Machine Learning Methods for Histogram Deconvolution in High Energy Physics

CERN (CMS) Summer Student Report

A.R. Wiederhold*
Supervisor: B. Isildak†

September 20, 2019

Keras sequential neural networks are developed to perform histogram deconvolution on Z–boson mass spectra generated by the MadGraph5_aMC@NLO event generator using Pythia8 and Delphes. Three ways of interpreting the problem with neural networks are presented, tested and then compared with each other and the popular unfolding method TUnfold. A bin classification method is identified as a robust deconvolution method, with results comparable to that of TUnfold.

I. Introduction

Particle detectors do not provide perfect measurements of the true values of the variables that they measure, this introduces smearing effects into particle data. In particular this smearing can be considered as a convolution on the true data to obtain the detected data [1]. A measured distribution can be related to the true distribution by the following equation:

$$N\left(y^{\mathrm{m}}\right) = \int_0^\infty n\left(y^{\mathrm{t}}\right) R\left(y^{\mathrm{m}},\ y^{\mathrm{t}}\right) dy^{\mathrm{t}} \qquad (1)$$

Where $N(y^{\mathrm{m}})$ is the number of events measured as y^{m}, $n(y^{\mathrm{t}})$ is the number of events with the true value y^{t} and $R(y^{\mathrm{m}}, y^{\mathrm{t}})$ is the probability of an event with the true value y^{t} being measured as y^{m}.

Another way to interpret smearing is to consider that when constructing a true histogram and a detected histogram, for each event, the true value and detected value of a measurement are placed into a bin in their respective histograms. If the histograms are constructed with identical binning regimes then one can say that the event has migrated by an amount equal to the difference between the true bin number and the detected bin number, this is the bin migration.

When making theoretical predictions of these measurements, the smearing effects of a particular detector are typically not taken into account, therefore in order to make an accurate comparison between experiment and theory, one must unfold/deconvolve the histogram of the measured values or fold/convolve their theoretically predicted histogram. A popular method to perform the unfolding of histograms is to use the ROOT [2] class TUnfold [3]; Alexander Glazov has demonstrated that machine learning methods can also be used to deconvolve histograms [4].

In that study toy data was used to demonstrate the applicability of neural networks to histogram deconvolution, this paper serves to extend that work by deconvolving simulated particle physics data, in particular Z–boson mass reconstruction data. Another example of toy data is used to verify that this method may be valid for different shapes of distributions. In [4] deconvolution is discussed as a neural network

*Email: a.wiederhold@warwick.ac.uk
†Email: bora.isildak@cern.ch

classification problem in which one trains their model to take input data and predict which 1-dimensional histogram bin that data belongs to. In the case of a Z–boson mass reconstruction, one could use the invariant mass of the Z reconstructed from the detected 4-momenta of its daughter particles as the input data; additional information from each particle event can also be used which has already been explored in the case of 1 additional variable, this study takes advantage of this to include > 6 additional variables.

This paper presents three ways to interpret this deconvolution problem using neural networks; regression, bin classification and migration classification.

II. Methodology

A. Data Production and Mass Reconstruction

MadGraph5_aMC@NLO [5] was used to generate one million $pp \to z$, $z \to e^+e^-$ or $\mu^+\mu^-$ events. Pythia8 [6] was used to simulate the showering and Delphes [7] simulated the response of a particle detector. The data was output in the form of a '.root' file which contained a tree with branches of data relating to the 'generated' particles of the event (these are the 'true' values), detected muons, detected electrons, scalar HT and missing energies. For each event, the total number of detected leptons and the kinematic variables (p^T, η, ϕ), isolation variable (IsoVar), electric charge (Q) and PDGID of each lepton in the daughter pair of the generated Z were extracted. The kinematic variables were used to construct 4-momenta for each lepton and the invariant mass of the 4-momenta is given as the reconstructed Z–mass, m_Z. Missing energies (MET, MET_η and MET_ϕ) and scalar HT were also extracted. Three extra variables were created:

