# Fast multilabel classification of HEP constraints with deep learning

Maien Binjonaid*

*Department of Physics and Astronomy, King Saud University, Riyadh, Saudi Arabia*

(Received 15 September 2024; accepted 3 March 2025; published 10 April 2025)

The shortcomings of the Standard Model (SM) motivate its extension to accommodate new expected phenomena, such as dark matter and neutrino masses. However, such extensions are generally more complex due to the presence of a large number of free parameters and additional phenomenology. Understanding how theoretical and experimental limits affect the parameter spaces of new models, individually and collectively, is of utmost importance for conducting model status analysis, motivating precise computations, or model building aimed at solving certain issues. However, checking the constraints usually requires a large amount of time using a chain of physics tools. We demonstrate, for the first time, the application of deep learning for the multilabel classification of a group of theoretical and experimental constraints in the dark doublet phase of the next-to two-Higgs-doublet model, as a representative nine-dimensional parameter space. We analyze the issue of class imbalance and the ability of the classifier to learn joint class distributions. We demonstrate the time advantage compared to physics tools, with the classifier achieving orders of magnitude faster checks on groups of constraints and strong performance. The classifier performed strongly in terms of identifying regions where all constraints are valid or invalid, as well as regions where one or more of the constraints are valid or invalid simultaneously. This approach can be applied to any extension beyond the SM with the potential to aid HEP tools or act as a surrogate for fast model status checks. To that end, we provide a Python tool HEPMLC for generating and investigating multilabel classifiers for SM extensions.

DOI: 10.1103/PhysRevD.111.075010

## I. INTRODUCTION

Machine learning (ML) and deep learning (DL), which are specific types of artificial intelligence (AI), have proven to be valuable tools in various research fields, including particle physics. These methods are particularly effective in tasks such as pattern recognition, anomaly detection, classification, and regression problems, often offering greater efficiency than traditional techniques [1,2]. AI has a wide range of applications in both theoretical and experimental studies of the Standard Model (SM) and its extensions, collectively called beyond the SM (BSM). These can be grouped into several areas, including signal-to-background analysis, jet tagging, parameter space scanning, learning properties and constraints, and predicting likelihoods, among many other applications (for example, [3–17]). A comprehensive review that is constantly updated can be found in Ref. [18].

As is well known, BSM extensions aim to address issues not explained by the SM, such as the origin of neutrino masses, the strong *CP* problem, the existence of dark matter (DM), which may well be of particle nature, and the lightness of the SM Higgs particle [19]. Models in this context often involve complex, multidimensional parameter spaces with numerous free parameters and new particles. In utilizing AI, some studies have focused on BSM signal-to-background analysis [20–23], flavor phenomenology [16], Higgsino jets [24], and R-parity violating supersymmetry [25].

The applications of AI in the process of scanning and analyzing constraints are progressing, but many aspects are still at an exploration phase as lessons are being learned about different methods. As for analyzing and classifying the properties of parameter spaces using AI, Ref. [26] presented one of the earliest uses of AI to classify parameter points in the minimal supersymmetric model (MSSM) as valid or invalid, based on ATLAS data, using a random forest (RF) classifier. This was followed by an active learning network [27] that consistently outperformed RF classifiers and significantly improved scanning times compared with random sampling. In Ref. [28], a heuristic search was combined with a generative adversarial network to classify the validity of parameter points in the next-to-minimal supersymmetric SM (NMSSM). A generative

*Contact author: maien@ksu.edu.sa

*Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI. Funded by SCOAP³.*

normalization flow method was used in Ref. [29] to explore the MSSM, only taking into account the mass of the SM Higgs boson as a constraint.

In Ref. [30], a trained neural network was proposed to predict higher likelihood points for a nested sampling algorithm, and applied to a toy model. Reference [31] introduced a criterion for scanning parameter spaces using active learning, which was implemented in Ref. [32]. A *Mathematica* tool was presented in Ref. [33] to set limits on the free parameters of a BSM models, and produce benchmark points using ML. Moreover, the work presented in Ref. [34] introduced a sampling technique that utilizes ML black-box optimization and applied it to the MSSM, potentially improving scanning efficiency by orders of magnitude, although only the mass of the SM Higgs boson was taken as a constraint or in combination with the constraint from dark matter relic density. A scanning tool was introduced in [35] which enables active learning scans within the public package SARAH [36].

However, most classification studies in the literature focus on single-label classification (SLC), where the target output is a binary class (i.e., valid or invalid). Specifically, when applying AI in exploring parameter spaces of BSM extensions, using specific physics tools/packages, to validate inputs against constraints, a single-label classifier would learn if the set of input parameters lead to an overall-valid/invalid result. That is, the physics tool determined that all constraints were passed (valid result) or that at least one constraint is not satisfied (invalid result). Therefore, the resulting trained AI classifier has no information regarding the status of the underlying constraints (they were hidden from the classifier as the decision of overall validity/invalidity was done by the physics tool and only the final decision is passed to the AI single-label classifier). While this approach has its own merit, and can be informative and efficient in aiding parameter space scans targeting only valid points, the nature of SLC is clearly restrictive since it cannot learn and provide insights into how each constraint (or category of constraints) affects the outcome or how different constraints might interact.

Understanding these aspects, regardless of AI, has been an integral part of particle phenomenology in studies that aim to systematically analyze the effects of constraints on a given model (i.e., status of the model), motivate or investigate effects of precise computations (e.g., higher-order corrections) or extensions that would open or close certain regions (see, e.g., [37–68]). These constraints can be theoretical or experimental. A well-known set of theoretical constraints are vacuum stability, triviality, and unitarity, which were essential in pinpointing the allowed range of the SM Higgs mass prior to its discovery [69]. Such constraints become more complicated in extensions of the SM, and are essential to take into account. Experimental constraints include electroweak precision tests, Higgs properties (e.g., couplings to SM particles), DM observations (if the model include a DM candidate), and flavor physics (e.g., B physics). All of these constraints and others that might be model specific (e.g., related to supersymmetry) must be considered. In BSM extensions, the relevant computations are based on the set of input parameters of a given model, and involves sophisticated calculations of every theoretical and observable aspect. This prolonged process has been advanced by a number of numerical tools and packages.[1] One example, which we utilize in this work, is ScannerS [71], enabling phenomenological analyses of extensions of the SM with doublets and singlets. The package includes routines to check theoretical and experimental constraints, and interfaces with N2HDECAY [72] (Higgs properties), micrOMEGAs [73] (DM observables), HiggsBounds [74], HiggsSignals [75] (Higgs constraints), and EVADE [76] (vacuum structure).

Consequently, when it comes to the potential application of AI for detailed analyses of constraints in extensions of the SM, and given that SLC is not suitable for this task, we are led to multilabel classification (MLC) [77–80], which has not been explored yet in this context. MLC addresses problems in which a set of inputs (features) is associated with multiple outputs (target labels), each of which can be binary or multiclass. Applying MLC to the study of parameter constraints means that the AI classifier learns the decision of the physics tools on the validity of each constraint (or category of constraints) individually, but it will also implicitly learn each combination of constraints (or category of constraints) collectively (i.e., the joint distribution of constraints/labels). So while the physics tools are concerned with applying individual constraints given a set of inputs, the classifier is learning patterns between the inputs associated with the joint distribution of constraints. This means that once a MLC is trained on a BSM extension, and depending on the required details and performance, it can potentially play the role of the chain of physics tools in validating the model and providing the insights that are usually provided by such tools (i.e., acting as a surrogate). The potential of such a method can be judged by achieving reasonable accuracy and time advantage.

MLC is significantly more involved than binary SLC, and it brings about new challenges such as dealing with class imbalance between combinations of labels. Hence, it is essential to test and validate its application within the context of a realistic BSM parameter space. It is then the goal of this study to develop a DL model for the MLC of parameter space constraints, focusing on learning their validity and invalidity, and demonstrating the feasibility of this method. We select four categories of constraints representing theoretical and experimental limits: namely,

[1]An extensive review of model building and the issues surrounding it is provided in [70].

boundedness from below ($BfB$), perturbative unitarity (PU), oblique parameters (STU), and bounds on SM-like and non-SM Higgs (Higgs). The classifier should be able to correctly predict the outcomes of individual constraints and any possible combinations. The success of the method is to be determined by appropriate performance-measuring metrics, the major one being the subset accuracy, which is a strict metric that considers an instance as correctly classified only if all labels are correctly identified at the same time.

To apply this method, we consider, as a representative case, the dark doublet phase of the next-to two-Higgs-doublet model (DDP-N2HDM). The model consists of nine free parameters, creating a multidimensional parameter space that can be challenging to analyze, making it a suitable case for applying AI techniques. Moreover, the model consists of two Higgs doublets and a real singlet. One doublet remains inert [i.e., does not acquire a vacuum expectation value (VEV)], thereby providing a candidate for DM that is stabilized through an unbroken discrete symmetry. The particle content of this model comprises two $CP$-even Higgs bosons (one of which is SM like), and a dark sector containing $CP$-even, $CP$-odd, and charged scalars, where the lightest neutral scalar represents the DM candidate. This model is interesting since its DM sector resembles that of the inert doublet model [81–83], while its Higgs sector resembles that of type 1 2HDM. However, the presence of a singlet component provides a distinctive feature to its Higgs bosons. The model has a rich phenomenology, which has been studied in several works (see, e.g., [84–91] and references therein).

To facilitate future work on other models and studies dedicated to further advance this application (e.g., optimization studies for MLC, generate multilabel classifiers for commonly studied models), we provide a Python tool HEPMLC that enables users to build multilabel classifiers using ML and DL for models within the public package ScannerS as a first stage, or for any labeled dataset provided by the user.

The paper is divided as follows: Section II introduces the N2HDM and details the constraints considered in this study. Section III provides a brief overview of MLC. Section IV covers the data generation using the physics tool, the AI methodology, and the DL architecture. Section V presents the results, followed by discussion and conclusions in Sec. VI. Finally, we describe the Python tool HEPMLC in the Appendix.

## II. THE PHYSICS MODEL AND CONSTRAINTS

### A. The N2HDM

We start with a brief overview of the N2HDM. The scalar potential is given by (following the notation in Ref. [83])

$$\begin{aligned} V = {} & m_{11}^2\Phi_1^\dagger\Phi_1 + m_{22}^2\Phi_2^\dagger\Phi_2 + \frac{\lambda_1}{2}\left(\Phi_1^\dagger\Phi_1\right)^2 + \frac{\lambda_2}{2}\left(\Phi_2^\dagger\Phi_2\right)^2 \\ & + \lambda_3\Phi_1^\dagger\Phi_1\Phi_2^\dagger\Phi_2 + \lambda_4\Phi_1^\dagger\Phi_2\Phi_2^\dagger\Phi_1 \\ & + \frac{\lambda_5}{2}\left[\left(\Phi_1^\dagger\Phi_2\right)^2 + \text{H.c.}\right] + \frac{1}{2}m_s^2\Phi_S^2 + \frac{\lambda_6}{8}\Phi_S^4 \\ & + \frac{\lambda_7}{2}\Phi_1^\dagger\Phi_1\Phi_S^2 + \frac{\lambda_8}{2}\Phi_2^\dagger\Phi_2\Phi_S^2, \end{aligned} \tag{1}$$

where $\Phi_i$ ($i = 1, 2$) represents scalar doublets, while the singlet field is denoted by $\Phi_S$. The potential has real parameters (mass terms and quartic couplings $\lambda_i$), and exact discrete symmetries; in particular, a $\mathbb{Z}_2^{(1)}$ symmetry under which all fields are even except for $\Phi_2$, and a $\mathbb{Z}_2^{(2)}$ symmetry under which all fields are even except for $\Phi_S$.

The VEVs of the fields are given by

$$\langle\Phi_1\rangle = \begin{pmatrix} 0 \\ \frac{v_1}{\sqrt{2}} \end{pmatrix}, \quad \langle\Phi_2\rangle = \begin{pmatrix} 0 \\ \frac{v_2}{\sqrt{2}} \end{pmatrix}, \quad \langle\Phi_S\rangle = v_s, \tag{2}$$

and the fields can be parametrized as follows:

$$\begin{aligned} \Phi_1 &= \begin{pmatrix} \phi_1^+ \\ \frac{1}{\sqrt{2}}(v_1 + \rho_1 + i\eta_1) \end{pmatrix}, \\ \Phi_2 &= \begin{pmatrix} \phi_2^+ \\ \frac{1}{\sqrt{2}}(v_2 + \rho_2 + i\eta_2) \end{pmatrix}, \quad \Phi_S = v_s + \rho_s, \end{aligned} \tag{3}$$

where $\phi_{1,2}^+$ are charged complex gauge eigenstates, $\rho_{1,2,s}$ are neutral $CP$-even gauge eigenstates, and $\eta_{1,2}$ are neutral $CP$-odd gauge eigenstates.

