Studying the validity of the ABCDisCo method of filtering QCD Intanton events from multiple perturbative background theories

Andrei Cristea, Merton College, University of Oxford – 17 October 2022
andrei-alexandru.cristea@merton.ox.ac.uk

1. Abstract

Yang-Mills theories within the Standard Model of particle physics predict topologically non-trivial objects which describe the tunnelling between classically degenerate vacuum states in Minkowski spacetime. Though predicted theoretically as long ago as the 1960's, no experimental evidence for these objects has yet been found. One example of such objects – known as "QCD (quantum chromodynamic) Instantons" – should be frequently created at the Large Hadron Collider, but are still to be identified from the existing data as they are eclipsed by large numbers of background events. This work aimed to deploy the ABCDisCo [1] method of filtering Instanton events from background events on simulated and real experimental data samples. It will also appraise the extent to which the performance of Machine Learning algorithms (neural networks) employed by this method is insensitive to changes in the expected theoretical behavior of the perturbative QCD backgrounds.

2. Introduction

One prediction of the standard model is the existence of pseudo-particles aptly named Instantons – non-perturbative topological objects localized in Minkowski spacetime. These objects are expected to behave as "soft-bombs" of large numbers of low-energy tracks in proton-proton collisions. This is a consequence of Instantons decaying into all kinematically available quark + antiquark pairs as well as multiple gluons. Thus, one expects events presenting Instantons to have more charged tracks and higher sphericity (i.e. distribution of momentum is more spherically symmetric).

Previously, Instanton events could hypothetically be found in recorded LHC data via threshold cuts in phase space. The ABCDisCo method improves this filtering by forming a 2D phase space from the independent outputs of two ML (Machine Learning) neural network classifiers that distinguish between Instanton events and background and applying statistical methods in order to estimate the background in the signal-dominated region. Study [2] determines that neural networks are best suited for this task.

In the Monte Carlo simulated event samples, background events may be generated using different perturbative QCD theories. In reality, it is unclear which ones better suit the real recorded LHC data. This project studies the performance and robustness of the decorrelated ML classifiers used by the ABCDisCo method, under a change between two different theories (EPOS and Pythia) that are used to simulate the background events upon which the algorithms are trained. This study can be further generalized to multiple available background theories. Section 3 describes the scope and method of the project, section 4 outlines the method and results and section 5 presents further matters to be investigated.

3. Background estimation using ABCD method

Collision events are studied by analyzing the charged-particle or debris tracks emerging from the collision vertex. Different physical aspects of these tracks (e.g., momentum, direction etc.) can be used to calculate various event-level features or event shape variables. Therefore, an Instanton events may be recognized by specific behaviors in one or more of these features and variables. This allows the selection of Instanton signal region via threshold cuts in phase space, where a cut is defined as a selection of bounds on event observables in which the desired Instanton processes are more prominent over background processes compared to the uncut sample [2].

The ABCD method allows the estimation of the background within a signal-dominated region as determined by a set of cuts in a 2D phase space, where the features/observables f and g comprising the phase space are approximately statistically independent for the background [1]. Provided that signal contamination is low (see [1] for detailed explanation) in the control regions B, C, D, the background in region A (see Figure 1) can be estimated using

$$N_{A,background}^{estimated} = \frac{N_{B,all} \times N_{C,all}}{N_{D,all}} \quad (1),$$

where N denotes number of events of a certain class (*background*, *signal* or *all*) in a certain region A, B, C or D.

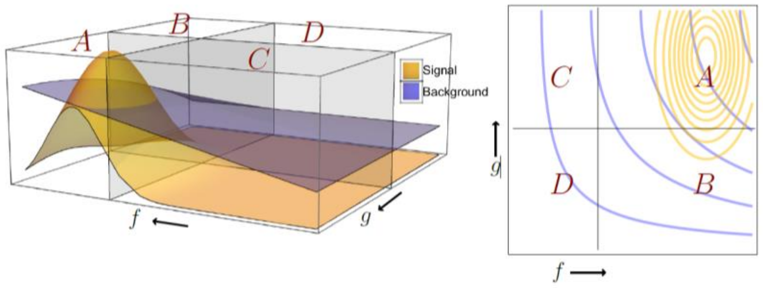

Figure 1 - Phase space of features f *and* g *depicting signal (Instantons) and background; A = signal regions; B, C, D = control regions;* [1]

It should be noted that the features f and g may be selected from the event-level features, but event variables have various degrees of correlation between them, making it difficult to meet the statistical independence condition required for the ABCD method. Additionally, selecting only two features from among the event variables implies neglecting information contained in the rest.

3.1 Double DisCo and phase space generation

A phase space of two approximately statistically independent variables f and g is generated using the outputs of two ML classifiers (filtering signal from background) before imposing the cuts that will define the ABCD regions. Decorrelation between the outputs of the two classifiers is enforced, as explained in [1], via the (double) DisCo method. This involves modifying the second of the two classifier's loss functions to

$$L_{DisCo} = L_{classifier}[f, y] + \alpha \times dCorr^2[f, g]|_{background} \quad (2),$$

where L_{DisCo} is the classifier's newly created loss function, $L_{classifier}[f, y]$ is the classifier's base loss function (e.g., binary cross entropy), f is the classifier's output, g is the feature to be decorrelated against (the previous classifier's output), $y \in \{0,1\}$ is the classification's target, in this case a Boolean (0 for certainly background – bkg. -, 1 for certainly signal – sig.), $dCorr^2[f, g]|_{background}$ is the *squared distance correlation* between f and g, a quantity characterizing correlation (ranging from 0 for no correlation to 1 for perfect correlation) and α is a tunable parameter allowing for various degrees of decorrelation. Notably, we expect that classification performance will decrease with higher imposed decorrelation.

