M1 Internship at CERN

Mathis Reymond

May - July 2023

Contents

1 Introduction

1.1 Reading remarks

Section 2 is independent from sections 3 and 4 except for sub-section 2.2. Section 4 is independent from Section 3, except for subsub-section 4.3.3.

1.2 Useful links

Here is my GitHub repository. Here is the main GitHub repository of the toy challenge.

1.3 Background and motivation

The following work was carried out during an internship, as a component of LBNL's Fair Universe project. See [1]. This project complements the previous HiggsML project. See [2].
High-energy physicists stationed at CERN heavily rely on simulations to replicate the collisions that are observed within the Large Hadron Collider (LHC). These particle collisions produce numerous smaller particles. When they propose a theory predicting the existence of a new particle, physicists use these simulations to search for evidence supporting its presence. For this purpose, they categorize the particles resulting from collisions into background particles (already known and uninteresting) and signal particles (the ones of interest). To perform this classification task, high-energy physicists are increasingly collaborating with machine learning scientists. A wealth of features are known about each particle, such as speed, energy, and angle measurements. Nevertheless, due to imperfection in the simulators, the simulated data are sensitive to systematic biases, making the classification task more challenging. Consequently, a major obstacle is to eliminate these biases from the data to enhance classification. To tackle this issue, the Fair Universe project aims at building an online challenge.

1.4 Challenge's setup

The Fair Universe's toy-challenge serves as a simplified framework of this problem. We have designed a straightforward simulator that aims to resemble a physics scenario and employs Gaussian and gamma distributions. In order to avoid working within a high-dimensional feature space, particles are simulated as 2D points, classified into either the signal or background class. The goal is to develop models capable of accurately classifying these points. The biases affecting the particles are represented as combinations of translation, rotation and scaling that affect all the points, regardless of their class. In the following sections, we explore domain adaptation techniques to establish a performance benchmark for evaluating challenge submissions. This is needed to identify relevant solutions proposed by contestants that outperform our own.

2 Domain Adversarial Neural Network

Let's introduce and compare domain adversarial models, building on prior art. Domain adversarial models are designed to perform domain adaptation, that is, to address the task of transferring classification knowledge from a source domain to a target domain, where the distributions of the data differ. We will focus on two implementations of Domain Adversarial Neural Network (DANN) : pivot implementation and

two-branched implementation. In the context of the challenge (sections 3 and 4) we only use the two-branched implementation.

2.1 Prior art : pivot implementation

They are several implementations for domain adversarial neural networks. Pivot implementation is one of them. It has been proposed in [3] and applied to overcome systematic uncertainty in physics data in [4]. See 1 for visualization of the implementation.

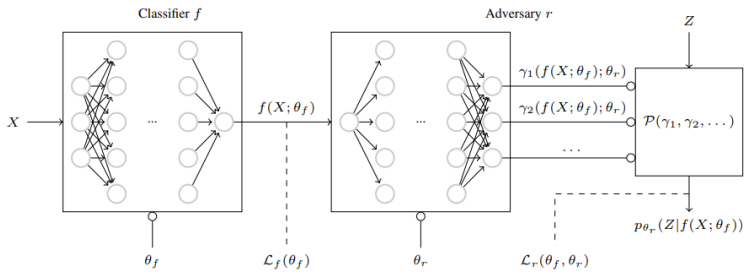


Figure 1: Pivot model's architecture

In this implementation are two components : the Classifier denoted as f and the Adversary denoted as r. f is meant to label classification. At the end of this first component is a layer with a single neuron which is used for label prediction ("signal" or "background"). This single-neuron layer is also used as input to r. r has another input Z, the value of the nuisance parameter.
To train this model, one has to generate several train sets with specific nuisance values Z, each of which correspond to one specific domain. The nuisance values used for generating these train sets have to be chosen so they span the whole nuisance range. During training, the pivot model learns label classification on domains that are a discrete approximation of all possible domains and learns to encode the input x in the first layers of f indiscriminately with respect to the domain. Once training is done, only the component f is used for predicting labels. When provided new data from an unknown domain, the component f already saw data from a close domain and knows how to encode the new data to perform domain-independent label classification.
Remarks :

1. This method involves training a single model capable of performing label classification on any test set.

2. The nuisance values of the train sets are intentionally selected to cover the entire range of nuisance.

3. The approach for utilizing nuisance values during training does not create any conflict with the fact that the nuisance values of the test sets are unknown. These values are supplied to the adversary component of the model, which is not engaged in predicting the labels after the training process.

2.2 Two-branched implementation

By comparison, [5] introduces another architecture where the network splits in two branches. Such a neural network is made of three components.

1. Feature extractor : This component is responsible for extracting meaningful features from the input data. The output layer of the feature extractor is shared between the label classifier and the domain adversary to learn domain-invariant representations of the data.

