and integrity of the binary classification neural network. Yet, bias can still compound in the use of an imbalanced MLP leading to useless, unbalanced contours or low significance scores when used to predict a score for signal events. Limiting in this way helps improve the generality of this MLP's performance as well, since once all of the signal data is consolidated it can be randomized and cut away as to make each MLP slightly different than the last despite the target samples being the same.

Comparing the loss vs. epochs curves of both train and test phases of the MLP is one way to understand how well the MLP generalizes [8]. These values are calculated after each epoch, or training cycle, and stored as objects in the model history [3]. The lower the loss values are, the better the neural network's prediction becomes, and seeing the loss decrease over time means the neural network is learning [11]. Loss being too high and not decreasing despite the amount of epochs means that the model is under-fitting, or defining a function

that poorly represents the data [11]. The opposite problem is over-training, where the the fit represents the training and testing data set almost perfectly, but introducing new data from different slepton models for testing leads to the MLP having poor predicting power [8].

The second test is the inverse of comparing precision curves, and provides essentially the same check of training/testing properties. This quantity, the precision, determines the ability of the MLP to pick out the signal from background, and is represented by Eq. (4), where TP is the number of true positives, and FP is the number of false positives [2].

$$P = \frac{TP}{TP + FP} \tag{6}$$

This output can be tracked over training and testing phases, and should increase with increasing epochs. If the precision curve is high to begin with and hardly improves over epochs, then the model is assumed to be over-training [8]. Under-training occurs for the opposite, when the precision curve is around 50 percent over the whole training and testing phase and occurs when not enough data is present for the training of a robust MLP. The precision curve fluctuating by a few points over epochs about a certain score for the test scores is also normal, and seeing this trend get tighter over epochs is an even better indication of learning taking place [2]. When both of the precision and loss plots agree, then it is assumed that the MLP architecture is sufficient for a proper model to be generated.

The next test is to check the MLP's Receiver Operating Characteristic (ROC) curve. ROC curves compare the True-Positive Rate (TPR) to False-Positive Rate (FPR) and plot the points on a graph showing TPR vs. FPR. TPR and FPR are calculated using Eqs. (5) and (6) where FN is the number of false negatives, and TN the number of true negatives, and TP and FP behave

as previously defined [2].

$$TPR = \frac{TP}{TP + FN} \tag{7}$$

$$FPR = \frac{FP}{FP + TN} \tag{8}$$

The area under this curve is indicative of the MLP's classification abilities. This area can range from 0 to 1, a score of 0 meaning the MLP missclassified every event, 1 where the MLP correctly classifies every event [2]. If the area under the ROC curve (AUC) is 0 or 1, then the MLP is assumed to be untrustworthy and the MLP design must be changed. If the curve is a straight line through the middle of the plot, then the MLP is essentially guessing and has not understood the data set. For this project, a threshold of an area greater than .80 is necessary to move forward with the MLP, as a MLP returning an AUC score less than this is not optimized. There is also a maximum threshold area above which the model becomes untrustworthy, which is .95 for this project. This number corresponds to the AUC score in which loss curves and precision curves start showing aspects of over-training, and the neural network can be better optimized.

The final test is plotting each event by its respective score given by the neural network. Once the MLP has been created, the events that were assigned to train and test groups are sent back to the MLP, which gives each of them a score. If the MLP was perfect, all of the diboson background events would be given a score of 0, while all of the slepton signal events would be given a score of 1. Plotting a weighted histogram of the diboson count per score and the slepton count per score shows how effective the MLP is at classifying events, and if there are any scores in which the neural network is over-training on. The weights are determined event by event through the proper weight in Eg. (3). Shown in Fig. 4, this is possibly the most effective measurement of the MLP performance, as it describes the skill of the MLP as well as exposing areas of

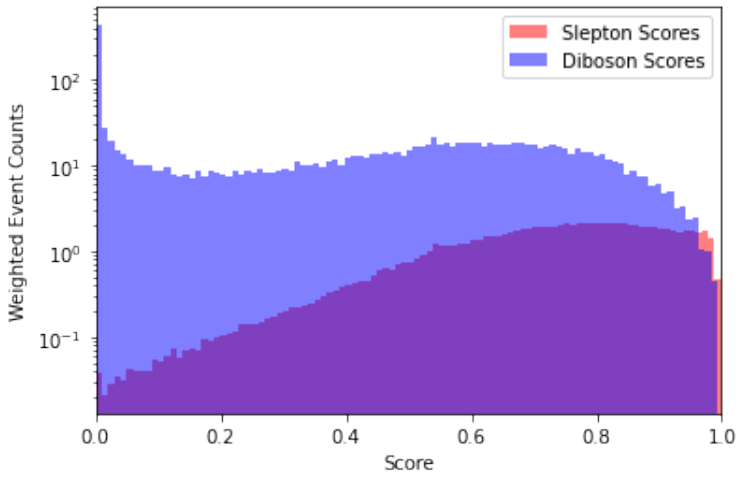

Figure 4: This figure displays an example of a weighted signal score histogram for each of the events used to train and test the MLP that created this histogram.

over/under-training in the model design. This plot is also useful in finding a region to calculate significance scores, where weighted signal events overcome background events, and will be returned to when discussing the results.

Through designing and back testing the MLP, an effective and trustworthy MLP can be created. The use of each of these statistics, plots, and hyper-parameters resulted in a robust predictor that could then be used in the face of new events outside of the training and testing data.

4 Outputs

When an MLP has been built and tested, the next step is to use it to make predictions on similar sets of data. Along with the outputs that evaluate MLP performance, there must be a set of outputs that target significance thresholds, significance scores, and sensitivity. Each of these play a role in excluding mass regions from the current mass parameter space of the slepton.