3.2 Classifier setup and event-level features used

The two classifiers g and f (hereafter referred to as classifiers A and B respectively) used to create the 2D phase space are trained to distinguish between Instanton signal events and QCD background on a data sample comprised of a mixture of signal and background. Subsequently, the performance study is executed on corresponding mixed samples, which are completely unused for previous training and hence "unseen" by the classifiers beforehand.

For this project, all ML classifier models, scalers and training are done in Keras [3], as it is an ML framework which allows the use of custom loss functions, unlike other frameworks such as scikit-learn [4]. However, scikit-learn is also used for utilitarian tasks such as splitting the data sets into testing and training samples or testing the algorithms.

The event-level features used in the training of the two classifiers are presented in Table 1. They are either weak discriminants between the signal and background (are very similar between these classes) or strong discriminants between the signal and background (are significantly different between these classes). Classifier A uses only weakly discriminating features, while classifier B uses both weakly and strongly discriminating features. This is done in order to avoid a high systematic correlation between the classifier outputs, which would be difficult to eliminate solely via the DisCo loss function.

Table 1 - Event level features used in the training of the A and B classifiers and their descriptions

Feature	**Description**	**Classifier A** (weakly discriminating)	**Classifier B**
Number of tracks (track multiplicity)	Recorded number of charged tracks of an event	No	Yes
ST	Sum of scalar transverse 3-momenta (momenta of tracks in transverse plane of event); for formula see [2]	No	Yes
Invariant mass	Sum of invariant masses of all track 4-momenta	No	Yes
Invariant mass per track	Invariant mass divided by number of event tracks	No	Yes
Transverse mass squared	Invariant mass if all track 4-momenta were projected onto the transverse plane and considered null; for formula see [2]	No	Yes
Transverse mass squared/track	Transverse mass squared divided by the number of event tracks	No	Yes
Magnitude of mean pseudorapidity	Absolute value of average of pseudorapidity over all event tracks $= \|\langle\eta\rangle\|$	Yes	Yes
Sphericity scalars (S – sphericity, A – Aplanarity, C and D)	Scalars characterizing track momenta spatial distribution; for formulas see [2]	Yes	Yes
2D transverse Thrust	Event shape variable characterizing jets produced in collision; for formula see [2]	Yes	Yes
Broadening	Event shape variable providing additional information about spatial positioning of collision tracks; for formula see [2]	Yes	Yes

4. Method and results

Multiple perturbative QCD theories exist for the Monte Carlo simulated background samples, but none of them perfectly reflect the real behavior of background events. Ideally however, the ML classifiers used to implement the above-described methods would be optimized to perform just as well on real data as on simulated data, after being trained on one or more background theories. In other words, for the ML classifiers to be useful, they need to be insensitive to slight theoretical variations between the behaviors of simulated and real background data. If they are not, the possibility of improving on this aspect is to be investigated.

An equivalent study is done to determine the degree to which the ML algorithms are or can be made insensitive to theoretical variations between two Monte Carlo simulated backgrounds: EPOS and Pythia.