2. Label classifier : The label classifier responsible for the primary task the model is designed for. It takes the learned features from the feature extractor and predicts the labels of the input data ("signal" or "background"). The label classifier is trained to minimize the standard task-specific loss to make accurate predictions on the source domain only, as the labels of the target data is unknown.

3. Domain classifier : The domain classifier takes the learned features from the feature extractor and tries to predict the domain of the input data ("source" or "target"). The objective of the domain classifier is to maximize the domain classification accuracy while back-propagating to the feature extractor in such a way that confuses the domain classifier, making it difficult to determine the domain from the extracted features.

See 2 for visualization of the implementation.

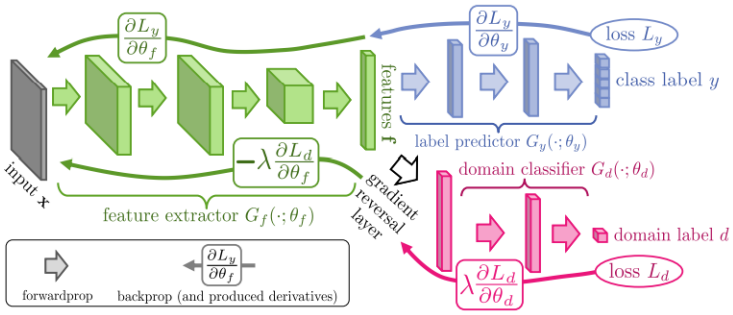


Figure 2: Two-branched model's architecture

DANN's training procedure requires to jointly train the label classifier, feature extractor, and domain classifier. The overall objective function during training can be formulated as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{label} - \lambda \mathcal{L}_{domain} \tag{1}$$

Where:

- $\mathcal{L}_{label}$ is the label classification loss between the predictions of the label classifier and the true class labels ("signal" or "background")
- $\mathcal{L}_{domain}$ is the domain classification loss between the predictions of the domain classifier and the true domain labels ("source" or "target")
- λ is a positive real number. It is a hyper-parameter that controls the importance of the domain classification loss in relation to the label classification loss.

The adversarial learning is induced by the minus sign in $\mathcal{L}_{total}$ expression. It leads the feature extractor to minimize the domain classification accuracy while the domain classifier learns to maximize it.

Training the two-branched model requires to generate a source set with no bias and the points of one test set, referred as target set. Source and target points are merged. When the model is learning from a source sample, the forward and back propagation happen both in the label classifier and the domain classifier. When training the model on a target sample, the forward and back propagation happen only in the domain classifier. As the label classifier and domain classifier learn to maximize label classification accuracy and domain classification accuracy, the feature extractor learns to minimize the domain classification accuracy while maximizing label classification accuracy. It is this adversarial objective between feature extractor and domain classifier that leads to domain-invariant extracted features at the end of the training, allowing the label classifier to predict the labels of source and target data indiscriminately. Remarks :

1. This method requires training as many models as there are test sets.
2. Each model capable of performing label classification on the test set used to train it.
3. The nuisance values of the train sets are intentionally selected to cover the entire range of nuisance.
4. The approach for utilizing test points during training does not create any conflict with the fact that the test labels are unknown.

Batch size is set as an hyper parameter. The number of batches depends on the number of epochs and the size of the source and target sets. Each training batch is half made of source samples and half made of target samples. Testing batches are fully made of target events.

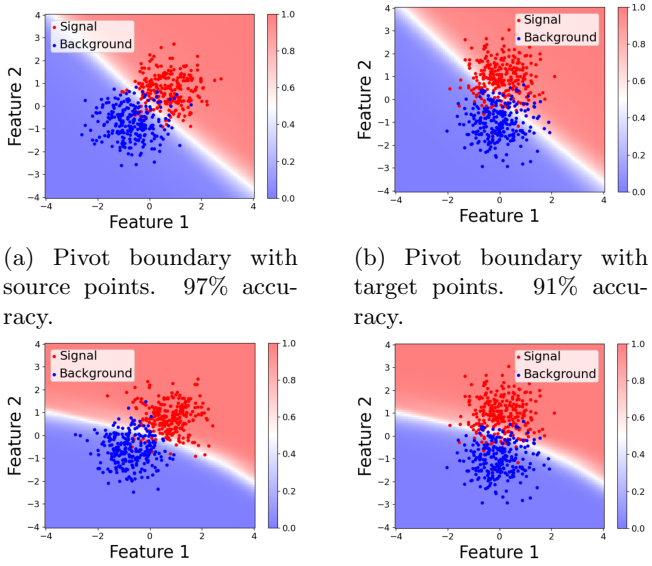
2.3 Comparison

2.3.1 [4] framework

Two objections can be raised to the pivot implementation :

1. At the end of the classifier f is a single-neuron layer on the top of which is build the adversary r. The impact of this shrinkage on the entire learning process has not been discussed yet. Contrarily, in the two-branched implementation, the domain classifier is built on the top of the feature f layer whose size is arbitrary. The implications of this difference in architecture on the learning process remain unexplored.
2. In the physics framework, the model has to predict the label ("signal" or "background") of actual LHC's events. However, all these events belong to the same domain and constitute one single test set. Hence, there might be no need to train a model to be compliant with any domain if it is possible to train a model specifically to be compliant with the test events' particular domain.