It is important to understand what these outputs will be used for, and how

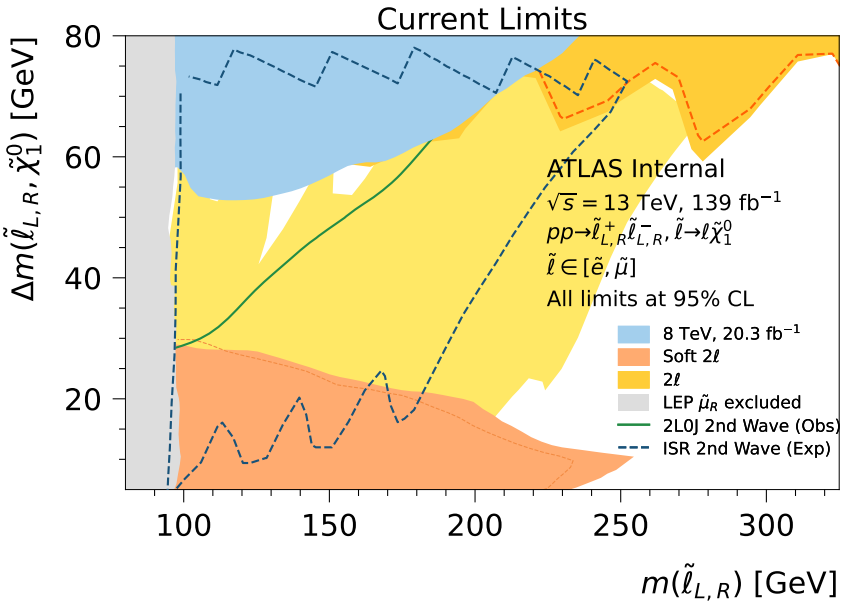

Figure 5: This figure displays the current exclusion plot of slepton events over a wide range on the parameter space. This data was formed through ATLAS searches listed in the legend on the figure.

they are applied to something physical, which in doing so grants researchers insight.

Fig. 5 shows the currently allowed parameter space of slepton mass vs. mass splitting from LHC run 2 data. By changing the MLP to focus on different regions of the slepton's parameter space, more and more regions can be excluded. If over repeated tests using different slepton models the statistics of each region stay the same then the parameter space can get tighter and tighter. The smaller this parameter space is, the better the slepton's kinematics can be defined which will boost the understanding of SUSY as a whole.

Thresholds and Significance

Using the MLP to predict how signal-like each of the events within a certain slepton mass/mass splitting grouping are will return the same scoring histograms used to check the performance of the MLP after training and testing. These histograms, shown in Fig. 4, is essential to understanding the significance of a given grouping, or how tuned the MLP is to this grouping of events. An MLP trained on similar slepton mass/mass splitting values as a set of events passed into the MLP will do better in distinguishing signal from background since the kinematic values (features passed in) will also be similar. Once the MLP predicts the signal score of each event, an iterative process can be started to determine an optimal signal score threshold, shown in Fig. 4 as the green line, that gives the best possible significance score. This process starts after the MLP has scored each event in a signal file representing a new point in the slepton parameter space and is paired with the diboson background. The threshold begins at a score of 0.99, and sums up the amount of both signal and background events respectively with scores in this region. Those values are then used in a significance calculation given by ATLAS [10], and shown in Eq. (9).

$$Z = \sqrt{2(n \ln \frac{n(b+\sigma^2)}{b^2+n\sigma^2} - (\frac{b}{\sigma})^2 \ln (1 + \frac{\sigma^2(n-b)}{b(b+\sigma^2)})} \tag{9}$$

The uncertainty used corresponds to the detector uncertainty in ATLAS of $\sigma = 0.3 * b$, where b is the number of background events, n is the number of total events

$$(n = s + b)$$

, and Z is the significance score. This process continues for each value on the x-axis of the signal score histogram by decreasing the starting score by

0.01, and re-summing all events within that new range. Since the region of interest is near the signal score of 1 this process is halted at a score of 0.9. This process happens for each new point in the slepton parameter space for a total of 60 different optimal thresholds corresponding to the 60 given slepton models per MLP. The distribution of optimal thresholds will be different when comparing one MLP to another as they are trained on different regions of the parameter space, so it is impossible to compare one MLP to another using the sensitivity plots generated through the optimal thresholds. A standard threshold is then devised by averaging all of the optimal thresholds after an MLP is used to probe the slepton parameter space, establishing an average optimal threshold for a given MLP, and then averaging that number between all four MLPs. By averaging all 60 of the optimal thresholds for each MLP, and then averaging those numbers, a common score is found as a way to compare MLP sensitivities on the same threshold. This threshold score was found to be 0.94. In order to create a reliable significance score, a threshold must also contain at least 3 of both background and signal events each. Both the optimal threshold significance and the standard threshold significance are saved per data set for use in determining a model's sensitivity.

Sensitivity

The final output of this project is a contour that acts as a performance diagnostic and provides insight into real world phenomena that could be leveraged for the use of SUSY discovery. By creating a contour that defines significance regions on the slepton mass vs. mass-splitting plane, its easy to see both where the MLP does the best and where we have the largest confidence in exclusion. How well a neural network performs on a given data set of slepton mass and mass splitting defines that MLP's sensitivity. High sensitivity neural networks are

MLPs trained on a specific mass splitting region, and can be efficient in areas very near to them on the mass spectrum, but their sensitivity drops off quickly as either slepton mass or mass splitting deviate from what the MLP is trained on. Low sensitivity MLPs are trained on a wide array of signal events from all different regions of the SUSY parameter space, and build an MLP that is more versatile, but much less detailed. A low sensitivity MLP will thus have a lower maximum significance score, but will fall off in sensitivity more gradually than a high sensitivity MLP. The use of both of these styles of MLP is instructive for mapping the SUSY parameter space confidently, as well as determining if certain neural network structures are superior for this task.

Mixed parameter space MLPs can also be utilized, as they attempt to bridge the gap and fill the shortcomings of the types of sensitivities mentioned before. These MLP's add versatility but still stay robust to the main region they are trained on. This type of neural network usually contains events from two or three different parameter spaces, and anymore than three causes the model to essentially become low sensitivity.

The image produced from these significance contours can be overlaid on the slepton's parameter space to show where the significance is highest, and where it starts to drop off for a given MLP. Multiple models can be combined to increase confidence around specific areas that are lacking from previous models.

5 Results

The results of this project can be divided up into two main outputs over 4 different MLPs. The 2 main outputs are the training/testing diagnostics and the MLP predictions with their resulting sensitivities. Each of the 4 MLPs is trained on signal events from different regions of the SUSY models, and the same set of diboson background events. The slepton and neutralino masses used

for these 4 neural networks are given in Table 1, 2, 3, and 4.

Slepton Mass	DM Mass	Split	Signal Count	Background Count	Total Event Count
200	170	30	251106	126906	253811

Table 1: Table describing an MLP built off of data with a slepton mass of 200 GeV and neutralino mass of 170 GeV.

Slepton Mass	DM Mass	Split	Signal Count	Background Count	Total Event Count
200	140	60	139743	126906	253811

Table 2: Table describing an MLP built off of data with a slepton mass of 200 GeV and neutralino mass of 140 GeV.

Slepton Mass	DM Mass	Split	Signal Count	Background Count	Total Event Count
200	170,140	30,60	390849	126906	253811

Table 3: Table describing an MLP built off of data with a slepton mass of 200 GeV and neutralino masses of 170 GeV and 140 GeV.

