

Application of Machine Learning techniques to the discovery of new physics

Óliver Partida^{a,c} (Researcher), Pere Masjuan^{b,d}



ARTICLE INFO

Keywords:

standard model
B-physics
neural networks
generative adversarial networks

ABSTRACT

This template helps you to create a properly formatted L^AT_EX manuscript.

`\beginabstract ... \endabstract` and `\begin{keyword} ... \end{keyword}` which contain the abstract and keywords respectively.

Each keyword shall be separated by a `\sep` command.

1. Introduction

Flavour Changing Neutral Current (FCNC) are interactions that change the flavor of a fermion without altering its electric charge. FCNC processes are forbidden at tree level in the Standard Model and highly suppressed at higher orders by the GIM mechanism. The GIM mechanism required the existence of a fourth quark in order to explain the suppression in loop diagrams of FCNC.

This makes FCNC one of the key processes to search for physics beyond the standard model, so called New Physics (NP), since any small deviations from the Standard Model expectations could have a big impact. Over the last few years, many observables related to the FCNC transitions $b \rightarrow sl^+l^-$ have exhibited deviations from SM expectations. Due to their suppression within the SM, these transitions are well known to have a high sensitivity to potential NP contributions. These anomalies can be classified in two sets: $b \rightarrow s\mu\mu$ related to observables testing only muonic transitions, called Lepton Flavor Dependent (LFD), and Lepton-Flavor Universality Violating (LFUV) anomalies that correspond to deviations in observables comparing muonic and electronic transitions. In 2013, using the 1 fb^{-1} dataset, the LHCb experiment measured the basis of optimized observables for $B \rightarrow K^*\mu\mu$, observing the so-called P'_5 anomaly, i.e., a sizable 3.7σ discrepancy between the measurement and the SM prediction in one bin for the angular observable P'_5 (Figure 1). A new discrepancy in the ratio $R_K^* = Br(B \rightarrow K^*\mu\mu)/Br(B \rightarrow K^*ee)$ (Figure 2) was also observed by LHCb, hinting at the violation of Lepton Flavor Universality (LFU) and suggesting that deviations from the SM are predominantly present in $B \rightarrow K\mu^+\mu^-$ transitions but not in $B \rightarrow Ke^+e^-$ ones. In order to evaluate the significance and coherence of these deviations, a global model-independent fit is the most efficient tool to determine if they contain patterns explained by NP.

The starting point is an effective Hamiltonian in which heavy degrees of freedom (the top quark, the W and Z bosons, the Higgs and any heavy new particle) are integrated out in short-distance Wilson coefficients C_i , leaving only a set of

operators \mathcal{O}_i describing the physics at long distances:

$$\mathcal{H}_{eff} = -\frac{4G_F}{\sqrt{2}} V_{tb} V_{ts}^* \sum_i C_i \mathcal{O}_i.$$

In the SM, the Hamiltonian contains 10 main operators with specific chiralities due to the V - A structure of the weak interactions. In presence of NP, additional operators may become of importance. Current analysis of anomalies in flavour physics are based on a linear regression of a χ^2 function. After taking into account correlation between theoretical predictions and experimental observables the χ^2 is built and minimized. For each measured observable, we have a theory prediction based on the Standard Model of Particle Physics (SM). With 180 observables, the χ^2 value of the SM reaches 225 points [1], which corresponds to a p-value of 1.4%. This indicates the SM to be very far to explain experimental measurements globally. The strategy then has been to include on top of the SM, new operators in the effective Hamiltonian to be able to account for such experimental discrepancies. In [1] including measurement updates (R_K, R_K^* and $B(B_s \rightarrow \mu^+\mu^-)$) a global model-independent analysis is performed yielding very similar results to the ones previously found in (Ref. [3],[4]) for the various NP scenarios of interest. Among the possible operators containing NP, in this exercise, which is a proof of concept more than an exhaustive analysis of data, we reduce such set to the two dominant operators (Ref. [1],[2],[3]):

$$\mathcal{O}_9 = \frac{e}{16\pi^2} m_b (\bar{s} \gamma_\mu P_L b) (\bar{\ell} \gamma^\mu \ell),$$

$$\mathcal{O}_{10} = \frac{e}{16\pi^2} m_b (\bar{s} \gamma_\mu P_L b) (\bar{\ell} \gamma^\mu \gamma_5 \ell).$$

Even though we limit ourselves to operators with left-handed chirality, in a most general framework one should include right handed operators.

In this work we want to try a different approach and instead of using a standard χ^2 minimization global fit we used several different machine learning techniques. We picked techniques from the two main approaches to solving machine learning problems, namely generative and discriminative. More precisely, we first tried simple neural networks, which belong to the discriminative methods, to find a mapping between C_9 and C_{10} coefficients and the corresponding

ORCID(s): 0000-0001-7511-2910 (Ó. Partida)

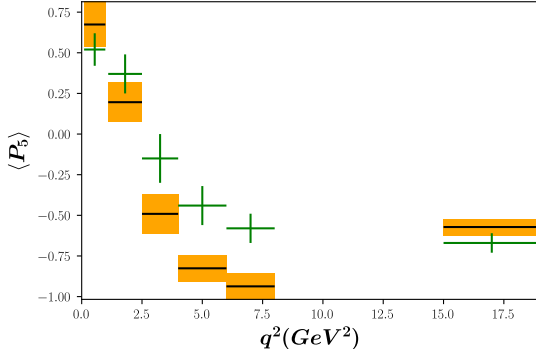


Figure 1: $P_5 = B \rightarrow K^* \mu \mu$. SM predictions (orange boxes) and experimental values (green crosses).