Moreover, adapting to multiple domains simultaneously can result in conflicting learning outcomes. In the context of the section discussed in [4], it leads to no adaptation at all. The systematic perturbation introduced in this context is a rotation angle, with a nominal value of $\frac{\pi}{4}$ and a nuisance value z within the range $[-\frac{\pi}{4}, \frac{\pi}{4}]$. The test (or target) set is generated with the angle set to $\frac{\pi}{2}$.
To train the pivot model, the authors generated 21 datasets with angles ranging from 0 to $\frac{20}{20}\frac{\pi}{2}$ in increments of $\frac{1}{20}\frac{\pi}{2}$. This implies that the pivot model attempts to adapt to



(a) Pivot boundary with source points. 97% accuracy.

(b) Pivot boundary with target points. 91% accuracy.

(c) Two-branched boundary with source points. 96% accuracy.

(d) Two-branched boundary with target points. 96% accuracy.

Figure 3: Decision boundaries of pivot and two-branched models

angles greater and smaller than $\frac{\pi}{4}$ simultaneously. Consequently, the decision boundary becomes optimal for an angle of $\frac{\pi}{4}$, which is equivalent to what a plain neural network would achieve. Consequently, the model performs exceptionally well on the nominal set (see 3a), which is uninteresting, but exhibits poor performance on the target set (see 3b).
In contrast, during training, the two-branched model is provided with a source set comprising the nominal value of the angle and the points from the target set. Thus, during the learning process, the model makes a trade-off between source and target domains. As a result, it may perform slightly worse on the source set (see 3c), which is not problematic since the performance on the source set is of no interest. However, it performs significantly better on the target set (see 3d).

2.3.2 Models comparison with our toy data

2.4 Further work

2.4.1 Architecture

In the pivot implementation, at the end of the classifier f is a layer with a single neuron which is used for class label prediction ("signal" or "background"). This single-neuron layer is also used as root for the adversary r part of the neural network. By comparison, the adversary (domain classifier) in the two-branched implementation is not built on the top of the label classifier. The two parts are built on the top of a feature layer whose size is arbitrary (thus, potentially greater than 1 !) which allows for no shrinkage. Intuitively, it seems natural to think that the shrinkage in the pivot implementation can impact negatively the performances of the model. Thus it might be interesting to quantify this impact. If the observed reduction in size or number of available data instances is posing a constraint, one possible approach to address

this limitation could involve exploring alternative encoding schemes for the two class labels. Rather than representing them simply as the numerical values 0 and 1, where the model's output is a single neuron ranging from 0 to 1, we could represent them as two hyperplanes (or two planes of smaller dimension, for less confusion) in an n-dimensional vector space. In this scenario, the model's output would be a vector of size n, and we would interpret the model's prediction as either "signal" or "background" depending on which hyperplane the output vector is closer to. By employing this more sophisticated encoding scheme, we may allow more information to flow from the classifier f to the adversary r, enhancing the model's ability to adapt properly to the domain.

2.4.2 Model's size

Implementing the two-branched approach in the context of physics simulation may not initially seem appealing, as it necessitates training a new model for each simulation run. However, it is important to note that in this toy-context, the two-branched model consists of only a few hundred parameters, whereas the pivot implementation involves about one hundred thousand of parameters. Therefore, it would be worthwhile to assess whether the computational cost of the two-branched implementation remains manageable in a realistic physics scenario. By conducting such an evaluation, we could determine if the benefits of reduced parameter complexity outweigh the potential drawbacks of training the model for each simulation, thus making it a feasible option for practical applications in physics simulations.

2.4.3 Shrinkage impact and potential remedy

3 Toy Challenge

3.1 Simulator

In the context of the toy challenge, we built a simple 2D simulator. Running the simulator one time produces one dataset, that is, a collection of 2D points with associated labels, either 0 for "background" class or 1 for "signal" class.

To simulate a dataset, we apply the following pipeline :

1. Provide input parameters to the simulator such as the total number of points to simulate N, the desired proportion p_b of background points, the nuisance values and distribution types (Gaussian, Poisson).

2. Define random variables B and S following a 2D probability law, one for background and one for signal.

3. Instantiate an empty dataframe with columns x_1 and x_2 to store the points and instantiate an empty list for their associated labels.

4. Sample Np_b points from B and store them in the Np_b first lines of the dataframe. Add Np_b times the label "0" to the labels list. In practise, p_b is such that Np_b is an integer.