The minimization conditions of the potential lead to

$$m_{11}^2 = -\frac{1}{2}(v_1^2\lambda_1 + v_2^2(\lambda_3 + \lambda_4 + \lambda_5) + v_s^2\lambda_7) \tag{4}$$

$$m_{22}^2 = -\frac{1}{2}(v_1^2(\lambda_3 + \lambda_4 + \lambda_5) + v_2^2\lambda_2 + v_s^2\lambda_8) \tag{5}$$

$$m_s^2 = -\frac{1}{2}(v_1^2\lambda_7 + v_2^2\lambda_8 + v_s^2\lambda_6). \tag{6}$$

An interesting variant of the model is the DDP-N2HDM, which is defined as the case where only $\Phi_1$ and $\Phi_S$ acquire VEVs ($v = 246$ GeV and $v_s$), while $\Phi_2$ remains inert (setting $v_2 = 0$ in the previous equations for the general N2HDM). This leads to a DM candidate coming from the inert doublet. The stability of the DM particle requires the preservation of the $\mathbb{Z}_2^{(1)}$ symmetry, while the other symmetry $\mathbb{Z}_2^{(2)}$ is broken by $\Phi_S$ when it obtains a nonzero VEV. Moreover, the mass eigenstates of the $CP$-even neutral Higgs bosons ($m_{H_1}$ and $m_{H_2}$) are derived through rotating the gauge eigenstates by a $3 \times 3$ matrix $R$ that depends on

the rotation angle $\alpha$ and whose components take the values $R_{11} = R_{23} = \cos\alpha$, $R_{13} = -R_{21} = \sin\alpha$, $R_{32} = 1$, while the other components are zero. The $CP$-odd and charged gauge eigenstates of the two doublets do not mix. Therefore, the mass eigenstates coming from the inert doublet are part of the dark sector, and hence denoted by $m_{H_D^\pm}$, $m_{A_D}$, and $m_{H_D}$ (the DM candidate).

The Yukawa Lagrangian is taken to be of type I in order to allow for comparison of different phases of the N2HDM. For the DDP of the N2HDM it is given as follows:

$$\mathcal{L}_{\text{Yukawa}} = -\bar{Q}_L^T Y_U \tilde{\Phi}_1 U_R - \bar{Q}_L^T Y_D \Phi_1 D_R - \bar{L}_L^T Y_L \Phi_1 E_R + \text{H.c.}, \tag{7}$$

where $Y$ are Yukawa coupling matricese, $Q_L$ and $L_L$ are doublets containing the left-handed fermions, while $U_R$, $D_R$, and $E_R$ encompass right-handed fermions.

The quartic couplings $\lambda_i$ can be expressed in terms of scalar masses, mixing angles, and VEVs [83]:

$$\lambda_1 = \frac{1}{v^2}\left(m_{H_1}^2 R_{11}^2 + m_{H_2}^2 R_{21}^2\right), \tag{8}$$

$$\lambda_3 = \frac{1}{v^2}\left(2m_{H_D^\pm}^2 - 2m_{22}^2 - v_s^2\lambda_8\right), \tag{9}$$

$$\lambda_4 = \frac{1}{v^2}\left(m_{A_D}^2 + m_{H_D}^2 - 2m_{H_D^\pm}^2\right), \tag{10}$$

$$\lambda_5 = \frac{1}{v^2}\left(m_{H_D}^2 - m_{A_D}^2\right), \tag{11}$$

$$\lambda_6 = \frac{1}{v_s^2}\left(m_{H_1}^2 R_{13}^2 + m_{H_2}^2 R_{23}^2\right), \tag{12}$$

$$\lambda_7 = \frac{1}{v v_s}\left(m_{H_1}^2 R_{11} R_{13} + m_{H_2}^2 R_{21} R_{23}\right), \tag{13}$$

where $R_{ij}$ are elements of the mixing matrix.

Finally, using the minimization conditions and the relations in Eq. (13), the model can be specified by the following input parameters: $m_{H_1}, m_{H_2}, m_{H_D}, m_{A_D}, m_{H_D^\pm}$, $m_{22}^2$, $v_s$, $\lambda_2$, $\lambda_8$, $\alpha$.

## B. The constraints

### 1. Boundedness from below

A crucial condition for a stable vacuum, meaning that the global minimum is associated with the electroweak symmetry breaking vacuum, is that the scalar potential must be bounded from below ($BfB$). The potential must remain positive in the limit where the fields reach infinity. The potential is $BfB$ if one of the following conditions is satisfied [86,92]:

(i) The first set of conditions is

$$\lambda_1 > 0, \quad \lambda_2 > 0, \quad \lambda_6 > 0,$$
$$\sqrt{\lambda_1\lambda_6} + \lambda_7 > 0,$$
$$\sqrt{\lambda_2\lambda_6} + \lambda_8 > 0,$$
$$\sqrt{\lambda_1\lambda_2} + \lambda_3 + D > 0,$$
$$\lambda_7 + \sqrt{\frac{\lambda_1}{\lambda_2}}\lambda_8 \geq 0,$$

where $D = \lambda_4 - \lambda_5$ if $\lambda_4 > \lambda_5$ and zero otherwise.

(ii) The second set of conditions is

$$\lambda_1 > 0, \quad \lambda_2 > 0, \quad \lambda_6 > 0,$$
$$\sqrt{\lambda_1\lambda_6} > -\lambda_7 \geq \sqrt{\frac{\lambda_1}{\lambda_2}}\lambda_8,$$
$$\sqrt{\lambda_2\lambda_6} \geq \lambda_8 > -\sqrt{\lambda_2\lambda_6},$$
$$\sqrt{(\lambda_7^2 - \lambda_1\lambda_6)(\lambda_8^2 - \lambda_2\lambda_6)} > \lambda_7\lambda_8 - (D + \lambda_3)\lambda_6.$$

### 2. Perturbative unitarity

To ensure tree-level perturbative unitarity (PU) of the model, one imposes an upper limit of $8\pi$ [93] on the largest eigenvalue of the tree level $2 \to 2$ scattering matrix $\mathcal{M}_{2\to2}$. For the N2HDM, accounting for the contributions from the additional singlet field, the full expression was computed in [86], and the following conditions must apply:

$$\left.\begin{aligned} &|\lambda_3 - \lambda_4| \\ &|\lambda_3 + 2\lambda_4 \pm 3\lambda_5| \\ &\left|\tfrac{1}{2}\left(\lambda_1 + \lambda_2 + \sqrt{(\lambda_1 - \lambda_2)^2 + 4\lambda_4^2}\right)\right| \\ &\left|\tfrac{1}{2}(\lambda_1 + \lambda_2 + \sqrt{(\lambda_1 - \lambda_2)^2 + 4\lambda_5^2})\right| \\ &|\lambda_7|, |\lambda_8| \\ &\tfrac{1}{2}|a_{1,2,3}| \end{aligned}\right\} < 8\pi. \tag{14}$$

where $a_{1,2,3}$ are given by [86]

$$a_1 = 4(-27\lambda_1\lambda_2\lambda_6 + 12\lambda_3^2\lambda_6 + 12\lambda_3\lambda_4\lambda_6 + 3\lambda_4^2\lambda_6 + 6\lambda_2\lambda_7^2 - 8\lambda_3\lambda_7\lambda_8 - 4\lambda_4\lambda_7\lambda_8 + 6\lambda_1\lambda_8^2), \tag{15}$$

$$a_2 = 36\lambda_1\lambda_2 - 16\lambda_3^2 - 16\lambda_3\lambda_4 - 4\lambda_4^2 + 18\lambda_1\lambda_6 + 18\lambda_2\lambda_6 - 4\lambda_7^2 - 4\lambda_8^2, \tag{16}$$

$$a_3 = -6(\lambda_1 + \lambda_2) - 3\lambda_6. \tag{17}$$

TABLE I. Values of the oblique parameters $S$, $T$, and $U$ with corresponding errors and correlation coefficients.

| Parameter | Value | Error | Correlation coefficient |
|---|---|---|---|
| $S$ | 0.04 | ±0.11 | +0.92 (S and T) |
| $T$ | 0.09 | ±0.14 | |
| $U$ | −0.02 | ±0.11 | −0.68 (S and U), −0.87 (T and U) |

#### 3. Oblique parameters

Weak interaction observables can be indirectly affected by corrections to self-energies of gauge bosons arising from new physics. These indirect effects can be parametrized via the so-called oblique parameters: $S$, $T$, and $U$ (referred to as STU) [94]. It is possible to constrain these parameters via precision electroweak measurements. For multi-Higgs doublet and singlet models, the oblique parameters can be computed as shown in Refs. [95,96]. Their values, fitted in Ref. [97], are listed in Table I.

#### 4. Higgs constraints

The Higgs sector of the model is constrained by checking its compatibility with data from various collider experiments (for reviews, see Refs. [98,99]). The properties include Higgs couplings, branching ratios, production cross sections, and signal rates. After computing the predictions of the model for these quantities, a comparison with experimental results is performed using the tools specified in Sec. III. Namely, the branching ratios $BR(H_i \to X)$ of neutral and charged Higgs bosons $H_i$ into various SM particles $X$ are computed and checked against Higgs data. Additionally, the production cross sections of the Higgs bosons in various channels, such as gluon-gluon fusion ($gg \to H$), are calculated to check the resulting signal rates against the LHC observations. Finally, charged Higgs bosons are required to be heavier than 70 GeV [82,100].

## III. OVERVIEW OF MULTILABEL CLASSIFICATION

There are various types of classification problems. The most common is single-label classification, in which a set of features (e.g., the input parameters of a physics model) is associated with one target label, $L$. The label can be either binary (0 or 1) or multiclass (A, B, C, …). The aim of AI is to assign a class to each instance of features. This also applies to regression tasks. However, the goal of such tasks is to predict continuous or discrete numbers (range). In contrast, MLC is concerned with predicting more than one label given a set of features. Each label can be either binary (e.g., $L_1 \in \{0,1\}, L_2 \in \{0,1\}, \ldots$) or multiclass (e.g., $L_1 \in \{a,b,c\}, L_2 \in \{a,b,c\}, \ldots$). As for multioutput regression, the goal is to predict numerical values for each label.

It is possible to convert an MLC problem into a single-label multiclass problem. In this case, the classes of a single label represent each possible combination of classes of multiple labels. For example, $n$ binary labels can be cast as a single multiclass label: $L \in \{(L_1 = 1 \text{ and } L_2 = 1), (L_1 = 1 \text{ and } L_2 = 0), \ldots\}$, where the dots here represent the rest of possible $2^n$ combinations. However, it is important to note that this practice could complicate the problem as the number of labels grows, since one would have to deal with each combination of classes separately. Moreover, the focus of this single-label multiclass classifier would be on the possible combinations only. It will not focus on each label individually, which could be problematic if one of the goals is to learn each individual label separately. With that in mind, MLC has the advantage of not only learning each label individually, but also the combinations of labels collectively, capturing any interrelations among them. We believe that MLC can be a very useful approach for studying parameter constraints in BSM extensions, especially if the aim is to gain detailed insights into the physics model.

## IV. FROM DATA TO DEEP LEARNING

In this section, we describe the methods used for data generation, handling class imbalance, optimization and fine-tuning, DL model building, evaluation metrics, and the ML baseline model. These methods are the base on which our HEPMLC tool is created. The tool is described in the Appendix.

### A. Data generation and class imbalance

One of the most important (and often challenging) aspects of applying ML/DL is to generate good quality data, where there is no severe class imbalance. We use a hybrid method to sample the parameter space of the DDP-N2HDM. Specifically, a combination of Latin hypercube sampling (LHS) and random sampling was employed, and the final dataset comprises 774,262 points, ensuring good representation of the parameter space. In both sampling methods, multiple independent scans with different seeds were conducted to ensure that the samples did not cluster around specific regions. Furthermore, we restrict our analysis to the normal hierarchy variant where $m_{H_1} = 125.09$ GeV, while $m_{H_2} > m_{H_1} + 3$ GeV. The scans span the ranges listed in Table II.

For each generated point, ScannerS [71,72] is utilized to compute the spectrum of the model, which includes physical masses, couplings, and several predicted quantities such as branching ratios, cross sections, and quantities relevant to constraints, such as the maximum eigenvalue of the 2 by 2 S-matrix and the oblique parameters. A comprehensive check of theoretical and experimental constraints is carried out using internal routines and interfaced tools. In particular, the theoretical checks include $BfB$ and PU, while the experimental ones cover STU and *Higgs* constraints. The STU parameters are computed using

TABLE II. Parameter ranges and descriptions used for Latin hypercube sampling and random sampling.

| Parameter | Range | Description |
|---|---|---|
| $m_{H_1}$ | 125.09 GeV | Mass of the first Higgs boson (SM-like) |
| $m_{H_2}$ | 128 to 1500 GeV | Mass of the second Higgs boson |
| $m_{H_D}$ | 1 to 1500 GeV | Mass of the DM candidate |
| $m_{A_D}$ | 1 to 1500 GeV | Mass of the dark pseudoscalar |
| $m_{H_D^\pm}$ | 1 to 1500 GeV | Mass of the charged dark Higgs boson |
| $\alpha$ | $-1.57$ to 1.57 | Mixing angle |
| $m_{22}^2$ | $10^{-3}$ to $5 \times 10^5$ GeV$^2$ | Mass-squared parameter of $\Phi_2$ |
| $\lambda_2$ | 0 to 20 | Quartic coupling constant |
| $\lambda_8$ | $-30$ to 30 | Quartic coupling constant |
| $v_s$ | 1 to 1500 GeV | Vacuum expectation value of the singlet field |

TABLE III. Class imbalance for individual labels in the dataset.

| Label | Invalid (0) | Valid (1) | Imbalance ratio |
|---|---|---|---|
| *BfB* | 308,111 | 466,151 | 1.513 |
| PU | 389,253 | 385,009 | 0.989 |
| STU | 341,024 | 433,238 | 1.270 |
| Higgs | 328,299 | 445,963 | 1.358 |

the formulas mentioned in Sec. II B and subsequently checked in ScannerS against their experimentally fitted values given in [97]. On the other hand, the *Higgs* constraints are checked using HiggsBounds 5.10.2 [74] and HiggsSignals 2.6.2 [75].

