EPJ Web of Conferences **214**, 06013 (2019) https://doi.org/10.1051/epjconf/201921406013
CHEP 2018

The use of adversaries for optimal neural network training

Anton Hawthorne-Gonzalvez[1] and *Martin* Sevior[1,*]

[1]School of Physics, The University of Melbourne, Parkville, Victoria, 3010, Australia

Abstract. B-decay data from the Belle experiment at the KEKB collider have a substantial background from $e^+e^- \to q\bar{q}$ events. To suppress this we employ deep neural network algorithms. These provide improved signal from background discrimination. However, the deep neural network develops a substantial correlation with the ΔE kinematic variable used to distinguish signal from background in the final fit due to its relationship with input variables. The effect of this correlation is reduced by deploying an adversarial neural network. Overall the adversarial deep neural network performs better than a Boosted Decision Tree algorithimn and a commercial package, NeuroBayes, which employs a neural net with a single hidden layer.

1 Physics Background

These proceedings are an abridged version of the paper published by Hawthorne-Gonzalvez and Sevior [1].

Rare decays of B-mesons, whose properties can be precisely predicted by the SM, are the subject of much experimental activity. Rare decays typically have branching ratios of the order of 10^{-5} or smaller. Therefore a large sample of $B\bar{B}$ pairs is required to make statistically significant measurements. The Belle experiment employs the KEKB accelerator to collide e^+e^- particles at the centre of mass energy of $\sqrt{s} = 10.58$ GeV, corresponding to the $\Upsilon(4S)$ resonance. The $\Upsilon(4S)$ subsequently decays primarily (more than 96% of the time) to $B\bar{B}$ pairs. In addition, the KEKB collider also initiates the $e^+e^- \to q\bar{q}$ reaction where $q \in \{u, d, s, c\}$ (continuum background). This occurs at a rate 3 times greater than $\Upsilon(4S)$ production.

The $B^0 \to K_S\pi^0$ decay, proceeds via the $b \to u$ transition and through 2nd-order electroweak loop diagrams [2].

CP-symmetry is the expectation that applying C (the charge operator - inverting all of the internal quantum numbers) and P (the parity operator - reversing all spacial coordinates) would have no effect on the physics of a process.

Both B^0 and $\bar{B}^0$ can decay to the *CP*-eigenstate $K_S\pi^0$. Direct *CP*-violation (DCPV), is when these decay rates are not equal, and in a data sample with equal numbers of B^0 and $\bar{B}^0$ pairs, the DCPV is quantified with $\mathcal{A}_{CP}$ which is defined as:

$$\mathcal{A}_{CP}(K_S\pi^0) = \frac{N(\bar{B}^0 \to K_S\pi^0) - N(B^0 \to K_S\pi^0)}{N(\bar{B}^0 \to K_S\pi^0) + N(B^0 \to K_S\pi^0)} \quad (1)$$

*e-mail: martines@unimelb.edu.au

Where N is the measured number of events for a given decay. The similar amplitudes of the contributing Feynman diagrams mean that the process can exhibit relatively large DCPV. The most recent Belle measurement is $\mathcal{A}_{CP}(K^0\pi^0) = +0.14 \pm 0.13(stat) \pm (0.06)(sys)$ [3], where the majority of the statistical uncertainty is due the large background to signal ratio. By combining measurements of all $B \to K\pi$ charge states, Beak et al. [4] predict that $\mathcal{A}_{CP}(K^0\pi^0) = -0.15 \pm 0.03$ and that a 5-σ deviation from this value would demonstrate that contributions from New Physics make an unambiguous effect on this mode. Accordingly, the most precise measurements for $\mathcal{A}_{CP}(K^0\pi^0)$ are vital. This is turn requires minimising the continuum background. We investigate advanced neural network (NN) machine learning algorithms to do this.

2 Analysis of $B \to K_s\pi^0$ decays

We begin by performing Monte-Carlo (MC) simulations of the $B \to K_S\pi^0$ process. We employ the EvtGen[5] package in which the $\Upsilon(4S)$ decays to a $B^0\bar{B}^0$ pair. From here, one, B^0_{sig}, will decay to $K_S\pi^0$ and the other, B^0_{tag}, will decay generically. Particles from this process are propagated through the detector with Geant3[6]. Three data-sets of one-million events each are generated, for training, validation and testing of the neural networks. Continuum MC data is generated by the Belle collaboration at six times the expected yield (six streams) over the entire experiment, where two streams are used for NN training, one for validation, and three for testing.