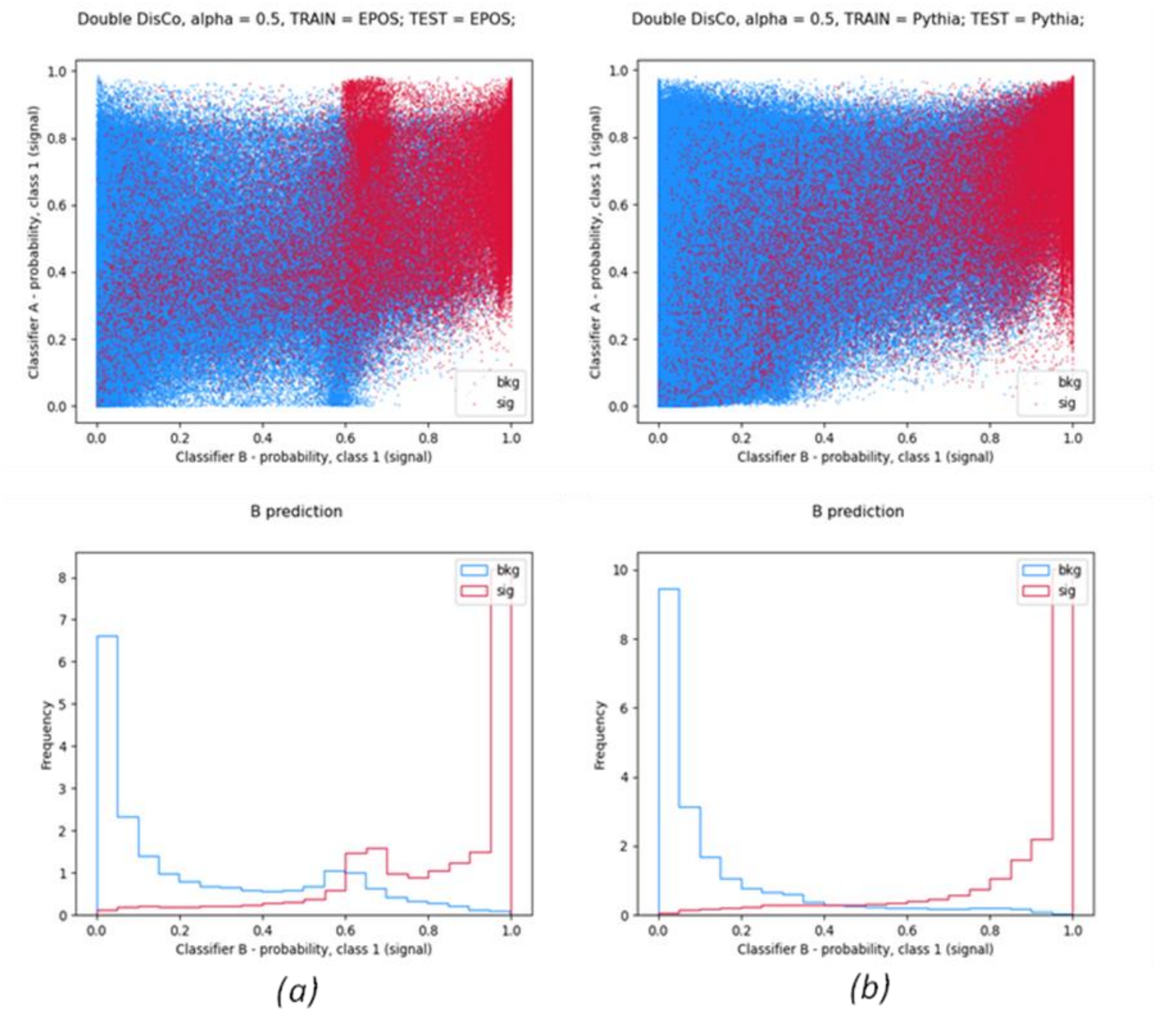

Figure 2 - scatter plots in phase space and classifier B cross-section for models trained and tested on the same background; red=sig, blue=bkg; (a) - EPOS training, EPOS testing, (b) - Pythia training, Pythia testing;

4.1 Method

To analyze how insensitive the ML algorithms can be to background variations, the DisCo decorrelated A and B neural network classifiers are trained and then used on simulated testing samples (signal +

background) to create scatter plots of the 2D phase space needed to apply the ABCD method. All data events are produced at center of mass (CM) energies of 13 TeV. All Instanton mass regime samples (> 20Gev, > 50Gev) were integrated in the final signal sample. Each event on the scatter plot is weighed using its theoretical weight, i.e., how many real events it corresponds to. The integrated luminosity used in the calculation of these weights is $L = 1\ pb^{-1}$. Scores and performance metrics are calculated using this weight, as the number of equivalent real events contained in that data point. Subsequently, the ABCD method is applied for a series of cuts defining the signal region and overall performance is evaluated using the metrics defined in 4.2. For clarity, a pair of ML classifier algorithms will be referred to as a model.

Firstly, a model is trained using the EPOS background sample and is tested on EPOS background. Similarly, another model is trained using the Pythia background sample and is tested on Pythia background. This is done as a consistency check and shows that the models are indeed viable for the ABCDisCo method. Representations of the scatter plots obtained are shown in Figure 2. These models are then cross-tested using the opposing background theories instead of their respective ones, i.e., the EPOS trained model is tested on the Pythia background and vice-versa. This is done to analyze how sensitive (or rather insensitive) the models are to variations in the background. The scatter plots obtained are shown in Figure 3.