The generated dataset contains the four aforementioned constraints as target labels, which are grouped into 16 different combinations (i.e., valid/invalid *BfB*, valid/invalid PU, valid/invalid STU, and valid/invalid *Higgs*).

The final dataset was accepted after we ensured that each label was not severely imbalanced, as shown in Table III. The valid/invalid ratio ranges from about 1 (PU) to 1.5 (*BfB*), which is not severe. Furthermore, Table IV shows the class imbalance in the joint class distribution, and we ensured that each case was represented by sufficient points to avoid severe joint class imbalance. This is a crucial step for MLC tasks, since significant imbalance in individual labels would negatively affect model training overall, while imbalance in the joint class distributions would negatively affect the ability of the DL model to learn the interplay between labels.

We addressed severe imbalance at the initial stages of data generation by performing targeted scans using the physics tool, thereby populating under-represented regions. By doing so, the DL model is exposed to a more comprehensive range of multilabel patterns, enabling it to learn the underlying joint distribution of constraints rather than only the most frequent patterns.

In the final dataset, a noticeable (not severe) class imbalance remains between the top (1_1_1_1) and bottom (0_1_1_0) joint classes, with a ratio of approximately 6:1. Also, since any combination containing at least one zero is overall invalid, we have a total of 152,974 valid cases and 621,288 invalid cases, corresponding to a 1:4 ratio. In practice, completely avoiding class imbalance is unrealistic for BSM models, as large regions of parameter space are expected to violate one or more constraints. We will address moderate class imbalance during training by selecting an appropriate loss function, which we will discuss later.

For completeness, we checked the vacuum structure using EVADE [76,87,101] and DM relic density (upper limit) and direct detection using micrOMEGAs 5.3.41 [73]. However, we do not consider these as target labels in training our DL model since we observe that they bring about severe class imbalance in the dataset (hindering the quality of the data), with the majority class being valid (i.e., both are valid in the majority of the parameter space). Handling this is possible, but would exponentially increase the number of regions that need to be populated from 16 (for four labels) to 64 (for six labels). This task is suitable for an analysis that utilizes high-performance computing and opens the door for exploring optimization techniques for fast sampling specific to MLC tasks, which we leave for a future work. Moreover, in the final dataset, valid regions with respect to the four target labels are also valid with respect to vacuum stability, and are almost entirely valid with respect to DM, expect for a very small region where $m_{\rm DM} \leq 100$ GeV. Since our focus is on creating a multilabel classifier for the four target labels specified previously, we will not discuss vacuum stability and DM any further.

In terms of data preprocessing, it is generally crucial to analyze and prepare the dataset carefully before passing it

TABLE IV. Class combinations and their counts in the dataset.

| *BfB*_PU_STU_Higgs (number of points in the dataset) | | | |
|---|---|---|---|
| 1_1_1_1 (152974) | 0_0_0_0 (91361) | 1_0_1_1 (75904) | 1_0_0_1 (46122) |
| 0_1_1_1 (40367) | 1_0_1_0 (39611) | 1_0_0_0 (39329) | 1_1_0_1 (39017) |
| 1_1_1_0 (37868) | 0_0_1_0 (35613) | 1_1_0_0 (35326) | 0_0_0_1 (34070) |
| 0_1_0_1 (30266) | 0_0_1_1 (27243) | 0_1_0_0 (25533) | 0_1_1_0 (23658) |

to the DL model. This includes an initial analysis of the distribution of input parameters (features) listed in Table II, and the class imbalance of the target label(s). In our case, we consider the standardization of features by applying (if necessary) a Yeo-Johnson (YJ) power transformation [102], which is suitable since some of the features take negative values (namely, $\alpha$ and $\lambda_8$), followed by scaling (i.e., applying z-score normalization). The data is then split into three disjoint sets for training (70%), validation (15%), and testing (15%). These tasks are carried out using SCIKIT-LEARN [103].

### B. DL model architecture and training

Creating a deep neural network model for MLC involves a number of steps, such as selecting appropriate values of hyperparameters, constructing the network architecture, and configuring the optimization process. In this work, the creation of the DL model is designed to support both static and dynamic hyperparameter setup as will be further explained shortly below. The construction is done using TensorFlow [104] and Keras [105]. A thorough experimentation phase is carried out by utilizing the Optuna framework [106], to obtain the best structure and values for the following:

(i) *Number of layers ($n_{\text{layers}}$)*. The number of dense layers in the network, chosen between 2 and 3.[2]
(ii) *Number of units per layer ($n_{\text{units},i}$)*. The number of neurons in each layer, with values ranging from 128 to 1024.
(iii) *Activation function*. The activation function used in each layer, either ReLU [107] or Leaky ReLU [108].
(iv) *Dropout rate*. The dropout rate [109] applied after each dense layer, ranging from 0.1 to 0.5.
(v) *Batch normalization*. A Boolean flag indicating whether batch normalization [110] is applied after each dense layer.
(vi) *Optimizer*. The optimization algorithm, either Adam [111] or Nadam [112].
(vii) *Learning rate*. The learning rate for the optimizer, selected on a logarithmic scale between $1 \times 10^{-5}$ and $1 \times 10^{-2}$.

It is important to note that batch normalization and dropout are introduced to prevent overfitting. The network is finalized with an output layer containing one neuron per label, using the sigmoid activation function [113].

### C. Model optimization and evaluation

The model is compiled using focal binary cross-entropy loss function [114],

$$\mathcal{L}_{\text{focal}}(y, p) = -[y(1-p)^{\gamma}\log(p) + (1-y)p^{\gamma}\log(1-p)], \tag{18}$$

where $p$ is the predicted probability, $y \in \{0, 1\}$ is the ground-truth label, and $\gamma$ is the focusing parameter (default value 2). This loss function pays special attention to hard examples that may be misclassified by the DL model, especially in the presence of class imbalance, by reducing the weight for cases that are well classified.

During the training and validation stage, a number of performance-measuring metrics, which are appropriate for MLC, are monitored to assess the DL model. These metrics include (for a review, see Ref. [115]):

(i) *Subset accuracy.* This is a distinct metric for MLC that considers an instance as correctly classified only if all labels are correctly predicted. For the entire dataset, it takes values between 0 (at least one label wrongly predicted for all instances) and 1 (all labels correctly predicted for all instances). We judge the success of our DL model based on this rather strict metric:

$$\text{Subset accuracy} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}(\hat{Y}_i = Y_i), \tag{19}$$

where $N$ is the number of samples, $i$ indexes each sample ($i = 1, \ldots, N$), $\mathbf{1}(\cdot)$ is the indicator function [$\mathbf{1}(\text{true}) = 1$, $\mathbf{1}(\text{false}) = 0$], $\hat{Y}_i$ is the predicted (multilabel) set for the $i$th sample, and $Y_i$ is the ground-truth (multilabel) set for the $i$th sample.

(ii) *Hamming loss*. This is a less strict metric representing the fraction of labels that are incorrectly predicted across all instances. This metric does not quantify whether all labels are incorrectly predicted for each instance. It takes values between 0 (all predictions correct) and 1 (all predictions incorrect):

$$\text{Hamming loss} = \frac{1}{N \times L}\sum_{i=1}^{N}\sum_{j=1}^{L}\mathbf{1}(y_{ij} \neq \hat{y}_{ij}). \tag{20}$$

where $N$ is the number of samples, $L$ is the number of labels per sample, $i$ indexes each sample, $j$ indexes each label ($j = 1, \ldots, L$), $\mathbf{1}(\cdot)$ is the indicator function, $y_{ij}$ is the ground-truth label, and $\hat{y}_{ij}$ is the predicted label for label $j$ of sample $i$.

(iii) *Matthews correlation coefficient (MCC)*. This metric takes into consideration true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). It combines them into a single formula and outputs 1 for perfect predictions,

[2]We have experimented with larger numbers of hidden layers and found no significant improvements compared to using three hidden layers.

0 for random classification, and −1 for completely wrong predictions:

$$\mathrm{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}. \tag{21}$$

(iv) *Macro F1 score*. This is the average of the F1 scores calculated for each label independently. The F1 score is the harmonic mean of precision ($TP/(TP+FP)$) and recall ($TP/(TP+FN)$):

$$\mathrm{F1}_j = \frac{2 \times \mathrm{precision}_j \times \mathrm{recall}_j}{\mathrm{precision}_j + \mathrm{recall}_j},$$
$$\text{macro-F1} = \frac{1}{L}\sum_{j=1}^{L} \mathrm{F1}_j, \tag{22}$$

where $L$ is the number of labels, and $j$ indexes the labels.

(v) *Accuracy*. This is used only when assessing each label individually, and represents the number of correct predictions over the total number of predictions:

$$\mathrm{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}, \tag{23}$$

where $TP$, $TN$, $FP$, and $FN$ are defined as above.

(vi) *Confusion matrix*. This matrix illustrates the number of TP, TN, FP, and FN for each label, providing detailed insight into where the DL model performs well and where it performs poorly.

In addition to these standard MLC metrics, we can further evaluate the capacity of the model to learn the unique class combinations (Table IV) by conducting a powerset evaluation. Namely, after the DL model is trained and labels are predicted, one treats each combination of labels as a separate class and computes precision, recall, and F1 scores. While subset accuracy is already a strict metric, it can be influenced by a few dominant combinations. By examining each combination individually, the powerset evaluation tells us if the model truly performs well across the entire range of label combinations, providing deeper insight into whether it has captured patterns between the combined labels.

In the process of optimizing our DL model, we introduce a custom score for Optuna to maximize during its trials. The score combines validation subset accuracy, validation loss, and validation MCC, and we define it as follows:

$$\begin{aligned}\text{Custom score} = {} & (\alpha \times \text{Val subset accuracy}) \\ & + \left(\beta \times \frac{1/(\text{Val loss}+\epsilon)}{1+1/(\text{Val loss}+\epsilon)}\right) \\ & + \left(\delta \times \frac{\text{Val MCC}+1}{2}\right),\end{aligned}$$

where $\alpha$, $\beta$, and $\delta$ represent scaling coefficients for metric terms which are all themselves designed to range from 0 and 1. $\epsilon$ is a small constant to prevent division by zero. The reason for introducing this score is that focusing on validation loss alone during the Optunatrials can lead to a model that underperforms with respect to the other validation metrics that we consider crucial. Once the experimentation phase is completed and the best parameters are found, we retrain the model using those parameters and apply early stopping based on minimizing the validation loss, which is common practice.

Finally, since to our knowledge there are no previous works in this context against which we can compare the performance of our DL model, we train a binary RF classifier [103] on the dataset as a baseline. The hyperparameters of the RF classifier are set to the default values given by SCIKIT-LEARN, which are: 100 decision trees, no restriction on the maximum depth, a minimum sample split of 2, a minimum sample leaf of 1, the Gini impurity criterion for the evaluation of splits, and the square root of the number of features as the number of features considered for each split. In addition, bootstrap sampling (bagging) was enabled to reduce overfitting.[3]

## V. RESULTS AND DISCUSSION

In this section, we present the results for our DL model and the physics analysis. For the DL model, we show the training history and the evaluation on a completely unseen test set comprising 116,014 points (instances). For the physics analysis, we use the correctly labeled instances from the test set and show, in the appropriate input $x$–$y$ plane, how the constraints affect the parameter space, both individually and collectively.

### A. Neural network architecture and training history

Our DL model was trained on the following input parameters (features) and constraints (binary labels):

(i) *Features*. $m_{H_2}$, $m_{H_D}$, $m_{A_D}$, $m_{H_D^\pm}$, $\alpha$, $\lambda_2$, $\lambda_8$, $v_s$, $m_{22}^2$.

(ii) *Labels*. $BfB$, PU, STU, Higgs.

The architecture of the DL model and its hyperparameters were selected after experimenting with 100 Optunatrials, which are shown in Table V. Namely, we have an input layer that receives preprocessed features, where both YJ transformation and standard scaling are applied. Next, we have three deep layers with 875 neurons for the

[3]We did not observe any significant improvements from manually tuning the hyperparameters of the RF classifier.

TABLE V. Optimized parameters for the d learning model.

| Parameter | Value |
|---|---|
| Apply Yeo-Johnson transformation | Yes |
| Apply standard scaler | Yes |
| Batch size | 781 |
| Number of layers | 3 |
| Units in layer 1 | 875 |
| Units in layer 2 | 938 |
| Units in layer 3 | 402 |
| Activation function | ReLU |
| Dropout rate | 0.117 |
| Apply batch normalization | Yes |
| Optimizer | Adam |
| Learning rate | 0.0003 |
| Custom score weights[a] | $\alpha = \beta = 0.4$, $\delta = 0.2$ |

[a]These were experimented with manually, so they are not suggested by Optuna.

first, 938 for the second, and 402 for the third. Each layer is followed by ReLU activation, then batch normalization and dropout (0.117). We finally have an output layer comprising four neurons (one for each label), and a Sigmoid activation function. The training is optimized via the Adam optimization algorithm with a learning rate of 0.0003.

