(Machine) Learning to Do More with Less

Timothy Cohen, Marat Freytsis, and Bryan Ostdiek

Institute of Theoretical Science, University of Oregon, Eugene, Oregon 97403, USA

E-mail: tcohen@uoregon.edu, freytsis@uoregon.edu, bostdiek@uoregon.edu

ABSTRACT: Determining the best method for training a machine learning algorithm is critical to maximizing its ability to classify data. In this paper, we compare the standard "fully supervised" approach (that relies on knowledge of event-by-event truth-level labels) with a recent proposal that instead utilizes class ratios as the only discriminating information provided during training. This so-called "weakly supervised" technique has access to less information than the fully supervised method and yet is still able to yield impressive discriminating power. In addition, weak supervision seems particularly well suited to particle physics since quantum mechanics is incompatible with the notion of mapping an individual event onto any single Feynman diagram. We examine the technique in detail – both analytically and numerically – with a focus on the robustness to issues of mischaracterizing the training samples. Weakly supervised networks turn out to be remarkably insensitive to systematic mismodeling. Furthermore, we demonstrate that the event level outputs for weakly versus fully supervised networks are probing different kinematics, even though the numerical quality metrics are essentially identical. This implies that it should be possible to improve the overall classification ability by combining the output from the two types of networks. For concreteness, we apply this technology to a signature of beyond the Standard Model physics to demonstrate that all these impressive features continue to hold in a scenario of relevance to the LHC. Example code is provided on GitHub.

Contents

1   Introduction

Machine learning currently stands as one of the most exciting fields in computer science. The basic premise is to develop tools that can classify multi-variate data into categories by examining the data itself. Applications of these powerful tools within the fundamental physics community have been steadily gaining traction. Collaborations at the LHC have been incorporating this technology into various aspects of their characterization of objects, *e.g.*, bottom [1, 2] and charm [3, 4] quark tagging. Applications in the field of jet substructure [5–11] are quickly maturing as well. Techniques such as deep learning, image recognition, adversarial networks, and symbolic regression have also found their way into the high energy literature [5–7, 11–26].

An outstanding problem for the reliable interpretation of machine learning outputs to physics problems has been to gain control over the propagation of uncertainties of the input data through the learning algorithms. Some aspects of this concern can be addressed by emphasizing data-driven techniques so that as little of the machine learning architecture as possible depends on our (necessarily incomplete) models of fundamental interactions. Others can be aided through the use of adversarial networks to ensure that the learning step does not overly focus on a poorly modeled or correlated feature [11, 22]. Often, the approach is to systematically vary properties of the input data, and use the spread in the output measurement to determine an error profile. However, in many cases the labels that come with the input data in order to train the algorithms are saddled with their own uncertainties that

are correlated in a complicated way with the data itself. And if a confidence interval needs to be assigned to a given output of the classification, the problem becomes harder still.

With these questions in mind, the goal of this work is to explore the properties of a recently proposed approach: *weakly supervised neural networks*; hereafter be referred to as Dery, Nachman, Rubbo, and Schwartzman (DNRS) [27]. Machine learning algorithms learn to distinguish between signal and background by training on some user specified input data. In DNRS this step is formulated so that the input data sets are only labeled by a measure of the fraction of signal events contained in the set.[1] This is in contrast with the more widespread "fully supervised neural network," where one of the training sets is labeled as pure signal and the other as pure background. It opens the possibility of extending data-driven techniques into the realm of supervised learning methods, as DNRS-type networks could be trained on real data where absolute purity is unachievable.

In our view, weak supervision as a formalism touches on a much deeper connection to the nature of observables in particle physics. Quantum mechanics implies that no single event can ever be mapped onto an individual Feynman diagram at the fundamental level – event by event, the distinction between signal and background is not a well-defined notion. More practically speaking, isolating interesting sets of LHC data will always require some preselection. Assuming the selection is made such that there will be a non-trivial signal efficiency, the bin will be a mix of signal and background events, making the connection with weakly supervised training obvious.

Given this motivation, it becomes interesting to explore weak supervision beyond the compelling statements made in DNRS. Therein, this technique was applied to the question of quark–gluon discrimination, demonstrating that weak supervision appears to perform just as well as fully supervised neural networks while providing a way to potentially avoid concerns about the imperfect modeling of quark and gluon jets in (simulated) fully-labeled data. However, the need for fractional labels derived from elsewhere only moves concerns about modeling uncertainties elsewhere. Although overall sample fractions are indeed more theoretically robust than detailed event-by-event modeling, they do still come with their own uncertainties, whose effect on the learning algorithms one would like to be able to track.

One of the main points of this work is to demonstrate that, in fact, weak supervision is robust to such uncertainties and systematic errors. As we will show below, the machine learning algorithm is smart enough to learn to distinguish signal from background when training on mixed samples, and that the performance is almost independent of the "correctness" of the fraction labels. We will demonstrate this

---

[1]This new approach is an extension of the "bag label" or "multiple-instance" formalism previously proposed in the computer science literature [28, 29]. This is a variation on supervised learning where individual labels are not known, but in a "bag" of samples, the number of positive samples may be known.

surprising statement numerically first using a toy model, and then, after providing analytic understanding of the behavior, on a more realistic collider physics example.

In particular, this observation broadens the appeal of weak supervision to questions where data-driven training is less viable, like beyond the Standard Model (BSM) searches, on which there has been less focus on applying machine learning techniques. (Although see *e.g.*, [14, 30–39] for a few exceptions.) With no BSM physics yet found at the LHC, there is no real-world data to train classifiers of BSM physics. However, concerns about propagation of uncertainties through the "black box" of machine learning are still present. Moreover, concerns about how accurately the Standard Model (SM) backgrounds are being modeled remain. We will demonstrate that in such a BSM scenario, for the reasons outlined above, weakly supervised classification still provides advantages.

The rest of this paper is organized as follows. Section 2 provides a review of neural networks generally, along with a discussion of both weakly and fully supervised training. Section 3 applies this technology to a toy model in order to compare the performance of weakly and fully supervised networks and to demonstrate the surprising robustness of weak supervision. Section 4 gives an analytic argument for this insensitivity to mislabeling. Section 5 applies these tools to a beyond the Standard Model example by comparing the signal of gluinos to $Z +$ jets. Finally, Section 6 provides a demonstration that weak and fully supervised networks probe complementary regions of phase space, and closes with a discussion of many future directions.

2   Review of Weak Versus Full Supervision

Multiple-instance learning [28, 29] concerns itself with a class of supervised machine learning problems wherein only the aggregate properties of sets are presented for training, while definite properties of individual members are unknown to the algorithm. While it has already seen application in more traditional machine learning domains, the use of these techniques in the context of high energy physics was only recently presented by DNRS under the moniker of weakly supervised classification. For physics applications, one appeal is that since truth-level labels are not required, the networks can be trained on real, as opposed to simulated, data. DNRS showed that weakly supervised classifiers as trained on distorted data demonstrated lower performance degradation than fully supervised ones. This motivates a more complete characterization of the error tolerance of weak supervision, and in doing so we find some surprising behavior and resultant additional applications. But first, in order to orient the reader, we review traditional fully supervised approaches to event classification and contrast them with the newer weakly supervised method.

2.1 Artificial Neural Networks

Throughout this paper, we will be examining binary classification between events of type 0 and 1 using an artificial neural network. A schematic of an example network shown in Fig. 1. Artificial neural networks are comprised of a set of layers, denoted by the dashed green boxes. In the figure, and in all of the results in this paper, the network consists of three layers: an input layer, a hidden layer, and the output layer.

Each layer contains a fixed number of nodes n (sometimes referred to as neurons), denoted by the yellow circles in the figure. The input layer contains a node for each feature, *i.e.*, independent variable, in the input data; n_{input} is fixed by the input data set. It is customary to add a constant *bias* node to this layer as well, which can be thought of as the analog of adding a y-intercept when fitting data with a line. For the hidden layer(s), there is the freedom to choose n for each layer. This is an example of a user-defined parameter or *hyper-parameter*, to be distinguished from the fitting parameters or *weights*. We follow the standard method and add a bias node to the hidden layer as well. The nodes of the input and hidden layer (including the bias nodes) can be viewed as two vectors, with dimension $n_{\text{inputs}} + 1$ and $n_{\text{hidden}} + 1$, respectively. Last, the output layer contains only one node, and the value it yields is the prediction of the network.

To pass the input signal to the output, the nodes of the layers need to be connected. The connections are marked by the green lines in Fig. 1, such that transferring from one layer to the next is simply matrix multiplication. We have represented these matrices in the sketch by the dotted yellow boxes, with labels $\mathsf{w}^{(1,2)}$. Thus, the hidden layer is connected to the input layer by $h_i = w_{ij}^{(1)} x_j$. The dimension of $\mathsf{w}^{(1)}$ is $n_{\text{hidden}} \times (n_{\text{input}} + 1)$, and each element of the matrix is a learnable weight. Similarly, the output layer is connected to the hidden layer via the matrix $\mathsf{w}^{(2)}$, which contains $1 \times (n_{\text{hidden}} + 1)$ learnable weights, giving $\mathcal{Y}_i = w_{ij}^{(2)} h_j^a$.

In the figure, the vectors $\mathbf{h}$ and $\mathcal{Y}$ are marked by blue circles, to represent the *activity* of the nodes. The activity of each node is then passed through an activation function, which is marked as the blue arrows taking $\mathbf{h} \to \mathbf{h}^a$ and $\mathcal{Y} \to y_{\text{p}}$. For all of the networks considered in this paper, we use the activation function

$$y(h) = \frac{1}{1 + e^{-h}} = \qquad , \tag{2.1}$$

also known as the logistic or sigmoid function: it maps $\mathbb{R} \to (0, 1)$ continuously, with a shape that mimics that of a step function. In general, the choice of the activation function is another hyper-parameter. Here, then the neuron can interpolate between on ($\equiv 1 \equiv$ signal) or off ($\equiv 0 \equiv$ background) depending on if the activity is either positive or negative. The sigmoid function is useful because it is smooth, continuous, and has a trivial derivative, which all contribute to efficient network training.