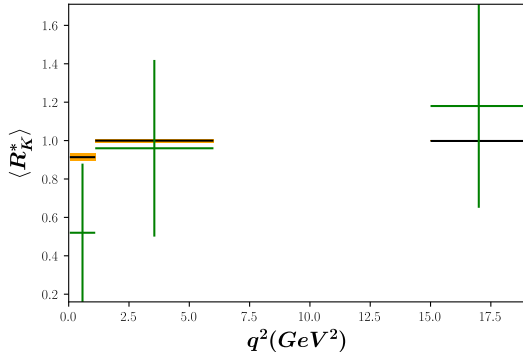


Figure 2: $R_K^* = Br(B \rightarrow K^* \mu \mu) / Br(B \rightarrow K^* e e)$. Ratios with different lepton at the final state, R_K ratios, are a clear indication of the Lepton Flavor Universality Violation something which is not expected from the theory.

model-generated bin values. After this we tried generative adversarial networks which are a novel approach to learning models from data in an unsupervised way. By learning the statistics of the training data set our hope was that new data with same statistics could be generated from the model and therefore be able to find better coefficients to globally fit our experimental data. Furthermore, we explored in this work the fact that a global fit to data using NP models does not imply that we should neglect Standard Model contributions, namely, hadronic contributions from QCD expressed through form factors. We model this hadronic uncertainties by introducing a second degree q^2 -dependent polynomial function. We explored two different scenarios depending on whether these coefficients are shared by the observables or independent. In chapter 2 we briefly introduce generative and discriminative models, the two main approaches to solving machine learning problems, and describe the two main techniques used in this work, Neural Networks and Generative Adversarial Networks. In chapter 3, we very briefly describe our original data. In chapter 4 we present our results and chapter 5 contains some conclusions and future work.

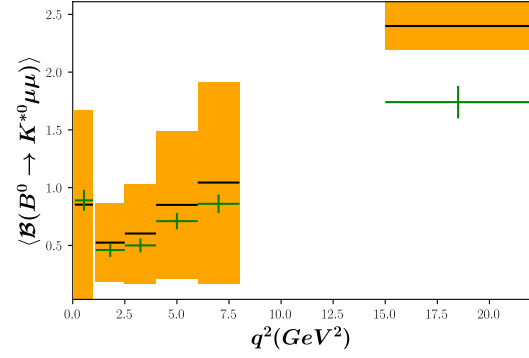


Figure 3: $B(B^0 \rightarrow K^{*0} \mu \mu)$. Branching ratios.

2. Machine learning techniques: Neural Networks And Generative Adversarial Networks

2.1. Generative vs. Discriminative models

Generative and discriminative models are the two main types of models used to solve machine learning classification problems. In classification tasks we are usually interested in learning the conditional distribution $p(C_k | \mathbf{x})$, where x is the observation and C_k is the corresponding class assigned to this observation, and then use this conditional distribution to make class assignments. Generative models tackle this problem by first determining the class-conditional distributions $p(\mathbf{x} | C_k)$ and the prior class probabilities $p(C_k)$ and the using Bayes' theorem

$$p(C_k | \mathbf{x}) = \frac{p(\mathbf{x} | C_k) p(C_k)}{p(\mathbf{x})}$$

to find the posterior class probabilities $p(C_k | \mathbf{x})$. It is also possible to model the joint distribution $p(\mathbf{x}, C_k)$ directly and normalize to obtain the posterior probabilities. Data from the input space can be generated by sampling from the learned model.

Discriminative classifiers, on the other hand, model the posterior $p(y | x)$ directly or learn a direct map from inputs x to class labels. Discriminative models that use probability distributions to solve the classification problem are called probabilistic discriminative models. For example, in the case of two classes, the posterior probability

$$p(C_1 | \mathbf{x}) = \frac{p(\mathbf{x} | C_1) p(C_1)}{p(\mathbf{x} | C_1) p(C_1) + p(\mathbf{x} | C_2) p(C_2)}$$

can be rewritten as

$$\frac{1}{1 + \exp(-a)} = \sigma(a),$$

where

$$a = \ln \frac{p(\mathbf{x} | C_1) p(C_1)}{p(\mathbf{x} | C_2) p(C_2)}$$

and $\sigma(a)$ is the logistic sigmoid function. For some specific conditional distributions $p(\mathbf{x}|C_k)$ the argument of the sigmoid function is a linear function of the inputs \mathbf{x}

$$a = \mathbf{w}^T \mathbf{x} + \mathbf{w}_0.$$

In discriminative modeling we can use maximum likelihood to directly find the parameters \mathbf{w} .

2.2. Neural Networks

Usually linear models for regression and classification are based on linear combinations of fixed nonlinear basis functions

$$y(\mathbf{x}, \mathbf{w}) = f\left(\sum_{j=1}^M w_j \phi_j(\mathbf{x})\right),$$

where $f(\cdot)$ is a nonlinear activation function in the case of classification and the identity in the case of regression. For example in polynomial regression when there is only one input variable the set of basis function $\phi_j(\mathbf{x})$ take the form:

$$[1, x, x^2, x^3 \dots]$$

One limitation of polynomial basis functions is that they are global functions of the input variable so input space regions are not independent. This problem can be alleviated by fitting different polynomials to different regions of space leading to *spline functions*. Neural networks make basis functions depend on adjustable parameters that are *learned* along network coefficients w_j during training. In the basic neural network model with just one hidden layer the output of each basis function is the result of applying a nonlinear function $h(\cdot)$ to a linear combination of the input variables x_1, \dots, x_D :