Slepton Mass	DM Mass	Split	Signal Count	Background Count	Total Event Count
100-300	25-290	10-75	117582	126906	235163

Table 4: Table describing an MLP built off of data with slepton masses ranging from 100-300 GeV and neutralino masses ranging from 25-290 GeV. This MLP also excludes events in the SUSY parameter space used by Tables 1 and 2.

The number of events used for either signal or background is limited to the smaller of the two event counts. The number of signal events are greater than background event counts for all MLPs except Table 4, which leaves Table 4's MLP with less data to use and develop with than the rest of the neural networks.

Performance Diagnostics

The first diagnostic we considered for the training and testing evaluation of the MLPs is the loss curve, which can be paired with the MLP's precision to demonstrate their agreement concerning performance.

Fig. 6 shows the loss and precision curves for the MLP resulting from training and testing on the slepton events described in Table 1. The loss curve shows a promising downwards trend for machine learning, with testing having less loss than training over the epochs due to dropout. While the precision scores do

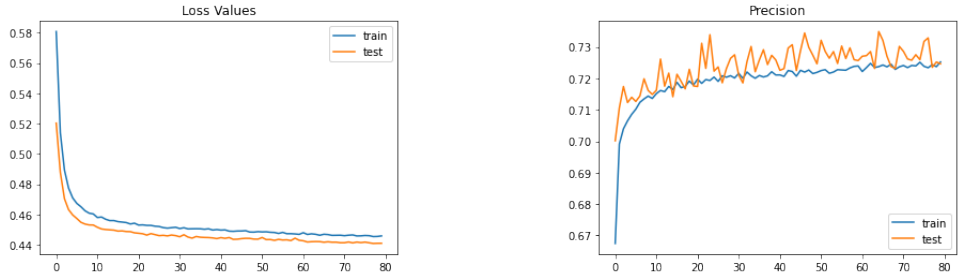

Figure 6: These are the loss and precision curves of a MLP trained on slepton events described by Table 1.

seem to shift abruptly from epoch to epoch, as mentioned earlier, the trend in the later half of the curve showing a tightening of the precision score range is a sign that the model is learning.

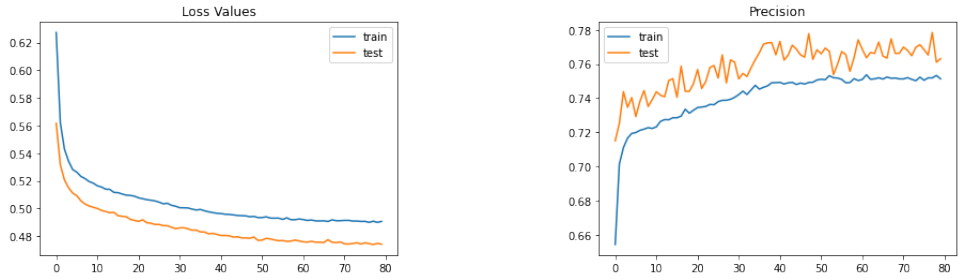

Figure 7: These curves display the loss and precision for an MLP built from events with a slepton mass of 200 GeV and mass splitting of 60 GeV.

Fig. 7 displays the loss and precision curves for the Table 2 MLP, which has similar trends to the Table 2 MLP's loss and precision curves. Not only does the Table 2 MLP indicate that there is true machine learning being done, but it also has the highest maximum precision value of all the MLPs. This comparison will be more apparent in the final contour results, but it alludes to machine learning neural networks training better on files with larger mass splittings.

The third MLP is a combination of data described in Tables 1 and 2, which is shown in Table 3. Fig. 8 displays precision and loss curves, returning the

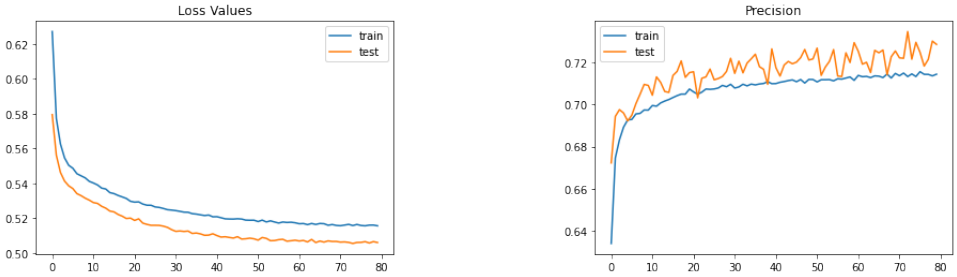

Figure 8: These precision and loss curves reference an MLP with data from Table 3.

highest loss so far by a couple percentage points. The precision is also well within expected ranges, and both curves show machine learning taking place. This precision curve is the least consistent as well, suggesting that the MLP is finding certain kinematic features that it can easily identify as signal events, while not having a great grasp of other features. This trend settles down as the epochs continue, again displaying machine learning at work.

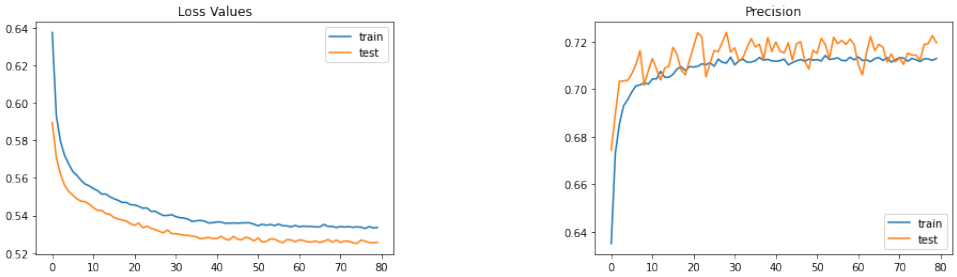

Figure 9: These precision and loss curves reference an MLP with data randomly selected from 58 sample signal files. These sample files have a minimum slepton mass of 100 GeV to a maximum of 300 GeV and a minimum mass splitting of 10 GeV to a maximum of 75 GeV

The final MLP is the Table 4 MLP, trained on signal events from the other 58 sample signal files that are not from the Table 1 and Table 2 files. This signal sample is smaller than the other MLPs with only 235163 events, and is the only data set to be passed into training that requires the background

events to be cut in order to reach a balanced data set as opposed to the signal events being cut. This may lead to the higher loss values seen in Fig. 9 as the neural network has no consistency in key kinematic features like slepton mass and mass splitting. The precision curve demonstrates one of the most noticeable convergence of scores out of all of the available MLPs. A convergence, shown as the precision curve's trend to stabilize around a certain number, suggests that the MLP starts to make connections between slepton events later on in the training than the other MLPs and could possibly benefit from more epochs than other MLPs. The training and testing phases stop for each of these MLPs when the training loss doesn't decrease for 30 epochs in a row, which is about 80 epochs of overall training. The ranges within which loss and precision stay is expected, and while the loss is higher than the other MLPs, it isn't so high as to require some unique architecture for the Table 4 MLP.