The training and validation histories are illustrated in Fig. 1, which presents the subset accuracy, loss, Hamming loss, MCC, and macro-F1 score as functions of epochs. These plots provide crucial information regarding the performance of the DL model during the training stage, and can reveal if it suffers from overfitting.

Starting with Fig. 1(a), we can see the evolution of the subset accuracy, which we consider the ultimate measure of our DL model given that it focuses on all labels at once. The figure shows a rapid improvement in the ability to predict all labels correctly, with accuracy stabilizing around 0.95 after approximately 60 epochs. The gap between the training and validation shrinks as the training advances, which indicates to us that the model is not overfitting, and is effectively learning global features of the data. Moreover, the loss plot [Fig. 1(b)] reveals a significant decrease in both training and validation loss during the early epochs, both plateauing at around 0.01 when epoch 80 is reached. This decrease in loss means that the DL model is able to minimize the difference between predicted and actual values, and the smallness of the gap at the end reassures us that the model is not overfitting. The Hamming loss, shown in Fig. 1(c), for both training and validation also decreases sharply in the early epochs and stabilizes at around 0.015. This reduction further confirms the ability of the DL model to make fewer incorrect predictions as training progresses. Next, the MCC plot [Fig. 1(d)] shows an increasing improvement during the initial epochs, with values stabilizing around 0.96 after approximately 60 epochs. This suggests that the DL model learns to make balanced predictions (decreasing cases of FP and FN), with consistent performance as seen from the curves of both the training and validation datasets. Figure 1(e) illustrates the macroaveraged F1 score and demonstrates a continuing improvement that stabilizes around 0.98. This high F1 score highlights that the DL model can balance precision and recall. The consistency between the training and validation in all of the metrics shown indicates that the DL model is robust and does not suffer from alarming overfitting.

### B. Performance evaluation

To confirm whether the trained DL model can generalize to completely unseen data, we evaluate its performance on the test dataset (116,140 points). In particular, since the main task is to optimize the DL model for MLC, we consider the subset accuracy (on the test dataset) to be the major indicator of its performance. Its value is found to be 0.96, which means that it correctly predicted the values of all labels simultaneously of 96% of the test dataset (111,504 points). To gain insight into how significant this result is, we compare the subset accuracy we obtained with that of an RF classifier, which we trained on the same dataset and evaluated on the same unseen test dataset. We find that the subset accuracy of the RF classifier in the test set is 0.78. Evidently, the DL multilabel classifier significantly outperforms the traditional RF classifier.

Furthermore, it is vital to assess the performance of the DL model on individual labels, which will evaluate it as a single-label classifier, along with being used for MLC. This can be done by computing the confusion matrix for each label. The results are presented in Fig. 2. Each confusion matrix illustrates the model's ability to distinguish a valid case (class 1) from an invalid case (class 0). For $BfB$, the confusion matrix [Fig. 2(a)] shows that the DL model was able to correctly identify 69,863 valid cases and 44,798 invalid cases, with 1181 FP and 298 FN predictions. As for PU, Fig. 2(b) illustrates only four FN and 355 FP misclassifications, while 58,069 cases were correctly classified as invalid and 57,712 correctly classified as valid. Next, for the STU label, the DL model correctly predicted 64,930 valid cases and 49,626 invalid cases, as shown in Fig. 2(c). The figure also shows 1474 FP and 110 FN cases. Finally, the confusion matrix for the *Higgs* label [Fig. 2(d)] shows that the DL model correctly identified 66,538 valid cases and 48,309 invalid cases, with 1151 FP and 142 FN cases. These results highlight a very strong performance in the single-label classification task.

The performance of the DL model on individual labels can also be measured by computing accuracy (for individual labels, not to be confused with subset accuracy, which is relevant for MLC), precision, recall, and F1 score from the confusion matrices in Fig. 2. Starting with the $BfB$ label, the accuracy is 0.9872, with a precision of 0.9834, a recall of 0.9957, and an F1 score of 0.9895. As for the PU label,

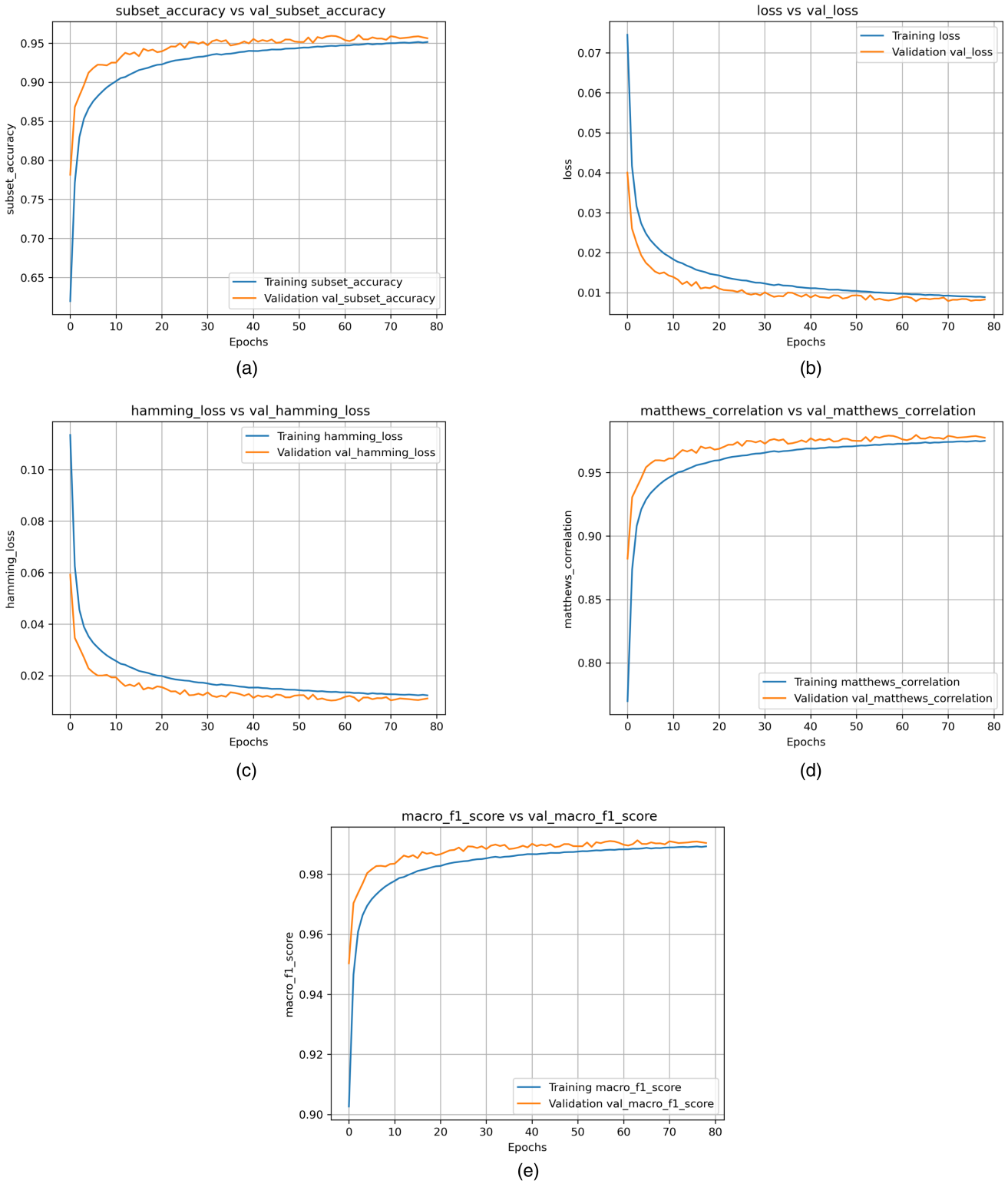



FIG. 1. Training history metrics for the model, including subset accuracy, loss, Hamming loss, Matthews correlation, and macro-F1 score for both training and validation datasets. (a) Subset accuracy. (b) Loss vs validation loss. (c) Hamming loss. (d) Matthews correlation. (e) Macro F1 score.

the accuracy is 0.997, with a precision of 0.9939, a recall of 0.9999, and an F1 score of 0.9969. Next, the STU label has an accuracy of 0.9864, with a precision of 0.9778, a recall of 0.9983, and an F1 score of 0.9879. Finally, the *Higgs* label shows an accuracy of 0.989, with a precision of 0.983, a recall of 0.9979, and an F1 score of 0.9904. Overall, these

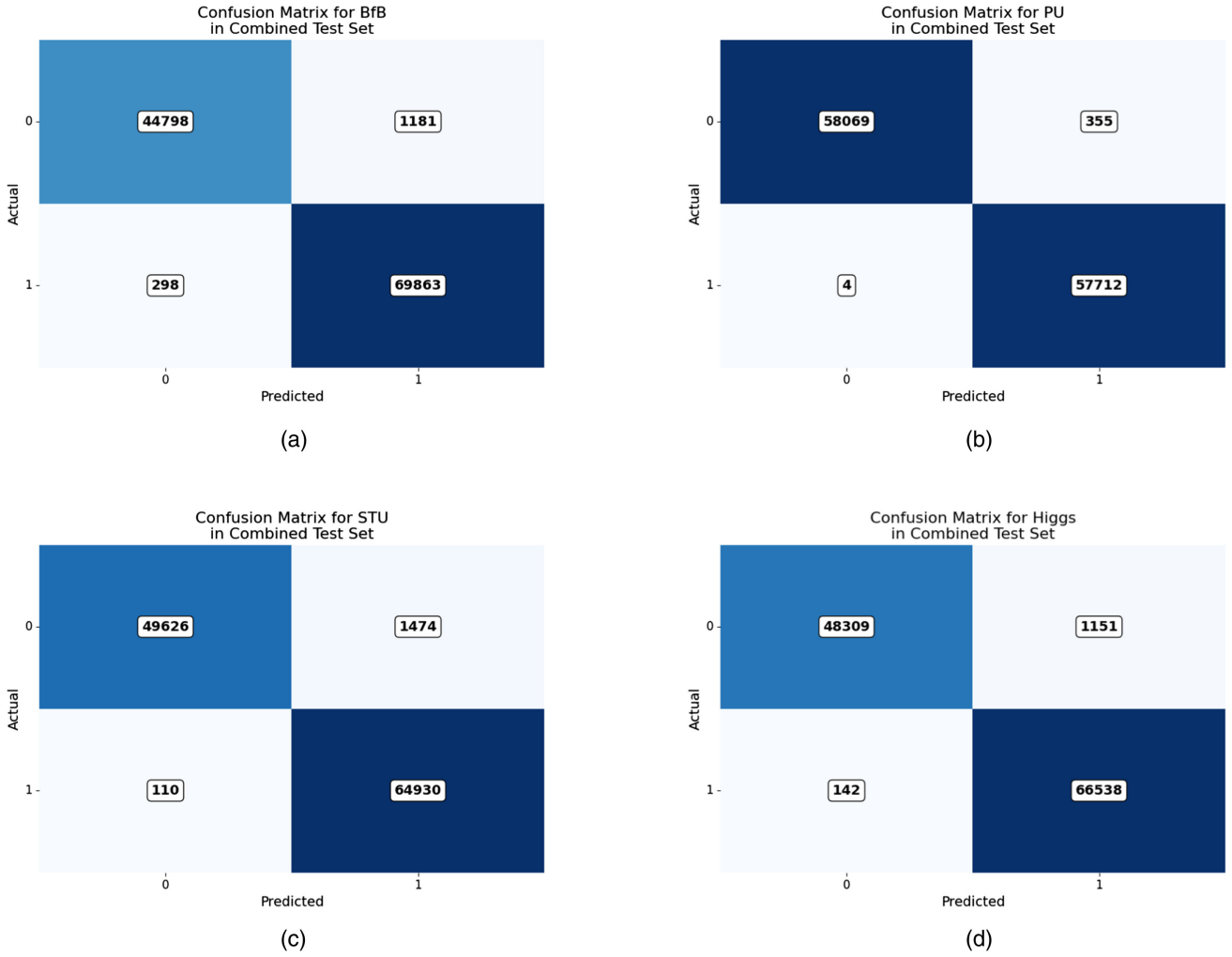



FIG. 2. Confusion matrices for the $BfB$, PU, STU, and Higgs labels, showing the model's performance in predicting valid and invalid cases for each label separately. (a) Confusion matrix for $BfB$. (b) Confusion matrix for PU. (c) Confusion matrix for STU. (d) Confusion matrix for Higgs.

metrics demonstrate strong performance across all labels, further indicating that our DL model can be effectively utilized for predicting each label individually (as a single-label classifier) as well as all labels at once (as a multilabel classifier).