A few common generalization can extend the flexibility of a neural network. For *deep learning* extra hidden layers could be added, with the number of hidden layers
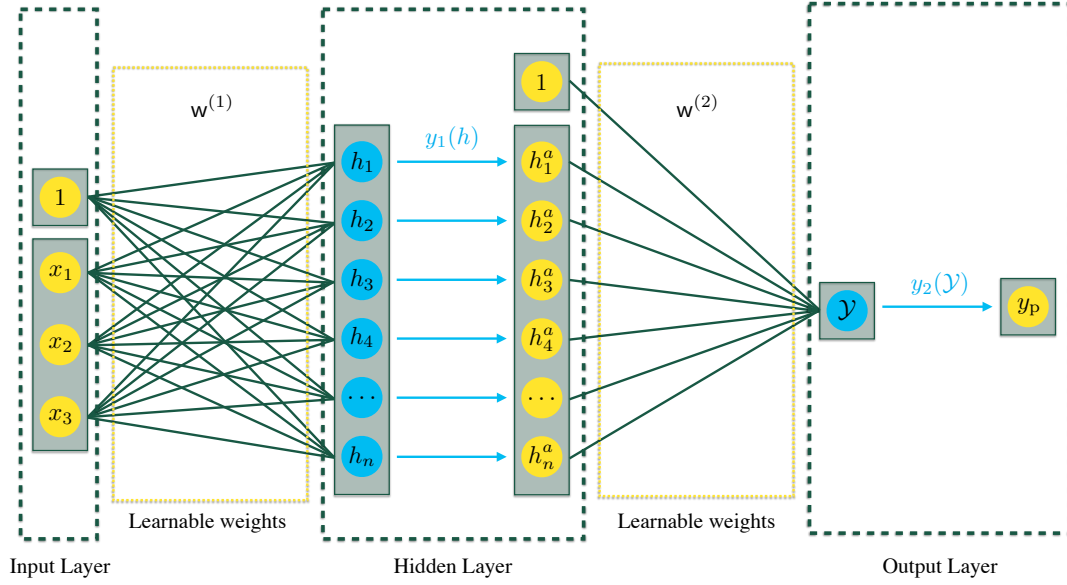
Figure 1: A schematic representation of a network with three layers (one hidden), such as the networks used in this paper. Each of the layers is marked by a dashed green box. All of the yellow circles represent a real number (node). The green lines are the connections between nodes and are each learnable parameters. The light blue arrows map the node activity (number in blue) to the node by applying the activation function.

and the number of nodes in each layer being free hyper-parameters. Also, while we use Eq. (2.1) for each of our activation steps, a wide variety of different function can be used on a node-by-node basis; it is not even necessary that they be deterministic.

To summarize, for a binary classification problem the output of the network $y_p = g(\mathbf{x}; \theta)$ for a given event can be generalized to a function of the event features $\mathbf{x}$ and the learnable parameters $\theta = \{\mathbf{w}^{(1,2)}\}$. For brevity we will not write out the full expression for y_p which can be inferred from the structure of Fig. 1. Once the network is trained (see the next subsection), θ becomes fixed, and an event will therefore always yield the same prediction. On the other hand, during training the goal is to find a well-performing set of parameters θ by minimizing a *loss function*, which compares the known labels of the training set with the network output as we will detail below. The loss function is minimized by taking its gradient with respect to the parameters in θ, for fixed $\mathbf{x}$. As a technical aside, the initial weights of $\mathbf{w}^{(1,2)}$ before training are chosen from a normal distribution. The specification of this initial condition is yet another hyper-parameter.

2.2 Loss Functions

While the general structure is the same for the standard fully supervised networks, and the weakly supervised networks proposed by DNRS, one design aspect that differentiates the two approaches is related to the information with which the network is given to train. This, in turn, determines the chosen loss function. For example, in linear regression problems, where one wants to predict a number and not a class, a common loss function is the mean squared error, determined by how far off the predicted value is from the true value. The weights can then be tuned to minimize the error.[2]

The approach in classification problems is similar, but instead of predicting a number, the goal is to predict the class. As shown above, the last step of our networks is to pass the output activity, $\mathcal{Y}$ through the activation function. The larger $\mathcal{Y}$ is, the closer to 1 the prediction. Conversely, large negative values of $\mathcal{Y}$ map to a prediction of class 0. The absolute size of $\mathcal{Y}$ can be thought of as how confident the network is about the prediction. This highlights why, while one could still use the mean squared error as the loss function, it is not ideal for classification. For instance, imagine comparing two events with truth labels $y_{t,i} = 0$, where one has an output activity of $\mathcal{Y} = 3 \rightarrow y_p \sim 0.95$ and the other has $\mathcal{Y} = 6 \rightarrow y_p \sim 0.998$. The error for the second one is only about 5% larger, even though the network was twice as 'confident' in its wrong assumption that the event was from class 1. The standard approach to classification uses a loss function which penalizes not just having the wrong answer, but also the level of confidence in the prediction.

Full supervision: For full supervision, the true class for each event is known, denoted by $y_{t,i}$ for event i. We implement the binary cross entropy (BCE) loss function,

$$\ell_{\mathrm{BCE}}\big(\{y_t\}, \{y_p\}\big) = \sum_{i \in \mathrm{samples}} \left[y_{t,i} \log \frac{1}{y_{p,i}} + (1 - y_{t,i}) \log \frac{1}{1 - y_{p,i}} \right]. \qquad (2.2)$$

The first term of ℓ_{BCE} is identically 0 for events from class 0, while the second term is 0 for events from class 1. Revisiting the example events with $\mathcal{Y} = 3$ and 6 from above, when $y_t = 0$, the loss is $\ell_{\mathrm{BCE}}(\{0,0\}, \{0.95, 0.998\}) \simeq 3 + 6$. Thus, while the network predicts that both events are signal like, the second event is penalized much more in the loss, because of the network's confidence in the wrong answer. The converse happens when the predicted class is correct. For example,

---

[2]As with all machine learning, there is always a worry that one *overtrains* such that the neural network becomes overly sensitive to the detailed properties of the training data set because of the large number of learnable parameters contained in θ. In order to mitigate this issue, we use an independent validation data set which is piped through the network at the end of training. Then we check that the loss function returns values that are within tolerance of the loss function outputs that are achieved for the training data.
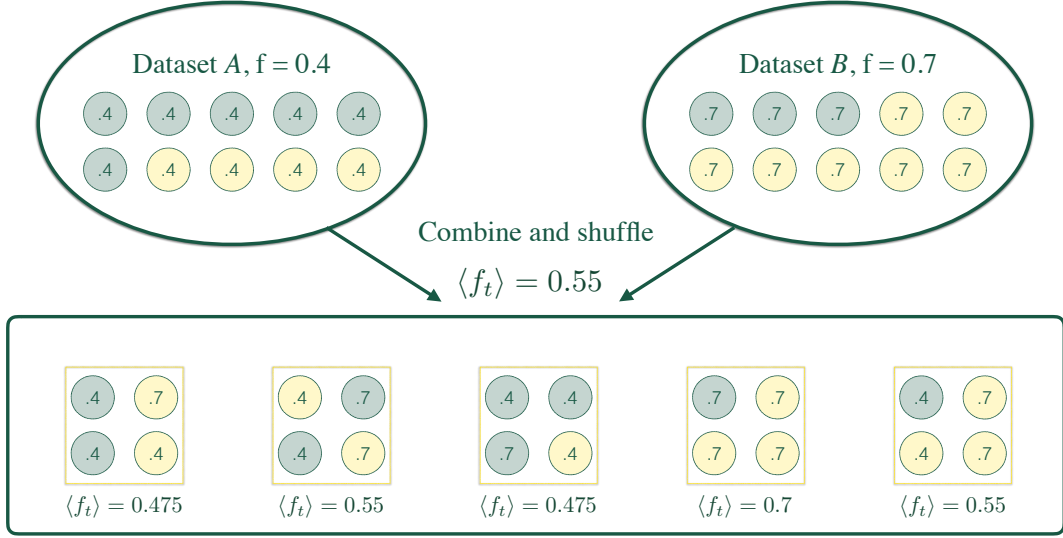
Figure 2: Schematic of the data flow when using weak supervision. We are given (at least) two data sets and told to learn two different classes (yellow and green), without knowing the truth value of any one event. Instead, each event is only labeled with the ratio of yellows to the total that are in its dataset. The datasets must be combined and shuffled into batches, that are used for training (the groups outlined in yellow squares) such that the minimization procedure sees many different ratios. For illustration in this figure, the batch size is four, and this represents one random permutation.

$\ell_{\mathrm{BCE}}(\{1,1\}, \{0.95, 0.998\}) \simeq 0.05 + 0.002$. The BCE loss function gives a comparable very small loss for all reasonably confident, correct predictions, but larger losses for increasingly confident, wrong predictions. It is worth stressing that this can only be used if the truth level class of each event is known.

Weak supervision: By contrast, for weak supervision, all that is known is the fraction of event classes in the training sample. This fraction contains all the information supplied for the supervision of the training and will be denoted by f_t. It will act similarly to y_t, however, f_t is the same value for all events in a sample, regardless of whether they are from class 0 or 1.

For illustration, see Fig. 2, where we have two unique datasets, A and B, and we want to classify events as green (class 0) or yellow (class 1). However, the truth level value of each event is not known, only the ratios in the data set. Each event carries the label f_t from the fraction of yellow circles in its dataset; all of the events from datasets A and B are marked with 0.4 and 0.7, respectively. Since the individual labels of 0 (green) and 1 (yellow) are unknown event by event, Eq. (2.2) cannot be used as the loss function.

Clearly, a new approach is required. To this end, DNRS introduced a loss function that compares the difference in the predicted fractions:[3]

$$\ell_{\text{weak}}(\{f_t\}, \{y_p\}) = \big| \langle f_{t,i} \rangle - \langle y_{p,i} \rangle \big|, \qquad (2.3)$$

where we use $\langle \cdots \rangle$ for the mean, and y_p is still the prediction of the network. First, take dataset A alone, the goal is to achieve $y_p = 1$ for the four yellow events and 0 for the six green events. Then $\langle r_t \rangle = 0.4 = \langle y_p \rangle$, which minimizes the cost. However, this is not a unique minimization; if every event has $y_p = 0.4$, the cost is trivially minimized. This issue is mitigated when multiple input data sets with different known fractions, conventionally termed *bags*, are utilized. Furthermore, training must be done in *batches* (random partitions of the original training set), rather than on the entire, combined dataset.

