DEUTSCHES ELEKTRONEN-SYNCHROTRON
Ein Forschungszentrum der Helmholtz-Gemeinschaft

DESY 20-073
arXiv:2005.00568
May 2020

Adversarial Domain Adaptation to Reduce Sample Bias of a High Energy Physics Classifier

J. M. Clavijo, P. Glaysher, J. M. Katzy
Deutsches Elektronen-Synchrotron DESY, Hamburg

ISSN 0418-9833

NOTKESTRASSE 85 - 22607 HAMBURG

To be sure that your reports and preprints are promptly included in the HEP literature database
send them to (if possible by air mail):

DESY Zentralbibliothek Notkestraße 85 22607 Hamburg Germany	DESY Bibliothek Platanenallee 6 15738 Zeuthen Germany

Prepared for submission to JINST

Adversarial domain adaptation to reduce sample bias of a high energy physics classifier

J.M. Clavijo,* **P. Glaysher*** **and J.M. Katzy***

Deutsches Elektronen-Synchrotron DESY
Notkestraße 85, 22607 Hamburg, Germany

E-mail: judith.katzy@desy.de

Abstract: We apply adversarial domain adaptation to reduce sample bias in a classification machine learning algorithm. We add a gradient reversal layer to a neural network to simultaneously classify signal versus background events, while minimising the difference of the classifier response to a background sample using an alternative MC model. We show this on the example of simulated events at the LHC with $t\bar{t}H$ signal versus $t\bar{t}b\bar{b}$ background classification.

Keywords: Data Processing, machine learning, supervised learning, adversarial network, adversarial domain adaptation, discriminator network, gradient reversal layer, neural network, training bias, Higgs boson, ttH, LHC

*All authors contributed equally.

Contents

1 Introduction

Many measurements and searches for new phenomena performed by the experiments at the Large Hadron Collider (LHC) use a classification algorithm, such as Boosted Decision Trees or Neural Networks, to discriminate the physics process of interest (signal) from other physics processes with similar signature (background). The algorithms are optimized using supervised training on detailed simulated Monte Carlo (MC) data sets, labeled as signal or background. The resulting classifier is applied to unlabeled data to separate signal and background, and measure the statistical significance of the signal or its strength, assuming that the simulated and the real data sets are identically distributed.

However, differences between real and simulated data sets always exist and the learner may pick up a discriminating feature which differs between the data sets, introducing a bias to the sample used for training. This problem is similar to that of visual recognition where training is performed on simulated pictures, the so-called source domain and applied to real photographs, the target domain. In order to avoid training specific to the source domain, algorithms of domain adaptation have been developed. In this paper, we apply the method of domain adaptation to high energy physics data.

In this paper we present a Domain Adversarial Neural Network (DANN) to classify events in the search for the $t\bar{t}H(H \rightarrow b\bar{b})$ process at the LHC, which is very rare and hard to separate from the $t\bar{t}$ + jets background [1]. The effect on the final result caused by the bias to the specific background model used for training is estimated using an alternative simulation, based on a different

physics model, which was not used for training. The difference in response is taken as uncertainty on the classification. This uncertainty happens to be the largest on the measurement, hampering the observation of the searched process. Therefore a solution to minimize this sample bias is of high importance. For the study presented here, the two background simulations correspond to the different domains. The domain adaptation is applied to reduce mentioned training bias.

The network structure consists of a common feature selector part and separate branches for label classification and domain adaptation, implemented via a gradient reversal layer as presented in [2]. This network structure differs from other adversarial approaches by including domain adaptation in the learning process via the common feature selector as proposed on theoretical grounds in [3]. This way, the network extracts discriminant features for the classification that are at the same time model-invariant. The use case presented here differs from [2] in the use of a set of features instead of images as input and the use of a bigger and more complex data set. Additionally, we make detailed performance analyses, evaluating the influence of several hyper-parameters and also exploring some issues that appear for this kind of architecture.

Adversarial classifiers were used in high energy physics before but not for domain adaptation, e.g. the decorrelation of a jet tagger from the jet mass [4] and the tuning of a classifier against a nuisance parameter [5].

In this paper we describe the network used in Sec. 2, followed by the details of the data sets used in Sec. 3. We systematically study the dependence on hyper parameters in Sec. 4, including some issues observed during the training. In Sec. 5, we expose the performance through different figure of merits related to physics searches and we include a feasibility study for a potential use with real unlabeled data. Finally, a summary and some conclusions are given in Sec. 6.

2 The Deep Adversarial Neural Network

We follow the architecture presented in [2] with a feed-forward neural network composed of three parts as shown in Fig. 1: a *feature extractor* which splits into the *label predictor*, performing the signal-background classification, and the *domain classifier*, that allows the domain adaptation. The last two parts works adversarially. For training and testing we have two data sets (domains): source and target, both containing signal and background events. The target domain is constructed as pseudo-data, meaning that it is treated as unlabeled and it has a signal to background proportion similar to the one expected in a real data sample. For measuring our algorithm performance we make use of the target labels in the final test.

For the label classification we train the network only using events from the source domain. The gate layer stops the target events propagation making the *label predictor* loss being evaluated only on the source events. This allows training the network on mixed samples of both domains. The classification is adapted to the target domain by connecting the *feature extractor* with the *domain classifier* through a gradient reversal layer. This layer does nothing during the forward propagation but inverts the sign of the gradients during the backpropagation. The *domain classifier* is trained to determine which domain the events belong to. Due to the gradient reversal, the *feature extractor* will be trained to not provide any feature that allows the domain classification. As a result of the adversarial training, the features in the last layer of the *feature extractor* will both allow the classification between signal and background and be model-independent. The gradients of the

reversal layer are scaled by the parameter λ allowing to regularize their influence and hence tune the importance of the label classification versus the model-independence.