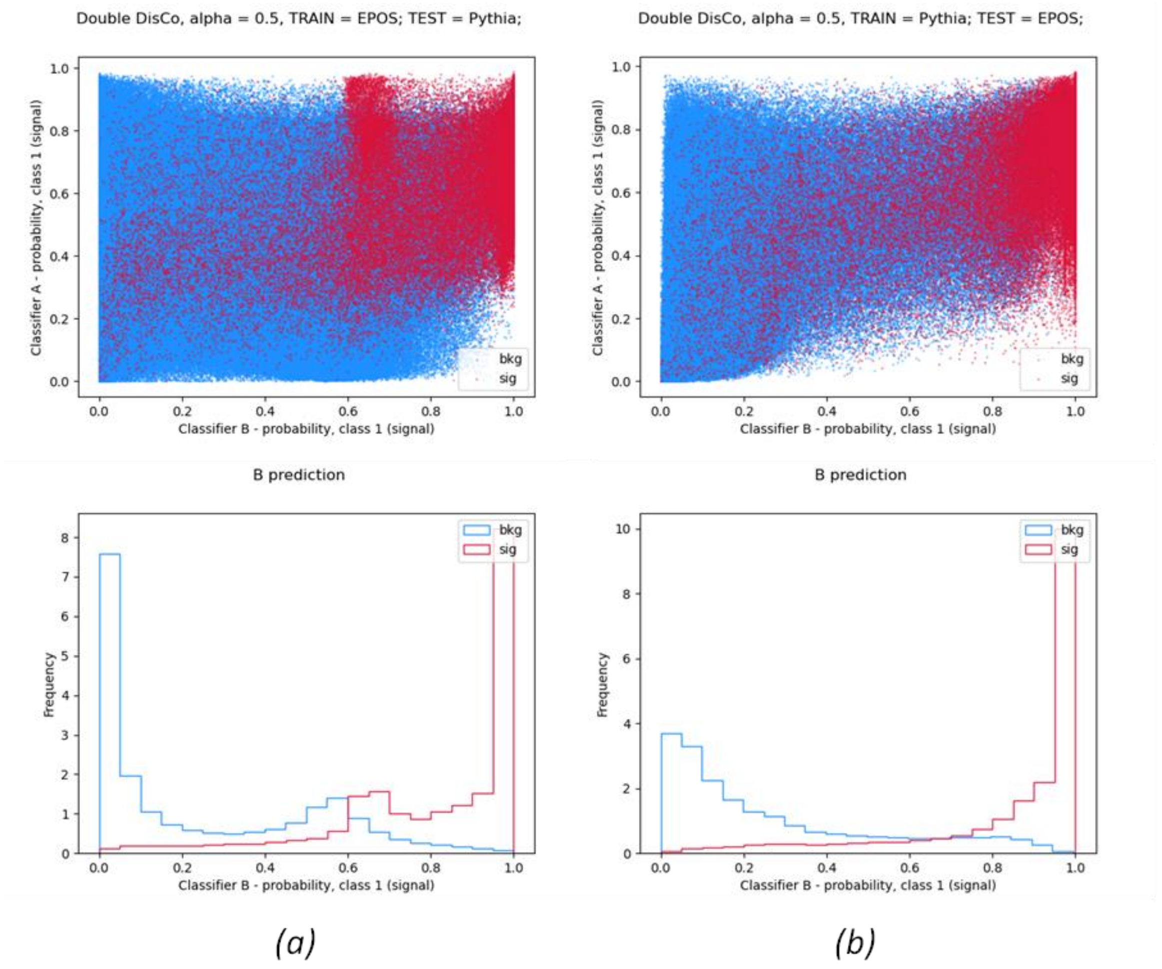

Figure 3 - scatter plots in phase space and classifier B cross-section for models trained and cross-tested on different backgrounds; red=sig, blue=bkg; (a) - EPOS training, Pythia testing, (b) - Pythia training, EPOS testing;

Finally, a third model is trained using both the EPOS and the Pythia sample and is then tested on each of the two theories separately. This step is executed under the proposal that training the model on all possible background theories will give better results in testing, even if real data corresponds more closely to one, or a hybrid of the training backgrounds. Intuitively, this one can argue this to be true by saying that if the classifiers learn from a sufficient number of background events corresponding to a sufficient number of theories, it will make contact with a sufficient number of events that accurately emulate real data. The results are plotted in Figure 4.

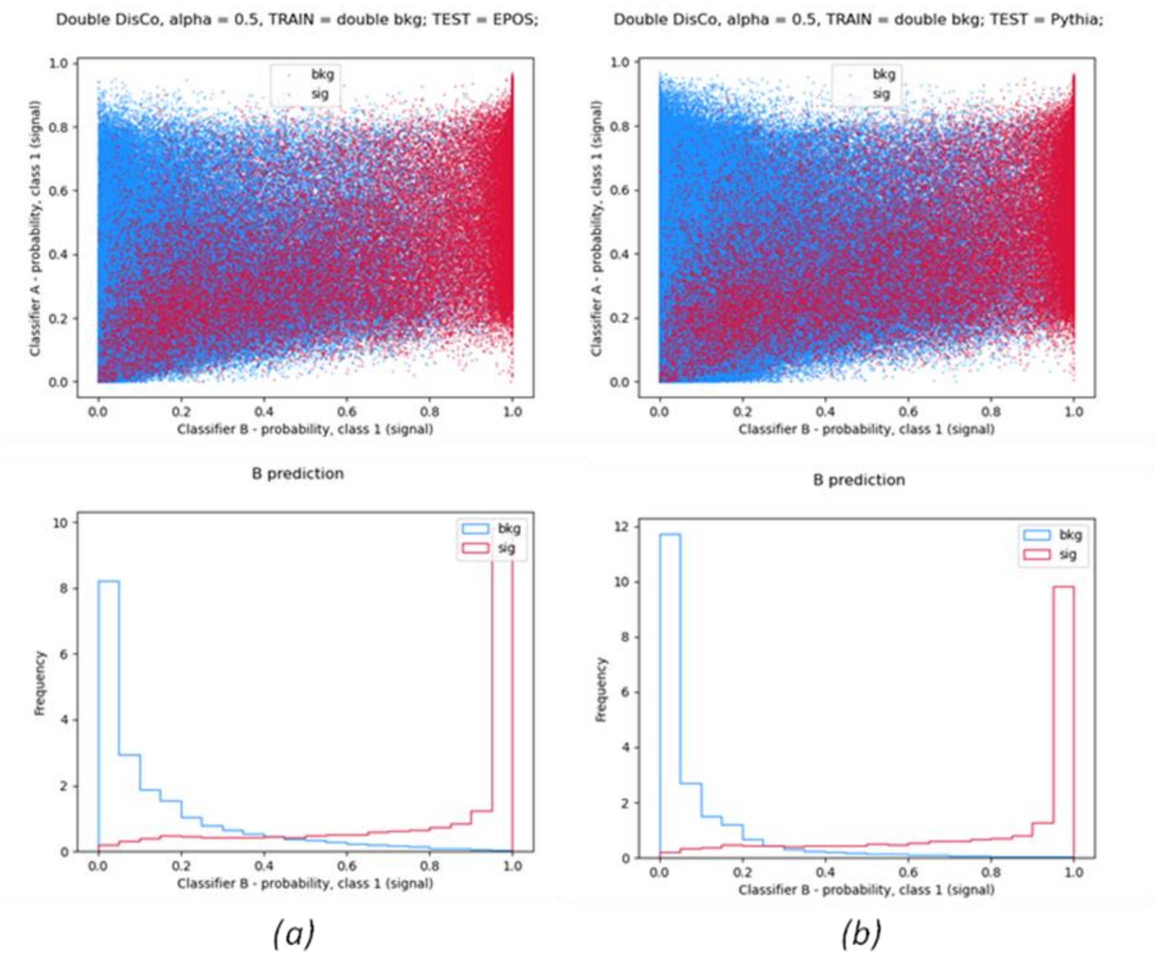

Figure 4 - scatter plots in phase space and classifier B cross-section for models trained on BOTH background theories and tested on each; red=sig, blue=bkg; (a) - double training, EPOS testing, (b) - double training, Pythia testing;