$$a_j = \sum_{i=1}^D w_{ji} x_i x_{j0}$$

$$z_j = h(a_j).$$

These values are again linearly combined and transformed using an appropriate function to give the final output:

$$a_k = \sum_{j=1}^D w_{kj} z_j w_{k0}$$

$$y_k = g(a_k).$$

For regression problems g is the identity so $y_k = a_k$. For binary classification g is the sigmoid function:

$$g(a) = \frac{1}{1 + \exp(-a)}.$$

The goal of machine learning algorithms is to produce a model that generalizes, that is, that predicts previously unseen observations. Overfitting occurs when a model fits the data in the training set well, while incurring larger generalization error. The reason is, models can be too complex and therefore able to memorize the training dataset, when


shown an example not previously seen in the training set. Early stopping is a technique that can be used to avoid overfitting. It allows you to stop the training process after a given number of iterations if the model performance on the test set decreases or stays constant. Regularization is another commonly used technique to avoid overfitting. It reduces the model complexity by adding a penalty term to the loss function. By reducing the absolute value of the weights only smooth functions are allowed. Another form of regularization which has appeared more recently in the context of neural networks is *Dropout* which reduces the model complexity by randomly dropping neurons from the neural network during training in each iteration.

2.3. Conditional Generative Adversarial Networks

Generative Adversarial Networks (GANs) are classified within the group of generative models first described in the 2014 paper by Ian Goodfellow [5]. GANs are a clever way of training a generative model by framing the problem as a supervised learning problem. They consists of two adversarial models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G . The generator's G loss during training depends on the probability of the discriminative model of making an error by assigning the class *real* to *fake* data, that is, data generated by the generator. Other generative models such as Gaussian Mixture Models are mostly based on maximum likelihood estimate however maximum likelihood estimation may not represent the complexity of the actual data distribution and cannot learn the high-dimensional data distributions. The generator in a GAN model takes a fixed-length random vector $z \sim p(z)$ as input and generates a sample $x \sim P_{data}^*(x)$ in the domain where $P_{data}^*(x)$ is the true distribution.

Conditional Generative Adversarial Networks are an extension of GANs where a conditional setting is applied. The generator receives a new input along the random variable z which adds extra information about the sample to be generated. Similarly the discriminator is trained to classify samples that incorporate this new information.

3. Data preparation and implementation

In this work we have included 37 observables. Observable is referred to a measurement of either an angular observable, a branching ratio, or a ratio, in a particular energy bin. We have included the measurements in different energy bins of the angular observables, P_1, P_2, P_4, P_5 , the branching ratios BrK^0, BrK^{0*} , and the ratios R_K, R_{K^*} performed by the LHCb collaboration. 

4. Results

4.1. Neural Network

4.1.1. Experiment 1

We randomly generated 2500 model coefficients pairs (C_9, C_{10}) by generating 50 values for $C_9 \in [-2, 0]$ and 50 values for $C_{10} \in [-1, 1]$ and forming the Cartesian product

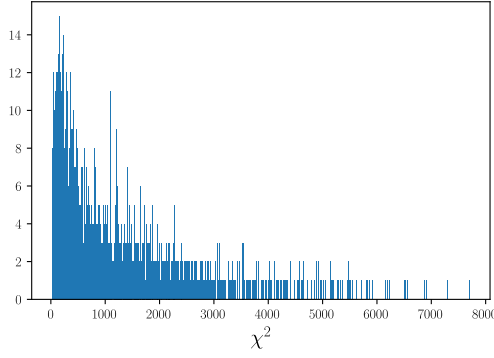


Figure 4: Neural network experiment 1. χ^2 value distribution of samples in the training set.

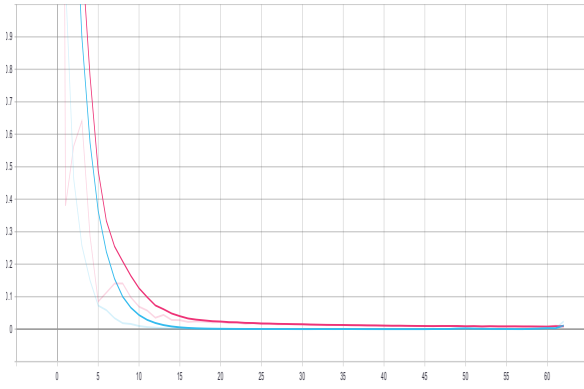


Figure 5: Neural Network experiment 1. Epoch mean square error training (blue) and validation (purple).

$(C_9, C_{10}) : C_9 \times C_{10}$. For each pair, all 37 predicted observable values were arranged in a 37-dimensional vector.

$$\mathbf{x} : \boxed{Obs_n}$$

$$\mathbf{y} : \boxed{C_9 \mid C_{10}}$$

where $n = 1, \dots, 37$. The pair $C_9 = -0.92$ and $C_{10} = 0.1$ generates a sample with $\chi^2 = 30.67$, the lowest in the training set. The result corresponds with the estimates of global fits based on minimizing the χ^2 function (Ref. [1]).

The neural network consisted of a 37-neuron input layer, one hidden layer with 2 neurons with RELU activation function and a 2-neuron output layer. We divided the original sample into a training (90%) and a validation set (10%). We selected a batch size of 512 and trained the model during 100 epochs with early stopping on validation mean square error equal to 10 to avoid overfitting. In figure 5 we see that the network has a low mean square error on both training and validation sets. After training, the network predicts a pair $C_9 = -0.75, C_{10} = 0.13$ with a $\chi^2 = 30.85$. We found a χ^2 very similar to the minimum in the training set but with different coefficients. The network is predicting these coefficients as the ones most likely to have generated the experimental data based on the examples it was given.