$$p^T_{\text{product}} = p^T_1 p^T_2 \tag{2}$$

$$\Delta\eta = |\eta_1 - \eta_2| \tag{3}$$

$$\Delta\phi = |\phi_1 - \phi_2| \tag{4}$$

This resulted in one truth variable (the generated m_Z) and 21 input variables. To identify the daughter pair of the Z in each event all possible pairings of the detected leptons that could be produced by a Z decay at leading order were found and the pair whose invariant mass was closest to that of the Z ($m_{\text{Z, PDG}}$), as reported by the PDG [8] to be equal to 91.1876 GeV, was chosen as the Z daughter pair. As in [9] any events in which the detected values did not satisfy $\text{IsoVar}_i < 0.1$ for $i \in \{1, 2\}$ were rejected. $\sim 200,000$ events with suitable lepton daughter pairs were found during the Z mass reconstruction.

Feature	Score (3 SF)
m_Z	30700
p^T_2	300
p^T_{product}	181
p^T_1	126
Scalar HT	60.7
$\Delta\eta$	2.33
MET	1.73
η_2	1.39
η_1	1.26
MET_ϕ	1.26
MET_η	1.17
IsoVar_1	1.11
$\Delta\phi$	1.05
ϕ_2	0.966
IsoVar_2	0.951
PDGID_1, PDGID_2	0.813
Q_1, Q_2	0.795
ϕ_1	0.469
# of Detected Leptons	0.237

Table 1: Scikit–learn feature scores.

Before fitting the neural networks the feature space[1] was examined to determine which features should be used. The Pearson correlation between all of the features was calculated to find any features that may be redundant, for example it is unlikely that including both the PDGID and charge of each lepton will be beneficial since the PDGID depends explicitly on the charge and they are therefore perfectly correlated. Scikit–learn [10] was used to evaluate feature importance and output the score associated with each feature, the results of this are shown in table 1. Using the feature scores and correlations, 7 features were chosen which had high scores and low correlations with other selected features, these were the 7 typically used in training. It was found that

[1] The dimension of which is equal to the total number of input variables.

using all 7 could lead to overfitting so only 3 or 5 of these features were used for most training. Before training, functions from Scikit–learn were used to scale the input variables in order to prevent one dominating all others by having much larger values and variance. For example the mass values[2] are ~ 90 whereas the isolation variables are ~ 1, so without scaling, the training would have likely been dominated by the masses and the isolation variables would have been mostly ignored.

In the case of migration classification one can limit the maximum migration in their data by removing events with migrations greater than the desired amount. This has the effect of reducing the total number of classes the model has to choose from in its predictions and could therefore result in a higher overall accuracy since the model can learn about each class better than if there were many more. One should note that this means that when using the model to predict migrations for real detected data it cannot be guaranteed (in fact it is unlikely) that the data will only have events with the same migration limits as chosen prior to training, this may result in poor performance in a real scenario in which the detected data can't be cut. Data which has had events removed for this reason will be referred to as the 'cut data' and data in which some of these events have been added back in will be referred to as 'uncut data'.

The 'true' values for the toy data were produced by randomly generating $1,000,000$ values, x, in the range $[1500, 3000]$ following the shape $a(1-x)^b/x^c$, where $a, b, c > 0$ are inspired by [11]. These values are then smeared using a gaussian distribution to obtain the 'detected' values. The mean and standard deviation of the gaussian used to smear each event are, respectively, x and σx where $\sigma \in (0, 1]$ is the relative smearing.

B. Models, Training and Application

Keras [12] with a TensorFlow [13] backend[3] was used to develop the neural networks for each method, summaries of which are shown in tables 2, 3 and 4. The loss in classification models was calculated using categorical cross–entropy with the 'Adadelta' optimiser whereas in the regression model it was calculated using mean–squared error and the 'Adam' optimiser. When training the models a maximum of 200 epochs was allowed, with an early stopping callback to stop training the model when it had stopped improving by a given amount to prevent overfitting. Another callback reduced the learning rate when the loss plateaued to help the model converge on minima.

