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

IOP Publishing Mach. Learn.: Sci. Technol. 3 (2022) 015014 https://doi.org/10.1088/2632-2153/ac3dde

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
15 June 2020

REVISED
9 August 2021

ACCEPTED FOR PUBLICATION
26 November 2021

PUBLISHED
27 December 2021

PAPER

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

J M Clavijo[1,4], P Glaysher[1,4], J Jitsev[2,3,5] and J M Katzy[1,4,5,**]

[1] Deutsches Elektronen-Synchrotron DESY Notkestraße 85, 22607 Hamburg, Germany
[2] Juelich Supercomputing Center (JSC), Institute for Advanced Simulation (IAS), Research Center Juelich (FZJ), Wilhelm-Johnen-Str., 52425 Juelich, Germany
[3] Helmholtz AI, Research Center Juelich (FZJ), 52425 Juelich, Germany
[4] Equal contribution.
[5] Equal advising.

** Author to whom any correspondence should be addressed.

E-mail: judith.katzy@desy.de

Keywords: adversarial training, adversarial neural network, domain adaptation, LHC, ttH

Abstract

We apply adversarial domain adaptation in unsupervised setting to reduce sample bias in a supervised high energy physics events classifier training. We make use of a neural network containing event and domain classifier with a gradient reversal layer to simultaneously enable signal versus background events classification on the one hand, while on the other hand minimizing the difference in response of the network to background samples originating from different Monte Carlo models via adversarial domain classification loss. We show the successful bias removal on the example of simulated events at the Large Hadron Collider with $t\bar{t}H$ signal versus $t\bar{t}b\bar{b}$ background classification and discuss implications and limitations of the method.

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 Monte Carlo (MC) simulation data sets, containing samples labeled as signal or background. The resulting classifier is applied to unlabeled data to separate signal and background, and to measure the statistical significance of the signal or its strength, assuming that the simulated and the real data sets are identically distributed.

However, significant differences between domains of real and simulated data sets always exist and the learner may pick up those domain-specific discriminating features that perform well on classification task in one domain while being not suitable for classification in the other, introducing a bias via the source samples used for training when attempting to classify samples from target domain. This problem is similar to that of visual recognition where, for instance, training may be performed on artificially generated images, the source domain, and applied to real photographs, the target domain. In order to avoid training a model that is suitable for classification on the source domain only, while failing when employed on target domain, algorithms of domain adaptation have been developed.

In this paper we apply the method of domain adaptation to a problem of classification on high energy physics data using a Domain Adversarial Neural Network [1] to classify events in the search for the $t\bar{t}H(H \to b\bar{b})$ process at the LHC, which is very rare and hard to separate from the abundant $t\bar{t}+$ jets background [2]. In the cited measurement work, a classifier is trained on labeled MC predictions to separate signal from background. The trained classifier is applied on MC where signal and background events are mixed according to the theoretical predicted fraction, and on data to obtain binned distributions of classifier output. The ratio of the resulting spectra is used in a profile likelihood fit to measure the signal ratio in data.

The effect on the final result caused by the bias for the specific MC background model of the source domain used for training is estimated using an alternative simulation of a target domain, based on a different physics model, which was not used for training. The difference between the classifier outputs of the different background MCs is taken as uncertainty on the classification in the fit. 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 extractor part and separate branches for label classification and domain adaptation, implemented via a gradient reversal layer as presented in [1]. This network structure differs from other adversarial approaches by including domain adaptation in the learning process via the shared feature extractor part used by both label and domain classifier as proposed on theoretical grounds in [3]. This way, the network model is pushed to extract discriminant features for the classification that are at the same time invariant to the different domains. The use case presented here differs from [1] as we provide a set of physical jet properties 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 training issues that appear for this kind of architecture.

Adversarial classifiers without domain adaptation were used in high energy physics before, e.g. to reduce theoretical uncertainties [4], to decorrelate a jet tagger from the jet mass [5] and to tune a classifier against a nuisance parameter [6]. An adversarial set-up involving domain adaptation without labels has been used for multi-class classification in a search for long lived particles [7]. In this paper, we systematically study the algorithm taking advantage of the two domains being labeled to control the results achieved without use of labels. We address the challenges for learning algorithms in domain adaptation, the dependence of the hyper-parameters specific to domain adaptation and potential bias of the classifier. Furthermore, since the use of domain adaptation without labels for multi-modal distributions can be a problem as pointed out in e.g. [8], we chose to use the domain adaptation algorithm for binary classification and proof its applicability for this use case in contrast to the multi-class application mentioned above.