Following the study done in [2], both classifiers, in each of the models, are consistently trained with a decorrelation parameter α set to a value of 0.5, for optimal classification-decorrelation ratio. Both classifiers are trained for 50 epochs each time and are always using the same signal sample for training. It is only the background sample that changes. The training and testing fractions of the total samples are kept separate throughout the project. A final important aspect of the training is that the event samples are imbalanced. In this case, the background samples are significantly larger than the signal sample. This can create a large bias towards the background, meaning that the classifiers will be unable to properly distinguish the Instanton events, due to the signal population being negligible by comparison to the background one. This is fixed by assigning weights to each sample, indicating the importance of each event in training. The weights assigned to events in a sample are identical and are equal to the inverse of the number of events in the given sample, normalized to the number of events in some chosen reference sample (e.g., if the background is twice as large as the signal, the background weights will be ½ of the signal weights). This discrepancy in sample sizes will also prevent the use of classical ML metrics such as accuracy, recall, precision, confusion matrices or ROC curves, as these are affected by sample length bias and are unsuitable for unbalanced data sets (class 0, bkg. is much larger or smaller than class 1, sig.). The alternative metrics used are described in the next section.

4.2 Metrics and performance

Following the above models training and testing, a series of cuts defining the signal region in the 2D phase space are generated. This is necessary because there is no clear optimal signal region so multiple sensible guesses may be viable. For each model and for each cut, the signal in the signal region is recorded. Similarly, in the case of both the EPOS and the Pythia testing, the background contaminations in the signal region are recorded and the appropriate metrics are calculated to determine the classifiers' and the models' performance. Note, as mentioned before, the numbers recorded or estimated in this part of the analysis correspond to the theoretical weights and are the equivalent numbers of real events.

Lastly, this procedure is repeated using the ABCD estimated background contaminations in the signal region. The relevant metrics are computed once again and compared to the values obtained using the recorded background contamination, in order to confirm that the ABCD method is indeed efficient.

The metrics used are the following[1]:

- **Sigma**: $\sigma = \frac{S}{\sqrt{S+\bar{B}+\sigma_B{}^2}}$, where S is the signal in the signal region, $\bar{B} = \frac{B_{EPOS}+B_{Pythia}}{2}$ is the mean background contamination between the EPOS and the Pythia testing of the model and $\sigma_B = \frac{|B_{EPOS}-B_{Pythia}|}{2}$ is the systematic error between the EPOS and the Pythia testing. For the model to be considered good, this metric is ideally > 5 for the corresponding cut defining the signal region.
- $\boldsymbol{\frac{\sigma_B}{\bar{B}}}$: the value of the ratio of the systematic error to the mean background contamination is < 0.1 for a well-performing model and for an adequate signal region cut.
- $\boldsymbol{\frac{S}{\bar{B}}}$: for a well-performing model and for an adequate signal region cut, the signal should be several times larger than the mean background contamination (e.g., > 4 but no strict value).

4.3 Results

The results following the analysis are the values obtained for the above-described metrics. These are presented for multiple signal region cuts, for each of the three models - EPOS, Pythia and double-background trained - in the Table 2, 3 and 4 respectively.

It can be seen that for certain cuts, generally the ones above approximately [0.7] for the B classifier, the metric scores correspond to what is understood as a good performance of the model, as indicated in section 4.2. Furthermore, the general trend is that performance increases with increasing values of the B classifier cut. Similarly, higher A classifier cuts for the signal region tend to generally provide better metric values, but are not as impactful as their B counterpart (this is expected as classifier A only uses weak discriminants).

[1] B_{EPOS} and B_{Pythia} are the background contaminations in the signal region, recorded or ABCD estimated by case.

However, it is important to note to excessively restrictive cuts on the signal region (e.g., $[\mathrm{A},\mathrm{B}]_{cuts} = [0.9, 0.9]$) may lead to erroneous or invalid results. This effect can be seen in the last three rows of Table 4, where the values taken by the various metrics are unrealistic – e.g., the values of the measured σ suddenly shoot up more than expected, while those of σ_{est} break the ascending trend - or do not exist, as a consequence of division by zero. A similar occurrence is also present in the last few lines of the σ column in Table 2. This happens because no background events are registered under the imposed cuts, as a result of them being too restrictive, or the behavior of that classifiers in that area is not representative of the whole phase space.

Table 2 - EPOS trained model metric results for various classifier [A, B] cuts. The index "est" refers to values estimated by ABCD method, while the others are recorded values; $\overline{\boldsymbol{B}} = \frac{B_{EPOS}+B_{Pythia}}{2}$, $\sigma_B = \frac{|B_{EPOS}-B_{Pythia}|}{2}$, where B_{EPOS} and B_{Pythia} are the background predictions in the two different testing cases.