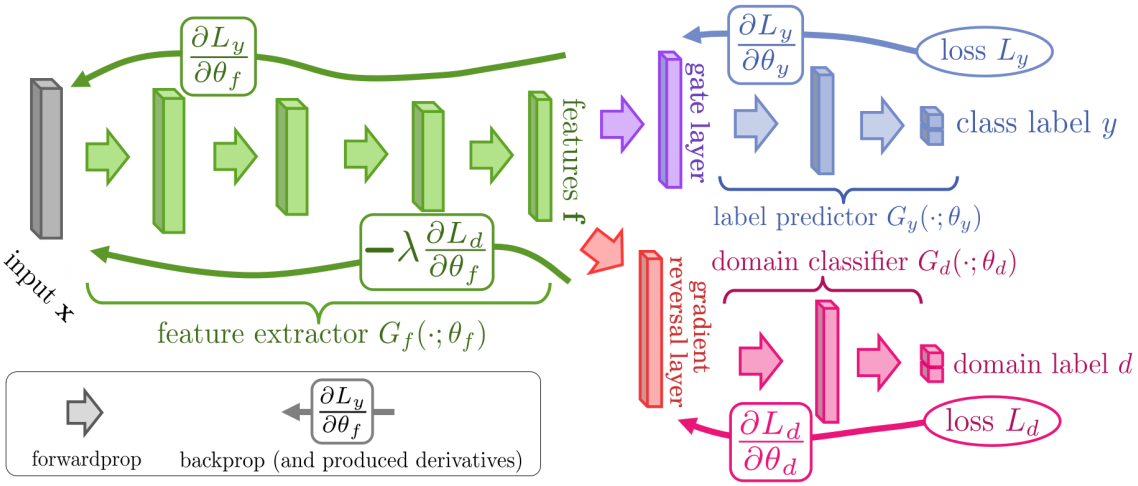


Figure 1: Domain-adversarial network as an alternative to reduce classification bias, adapted from [2].

In order to have balanced classes for each classification the event weights of the source domain are scaled as required. For the label predictor the weights are such that the effective number of signal and background events are the same. For the domain classifier, the weights are scaled to match the signal to background ratio existing in the target domain.

3 Data sets

The feature selection for the input of the network was inspired by the analysis presented in [1] to separate $t\bar{t}H$ from the $t\bar{t}$ + $b\bar{b}$ background. In total 41 geometrical and kinematic quantities are used as input to the network, such as the angular distance between different jets and/or leptons, the mass of various jet and lepton systems and the event topology. The complete list of features, their correlations and the relative importance are given in [6].

We use MC samples provided by the HepSim Group [7]. The ttH signal sample containing $13 \cdot 10^6$ events was generated with MadGraph [8] matched to the Herwig6 parton shower [9]. Two background samples were generated. One, used for the source domain, with $2 \cdot 10^6$ events of top quark pair production with additional light quarks using MadGraph matched to the Herwig6 merged with 10^7 top quark pair events with additional bottom-quarks using MadGraph matched to Pythia6 [10]. The other background sample, which is used for the target domain, contains $3 \cdot 10^7$ events of top quark pair production in association with bottom quark pairs, generated with the PowhegBox+OpenLoops [11] and is matched to Pythia8 for the full event generation including the prediction of additional light quarks.

The ATLAS detector response was simulated using Delphes simulation [12]. For this study, reconstructed leptons, jets and bottom[1] quark iniated jets (called *b*-jets in the following) are used.

[1] *bottom* stands for bottom and anti-bottom quarks

The identification efficiency of b-jets was taken from [13], assuming the reconstructed b-jets to have a 70% tagging probability with a corresponding light jet/c-jet rejection probability parameterization.

Events selected for the neural network training were required to fulfil the following criteria:

- one electron or muon with transverse momentum $p_T \geq 20\,\text{GeV}$
- at least 5 jets with $p_T \geq 25\,\text{GeV}$
- at least 3 b-jets.

With this selection applied the source and target data sets where constructed with $546 \cdot 10^3$ signal each and same amount of background events, using statistically independent events from the same simulation as signal but different background simulations for source and target. One half from each data set was left for testing purposes, the remaining were used for training. For the target domain only 14368 signal events were randomly selected for training, to match the 5:95 ratio of signal to background estimated in real data.

4 Network set-up and training

The network was implemented using the Keras package v2.2.4 [14] with TensorFlow v1.12 [15] as back-end. The training set-up is described in Sec. 4.1. A hyper-parameter scan was done to optimize the performance of the network, as described in Sec. 4.2. Some special considerations for the loss function and its optimization are described in Sec. 4.3 and Sec. 4.4, respectively.

4.1 Training set-up

The initial weights of the network were set by the Xavier initializer, as suggested in [16]. The number of training epochs was dynamically selected with the following condition applied: the training were stopped if the running average over 50 epochs in the total loss does not decrease more than 0.05% with respect to the previous 50 epochs. This number was restricted to the interval [200, 1000]. The lower limit was set to skip some random fluctuations at the beginning. The upper limit is just a big value that was never reached with the specified condition. After the training was stopped the weights of the network in the epoch with the lowest label predictor loss were selected. A batch size of 16384 was used. Each batch was composed by source and target events in a 1:1 proportion. The events were randomly shuffled at each epoch, resulting in a different batch selection each time. The *domain classifier* and *label predictor* outputs were set to have two neurons each, using softmax activation function and cross-entropy loss in both (Sec. 4.3 describes an alternative). The RMSProp keras optimizer was used, with the parameters: learning_rate = 0.001 and rho = 0.9.