In this paper we describe the network used in section 2, followed by the details of the data sets used in section 3. We systematically study the dependence on hyper parameters in section 4, including some issues observed during the training. In section 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 section 6.

2. The deep adversarial neural network

We follow the architecture presented in [1] with a feed-forward neural network composed of three parts as shown in figure 1: a *feature extractor* which splits into the *label predictor*, performing the signal-background classification, and the *domain classifier*. Domain adaptation is enabled via an adversarial interplay between domain classifier and feature extractor. 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 a representative 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 flowing from domain classifier during the backpropagation. The *domain classifier* is trained to determine which domain the events belong to. Due to the gradient reversal, the *feature extractor* is pushed to form such feature representation that do not allow to distinguish between two different domains, thus avoiding the sample bias. As a result of such adversarial training, the features in the last layer of the *feature extractor* will both allow the classification between signal and background and become domain invariant, rendering classification model domain-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 domain invariance.

In order to have balanced classes for each classification the event weights of the source domain are scaled as required according to class ratios. 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.

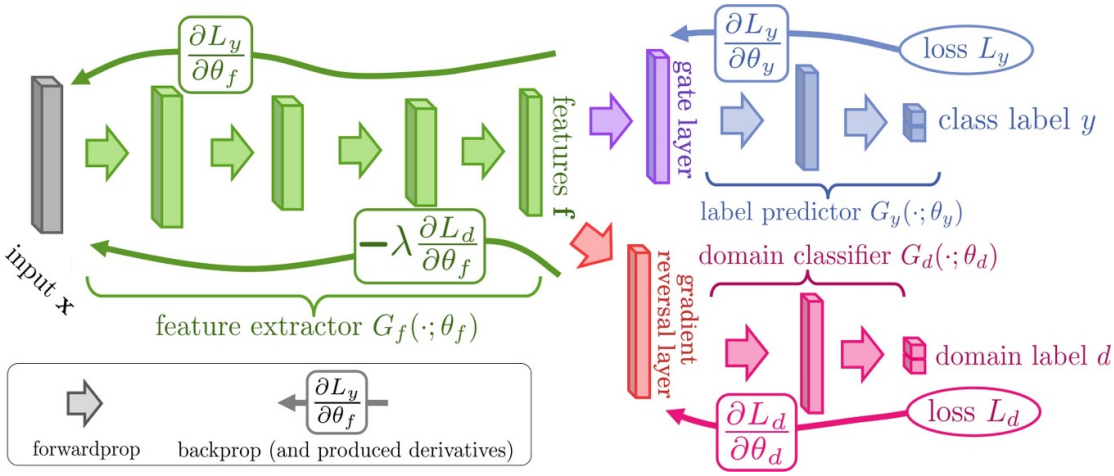


Figure 1. Domain-adversarial network as an alternative to reduce classification bias. Reproduced from [1]. CC BY 4.0.

3. Data sets

The feature selection for the input of the network was inspired by the analysis presented in [2] 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 [9].

We use MC samples provided by the HepSim Group [10]. The ttH signal sample containing 13×10^6 events was generated with MadGraph [11] matched to the Herwig6 parton shower [12]. Two background samples were generated, significantly differing in the theoretical predictions. One, used for the source domain, with 2×10^6 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 [13]. The other background sample, which is used for the target domain, contains 3×10^7 events of top quark pair production in association with bottom quark pairs, generated with the PowhegBox+OpenLoops [14] 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 [15]. For this study, reconstructed leptons, jets and bottom[6] quark initiated jets (called b-jets in the following) are used. Jets are reconstructed using the anti-kT algorithm [16] with a radius of $R=0.4$. The identification efficiency of b-jets was taken from [17], 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 $p_T \geqslant 20$ GeV
- at least five jets with $p_T \geqslant 25$ GeV
- at least three b-jets.

With this selection applied the source and target data sets where constructed with 546×10^3 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 14 368 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 v2.2.4 [18] with TensorFlow v1.12 [19] as back-end library. The training set-up is described in section 4.1. A hyper-parameter scan was done to optimize the

[6] *bottom* stands for bottom and anti-bottom quarks.

performance of the network, as described in section 4.2. Some special considerations for the loss function and its optimization are described in sections 4.3 and 4.4, respectively.

4.1. Training set-up

The initial weights of the network were set by the Xavier initializer, as suggested in [20]. 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 16 384 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 (section 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 [21], using the Tree of Parzen Estimators 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):

$$L = L_y + L_d. \tag{1}$$

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:

$$\frac{\partial L}{\partial \theta_f} = \frac{\partial L_y}{\partial \theta_f} - \lambda \frac{\partial L_d}{\partial \theta_f}. \tag{2}$$

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*, and linear loss in the *domain classifier*.