To construct a deconvolved histogram from the bin classifier, for each event every bin was filled with a weight equal to the probability assigned to it by the model. Similarly, for the migration classifier the probabilities for each class were used to weight the bin fillings, which bin should be filled for each probability was determined by adding the corresponding migration to the bin number corresponding to the reconstructed mass for the event. The regression histogram was simply produced by filling it with the masses predicted by the model.

It can be shown that using the probability distributions produced by the classification models to fill the histograms results in an operation on the data that seems to mimic a convolution in such a way that it restores the true distribution from the detected distribution. The number of events placed into the j-th bin by this method is given by the following equation:

$$N_j = \sum_{k=1}^{n_{\text{events}}} \sum_{i=1}^{n_{\text{bins}}} p_k^i \delta_{ij}. \tag{5}$$

Where N_j is the number of events in the j-th bin of the deconvolved histogram, p_k^i is the probability assigned by the model to the i-th bin in the deconvolved histogram for the k-th event and δ_{ij} is the Kronecker delta. By comparing equation 5 with equation 1, one can observe that they are similar to eachother. However in equation 1 the distributions are continuous and hence an integral equation is necessary, in the case of equation 5 the distributions are now discrete (histograms) and so it is only concerned with counting how

[2] Units have been omitted since they are not relevant to the comparison of absolute value between different features.
[3] I used Tensorflow–GPU; results should be the same for Tensorflow–CPU.

many events are in each bin rather than how many events have specific values. Furthermore equation 1 operates on the true distribution after all events have been counted, whereas equation 5 builds up the output distribution N_j on an event–by–event basis. Finally the measured value/bin does not 'appear' explicitly in equation 5 since it is used by the neural network model to determine p_k and so it 'appears' implicitly in this equation. So it can be understood that equation 5 mimics a convolution on an event–by–event basis such that a close approximation to the true distribution can be obtained from a detected distribution.

Layer Number	Layer Type	Layer Specific Details
0	Input	Input Dimension: Number of kept features, Output Dimension: 100, Activation: Linear
1	ReLU	Output Dimension: Number of bins
2	Output	Output Dimension: Number of bins, Activation: Softmax

Table 2: Summary of the model used to perform bin classification.

Layer Number	Layer Type	Layer Specific Details
0	Input	Input Dimension: Number of kept features, Output Dimension: 100, Kernel Initializer: Glorot normal, Activation: Linear
1	ReLU	Output Dimension: 600
2	Batch Normalization	
3	Dropout	Ratio: 0.1, Seed: 23
4	Tanh	Output Dimension: 750
5	Batch Normalization	
6	Dropout	Ratio: 0.1, Seed: 46
7	ReLU	Output Dimension: 600
8	Batch Normalization	
9	Output	Output Dimension: Number of classes, Activation: Softmax

Table 3: Summary of the model used to perform migration classification.

Layer Number	Layer Type	Layer Specific Details
0	Input	Input Dimension: Number of kept features, Output Dimension: 150, Kernel Initializer: Glorot normal, Activation: Linear
1	ReLU	Output Dimension: 375
2	Batch Normalization	
3	Output	Output Dimension: 1, Activation: Linear

Table 4: Summary of the model used to perform Z mass regression.

C. Model Evaluation

For classification models, Keras provided measurements of model accuracy which were used during training to monitor how similar the performance was between the training and test data, if the training/test accuracy was much greater than the test/training accuracy then the model was overfitting/underfitting. This accuracy was calculated by finding the percentage of events for which the highest probability was assigned to the correct class. Since Keras does not provide this calculation for the regression model, the mean–squared error and mean–absolute error were used to check for overfitting during training. Using the model's predictions for the data, the percentage of events predicted correctly could be calculated to obtain the accuracy in the same way as for the classification model. Model accuracy sets an upper bound on how much the deconvolved histogram will differ from the true

histogram and therefore an accuracy $\sim$ 100% was initially aimed for. During the study it was found that with the classification models, the accuracy was not a good predictor of whether or not the final deconvolved histogram was a good fit with the true histogram or not. For example in bin classification with 20 bins and an accuracy of $\sim$ 55% a deconvolved histogram could be obtained with a Kolmogorov–Smirnov Test score $\sim$ 1, an Anderson–Darling Test score $\sim$ 1 and a χ^2 per degree of freedom value < 1.