4.2 Hyper-parameter optimization

The hyper-parameters of the network were chosen with the help of the Hyperopt library [17], using the Tree of Parzen Estimators (TPE) algorithm implemented on it. The number of layers in each part of the network was let vary from 1 to 8. Each layer could have a number of neurons between 5 and 100, but having a linear behavior in each part of the network (either decreasing or increasing). For the activation function of the hidden layers ReLU, tanh and ELU were tested. Each of this

hyper-parameters were sampled from a uniform distribution. Additionally, the λ parameter was sampled from a log-uniform distribution in the range [1, 1000], with this giving more priority to low values as these were found to give better results.

The additive inverse of the label *label predictor* area under the receiver operating characteristic curve for the target domain was used as the loss to minimize. Three independent optimizations where performed in parallel in order to have a better view of the hyper-parameter space. This also helps to detect if the global minimum of the loss is found. Approximately 1000 iterations where performed in each case.

By analyzing the sets of parameters with good performance and the decisions made by the sampling algorithm, we were able to draw the following conclusions:

- The optimal number of layers in the *label predictor* is one: only the output layer. Two is also good in cases of a very complex *feature extractor*.
- Higher complexity in the *feature extractor* provides performance improvement but also makes the network more prone to over-training.
- The number of neurons in the last layer of the feature extractor should be at most the same that in the input. We think this number is also related to the correlations in the input features: a smaller number for high correlations could provide a better optimized feature extraction.
- An increase in the domain classifier complexity does not cause significant improvements, but it needs at least a similar complexity than the feature extractor in order to provide good corrections.
- The performance with ELU and tanh as activation function for the hidden layers was very similar. ReLU was significantly worse.

Finally the *feature extractor* was chosen to have four layers with 20, 16, 13 and 10 neurons respectively, the *label predictor* with only the output layer (2 neurons) and the *domain classifier* with four layers of 20, 35, 50 and 2 neurons respectively. The ELU activation function was used in all the hidden layers.

Note that due to the non-deterministic nature of the training process, results during the optimization were sometimes not representative of the behavior for each set of hyper-parameters tested. Set-ups with higher performance were found, but its results were not reproduced in further tests. Therefore, we chose a configuration with stable results instead of the best one reported by the optimization process. It also had the advantage of being not complex enough to be affected by over-training.

4.3 Loss and activation functions for the outputs

The total loss of the network (L) is given by the sum of the individual losses of the *label predictor* (L_y) and *domain classifier* (L_d):

$$L = L_y + L_d \tag{4.1}$$

The gradient reversal layer affect the backpropagation in such a way that the gradients of the total loss with respect to the *feature extractor* weights (θ_f) are computed as:

$$\frac{\partial L}{\partial \theta_f} = \frac{\partial L_y}{\partial \theta_f} - \lambda \frac{\partial L_d}{\partial \theta_f} \tag{4.2}$$

Two alternatives were used for computing the loss: set-up A with a softmax activation and cross-entropy loss in both outputs, and set-up B with softmax and cross-entropy loss in the *label predictor* but linear activation plus a linear loss in the *domain classifier*.

The cross-entropy loss for a single event E_i is given by:

$$L_i = \begin{cases} -\ln(y_i) & \text{if } E_i \in \text{class 1} \\ -\ln(1 - y_i) & \text{if } E_i \in \text{class 0} \end{cases} \tag{4.3}$$

where y_i represents the network output for that event. Note that even though we have a two-neuron output we refer to y_i as a single value since the second neuron behaves as 1 minus the first. Class 0 corresponds to background and class 1 to signal for the *label predictor*. A perfect classification yields a loss of 0, value toward which the loss is optimized.

Set-up A also uses this loss for the *domain classifier*, with y_i in equation 4.3 corresponding to the *domain classifier* output and classes 0 and 1 corresponding to target and source domains respectively. In this case, perfect separation also results in a loss of 0 but a separation between the domains is not intended. Instead, the network response should be the same for both classes of events which is provided as an additional restriction via the gradient reversal layer. The *domain classifier* loss is minimized but, under this restriction, the lowest achievable loss is when the response for both classes, i.e. source and target, is $y_i = 0.5$, resulting in a loss of $-\ln 0.5 \approx 0.693$. This behavior is visible in Fig. 2b, where the predicted loss of 0.693 is reached in the first epochs and kept most of the training. It should be noted that this poses an extra requirement on the *feature extractor*, which besides providing domain independent features, is also optimised to provide features for which the *domain classifier* output are exactly 0.5 for all events.

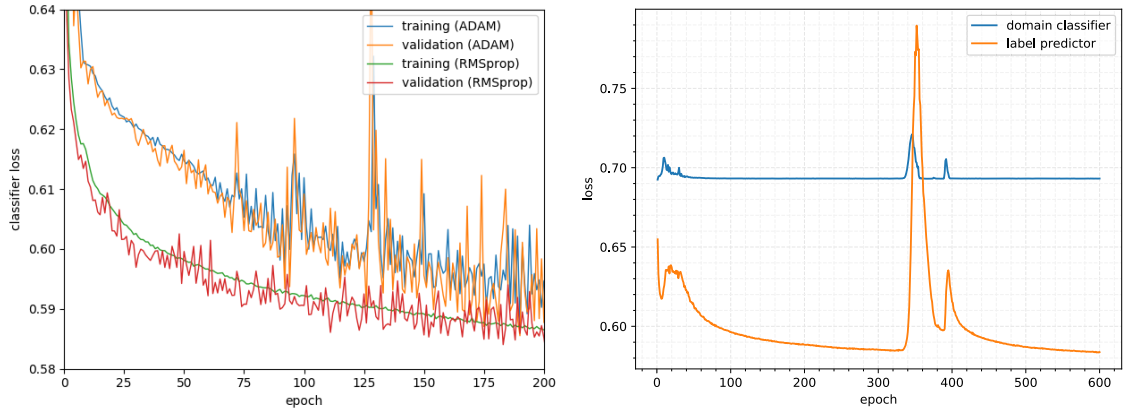
We found that deviations in the output of the *domain classifier* from the optimal value of $y_i = 0.5$ had severe influences on the classification in general. Analyzing at a lower level we found that these changes were driven by huge gradients back-propagated from the domain classifier loss, further amplified by λ as $\lambda > 1$ was used. To avoid the change in the gradients under y_i deviations we tested a set-up where the derivatives of the *domain classifier* loss were independent of the y_i values. To achieve this behaviour, we removed the activation function from the *domain classifier* output and changed the loss to a linear function, computed for a single event (E_i) as:

$$L_i = \begin{cases} -y_i & \text{if } E_i \in \text{source} \\ y_i & \text{if } E_i \in \text{target} \end{cases} \tag{4.4}$$

This new set-up has also the advantage that y_i is not limited to 0.5 in the optimized case, since now, if the condition of no domain separation is met, this loss has a value of 0 for any value of of the *domain classifier* output so the *feature extractor* has more freedom during the optimization.

4.4 Training of the neural network

The ADAM optimizer [18], being commonly used nowadays, was used as starting point. However, we noticed severe oscillations of the label predictor loss, as shown in Fig. 2a. These oscillations are caused by fluctuations in the domain classifier part on which the label predictor reacts in the common minimization algorithms of the global loss. By inspecting at the gradients for each weight during training, we found that the fluctuations are caused by the momentum term in the ADAM optimizer which is defined as decaying average of the past gradients. While the *domain classifier* loss fluctuate around their optimal value with high frequency, the approximated gradients by ADAM are slower to respond due to the momentum. The RMSprop optimizer does not have a momentum term and therefore provides a smoother loss during the training, as shown in the green and red lines in Fig. 2. It was hence the chosen optimizer.



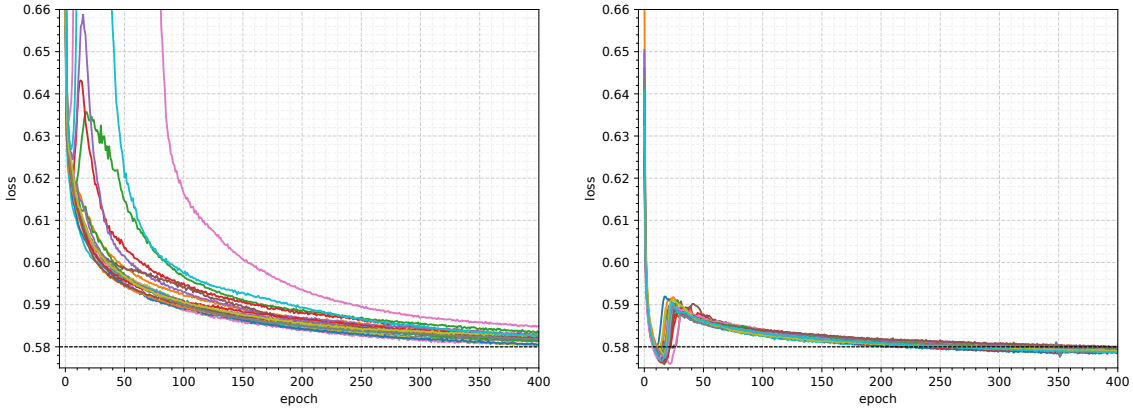
(a) (b)

Figure 2: Examples of fluctuations observed during training of the network using set-up A. (a) Training and validation loss for the label predictor using the ADAM and RMSprop optimizers. (b) Big spikes of the label predictor and domain classifier losses (case with $\lambda = 20$).

Beside those small fluctuation described above, infrequent large spikes where found. One of them is shown in Fig. 2b, where L_y minimizes smoothly for over 300 epochs but suddenly L_y raise to huge values together with L_d. Running 3000 independent trainings we found that these spikes appear in around a 0.7% of the cases for set-up A and 1.6% for set-up B.

Performing some analyses we found that changing the weights of the network to the ones used 10 epochs before makes the spikes not to appear again. This indicates that the cause of the spikes involves some random fluctuation related to the adversarial training with the gradient reversal layer. We also found that the frequency of these spikes increases by increasing the value of λ.

Furthermore, comparing set-ups A and B, we found in A stronger dependence of the learning curves on the randomization (initial weights, data shuffling, etc.), which is demonstrated in Fig. 3. The learning curves for set-up B agree better indicating a more stable training. They also converge faster. The stopping criterion is reached in set-up A after around 600 epochs but after about 400 epochs in set-up B.



(a) (b)

Figure 3: *Label predictor* loss in 20 different trainings for (a) set-up A with $\lambda = 20$ and (b) set-up B with $\lambda = 10^{-5}$. For better visibility, the y-axis is limited to 0.66 cutting off high fluctuations. Note that set-up B reaches better *label predictor* loss.

4.5 Regularization of the domain classification

The parameter λ controls the influence of the label predictor response on the final result, i.e. how much the responses of the source and target data in the classification should agree. High λ values provide a good agreement but restrict the ability of the feature extractor to provide useful features for the classification, low values give more freedom but might not be enough for obtaining a good agreement between the domains. To give an example, Figure 4 shows the discriminant output for the set-up A for values of λ between 0 and 20. A large difference between source and target domain can be observed for $\lambda = 0$, while with increasing values of λ the influence of the domain classifier on the label prediction increases and a similar response to both background samples can be reached at the highest value of λ. The optimal lambda value is specific to the problem and the performance measure applied as will be discussed in the following.