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

$$L_i = \begin{cases} -\ln(y_i) & \text{if } E_i \in \text{class 1} \\ -\ln(1-y_i) & \text{if } E_i \in \text{class 0} \end{cases} \tag{3}$$

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 (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 figure 2(b), 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 optimized 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 behavior, 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:

$$L_i = \begin{cases} -y_i & \text{if } E_i \in \text{source} \\ y_i & \text{if } E_i \in \text{target.} \end{cases} \tag{4}$$

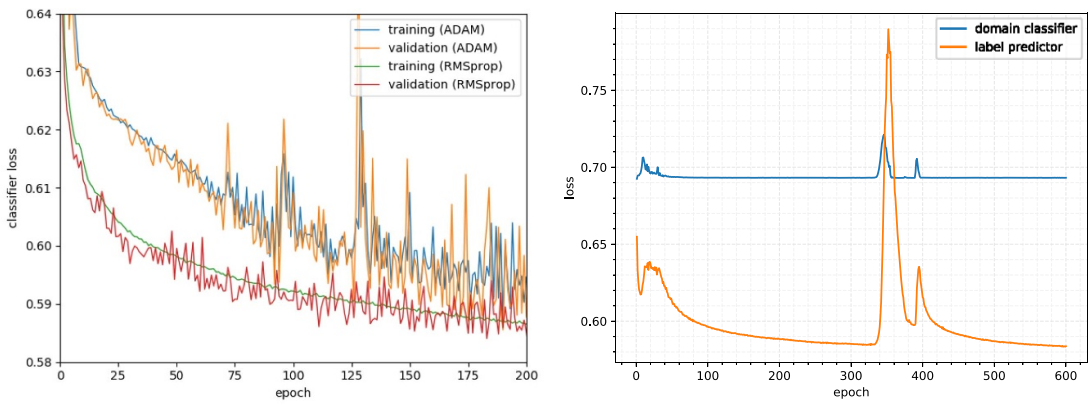
This new set-up has also the advantage that y_i is not limited to 0.5 in the optimized case, since now, if the condition of no domain separation is met, this loss has a value of 0 for any value of the *domain classifier* output so the *feature extractor* has more freedom during the optimization.

4.4. Training of the neural network

The ADAM optimizer [22], being commonly used nowadays, was used as starting point. ADAM is an extension of RMSProp with SGD Momentum i.e. adding momentum terms defined as decaying average of the past gradients. The momentum terms should help to faster escape from highly sub-optimal loss regions. However, when we used the default values of the momentum term ($\mu = 0.9$) we noticed severe oscillations of the label predictor loss, as shown in figure 2(a). These oscillations seem to be caused by fluctuations in the domain classifier loss part on which the label predictor has then to react in the common effort to minimize the global loss. We switched to RMSProp, which does not use a momentum term, resulting in a more stable loss course during the training. We therefore did not attempt to further use ADAM.

Beside those small fluctuation described above, infrequent large spikes where found. One of them is shown in figure 2(b), 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 analyses we found that changing the weights of the network to the ones used ten epochs before makes the spikes vanish. This indicates that the cause of the spikes involves initial random fluctuation



(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 $\lambda = 20$).

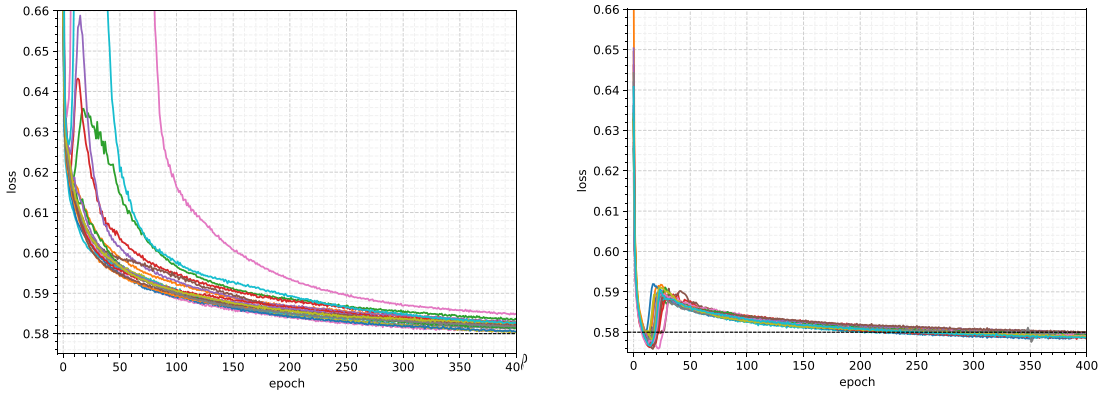


Figure 3. *Label predictor* loss in 20 different trainings starting from different random seeds for (a) set-up A with $\lambda = 20$ and (b) set-up B with $\lambda = 10^{-5}$. For better visibility, the y-axis is limited to 0.66 cutting off high fluctuations. Note that set-up B reaches better *label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

related to the adversarial training with the gradient reversal layer. Backing up this assumption, 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 figure 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.