In order to test the histogram fits quantitatively ROOT's built–in Kolmogorov–Smirnov (KS), Anderson–Darling (AD) and χ^2 per degree of freedom (χ^2/DOF) tests were used to compare the deconvolved histogram to the true histogram. In the the KS and AD tests a result of 1 implies a good fit while in the χ^2/DOF test a result < 1 implies a good fit. In ROOT the KS test is known to produce incorrect results for binned data (histograms) [2] and will give a higher value than it should, however it can still be used to indicate whether or not a histogram has a good fit and can be compared against other KS results to compare performance between different models.

To test the performance of the migration deconvolution method it is not sufficient to only deconvolve the cut data. Cut data is not a realistic example of data that would be used if this method were to be employed on real experimental data, since cut data relies on already knowing the truth values for each event. Uncut test data is prepared by storing the data that is 'cut out' during data preparation and adding some[4] of it to the cut test data.

To test the robustness of the models, 10-fold cross–validation tests were performed for which KS, AD and χ^2/DOF values were determined for each trained model. Large standard deviations in these results imply that the model is not robust since even if it could obtain a good fit for some data it may not produce a similarly good fit for different but similar data and therefore is not a reliable model.

III. Results and Discussion

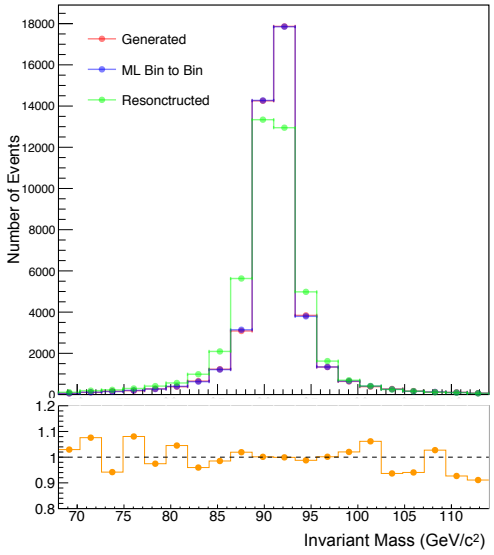

Figure 1: Histogram deconvolution result and ML/Gen histogram ratio from bin classification.

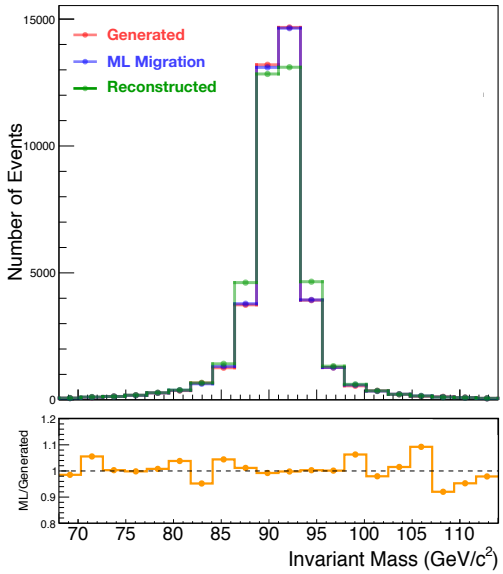

Figure 2: Histogram deconvolution result and ML/Gen histogram ratio from migration classification.