$[\mathbf{A},\mathbf{B}]_{cuts}$	$\boldsymbol{\sigma}$	$\frac{\boldsymbol{\sigma_B}}{\overline{\boldsymbol{B}}}$	$\frac{\boldsymbol{S}}{\overline{\boldsymbol{B}}}$	$\boldsymbol{\sigma_{est}}$	$\frac{\boldsymbol{\sigma_B}}{\overline{\boldsymbol{B}}}_{est}$	$\frac{\boldsymbol{S}}{\overline{\boldsymbol{B}}_{est}}$
[0.3, 0.6]	1.46047	0.128548	0.187741	9.409089	0.07832	0.736924
[0.3, 0.7]	2.413413	0.130107	0.314002	14.42553	0.155893	2.248833
[0.3, 0.8]	5.047329	0.100702	0.508275	41.95013	0.134706	5.65093
[0.3, 0.9]	27.89286	0.040044	1.116937	527.0946	0.053862	28.40352
[0.4, 0.6]	1.641433	0.126995	0.208453	5.791429	0.103451	0.59913
[0.4, 0.7]	2.76354	0.123012	0.339949	9.107397	0.14422	1.313473
[0.4, 0.8]	5.660431	0.095757	0.542024	23.79058	0.106246	2.527648
[0.4, 0.9]	35.37698	0.033075	1.170106	126.035	0.057507	7.248073
[0.5, 0.6]	1.986073	0.122797	0.243884	5.225905	0.108541	0.567223
[0.5, 0.7]	3.203208	0.120205	0.385041	9.008521	0.123549	1.112997
[0.5, 0.8]	6.055263	0.099326	0.601443	25.0878	0.079023	1.982509
[0.5, 0.9]	28.14346	0.044905	1.263783	426.4693	0.011432	4.877439
[0.6, 0.6]	2.480699	0.116598	0.289245	5.14239	0.115837	0.59568
[0.6, 0.7]	3.982131	0.110992	0.441985	9.049125	0.122221	1.105992
[0.6, 0.8]	7.140024	0.0962	0.686873	24.27579	0.077156	1.873015
[0.6, 0.9]	29.5896	0.048855	1.445609	444.1734	0.009253	4.11217
[0.7, 0.6]	3.24703	0.112038	0.363789	5.523462	0.120005	0.662843
[0.7, 0.7]	5.353788	0.099983	0.53529	9.709737	0.122346	1.187944
[0.7, 0.8]	10.41693	0.078397	0.816653	23.24728	0.083846	1.949199
[0.7, 0.9]	43.01788	0.039158	1.68449	201.2078	0.020171	4.059141
[0.8, 0.6]	5.265031	0.098702	0.519671	6.280544	0.130846	0.821784
[0.8, 0.7]	7.856841	0.09414	0.739643	11.41208	0.123749	1.412237
[0.8, 0.8]	15.74094	0.072092	1.134794	25.85261	0.086792	2.243818
[0.8, 0.9]	53.92848	0.040987	2.210447	164.6859	0.02773	4.567754
[0.9, 0.6]	27.67053	0.051651	1.429311	5.123006	0.177354	0.908587
[0.9, 0.7]	37.18029	0.057825	2.150238	9.066924	0.144572	1.310833
[0.9, 0.8]	16.93019	0.187981	3.182641	20.52562	0.100551	2.063959
[0.9, 0.9]	24.34425	0.195162	4.751353	87.17354	0.048686	4.247378

The above-described results indicate that the methods used are viable, provided an appropriate choice for the ABCD regions is made and that the models are somewhat, but not fully insensitive to changes in background theories between training and testing, as the metric scores' values are not consistently above the ideal threshold. Notably, the double-background training performs better overall than both the EPOS and Pythia models, having higher metric scores than these. This confirms the assumption that training on multiple possible background theories increases the efficiency and reduces the sensitivity to changes in the testing background theory.

Table 3 - Pythia trained model metric results for various classifier [A, B] cuts. The index "est" refers to values estimated by ABCD method, while the others are recorded values; $\overline{\boldsymbol{B}} = \frac{B_{EPOS}+B_{Pythia}}{2}$, $\sigma_B = \frac{|B_{EPOS}-B_{Pythia}|}{2}$, *where* B_{EPOS} *and* B_{Pythia} *are the background predictions in the two different testing cases.*

$[\mathbf{A},\mathbf{B}]_{cuts}$	$\boldsymbol{\sigma}$	$\frac{\sigma_B}{\overline{B}}$	$\frac{S}{\overline{B}}$	$\boldsymbol{\sigma_{est}}$	$\frac{\sigma_B}{\overline{B}}_{est}$	$\frac{S}{\overline{B}_{est}}$
[0.3, 0.6]	1.219	0.145499	0.177363	8.944177	0.178882	1.59995
[0.3, 0.7]	1.648924	0.140114	0.231037	14.28057	0.167755	2.395631
[0.3, 0.8]	2.902658	0.122693	0.356137	29.51513	0.162513	4.796592
[0.3, 0.9]	16.15981	0.061749	0.997845	276.1041	0.076038	20.99657
[0.4, 0.6]	1.30619	0.141738	0.185136	5.789327	0.158849	0.919631
[0.4, 0.7]	1.767578	0.13582	0.240072	8.213062	0.155654	1.278399
[0.4, 0.8]	3.098447	0.118962	0.368599	16.09337	0.134659	2.167122
[0.4, 0.9]	16.90913	0.059861	1.012194	116.7113	0.068224	7.962673
[0.5, 0.6]	1.470527	0.137284	0.20188	4.912673	0.146487	0.719644
[0.5, 0.7]	1.964141	0.13219	0.259641	6.97958	0.137621	0.960538
[0.5, 0.8]	3.433113	0.11508	0.395083	13.01894	0.11759	1.530904
[0.5, 0.9]	18.83508	0.055962	1.054044	96.89688	0.049283	4.775467
[0.6, 0.6]	1.686693	0.137737	0.23232	4.770233	0.144685	0.69018
[0.6, 0.7]	2.199901	0.134568	0.296036	6.857731	0.132024	0.905388
[0.6, 0.8]	3.652126	0.121397	0.443356	13.36957	0.105182	1.406234
[0.6, 0.9]	16.52951	0.069344	1.146223	145.4943	0.02676	3.893626
[0.7, 0.6]	2.015227	0.139784	0.281696	5.370871	0.138935	0.746201
[0.7, 0.7]	2.574282	0.138124	0.35557	7.645695	0.125988	0.963262
[0.7, 0.8]	4.260728	0.122914	0.523704	14.34788	0.102188	1.466188
[0.7, 0.9]	14.43372	0.091411	1.319398	182.5441	0.020238	3.694736
[0.8, 0.6]	2.499443	0.150588	0.376386	7.950193	0.124381	0.988854
[0.8, 0.7]	3.102116	0.149208	0.46286	11.10976	0.114042	1.266977
[0.8, 0.8]	4.626764	0.139199	0.644039	20.5687	0.091137	1.874577
[0.8, 0.9]	15.72947	0.094607	1.488116	162.7831	0.026967	4.390331
[0.9, 0.6]	3.023568	0.212475	0.642433	18.90861	0.090437	1.710056
[0.9, 0.7]	3.115674	0.232158	0.723328	29.71581	0.075801	2.25252
[0.9, 0.8]	3.814249	0.224017	0.854457	54.26459	0.062872	3.411929
[0.9, 0.9]	7.359736	0.22939	1.688249	643.9025	0.011444	7.432676