5. Sample $N(1-p_b)$ points from S and store them in the dataframe after the background points. Add $N(1-p_b)$ times the label "1" to the labels list, after background labels.

6. Shuffle dataframe's lines and labels in parallel.

3.2 Evaluation

Contestants, are given the simulator so they can generate as much data (both 2D points and labels) as they want for training. Especially, they are able to introduce the nuisance values they want to generate training data. They are asked to predict the labels of 2D points from tests sets generated by the organizers with the same simulator. In this framework, contestants don't have access to :

- class labels of test points
- nuisance values used by organizers to generate test sets

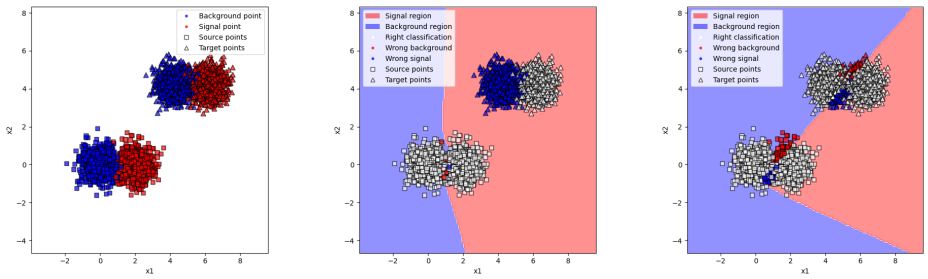
In this framework, the figure of merit is balanced accuracy. This metric is useful to take into consideration the imbalance of datasets, where there is typically only 10% signal points. However, the metric used is not our main concern in this section as we will mainly provide a qualitative analysis of DANN's behaviour in various situations.

3.3 Baselines

3.3.1 Two-branched results

See section 2.2 and [5] for two-branched model training framework.

So during the training, each point from the source set is feed-forwarded through the domain classifier and the class classifier, but each point from the target set is only feed-forwarded through the domain classifier. It means the parameters of the class classifier are not updated when the model is learning from a target point. As consequence, the model does not learn to classify events from the target set. It learns to transfer adequately its knowledge of the source set to classify the target set.



(a) One train (or source) set and one test (or target) set

(b) Classic neural network's decision region

(c) Two-branched model's decision region

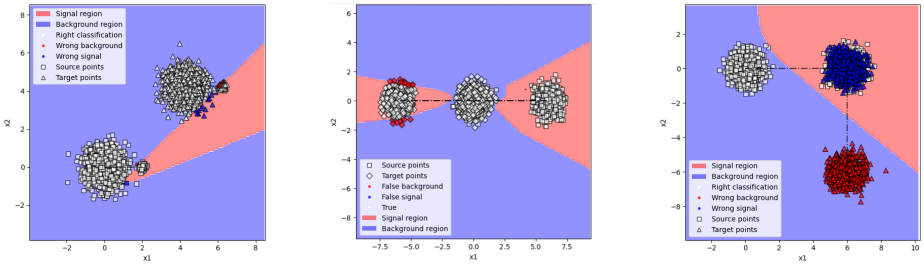
Figure 4: Classic neural network versus two-branched model behaviour

In 4a are given one source set, at the bottom, and one target set, at the top. The source set is generated by the contestant while the target one is generated by the organizers. Here we observe the the datasets from the point of view of the organiser as we visualize the target labels. Note that these target labels, despite beeing visualized in 4, are not used by the two-branched model in any way.

4b shows the decision regions of a trained neural network without domain adaptation. The neural network is simply provided the source set during training. As a result, the decision boundary is close to a vertical line separating signal from background points in the source set. However, since this naive model didn't learn to transfer its knowledge to the target set, it performs poorly on it. In this case, it classifies all the

target points as signal, resulting in a 50% balanced accuracy score on the target set.

However, the figure depicted in 4c shows the decision regions of the two-branched model. We see that they are more elaborated and, in particular, the decision boundary goes through the target set, which results in classifying it more accurately with a 94% balanced accuracy score. We also observe that the classification on the source set is slightly worse as the model did a trade of between balanced accuracy on the source set and domain adaptation. However, its no problem to loose classification power on the source set as it is generated by the contestant, only the performance on target set matters.
From a intuitive and geometrical understanding, we can say that, as compared to a simple neural network trained on the source set, the two-branched model learns to bend the decision boundary in the direction of the target set, and, sometimes, under certain circumstances, this bending results in classifying target events reasonably well.



(a) Source and target sets contain 10% signal points. Signal distributions standard deviations are a fifth of background distributions ones

(b) Nuisance is pure rotation : source and target background distributions are confounded

(c) Source distribution are horizontally aligned. Target distribution are vertically aligned

Figure 5: Three specific use cases of the two-branched model

In 5a, as seen in figure 4, we observe that the model performs accurate classification of signal versus background on both the source set and the target set to which it has transferred its knowledge. This case is similar to a physical scenario in the sense that the standard deviation of the signal distribution is small relative to that of the background, and the datasets contain only 10% of signal points.