4.1.2. Experiment 2

We randomly generated 100 model coefficients pairs (C_9, C_{10}) by generating 10 values for $C_9 \in [-2, 0]$ and 10 values for $C_{10} \in [-1, 1]$ and computed the predicted values for each of the different energy bins. This time, to account for the hadronic uncertainties emerging from form factor contributions, a function $F(q^2) = a_0 + a_1 q^2 + a_2 (q^2)^2$ is added to each observable bin. The coefficients were generated by sampling from a random uniform distribution in the range $[-0.01, 0.01]$. Coefficients a_1 and a_2 were further scaled dividing by 100 and 1000 respectively. We selected 100 different coefficients a_0, a_1, a_2 for each of the observables $P_1, P_2, P_4, P_5, BrK^0, BrK^{0*}, R_K, R_{K^*}$.

$$\mathbf{x} : \boxed{Obs_n}$$

$$\mathbf{y} : \boxed{C_9 \mid C_{10} \mid a_{ij}}$$

where $n = 1, \dots, 37, i = 0, 1, 2, j = 1, \dots, 8$ and a_{ij} is the i coefficient of observable j .

The neural network consisted of just one hidden layer with only two neurons. After testing different network configurations we realized that adding many neurons or layers resulted in quickly overfitting and the neural network predicting values with high χ^2 values. In figure 7 we see that the mean square error decreases for both the training and the validation sets. We use early stopping to avoid overfitting so the training process is stopped if the mean square error does not decrease after 10 iterations.

When generating the training dataset to feed the neural network we found the sample with the lowest χ^2 value with the following coefficients:

$C_9 = -0.89, C_{10} = 0.11, \chi^2 = 28.19$	a_0	a_1	a_2
P_1	0	0	0
P_2	-0.01	0	0
P_4	0	0	0
P_5	-0.01	0	0
BrK^{0*}	-0.01	0	0
BrK^0	0	0	0
R_K	0.01	0	0
R_{K^*}	0	0	0

After training the neural network, it predicted the following coefficients for the experimental samples:

$C_9 = -0.79, C_{10} = -0.07, \chi^2 = 29.5$	a_0	a_1	a_2
P_1	0	0	0
P_2	0	0	0
P_4	0.03	0	0
P_5	0.02	0	0
BrK^{0*}	-0.01	0	0
BrK^0	-0.01	0	0
R_K	-0.01	0	0
R_{K^*}	-0.01	0	0

Again, as in experiment 1 above, the neural network is predicting for the experimental data, coefficients which give good χ^2 values, similar to the minimum in the training set, but with different C_9 and C_{10} coefficients.

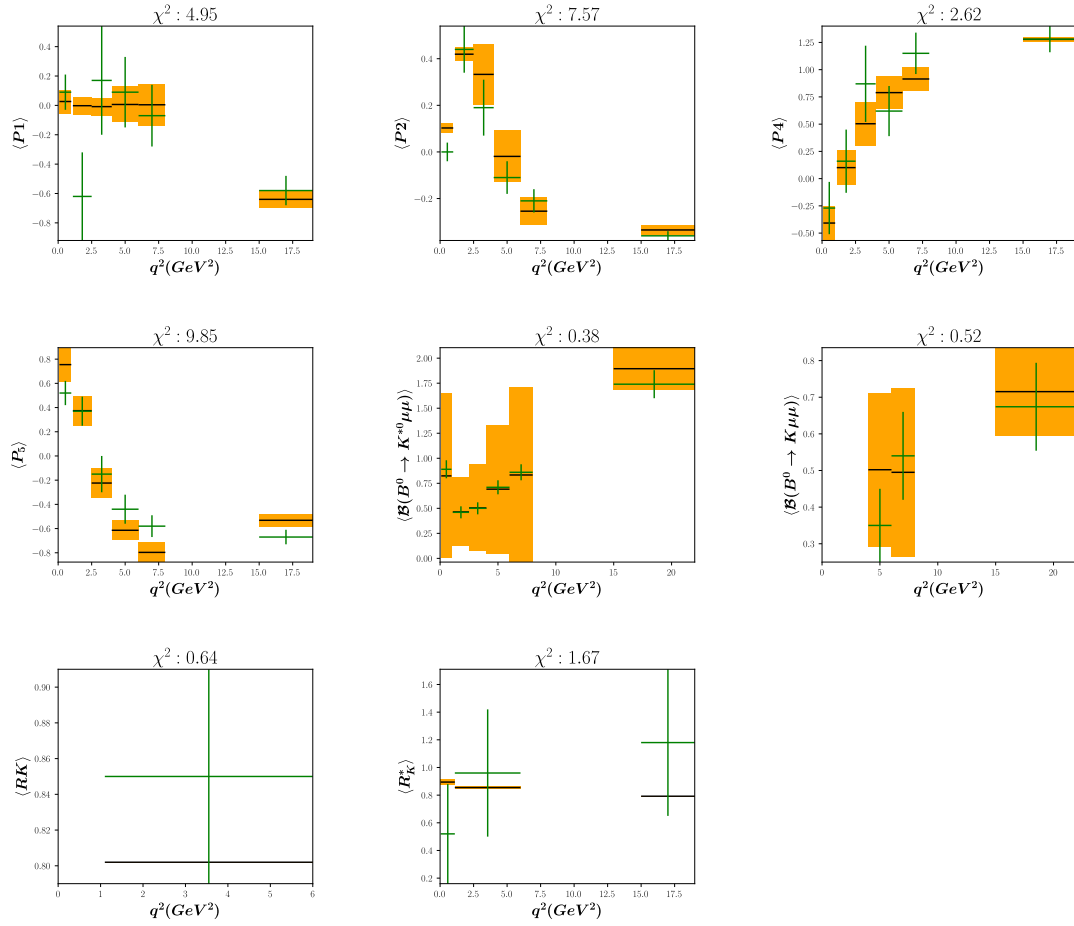
Model: C9: -0.89, C10: 0.11, χ^2 : 28.19

Figure 6: Best sample in training set. $\chi^2 = 28.19$ (orange) and experimental bin values (blue). This sample was generated by randomly selecting coefficients.