5. Further work

Further attempts at improving the method were made but not finalized throughout the course of this project. They are described below as possible useful insight into ML methods and as suggestions for further work on the topic of this project.

Table 4 - Double (EPOS + Pythia) trained model metric results for various classifier [A, B] cuts. The index "est" refers to values estimated by ABCD method, while the others are recorded values; $\overline{\boldsymbol{B}} = \frac{B_{EPOS}+B_{Pythia}}{2}$, $\sigma_B = \frac{|B_{EPOS}-B_{Pythia}|}{2}$, *where* B_{EPOS} *and* B_{Pythia} *are the background predictions in the two different testing cases.*

$[\mathbf{A},\mathbf{B}]_{cuts}$	σ	$\frac{\sigma_B}{\overline{B}}$	$\frac{S}{\overline{\overline{B}}}$	σ_{est}	$\frac{\sigma_B}{\overline{B}}_{est}$	$\frac{S}{\overline{\overline{B}}_{est}}$
[0.3, 0.6]	4.005337	0.119322	0.477924	15.14583	0.116299	1.761452
[0.3, 0.7]	7.171476	0.102407	0.734409	31.03176	0.087396	2.712064
[0.3, 0.8]	15.32172	0.084162	1.28951	111.1039	0.041979	4.664171
[0.3, 0.9]	37.86216	0.080793	3.059002	1104.8	0.007138	7.907243
[0.4, 0.6]	4.632487	0.116733	0.540764	15.68435	0.103923	1.62997
[0.4, 0.7]	8.052455	0.103031	0.829655	29.74915	0.081702	2.430578
[0.4, 0.8]	16.80957	0.087443	1.469873	93.62958	0.042266	3.957456
[0.4, 0.9]	39.81404	0.086768	3.454593	4935.192	0.001237	6.532097
[0.5, 0.6]	5.08865	0.122028	0.620958	15.41241	0.107474	1.656437
[0.5, 0.7]	8.770339	0.10861	0.952546	28.75495	0.084387	2.426564
[0.5, 0.8]	17.65586	0.095055	1.678284	85.47578	0.045162	3.860332
[0.5, 0.9]	48.81745	0.081654	3.986187	568.4091	0.010916	6.211525
[0.6, 0.6]	5.637241	0.128911	0.726701	17.27444	0.104081	1.797941
[0.6, 0.7]	11.44491	0.09805	1.122169	32.73809	0.079683	2.608693
[0.6, 0.8]	24.61991	0.080548	1.983083	82.18574	0.049076	4.033451
[0.6, 0.9]	60.67443	0.077979	4.73139	494.2883	0.012806	6.336786
[0.7, 0.6]	8.664021	0.102313	0.886443	19.26655	0.109802	2.11552
[0.7, 0.7]	16.11234	0.0845	1.3615	35.63124	0.08529	3.039023
[0.7, 0.8]	28.68815	0.082985	2.380691	88.58106	0.052037	4.609763
[0.7, 0.9]	42.35568	0.131995	5.590802	404.8346	0.017347	7.03169
[0.8, 0.6]	12.07591	0.112744	1.361499	20.6682	0.1548	3.199476
[0.8, 0.7]	19.7207	0.109697	2.163332	38.50912	0.120587	4.643888
[0.8, 0.8]	32.27366	0.10631	3.431129	81.99241	0.085521	7.013359
[0.8, 0.9]	27.49399	0.300349	8.257983	205.5792	0.051328	10.56475
[0.9, 0.6]	13.31957	0.404194	5.384259	26.23424	0.15072	3.955727
[0.9, 0.7]	254.5278	1	263.2215	70.15751	0.082712	5.819647
[0.9, 0.8]	454.2918	1	511.9102	162.737	0.05539	9.153215
[0.9, 0.9]	986.5931		inf	425.4629	0.033098	15.72229

5.1 PCA – Principal component analysis

Principal component analysis (PCA) is a technique from statistics for simplifying a data set [5]. It was developed by Karl Pearson (1901) and Harold Hotelling (1933). It is generally used in ML methods as a pre-processing step, in order to reduce the dimensionality of a data set, while keeping all relevant information, from all the features comprising the original phase space. This implies creating a new set of variables (that are not necessarily physically interpretable) from the original ones and only keeping the most significant of these newly created features. More explicitly, PCA is a linear transformation that transforms the data to a new coordinate system such that the new set of variables, the principal components, are linear functions of the original variables, are uncorrelated, and the greatest variance by any projection of the data comes to lie on the first coordinate, the second greatest variance on the second coordinate, and so on. In practice, this is achieved by computing the covariance matrix for the full data set. Next, the eigenvectors and eigenvalues of the covariance matrix are computed, and sorted according to decreasing eigenvalue [5].