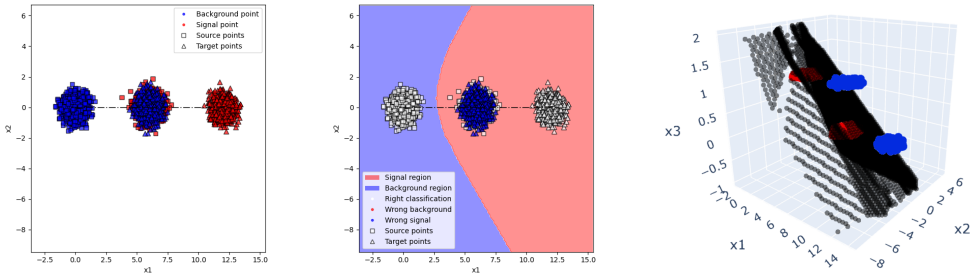
Figure 5b illustrates a pure rotation bias case with a 180-degree angle. The two background distributions overlap, and the model effectively transfers its knowledge from the source domain to the target domain. In addition to the parabolic decision boundary separating the source set, the model constructs a new decision boundary, similar to the first one, which classifies the points of the target set quite effectively.

Finally, in figure 5c, a degenerate case is shown where the source set and the target set have such a relative proportion that transferring knowledge from the source set to the target set results in constructing a decision boundary that perfectly separates the target set but with 100% error. This case illustrates a limitation of the two-branched model, which, without access to the labels of the test set, can operate correctly but produce very poor classification. However, it is worth noting that this case is degenerate, and in practice, with smaller nuisance magnitudes, the model's performance should not degrade as much.

3.3.2 3D two-branched

In this section we introduce an improvement for DANN that we apply to the two-branched architecture.

If nuisance magnitude is sufficiently big, it is possible to find cases as depicted by 6a, where geometrical limitations hamper model's learning. In this case, source signal and target background distributions are overlapping.



(a) Source signal points overlap with target background points, making it harder for the model to achieve accurate classification on the target set.

(b) Decision regions of the two-branched model after training. Classification is accurate on the source set, but the model failed to transfer classification knowledge to the target set.

(c) Decision surface of the improved two-branched model after training. Source (resp.target) points have $x_3 = 1$ (resp. $x_3 = 0$). red : true signal points. blue : true background points. black : decision surface.

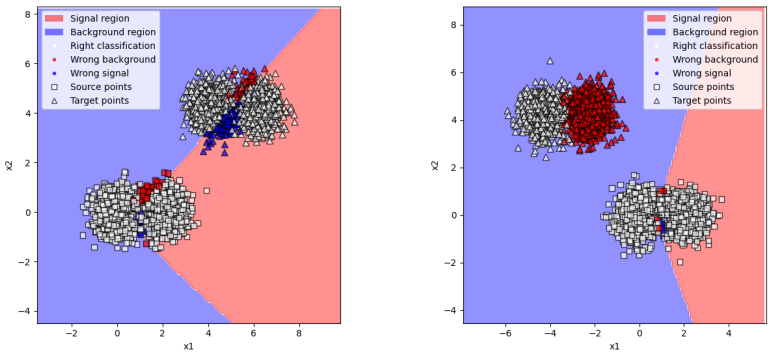
Figure 6: Classic two-branched versus improved two-branched

In these cases, the two-branched model struggles to efficiently extend the decision boundary for accurate classification on the target set. See 6b. To enhance this, creating an additional dimension in the feature space by assigning a unique value (x_s for source samples and x_t for target ones) alongside the existing x_1 and x_2 could potentially yield improved classification. This approach guarantees the absence of overlap between the source and target distributions. It also enables the two-branched model to leverage information about the set membership of each sample both from the domain classifier and this new feature. Note that this technique is legitimate within the context of a challenge. Consequently, the model effectively extends the decision boundary toward the target set, achieving precise classification. As a result, in the specific example depicted in 6, the source signal and target background distributions are no longer overlapping, and the two-branched model can propagate the decision separation in the direction of the target set in a way that results in an accurate classification. See 6c.

Although very effective on cases with big nuisance magnitude, a quantitative analysis of the performances of this augmented version of DANN reveals no substantial improvement on toy-simulated data with reasonable nuisance magnitudes. However, it still awaits testing on data more closely aligned with physics scenario. Upon such testing, its potential could be unleashed, given its presumed greater adaptability and flexibility as compared to regular DANN.