The second diagnostic is the ROC curve, which will describe how well our neural network worked and acts as more of a performance diagnostic than any other test.

MODEL	AUC Score (%)
Table 1	87
Table 2	84
Table 3	82
Table 4	81

Table 5: This table shows the area under the curves for each of the ROC curves returned after scoring the MLPs on their respective ability to discern between signal and background.

As shown in both Fig 10 and Table 5, the MLP derived from Table 1 has

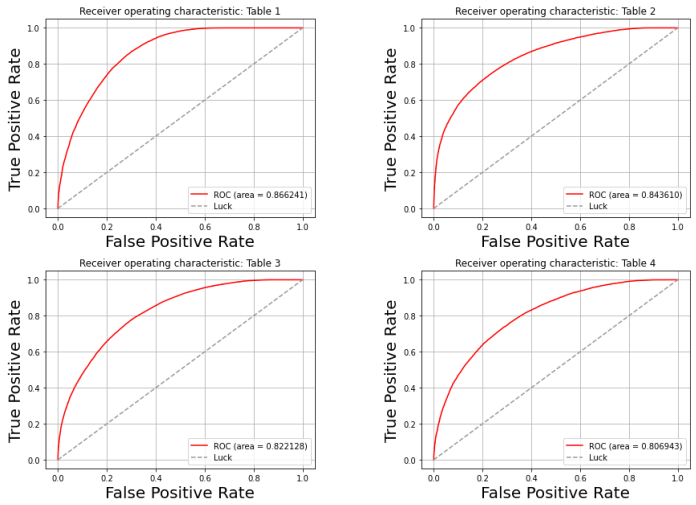

Figure 10: ROC curves of all four trained MLPs. The AUC score is found by calculating the areas under each of the curves, while luck defines the line that would have a score of 0.5 by not learning the data over epochs creating a MLP that can't be trusted as a predictor of signal events.

the best ROC score of 87%. The rest decrease from there but all stay at 80% or above.

The final diagnostic to be examined is the signal score histogram of the training and testing data set on its own model. Using the knowledge of how a perfect model would score these events, giving all signal events 1 and all background events 0, the signal score histogram can be used as a performance diagnostic as well as a method of defining sensitivities.

Fig 11 shows the signal score histograms for each MLP, with all of the histograms sharing the common feature of their respective SUSY model's signal overcoming background around the 1.0 threshold as it should. These signal score histograms are also weighted, which is impressive that signal overcomes background at all due to how the signal weights heavily suppress their affect on the count per bin. The significance scores, shown in Table 6, are derived from Eq. (6) and the threshold strategy mentioned prior.

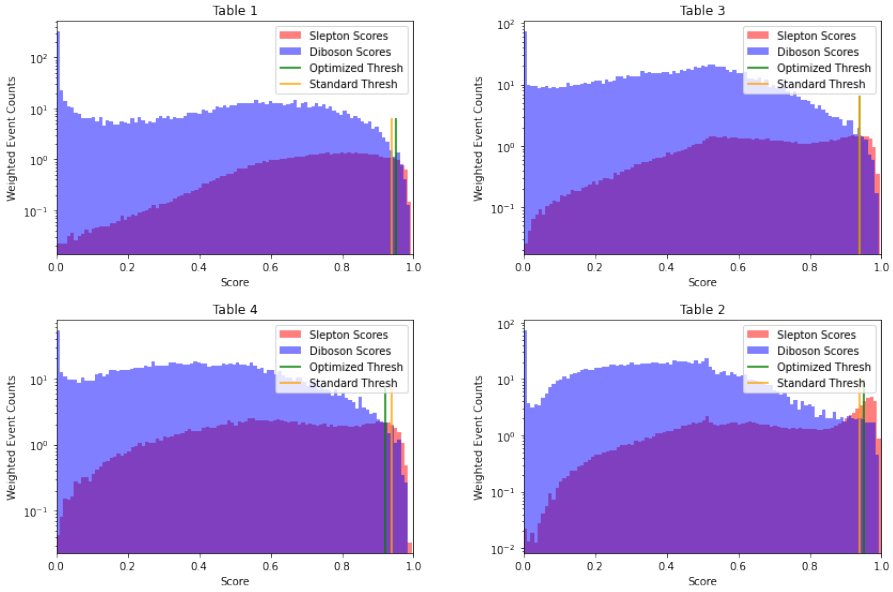

Figure 11: These four histograms define the signal score distributions of each of the MLPs. The same set of data that was used to train and test the MLP is given the the fully constructed MLP to score each event which gives the user and idea of how well the MLP will perform. The Standard threshold stays at a score of 0.94, while the optimized score changes between 0.9 and 1.0 according to the highest achievable significance score.

MODEL	Significance
Table 1	1.38
Table 2	3.65
Table 3	1.88
Table 4	2.06

Table 6: This table displays each of the significance scores found by summing the event counts from each MLP's respective histogram at the optimal score threshold and using Eq. (6). The significance scores are unit-less.

Comparing each of these significance scores illustrates how quickly significance increases when signal overcomes background in Fig 11. The score histogram for Table 2 returns a significance at the optimal score threshold of about double the next highest significance score reached by Table 4's optimal score threshold. The score histogram itself for Table 2 not only displays signal overcoming background counts per bin near a score of 1.0, but also spiking of signal scores demonstrating this MLP's ability to very effectively classify signal and background events. Despite not having the highest AUC score, this MLP does the best in practice which is implied by its high precision.

Significance Contours

Here it is useful to focus on each MLP separately and analyze each of the sensitivity contours with the training data in mind. Each color division in the contours will represent a change in the confidence interval (significance) σ, denoting the regions of the slepton parameter space in which the MLP's confidence drops. Comparing the standard threshold sensitivity contours of different MLPs to each other will also give insight into how a binary classification MLP handles a data set of this orientation in terms of trends and general preferences.