It should be noted, however, that the use of PCA is not always appropriate, as some variables presenting low variance might have high predictive value. Thus, appropriate evaluation of feature relevance is imperatively required before applying PCA.

An algorithm applying PCA to the data sets was successfully created in the course of this project, but was ultimately not applied. This is because feature relevance (importance) is difficult to analyze for imbalanced data sets and thus proved too long of a task for the time allocated to this work. Nonetheless, PCA could possibly improve the performance of the ABCDisCo method by improving the classification power of the ML classifiers.

5.2 ICA – Independent component analysis

Independent component analysis (ICA) is a method for automatically identifying the underlying factors in a given data set [6]. Essentially, it distinguishes between different independent sources in mixed data-sets or variables (e.g., two voices overlapping picked up by two different microphones can be disentangled via ICA). Similar to the ABCDisCo method, it can be implemented through ML algorithms. A study of this method was proposed during the course of this project as an alternative to the ABCDisCo method for filtering Instanton events. Subsequently, the distinguished signals were to be compared with the ones found in the simulated data sets.

The main problems with implementing ICA were the multidimensionality of the data set (several event level features) and the number of degrees of freedom in the data set as opposed to the receptor (i.e., 2 degrees of freedom – signal and background – but only 1 entangled data set[2]). These difficulties are solved independently in [7], [8] and [9], but their unification surpassed my current expertise.

5.3 Decorrelation artifacts

As can be seen in the top-left corner of Figure 2., in the red signal scatterplot, higher degrees of decorrelation can begin to create artifacts (thick red line at ~ [0.7 B]), as a consequence of a reduction in classifying capacity. This happens if the classifier's loss function attributes a higher importance to decorrelation than classification and usually leads to the creation of a region where higher than expected signal and background densities overlap, as in the case described above.

On a case-by-case basis, the location of these artifacts can be useful or detrimental in determining an appropriate signal region and achieving overall satisfying results. It is certainly worth investigating whether adequate predictions of artifact locations can be made and whether their effects, in conjunction with a higher decorrelation parameter, can be exploited to boost the performance of the ABCDisCo method.

6. Conclusions

Machine learning methods, particularly neural network classifiers, are proven to be adequate and reliable in applying the ABCDisCo method to filtering Instanton events from QCD backgrounds, with metric scores consistently close to their desired values, provided appropriate cuts for the signal region are imposed (e.g., $[\mathrm{A}, \mathrm{B}]_{cuts} = [0.7, 0.8]$ in the cases presented above).

In this context, the ML classifiers show some degree of insensitivity to changes in the background theory used to train them (EPOS or Pythia here), retaining their satisfying performance even in cross-testing

[2] It would seem like a simple solution is to split the data set, but the signals to distinguish are the Instantons and the backgrounds, not the individual events, which are independent from each other. Splitting the data set would just duplicate the problem to be solved, as it increases the number of degrees of freedom on each side.

trials while providing the expected signal shapes. However, a sizeable improvement of this analysis can be made by feeding the classifiers multiple background theories in the testing stage. As a result, discrepancies between the simulated background events and real data can be mitigated via training on multiple suggested theoretical models.

Notably, careful pre-processing is required if the simulated training data sets are imbalanced and an appropriate choice of signal region must be made. An additional consequence of the data sets' imbalance is the implementation of the new σ, $\frac{\sigma_B}{B}$ and $\frac{S}{B}$ scores.

Suggestions for further analysis and improvements are made in the latter sections, such as implementing a PCA pre-processing step. Additionally, studies of decorrelation and the phase-space artifacts that it produces could help improve the performance of DisCo classifier models on a case-to-case basis.

7. Acknowledgements

I would like to thank Professor Alan Barr for offering me the opportunity of contributing to the field of Particle Physics research and for guiding me through this summer project with great patience and excitement. I would also like to thank Ynyr Harris for his constant advice and for providing me with the necessary data to carry out the analysis. Finally, this project was funded by Merton College through the Summer Project scheme, without which it would not have been possible.

References

[1] Kasieczka, Gregor & Nachman, Benjamin & Schwartz, Matthew & Shih, David. (2020). ABCDisCo: Automating the ABCD Method with Machine Learning.

[2] Nicholas Mitchell. (2021). Investigating the improvement of Instanton process classification via machine-learning methods

[3] Keras (Python library). https://keras.io/, 2020.

[4] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011

[5] Sewell M. Principal Component Analysis. London: University College London; 2008

[6] James V. Stone, Independent component analysis: an introduction, Trends in Cognitive Sciences, Volume 6, Issue 2, 2002, Pages 59-64, ISSN 1364-6613,https://doi.org/10.1016/S1364-6613(00)01813-1, (https://www.sciencedirect.com/science/article/pii/S1364661300018131)

[7] Kim, T., Eltoft, T., Lee, TW. (2006). Independent Vector Analysis: An Extension of ICA to Multivariate Components. In: Rosca, J., Erdogmus, D., Príncipe, J.C., Haykin, S. (eds) Independent Component Analysis and Blind Signal Separation. ICA 2006. Lecture Notes in Computer Science, vol 3889. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11679363_21

[8] J. . -F. Cardoso, "Multidimensional independent component analysis," Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181), 1998, pp. 1941-1944 vol.4, doi: 10.1109/ICASSP.1998.681443.

[9] M. Xue, Y. Xiaolong, H. Jiashuang, M. Shilin and J. WenTian, "Separation of single-channel mixed signals based on the frequency-division of a convolution-type wavelet packet," The 27th Chinese Control and Decision Conference (2015 CCDC), 2015, pp. 5606-5611, doi: 10.1109/CCDC.2015.7161798.