Real off-resonance (at a center of mass (COM) energy of 10.52 GeV) data is available at 10.35% of the expected continuum yield at the $\Upsilon(4S)$ resonance, and is used to finally validate NN performance.

Two kinematic variables are employed in a maximum likelihood fit to discriminate between $B^0 \to K^0\pi^0$ events and backgrounds. These are ΔE and M_{bc}^{corr}. ΔE is the difference between the reconstructed B meson energy (E_B) and half of the e^+e^- COM energy, given by:

$$\Delta E = E_B - E_{beam} \tag{2}$$

Where E_{beam} is the beam energy in COM frame. ΔE peaks at 0.0 GeV for signal and has a continuous distribution for continuum background. Although ΔE peaks at 0.0 GeV for signal, it has a significant asymmetretic distribution due to energy leakage from the electromagnetic calorimeter (ECL), employed to measure the energy of the photons from the decay of π^0s. M_{bc}^{corr} is the beam constrained mass corrected for calorimeter energy leakage [1] of the event. This peaks at the B-meson mass for signal events.

We place the selection criteria that 5.265 GeVc^{-2} < M_{bc}^{corr} < 5.3 GeVc^{-2} and −0.4 GeV < ΔE < 0.3 GeV.

There is far more continuum than signal, (around 54 times as much) and in order to reduce this, nineteen additional kinematic variables are calculated, (taking advantage of the differing decay topologies between signal and continuum) and employed as input into a NN [1]. The NN provides a classification, (a variable upon which a selection criteria can be placed) based on these inputs.

3 Deep Neural Networks with TensorFlow

In order to maximise the continuum suppression, we build a NN architecture from the ground up using TensorFlow [7]. We implement a deep NN and employ state of the art algorithms which together, have the potential to provide better classification than is achieved using single-layer NNs.

The training data-set consists of 125,000 (correctly reconstructed) signal and 125,000 continuum events. First the data for each of the nineteen kinematic variables is pre-processed by implementing equal frequency binning. Here each variable is transformed so that the total 250,000 events, comprising signal and background, are evenly distributed among 500 equally spaced bins over the output range -1 to +1. During training, we employ the ADAM algorithm [8] to minimise the cross-entropy ($\mathcal{L}_{class}$) as defined in equation 3.

$$\mathcal{L}_{class}(\vec{x}, \hat{y}) = -\hat{y} \cdot log\left(y\left(\vec{x}\right)\right) - (1 - \hat{y}) \cdot log\left(1 - y\left(\vec{x}\right)\right) \tag{3}$$

Where $y(\vec{x})$ is the NN output given the vector of inputs (the kinematic variables) $\vec{x}$. $\hat{y}$ is the known (target) value (1 or 0 for signal or continuum respectively).

The performance of the NN is measured by calculating the cross-entropy on the validation data-sets. The architecture of the NN (i.e. the number of hidden layers, nodes per layer and activation functions) and the training algorithm parameters (batch size, learning rate, training steps etc.) are the hyper-parameters that must be specified. The hyper-parameter space is immense so finding the best configurations requires an efficient algorithm. The best hyper-parameter configuration of the TensorFlow network was found using HyperBand [9], which narrows down the best configuration from a large random sample of hyper-parameter configurations. It achieves this by training a few configurations over the full training run, many configurations for a small fraction of the full training run, and a range in between. This combines the need to check many configurations with the need to better evaluate each given configuration. Once the best configuration is found, the NN is trained with this set of hyper-parameters. The NN is then applied to the testing data-set and employed for further physics analysis. The hyper-parameter configuration for the best performing network is given in Ref. [1].

The global performance of the optimised TensorFlow NN (TF1) is characterized by the Area Under the Curve, (AUC), of the Receiver-Operator Curve [1]. An AUC = 0.5 implies no effect while AUC = 1.0 implies perfect classification. We find an AUC = 0.9501 for TF1 from processing the testing data-sets.