4.5. Tuning impact of the adversarial domain classification

The parameter λ controls the influence of the label predictor and domain classifier responses on the total loss. It determines how much the responses to source and target data input produced by feature extractor should agree. High λ values forces a strong agreement but may impair the ability of the feature extractor to provide useful features for the classification, low values give more freedom for the feature representation density distribution but might not be enough for obtaining a good agreement between the domains and thus introduce a bias for source domain samples. 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 feature extractor response density can be observed for $\lambda = 0$, while with increasing values of λ the influence of the domain classifier on the density alignment and consequently also on label prediction increases and finally a very strong agreement between feature extractor responses to both background samples is reached at the highest value of λ, while label predictor performance deteriorates. The optimal lambda value is specific to the problem and the performance measure applied as will be discussed in the following.

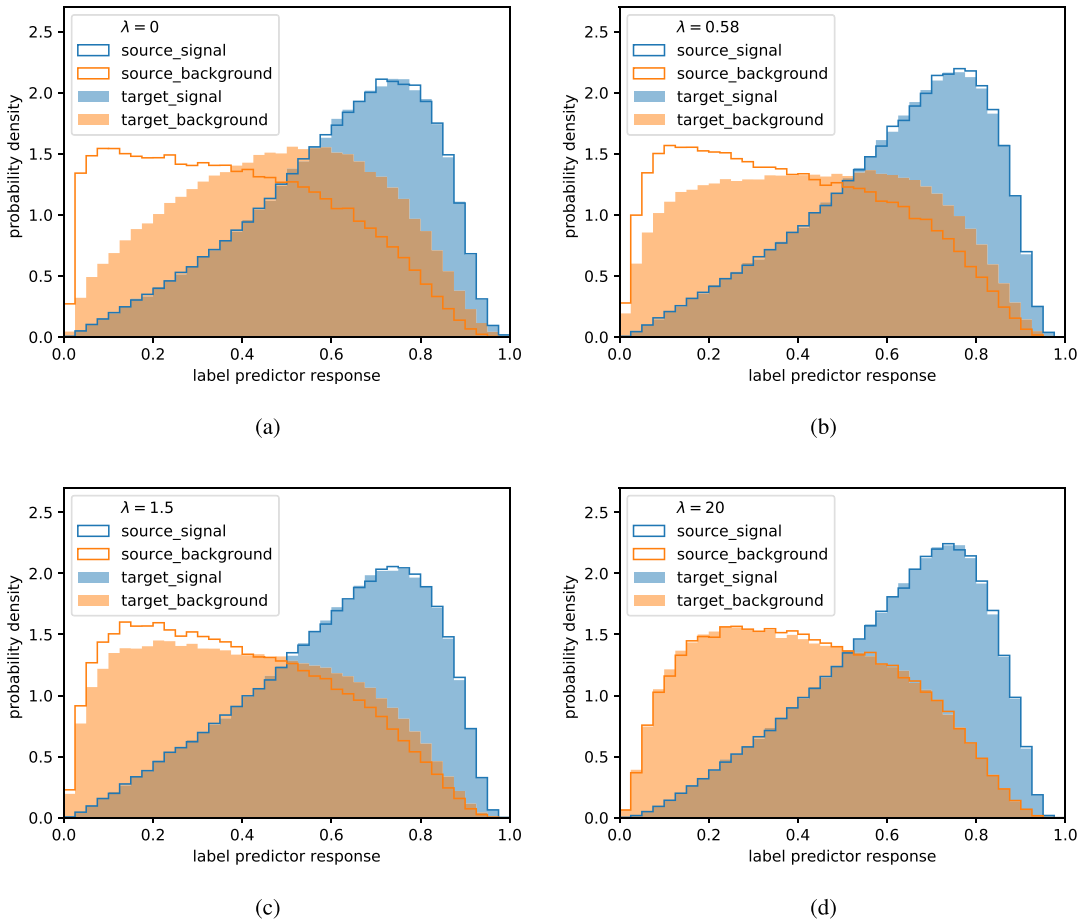


Figure 4. Label predictor response to signal (blue) and background (orange) for different values of λ. The label predictor is trained on the source domain and applied to a statistically independent part of the source domain (lines) and the target domain (area). Each of the distribution is normalized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