5 Results

The performance of the network depends on the relative importance of the adversarial branch steered by the parameter λ. As for any hyper-parameter, the values of λ are specific to the network architecture and data sets used and need to be determined for each particular use case. We consider three measures of performance, demonstrating the bias without the adversarial treatment and their improvement when the adversarial branch is included.

First we report AUC which is a common performance measure for binary classifiers. Since a good value for λ was still not selected we made a scan over a range of possible values (Fig. 5). We extend it with the Kolmogorov-Smirnov distance as a measure of agreement between the response of the two domains. This distance is given by the maximum absolute difference between the cumulative distributions of the normalized *label predictor* response for the two domains. The best choice of λ is the value for which the maximum source domain AUC is achieved among those with

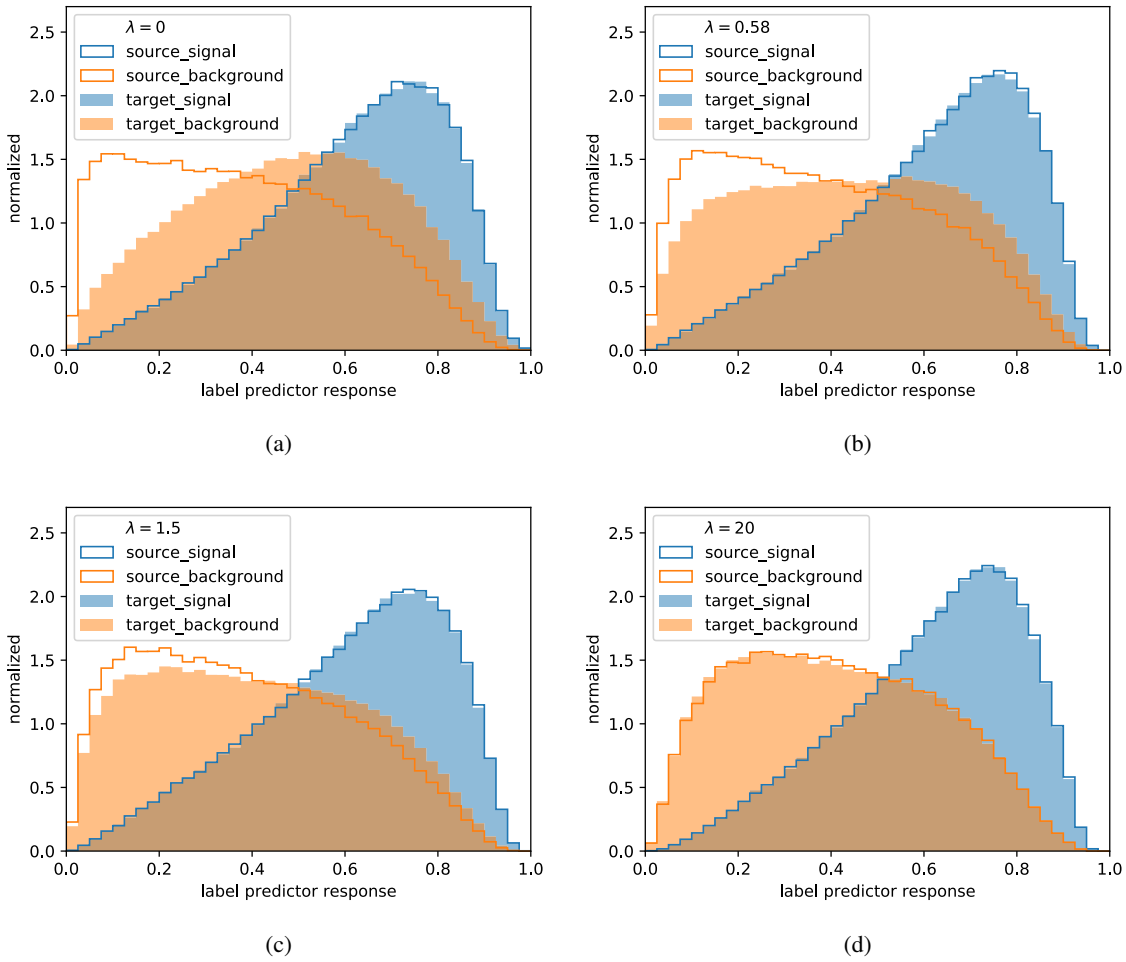
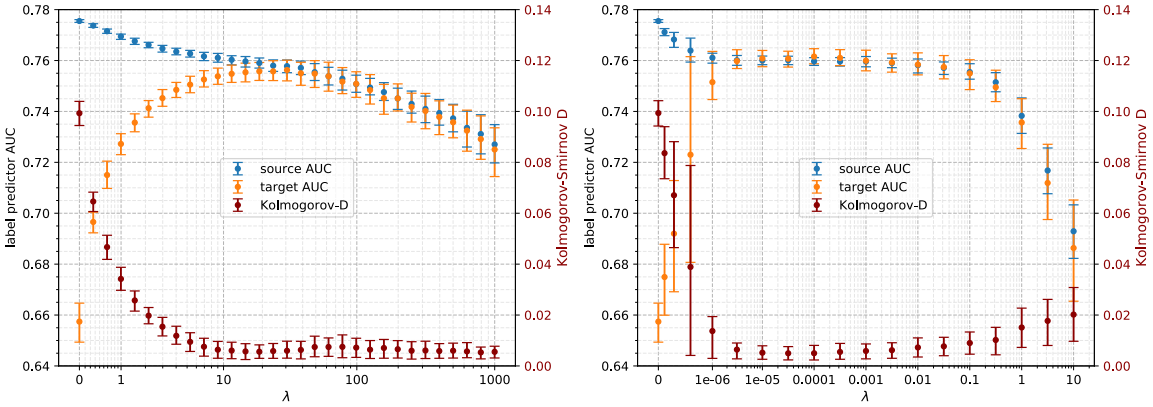