3.4 Bending problem

In this paragraph, we will discuss a problem that we observed with DANN. It arose when making with qualitative test, with big nuisance values. First, something we didn't questioned yet is the specific shape of the decision boundary. For instance, in 7a, the upper part of decision boundary goes through the test set, which is expected, but the behaviour of its lower part is not controlled, and we find no trivial reason for it to make such an angle with the upper part. Intuitively, it would be reasonable if the decision boundary was a simple line, of if it was vertical bellow the source set. Moreover, we observed that in 7b, DANN failed to propagate the decision boundary in the direction of the test set, despite the fact that data generation is analogue to 7a : nuisance is 6 magnitude translation but the translation direction is given by the angle $\frac{3\pi}{4}$ instead of $\frac{\pi}{4}$. Providing more training epochs fixes this problem. However, it suggests that the DANN has preference for bending the decision boundary toward positive x_1.



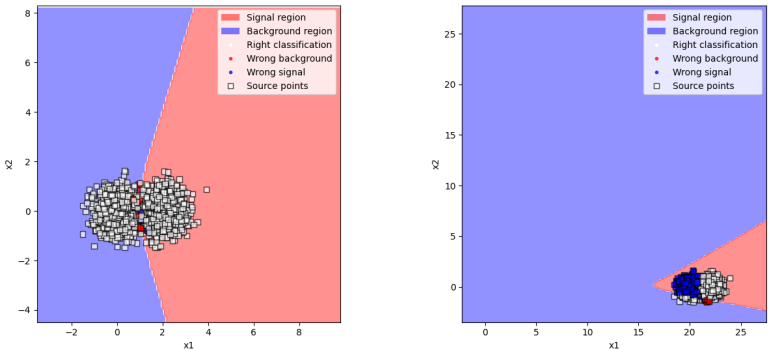
(a) (b)

Figure 7: Classic two-branched versus improved two-branched

Trying to understand the origin of the problem, we came back to a simpler model. We set the λ parameter to 0 so the DANN reduces to a simple neural network with no adversarial learning. As we observe in 8a, the decision boundary has the same shape as in 7b : not as bent as in 7a, but clearly not a vertical line. And as we repeat the training process, the decision boundary always bend toward positive x_1, which implies that this bending is not a learning instability, in which case, the decision boundary would have bent sometimes toward positive x_1 and sometimes toward negative x_1. In such a simple framework, the only remaining asymmetry lies in a the source data. And it appears that in our data generating process, we arbitrarily decided to center the background distribution in $(0, 0)$. In the specific example depicted in 7 ad 8, the signal distribution is centered in $(2, 0)$. As a result, there is an asymmetry of the source data with respect to the x_2 axis as its barycenter is in $(1, 0)$. Here, the Pierre Currie's principle finds an application : "When certain effects show a certain dissymmetry, this dissymmetry must be found in the causes which gave rise to them". It appears that the asymmetry observed in DANN behaviour - decision boundary bent toward positive x_1 - arise from the asymmetry of source data - most of the source samples have positive x_1.

Moreover, we can say that this asymmetry is an absolute asymmetry, as opposed to a relative asymmetry, in that sense that signal and background distributions are

symmetric with respect to there barycenter (relative symmetry) but are not symmetric with respect to the x_2 axis (absolute asymmetry). It reminds us that while we, humans, use our eyes and rely only on the relative position of the points of each class to perform classification, a neural network, him, rely only on the absolute values of the samples to perform the classification. A fortiori, after a 20 magnitude translation along x_1 of the source set, the neural network struggles even more to perform the classification. See 8b. Moreover, if the barycenter of the source set has negative x_1 coordinate, we observe that the decision boundary is always bent toward negative x_1. Furthermore, if the barycenter has x_1 coordinate equal to 0, the source set becomes absolutely symmetric. The decision boundary keep bending, but it bends as often toward positive x_1 as it bends toward negative x_1.



(a) (b)

Figure 8: Classic two-branched versus improved two-branched

Finally, the specific shape of the decision boundary (parabolic, in examples 7, 8) is determined by the activation function and output function of the neural network (ReLU and Softmax in previous examples). If we use hyperbolic tangent as activation function, then the decision boundary is no longer parabolic and has sn "S" shape. If we use a linear activation function, then the decision boundary is a line but in the context of the bending problem, the line was always bent in the same direction. Thus, the bending cannot be solved by changing the activation function, it is inherent to Softmax neural networks.

A pending question remain : will this bending problem negatively impact DANN's performances on real physics data ? Fortunately, in a realistic scenario, the impact of this issue is lessened by two factors :

1. the nuisance magnitude is so small that it is not possible to perform the classification of labelled data by eye. For example, if the nuisance was a translation, it would not be possible to determine the translation direction by looking at the points.

2. physicist apply preprocessing techniques to prevent this kind of issues.

See section 4.5 for a quantitative evaluation of the problem in a more realistic scenario.