Figures 1 and 2 show, respectively, the result of using the bin classification method and migration classification method to perform histogram deconvolution on a Z–mass histogram with 20 bins. The bin migration classification allowed events to migrate by at most 1 bin. One can see that, qualitatively, the deconvolved histograms fit very well with the true ones, and the ratio plot further shows that their bin entries never differ by more than 10%.

[4]The amount added in is such that it preserves the training/test data ratio, e.g. for an 8 : 2 ratio only 20% of the 'cut out' data would be added into the test data.

Migrations	% of Data	Cut KS	Cut χ^2 /DOF	Uncut KS	Uncut χ^2 / DOF
$\pm$ 1	91.5	0.902	0.775	0.00	20.0
$\pm$ 2	97.8	0.673	1.03	0.00	8.30
$\pm$ 3	98.8	1.00	0.558	0.00	5.49
$\pm$ 4	99.1	0.391	1.99	0.00	6.23
$\pm$ 5	99.4	0.998	0.686	0.0986	3.40
$^{+17}_{-5}$	100	0.914	1.25	0.914	1.25

Table 5: Performance comparison (on a 20 bin histogram) between reconstructing cut data and uncut data. Every value is stated up to 3 SF.

Table 5 illustrates the effect that cutting data for the migration method has on the performance of that method. The second column states the percentage of events that migrated by at most n bins, where n is given in the first column. One can see that despite successfully deconvolving the cut data, when used to deconvolve the uncut data the method yields unacceptable scores. If one chooses migration limits such that no data is cut then this method successfully deconvolves the data. However this often results in more classes than if bin classification was used; and since the model is deeper than that for bin classification, and the more classes a model trains to predict the longer the training takes, this method will often take a longer amount of time to produce a result than the bin classification method would. Furthermore there is no guarantee that the result will be better than (or even as good as) one obtained from bin classification. Therefore this method, in its current state, offers no advantage over bin classification.

Method	KS Test (Mean $\pm$ STD) (3 SF)				
	20 Bins	40 Bins	60 Bins	80 Bins	100 Bins
Bin	1.00 $\pm$ 0.00	0.925 $^{+0.075}_{-0.223}$	1.00 $\pm$ 0.00	0.956 $^{+0.044}_{-0.131}$	1.00 $\pm$ 0.00
Regression	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00
TUnfold	0.898 $^{+0.102}_{-0.193}$	1.00 $\pm$ 0.00	0.998 $^{+0.002}_{-0.006}$	0.986 $^{+0.014}_{-0.029}$	0.997 $^{+0.003}_{-0.010}$

Table 6: Kolmogorov–Smirnov test scores from 10-fold cross–validation.

Method	χ^2/DOF (Mean $\pm$ STD) (3 SF)				
	20 Bins	40 Bins	60 Bins	80 Bins	100 Bins
Bin	0.0639 $\pm$ 0.0332	0.120 $^{+0.181}_{-0.120}$	0.0170 $\pm$ 0.0073	0.091 $^{+0.130}_{-0.091}$	0.0357 $\pm$ 0.0279
Regression	15.4 $\pm$ 1.6	10.3 $\pm$ 3.4	6.64 $\pm$ 1.05	5.68 $\pm$ 1.62	4.63 $\pm$ 0.89
TUnfold	1.39 $\pm$ 0.55	1.01 $\pm$ 0.34	1.03 $\pm$ 0.30	1.02 $\pm$ 0.18	1.11 $\pm$ 0.32

Table 7: χ^2/DOF test scores from 10-fold cross–validation.

Method	AD Test (Mean $\pm$ STD) (3 SF)				
	20 Bins	40 Bins	60 Bins	80 Bins	100 Bins
Bin	0.988 $^{+0.012}_{-0.030}$	0.905 $^{+0.095}_{-0.272}$	1.00 $\pm$ 0.00	0.935 $^{+0.065}_{-0.189}$	0.999 $^{+0.001}_{-0.004}$
Regression	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00
TUnfold	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00

Table 8: Anderson–Darling test scores from 10-fold cross–validation.