Figure 4: Label predictor response to signal (blue) and background (orange) for different values of λ. The label predictor is trained on the source domain and applied to a statistically independent part of the source domain (lines) and the target domain (area). Each of the distribution is normalised to 1. (a) $\lambda = 0$ (b) $\lambda = 0.58$ (c) $\lambda = 1.5$ (d) $\lambda = 20$. Discussion see text.

the lowest Kolmogorv-Smirnov distance. This criterion for the optimal λ has the advantage that it can be applied without using the target labels. To validate it of λ selection we compute the AUC obtained for the target domain as in our study target labels were provided by the simulation.

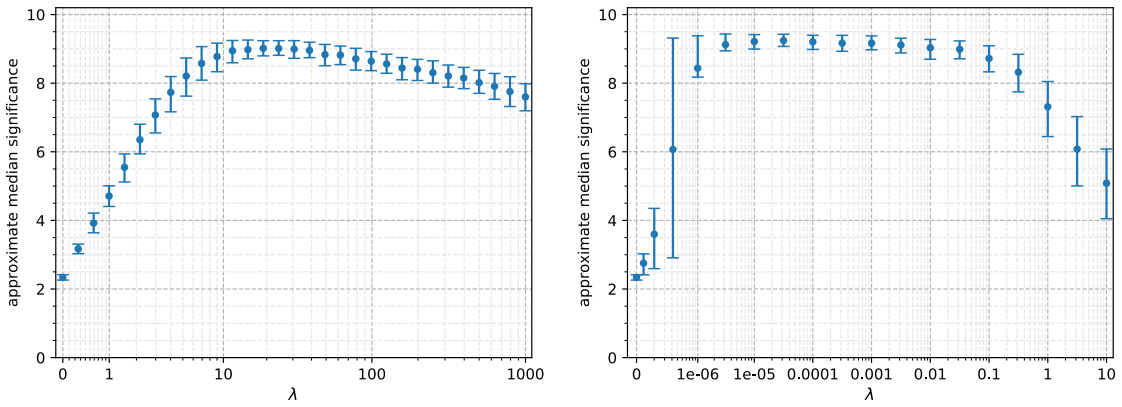
With $\lambda = 0$, corresponding to no adversarial network, an AUC on the target domain of 0.657 is achieved. This value is improved to 0.756 using $\lambda = 20$ for set-up A, and 0.760 using $\lambda = 10^{-5}$ for set-up B. This improvements have the cost of reducing the AUC obtained for the source domain from 0.776 in the no adversarial case, to 0.757 and 0.760 for set-ups A and B respectively with the selected λ values. Increasing λ above those values only decreases the performance, but in the case of set-up B a plateau exist such that taking λ values up to 100 times the selected one keeps the same performance.

To further approximate the significance as reported in Higgs discovery searches as performance



(a) (b)

Figure 5: Performance measured as the area under the ROC curve (AUC) for several values of λ. The difference between the response for source and target is measured as the Kolmogorov-Smirnov distance. (a) set-up A (b) set-up B. Each point represent the average over $\sim$ 200 independent training processes (with different random numbers). The error bars represent the 15.8 and 84.2 percentiles, corresponding to $\pm 1\sigma$ in a normal distributed variable.



(a) (b)

Figure 6: Approximate median significance (in units of standard deviations) as a function of λ, computed for 50000 events consisting of 5% signal and 95% background. (a) set-up A (b) set-up B. Each point represent the average over $\sim$ 200 independent trainings and the error bars represent the 15.8 and 84.2 percentiles.

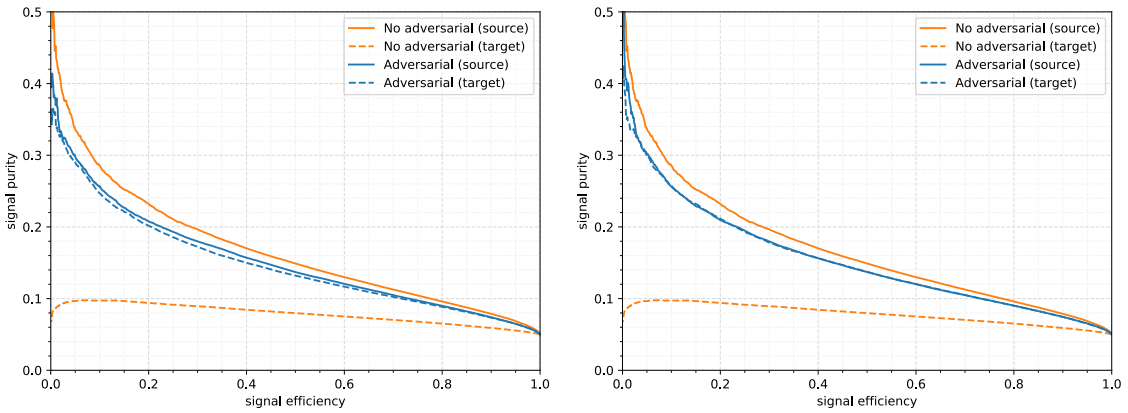
measure, we use the approximate median significance (AMS) as proposed in [19]. This definition corresponds to a hypothesis test of signal discovery versus background only hypothesis by taking

systematic uncertainties into account. It is calculated as:

$$\text{AMS} = \sqrt{\sum_i \left\{ 2\left[(s_i + b_i) \ln \frac{s_i + b_i}{b_{0i}} - s_i - b_i + b_{0i} \right] + \frac{(b_i - b_{0i})^2}{\sigma_{bi}^2} \right\}} \tag{5.1a}$$

$$b_{0i} = \frac{1}{2}\left(b_i - \sigma_{bi}^2 + \sqrt{(b_i - \sigma_{bi}^2)^2 + 4(s_i + b_i)\sigma_{bi}^2} \right) \tag{5.1b}$$

where the sum is over the bins in the histogram of the response, s_i and b_i represents the signal and background counts in the bin i for the source domain and $\sigma_{b_i}^2 = \frac{1}{2}(b_i - b_i^{alt.})^2 + (0.1\ b_i)^2$ is an estimator of the variance on the background counts. The variance is computed from the difference between b_i and the background count for the target domain in the same bin ($b_i^{alt.}$) plus a flat 10% uncertainty on the background, approximating the values of the reference analysis. The AMS is a valid simplification of the significance in the context of this paper as long as we consider only the qualitative behavior, not the absolute values. The AMS as a function of λ is shown in Fig. 6. A low significance is observed in the case when the response for both domains disagree. The significance increases with λ until reaching a maximum at similar positions of the maximal AUC where source and target values agree (Fig. 5). For higher values of λ the significance decreases, reflecting the loss of classification power.



(a) (b)

Figure 7: Signal purity p_{sig} versus signal efficiency ϵ_{sig} of the label predictor, calculated for the source (solid, upper line) and target (dashed, lower line) sample, with (blue) and without (orange) the adversarial architecture. (a) set-up A (b) set-up B.

Using the optimized λ setting we measure the performance in terms of signal purity, which is related to the sensitivity of the measurement. It is defined as the ratio between number of signal events (s) and the total of events ($s + b$) that fall above a specific cut in the *label predictor* response: $p_{sig} = \frac{s}{s+b}$. Each possible cut corresponds to a signal efficiency ($\epsilon_{sig} = \frac{s}{N_s}$), which is defined as the fraction of signal selected (s) from the total number of signal events (N_s). Figure 7 shows the whole profile of the signal purity as a function of the signal efficiency. The expected signal to background composition of 5:95 is taken into account. Classification without the adversarial part

reaches around a 9% higher purity on the source domain, but very low values for the target domain. The adversarial network yields very close values for both source and target domains.

To give a numerical example taking the signal purity as an approximation of the analysis sensitivity, we take the results for the source domain as the central value and the difference between the two domains as a 1σ uncertainty. For $\epsilon_{sig} = 0.5$ we get:

- no adversarial network: $p_{sig} = 0.148 \pm 0.069$
- adversarial set-up A: $p_{sig} = 0.137 \pm 0.005$
- adversarial set-up B: $p_{sig} = 0.1369 \pm 0.0004$

The relative impact due to the choice of the background model on the signal purity, ignoring other sources of uncertainty, can be improved from 47% to less than a 4% by employing the adversarial network.

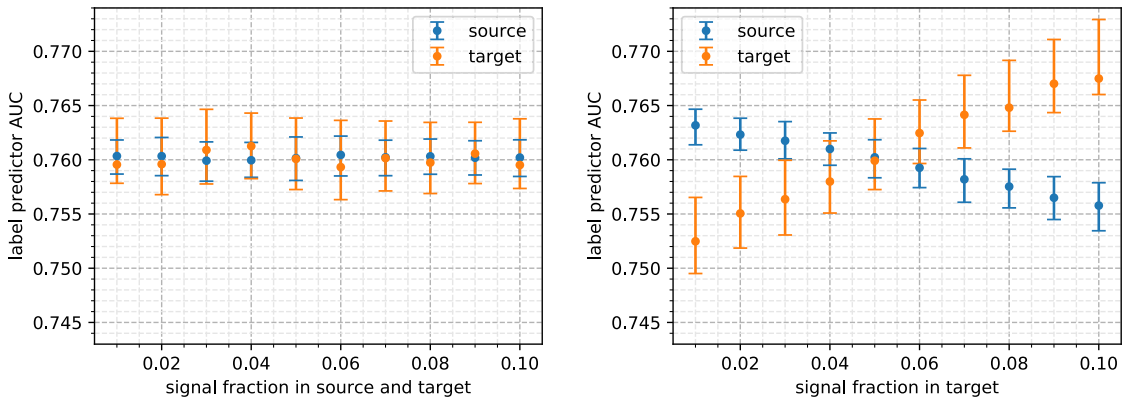
5.1 Sensitivity to signal over background ratio

For the results presented so far, the signal to background ratio was set to the predicted value of 5:95 in the target domain, while scaling the source to the same ratio in the *domain classifier*. Despite using the same signal model and only differing background models in both domains, the signal was added to the discrimination to provide a configuration with a pseudo-data that can easily be extended to future scenarios in which both signal and background differ, or in which no class labels are available in the target domain, as the case of real collision data. The dependence of the *label predictor* output on the chosen signal to background ratio was tested. It was found that a change in its value had no impact, as long as it is the same in both source and target (Fig. 8a). If there is a discrepancy in the signal fraction for both domains a small bias is introduced. This behavior is shown in Fig. 8b, where a fixed value of 5% was used in the source but different values existed in the target domain. It is therefore important to properly estimate the signal to background ratio in the target domain when using unlabeled data. The behavior shown here applies only to the cases when the signal in both domains is taken from the same simulation or when the signal simulation is good enough, leading to differences only in the background.