We can use Fig. 2, where we indeed have our datasets with different fractions, to understand this explicitly. The two datasets are combined (and shuffled) into one dataset; each of the 10 events from A and B has been copied to the lower box. If we now train on the whole, combined dataset, we run into the same problem as when we only had one dataset. The combined $\langle f_t \rangle = 0.55$, so training would lead to $y_p \simeq 0.55$ for every event. The solution is to break the dataset up into batches, in the example we have split the dataset up into 5 batches each with 4 events, denoted by the smaller boxes.

We proceed with training, treating each batch as its own independent dataset. The key here, is that $\langle f_t \rangle$ is not calculated by the color (the true label which is unknown) but by each of the individual $f_{t,i}$, which comes from whether it was part of A or B. The first batch in Fig. 2 contains two green and two yellow circles, but only one of them comes from B, so $\langle f_t \rangle = 0.475$ for that batch. The weights, θ, of the network are updated using the gradient of the loss function, yielding a y_p output that is closer to 0.475 for these events. The amount that the weights change is controlled by the learning algorithm and the learning rate. After updating the weights once for the first batch, we then move to the next batch, where the process is repeated. This continues until the weights have been updated for each of the batches.

In practice, only running through the training samples once does not allow for enough θ updates to yield good discrimination, although this does depend on the overall size of the dataset. The process of shuffling the data, splitting it into batches, and training should be repeated many times; this choice is known as the number of *epochs* and is a hyper-parameter, as is the size of the batches. Now all relevant hyper-parameters have been defined, and Table 1 contains the choices used for the networks in this work. Note that while batches are often used in fully supervised networks for computational efficiency, it is not required by the algorithm. The implementation of weak supervision, however, would not work without batching.

---

[3]A comment on notation compared to DNRS: we use f for the fraction in weak learning and y for full supervision.

Choice	Toy Full	Toy Weak	BSM Full	BSM Weak
Loss function	BCE	Weak	BCE	Weak
n_{input}	3	3	11	11
Hidden Nodes	10	30	30	30
Activation	Sigmoid	Sigmoid	Sigmoid	Sigmoid
Initialization	Normal	Normal	Normal	Normal
Learning algorithm	SGD	Adam	SGD	Adam
Learning rate	0.01	0.0015	0.01	0.00001
Batch size	32	32	32	32
Epochs	40	100	10	10

Table 1: Values of hyper-parameters chosen for the networks used below. Here, we mimic the choices made in DNRS and leave exploring variant settings in detail to other work. The learning is implemented using KERAS [40] with the ADAM optimizer [41] for weak supervision and stochastic gradient descent (SGD) for the fully supervised case.

Figure 2 has been used as a tool to help visualize the setup of weak supervision, but as such, it is highly idealized. When actually implementing weak supervision on data, the two datasets do not need to be the same size. In addition, if weak supervision is used on real data, as proposed in DNRS, the true label of events will be unknown. In the context of particle physics, the different samples could be obtained by mutually exclusive bins in the transverse momentum or pseudorapidity, for example.[4] In such a situation, theory could predict the ratios, even without having a labeled dataset. However, there are always errors inherent in the predictions, and understanding their impact will occupy much of what follows.

Next, we consider a simple toy model and its behavior under the distortion of the input labels. Applications to collider physics are presented later in Section 5.

3 Label Insensitivity in Toy Models

To gain some intuition, we begin by working with simple toy models built from multivariate Gaussians. Distributions for events of class 0 and 1 are produced in three independent variables. To ensure that the impressive performance of the weak learning algorithm are robust to complicated distributions, we make each variable

---

[4]There is an important assumption which is that the underlying distributions of the features should remain consistent for a given class across the different datasets. This implies one must be careful when choosing the binning that yields the input data.
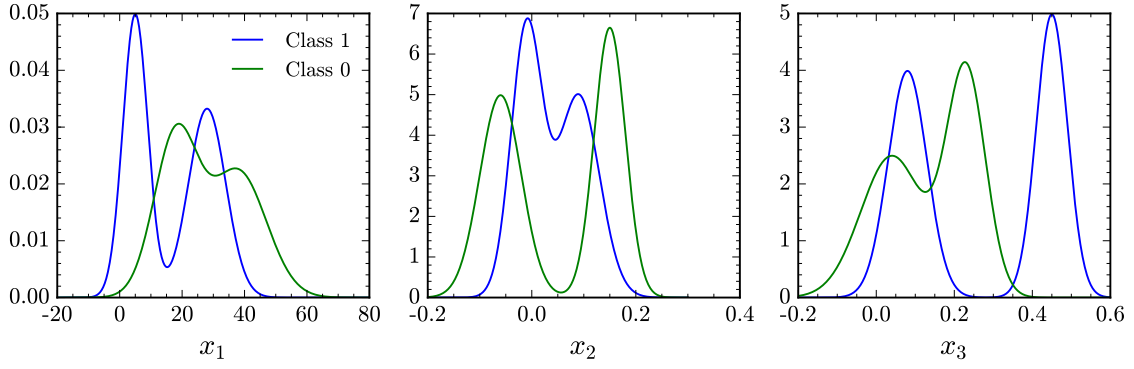
Figure 3: Probability density functions for our toy model. The means and standard deviations of the individual modes are found in Table 2.

| Feature	$\mu_1^{(1)}$	$\sigma_1^{(1)}$	$\mu_1^{(2)}$	$\sigma_1^{(2)}$	$\mu_0^{(1)}$	$\sigma_0^{(1)}$	$\mu_0^{(2)}$	$\sigma_0^{(2)}$
x_1	26	8	5	4	18	7	38	9
x_2	0.09	0.04	-0.01	0.03	-0.06	0.04	0.15	0.03
x_3	0.45	0.04	0.08	0.05	0.23	0.05	0.04	0.08

Table 2: Mean and standard deviations of the normal distributions used for the classes 0 and 1. Each feature (variable) is derived from the bimodal distribution in Eq. (3.1) with the events spread evenly between distributions $j = 1, 2$.

bimodal:

$$P(x_i) = \sum_{j=1}^{2} \frac{1}{2} \frac{1}{\sigma_i^{(j)} \sqrt{2\,\pi}} \exp\left[-\frac{1}{2} \left(\frac{x_i - \mu_i^{(j)}}{\sigma_i^{(j)}} \right)^2 \right] \qquad (3.1)$$

These were chosen such that a completely unsupervised clustering algorithm (such as k-means [42–44]) would not group the events correctly. Figure 3 plots the distributions for each of the features in the toy model utilized for the rest of this section, with their underlying parameters in Table 2. The hyper-parameters of the trained network used in our studies are provided for reference in Table 1.

The resulting networks are trained on this toy model with two datasets. Both contained 200 000 samples with fractions of 0.4 and 0.7 of class 1 events. The two training datasets were combined and shuffled (as shown in Fig. 2) with 20 % of the sample set aside as a validation set. The classifier's performance is then tested on another dataset of the same size with 0.55 of the events in class 1. The `StandardScaler`
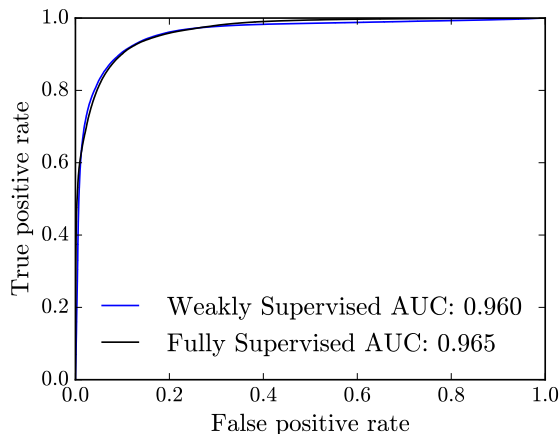
Figure 4: Receiver operator characteristic (ROC) curve for the weakly supervised and fully supervised networks. The performance is very similar for the two networks.

of the `scikit-learn` [45] package is used to normalize and center the data from the training set. These transformations are applied to both the validation and test sets.

To quantify the performance of various classifiers, we construct receiver operator characteristic (ROC) curves. A ROC curve is generated by plotting the true positive rate against the false positive rate, where the cut on the output discriminator y_p varies along the curve.[5] For a given network and set to be classified, the ROC curve is a measure of its performance for binary classification. A perfect classifier would always have a true positive rate of 1.0 with no false positives. Thus, curves pushed to the upper-left corner are better (a diagonal line of is then equivalent to a 50–50 guess). One common metric to compare classifiers is to take the integral of these curves, defined as the area under the curve (AUC), with an AUC of 1 being perfect.

Figure 4 shows the two resulting ROC curves comparing the weakly and fully supervised networks on the same training and test data. The AUC for the networks are 0.960 and 0.965, respectively. This serves to validate the claim that weak supervision can yield comparable results to a fully supervised network, despite not knowing event-by-event labels.