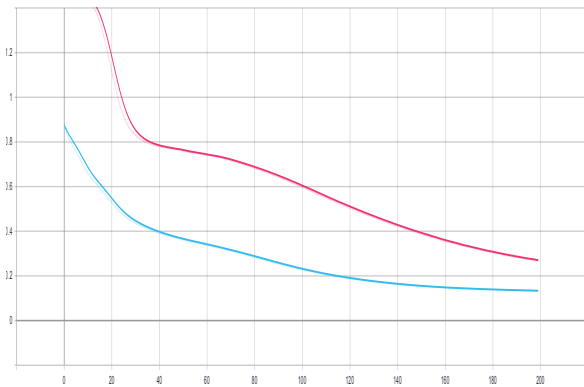


Figure 7: Neural Network experiment 2 (All). Mean square error training (blue) and validation (purple).

4.2. GAN

Our goal is to model the probability distribution of our training dataset using a GAN. Once we have a trained model we can sample from it. We want to test if the GAN is able to produce samples with χ^2 values lower than those in the training set.

4.2.1. GAN architecture

The GAN has been built using TensorFlow, and open-source software library developed by Google and Keras, an open-source neural-network library written in Python that allows you to build different Neural Network architectures in terms of layers. Training GAN models is hard, model parameters might oscillate, destabilize and never converge, the generator might collapse producing limited varieties of samples, the discriminator can get so successful that the generator gradient vanishes and learns nothing, an unbalance between the generator and discriminator produces overfit-

ting and last but not least GAN are highly sensitive to hyperparameter selections. In this work we have selected the following architecture:

Generator:

	Num. neurons	Activation
Input layer	60	RELU
Dense layer	500	RELU
Dense layer	500	RELU
Dense layer	784	RELU
Output layer	Depends on experiment	RELU

Discriminator:

	Num. neurons	Activation
Input layer	Depends on experiment	RELU
Dense layer	500	RELU
Dense layer	500	RELU
Dropout layer	Rate: 0.5	RELU
Output layer	1	Sigmoid

4.2.2. Experiment 1

In this experiment our training set consists of 2500 40-dimensional vectors generated by randomly sampling C_9 and C_{10} pairs in the ranges $[-2, 0]$, $[-1, 1]$. The χ^2 value is also computed and added to the final vector:

$$\mathbf{x} : \begin{bmatrix} Obs_n & C_9 & C_{10} & \chi^2 \end{bmatrix}$$

In figure 8 we plotted the losses during training of both the discriminator and the generator. We would expect in the end the generator's loss to be lower than the discriminator's but for the generator to be able to learn the discriminator should always be a bit ahead so there is gradient information to guide the generator. In this experiment we see that the generator's loss is a bit higher but the samples generated are still good and similar to the ones in the training set. By visually checking the generated samples we see that the generator is producing better sample with each iteration. It could happen that both are good at their task and even when the generator is generating good samples the discriminator can be so powerful that could still identify them as artificial. The loss in GAN measures how well we are doing compared with our opponent. Often, the generator cost increases but the image quality is actually improving. The discriminator's accuracy should be around 0.5. This would mean the discriminator fails to differentiate between real and generated samples. But for the same reason explained above we would need a visual check to test the quality of the generated samples. We sampled 10000 vectors from the GAN. The minimum χ^2 was 30.49 with $C_9 = -0.84$ and $C_{10} = 0.1$ a bit lower than the lowest in the training set, 30.67. In figure 9 we plotted the observables generated by this C_9, C_{10} pair.

4.2.3. Experiment 2.

In this experiment we add the q^2 -dependent coefficients a_0, a_1, a_2 . Each observable, $P_1, P_2, P_4, P_5, BrK^0, BrK^{0*}$,

$C_9 = 0.38, C_{10} = 5.9$ $\chi^2 = 581.90$	a_0	a_1	a_2
P_1	0.02	0.0008	0.004
P_2	0.01	-0.03	0.001
P_4	0.006	-0.001	-0.003
P_5	0.007	0.003	-0.0002
BrK^{0*}	-0.0002	-0.005	-0.001
BrK^0	-0.001	-0.004	0.005
R_K	0.0003	-0.001	-0.002
R_{K^*}	0.01	0.002	-0.00001

Table 1
GAN experiment 2 results.

$C_9 = 0.38, C_{10} = 5.9$ $\chi^2 = 581.90$	a_0	a_1	a_2
P_1	0.02	0.0008	0.004
P_2	0.01	-0.03	0.001
P_4	0.006	-0.001	-0.003
P_5	0.007	0.003	-0.0002
BrK^{0*}	-0.0002	-0.005	-0.001
BrK^0	-0.001	-0.004	0.005
R_K	0.0003	-0.001	-0.002
R_{K^*}	0.01	0.002	-0.00001

Table 2
GAN experiment 3 results.