We observed that DANN's behaviour was not symmetrical in that sens that in the case of a translation nuisance with signal and background horizontaly aligned, it was

harder for it to propagate decision boundary to a target set at his right than at his left PIC. Then we tried to set the λ parameter to 0, which is supposed to anihilate the adversarial behaviour, rendering DANN to be a simple neural network with no learning on the target set. In such a context, we would have expected the decision boundary to be vertical line, but it is not. The shape of the decision boundary is a parabola bent toward the right and, which is more problematic, it is a constantly bent toward the right. The reason for the shape is the use of relu activation function, using hyperbolic tangent produce another curve for the decision boundary PIC. Moreover, using a linear activation produce a line for the decision boundary, however, it is never vertical but persistently titled in the same direction PIC. The reason for this shift in the decision boundary despite the symetrical appearance of the two distributions is that, as opposed to other methods, neural networks takes absolute position as input, not only relative position of the points. Thus, if the relative position of the points remains the same but they are send far from the origin, the model would perform poorly to classify them PIC. Then, the learning on the source set could be improved by adding more compexity to the model, which would results to a better classification on the source set. However, outside the region of the source points, the decision boundary remains uncontrolled, which is a problem to classify target points as the domain adaptation would be impacted. The point is that when the lambda parameter is not nul, the DANN behaves as a regular neural network biased in a favorable way by the adversary ; thus, the imperfections of the behaviour of a regular neural network will reverberate on DANN's behaviour.

3.5 Further work

- Coping with instability - 3D DANN : distance on the third axis - Play with feature space - Improve domain classifier - Machine learning class label classifier

4 Physics Update

4.1 Simulator

A real physics simulator generates a lot of events, most of which is obvious background events. Thus, a first step is to get rid off these events. Even after this filtering, the background events remain far more numerous than signal events in datasets (roughly two orders of magnitude).

We represent the number of background events by a random variable which is assumed, according to previous experiments, to follow a Poisson distribution with parameter $\beta \in \mathbb{R}_+^*$. We assume, from theoretical knowledge, that we can represent the number of signal events by a random variable following a Poisson distribution with parameter $\gamma \in \mathbb{R}_+^*$. We note $\mu \in \mathbb{R}_+$, the outcome of a random variable that represents the shift in the number of signal events. Finally, we define $\nu = \beta + \mu\gamma$, the parameter of the Poisson law followed by the random variable that represents the total number of events observed in a simulation run. The μ parameter represents how much the expected number signal events simulated deviates from what is claimed by theory.

Virtually, to simulate a dataset, we follow these steps :

1. we arbitrarily set π, the expected balanced between background and signal events as stated by theory.

2. we arbitrarily set ν_1, the expected number of events if $\mu = 1$. This choice determines the approximate scale of the dataset's size.
3. we compute the expected number of background events, $\beta = \nu_1\pi$
4. we compute the expected number of signal events stated by theory, $\gamma = \nu_1(1-\pi)$
5. we sample one value of μ
6. we compute $\nu = \nu_1(\mu\pi + (1-\pi))$ the expected number of events in the dataset.
7. we compute $p_b = (1-\pi)/(\mu\pi + (1-\pi))$ the proportion of signal events in the dataset.
8. we compute $p_s = \pi\mu/(\mu\pi + (1-\pi))$ the proportion of background events in the dataset.
9. we draw N from Poisson(nu) for the actual number of events in the dataset.
10. we draw z, the nuisance value, uniformly from a reasonable range.
11. we generate Np_b signal events and Np_s background events, each of which is biased by z.

Thus, each simulated dataset has a specific nuisance value, a specific number of events and a specific background to signal ratio.

In this framework, we don't allow contestants don't have access to :

- class label of test events
- z value of test sets
- μ value of test sets

4.2 Evaluation

In this framework, contestants are asked to estimate μ for each given test set. The goal is to mimize $|\Delta\mu| := |\mu - \hat{\mu}|$, where $\hat{\mu}$ is the μ estimation performed by contestants. Thus, the classification task is combined to this new estimation task.

4.3 Baseline

As a baseline for μ estimation, we propose the following pipeline

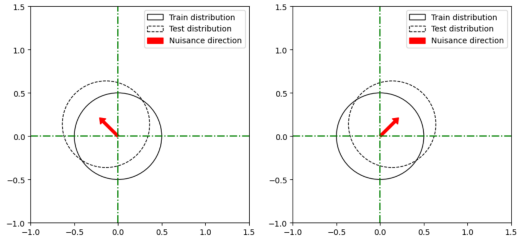
1. Given a dataset with unknown μ
2. Generate training dataset with $\mu = 1$
3. Train a classification model
4. Choose a threshold for selecting signal. It defines a Region Of Interest (ROI)
5. Count b_{ROI}, the number of false positive in training set
6. Count s_{ROI}, the number of true positive in training set
7. Count n_{ROI}, the number positive in test set
8. Compute $\hat{\mu} = \frac{n_{ROI} - b_{ROI}}{s_{ROI}}$, the estimation of μ.