To study the sensitivity of the method to accurate label information, we explore how the network performs when there is uncertainty in the fraction labels of the datasets. The same data structure, with dataset fractions 0.4 and 0.7, each with 200 000 samples, are used throughout. However, for the set with a fraction of 0.7, the actual number fed into the f_t for these samples is varied between 0.0 and 1.0 in steps of 0.1. With the weak loss functions in Eq. (2.3), changing f_t should result in

---

[5]Note that constructing a ROC curve requires passing a fully labeled data set through the already-trained network. We use our test data set for this purpose when constructing all ROC curves in this paper.
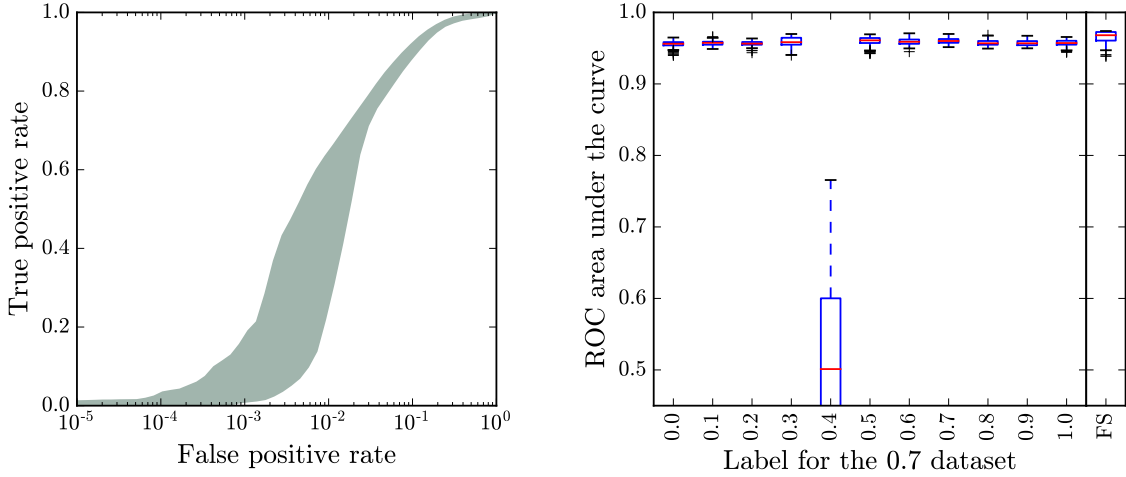
Figure 5: The left panel shows the spread derived from the different networks trained using the correct choice of 0.4 and 0.7 for the labels. The area under the curve is very similar for all of them, but the curves differ at very low values of the false positive rate, demonstrating the possible spread in network performance trained on the same data. The right panel shows a comparison of the fully supervised network (labeled "FS") and the weakly supervised network when one of the datasets labels does not match its true ratio. One data set has a true ratio of 0.4 and is labeled as such. The other dataset has a ratio of 0.7, but the network is trained using the label as specified on the x-axis.

a change in the prediction. As there are some stochastic steps involved in training the networks (both fully and weakly supervised) we repeat the process 100 times at each value of the label to get a sense of the uncertainty.

In the left panel of Fig. 5, we have plotted the spread in ROC curves for the different trainings for the networks using the true labels of 0.4 and 0.7. We see that while the integrated area under the curve is very similar for the set, they behave quite differently when high purity samples are requested. The right panel contains box-and-whisker plots (boxplots) of the AUC for each of the labels using all 100 trainings, along with the 100 instances of the fully supervised network. The boxplots have the red line at the median of the 100 samples, with the boxes extending between the 25% and 75% quantile (50 % of the samples are in the box). The whiskers extend to the remaining events, unless the point is an outlier, *i.e.*, more than 1.5 out of the quartile range, in which case it is marked by a cross.

Amazingly, even though the mislabeled dataset really has a ratio of 0.7, it is difficult to see any real difference in performance for the mislabeled datasets, with the exception of the 0.4 case.[6] While they do slightly worse than the fully supervised

---

[6]For the case where the 0.7 dataset is labeled as 0.4, both datasets are labeled with the same value, so the network defaults to the trivial solution, guessing a value of around 0.4 for every sample in the set. The performance of the network is in fact random. However, we do not allow for an
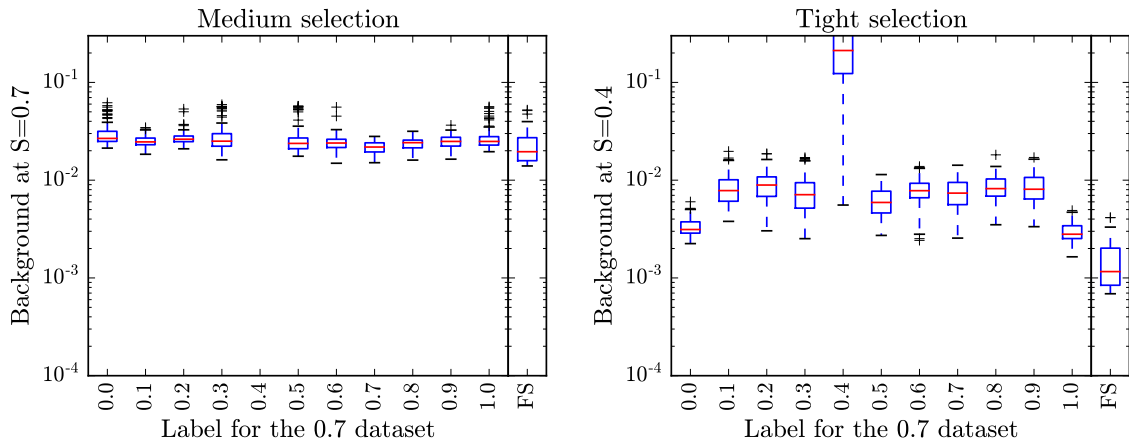
**Figure 6**: Background efficiencies at fixed values of signal efficiency.

network, their performance shows remarkable resilience with respect to training set mislabeling.

The AUC gives some sense of how good the classifier is, but actually using the trained network for predictions requires choosing an operating point somewhere along the curve. For instance, tagging of objects at the LHC (such as a b-jet or a hadronically decaying τ) often have different working points, where the true positive rate (signal efficiency) have a fixed value. We mimic this, using a signal efficiency of 40 % as tight selection, and 70 % as a medium selection. The best performing network will then allow the least amount of background at these fixed values of signal efficiencies. Figure 6 shows boxplots of the background efficiency for the different mislabelings at the medium (left) and tight (right) working points. At the medium working point, the median background efficiency for the fully supervised network is around 2 %. When the data is (correctly) labeled with 0.7, a similar background rejection is achieved in the weakly supervised network. However, this mislabeled datasets behave slightly worse, with median background rejections around 3–4 %. There is no obvious trend where making the mislabeling worse leads to further degraded performance.

The tight working point of Fig. 6 does show some sensitivity to the size of labeling error. For the fully supervised network, this loss in signal efficiency leads to a reduction in background by slightly more than an order of magnitude. However, the weakly supervised networks do not show the same level of increase in the background rejection. It is not surprising that working at points with high background rejection, the fully supervised networks can outperform weak supervision. However, what is truly unexpected is that the mislabeled data sets can yield better rejection than when the ratio is correctly labeled.

AUC of less than 0.5, and instead flip the guess when this happens.
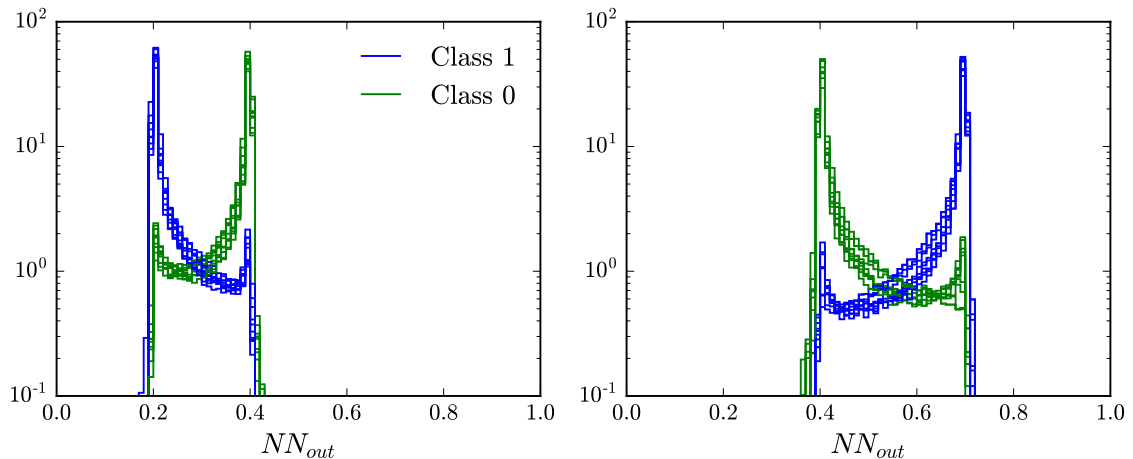
Figure 7: Outputs of the weakly supervised neural network. The blue lines show distribution of signal for 10 separate trainings, while the green lines show the background. The left and right panels represent when the data set which is has a truth value ratio of 0.7 is labeled as either 0.2 or 0.7, respectively. As pointed out in DNRS, the possibility of class-flipping exists for weakly supervised networks. To get the signal events in the left panel, the cut on the network output is an upper bound, while in the right panel it is a lower bound.

A suggestive aspect of the classifier resilience can be seen in Fig. 7, which shows 10 overlaid histograms of the neural network outputs for the signal and background test samples. The left panel shows distributions for when the data set which has a true fraction of 0.7 is mislabeled as 0.2, while in the right panel that sample is labeled with the true fraction. We see that the network changes the preferred location for the signal samples, but that the classifier distributions of events by class look like rescaled versions of each other. Thus, while the predicted value for any given sample may change, the network is still able to tell a difference between signal and background.

The change in the y_p distributions shown in Fig. 7 provides one possible explanation for why the trainings labeled with 1 or 0 in the tight selection can perform better than with the true label of 0.7. Recalling that the last step of the network is passing through the sigmoid activation function, with the hope of mapping the events to close to 0 or 1. However, these distributions show that network output is close to the fraction labels instead. Thus the network is not allowed to be 'confident' and really push the prediction to 0 or 1 with large sizes of the activity. However, if mislabeled with a 0 or 1, this is possible. Trying to obtain pure samples is similar to asking for high levels of confidence in the prediction, so this could explain the surprising result.

The results of this section have focused on when only one of the fraction labels is mislabeled. However, it is realistic to think that an uncertainty in the fraction could

extend to both datasets. We have checked multiple ways of mislabeling both datasets including both fractions shifting up or down and the fractions shifting towards and away from each other. In each instance, we find that the results presented here hold, and there is no significant change in the performance of the network. The next section examines the conditions such that weakly supervised networks can remain so robust to input fraction errors.

4   Analytic Condition for Label Insensitivity

The behavior of the classifier in the previous section with respect to a mislabeling of the fractional labels seems surprising. After all, it is as though we have removed essentially all supervision from our supervised learning algorithm without any significant degradation of performance. This is, in fact, true, although only to a point. This insensitivity to inputs can be traced to the fact that the optimal classifier is not, in fact, unique, but rather parametrized by a family of functions related by all possible monotonic mappings. The putative optimal classifier that an algorithm reconstructs can consequently suffer from quite large distortions without degrading its overall performance. However, the mislabeling of the training data does extract a price, as the particular cut associated with a given working point will end up misidentified. These claims are made more precise below, along with giving a relatively simple (to state, if not implement) breakdown criterion at which performance will start getting worse.