3.1 Analysis of the TensorFlow Neural-Network Performance

To evaluate TF1, we compare it to the performance of the NeuroBayes neural network package (NB) [10] as well as the Boosted Decision Tree (BDT) algorithim as implmented by the TMVA - Toolkit for MultiVariate Analysis package [11]. Both NB and the BDT method of TMVA are widely used by the Belle Collaboration for event classification. The internal architecture of NB consists of one hidden layer where the number of nodes was set to the default value of 21.

The input data is pre-processed by NB to transform each variable into a Gaussian distribution. The batch-size was 100, and NB was trained over 150 epochs using the Broyden–Fletcher–Goldfarb–Shanno algorithm (see [12] for more information). Regularisation is employed using the ‘Bayesian regularisation procedure’ (see [13] for details on the NeuroBayes algorithm). During training NeuroBayes employs pruning and removal of the least important weights to prevent over-training.

NB was trained on the same data-sets as TF1. The set up was not tweaked to improve performance, therefore this network was not applied to the validation data-sets, and instead applied directly to the testing data-sets. The performance of the TMVA BDT was measured using the same testing data-sets as NB. The AUC of the algorithim is 0.9267.

For TF1, a value of NN_{cut} (a selection criteria for which $NN > NN_{cut}$) chosen to keep 13.00% of continuum, leaves 88.11% of signal (in contrast with 79.01% from NB). Alternatively a value of NN_{cut} chosen to keep 70.20% of signal using TF1, leaves 3.35% of continuum remaining (in contrast with 7.65% from NB). These results show that, depending on the NN_{cut} choice, the continuum background for a given signal efficiency could be reduced by over a factor of 2 by employing TF1 rather than NB.

3.2 ΔE - Classifier Correlations

Investigation into the ΔE distribution at different NN_{cut} values for TF1 shows unexpected results. When looking at the continuum ΔE distributions for different *NN* slices, the distribution is sculpted to be more signal-like as *NN* increases. On the flip side, for a low *NN* the distribution shows the reverse, a trough where signal peaks. Figure 1 shows the continuum ΔE distributions at different *NN* ranges. The off-resonance data also shows the same effect. Similarly for the signal distribution, where the ΔE distribution becomes less signal-like as *NN* decreases. This effect was not evident when using NB. Nor is it evident when using the BDT algorithim. This correlation can also be observed by looking at the continuum rejec-

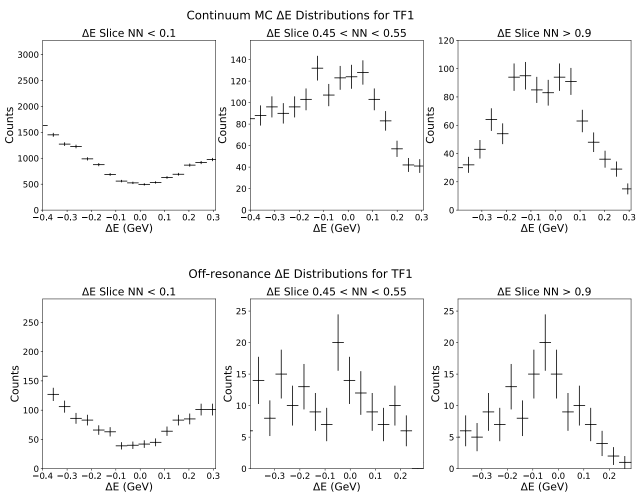

Figure 1: Showing the scultping of the ΔE distributions at different *NN* (TF1) slices for $e^+e^- \to q\bar{q}$ events. The effect is seen in both continuum MC (top row) and off-resonance real data (bottom row).

tion percentage in an enhanced signal region of $-0.1 < \Delta E < 0.1$ (“tight region"). A NN_{cut} selection is imposed on both NN outputs to keep the signal efficiency the same; 92.5% over the full ΔE range. The continuum rejection rate for the TensorFlow NN is 80.09% over the full ΔE range but only 63.4% in the tighter region. In comparison, the NeuroBayes NN, has continuum rejection rates of 66.7% and 64.08% for the full and tighter ΔE regions.

The TensorFlow NN learned that there is a relation between the ΔE value, and whether an event is signal or continuum. This is not observed by either the NB or the BDT algorithimns. The behaviour of TF1 is due to correlations between ΔE and some of the kinematic variables on which the NN is trained. These are listed in Ref. [1].