In ascending order of significance scores, the lowest scoring MLP, Table 1, is represented in Fig 12. Table 1's MLP reaches its highest σ in the mass splitting (Y-axis) region between 20 and 70 GeV, and slepton mass (X-axis) region between 100 and 175 GeV. The confidence of the model at 3σ extending out to about 200 GeV slepton mass aligns with the training of the model on the Table 1 data, while the noticeably extended region of higher sensitivity around the 30 GeV mass splitting point corroborates this statement. The standard threshold contains a much smaller 4σ confidence interval, highlighting the efficiency of the threshold optimization process. These two contours also demonstrate the dif-

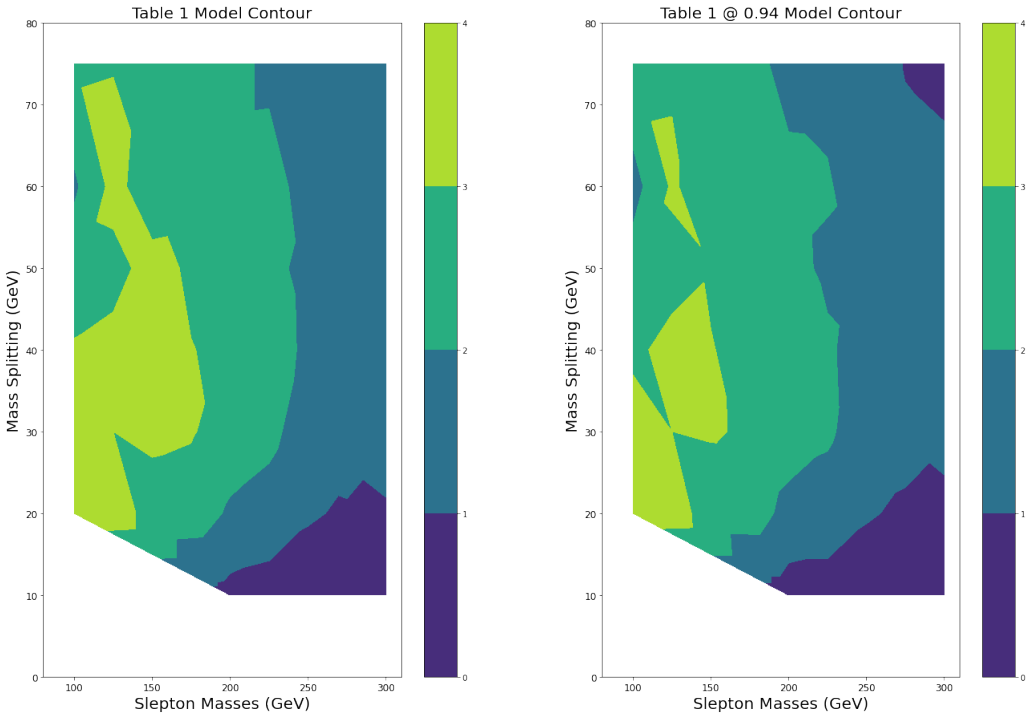

Figure 12: Significance contours of the Table 1 MLP for the optimized thresholds (left) and standard threshold of 0.94 (right). The confidence levels start at a minimum of 0σ and reach a maximum of 4σ.

ference that can sometimes exist between the standard score threshold and the optimal score threshold. The standard score threshold in Fig. 12 struggles to generalize in regions further away from the slepton parameter space described by Table 1, leading to worse confidence intervals overall.

The combined MLP from Table 3 correlating to the 3rd highest significance score has its sensitivity contour shown in Fig. 13. This contour was able to break into the 5σ confidence interval mainly due to the Table 2 data in the training set. Fig. 13 again displays the power of the optimized score threshold in boosting confidence intervals and confidence levels of these contours. While the standard score threshold does have wider confidence intervals than those in Fig. 12, it doesn't find the best possible scores from the MLP's classification

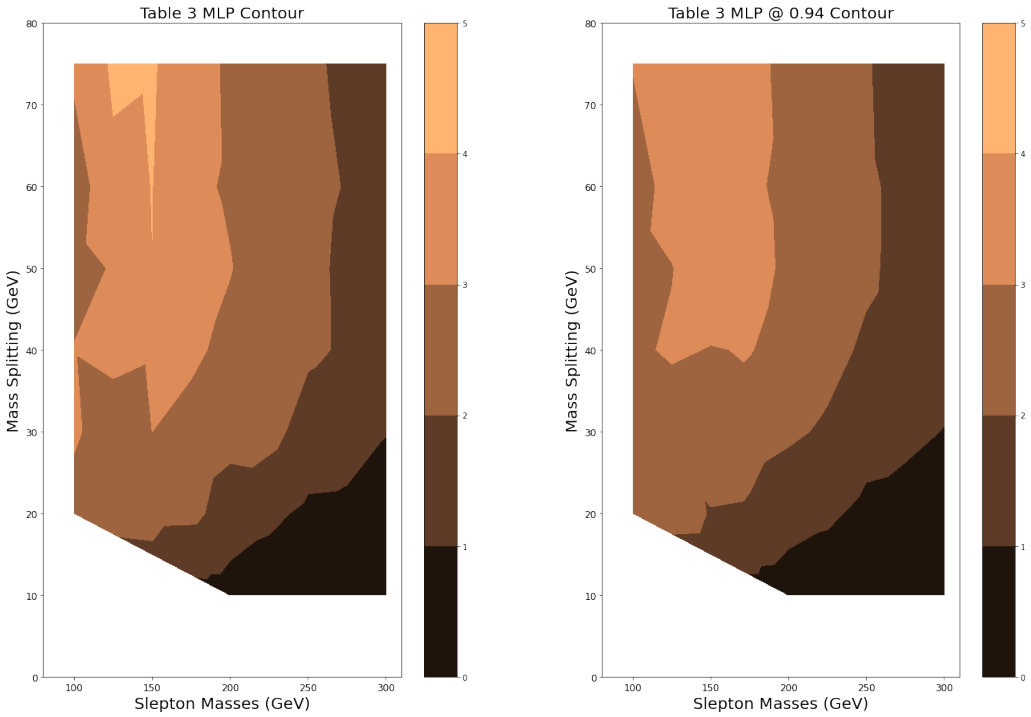

Figure 13: Significance contours resulting from the use of the combined MLP from sample slepton parameter space described in Table 3. The confidence levels vary from 0σ to 5σ.

abilities.

While the MLP built from Table 4 reached an optimal threshold scoring it the 2nd highest significance of the four models, its confidence intervals are weaker than the MLP built from Table 3.