First, we consider the behavior of an optimal classifier given correct inputs and infinite training statistics. For a given dataset to be classified, the optimal classifier, given the inputs, is one that best approximates the probability of an event being in a given event class at every point in feature space. The ratio of these probabilities can then act as a classifier variable that can be cut on. Expanding on the arguments of DNRS, let us consider a discretization of our input features. Every training set then corresponds to an n-dimensional histogram, n being the number of features. Referring to each bin by the collective coordinate i, the distributions for each event class can then be found by inverting the known distributions on a bin-by-bin basis.[7] Here, $h_{A,i}, h_{B,i}$ denote the number of events in bin i of training sets A, B, and $f_{A,B}$ the total fraction of event class 1 in each set.

$$
\begin{aligned}
h_{A,i} &= f_A h_{1,i} + (1 - f_A) h_{0,i} \\
h_{B,i} &= f_B h_{1,i} + (1 - f_B) h_{0,i}
\end{aligned}
\quad \Longrightarrow \quad
\begin{aligned}
h_{0,i} &= \frac{f_A h_{B,i} - f_B h_{A,i}}{f_A - f_B} \\
h_{1,i} &= \frac{(1 - f_B) h_{A,i} - (1 - f_A) h_{B,i}}{f_A - f_B}
\end{aligned}
\tag{4.1}
$$

---

[7] With more than two training samples, the system is over-constrained. Analytically this could be dealt with by solving the resulting equations with respect to some test statistic, *e.g.*, least-squares. In a machine learning context, the training procedure packages together the freedom associated with this choice with that of approximating the distributions over feature space into one step.
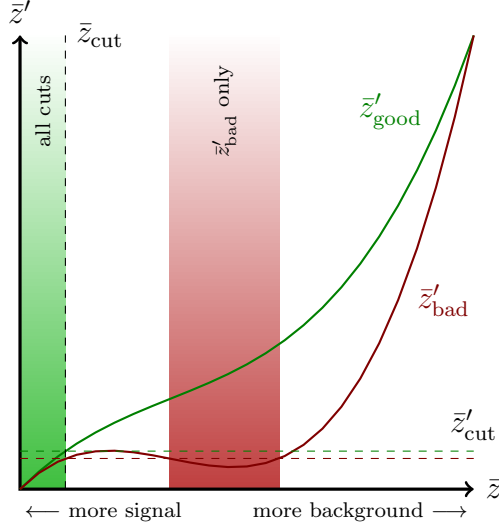
Figure 8: The effect of a cut on a deformed classifier $\bar{z}_i'$ in terms of cuts on an optimal classifier $\bar{z}_i$. As long as the mapping between the two is monotonic, a single cut on $\bar{z}_i'$ will always correspond to some cut on the optimal classifier. As soon as monotonicity breaks down, some cuts on $\bar{z}_i'$ will be contaminated by less signal-pure regions of feature space.

In the limit of infinite statistics, this inversion will accurately compute the event class distributions in every bin. An optimal classification could then be achieved given by cutting on

$$\bar{z}_i = \frac{h_{1,i}}{h_{0,i} + h_{1,i}} \,. \tag{4.2}$$

Crucially for our results, this choice is not unique.

Why this should be so can be seen from Fig. 8. Since $\bar{z}_i$ is an optimal classifier by construction, a single cut on $\bar{z}_i$ will correspond to the best possible purity that can be achieved for a given signal acceptance rate. As long as the mapping from $\bar{z}_i$ to $\bar{z}_i'$ is monotonically in- or decreasing over the range of the original classifier, in this case $\bar{z}_i \in [0,1]$, a single cut on $\bar{z}_i'$ will correspond to some cut on the optimal classifier. As soon as the monotonicity breaks down, there will be some choice of cut on $\bar{z}_i'$ for which less signal-pure regions of feature space are included without optimally increasing signal acceptance. If we can identify the point at which monotonicity breaks down as a function of input mislabeling, we can predict at what point performance will degrade.

The explicit form of the optimal classifier allows us to see why mislabelling the training sets has so little effect. From Eq. (4.1), we write

$$\bar{z}_i = \frac{h_{1,i}}{h_{0,i} + h_{1,i}} = \frac{(1-f_A)h_{B,i} - (1-f_B)h_{A,i}}{(1-2f_A)h_{B,i} - (1-2f_B)h_{A,i}} = \frac{1-f_B}{1-2f_B} \frac{\frac{1-f_A}{1-f_B} - r_i}{\frac{1-2f_A}{1-2f_B} - r_i}, \tag{4.3}$$

where we have defined $r_i = h_{A,i}/h_{B,i}$. As one would expect, the classifier checks if r_i is close to the ratio one would expect for a region dominated by one class and returns a value close to that class. The mislabeling of Section 3 could then be captured by a fractional shift $f_A \to f_A + \delta$, leading to the classifier being reconstructed as

$$\bar{z}_i' = \frac{1 - f_B}{1 - 2f_B} \frac{\frac{1-f_A-\delta}{1-f_B} - r_i}{\frac{1-2f_A-2\delta}{1-2f_B} - r_i} = \bar{z}_i + \delta \left(\frac{2\bar{z}_i^2 - \bar{z}_i}{(1 - f_A) - (1 - f_B)r_i - 2\bar{z}_i\delta} \right)$$

$$= \bar{z}_i + \delta \left(\frac{2\bar{z}_i^2 - \bar{z}_i}{(1 - f_A) - (1 - f_B)r_i} \right) + O(\delta^2). \quad (4.4)$$

If the mapping from $\bar{z}_i$ to $\bar{z}_i'$ is given by a monotonic function for positive $\bar{z}_i$, then by the discussion above, they actually correspond to the same optimal classifier, and no loss of performance will occur. Since $\bar{z}_i$ can in principle take on any value between 0 and 1, to determine monotonicity we are justified in treating it as a continuous parameter, so that

$$\frac{\mathrm{d}\bar{z}_i'}{\mathrm{d}\bar{z}_i} = 1 + \delta \left(\frac{\left[(1 - f_A) - (1 - f_B)r_i\right](4\bar{z}_i - 1) - 4\bar{z}_i\delta^2}{\left[(1 - f_A) - (1 - f_B)r_i - 2\bar{z}_i\delta\right]^2} \right)$$

$$= 1 + \delta \left(\frac{4\bar{z}_i - 1}{(1 - f_A) - (1 - f_B)r_i} \right) + O(\delta^2) \quad (4.5)$$

Near $\bar{z}_i \approx 0.25$, the slope is manifestly positive, so that for the mapping to be monotonic, it must be monotonically increasing. This will be true if the second term remains larger than -1. While this is ultimately dependent on the unspecified function $r_i(\bar{z}_i)$, for it is clear that as $|\delta|$ becomes smaller, the condition becomes easier to achieve, and that for distributions in which the denominator never becomes particularly small, it will remain satisfied even for moderate values of δ. Again, as stressed with Fig. 8 above, the mapping from $\bar{z} \to \bar{z}'$ must be monotonic if it is to stay within the family of optimal classifiers.

What is happening here is that in the limit of infinite statistics, the weakly supervised classification problem really has an analytically optimal solution. But by distorting the inputs, the event class probabilities at each point in feature space seen by the learning algorithm are incorrect. However, this only causes improper separation of signal and background if feature space points with different event class probabilities are mapped to the same incorrect probabilities. As captured in Eq. (4.4), this requires a conspiracy between the mislabeling and the event density in the two training sets at a particular point in feature space, two entirely independent quantities. As a result, for most distortions the actual contours of the classifier output swept out in feature space do not change, only the numbers associated with them. Thus, the danger still remains that the wrong working point is used when a particular signal strength is required, but it will correspond to a different optimal working point rather than a suboptimal algorithm.

5 Label Insensitivity for LHC Physics

Part of the motivation provided in DNRS for the use of weakly supervised networks is that they can, in principle, be trained on real data, and thus be robust against modeling errors. This strategy applies well to tagging SM objects, as demonstrated there, using weakly supervised networks to distinguish between quark and gluon jets. This motivation may not apply to searches for BSM physics, where there is no way to guarantee that new physics is in the real data, let alone that we know the ratio.[8] However, the use of weakly supervised networks could provide a safeguard both against mismodeling and signal–background contamination BSM searches.

5.1 Gluino versus $Z +$ jets

The particular BSM scenario we choose to study is the canonical gluino–neutralino simplified model. We set the mass of the gluino to be $2 \, \text{TeV}$, which is near the current limits [46, 47]. The gluinos are pair produced, and then decay to the neutralino (which we take to be massless) and a pair of quarks. This yields a signature of many jets and missing energy. One of the dominant backgrounds for these types of searches is $Z +$ jets, with Z decaying to neutrinos. As this is a proof of concept, rather than a dedicated search, we only use this background, with one hard jet in the initial interaction.

MADGRAPH5 [48] is used to generate the signal and background Monte Carlo samples. The events are showered and hadronized using PYTHIA6.4 [49]. Detector simulation is done with DELPHES3 [50] using the default detector card, modified to use anti-k_t jets [51] with a radius of $R = 0.4$, as calculated with FASTJET [52]. We initially generate $500\,000$ events for the each of the signal scenarios and $1\,000\,000$ for the background. The background samples have an extra, generator-level cut such that the jet has $p_T > 150 \, \text{GeV}$ and the minimum missing energy is $150 \, \text{GeV}$. At the detector level, we impose an additional cut so that both the missing energy and the transverse momentum of the hardest jet is greater than $200 \, \text{GeV}$. For any jet to be considered, it must have $|\eta| < 2.5$, and we record the p_T of up to 10 jets, as long as they have $p_T > 40 \, \text{GeV}$. Additionally, any events that contain an isolated lepton are vetoed.