4 Adversarial Neural Networks With TensorFlow

The correlation with ΔE significantly reduces the effectiveness of the continuum suppression of TF1 in the signal region when performing a unbinned maximum likelihood fit to the data. While removing the correlated variables would reduce the effect, there is discriminating power in the variables which we would lose if we adopt this approach. An alternative was to investigate employing an adversarial neural network (ANN).

An ANN can be used to reduce the correlations between the output of a NN (referred to as the classifying neural network) and other parameters associated with the event. This method is used to reduce the correlation between NN and ΔE, although in principle this could also be used for any one of, or multiple parameters that have correlations with NN. The method laid out here closely follows that in [14].

The adversarial network models the ΔE distribution by taking NN as input, and predicting the value of ΔE that a given event will have. The adversarial network used in this study has one input (NN), two hidden layers with 20 nodes each, and 15 outputs. The result is to model ΔE with five Gaussians (indexed by i), with 3 outputs for each, corresponding to the means ($\mu_i(NN)$), widths ($\sigma_i(NN)$), and fractional weighting of that Gaussian ($f_i(NN)$). The fractions are not normalised so they are first passed through a softmax function (giving $f_i'(NN)$, scaled to sum to one). The adversary loss function (for a single event) is given by:

$$\mathcal{L}_{adv}(NN, \Delta E) = -log\left(\sum_{i=1}^{5} \frac{f_i'(NN)}{\sqrt{2\pi\sigma_{i^2}(NN)}} \exp\left\{\frac{-(\mu_i(NN) - \Delta E)^2}{2\sigma_i^2(NN)}\right\}\right) \tag{4}$$

Training the ANN to minimise this loss function allows it to predict the ΔE distribution from the input NN distribution if the two parameters are correlated and the ΔE distribution can be modelled with five Gaussians. We want to penalise the classifying neural network if NN is correlated to ΔE, so the classifier is further trained to minimise:

$$\mathcal{L}_{tot} = \mathcal{L}_{class} - \lambda_{adv}\mathcal{L}_{adv} \tag{5}$$

Where $\mathcal{L}_{class}$ is our loss function for the classifier network (the cross entropy defined in 3), and λ_{adv} is a constant chosen to specify how much to penalise $\Delta E - NN$ correlations. Training to this new loss function has the desired impact of reducing the correlations at the cost of the continuum suppression. A λ_{adv} of zero would result in a classifier identical to TF1, whereas a larger λ_{adv} results in reduced $NN - \Delta E$ correlations but worse classifying power over the whole range of ΔE. The configuration of the networks is shown in Ref. [1].

There are now additional hyper-parameters associated with the architecture of the adversary network. These include the number of outputs (related to the number of Gaussians with which to model the ΔE distribution), the number of hidden layers and nodes per hidden layer. The additional hyper-parameters associated with training the adversary network are its batch-size, training steps and learning rate.

The classifier neural network has the same architecture, and is initialised to the optimal weight values from the previous training in 3.1. For every classifier training step, the adversary network is first trained for 100 steps using the ADAM optimiser with a batch size of 125 and a learning rate of 0.01. The learning rate for training the classifier is reduced to 10^{-6} and the training is run for 4 epochs.

The hyper-parameters for the classifier, the ANN as well as a complete description of the method employed to train the classifier and ANN are provided in Ref. [1].

4.1 Analysis of the TensorFlow With Adversary Neural-Network Performance

An investigation was performed by repeating the procedure for λ_{adv} values of 0.25, 0.5, 0.75, 1.0, 1.5, 2.0, 3.0, 4.0 and 5.0. The testing data-sets are processed by each of the NNs (TF2) trained with the different λ_{adv} values. The continuum rejection rates in the full ($-0.4 < \Delta E <$