Tables 6, 7 and 8 show, respectively, the KS, χ^2/DOF and AD results of 10-fold cross–validation for bin classification, regression and TUnfold. The results for bin classification show that for histograms with 20–100 bins it can achieve KS and AD scores $\sim$ 1 and χ^2/DOF $\ll 1$. The χ^2/DOF standard deviations are small enough that within these bounds the values would still suggest a very good fit. The KS and AD standard deviations are relatively large compared to the mean values, however the KS scores are still comparable to those for TUnfold and the AD scores are much higher than for TUnfold. This implies that using this network results in a robust deconvolution method. The results for regression show that the method used is not able to perform deconvolution and an alternative method is required for regression to work for this problem.

Table 9 shows that with no alteration to the bin classification model or the interpretation of its predictions, deconvolution of a completely different distribution is successful with KS and AD $\sim$ 1 and χ^2/DOF < 1. The values for the reconstructed data demonstrate that this machine learning method vastly improves the quality of the histogram. Even with a poor 'detector resolution' which corresponds to $\sigma = 0.10$ the method obtains a histogram with a good fit. Figure 3 shows the histograms corresponding to the $\sigma = 0.10$ case which illustrate the qualitative success of this method with a steeply falling distribution.

Method	KS Test		χ^2/DOF		AD Test	
	$\sigma = 0.03$	$\sigma = 0.10$	$\sigma = 0.03$	$\sigma = 0.10$	$\sigma = 0.03$	$\sigma = 0.10$
Bin	0.981	1.00	0.491	0.824	0.991	0.985
Reco.	0.00	0.00	230	1020	0.00	0.00

Table 9: Test scores obtained by applying bin classification to toy data (60 bins) with a steeply falling distribution. All values are stated up to 3 SF.

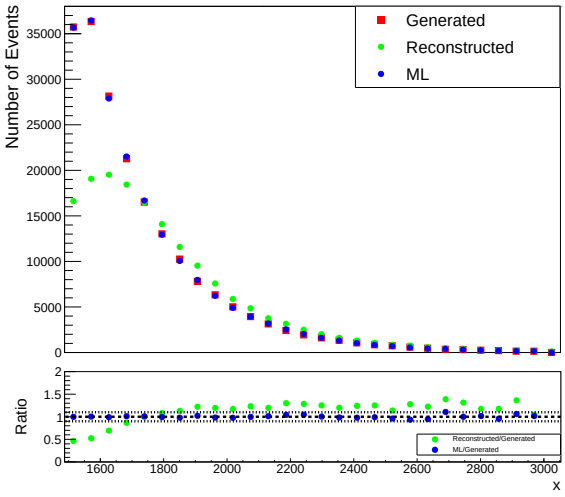

Figure 3: Bin classification deconvolution result for the toy data.

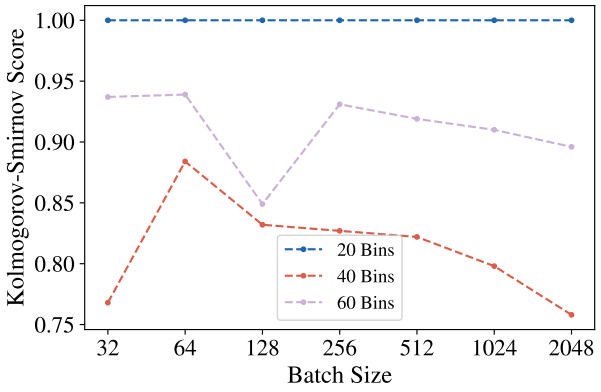

Figure 4: KS dependence on batch size.