Finally, we further validate that our DL model captures the underlying joint distribution of labels (Table IV). We treated each of the 16 possible label combinations as a single class and conducted a powerset evaluation. Table VI presents the precision, recall, F1 score, and support for each of these combined classes, while the accuracy is the subset accuracy found before. Notably, all combinations achieve consistently high performance metrics ranging from F1 score of ∼93% to ∼98%. Moreover, we can see from the confusion matrix in Fig. 3 that misclassification is rare. However, when the joint class (1_1_1_1) is misclassified, it is mainly misclassified as a joint class with at least three valid labels. The same observation is true for the joint class (0_0_0_0) and the other classes in which the

TABLE VI. Powerset classification metrics for each label combination in the test dataset.

| $BfB$_PU_STU_Higgs | Precision | Recall | F1 score | Support |
|---|---|---|---|---|
| 0_0_0_0 | 0.9923 | 0.9623 | 0.9771 | 13,584 |
| 0_0_0_1 | 0.9693 | 0.9479 | 0.9585 | 5125 |
| 0_0_1_0 | 0.9548 | 0.9684 | 0.9616 | 5412 |
| 0_0_1_1 | 0.9509 | 0.9593 | 0.9551 | 4100 |
| 0_1_0_0 | 0.9726 | 0.9047 | 0.9374 | 3881 |
| 0_1_0_1 | 0.9700 | 0.9199 | 0.9443 | 4393 |
| 0_1_1_0 | 0.9503 | 0.9045 | 0.9268 | 3549 |
| 0_1_1_1 | 0.9448 | 0.9166 | 0.9305 | 5935 |
| 1_0_0_0 | 0.9673 | 0.9539 | 0.9606 | 5924 |
| 1_0_0_1 | 0.9730 | 0.9581 | 0.9655 | 6970 |
| 1_0_1_0 | 0.9595 | 0.9696 | 0.9646 | 5893 |
| 1_0_1_1 | 0.9653 | 0.9858 | 0.9754 | 11,416 |
| 1_1_0_0 | 0.9517 | 0.9414 | 0.9465 | 5444 |
| 1_1_0_1 | 0.9395 | 0.9377 | 0.9386 | 5779 |
| 1_1_1_0 | 0.9230 | 0.9463 | 0.9345 | 5773 |
| 1_1_1_1 | 0.9520 | 0.9961 | 0.9735 | 22,962 |

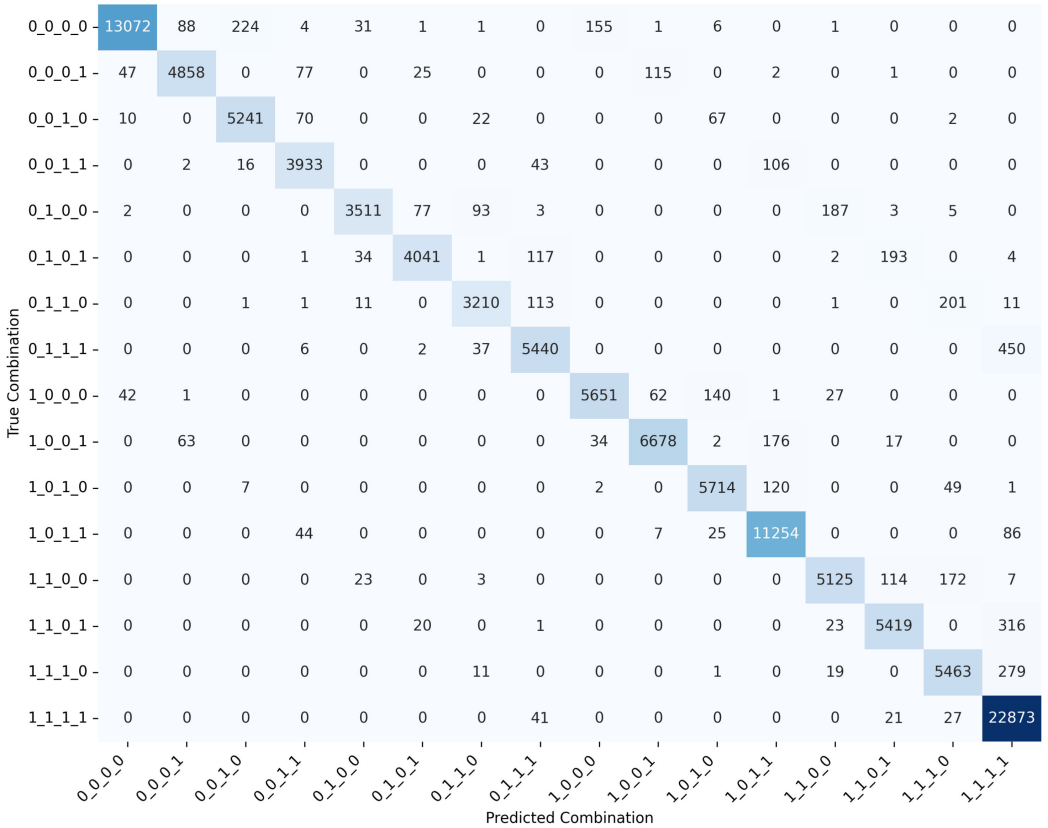



FIG. 3. Powerset-based confusion matrix for the multilabel classifier. Each cell $(i, j)$ shows how often class $i$ (true) is predicted as class $j$.

misclassification arises due to one constraint. There are no cases in which a completely invalid result was identified as completely valid. These results strongly suggest that the model effectively learns and reproduces the joint distribution of the constraints, rather than simply focusing on the most frequent combinations, and provides reliable predictions.

### C. Time advantage analysis

To quantify the practical time advantage of the DL approach, we conducted a timing analysis comparing the traditional physics-based calculations, using ScannerS, with our trained multilabel classifier. The tests were performed on a workstation equipped with an Intel Xeon Silver 4114 CPU (2.20 GHz, 10 cores/20 threads) and 32 GB of RAM. Additionally, to factor out any internal effects on the physics tool, we used the already generated test dataset (116,140 points), and executed the `check` function of ScannerS, which is designed to work on previously sampled data. We restrict ScannerS to only check (i.e., classify) the four relevant constraints; hence, eliminating any time taken to generate samples, validate input parameters, or check DM and vacuum stability. We find that ScannerS required 31,644.96 seconds (about nine hours) to complete the `check` job.

As for the DL method, we note that applying the Yeo-Johnson transformation and scaling to the test dataset required 1.06 seconds before running the classifier. During the evaluation of the classifier on the test dataset, we used the default Keras batch size of 32 (not to be confused with batch size used for the training job). This resulted in a prediction time of 103.68 seconds. Then we increased the inference batch size to 2048 and 8192, which resulted in reducing the time to 3.74 and 2.78 seconds, respectively, with no loss in predictive performance. We note that, although the workstation included an AMD Radeon Pro WX 5100 Graphics Processing Unit (GPU), the classifier ran on CPU only, as our standard TensorFlow build could not utilize the GPU. We expect the utilization of GPU to further reduce the time for the classifier to complete its prediction job. Table VII summarizes the execution times.

TABLE VII. Execution time comparison.

| Method | Time (seconds) | Speedup factor |
|---|---|---|
| ScannerS | 31,644.96 | 1.0× |
| *Multilabel classifier (CPU only)* | | |
| Default (batch size = 32) | 103.68 | 305.2× |
| Batch size = 2048 | 3.74 | 8464.9× |
| Batch size = 8192 | 2.78 | 11,383.4× |

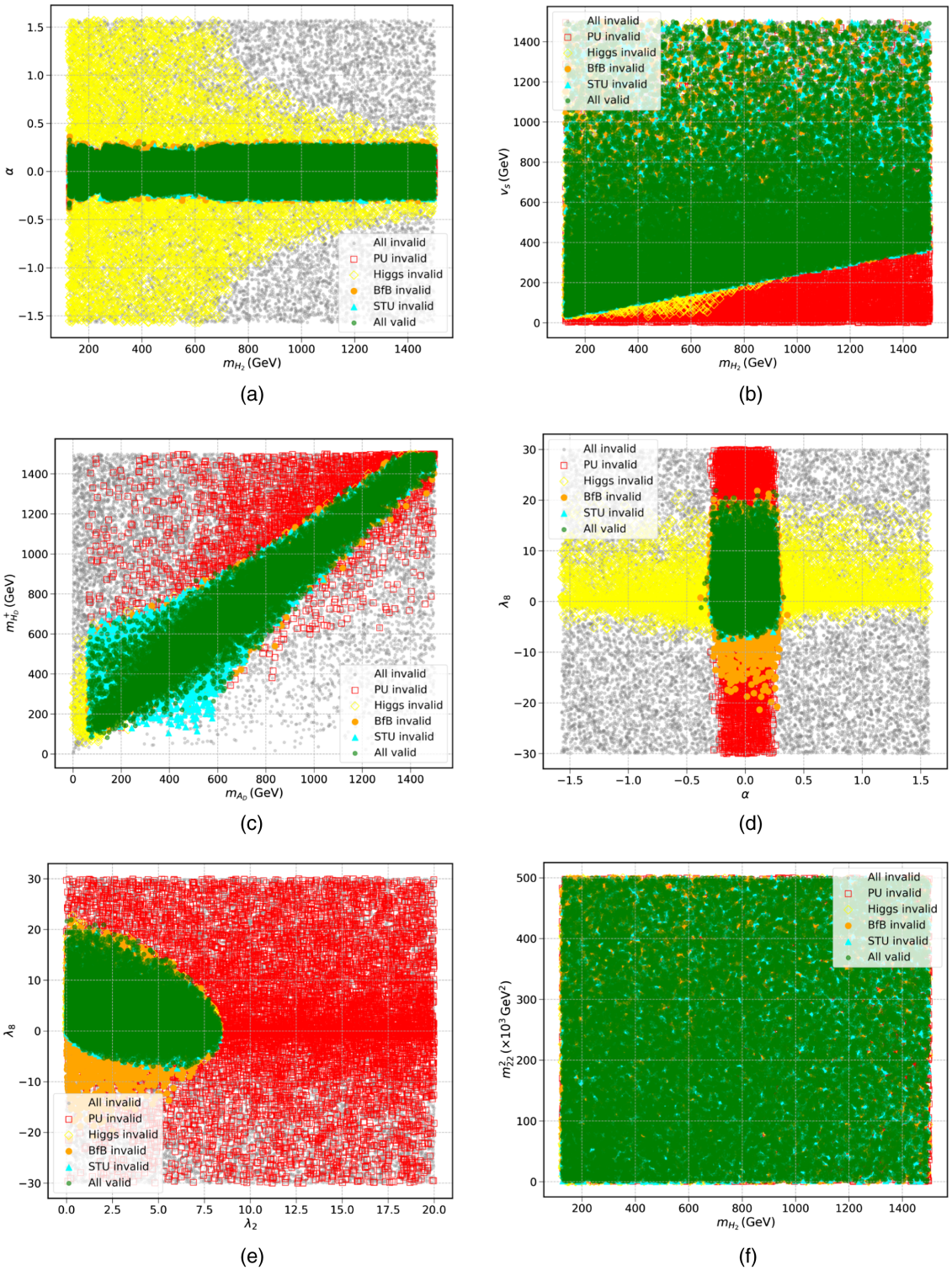



FIG. 4. Visualization of the model's parameter space under different constraint validity scenarios. Green points indicate all constraints are satisfied, while red, orange, yellow, and cyan points show where $BfB$, PU, STU, and Higgs constraints, respectively, are invalid. (a) Comparison of all-valid and invalid ($BfB$, PU, STU, Higgs) scenarios in the $m_{H_2}$ vs $\alpha$ plane. (b) Comparison of all-valid and invalid ($BfB$, PU, STU, Higgs) scenarios in the $m_{H_2}$ vs $v_s$ plane. (c) Comparison of all-valid and invalid ($BfB$, PU, STU, Higgs) scenarios in the $m_A$ vs $m_{H^\pm}$ plane. (d) Comparison of all-valid and invalid ($BfB$, PU, STU, Higgs) scenarios in the $\alpha$ vs $\lambda_8$ plane. (e) Comparison of all-valid and invalid ($BfB$, PU, STU, Higgs) scenarios in the $\lambda_2$ vs $\lambda_8$ plane. (f) Comparison of all-valid and invalid ($BfB$, PU, STU, Higgs) scenarios in the $m_{H_2}$ vs $m_{22}^2$ plane.

The results demonstrate a remarkable improvement in computational efficiency. The trained classifier, running on CPU, achieved a speedup factor of over 11,000 compared to the traditional physics-based calculations, while maintaining the high subset accuracy. Indeed, the substantial reduction in computation time, combined with strong performance across different evaluation metrics, suggests that our DL model could serve as an efficient surrogate for rapid parameter space explorations and checks in the DDP-N2HDM, enabling faster theoretical studies and phenomenological analyses.

### D. Physics analysis

Using the 96% correctly labeled points by the DL model, which comprises 111,504 points, Fig. 4 shows the parameter space in the $m_{H_2} - \alpha$, $m_{H_2} - v_s$, $m_{A_D} - m^+_{H_D}$, $\alpha - \lambda_8$, and $\lambda_2 - \lambda_8$ planes. These were selected after inspecting all possible pairings of the input parameters, as they were found to distinctly show the impact of each constraint. In each graph, valid points are represented in green, while points that are invalid due to only one of the constraints are shown as follows: red (PU invalid), yellow (*Higgs* invalid), orange ($BfB$ invalid), and cyan (STU invalid). Points that are invalid due to all constraints being violated at the same time are shown in gray. The plotting scheme follows a specific order: the data with all constraints invalid is laid as background, followed by invalid PU, invalid *Higgs*, invalid $BfB$, and finally invalid STU. Valid points are placed on top of all others, so in regions where valid and invalid points overlap, valid points are prioritized. This ordering reflects the impact of each constraint, with those plotted last having the least impact.

In Fig. 4(a), we observe how *Higgs* constraints rule out the wedge-shaped regions where $m_{H_2}$ is below 1200 GeV, and $|\alpha| > 0.3$. In this plane, we can see that this constraint is significant in its own right, while points ruled out due to only one of the other constraints reside beneath the valid region, except for a few points where we see marginal effects from $BfB$ and STU. The case where all constraints are violated at the same time (gray points) is also significant here as it rules out the top left and right corners.