R_K, R_{K^*} , has each own different set of coefficients:

$$\mathbf{x} : \begin{bmatrix} Obs_n & C_9 & C_{10} & a_{ij} \end{bmatrix}$$

where $i = 0, 1, 2$ and $k, j = 1, \dots, 8$ and $a_{ij} \neq a_{ik}$.

4.2.4. Experiment 3.

This experiment is a repetition of experiment 2 but we add a constraint to the q^2 -dependent coefficients, namely, all the observables share the same coefficients a_0, a_1, a_2 .

$$\mathbf{x} : \begin{bmatrix} Obs_n & C_9 & C_{10} & a_{ij} \end{bmatrix}$$

where $n = 1, \dots, 37$, $i = 0, 1, 2$ and $k, j = 1, \dots, 8$ and $a_{ij} = a_{ik}$

The sample with the lowest χ^2 value in the training set was found with the following coefficients:

$C_9 = -1.33, C_{10} = -0.44$ $\chi^2 = 23.33$	a_0	a_1	a_2
P_1	-0.093	0.001	0
P_2	-0.093	0.001	0
P_4	-0.093	0.001	0
P_5	-0.093	0.001	0
BrK^{0*}	-0.093	0.001	0
BrK^0	-0.093	0.001	0
R_K	-0.093	0.001	0
R_{K^*}	-0.093	0.001	0

Figure 10 shows this sample.

4.2.5. Experiment 4.

In this experiment we used the same training data as in experiment 2 but reshaped each 64-dimensional \mathbf{x} vector into

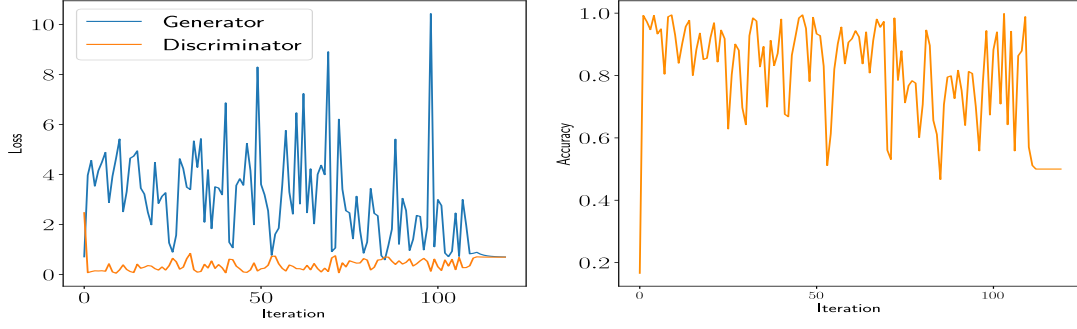


Figure 8: GAN experiment 1. Left : Discriminator vs Generators loss during training. Right : Discriminator accuracy.

an 8×8 image. A GAN was trained on this data set but this time both the discriminator and generator were convolutional neural networks (CNNs) instead of plain neural networks. Convolutional layers are the major building blocks used in CNNs. These layers allows CNNs to automatically learn a large number of filters which can be thought of as image features which best help to classify the image. In this experiment the generator consisted of one dense layer with 64 neurons, three convolutional layers of 128, 64 and 1 filters each. Each convolutional layer except the last one were followed by a RELU activation layer. The generator consisted of just two convolutional layers with 8 and 64 filters each. In this case each convolutional layer was followed by a LeakyRELU activation layer and one Dropout layer with the rate parameter set to 0.5. Finally an output layer with sigmoid activation function. In figure 11 we plotted some randomly picked images from the training set and in figure 12 some randomly generated images from the model's generator after training. The lower part of the images correspond to the q^2 -dependent coefficients. After training we sampled 100000 points from the model. All the points generated had very high χ^2 values, this could be due to the high sensitivity of χ^2 to small errors in the generation of the q^2 -dependent coefficients. Although results are not good, improving the quality of the generated images or samples by experimenting with other GAN architectures or performing a more thorough search in hyper-parameter space could lead to obtaining samples with lower χ^2 values along the q^2 -dependent coefficients used to generate them.

5. Conclusions

In this work we attempted some machine learning techniques to finding better global model descriptions to current high energy physics experimental data. We focused our attention on Neural Networks and Generative Adversarial Networks. In this work we included in our NP models the Standard Model hadronic uncertainties emerging from form factor contributions. Neural networks predicted values very similar to the ones obtained by χ^2 minimization. When taking into account the form factors contributions the network

$C_9 = 0.38, C_{10} = 5.9$ $\chi^2 = 581.90$	a_0	a_1	a_2
P_1	0.02	0.0008	0.004
P_2	0.01	-0.03	0.001
P_4	0.006	-0.001	-0.003
P_5	0.007	0.003	-0.0002
BrK^{0*}	-0.0002	-0.005	-0.001
BrK^0	-0.001	-0.004	0.005
R_K	0.0003	-0.001	-0.002
R_{K^*}	0.01	0.002	-0.00001

Table 3
GAN experiment 4 results.

predicted coefficients which did not improve the lowest χ^2 value in the training set although new C_9 and C_{10} coefficients were predicted. By using GAN networks we hoped that modeling the statistics of our training data, observable values and coefficients, would allow us to sample coefficients which would improve the ones already present in the training set. Although we found an improvement in χ^2 taking into account the form factor contributions while generating the training set by brute force, the GAN was not able to output any improved sample. This could be due to either a lack of statistics or a not optimal network architecture. Future work could explore more thoroughly both, networks with different complexities and hyper-parameter space. We did not take into account in this work correlations present in the data which could help in guiding the networks to find better results.