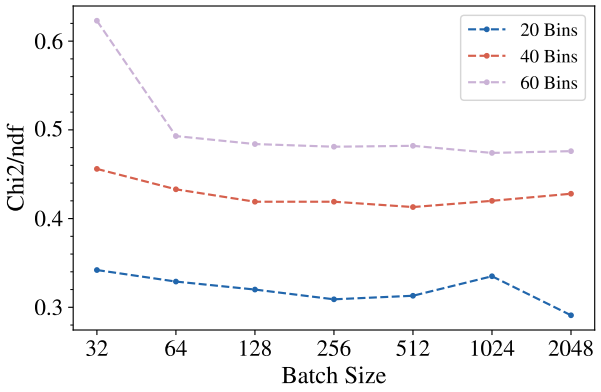

Figure 5: χ^2/DOF dependence on batch size.

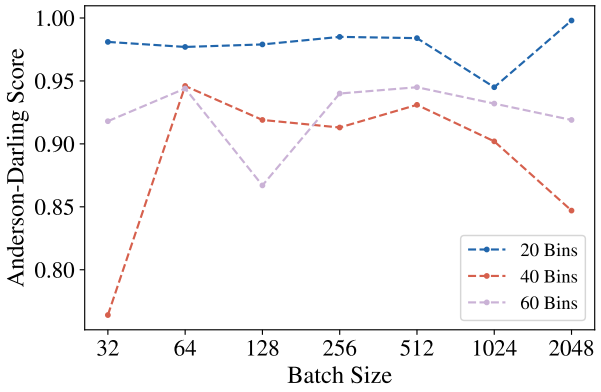

Figure 6: AD dependence on batch size.

Figures 4, 5 and 6 show, respectively, how changing the batch size used during bin classification training affects the KS, χ^2/DOF and AD scores of the deconvolved histogram. This was performed using 5 input features and 60 bins. These figures show that the batch size has a negligible influence on the success of the method which may imply that the model does not 'learn' the shape of the distribution.

The motivation for this idea is as follows. Batches containing more than one event but not the whole dataset are used during training, this is a technique called 'mini–batch gradient descent'. In this technique, for each batch of data, the program uses the weights of the neural network to calculate the loss for that particular batch and updates these weights to minimise the loss. Minimising the loss is equivalent to maximising the prediction probability for the correct class in each event of the batch, which ideally would result in a prediction probability of 1 for the correct class in each event. This however is unlikely since each event uses the same network structure and it is therefore unlikely that the model can find weights that result in a value of 1 for the correct probability for every event simultaneously. So in each batch the weights are updated such that on average for the events in that batch the correct prediction probability is maximised.

If one considers the case in which there is two classes to choose from, then in a single batch of 32 events if class 1 has 30 events and class 2 has only 2 events then the weights may be updated such that a high probability is assigned to class 1 for most events since this would correspond to a very low loss value. In effect the network would be learning that the vast majority of events correspond to class 1 and therefore predicting that class for every event minimises the loss function better than sometimes predicting class 2, and risking being wrong for more than 2/32 events.

In the case of the m_Z spectrum with 20 classes, by looking at the generated histogram, one can see that the classes/bins have a similar imbalance to this simple example. So, by this logic, if one picks a batch size large enough to have a similar class ratio as the whole data set, then one could expect that the network would learn to predict each event with a bias towards the most common classes and would therefore 'learn' the shape of the training histogram instead of learning to predict each event correctly to minimise the loss function. So one would expect the goodness of fit of the deconvolved histogram to improve with an increasing batch size. However, figures 4, 5 and 6 show that the KS, χ^2/DOF and AD are independent of the batch size and therefore the model may not be biased towards the shape of the training histogram.

IV. Conclusions

Our results show that our attempted regression and migration methods cannot deconvolve our histograms successfully. The migration method shows some promise since it can obtain a good fit with cut data but not with uncut data. Good results can only be obtained if the data is not cut at all which, depending on the exact dataset, can result in more classes than the bin classification method. So this migration method does not simplify the problem that the neural network is attempting to solve and offers no advantage over the bin classification method. It might be possible to find a regression method which is successful and robust, one method to consider is quantile regression since this could be used to create probability distributions similar to the bin classification method and result in another convolution mimicking method. In order for migration to be viable a method that either uses migration cutting but results in good fits with uncut data, or that does not cut data but is simply faster at training than bin classification is required.