Next, Fig. 4(b), shows how PU erases the bottom triangle requiring $v_s$ to steadily grow from close to 1 to value above 200 GeV. In this plane, *Higgs* constraints are also apparent in regions around $m_{H_2} \sim 600$ and $1 < v_s < 200$ GeV, while the other constraints (STU and $BfB$) have a small visible effect for larger values of $v_s$.

Moreover, Fig. 4(c), illustrates the effect of STU in the region where $m_{A_D}$ is between 200 and 600 GeV, and $m^+_{H_D}$ is between 150 and 400 GeV. The *Higgs* constraint rules out regions where $m_{A_D}$ is below 100 GeV, while PU rules out distinct regions on the sides of the valid band for values of $m^+_{H_D} > 600$ GeV, while all constraints being invalid at the same time rule out the remaining parameter space.

The $\alpha$-$\lambda_2$ plane presented in Fig. 4(d) clearly illustrates the individual effects of the *Higgs*, PU, and $BfB$ constraints, while the case of all constraints being invalid excludes the boxed corners in this parameter space.

Next, the $\lambda_2 - \lambda_8$ plane in Fig. 4(e) again shows the wide impact of PU on the parameter space. The effect of $BfB$ is also clear, especially on values of $\lambda_2$ between 0 and 7.5, and $\lambda_8$ between 0 and lower $-20$, which we saw before for $\lambda_8$ in the previously mentioned plot. The *Higgs* and STU limits do not play a role in constraining this plane. Figure 4(f) demonstrates that all four constraints considered in this work do not affect the $m_{H_2} - m^2_{22}$ plane, as all of the ranges of their values are present in the valid points.

Overall, we see from Fig. 4 that, after careful consideration of the main issues surrounding the generation of a multilabel classifier, the trained classifier correctly predicted the four constraints on the parameter space within seconds and with very strong performance. It provided a picture that indeed represents the status of the model, and factual statements about the model and its parameters were made; therefore, demonstrating the success and promise of this DL method for the task at hand.

To finalize with a broad picture, we point out only two possible directions that can be considered based on the obtained insights (model status). First, we see that the $PU$ constraint had a significant impact, even though the relevant computations within ScannerS implement tree-level precision. However, it was shown in [54] that higher order corrections can affect unitarity constraints, by either closing or opening regions in the parameter space. One can consider this an interesting direction to take for the DDP-N2HDM. Second, Higgs constraints also had a significant impact on this model, which has a real singlet field. If one extends the model to a complex singlet, then that will lead to an additional $CP$-odd state $A$, effectively changing the phenomenology and affecting Higgs constraints. For example, if $A$ can decay to $H_2$, then LHC searches for $pp \to A \to H_2 Z$ would constrain both $A$ and $H_2$. Also in regions where $H_2$ can decay to $A$, the constraints on $H_2$ from $pp \to H_2 \to H_1 H_1 / VV / f\bar{f}$ would change. Complexifying the singlet will surely have other effects to be determined by a dedicated comparative study.

## VI. CONCLUSION

In this work, we have discussed how validating parameter points in BSM extensions involves a complicated chain of checks for several theoretical and experimental constraints. This is usually performed by utilizing dedicated physics tools chained together, requiring a large amount of time. We discussed current advances in utilizing AI methods in the exploration of parameter spaces of SM extensions, and the limitations of single-label tasks in terms of learning the effects of groups of constraints on the models, which is an important aspect of particle phenomenology research. This motivates us to explore, for the first

time, the feasibility of training an MLC using DL to perform such a task, focusing on the DDP-N2HDM as a representative multidimensional model. A total of nine free parameters and four constraints ($BfB$, PU, STU, and Higgs) were taken into account as features and binary target labels. The dataset was generated using a hybrid method combining LHS and random sampling, ensuring good coverage of the parameter space. The dataset was then subjected to appropriate preprocessing, including YJ transformation and scaling.

After experimentation, the DL multilabel classifier was built using relevant metrics for measuring performance. We took every precaution to ensure that it does not overfit the data. Upon applying the resulting classifier to unseen test data, we demonstrated that the DL model performs exceptionally well, achieving a subset accuracy of 0.96, significantly outperforming our baseline RF classifier, which achieved a subset accuracy of 0.78. Additionally, we showed the strong performance of the DL model on individual labels, with all metrics reaching nearly perfect scores. Furthermore, we demonstrated that the classifier effectively learned joint class distributions, capturing the interplay between different constraints, with F1 scores varying between 93% and 98%.

We have carried out a timing analysis, in which we showed that the traditional physics tool required about nine hours to check the constraints on 116,140 points, while the trained multilabel classifier performed the job in less than three seconds, which is orders of magnitude faster. Hence, the classifier can act as a fast surrogate for model status checks with respect to the four considered constraints.

Using the correctly classified parameter space by the DL model, and with an appropriate choice of parameter planes, we demonstrated the effect of each constraint individually when it is violated, as well as the collective effect when multiple constraints are violated. We observed how the collective violation of constraints rules out certain regions that are not excluded by only one constraint, and pointed out that regions where only one constraint is violated might lead to new research directions. This further demonstrates the success of the method, and its promise for further applications.

While the DL model was developed specifically for the DDP-N2HDM, the approach is generalizable. To facilitate that, we have provided a Python tool HEPMLC, following the methodology described in this paper, to create multilabel classifiers for models implemented in the public tool ScannerS (if the user requires data generation) or for labeled datasets provided by the user.

There are several directions that can be taken in future studies, as the field is still in its infancy. One pressing issue is generating high-quality data in a relatively short time to train multilabel classifiers. Utilizing some of the newly explored single-label methods for fast scans might be an option, by sequentially adapting the fast scanner for each required constraint. This includes utilizing or creating regressors for certain computational jobs. Additionally, it is important to study the impact of new or updated constraints, since the ability to update the classifier smoothly and without requiring full retraining on new large datasets would be a significant advantage. This might be carried out by fine-tuning the classifier with new data; however, the issue of catastrophic forgetting needs to be investigated.

Finally, as the community is still learning, and work in this direction is progressing, we anticipate many exciting developments that exploit the full power of AI methods in exploring parameter spaces of BSM extensions, with the potential to minimize the need for traditional methods for some time-consuming tasks such as scanning and checking HEP constraints, which are subject to updates due to more precise computations and/or new limits from experimental searches.

## ACKNOWLEDGMENTS

I would like to thank CERN for hospitality during the summer where part of this work was conducted. I also thank the anonymous reviewers for their feedback which helped improve the paper. This work is supported by King Saud University.

## DATA AVAILABILITY

The data that support the findings of this article are openly available [116].

## APPENDIX: APPLICATIONS WITH HEPMLC

In this appendix, we describe HEPMLC, a Python-based tool designed for training multilabel classifiers of BSM extensions. The tool is publicly available in Ref. [117] and provides a streamlined workflow for training and evaluating neural networks (NNs) to predict theoretical and experimental constraints defined as target (binary) labels.

### 1. Installation and setup

HEPMLC is designed to work standalone, if the user provides a dataset containing inputs and constraints

(as binary labels), or to work within ScannerS to generate a dataset for any of the implemented models. In the latter case, after installing ScannerS, HEPMLC can be installed by cloning the repository into the build directory:

```
cd ScannerS/build
git clone [118]
```

### 2. Tool structure

HEPMLC provides a modular structure with the following components:

(i) `src/preprocessing/`: Data preprocessing and analysis modules.
(ii) `src/modeling/`: NN architecture, optimization, fine-tuning, and training modules.
(iii) `src/utils/`: Utilities for interfacing with ScannerS.
(iv) `notebooks/`: Jupyter notebooks with detailed usage exploiting the full functionality of the tool.
(v) `HEPMLC.ipynb`: Jupyter notebooks with a basic example usage.

### 3. Example usage workflow

A basic example is provided through `HEPMLC.ipynb`, which guides users through the following steps:

#### a. Import HEPMLC modules

Along with essential libraries, we import the tool's modules, which are defined in the `/src` directory.

```
# Import required modules. These are defined in /src.
from utils.model_reader import ModelReader
from utils.scanner_runner import ScannerRunner
from preprocessing.preprocessor import FeaturePreprocessor
from modeling.architecture import ModelBuilder
from modeling.trainer import ModelTrainer
```

#### b. Physics model selection

Users can select and configure any physics model available in ScannerS. For each model, the tool displays:

(i) Available input parameters and their ranges.
(ii) Theoretical and experimental constraints.
(iii) Configuration options for constraints (apply/ignore/skip).

```
# List available models
model_reader = ModelReader(scanner_path=scanner_path)
print("Available Models:")
```

Output:

```
Available Models:
Complex 2HDM flipped (C2HDM_FL.ini)
Complex 2HDM lepton specific (C2HDM_LS.ini)
Complex 2HDM type 1 (C2HDM_T1.ini)
Complex 2HDM type 2 (C2HDM_T2.ini)
CP-violating dark matter (CPVDM.ini)
Complex singlet broken phase (CxSMBroken.ini)
Complex singlet dark (CxSMDark.ini)
N2HDM broken type 2 (N2HDMBroken_T2.ini)
N2HDM dark D (N2HDMDarkD.ini)
N2HDM dark SD (N2HDMDarkSD.ini)
N2HDM dark S type 1 (N2HDMDarkS_T1.ini)
Real 2HDM flipped (R2HDM_FL.ini)
Real 2HDM lepton specific (R2HDM_LS.ini)
Real 2HDM type 1 (R2HDM_T1.ini)
```

```
Real 2HDM type 2 (R2HDM_T2.ini)
TRSM broken phase (TRSMBroken.ini)
```

```
# Select model and configure
selected_model = "N2HDMDarkD.ini" # Change this to your chosen model
features, constraints = model_reader.read_model(selected_model)
```

Output:

```
Features and ranges:
- mHa: [125.09, 125.09]
- mHb: [50.0, 1000.0]
- mHD: [1.0, 1500.0]
- mAD: [1.0, 1500.0]
- mHDp: [1.0, 1500.0]
- alpha: [-1.57, 1.57]
- m22sq: [0.001, 500000.0]
- L2: [0.0, 20.0]
- L8: [-30.0, 30.0]
- vs: [1.0, 1500.0]

Constraints:
- BfB: ignore
- Uni: ignore
- STU: ignore
- Higgs: ignore
- VacStab: skip
-DM: skip
```

### c. Data generation

The tool interfaces with ScannerS to:

(i) Generate training data with specified parameters (features and labels).
(ii) Analyze feature distributions, and class distributions of the target labels.
(iii) Provide recommendations for additional scans if severe class imbalance is detected.

```
# Configure scan parameters
n_points = 1000 # Number of points to generate
output_file = os.path.join(build_dir, 'HEPMLC', results_dir, "scan_data.tsv")

# Run scan
scanner = ScannerRunner()
scanner.run_scan(selected_model, n_points, output_file)

# Analyze class balance
feature_cols = ['mH2', 'mHD', 'mAD', 'mHDp', 'alpha', 'L2', 'L8', 'vs', 'm22sq']
label_cols = ['valid_BFB', 'valid_Uni', 'valid_STU', 'valid_Higgs']

stats = scanner.analyze_class_balance(output_file, label_cols)
scanner.plot_class_distribution(stats, os.path.join(results_dir,
  'class_distribution'))
```

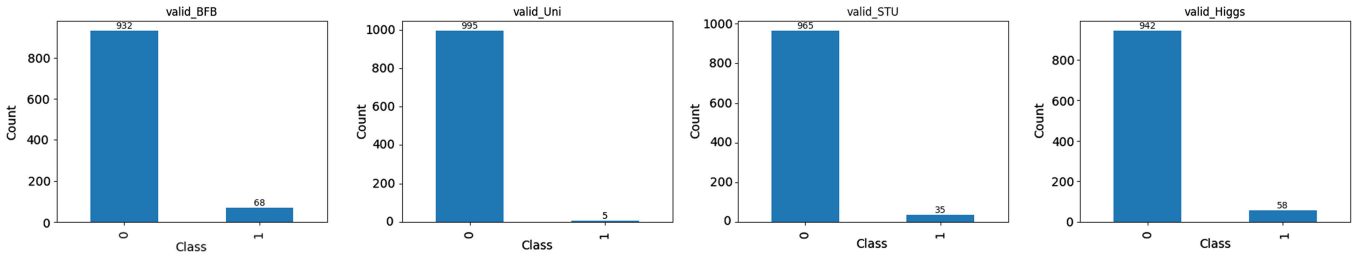



FIG. 5. Output showing class balance for each label.