6 Conclusion

We successfully built a feed-forward fully connected adversarial neural network for a high energy physics use case. We demonstrate that adding a *domain classifier* sub-network with a gradient reversal layer helps removing training bias while retaining most of the nominal classification power. We analyzed the dependence on the hyper-parameters of the network. We studied the training stability issues that appear due to the addition of a gradient reversal layer. We demonstrated that by using linear activation and loss functions, stability and convergence can be significantly improved and better performance of the network can be achieved.

For the example use case of the ttH(bb) analysis, we demonstrate that the adversarial domain adaptation can produce an almost completely background generator independent label predictor while preserving most of the classification power. We report the improvements with different



(a) (b)

Figure 8: AUC ROC for the label predictor (a) as a function of the signal fraction in source and target and (b) as a function of the signal fraction in target with a fixed 5% signal fraction in the source. These plots were produced for set-up B with $\lambda = 10^{-5}$. Set-up A exhibits a similar behavior.

figures of merit. With the Monte Carlo samples used in this study, the uncertainty due to the choice of background model on the expected signal purity (a proxy measure for the sensitivity of the analysis) can be improved from a 47% to a 0.3% for the chosen signal efficiency of 50%. Significant improvements are also reached in the approximated median significance.

Although not demonstrated, there is no limitation to extend this approach to discriminate against multiple alternative domains, i.e. sources of uncertainty, during training. This approach is also expected to work with a target sample of real collision data as no label information is used in training of the *domain classifier*. However, even though the labels are not used we show that in such case, a mismatch of the signal to background ratio in source and target leads to a small bias of the classification and will need to be addressed in an analysis.

Acknowledgments

We acknowledge that Ilyas Fatkhullin contributed at an early stage of the analysis [20].

References

[1] M. Aaboud *et al.* [ATLAS Collaboration], *Search for the standard model Higgs boson produced in association with top quarks and decaying into a $b\bar{b}$ pair in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*, Phys. Rev. D **97** (2018) no.7, 072016 [arXiv:1712.08895[hep-ex]].

[2] Y. Ganin *et al.*, *Domain-Adversarial Training of Neural Networks*, Journal of Machine Learning Research 2016, vol. 17, p. 1-35 [arXiv:1505.07818 [stat.ML]] (2015).

[3] Ben-David, Shai and John Blitzer and Crammer, Koby and Fernando Pereira, *Analysis of Representations for Domain Adaptation*, in Advances in Neural Information Processing Systems 19, p137–144 (2007).

[4] C. Shimmin *et al.*, *Decorrelated Jet Substructure Tagging using Adversarial Neural Networks*, Phys. Rev. D 96, 074034 (2017), [arxiv:1703.03507 [hep-ex]].

[5] G. Louppe, M. Kagan and K. Cranmer, *Learning to Pivot with Adversarial Networks*, Advances in Neural Information Processing Systems 30, p981 (2017), [arxiv:1611.01046 [stat.ML]] (2016).

[6] P. Glaysher, J. M. Katzy and S. An, *Iterative subtraction method for Feature Ranking*, [arXiv:1906.05718 [physics.data-an]](2019).

[7] S. V. Chekanov [HepSim Group], Advances in High Energy Physics, vol. 2015, Article ID 136093, 7, [arxiv:1403.1886 [hep-ph]].

[8] J. Alwall, M. Herquet, F. Maltoni, O. Mattelaer and T. Stelzer, JHEP **1106** (2011) 128, doi:10.1007/JHEP06(2011)128, [arXiv:1106.0522 [hep-ph]].

[9] G. Corcella, I. Knowles, G. Marchesini, S. Moretti, K. Odagiri, P. Richardson, M. Seymour, B. Webber JHEP **0101:**010 (2001), [arXiv:hep-ph/0011363].

[10] T. Sjostrand, S. Mrenna and P. Z. Skands, JHEP **0605** (2006) 026 [hep-ph/0603175].

[11] T. Jezo, J. Lindert, N. Moretti, S. Pozzorini, *New NLOPS predictions for $t\bar{t}$ + b-jet production at the LHC*, Eur. Phys, J. C78-6-502 (2018), [arXiv:1802.00426 [hep-ph]].

[12] J. de Favereau *et al.* [DELPHES 3 Collaboration], *DELPHES 3, A modular framework for fast simulation of a generic collider experiment*, JHEP **1402** (2014) 057, [arXiv:1307.6346 [hep-ex]].

[13] M. Aaboud *et al.* [ATLAS Collaboration], *Measurements of b-jet tagging efficiency with the ATLAS detector using $t\bar{t}$ events at $\sqrt{s}$ = 13 TeV*, JHEP **1808** (2018) 089 [arXiv:1805.01845 [hep-ex]].

[14] F. Chollet *et al.*, *Keras*, https://github.com/fchollet/keras (2015).

[15] M. Abadi et.al., *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, [https://www.tensorflow.org/]

[16] X. Glorot, Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, Proc. of 13th Intern.Conf.on Artificial Intelligence and Stat.(AISTATS) Vol.9 of JMLR (2010).

[17] J. Bergstra, D. Yamis, D.D. Cox, *Making a Science of Model Search*, In Proc. of 30th Int. Conf. on Machine Learning (ICML 2013) Volume 28 p.I-115 (2013).

[18] Diederik P. Kingma, Jimmy Ba, *Adam a method for stochastic optimisation*

[19] C.Adam-Bourdarios, G.Cowan, C. Germain, I.Guuon, B. Kegl, D.Rousseau, JMLR: Workshop and Conference Proceedings: 42:19-55 (2015)

[20] I. Fatkhullin, DESY Summer Student Program's report (2019) [https://www.desy.de/f/students/2019/reports/Ilyas.Fatkhullin.pdf]