Shown in Fig. 14, the Table 4 MLP shows wide yet shallow confidence intervals, reaching a maximum sensitivity at 4σ. This neural network had the best flexibility of all the MLPs, even slightly better than Table 3's MLP, performing better in regions of low slepton mass and low mass splitting. Despite other MLPs struggling here, the versatility of the Table 4 MLP after being trained on events from ranges of slepton masses and mass splittings, allow it to retain confidence in regions with less data. The optimized contour (left) also shows

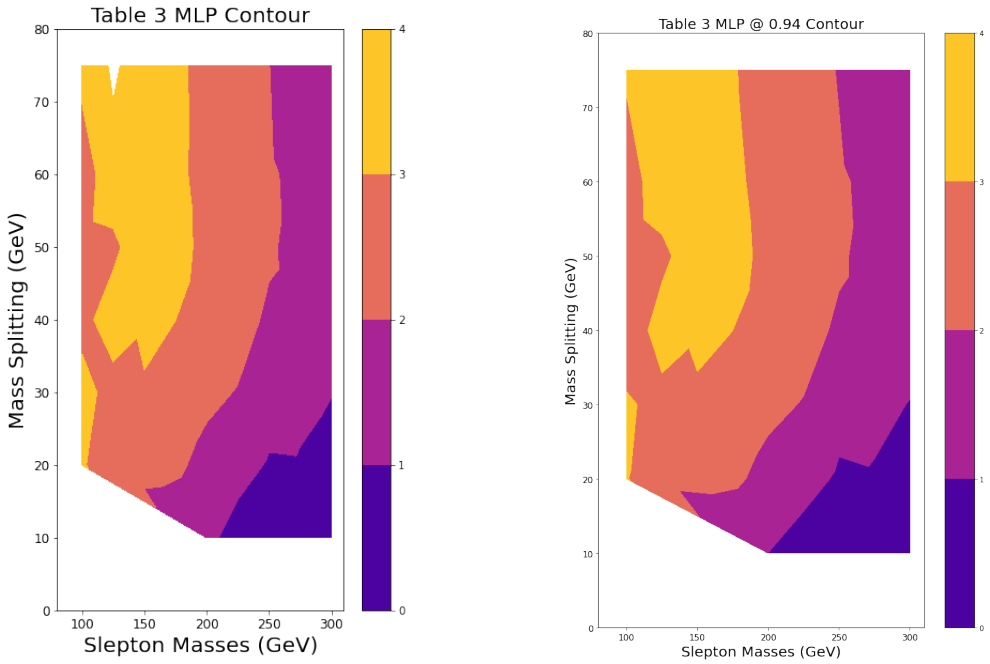

Figure 14: Significance contours of the Multi-model MLP being used on the sample slepton file array. This MLP displays lots of generality with a lower end maximum confidence level of 4σ but wider confidence intervals than MLPs trained on one point in the slepton parameter space.

a small sliver of the 5σ confidence level, but not enough to conclude that his model can consistently reach a 5σ maximum.

Table 2's MLP generated both the highest significance score of all models. Fig. 15 displays the MLP's sensitivity contour which reaches all the way to 6σ in the high mass splitting, low slepton mass region. The sensitivity is very high, as it is expected to be, but it also falls off as the MLP tries to make predictions in regions far outside of its training data. The area of mass splitting less than 30 GeV is in the confidence interval less than 3σ and can be safely ruled out according to this MLP. The 6σ region is most likely a product of the larger mass splitting, alluding to a lighter neutralino mass compared to the parent slepton as especially sensitive. This MLP proves to be the best MLP for not only high

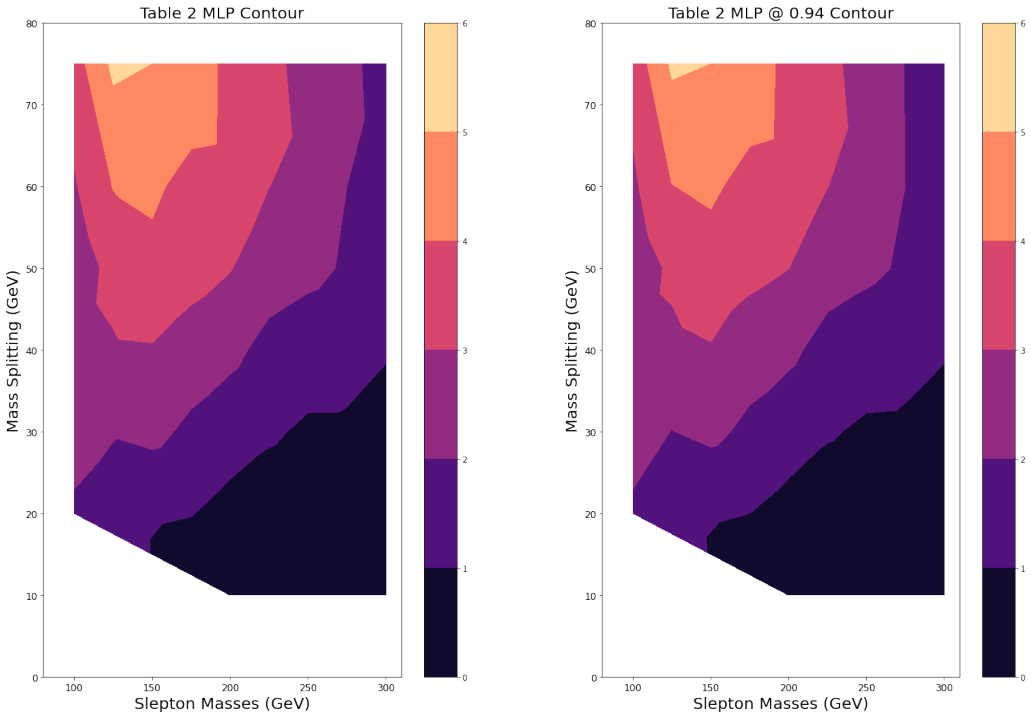

Figure 15: The MLP formed from Table 2 data contours display the highest confidence level which go slightly above 5σ across both the optimized threshold cuts (left) and standard threshold cuts (right).

confidence levels, but also wide confidence intervals with adaptability to regions outside of the training set.

6 Discussion

There are two main sets of results present here which need interpretation in the context of machine learning. The feedback diagnostics from the neural networks are of utmost importance in determining if machine learning methods are accurate enough platforms to be used in the classification task of high energy particle physics. Sensitivity contours are only useful if the feedback diagnostics confirm good performance and true machine learning taking place. The sensi-

tivity contours must also have high enough confidence levels in order to validate their usage over other methods of SUSY event detection.