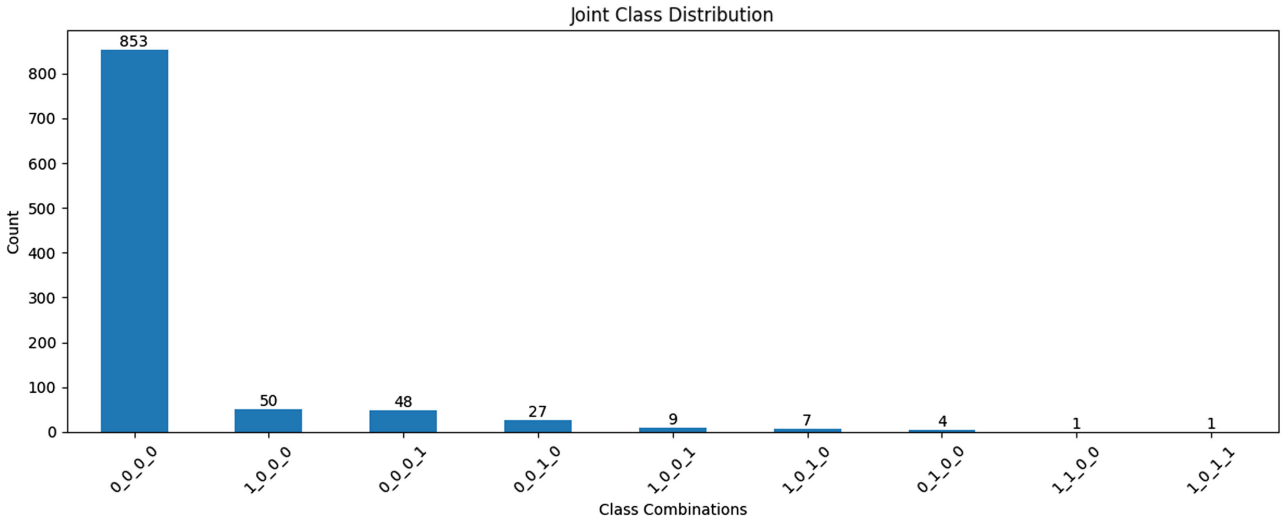



FIG. 6. Output showing joint class distributions.

The output is shown in Figs. 5 and 6.

### d. Data preprocessing and splitting

```
# Configure preprocessing
apply_yj = True # Apply Yeo-Johnson transformation
apply_scaler = True # Apply Standard scaling

# Initialize preprocessor
preprocessor = FeaturePreprocessor(apply_yj=apply_yj, apply_scaler=apply_scaler)

# Split data (70-15-15)
X = data[feature_cols]
y = data[label_cols]

X_temp, X_test, y_temp, y_test = train_test_split(X, y, test_size=0.15, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_temp, y_temp, test_size=0.176,
  random_state=42)

print(f"Training set: {len(X_train)} samples")
print(f"Validation set: {len(X_val)} samples")
print(f"Test set: {len(X_test)} samples")

# Preprocess data
X_train_processed = preprocessor.fit_transform(X_train)
```

*(Table continued)*

*(Continued)*

```
X_val_processed = preprocessor.transform(X_val)
X_test_processed = preprocessor.transform(X_test)

# Save preprocessor
preprocessor.save_transformers(os.path.join(results_dir, 'preprocessor'))
```

### e. Model configuration and training

Users can modify the model's (hyper)parameters through the interface. Training progress is monitored with multiple metrics:

(i) Subset accuracy.
(ii) Hamming loss.
(iii) Matthews correlation coefficient.
(iv) Macro F1 score.

```
# Model configuration (default parameters from paper)
model_params = {
  "n_layers": 2,
  "n_units_0": 64,
  "n_units_1": 128,
  "activation": "relu,"
  "dropout_rate': 0.01, #0.117,
  "apply_batch_norm": True,
  "optimizer": "adam,"
  "regularization": None,
  "reg_lambda": 0.05,
  "learning_rate": 0.000263,
  "batch_size": 64
}

# Training configuration
training_params = {
  "epochs": 15,
  "patience": 5, # Early stopping patience
}

# Initialize model and trainer
builder = ModelBuilder(
  input_shape=(len(feature_cols),),
  num_outputs=len(label_cols)
)

trainer = ModelTrainer(
  model_builder=builder,
  feature_cols=feature_cols,
  label_cols=label_cols,
  output_dir=os.path.join(results_dir, "model")
)

# Train model
model = trainer.train(
  X_train=X_train_processed,
  y_train=y_train,
```

*(Table continued)*

*(Continued)*

```
    X_val=X_val_processed,
    y_val=y_val,
    params=model_params,
    epochs=training_params["epochs"]
  )
```

### f. Evaluation

The tool provides comprehensive evaluation including:

(i) Individual label performance metrics.
(ii) Confusion matrices.
(iii) Powerset-based evaluation.
(iv) Joint label distribution analysis.

```
# Evaluate on test set
trainer.evaluate(
  model=model,
  X_test=X_test_processed,
  y_test=y_test
)

print("\nAll results have been saved in the Results directory.")
```

The output of the last two steps contains detailed `csv` files and plots on the training history, and performance of the classifier. Example outputs can be found in the Results folder within the Github repository.


[1] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning* (MIT Press, Cambridge, MA, 2016), http://www.deeplearningbook.org.

[2] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar, *Foundations of Machine Learning*, (The MIT Press, Cambridge, MA, 2012).

[3] Dan Guest, Kyle Cranmer, and Daniel Whiteson, Deep learning and its application to LHC physics, Annu. Rev. Nucl. Part. Sci. **68**, 161 (2018).

[4] Kim Albertsson *et al.*, Machine learning in high energy physics community white paper, J. Phys. Conf. Ser. **1085**, 022008 (2018).

[5] Alexander Radovic, Mike Williams, David Rousseau, Michael Kagan, Daniele Bonacorsi, Alexander Himmel, Adam Aurisano, Kazuhiro Terao, and Taritree Wongjirad, Machine learning at the energy and intensity frontiers of particle physics, Nature (London) **560**, 41 (2018).

[6] Dimitri Bourilkov, Machine and deep learning applications in particle physics, Int. J. Mod. Phys. A **34**, 1930019 (2020).

[7] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová, Machine learning and the physical sciences, Rev. Mod. Phys. **91**, 045002 (2019).

[8] Murat Abdughani, Jie Ren, Lei Wu, Jin Min Yang, and Jun Zhao, Supervised deep learning in high energy phenomenology: A mini review, Commun. Theor. Phys. **71**, 955 (2019).

[9] Huilin Qu and Loukas Gouskos, ParticleNet: Jet tagging via particle clouds, Phys. Rev. D **101**, 056019 (2020).

[10] Eric A. Moreno, Thong Q. Nguyen, Jean-Roch Vlimant, Olmo Cerri, Harvey B. Newman, Avikar Periwal, Maria Spiropulu, Javier M. Duarte, and Maurizio Pierini, Interaction networks for the identification of boosted $H \to b\bar{b}$ decays, Phys. Rev. D **102**, 012010 (2020).

[11] Alexander Bogatskiy, Brandon Anderson, Jan T. Offermann, Marwah Roussi, David W. Miller, and Risi Kondor, Lorentz group equivariant neural network for particle physics, arXiv:2006.04780.

[12] Matthew D. Schwartz, Modern machine learning and particle physics, arXiv:2103.12226.

[13] Georgia Karagiorgi, Gregor Kasieczka, Scott Kravitz, Benjamin Nachman, and David Shih, Machine learning in the search for new fundamental physics, Nat. Rev. Phys. **4**, 399 (2022).

[14] Huilin Qu, Congqiao Li, and Sitian Qian, Particle transformer for jet tagging, *Proceedings of the 39th International*

*Conference on Machine Learning* (PMLR, 2022), Vol. 162, pp. 18281–18292.
[15] Phiala Shanahan *et al.*, Snowmass 2021 computational frontier CompF03 topical group report: Machine learning, arXiv:2209.07559.
[16] Jorge Alda, Jaume Guasch, and Siannah Penaranda, Using machine learning techniques in phenomenological studies on flavour physics, J. High Energy Phys. 07 (2022) 115.
[17] S. Navas *et al.*, Review of particle physics, Phys. Rev. D **110**, 030001 (2024).
[18] Matthew Feickert and Benjamin Nachman, A living review of machine learning for particle physics, arXiv:2102.02770.
[19] Mary K. Gaillard, Paul D. Grannis, and Frank J. Sciulli, The standard model of particle physics, Rev. Mod. Phys. **71**, S96 (1999).
[20] D. Cogollo, F. F. Freitas, C. A. de S. Pires, Yohan M. Oviedo-Torres, and P. Vasconcelos, Deep learning analysis of the inverse seesaw in a 3-3-1 model at the LHC, Phys. Lett. B **811**, 135931 (2020).
[21] A. Hammad, S. Khalil, and S. Moretti, Search for mono-Higgs signals in bb final states using deep neural networks, Phys. Rev. D **107**, 075027 (2023).
[22] W. Esmail, A. Hammad, and S. Moretti, Sharpening the $A \to Z^{(*)}h$ signature of the type-II 2HDM at the LHC through advanced machine learning, J. High Energy Phys. 11 (2023) 020.
[23] Mohamed Belfkir, Adil Jueid, and Salah Nasri, Boosting dark matter searches at muon colliders with machine learning: The mono-Higgs channel as a case study, Prog. Theor. Exp. Phys. **2023**, 123B03 (2023).
[24] Huifang Lv, Daohan Wang, and Lei Wu, Deep learning jet images as a probe of light Higgsino dark matter at the LHC, Phys. Rev. D **106**, 055008 (2022).
[25] Jun Guo, Jinmian Li, Tianjun Li, Fangzhou Xu, and Wenxing Zhang, Deep learning for $r$-parity violating supersymmetry searches at the LHC, Phys. Rev. D **98**, 076017 (2018).
[26] Sascha Caron, Jong Soo Kim, Krzysztof Rolbiecki, Roberto Ruiz de Austri, and Bob Stienen, The BSM-AI project: SUSY-AI–generalizing LHC limits on supersymmetry with machine learning, Eur. Phys. J. C **77**, 257 (2017).
[27] Sascha Caron, Tom Heskes, Sydney Otten, and Bob Stienen, Constraining the parameters of high-dimensional models with active learning, Eur. Phys. J. C **79**, 944 (2019).
[28] Kun Wang and Jingya Zhu, A novel scenario in the semi-constrained NMSSM, J. High Energy Phys. 06 (2020) 078.
[29] Jacob Hollingsworth, Michael Ratz, Philip Tanedo, and Daniel Whiteson, Efficient sampling of constrained high-dimensional theoretical spaces with machine learning, Eur. Phys. J. C **81**, 1138 (2021).
[30] Rajneil Baruah, Subhadeep Mondal, Sunando Kumar Patra, and Satyajit Roy, Probing intractable beyond-standard-model parameter spaces armed with machine learning, Eur. Phys. J. Special Topics **233**, 2597 (2024).
[31] Jie Ren, Lei Wu, Jin Min Yang, and Jun Zhao, Exploring supersymmetry with machine learning, Nucl. Phys. **B943**, 114613 (2019).
[32] Florian Staub, xBIT: An easy to use scanning tool with machine learning abilities, arXiv:1906.03277.
[33] Marco Antonio Arroyo-Ureña, R. Gaitán, and T. A. Valencia-Pérez, SpaceMath version 1.0 A *Mathematica* package for beyond the standard model parameter space searches, Rev. Mex. Fis. E **19**, 020206 (2022).
[34] Fernando Abreu de Souza, Miguel Crispim Romão, Nuno Filipe Castro, Mehraveh Nikjoo, and Werner Porod, Exploring parameter spaces with artificial intelligence and machine learning black-box optimization algorithms, Phys. Rev. D **107**, 035004 (2023).
[35] Mark D. Goodsell and Ari Joury, BSMArt: Simple and fast parameter space scans, Comput. Phys. Commun. **297**, 109057 (2024).
[36] Florian Staub, SARAH 4: A tool for (not only SUSY) model builders, Comput. Phys. Commun. **185**, 1773 (2014).
[37] Howard Baer, Csaba Balazs, Alexander Belyaev, J. Kenichi Mizukoshi, Xerxes Tata, and Yili Wang, Updated constraints on the minimal supergravity model, J. High Energy Phys. 07 (2002) 050.
[38] John R. Ellis, Keith A. Olive, and Yudi Santoso, The MSSM parameter space with nonuniversal Higgs masses, Phys. Lett. B **539**, 107 (2002).
[39] Florian Domingo and Ulrich Ellwanger, Updated constraints from $B$ physics on the MSSM and the NMSSM, J. High Energy Phys. 12 (2007) 090.
[40] A. Djouadi *et al.*, Benchmark scenarios for the NMSSM, J. High Energy Phys. 07 (2008) 002.
[41] A. Djouadi, U. Ellwanger, and A. M. Teixeira, The constrained next-to-minimal supersymmetric Standard Model, Phys. Rev. Lett. **101**, 101802 (2008).
[42] O. Buchmueller *et al.*, The CMSSM and NUHM1 in light of 7 TeV LHC, $B_s \to \mu^+\mu^-$ and XENON100 data, Eur. Phys. J. C **72**, 2243 (2012).
[43] Alejandro Celis, Victor Ilisie, and Antonio Pich, LHC constraints on two-Higgs doublet models, J. High Energy Phys. 07 (2013) 053.
[44] Otto Eberhardt, Ulrich Nierste, and Martin Wiebusch, Status of the two-Higgs-doublet model of type II, J. High Energy Phys. 07 (2013) 118.
[45] Nathaniel Craig, The state of supersymmetry after run I of the LHC, arXiv:1309.0528.
[46] O. Buchmueller *et al.*, The NUHM2 after LHC Run 1, Eur. Phys. J. C **74**, 3212 (2014).
[47] K. J. de Vries *et al.*, The pMSSM10 after LHC Run 1, Eur. Phys. J. C **75**, 422 (2015).
[48] Florian Staub, Exploring new models in all detail with SARAH, Adv. High Energy Phys. **2015**, 840780 (2015).
[49] Florian Staub, Reopen parameter regions in two-Higgs doublet models, Phys. Lett. B **776**, 407 (2018).
[50] Peter Athron *et al.*, Status of the scalar singlet dark matter model, Eur. Phys. J. C **77**, 568 (2017).
[51] Manuel E. Krauss and Florian Staub, Perturbativity constraints in BSM models, Eur. Phys. J. C **78**, 185 (2018).
[52] Mark D. Goodsell and Florian Staub, Unitarity constraints on general scalar couplings with SARAH, Eur. Phys. J. C **78**, 649 (2018).