Of the 10^6 Monte Carlo events for the background, $264\,303$ passed the cuts. Meanwhile, $473\,359$ out of the $500\,000$ events passes the initial cuts. More events are vetoed (due to leptons) in the 3rd generation fermionic decays. Out of all the events passing the cuts, $10\,\%$ are set aside to use as a test set. The remaining are split $80\,\%$ for training and $20\,\%$ for validation.

We first start with fully supervised training with the goal of distinguishing between the background and the gluino pair production. The data from the training set

---

[8]It is interesting to consider if this could be used to help distinguish between BSM scenarios when a large excess is discovered. We leave studying this application to future work.

| Network	AUC	Cut	Background efficiency	Signal efficiency
Full	0.99992	$1 - 1.39 \times 10^{-5}$	3.78×10^{-5}	0.843
Weak	0.99991	0.757	3.78×10^{-5}	0.849

Table 3: Metrics for training networks to distinguish gluino pair production with decays to 1st generation quarks from the dominant Z + jet background.

is centered and normalized using the `StandardScaler` function in the `scikit-learn` package [45], which is likewise applied to the validation and test data. A fully supervised network is trained within the KERAS framework [40] using one hidden layer with 30 neurons, where each layer is initialized with the normal distribution and uses a sigmoid activation. We use 11 inputs for the network, the missing transverse energy along with the transverse momentum of the first 10 jets passing the selection cuts. If there are less than 10 jets, the corresponding input is 0. The minimization is done using SGD with a learning rate of 0.01. The resulting network is an excellent classifier; the area under the ROC curve is 0.99992.

Next, we do the analysis on the same gluino and background events but use weak supervision. To do this, we split the data into two different training sets. The first set contains 80 % of the background and 40 % of the signal events. The second set has the remaining 20 % of the background and 60 % of the signal. The validation sets are split in the same fashion. This leads to the ratios of the two data sets being 0.47 and 0.83, respectively. The weakly supervised network contains 30 neurons in the hidden layer and is minimized using the ADAM optimizer [41] with a learning rate of 0.00001 for 10 epochs using batch sizes of 32. The area under the ROC curve (when using the test data) comes to 0.99991.

Both networks are again excellent classifiers, exemplified by the area under the ROC curves. However, as we argued before, this is not the best measure for the performance of a classifier in the context of a particle physics experiment. Instead, here we are imagining using the networks for a discovery of new physics. In this case, we want to choose an operating point where a small (fixed) amount of background would be accepted, with the signal efficiency maximized. The test set (as described above) has around 20,000 background events total. We set the cut such that only one background event passes. The resulting cut is shown in Table 3. After making this cut, the signal efficiency comes to 0.843 and 0.849 for the fully and weakly supervised networks, respectively.

The networks trained above were based on 10^6 Monte Carlo events. However, the cross sections for these processes are so large that the effective luminosity for these events is not large enough to span the full relevant phase space. For example,
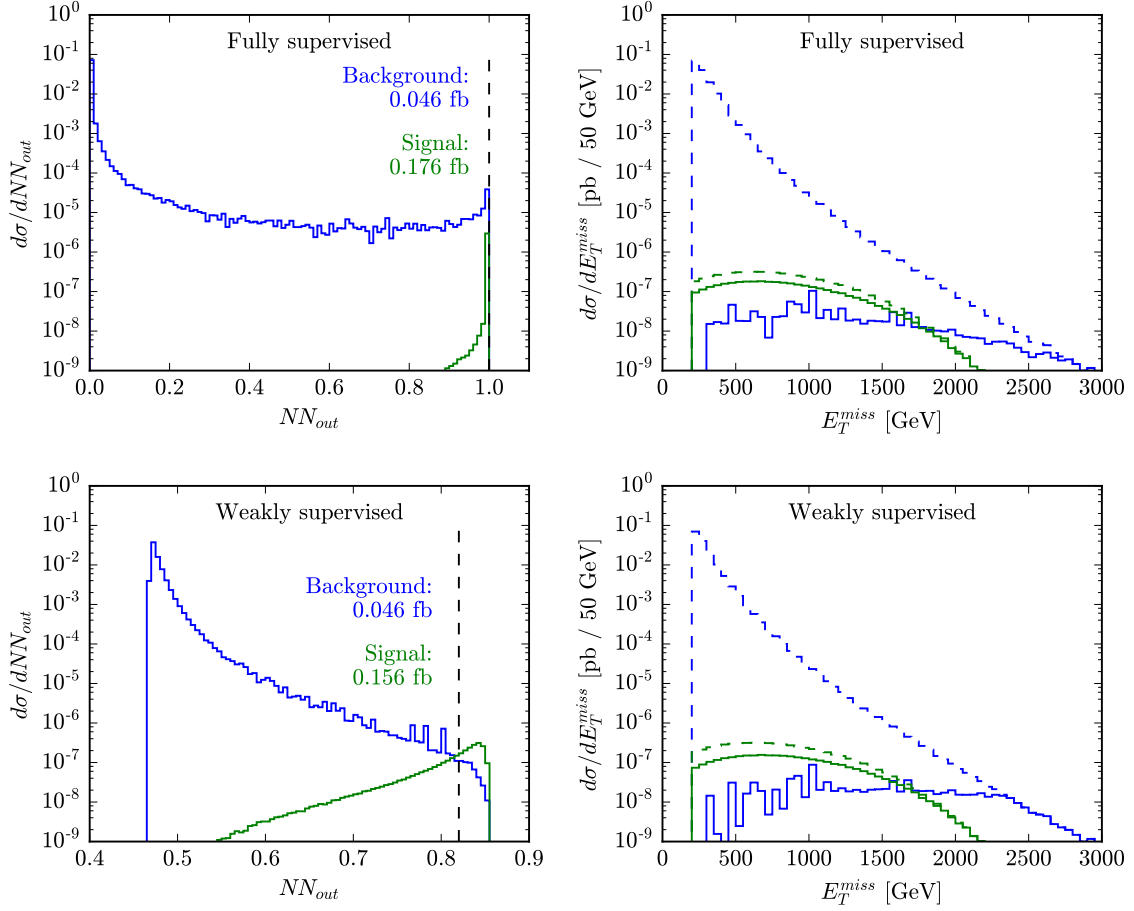
Figure 9: The left panels show the differential cross section as a function of the neural network outputs for the fully and weakly supervised networks on the top and bottom, respectively. The dashed vertical lines show where the cut is placed, while the labels mark the signal and background cross sections passing the cut. The cut was chosen such that both networks yield the same number of background events. Note that the top left plot confusingly never shows $S/B > 1$ due to a finite bin size artifact. The right panels show the effect of the cut on the missing energy spectrum. The dashed (solid) lines show the distributions before (after) the cut is made. Both of the neural networks are able to reject the dominant background at low values of missing energy, while keeping most of the similar signal events.

the background in the original dataset has a maximum missing energy of about 1 TeV. However, if we were to simulate the same integrated luminosity for the signal and background, the background could have missing energy as large as 3 TeV.

To account for this, we further generated another sample of backgrounds using generator level bins of missing energy to get non-trivial statistics for rare events. The new events contain much larger missing energy and jet momenta for the background

events than were realized in any of the training examples. Figure 9 shows the resulting predictions for these events from the two networks, weighted by their cross section. The networks again behave similarly, and can effectively separate the signal from the background. This is very impressive as the background events are probing different regions of phase space than were spanned in the training set.

As a final probe, we next make even more stringent cuts on the neural network outputs (while maintaining that the background cross section making it through each network is the same). Signal cross sections of 0.176 and 0.156 fb can be kept for the fully and weakly supervised networks, with 0.046 fb of background. The right panels of Fig. 9 show the differential cross sections as a function of the missing energy. The dashed lines represent the distributions before making the cuts, while the solid lines are after the cut is made. We see that both networks gain their separating power by cutting out background with relatively small amounts of missing energy.

5.2 Mismodeling

Armed with these concrete comparisons between weakly and fully supervised networks, we will explore a concrete example of mismodeling that would lead to a change in the fraction labels provided at the training step. In particular, we will see that for the class of mismodeling effects we study here, the performance of a fully supervised network degrades, while the weakly supervised networks remain robust.
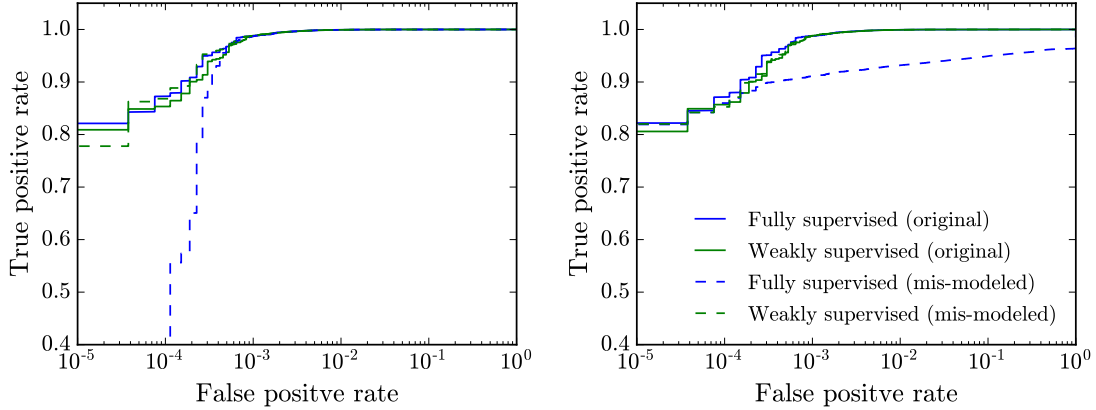
Figure 10: ROC curves showing the response of the network to mismodeled data. The mismodeling in the left panel shows the results of taking a random 15 % of the signal events and labeling them as background before training. The right panel demonstrates what happens under a phase space swap, where we mislabel the 10 % most signal-like background event and the 15 % most background-like signal events. The fully supervised network trained on the mislabeled data performs much worse at small false positive rates than when the data is not mislabeled. The weakly supervised network does not rely on individual event labels, and therefore shows no loss in performance.

	f_t label		
Dataset	Original	Random 15%	Phase space swap
A	0.472	0.374 (0.585)	0.416 (0.593)
B	0.843	0.782 (0.769)	0.810 (0.747)

Table 4: Labels for the fractions of the two datasets used when the data is mismodeled. The numbers in parenthesis show the true fraction contained in the dataset.