We have shown that bin classification with our alternate interpretation of the predictions is a powerful method for histogram deconvolution that works for a range of total bin numbers and two different distribution shapes. In comparison to the method in [4] our method uses much less data and only needs one model training session which considerably reduces the total time taken for the deconvolution. Furthermore the model we used for bin classification is much simpler than the model in [4] which had the output dimension of the hidden layer scaling quadratically, instead of linearly, with the number of bins. For 100 bins our method with the model from [4] took ~ 14 seconds per epoch to deconvolve a 100 bin histogram whereas for our model it only takes ~ 1 second per epoch. One way to improve our method would be to find a model that works as well as (or better than) the current one without scaling the number of nodes with the total bin number as much as the current model does. Currently this method takes ~ 3 minutes for a single training session with a batch size of 2048 events, $\sim 200,000$ events in total using Tensorflow–GPU on a laptop with an Nvidia GeForce GTX 1060M GPU. This is quite fast for neural network training but TUnfold takes only a couple of seconds so for an unfolding method it is quite slow. Nevertheless a few minutes is not terribly long to wait and so this method could be a sensible way to perform histogram deconvolution.

Finally, it should be noted that when measuring data with a cut, e.g. only recording events for which the invariant mass is within some chosen interval, migrations may cause an event to not be recorded because its measured value is outside the interval despite its true value being inside the interval. Conversely an event may be recorded because the measured value is within the interval despite the true value not being within the interval. The effect of such migrations has not been considered in this study and remains a topic for future work. Furthermore we have not determined the uncertainties on our deconvolved histograms but this will be addressed in a future paper.

V. Acknowledgements

I would like to thank Dr Isildak for his expert guidance and for allowing me to assist with his research. I would also like to thank those responsible for the CERN Summer Student Program for their support, funding and the opportunity to work at CERN.

References

[1] Volker Blobel, *An Unfolding Method For High Energy Physics Experiments*, arXiv:hep-ex/0208022v1

[2] Rene Brun and Fons Rademakers, *ROOT - An Object Oriented Data Analysis Framework*, Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86. See also `http://root.cern.ch/`.

[3] S. Schmitt, *TUnfold, an algorithm for correcting migration effects in high energy physics*, Journal of Instrumentation, 7 (2012), pp. T10003T10003.

[4] Alexander Glazov, *Machine learning as an instrument for data unfolding.* arXiv. 1712.01814 (2017).

[5] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, and M. Zaro, *The automated computation of tree-level and next-to- leading order differential cross sections, and their matching to parton shower simulations*, JHEP, 07, 2014.

[6] T. Sjöstrand, S. Mrenna and P. Skands, *A brief introduction to PYTHIA 8.1.* Comput. Phys. Comm. 178, 2008.

[7] The DELPHES 3 collaboration, J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi, *Delphes 3: a modular framework for fast simulation of a generic collider experiment.* Journal of High Energy Physics, 2014.

[8] Particle Data Group (PDG), *Review of particle physics*, Phys. Rev. D, 98 (2018), p. 030001.

[9] The CMS Collaboration, *Performance of electron reconstruction and selection with the CMS detector in proton-proton collisions at* $\sqrt{s} = 8$ *TeV*, 2015 JINST 10 P06005

[10] Pedregosa et al., *Scikit-learn: Machine Learning in Python*, JMLR 12, pp. 2825-2830, 2011

[11] CMS Collaboration, *Search for narrow resonances and quantum black holes in inclusive and b-tagged dijet mass spectra from pp collisions at* $\sqrt{s} = 7$ *TeV*, JHEP 01 (2013) 013, doi:10.1007/JHEP01(2013)013, arXiv:1210.2387.

[12] François Chollet et al., *Keras*, 2015, `https://keras.io`

[13] Martín Abadi et al., *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015. Software available from tensorflow.org.