References

- [1] Algueró, M., Capdevila, B., Crivellin, A., Descotes-Genon, S., Masjuan, P., Matias, J., Virto, J., 2019a. Emerging patterns of new physics with and without lepton flavour universal contributions. The European Physical Journal C 79. URL: <http://dx.doi.org/10.1140/epjc/s10052-019-7216-3>, doi:10.1140/epjc/s10052-019-7216-3.
- [2] Algueró, M., Capdevila, B., Descotes-Genon, S., Masjuan, P., Matias, J., 2019b. Are we overlooking lepton-flavor-universal new physics in bs ? Physical Review D 99. URL: <http://dx.doi.org/10.1103/PhysRevD.99.075017>, doi:10.1103/physrevd.99.075017.
- [3] Capdevila, B., Crivellin, A., Descotes-Genon, S., Matias, J., Virto, J., 2017. Patterns of new physics in $b \rightarrow s\ell^+\ell^-$ transitions in the light of

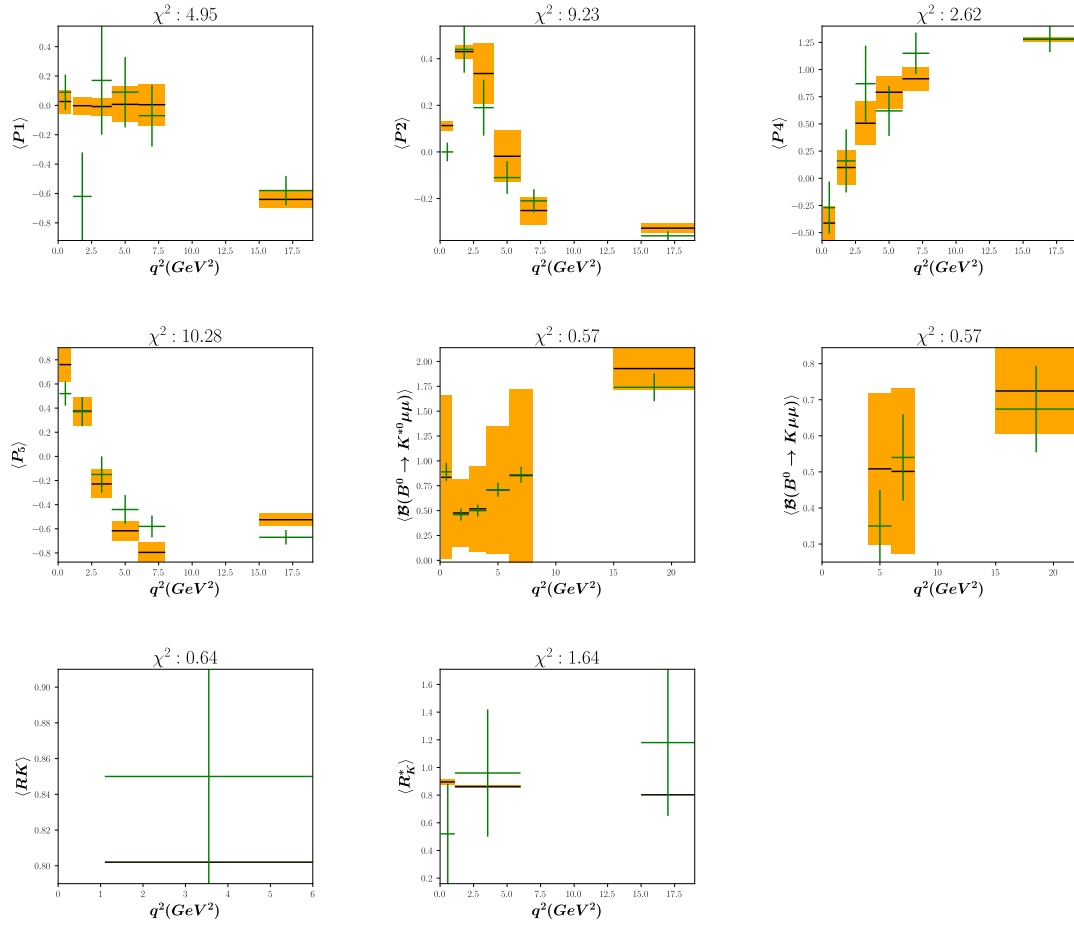
Model: C9: -0.85, C10: 0.1, $\chi^2 : 30.497$ 

Figure 9: GAN experiment 1. Generated sample with minimum $\chi^2 = 30.49$ (orange) and experimental bin values (green crosses).

recent data. arXiv:1704.05340.

- [4] Descotes-Genon, S., Hofer, L., Matias, J., Virto, J., 2016. Global analysis of b s anomalies. Journal of High Energy Physics 2016. URL: [http://dx.doi.org/10.1007/JHEP06\(2016\)092](http://dx.doi.org/10.1007/JHEP06(2016)092), doi:10.1007/jhep06(2016)092.
- [5] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial networks. arXiv:1406.2661.