An easy method for discerning the validity of loss, performance, and the ROC curve is to analyse each of their respective ranges in which the curves occupy. Each MLP had loss and precision curves in trustworthy regions, no loss curve drops below 0.4, and no precision curve goes above 0.8. Testing loss does its best around 0.44 in Table 1's MLP, and does the worst with the Multi-model at 0.52. This difference in minimum and maximum loss values is not only small but expected when considering the models for each case. Table 1's and Table 2's MLPs each have over 100,00 signal events all with the same kinematics, leading to increase sensitivity for SUSY events from these models, which is why Table 1's and Table 2's MLPs have the lower loss values between all 4 MLPs. For precision, the same is true, with MLPs from Table 1 and Table 2 having the best of the 4 MLP's scores. Table 2's MLP clearly has the best precision of all of the models with a high of 0.78, while the Table 4 and Table 3 MLPs have the lowest maximum precision at a value of about 0.72. The testing precision curves of each of these MLP's noticeably fluctuate, displaying the usage of a rigorous MLP and true machine learning through the decreasing of said fluctuations as the epochs progress. These values are congruent with the AUC for the ROC while also showing interesting interactions between the three performance diagnostics. All of the AUC scores being in the 80-90 percent range is a good sign, and reassuring that the MLPs are sufficient to be used in application. It may seem surprising, based off their definitions, that the MLP with the highest AUC score does not coincide with the MLP that has the highest precision values. The two diagnostics both display a given MLP's ability to distinguish between signal and background, but the AUC judges the MLP's probability of ranking events correctly while the precision judges the

MLP's ability to correctly find signal events. This gives AUC score a boost as it is increased by correctly identifying both signal and background.

The score histograms of each of the MLP's are very similar as shown in Fig 11. All of the histograms show small, signal dominant regions near the 1.0 score and reaching to about 0.9 before background surpasses it. Table 3's MLP has its optimized and standard thresholds placed on the same score of 0.94, highlighting the effectiveness of the standard threshold put in place. Table 4's MLP has an optimized threshold cut closer to 0.9 than the standard threshold, but still within scores of 0.3. One interesting observation that can be made from these histograms is how well Table 2's MLP does at picking out signal, demonstrated by not only a dominance in signal near 1.0, but a strong spike in signal. This pattern was recovered every time over 10 different builds of the Table 2 MLP, and is the reason that the confidence level in this MLP is so high. This same curve can be seen in the Table 3 MLP, but is severely blunted by statistics from Table 1. While Table 4's MLP also demonstrates the ability to separate signal from background above 0.9, the signal distribution is much more spread out over all scores, and isn't concentrated at the end like in Table 2's and Table 3's MLPs. Hence, the Table 4 MLP lacks the spike of events at the end.

Analysing the sensitivity contours clearly shows that the Table 2 MLP's distinguishing power over the entire range of slepton masses in the high mass splitting region (>60 GeV) makes this model the most useful for this area of study. The adequate range of the $> 4\sigma$ region is also promising, and highlights a consistent starting point for more intensive searches for SUSY. Over multiple runs of this MLP, Table 2 was able to extend the 2-3σ region out all the way to 300 GeV slepton mass at as low as 50 GeV mass splitting while still retaining a $> 5\sigma$ high confidence level and the features seen in Fig. 15. The contour

generated from the optimized thresholds has a similar set of confidence intervals as the standard threshold contour, showing that Table 2's MLP found a large amount of optimal thresholds on the 0.94 score. The standard threshold was also able to retain the $> 5\sigma$ confidence level. Table 3's MLP has the next highest confidence level, and displays the most depth of any of the MLPs. Table 3's MLP acts as a midway model between the specificity of the single file MLPs (Table 1,Table 2) in its adaptability to mass-splitting and slepton mass ranges outside its training data, but also retains the higher sensitivity to reach a $> 4\sigma$ confidence level. The tapering of these confidence intervals over the slepton mass demonstrates the MLP's understanding of both the high and low mass-splitting regions, but also showing a bias towards the higher mass-splitting region. The 2-3σ range extends out to 250 GeV slepton mass for the higher mass splitting region demonstrating extrapolation capabilities in slepton mass for well trained MLPs, as no events in this training set had slepton masses higher than 200 GeV. The standard threshold again displays more weaker sensitivity than the optimized threshold. This MLP also gives insight into why the mass splitting is so effective in training MLPs on slepton data. The higher the mass splitting between the slepton-neutralino pair, the more momentum the lepton has in the final state. When this lepton has more energy it is more likely to be picked up by a detector as a final state parton of the slepton decay channel, which adds sensitivity. Its easy to see in Table 3's sensitivity contour that while the Table 1 data extends the 4-5σ range further down the mass-splitting axis, the contour never goes above 5σ like Table 2's MLP using some of the same data. The Table 4 MLP functions similarly to the Table 3's MLP, as both are trained on data sets whose events come from different signal files. The 2-3σ confidence interval extending out to 250 GeV slepton mass in high mass splitting areas rivals the ability of Table 3's, with this confidence interval only regressing slightly as the

mass splitting decreases. Even though this neural network may lack in high sensitivity regions, its distinguishing power in parameter spaces with limited training data lead to the assumption that a similar neural network could perform exceptionally given more background and signal data. The standard threshold contour has similar confidence intervals to the optimized threshold once again, meaning the threshold significance scores are near to each other. The contours have an identical 2-3σ region so the difference in the thresholds does not have much effect for this MLP. Over repeated trials using this data set to form an MLP, the MLP would rarely reach a sensitivity that generated a small $> 4\sigma$ confidence level, but since this was a rare occurrence the $> 4\sigma$ region can be mostly ignored. This $> 4\sigma$ region is shown in Fig. 13 left as the small white spike at the top left of the optimized threshold sensitivity contour. While currently Table 4's MLP seems to act as a less effective version of Table 3's MLP, more signal data could bring out the $> 4\sigma$ region further and improve the statistics of this MLP.

Table 1's MLP is the most sensitive to the region of the mass spectrum in which it was trained on, giving it the smallest 2-3σ and above region. Over multiple runs of Table 1's MLP being used to predict the significances of 60 slepton files, the MLP's sensitivity varied noticeably. Certain sensitivity contours would fail to reach 4σ confidence intervals, and despite these being the outliers, it was very common to see the 3-4σ region as much smaller than shown in Fig. 11. This MLP is very sensitive to the region it was trained on without much extrapolation abilities towards mass spectrum regions of higher slepton mass. The success of Table 1's MLP demonstrates that neural networks of this structure do best with high mass splitting events, while it suffers when faced with lower mass splitting data sets whose final state momenta is more evenly split between the lepton and neutralino.

Takeaway and Future Projects

After examining the exclusion contours created by each of the MLPs, it is clear that the neural networks are sensitive to the the region of the parameter space within 40-75 GeV mass splitting with a slepton mass range of 100-200 GeV. Table 2's MLP proved through multiple tests that it is the best at classifying signal and background events. The use of high mass splitting slepton events is the common factor in high sensitivity with good generalization over the parameter space. Mixing in events that occupy different regions of the mass spectrum improve a MLPs flexibility to better predict events in untrained regions but including too many events with smaller mass splittings will decrease the maximum sensitivity below 5σ or even 3σ. Finding a better balance between the high performance of the high mass splitting events while mixing in fewer low mass splitting events that add to generality could make a more sensitive MLP than Table 2.