5. Results

The performance of the network depends on the relative importance of the adversarial branch containing domain classifier 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 (figure 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 the lowest Kolmogorov–Smirnov distance. This criterion for the optimal λ has the advantage that it can be computed without using the target labels, i.e. using labeled source and unlabeled target data. To demonstrate that the criterion for λ selection leads to desired behavior on target data, we compute the AUC for the target domain, as in our study target labels were provided by the simulation. As depicted in figure 5, the closest match between source and target domain and highest AUC performance is achieved when using lambda values obtained from the criterion procedure.

With $\lambda = 0$, corresponding to absence of 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.

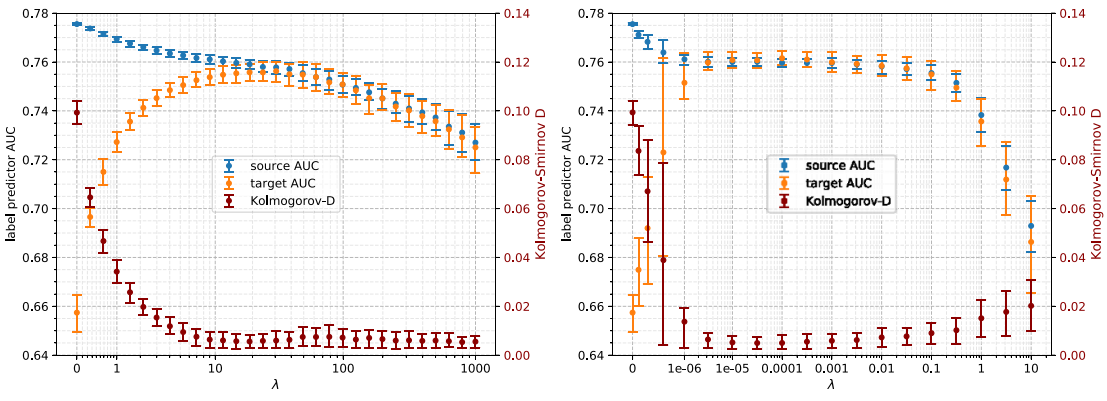


Figure 5. Performance measured as the area under the ROC curve (AUC) for several values of λ. The difference between the response for source and target is measured as the Kolmogorov–Smirnov distance. (a) set-up A, (b) set-up B. Each point represent the average over ~200 independent training processes (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.

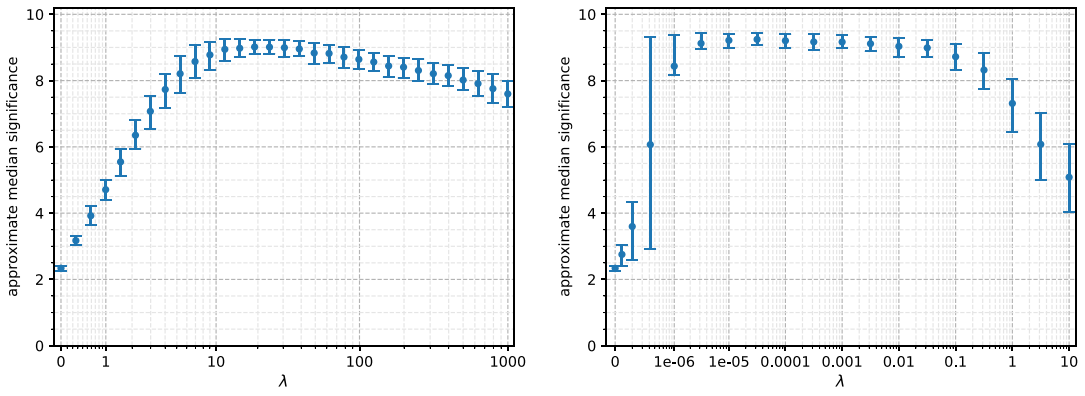


Figure 6. Approximate median significance (in units of standard deviations) as a function of λ, computed for 50 000 events consisting of 5% signal and 95% background. (a) set-up A, (b) set-up B. Each point represent the average over ~200 independent trainings and the error bars represent the 15.8 and 84.2 percentiles.