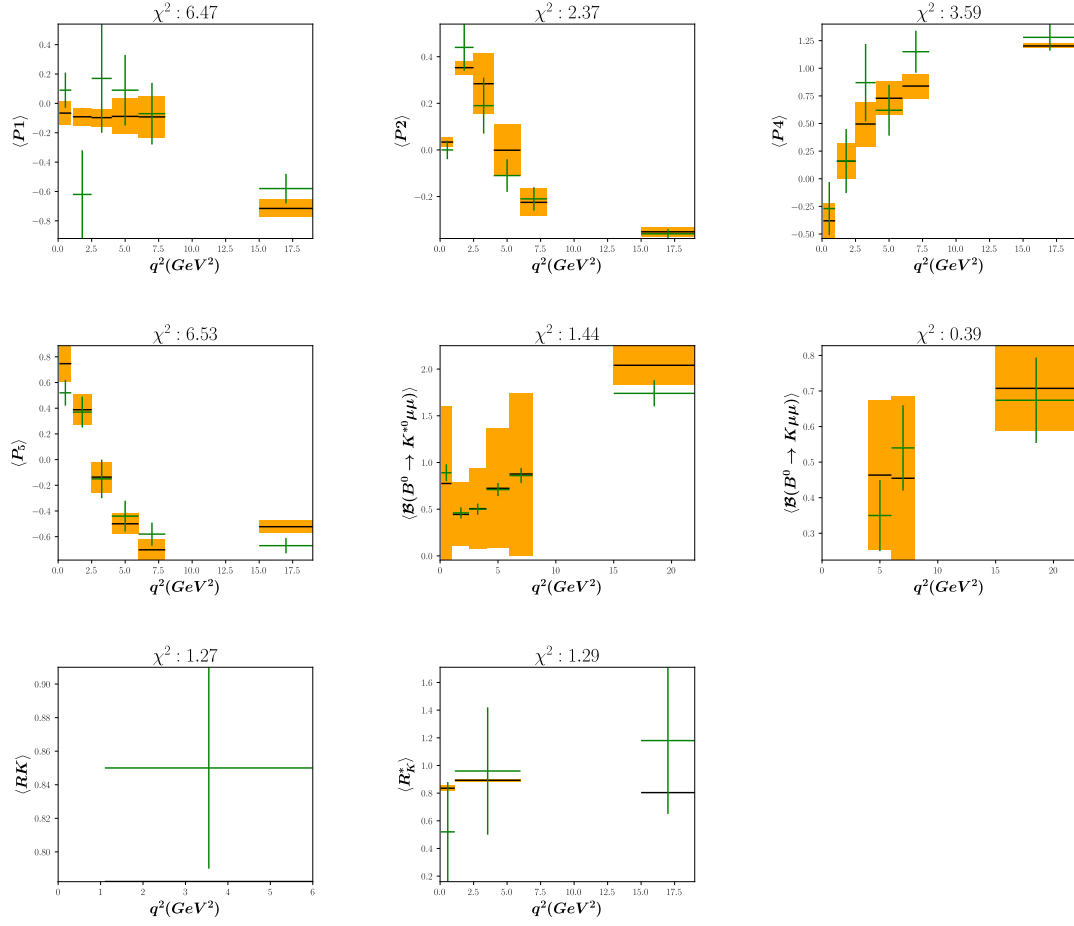
Model: C9: -1.33, C10: -0.44, χ^2 : 23.339

Figure 10: GAN experiment 3. Sample with lowest χ^2 value in the training set. q^2 -dependent coefficients are shared between observables.

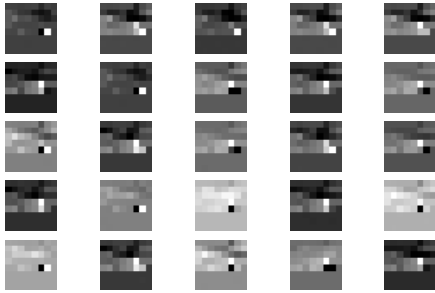


Figure 11: GAN experiment 4. Images randomly picked from the training set.

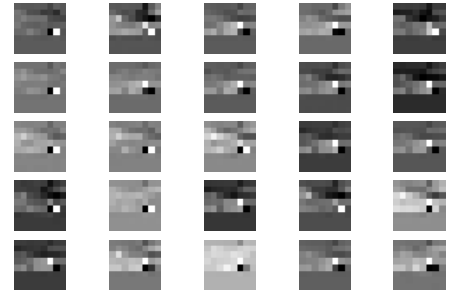


Figure 12: GAN experiment 4. Images generated by GAN after learning.

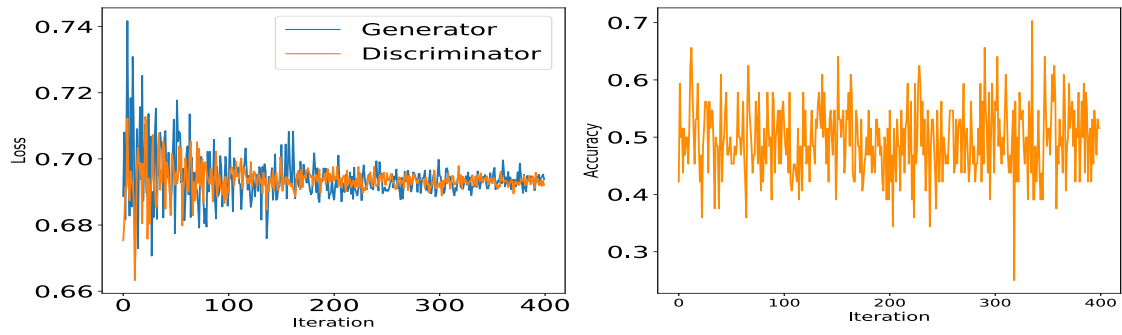


Figure 13: GAN experiment 4. Left : Discriminator vs Generators loss during training. **Right :** Discriminator accuracy.