In order to mock up the effects of this mismodeling, we take the original set of training and validation events and use the fully supervised network to classify them. Two tests are then performed and their results are presented in Fig. 10. In the first, 15 % of the signal events are chosen at random and artificially mislabeled as background (left panel). In the second, we perform a *phase space swap*. Specifically, we change the labels between the most-signal like 10 % of the background events and the most background-like 15 % of the signal events (right panel). In effect, this changes the underlying missing energy and jet momentum distributions for the training samples and leads to mislabeling of the fractions used in the weakly supervised networks. The events are then split into subgroups as was done above to make samples with different ratios, where the ratio is now calculated based on the updated mismodeled labels. A new weakly supervised network is trained on these misclassified events as well by reporting a new incorrect signal fraction corresponding to the number of signal events flipped for the fully supervised case. Table 4 shows the values of the fractions used in the trainings, as well as the fractions that are actually present in the datasets.

The ROC curves for the new networks when tested on the true (not altered) labels are shown in Fig. 10. The new fully supervised network shows a distinct drop in performance when trained on the mismodeled data. When the data is mismodeled, the area under the ROC curve is effectively unchanged for the 15 % mislabeling test, and are 4.1 % smaller for the phase space swap test. Even though the AUC does not change in the left panel, the true positive rate is impacted. However, the weakly supervised network shows no change in performance because it does not rely on individual event labels. Weakly supervised networks can achieve better results than fully supervised networks even in the presence of systematic errors that would lead to this kind of effect. This result, along with the robustness shown in Fig. 5, motivate the use of weakly supervised networks at the LHC.

6   Discussion and Future Directions

In this paper, we have studied the robustness of weakly supervised neural networks as proposed by DNRS. We confirmed that weakly supervised networks are essentially as performant as fully supervised networks. We further demonstrated that weakly supervised networks are extremely robust to systematic mismodeling effects, and provided an analytic argument to explain this unexpected feature. To our knowledge, our work is the first application of these kinds of networks to beyond the Standard Model collider search scenarios.

There are many future directions left to explore. For example, we provided an oversimplified background sample by only simulating the dominant process, but in practice one should train using a more realistic background involving all relevant Standard Model processes. We did some very simple explorations of the systematics of mismodeling, and of applying a network to a signal that is not identical to the one used in training, but there is clearly much to continue to explore. We also did not explore changing the parameters of the networks, for example one could tune hyper-parameters, one could use weight decay [53], and so on; a full optimization study should also be performed.

It would also be interesting to explore applications to subtle signatures of new physics. Data from the LHC is used to tune Monte Carlo event generators, which are subsequently used to simulate backgrounds for the LHC. It is therefore conceivable that a marginal signal of new physics from previously recorded data could be absorbed by this procedure. This would imply that Standard Model distributions would be distorted by the inclusion of some BSM signal. Examples where this could occur are models with quirks or hidden valleys, where the BSM physics might end up lurking in very soft hadronic data or a complex hadronic final state [54–57]. It is plausible that weak supervision could be used to tease out these difficult signals without an over-reliance on simulations.

We will conclude with one final observation. Although this paper was largely devoted to pitting them against each other, it turns out that fully and weakly supervised networks are complementary. While the performance of both networks has been shown to be very similar, on an event-by-event basis the two networks are relying on different events for maximal discrimination power. This implies that even better classification is possible using the combined output from both types of networks.

To demonstrate this, Fig. 11 provides a 2D histogram for the weighted signal (green) and background (purple) events. The output of the weakly supervised network for each event lies along the horizontal axis. The fully supervised network places the signal events very close to 1. Therefore, we transform the output of the fully supervised network by taking $\log(2-\text{output})$, so that events close to 1 are given by a small number, and events close to zero are mapped to $\log 2$. The outputs are then spread further by taking the $\log_{10}$, as shown on the vertical axis. The dashed
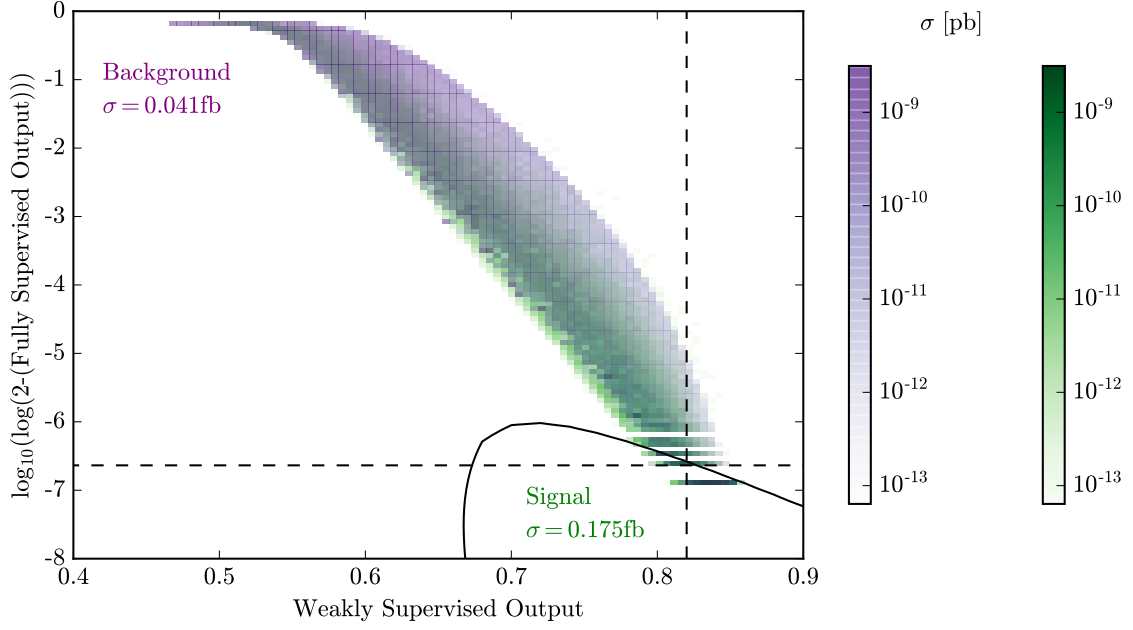
Figure 11: Two-dimensional histogram of the weighted Z+jets background and gluino pair production signal events. The horizontal axis corresponds to the output of the weakly supervised network, with the background peaked towards values of 0.4 and signal peaked towards 0.9. The vertical axis is the output for each event from the fully supervised network, transformed to separate the values close to 1. The larger, negative values correspond to signal, while close to 0 is the background. The two dashed lines show the values of the cuts places on either individual network. The solid line represents the cut found by training a fully supervised network which uses only these two axes as input. This cut improves the signal to background ratio by 11.6 % from the best cut on either network alone.

lines show the (transformed) cuts used at the end of Section 5.1, with the signal region being to the right or bottom for the weakly and fully supervised networks, respectively.

A new, fully supervised network with 30 nodes on the hidden layer is trained only using the outputs of the previous networks as input. The network is trained using the ADAM optimizer with a learning rate of 0.001 for 100 epochs using all of the weighted background and signal events. The events were first re-weighted so that the summed weights for all of the background is only 5 times that of summed signal events. With this network, we then choose a value of cut which gives a similar background cross section as used before. The resulting signal cross section after the cut is 0.175 fb and the background cross section is 0.041 fb. This is an 11.6 % improvement in the signal to background ratio compared to the fully supervised network alone. Furthermore, it appears that the lower edge of the signal blob also does not contain background, so one could add in these extra regions by hand for an additional improvement.

While the idea of ensembleing or stacking machine learning models together to boost performance is not new [58–60], this example shows that it can be done simply using both fully and weakly supervised networks.

Machine learning as applied to LHC is still in a rudimentary phase. As these tools become more relevant, it is crucial that systematic studies like the one presented here are done. Clearly weakly supervised neural networks are extremely powerful, and we look forward to seeing their application to a variety of physical processes in the future.

Acknowledgments

References

[1] **ATLAS** Collaboration, G. Aad *et al.*, "Performance of *b*-Jet Identification in the ATLAS Experiment," *JINST* **11** (2016) no. 04, P04008, `arXiv:1512.01094 [hep-ex]`.

[2] **CMS** Collaboration, C. Collaboration, "Identification of b quark jets at the CMS Experiment in the LHC Run 2,".

[3] "Performance and Calibration of the JetFitterCharm Algorithm for c-Jet Identification," Tech. Rep. ATL-PHYS-PUB-2015-001, CERN, Geneva, Jan, 2015. `http://cds.cern.ch/record/1980463`.

[4] **CMS** Collaboration, C. Collaboration, "Identification of c-quark jets at the CMS experiment,".

[5] J. Cogan, M. Kagan, E. Strauss, and A. Schwarztman, "Jet-Images: Computer Vision Inspired Techniques for Jet Tagging," *JHEP* **02** (2015) 118, `arXiv:1407.5675 [hep-ph]`.

[6] L. G. Almeida, M. Backovi, M. Cliche, S. J. Lee, and M. Perelstein, "Playing Tag with ANN: Boosted Top Identification with Pattern Recognition," *JHEP* **07** (2015) 086, `arXiv:1501.05968 [hep-ph]`.

[7] L. de Oliveira, M. Kagan, L. Mackey, B. Nachman, and A. Schwartzman, "Jet-images deep learning edition," *JHEP* **07** (2016) 069, `arXiv:1511.05190 [hep-ph]`.

[8] P. Baldi, K. Bauer, C. Eng, P. Sadowski, and D. Whiteson, "Jet Substructure Classification in High-Energy Physics with Deep Neural Networks," *Phys. Rev.* **D93** (2016) no. 9, 094034, `arXiv:1603.09349 [hep-ex]`.

[9] D. Guest, J. Collado, P. Baldi, S.-C. Hsu, G. Urban, and D. Whiteson, "Jet Flavor Classification in High-Energy Physics with Deep Neural Networks," *Phys. Rev.* **D94** (2016) no. 11, 112002, `arXiv:1607.08633 [hep-ex]`.

[10] K. Datta and A. Larkoski, "How Much Information is in a Jet?," `arXiv:1704.08249 [hep-ph]`.