To further approximate the significance as reported in Higgs discovery searches as performance measure, we use the approximate median significance (AMS) as proposed in [23]. This definition corresponds to a test of the signal discovery versus background only hypothesis by taking systematic uncertainties into account. It is calculated as:

$$\mathrm{AMS} = \sqrt{\sum_i \left\{ 2\left[(s_i + b_i)\ln\frac{s_i + b_i}{b_{0i}} - s_i - b_i + b_{0i} \right] + \frac{(b_i - b_{0i})^2}{\sigma_{bi}^2} \right\}} \tag{5a}$$

$$b_{0i} = \frac{1}{2}\left(b_i - \sigma_{bi}^2 + \sqrt{(b_i - \sigma_{bi}^2)^2 + 4(s_i + b_i)\sigma_{bi}^2} \right) \tag{5b}$$

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 figure 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

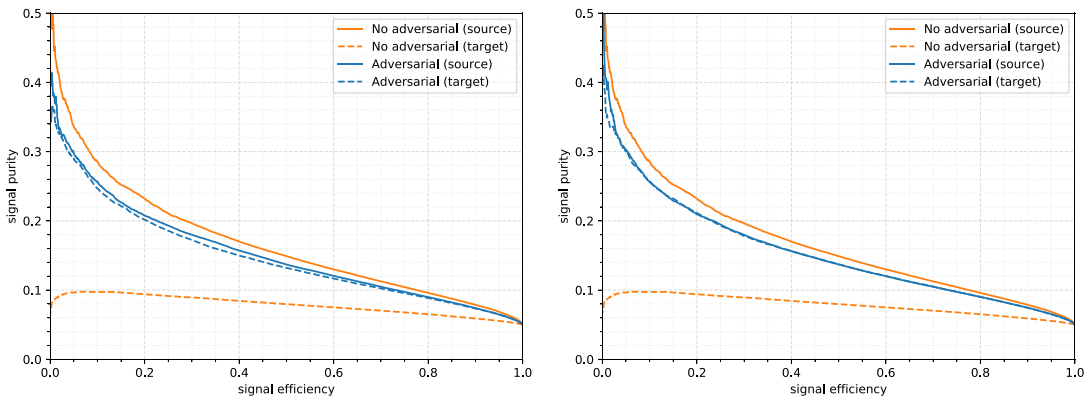


Figure 7. Signal purity p_{sig} versus signal efficiency ε_{sig} of the label predictor, calculated for the source (solid, upper line) and target (dashed, lower line) sample, with (blue) and without (orange) the adversarial architecture. (a) set-up A, (b) set-up B.

positions of the maximal AUC where source and target values agree (figure 5). For higher values of λ the significance decreases, reflecting the loss of classification power.

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 uncertainty due to the choice of the background model on the signal purity, ignoring other sources of uncertainty, can be improved from 47% to 4% (0.3%) by employing the adversarial network in set-up A (set-up B).

5.1. Extension of the method toward training with real collision data

In this study, no labels were used for computing loss of the domain classifier (except its alignment for signal and background ratio that was so far taken to be the same as for the source domain). One natural extension of the method would be to use real collision data to train domain classifier for adaptation to real data domain. However, in real collision data the ratio of signal to background is only known with limited precision obtained from previous measurements or theoretical predictions. 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*. To check the stability of our results, 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 (figure 8(a)).

However, if there is a discrepancy in the signal to background fraction between the two domains, a small bias is introduced. This behavior is shown in figure 8(b), where a fixed value of 5% was used for the source signal fraction, while varying signal fraction in the target domain. By varying the signal-to-background ratio by a factor of two away from the ratio in the source domain, a 1.4% bias was introduced on AUC which is however, still small compared to case without adversarial training. It becomes therefore important to get a good estimate for the signal to background ratio in the target domain when using unlabeled data and to properly account for the effect of this bias on the final result.