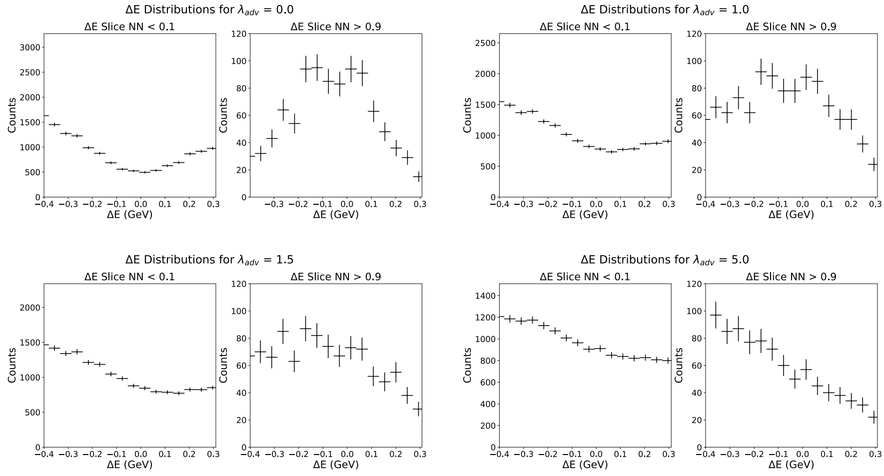

Figure 2: Showing the continuum ΔE slices at $NN < 0.1$ and $NN > 0.9$ for λ_{adv} between 0.0 and 5.0 as shown.

0.3) and tight ($-0.1 < \Delta E < 0.1$) ranges, for each of these NNs along with the NeuroBayes NN, BDT and original TensorFlow NN without adversary, are shown in Table 1. As before, these rejection rates are calculated after imposing selection criteria on *NN* such that 92.5% of signal remains in the full ΔE range.

We find that the $\Delta E - NN$ sculpting steadily decreases with increasing λ_{adv}. The $\Delta E - NN$ correlation decreases until $\lambda_{adv} = 1.0$, after this, the correlation becomes increasingly negative even as the scultping effect continues to diminish. The background rejection rate also becomes worse for $\lambda_{adv} > 1.0$. The continuum MC (testing data-set) ΔE distributions (at different *NN* slices) for each λ_{adv} are shown in Figure 2. We choose $\lambda_{adv} = 1.5$ as the point of comparison for the rest of the discussion as this is best compromise between minimizing the scuplting, the correlation with ΔE while retaining the best background rejection rates.

λ_{adv}	RR Full-ΔE Range	RR Tight-ΔE Range	Correlation with ΔE
N/A (TF1)	81.3%	63.2%	0.114
0.25 (TF2)	80.6%	65.1%	0.080
0.50 (TF2)	79.7%	66.6%	0.057
1.00 (TF2)	78.4%	67.7%	0.012
1.50 (TF2)	74.6%	67.4%	−0.024
3.0 (TF2)	64.3%	63.0%	−0.106
5.0 (TF2)	51.9%	53.4%	−0.175
(NB)	66.7%	64.08%	0.058
(NB, reduced)	63.0%	64.08%	−0.001
(BDT)	73.7%	67.2%	0.262
(BDT, reduced)	66.9%	66.5%	0.054

Table 1: The continuum MC rejection rates (RR) for the full and tight ΔE regions with a signal acceptance of 92.5%. Column 4 shows the correlation between ΔE and the discriminating variable. 'N/A (TF1)' refers to TensorFlow NN trained without adversary, '(TF2)' refers to the adversarialy trained NNs, '(NB)' refers to the NeuroBayes NN, '(NB, reduced)' is NeuroBayes without correlated variables. '(BDT)' is the Boosted Decision Tree algorithim. '(BDT, reduced)' is BDT without correlated variables.

5 Validation with off-resonance data

To fully evaluate the performance of the TF2, we process real off-resonance data to test the MC simulations.

The off-resonance rejection rates (for NN_{cut} keeping 92.5% signal) for the NeuroBayes NN, TF1 and TF2 with $\lambda_{adv} = 1.5$ are shown in table 2. As compared to the results in Table 1, the rejection rates are similar although poorer for off-resonance (to be expected as the NNs were trained entirely with MC simulations which employ parameterized theoretical model to simulate $e^+e^- \rightarrow q\bar{q}$ interactions) but are consistently worse regardless of which classifier was used. Moreover, the trends identified earlier for TF1 and TF2 versus NB are the same. TF1 provides the best rejection over the full ΔE range and TF2 provides the best performance in the tight ΔE range. We conclude that the NN's do significantly reduce the experimental continuum background at rates close to those predicted by the MC studies.