[53] Manuel E. Krauss and Florian Staub, Unitarity constraints in triplet extensions beyond the large s limit, Phys. Rev. D **98**, 015041 (2018).
[54] Mark D. Goodsell and Florian Staub, Improved unitarity constraints in two-Higgs-doublet-models, Phys. Lett. B **788**, 206 (2019).
[55] Florian Staub, Theoretical constraints on supersymmetric models: Perturbative unitarity vs. vacuum stability, Phys. Lett. B **789**, 203 (2019).
[56] Howard Baer, Vernon Barger, Shadman Salam, Dibyashree Sengupta, and Kuver Sinha, Status of weak scale supersymmetry after LHC Run 2 and ton-scale noble liquid WIMP searches, Eur. Phys. J. Special Topics **229**, 3085 (2020).
[57] John Ellis, Keith A. Olive, Vassilis C. Spanos, and Ioanna D. Stamou, The CMSSM survives Planck, the LHC, LUX-ZEPLIN, Fermi-LAT, H.E.S.S. and IceCube, Eur. Phys. J. C **83**, 246 (2023).
[58] Wararat Treesukrat, Kem Pumsa-ard, Nopmanee Supanam, and Patipan Uttayarat, Upper limit on dark matter mass in the inert doublet model, Phys. Rev. D **110**, 115046 (2024).
[59] S. F. King and P. L. White, Nonminimal supersymmetric Higgs bosons at LEP-2, Phys. Rev. D **53**, 4049 (1996).
[60] M. Masip, R. Munoz-Tapia, and A. Pomarol, Limits on the mass of the lightest Higgs in supersymmetric models, Phys. Rev. D **57**, R5340 (1998).
[61] Riccardo Barbieri, Lawrence J. Hall, Anastasios Y. Papaioannou, Duccio Pappadopulo, and Vyacheslav S. Rychkov, An alternative NMSSM phenomenology with manifest perturbative unification, J. High Energy Phys. 03 (2008) 005.
[62] S. F. King, M. Muhlleitner, and R. Nevzorov, NMSSM Higgs benchmarks near 125 GeV, Nucl. Phys. **B860**, 207 (2012).
[63] Laura Lopez Honorez, Emmanuel Nezri, Josep F. Oliver, and Michel H. G. Tytgat, The inert doublet model: An archetype for dark matter, J. Cosmol. Astropart. Phys. 02 (2007) 028.
[64] Riccardo Barbieri, Lawrence J. Hall, and Vyacheslav S. Rychkov, Improved naturalness with a heavy Higgs: An alternative road to LHC physics, Phys. Rev. D **74**, 015007 (2006).
[65] Abdesslam Arhrib, Yue-Lin Sming Tsai, Qiang Yuan, and Tzu-Chiang Yuan, An updated analysis of inert Higgs doublet model in light of the recent results from LUX, PLANCK, AMS-02 and LHC, J. Cosmol. Astropart. Phys. 06 (2014) 030.
[66] Laura Lopez Honorez and Carlos E. Yaguna, A new viable region of the inert doublet model, J. Cosmol. Astropart. Phys. 01 (2011) 002.
[67] M. A. Arroyo-Ureña, R. Gaitan, R. Martinez, and J. H. Montes de Oca Yemha, Dark matter in inert doublet model with one scalar singlet and $U(1)_X$ gauge symmetry, Eur. Phys. J. C **80**, 788 (2020).
[68] Lobsang Dhargyal, Phenomenology of $U(1)_F$ extension of inert-doublet model with exotic scalars and leptons, Eur. Phys. J. C **78**, 150 (2018).
[69] J. F. Gunion, *The Higgs Hunter's Guide*, 1st ed. (CRC Press, 2000).
[70] Igor P. Ivanov, Building and testing models with extended Higgs sectors, Prog. Part. Nucl. Phys. **95**, 160 (2017).
[71] Margarete Mühlleitner, Marco O. P. Sampaio, Rui Santos, and Jonas Wittbrodt, ScannerS: Parameter scans in extended scalar sectors, Eur. Phys. J. C **82**, 198 (2022).
[72] Isabell Engeln, Margarete Mühlleitner, and Jonas Wittbrodt, N2HDECAY: Higgs Boson decays in the different phases of the N2HDM, Comput. Phys. Commun. **234**, 256 (2019).
[73] Genevieve Belanger, Ali Mjallal, and Alexander Pukhov, Recasting direct detection limits within micrOMEGAs and implication for non-standard dark matter scenarios, Eur. Phys. J. C **81**, 239 (2021).
[74] Philip Bechtle, Daniel Dercks, Sven Heinemeyer, Tobias Klingl, Tim Stefaniak, Georg Weiglein, and Jonas Wittbrodt, HiggsBounds-5: Testing Higgs sectors in the LHC 13 TeV era, Eur. Phys. J. C **80**, 1211 (2020).
[75] Philip Bechtle, Sven Heinemeyer, Tobias Klingl, Tim Stefaniak, Georg Weiglein, and Jonas Wittbrodt, HiggsSignals-2: Probing new physics with precision Higgs measurements in the LHC 13 TeV era, Eur. Phys. J. C **81**, 145 (2021).
[76] Jonas Wittbrodt *et al.*, EVADE, https://gitlab.com/jonaswittbrodt/EVADE (2017).
[77] Grigorios Tsoumakas and Ioannis Katakis, Multi-label classification: An overview, Int. J. Data Warehousing Mining **3**, 1 (2007).
[78] André C. P. L. F. de Carvalho and Alex A. Freitas, *A Tutorial on Multi-Label Classification Techniques* (Springer, Berlin, Heidelberg, 2009), pp. 177–195.
[79] Francisco Herrera, Francisco Charte, Antonio J. Rivera, and María J. del Jesus, *Multilabel Classification* (Springer International Publishing, Cham, 2016), pp. 17–31.
[80] Adane Nega Tarekegn, Mario Giacobini, and Krzysztof Michalak, A review of methods for imbalanced multi-label classification, Pattern Recognit. **118**, 107965 (2021).
[81] G. C. Branco, P. M. Ferreira, L. Lavoura, M. N. Rebelo, Marc Sher, and Joao P. Silva, Theory and phenomenology of two-Higgs-doublet models, Phys. Rep. **516**, 1 (2012).
[82] Alexander Belyaev, Giacomo Cacciapaglia, Igor P. Ivanov, Felipe Rojas-Abatte, and Marc Thomas, Anatomy of the inert two Higgs doublet model in the light of the LHC and non-LHC dark matter searches, Phys. Rev. D **97**, 035011 (2018).
[83] Isabell Engeln, Pedro Ferreira, M. Margarete Mühlleitner, Rui Santos, and Jonas Wittbrodt, The dark phases of the N2HDM, J. High Energy Phys. 08 (2020) 085.
[84] Chien-Yi Chen, Michael Freid, and Marc Sher, Next-to-minimal two Higgs doublet model, Phys. Rev. D **89**, 075009 (2014).
[85] Aleksandra Drozd, Bohdan Grzadkowski, John F. Gunion, and Yun Jiang, Extending two-Higgs-doublet models by a singlet scalar field—The case for dark matter, J. High Energy Phys. 11 (2014) 105.
[86] Margarete Muhlleitner, Marco O. P. Sampaio, Rui Santos, and Jonas Wittbrodt, The N2HDM under theoretical and experimental scrutiny, J. High Energy Phys. 03 (2017) 094.
[87] P. M. Ferreira, Margarete Mühlleitner, Rui Santos, Georg Weiglein, and Jonas Wittbrodt, Vacuum instabilities in the N2HDM, J. High Energy Phys. 09 (2019) 006.

[88] Duarte Azevedo, Pedro Gabriel, Margarete Muhlleitner, Kodai Sakurai, and Rui Santos, One-loop corrections to the Higgs boson invisible decay in the dark doublet phase of the N2HDM, J. High Energy Phys. 10 (2021) 044.
[89] Thomas Biekötter, Sven Heinemeyer, José Miguel No, María Olalla Olea, and Georg Weiglein, Fate of electroweak symmetry in the early Universe: Non-restoration and trapped vacua in the N2HDM, J. Cosmol. Astropart. Phys. 06 (2021) 018.
[90] Seraina Glaus, Margarete Mühlleitner, Jonas Müller, Shruti Patel, and Rui Santos, Electroweak corrections to dark matter direct detection in the dark singlet phase of the N2HDM, Phys. Lett. B **833**, 137342 (2022).
[91] Maien Binjonaid, Invisible dark matter decays of a non-standard model like CP-even scalar boson, J. King Saud Univ. Eng. Sci. **36**, 103058 (2024).
[92] K. G. Klimenko, On necessary and sufficient conditions for some Higgs potentials to be bounded from below, Theor. Math. Phys. **62**, 58 (1985).
[93] J. Horejsi and M. Kladiva, Tree-unitarity bounds for THDM Higgs masses revisited, Eur. Phys. J. C **46**, 81 (2006).
[94] Michael E. Peskin and Tatsu Takeuchi, Estimation of oblique electroweak corrections, Phys. Rev. D **46**, 381 (1992).
[95] W. Grimus, L. Lavoura, O. M. Ogreid, and P. Osland, A precision constraint on multi-Higgs-doublet models, J. Phys. G **35**, 075001 (2008).
[96] W. Grimus, L. Lavoura, O. M. Ogreid, and P. Osland, The oblique parameters in multi-Higgs-doublet models, Nucl. Phys. **B801**, 81 (2008).
[97] Johannes Haller, Andreas Hoecker, Roman Kogler, Klaus Mönig, Thomas Peiffer, and Jörg Stelzer, Update of the global electroweak fit and constraints on two-Higgs-doublet models, Eur. Phys. J. C **78**, 675 (2018).
[98] Armen Tumasyan *et al.*, A portrait of the Higgs boson by the CMS experiment ten years after the discovery, Nature (London) **607**, 60 (2022); **623**, E4 (2023).
[99] Georges Aad *et al.*, A detailed map of Higgs boson interactions by the ATLAS experiment ten years after the discovery, Nature (London) **607**, 52 (2022); **612**, E24 (2022).
[100] Aaron Pierce and Jesse Thaler, Natural dark matter from an unnatural Higgs boson and new colored particles at the TeV scale, J. High Energy Phys. 08 (2007) 026.
[101] Wolfgang G. Hollik, Georg Weiglein, and Jonas Wittbrodt, Impact of vacuum stability constraints on the phenomenology of supersymmetric models, J. High Energy Phys. 03 (2019) 109.
[102] In-Kwon Yeo and Richard A. Johnson, A new family of power transformations to improve normality or symmetry, Biometrika **87**, 954 (2000).
[103] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg *et al.*, SCIKIT-LEARN: Machine learning in Python, J. Mach. Learn. Res. **12**, 2825 (2011).
[104] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard *et al.*, TensorFlow: A system for large-scale machine learning, in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (2016), pp. 265–283, https://www.tensorflow.org/.
[105] Keras Team, Keras: Deep learning for humans, https://github.com/keras-team/keras (2015).
[106] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama, Optuna: A next-generation hyperparameter optimization framework, in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (ACM, New York, 2019), pp. 2623–2631, 10.1145/3292500.3330701.
[107] Vinod Nair and Geoffrey E. Hinton, Rectified linear units improve restricted boltzmann machines, in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (Omnipress, Madison, 2010), pp. 807–814.
[108] Andrew L. Maas, Awni Y. Hannun, Andrew Y Ng *et al.*, Rectifier nonlinearities improve neural network acoustic models, in *Proceedings of International Conference on Machine Learning (ICML), Atlanta, GA* (JMLR.org, Atlanta, 2013), Vol. 30, p. 3.
[109] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. **15**, 1929 (2014).
[110] Sergey Ioffe, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv:1502.03167.
[111] Diederik P. Kingma, Adam: A method for stochastic optimization, arXiv:1412.6980.
[112] Timothy Dozat, Incorporating Nesterov momentum into Adam, in *Presented at the ICLR Workshop* (2016), https://openreview.net/pdf?id=OM0jvwB8jIp57ZJjtNEZ.
[113] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri, Activation functions in deep learning: A comprehensive survey and benchmark, Neurocomput. Variable Star Bull. **503**, 92 (2022).
[114] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár, Focal loss for dense object detection, in *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 2980–2988, 10.1109/ICCV.2017.324.
[115] Margherita Grandini, Enrico Bagli, and Giorgio Visani, Metrics for multi-class classification: An overview, arXiv:2008.05756.
[116] Maien Binjonaid, DDP-N2HEDM-MLC, 10.17632/sg699fykng.1 (2025).
[117] https://github.com/drmaien/HEPMLC
[118] https://github.com/drmaien/HEPMLC.git