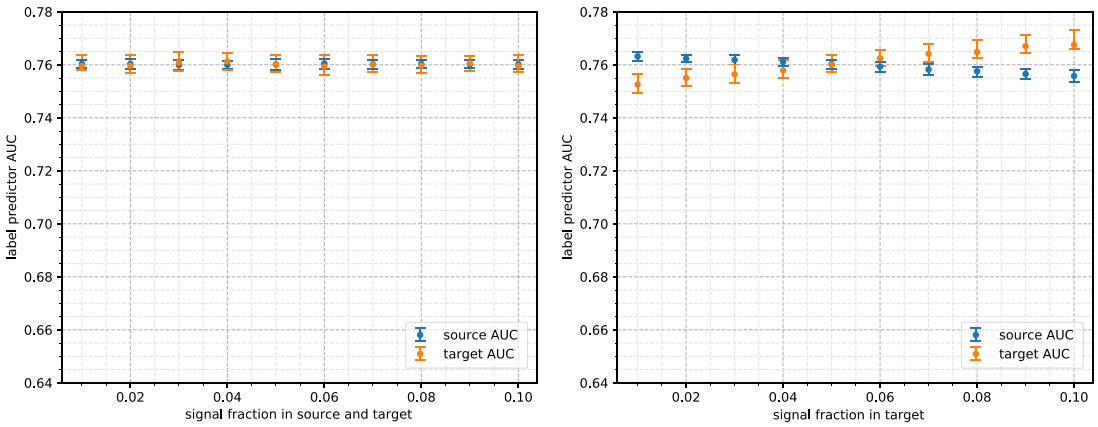


Figure 8. AUC ROC for the label predictor (a) as a function of the signal fraction in source and target and (b) as a function of the varying signal fraction in target with a fixed 5% signal fraction in the source. These plots were produced for set-up B with $\lambda = 10^{-5}$. Set-up A exhibits a similar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

We hypothesized that the gap we observe between performance for classifying events in the source or in the target domain when signal and background ratio do not match across domains may be caused by the shift of label distribution. Following number of works attempting to address this issue in unsupervised setting [8], we applied one such approach to see whether gap issue can be tackled. The implicit alignment approach [24] points out that, among other issues, the differences in label distribution may provide a harmful shortcut to identify the respective domain just on the basis of differences in label frequencies. This may strongly impair domain adaptation by ignoring actual differences in data distributions and thus not handling properly data distribution shift. To circumvent that, authors propose creating re-balanced mini-batches for training domain classifier using pseudo-labels delivered by the label predictor for target inputs, arguing for removing label frequency differences between domains in this way. We saw however that generated pseudo labels have very low reliability, which in turn seems to strongly impair the composition of re-balanced mini batches and do not result in reduction of the classification gap between domains in our case—on the contrary, the gap falls back to the state observed without any domain adaptation. This is not suitable for use case of real collision data adaptation in unsupervised setting, and makes the method in the current form rely on faithful estimate of signal to background ratio in the real collision data as pointed out above.

6. Conclusion

We successfully built a feed-forward fully connected adversarial neural network for performing domain adaptation on high energy physics data to enable event classification in the search for the $t\bar{t}H(H \to b\bar{b})$ process at the LHC. 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 a label predictor that is almost completely independent of the domain background model while preserving most of the classification power for target domain. We report the improvements using different measures. Taking the expected signal purity for a signal efficiency of 50% as a proxy measure for the sensitivity of the analysis, the uncertainty due to the choice of background model can be strongly reduced from a 47% to a 0.3% with the MC samples used in this study. Significant improvements are also reached in the approximated median significance. Although not demonstrated, we do not expect limitations when extending this approach to adapt to multiple alternative domains, i.e. sources of uncertainty, during training.

Application of our approach to target samples from real collision data was discussed where no explicit label information from target is required for training of the *domain classifier*. For the selection of optimal value for hyperparameter λ that controls the impact of adversarial domain classifier on label predictor, we designed a procedure that does not require labeled target data. However, while per input event example labels

from the real target are not necessary for training procedure, we show that in absence of a faithful estimate of the signal to background ratio for the real data target domain, misalignment of the signal to background ratio between source and target domains may lead to a small bias in the classification. This small bias and its impact has then to be addressed in a further downstream analysis.

Using a different ratio of signal to background in source and target domains introduces label distribution shift to the original formulation of the problem, in addition to the already existing data distribution shift given by the different background models in the two domains. Handling both data and label distribution shift for domain adaptation is still a largely unresolved problem in machine learning. For the unsupervised domain adaptation setting we have worked with here, our observations with a recently introduced implicit alignment approach [24] that makes use of pseudo-labels suggests that quality of pseudo labels required for such an approach to cope with both data and label distribution shift is not sufficient for our case. Application of our method to real experimental collision data adaptation in unsupervised setting in its current form will have to therefore rely on fair estimates of signal to background ratio in the real data.

As discussed above, we see the differences in label frequency between the domains which provides a shortcut for domain identification [24] and harms domain adaptation, as one central issue hampering successful domain adaptation given unknown, different signal background ratios in our case. One potential solution that we envisage for the follow-up work may employ a network architecture that uses yet another adversarial branch dealing explicitly with the task to erase harmful signal-background label information from domain classifier. Given the current progress, we anticipate that this and other advanced approaches [25] will render our method also capable of handling label shift as well and enable successful adaptation to real collision data in fully unsupervised manner.