[11] C. Shimmin, P. Sadowski, P. Baldi, E. Weik, D. Whiteson, E. Goul, and A. Sgaard, "Decorrelated Jet Substructure Tagging using Adversarial Neural Networks," `arXiv:1703.03507 [hep-ex]`.

[12] K. Cranmer and R. S. Bowman, "PhysicsGP: A Genetic Programming Approach to Event Selection," *Comput. Phys. Commun.* **167** (2005) 165–176, `arXiv:physics/0402030 [physics.data-an]`.

[13] S. Whiteson and D. Whiteson, "Machine learning for event selection in high energy physics," *Engineering Applications of Artificial Intelligence* **22** (2009) no. 8, 1203 – 1217. http://www.sciencedirect.com/science/article/pii/S0952197609000827.

[14] P. Baldi, P. Sadowski, and D. Whiteson, "Searching for Exotic Particles in High-Energy Physics with Deep Learning," *Nature Commun.* **5** (2014) 4308, `arXiv:1402.4735 [hep-ph]`.

[15] J. Searcy, L. Huang, M.-A. Pleier, and J. Zhu, "Determination of the WW polarization fractions in $pp \to W^{\pm}W^{\pm}jj$ using a deep machine learning technique," *Phys. Rev.* **D93** (2016) no. 9, 094033, `arXiv:1510.01691 [hep-ph]`.

[16] P. Baldi, K. Cranmer, T. Faucett, P. Sadowski, and D. Whiteson, "Parameterized neural networks for high-energy physics," *Eur. Phys. J.* **C76** (2016) no. 5, 235, `arXiv:1601.07913 [hep-ex]`.

[17] P. T. Komiske, E. M. Metodiev, and M. D. Schwartz, "Deep learning in color: towards automated quark/gluon jet discrimination," *JHEP* **01** (2017) 110, `arXiv:1612.01551 [hep-ph]`.

[18] J. Barnard, E. N. Dawe, M. J. Dolan, and N. Rajcic, "Parton Shower Uncertainties in Jet Substructure Analyses with Deep Neural Networks," *Phys. Rev.* **D95** (2017) no. 1, 014018, `arXiv:1609.00607 [hep-ph]`.

[19] L.-G. Pang, K. Zhou, N. Su, H. Petersen, H. Stcker, and X.-N. Wang, "An EoS-meter of QCD transition from deep learning," `arXiv:1612.04262 [hep-ph]`.

[20] G. Kasieczka, T. Plehn, M. Russell, and T. Schell, "Deep-learning Top Taggers or The End of QCD?," *JHEP* **05** (2017) 006, `arXiv:1701.08784 [hep-ph]`.

[21] G. Louppe, M. Kagan, and K. Cranmer, "Learning to Pivot with Adversarial Networks," `arXiv:1611.01046 [stat.ME]`.

[22] L. de Oliveira, M. Paganini, and B. Nachman, "Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis," arXiv:1701.05927 [stat.ML].

[23] J. Pearkes, W. Fedorko, A. Lister, and C. Gay, "Jet Constituents for Deep Neural Network Based Top Quark Tagging," arXiv:1704.02124 [hep-ex].

[24] G. Louppe, K. Cho, C. Becot, and K. Cranmer, "QCD-Aware Recursive Neural Networks for Jet Physics," arXiv:1702.00748 [hep-ph].

[25] B. T. Huffman, T. Russell, and J. Tseng, "Tagging b quarks without tracks using an Artificial Neural Network algorithm," arXiv:1701.06832 [hep-ex].

[26] Y.-H. He, "Deep-Learning the Landscape," arXiv:1706.02714 [hep-th].

[27] L. M. Dery, B. Nachman, F. Rubbo, and A. Schwartzman, "Weakly Supervised Classification in High Energy Physics," arXiv:1702.00414 [hep-ph].

[28] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Prez, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial Intelligence* **89** (1997) no. 1, 31 – 71. http://www.sciencedirect.com/science/article/pii/S0004370296000343.

[29] J. Amores, "Multiple instance classification: Review, taxonomy and comparative study," *Artificial Intelligence* **201** (2013) 81 – 105. http://www.sciencedirect.com/science/article/pii/S0004370213000581.

[30] S. D. Ellis, T. S. Roy, and J. Scholtz, "Jets and Photons," *Phys. Rev. Lett.* **110** (2013) no. 12, 122003, arXiv:1210.1855 [hep-ph].

[31] S. D. Ellis, T. S. Roy, and J. Scholtz, "Phenomenology of Photon-Jets," *Phys. Rev.* **D87** (2013) no. 1, 014015, arXiv:1210.3657 [hep-ph].

[32] T. Cohen, M. J. Dolan, S. El Hedri, J. Hirschauer, N. Tran, and A. Whitbeck, "Dissecting Jets and Missing Energy Searches Using n-body Extended Simplified Models," *JHEP* **08** (2016) 038, arXiv:1605.01416 [hep-ph].

[33] S. Iwamoto, G. Lee, Y. Shadmi, and Y. Weiss, "Tagging new physics with charm," arXiv:1703.05748 [hep-ph].

[34] G. Barello, S. Chang, C. A. Newby, and B. Ostdiek, "Don't be left in the dark: Improving LHC searches for dark photons using lepton-jet substructure," *Phys. Rev.* **D95** (2017) no. 5, 055007, arXiv:1612.00026 [hep-ph].

[35] A. Buckley, A. Shilton, and M. J. White, "Fast supersymmetry phenomenology at the Large Hadron Collider using machine learning techniques," *Comput. Phys. Commun.* **183** (2012) 960–970, arXiv:1106.4613 [hep-ph].

[36] N. Bornhauser and M. Drees, "Determination of the CMSSM Parameters using Neural Networks," *Phys. Rev.* **D88** (2013) 075016, arXiv:1307.3383 [hep-ph].

[37] S. Caron, J. S. Kim, K. Rolbiecki, R. Ruiz de Austri, and B. Stienen, "The BSM-AI project: SUSY-AIgeneralizing LHC limits on supersymmetry with machine learning," *Eur. Phys. J.* **C77** (2017) no. 4, 257, arXiv:1605.02797 [hep-ph].

[38] G. Bertone, M. P. Deisenroth, J. S. Kim, S. Liem, R. Ruiz de Austri, and M. Welling, "Accelerating the BSM interpretation of LHC data with machine learning," `arXiv:1611.02704 [hep-ph]`.

[39] P. Bechtle, S. Belkner, D. Dercks, M. Hamer, T. Keller, M. Krmer, B. Sarrazin, J. Schtte-Engel, and J. Tattersall, "SCYNet: Testing supersymmetric models at the LHC with neural networks," `arXiv:1703.01309 [hep-ph]`.

[40] F. Chollet, "Keras." `https://github.com/fchollet/keras`, 2015.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR* **abs/1412.6980** (2014) . `http://arxiv.org/abs/1412.6980`.

[42] J. MacQueen, "Some methods for classification and analysis of multivariate observations.." Proc. 5th Berkeley Symp. Math. Stat. Probab., Univ. Calif. 1965/66, 1, 281-297 (1967)., 1967.

[43] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory* **28** (March, 1982) 129–137.

[44] "Abstracts," *Biometrics* **21** (1965) no. 3, 761–777. `http://www.jstor.org/stable/2528559`.

[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research* **12** (2011) 2825–2830.

[46] **ATLAS Collaboration** Collaboration, "Search for squarks and gluinos in final states with jets and missing transverse momentum using 36 fb^1 of $\sqrt{s} = 13$ TeV pp collision data with the ATLAS detector," Tech. Rep. ATLAS-CONF-2017-022, CERN, Geneva, Apr, 2017. `https://cds.cern.ch/record/2258145`.

[47] **CMS** Collaboration, C. Collaboration, "Search for supersymmetry in multijet events with missing transverse momentum in proton-proton collisions at 13 TeV,".

[48] J. Alwall, 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," *JHEP* **07** (2014) 079, `arXiv:1405.0301 [hep-ph]`.

[49] T. Sjostrand, S. Mrenna, and P. Z. Skands, "PYTHIA 6.4 Physics and Manual," *JHEP* **05** (2006) 026, `arXiv:hep-ph/0603175 [hep-ph]`.

[50] **DELPHES 3** Collaboration, J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lematre, A. Mertens, and M. Selvaggi, "DELPHES 3, A modular framework for fast simulation of a generic collider experiment," *JHEP* **02** (2014) 057, `arXiv:1307.6346 [hep-ex]`.

[51] M. Cacciari, G. P. Salam, and G. Soyez, "FastJet User Manual," *Eur. Phys. J.* **C72** (2012) 1896, `arXiv:1111.6097 [hep-ph]`.

[52] M. Cacciari, G. P. Salam, and G. Soyez, "The Anti-k(t) jet clustering algorithm," *JHEP* **04** (2008) 063, `arXiv:0802.1189 [hep-ph]`.

[53] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Proceedings of the 4th International Conference on Neural Information Processing Systems*, NIPS'91, pp. 950–957. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991. `http://dl.acm.org/citation.cfm?id=2986916.2987033`.

[54] M. J. Strassler and K. M. Zurek, "Echoes of a hidden valley at hadron colliders," *Phys. Lett.* **B651** (2007) 374–379, `arXiv:hep-ph/0604261 [hep-ph]`.

[55] J. Kang and M. A. Luty, "Macroscopic Strings and 'Quirks' at Colliders," *JHEP* **11** (2009) 065, `arXiv:0805.4642 [hep-ph]`.

[56] R. Harnik and T. Wizansky, "Signals of New Physics in the Underlying Event," *Phys. Rev.* **D80** (2009) 075015, `arXiv:0810.3948 [hep-ph]`.

[57] S. Knapen, S. Pagan Griso, M. Papucci, and D. J. Robinson, "Triggering Soft Bombs at the LHC," `arXiv:1612.00850 [hep-ph]`.

[58] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and Systems Magazine* **6** (Third, 2006) 21–45.

[59] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review* **33** (2010) no. 1, 1–39. `http://dx.doi.org/10.1007/s10462-009-9124-7`.

[60] R. Maclin and D. W. Opitz, "Popular ensemble methods: An empirical study," *CoRR* **abs/1106.0257** (2011) . `http://arxiv.org/abs/1106.0257`.