Training and using MLPs in particle physics is a viable option to increase research efficiency and accuracy while still providing diagnostics for trustworthy usage. These methods are new allowing for a plethora of new paths to explore in the machine learning space. Creating MLPs based on larger sets of background and signal data could improve performance statistics and give more robust contours. This same build architecture can also be used for different SUSY tests provided the correct QFT process data for both SUSY and background particles. Finally, a classification neural network could be used to not only categorize signal and background, but also mass ranges, particle class, and other kinematic identification which could further improve contours. Changing the neural network from binary classification to classification does change the neural network set up, but the input data can stay the same. Classification net-

works open the possibility for integration of other particle physics operations as well. These multi-faceted machine learning code modules could turn complex data manipulation and analysis into simple plug-and-play Python scripts which facilitate research and experimentation.

7 References

1. Griffiths, David J. Introduction to Elementary Particles. 2., rev. Ed., 5. reprint. Physics Textbook. Weinheim: Wiley-VCH, 2011.

2. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org.

3. Keras. "Keras Developer Guides." Accessed July 25, 2023. https://keras.io/guides/.

4. Kingma, Diederik P., and Jimmy Ba. "Adam: A Method for Stochastic Optimization." arXiv, January 29, 2017. http://arxiv.org/abs/1412.6980.

5. CERN. "Launch of the LHC Run 3," July 5, 2022. https://home.cern/events/launch-lhc-run-3.

6. Martin Erdmann, Jonas Glombitza, Gregor Kasieczka, and Uwe Klemradt. "Scope of This Textbook." In Deep Learning for Physics Research, 3–12. World Scientific, 2021. https://doi.org/10.1142/97898112374610001.

7. Martin, S. P. (1998). A Supersymmetry Primer. In Perspectives on Supersymmetry (Vol. 18, pp. 1–98). World Scientific. https://arxiv.org/abs/hep-ph/9709356

8. Mohammed, Nechba, Mouhajir Mohamed, and Sedjari Yassine. "High Performance Computing Applied to Logistic Regression: A CPU and GPU Implementation Comparison." arXiv, August 19, 2023. http://arxiv.org/abs/2308.10037.

9. Pedregosa, F, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, and P Prettenhofer. "Scikit-Learn:" Journal of Machine Learning 12 (2011): 2825–30.

10. Redlinger, George, and Paul de Jong. "Broken Symmetry: Searches for Supersymmetry at the LHC." A Commentary by ATLAS Physicists Paul

de Jong and George Redlinger on the History, Progress and Future of the Search for Supersymmetry (blog), December 8, 2017. https://atlas.cern/updates/feature/supersymmetry.

11. Saravanan, Prashanth. "Understanding Loss Functions in Machine Learning." Section, February 15, 2021. https://www.section.io/engineering-education/understanding-loss-functions-in-machine-learning/.

12. Sjöstrand, Torbjörn, Leif Lönnblad, and Stephen Mrenna. "PYTHIA 6.2 Physics and Manual." arXiv, August 31, 2001. http://arxiv.org/abs/hep-ph/0108264.

13. Steerenberg, Rende. "Accelerator Report: Crescendo at the LHC Following the First Stable Beams at 6.8 TeV," April 27, 2023. https://home.cern/news/news/accelerators/accelerator-report-crescendo-lhc-following-first-stable-beams-68-tev.

14. The ATLAS Collaboration. "Formulae for Estimating Significance." CERN, September 29, 2020, 24.

15. The GAMBIT Collaboration, Viktor Ananyev, Csaba Balázs, Ankit Beniwal, Lasse Lorentz Braseth, Andy Buckley, Jonathan Butterworth, et al. "Collider Constraints on Electroweakinos in the Presence of a Light Gravitino." arXiv, March 16, 2023. http://arxiv.org/abs/2303.09082.

16. Valdenegro-Toro, Matias, and Matthia Sabatelli. "Machine Learning Students Overfit to Overfitting." arXiv, September 7, 2022. http://arxiv.org/abs/2209.03032.

17. Alwall, J., R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H.-S. Shao, T. Stelzer, P. Torrielli, and M. Zaro. "The Automated Computation of Tree-Level and next-to-Leading Order Differential Cross Sections,

and Their Matching to Parton Shower Simulations." Journal of High Energy Physics 2014, no. 7 (July 2014): 79. https://doi.org/10.1007/JHEP07(2014)079.

18. Frederix, R., S. Frixione, V. Hirschi, D. Pagani, H.-S. Shao, and M. Zaro. "The Automation of Next-to-Leading Order Electroweak Calculations." Journal of High Energy Physics 2018, no. 7 (July 2018): 185. https://doi.org/10.1007/JHEP07(2018)185.

19. ATLAS Collaboration. (2020). Searches for electroweak production of supersymmetric particles with compressed mass spectra in $\sqrt{s} = 13$ TeV pp collisions with the ATLAS detector. Physical Review D, 101(5), 052005. https://doi.org/10.1103/PhysRevD.101.052005

20. Harris, Charles R., K. Jarrod Millman, Stéfan J. Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al. "Array Programming with NumPy." Nature 585, no. 7825 (September 17, 2020): 357–62. https://doi.org/10.1038/s41586-020-2649-2.

21. Hunter, J. D. "Matplotlib: A 2D Graphics Environment." Computing in Science & Engineering 9, no. 3 (2007): 90–95. https://doi.org/10.1109/MCSE.2007.55.

22. McKinney, Wes, and Others. "Data Structures for Statistical Computing in Python," Proceedings of the 9th Python in Science Conference, 445 (2010): 51–56.

23. Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems." arXiv, March 16, 2016. http://arxiv.org/abs/1603.04467.

24. Alwall, Johan, Michel Herquet, Fabio Maltoni, Olivier Mattelaer, and Tim Stelzer. "MadGraph 5: Going Beyond." Journal of High Energy Physics

2011, no. 6 (June 2011): 128. https://doi.org/10.1007/JHEP06(2011)128.

25. Zinser, Markus. "The ATLAS Experiment." In Search for New Heavy Charged Bosons and Measurement of High-Mass Drell-Yan Production in Proton—Proton Collisions, by Markus Zinser, 53–72. Springer Theses. Cham: Springer International Publishing, 2018. https://doi.org/10.1007/978-3-030-00650-15.

26. Jurafsky, Daniel, and James H. Martin. "Chapter 5 - Logistic Regression." In Speech and Language Processing, 3rd ed., 11–17, 2023. https://web.stanford.edu/ jurafsky/slp3/5.pdf.