4.4 Method and results

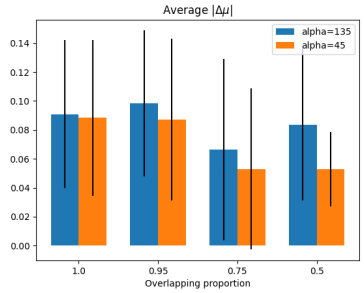
We did not conduct an overarching assessment of the models through a global test.

4.5 Bending problem quantification

In this section, we employ this novel physics framework to assess the extent to which the bending issue may pose a problem. See 3.4. We consider a translation nuisance either with a direction angle of 45° or 135°. As earlier, source background distribution is centered in $(0, 0)$, which causes an asymmetry with respect to the x_2 axis. To assess how "far" the source domain is from the target domain, we introduce a overlap percentage between source and target distributions. Contrary to the nuisance magnitude, this metric takes into consideration the data dispersion. More precisely, it measures the percentage of overlap between the source and target one-σ background distribution circles. See 9a for illustration.



(a) Gaussian distributions with a standard deviation of 0.5 are represented by one σ circles. In both cases, the training and testing circles overlap by 75%, resulting in a center-to-center distance of 0.195 units.



(b) Average of μ estimates across 12 datasets with various overlapping proportions. Alpha represents the angle defining the translation.

Figure 9: Evaluating bending problem's impact on two-branched model's performance

We consider 4 overlapping proportions : $1, 0.95, 0.75$ and 0.5. An overlapping proportion of 1 means the nuisance magnitude is 0, it is a control case, where we expect no difference in results between direction angle of 45° and 135°. Overlapping proportion of 0.95 is close to physics nuisance impact on simulated data. We set parameters for simulation. They will remain the same except for the magnitude and direction of the nuisance translation. For each overlapping proportion and each angle direction, we generate 10 couples source/test sets within the physics framework stated above. For each of couple, we train a two-branched model and evaluate it by estimating μ and computing $|\Delta\mu| := |\mu - \hat{\mu}|$. Finally, we average the 10 $|\Delta\mu|$ and report it in 9b.
We have observed that, on average, the magnitude of $|\Delta\mu|$ tends to be slightly smaller in cases with a translation direction of 45° when compared to cases with a translation direction of 135°. This indicates that a 45° translation results in a more accurate estimation of μ. Furthermore, we have noticed that the difference in $|\Delta\mu|$ averages between the 45° and 135° translations increases as the overlapping proportion decreases. This suggests that the impact of the bending problem becomes more pronounced as the translation magnitude increases.
However, due to the high variance in $|\Delta\mu|$ averages, it is challenging to conclude that the bending problem will significantly affect the performance of the two-branched

model on real physics data, where the degree of overlap is high. Additionally, preprocessing steps carried out by physicists should further help mitigate the impact of the bending problem.

5 Acknowledgements

First and foremost, I would like to express my sincere gratitude to Isabelle Guyon and David Rousseau for their exceptional guidance throughout this research endeavor. I am thankful to Ihsan Ullah for his technical support with the Fair Universe project and assistance in coding. I would also like to acknowledge Ragansu Chakkappai for his enlightening remarks about my work. His constructive feedback and thought-provoking discussions about DANN have genuinely guided me to explore various perspectives and refine my ideas.

References

[1] Patrick Riley. *New 'Fair Universe' Project Aims to Build Supercomputer-Scale AI Benchmarks for HEP and Beyond*. Nov. 2022. URL: https://cs.lbl.gov/news-media/news/2022/new-fair-universe-project-aims-to-build-supercomputer-scale-ai-benchmarks-for-hep-and-beyond/.

[2] Claire Adam-Bourdarios et al. "The Higgs boson machine learning challenge". In: *Proceedings of the NIPS 2014 Workshop on High-energy Physics and Machine Learning*. Ed. by Glen Cowan et al. Vol. 42. Proceedings of Machine Learning Research. Montreal, Canada: PMLR, Dec. 2015, pp. 19–55. URL: https://proceedings.mlr.press/v42/cowa14.html.

[3] Gilles Louppe, Michael Kagan, and Kyle Cranmer. *Learning to Pivot with Adversarial Networks*. 2017. arXiv: 1611.01046 [stat.ML].

[4] Aishik Ghosh, Benjamin Nachman, and Daniel Whiteson. "Uncertainty-aware machine learning for high energy physics". In: *Physical Review D* 104.5 (Sept. 2021). DOI: 10.1103/physrevd.104.056026. URL: https://doi.org/10.1103%2Fphysrevd.104.056026.

[5] Yaroslav Ganin et al. *Domain-Adversarial Training of Neural Networks*. 2016. arXiv: 1505.07818 [stat.ML].