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

Acknowledgment

We acknowledge that Ilyas Fatkhullin contributed at an early stage of the analysis [26].

ORCID iDs

J M Clavijo https://orcid.org/0000-0003-3210-1722
J M Katzy https://orcid.org/0000-0003-3121-395X

References

[1] Ganin Y *et al* 2016 Domain-adversarial training of neural networks *J. Mach. Learn. Res.* **17** 1–35
[2] Aaboud M *et al* (ATLAS Collaboration) 2018 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** 072016
[3] Ben-David S, Blitzer J, Crammer K and Pereira F 2007 Analysis of representations for domain adaptation *Advances in Neural Information Processing Systems* vol 19 pp 137–44
[4] Englert C, Galler P, Harris P and Spannowsky M 2019 Machine learning uncertainties with adversarial neural networks *Eur. Phys. J.* C **79** 4
[5] Shimmin C *et al* 2017 Decorrelated jet substructure tagging using adversarial neural networks *Phys. Rev.* D **96** 074034
[6] Louppe G, Kagan M and Cranmer K 2017 Learning to pivot with adversarial networks *Advances in Neural Information Processing Systems* vol 30 p 981
[7] Sirunyan A M *et al* 2020 A deep neural network to search for new long-lived particles decaying to jets *Mach. Learn.: Sci. Technol.* **1** 035012
[8] Mingsheng L, Zhangjie C, Jianmin W Jordan M I 2018 Conditional adversarial domain adaptation *Advances in Neural Information Processing Systems* vol 31, ed S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi and R Garnett (Curran Associates, Inc.) p 1640ff
[9] Glaysher P, Katzy J M and An S 2019 Iterative subtraction method for feature ranking (arXiv:1906.05718 [physics.data-an])
[10] Chekanov S V 2015 *Adv. High Energy Phys.* **2015** 136093
[11] Alwall J, Herquet M, Maltoni F, Mattelaer O and Stelzer T 2011 *J. High Energy Phys.* **1106** 128
[12] Corcella G, Knowles I, Marchesini G, Moretti S, Odagiri K, Richardson P, Seymour M and Webber B 2001 *J. High Energy Phys.* **2001** 010
[13] Sjostrand T, Mrenna S and Skands P Z 2006 *J. High Energy Phys.* **0605** 026
[14] Jezo T, Lindert J, Moretti N and Pozzorini S 2018 New NLOPS predictions for $t\bar{t} + b$-jet production at the LHC *Eur. Phys. J.* C **78** 502
[15] Favereau J de *et al* (DELPHES 3 Collaboration) 2014 DELPHES 3, a modular framework for fast simulation of a generic collider experiment *J. High Energy Phys.* **1402** 057
[16] Cacciari M, Salam G P and Soyez G 2008 *J. High Energy Phys.* **04** 063

[17] Aaboud M et al ATLAS Collaboration 2018 Measurements of b-jet tagging efficiency with the ATLAS detector using $t\bar{t}$ events at $\sqrt{s} = 13$ TeV *J. High Energy Phys.* **2018** 89
[18] Chollet F et al 2015 Keras (available at: https://github.com/fchollet/keras)
[19] Abadi M et al 2015 TensorFlow: large-scale machine learning on heterogeneous systems (available at: www.tensorflow.org/)
[20] Glorot X and Bengio Y 2010 Understanding the difficulty of training deep feedforward neural networks *Proc. 13th Int. Conf.on Artificial Intelligence and Stat. (AISTATS)* vol 9 (JMLR)
[21] Bergstra J, Yamis D and Cox D D 2013 Making a science of model search *Proc. 30th Int. Conf. on Machine Learning (ICML 2013)* vol 28 p I-115
[22] Kingma D P and Jimmy B 2014 Adam: a method for stochastic optimisation (arXiv:1412.6980)
[23] Adam-Bourdarios C, Cowan G, Germain C, Guuon I, Kegl B and Rousseau D 2015 *Proc. NIPS 2014 Workshop on High-energy Physics and Machine Learning* vol 42 pp 19–55
[24] Jiang X, Lao Q, Matwin S and Havaei M 2020 Implicit class-conditioned domain alignment for unsupervised domain adaptation *Int. Conf. on Machine Learning (PMLR)* pp 4816–27
[25] Prabhu V, Khare S, Kartik D, Hoffman J 2020 SENTRY: selective entropy optimization via committee consistency for unsupervised domain adaptation (arXiv:2012.11460 [cs.CV])
[26] Fatkhullin I 2019 DESY summer student program's report (available at: www.desy.de/f/students/2019/reports/Ilyas.Fatkhullin.pdf)