λ_{adv}	Off-Resonance RR Full-ΔE Range	Off-Resonance RR Tight-ΔE Range
N/A (TF1)	77.7%	58.6%
1.50 (TF2)	69.0%	60.5%
N/A (NB)	61.1%	59.8%
N/A (BDT)	68.7%	60.4%

Table 2: The off-resonance rejection rates (RR) for the full and tight ΔE regions. The signal acceptance was set to 92.5%

6 Conclusions

While the purpose-built deep NN developed with TensorFlow (TF1) has better continuum suppression than the commercial NeuroBayes (NB) package and the Boosted Descision Tree (BDT) algorithim, it achieved this at the cost of significant sculpting of the ΔE distribution which is used to distinguish signal from background. In fact in the most sensitive region to signal, -0.1 GeV < ΔE < 0.1 GeV, TF1 had a poorer performance than NB and BDT. Setting the classifier to accept 92.5% of the signal we find background rejection rates of 63.4% vs 64.8% and 67.2% for TF1, NB and BDT respectively. We then employed an adversarial NN (TF2) to counter-act this correlation. With $\lambda_{adv} = 1.5$, TF2 achieved improved performance compared to NB in both the most sensitive (67.4% vs 64.1%) and full (74.6% vs 66.7%) regions of ΔE. While the background rejection rate of TF2 with $\lambda_{adv} = 1.5$ was only slightly better than the BDT in the most sensitive region (67.4% vs 67.2%), the correlation with ΔE for TF2 was significantly better than the BDT (-0.024 vs 0.264). While some sculpting remains at $\lambda_{adv} = 1.5$, this ΔE distribution is still significantly different from that of the signal and the overall correlation between TF2 and ΔE is close to minimized at $\lambda_{adv} = 1.5$. Consequently the adversarial neural network allows us to employ discriminating variables correlated with ΔE to improve background

7 Acknowledgements

We gratefully acknowledge our colleagues within the Belle collaboration for all the work required to make this investigation possible and for their permission to employ Belle data and tools for this study. We would particularly like to thank Thomas Keck of the Karlsruher

Institut für Technologie for very helpful discussions about adversarial neural networks. We thank the KEKB group for the excellent operation of the accelerator; the KEK cryogenics group for the efficient operation of the solenoid; and the KEK computer group, the National Institute of Informatics, and the PNNL/EMSL computing group for valuable computing and SINET5 network support. We acknowledge financial support from the Australian Research Council grant DP180102629.

References

[1] A. Hawthorne-Gonzalvez, M. Sevior, Nuclear Int. and Methods in Physics Research, **A913**, 54 (2019), `1712.07790v2`

[2] W.H. S.W. Lin, Y. Unno et al., Nature **452**, 332 (2008)

[3] M. Fujikawa et al. (Belle), Phys. Rev. **D81**, 011101 (2010)

[4] S. Baek et al., Phys. Lett. **B678**, 97 (2009)

[5] D.J. Lange, Nucl. Instrum. Meth. **A462**, 152 (2001)

[6] R. Brun, L. Urban, F. Carminati, S. Giani, M. Maire, A. McPherson, F. Bruyant, G. Patrick, Tech. rep., CERN (1993), `http://cds.cern.ch/record/1073159/files/cer-002728534.pdf`

[7] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, *Tensorflow: Large-scale machine learning on heterogeneous distributed systems* (2016), `1603.04467`

[8] D.P. Kingma, J. Ba, *Adam: A method for stochastic optimization* (2014), `1412.6980`

[9] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, *Hyperband: A novel bandit-based approach to hyperparameter optimization* (2016), `1603.06560`

[10] M. Feindt, U. Kerzel, AIP Conference Proceedings 1504, 1013 (2012)Nucl. Instrum. Meth. **A559**, 190 (2006)

[11] J. Therhaag et al., AIP Conference Proceedings **1504**, 1013 (2012)

[12] R.H. Byrd, P. Lu, J. Nocedal, C. Zhu, SIAM Journal on Scientific Computing **16**, 1190 (1995)

[13] M. Feindt, *A Neural Bayesian Estimator for Conditional Probability Densities* (2004), `physics/0402093`

[14] G. Louppe, M. Kagan, K. Cranmer, *Learning to pivot with adversarial networks* (2016), `1611.01046`