Convolutional Neural Network for Track Seed Filtering at the CMS High-Level Trigger

IOP Conf. Series: Journal of Physics: Conf. Series **1085** (2018) 042040 doi:10.1088/1742-6596/1085/4/042040

Convolutional Neural Network for Track Seed Filtering at the CMS High-Level Trigger

Adriano Di Florio

Dipartimento Interateneo di Fisica di Bari and I.N.F.N.-Sezione di Bari
via Amendola 173, 70126 Bari, Italy

E-mail: adriano.diflorio@ba.infn.it

Felice Pantaleo

CERN, European Organization for Nuclear Research
Geneva 23, CH-1211 Geneva, Switzerland

E-mail: felice.pantaleo@cern.ch

Antonio Carta

Dipartimento di Informatica, Università di Pisa
Largo Bruno Pontecorvo 3, I-56127 Pisa, Italy

E-mail: antonio.carta@di.unipi.it

on behalf of the CMS collaboration

Abstract.

Starting with Run II, future development projects for the Large Hadron Collider will constantly bring nominal luminosity increase, with the ultimate goal of reaching a peak luminosity of $5{\cdot}10^{34} cm^{-2} s^{-1}$ for ATLAS and CMS experiments planned for the High Luminosity LHC (HL-LHC) upgrade. This rise in luminosity will directly result in an increased number of simultaneous proton collisions (pileup), up to 200, that will pose new challenges for the CMS detector and, specifically, for track reconstruction in the Silicon Pixel Tracker. One of the first steps of the track finding work-flow is the creation of track seeds, i.e. compatible pairs of hits from different detector layers, that are subsequently fed to higher level pattern recognition steps. However, the set of compatible hit pairs is highly affected by combinatorial background resulting in the next steps of the tracking algorithm to process a significant fraction of fake doublets. A possible way of reducing this effect is taking into account the shape of the hit pixel cluster to check the compatibility between two hits. To each doublet is attached a collection of two images built with the ADC levels of the pixels forming the hit cluster. Thus the task of fake rejection can be seen as an image classification problem for which *Convolutional Neural Networks* (CNNs) have been widely proven to provide reliable results. In this work we present our studies on CNNs applications to the filtering of track pixel seeds. We will show the results obtained for simulated event reconstructed in CMS detector, focusing on the estimation of efficiency and fake rejection performances of our CNN classifier.

1. The CMS Trigger System

The Compact Muon Solenoid (CMS) [1] is a general purpose detector designed for the precision measurement of leptons, photons, and jets, among other physics objects, in proton-proton as well as heavy ion collisions at the CERN LHC [2]. The LHC is designed to collide protons at a center-of-mass energy of 14 TeV and a luminosity of $10^{34} cm^{-2} s^{-1}$. At design luminosity, the pp interaction rate is about 1 GHz but only a small fraction of these collisions contains events that can be interesting for CMS physics analyses and can be stored to be accessible offline. The trigger system is devoted to the selection of these events from the totality of the inelastic collision events. In order to accomplish this task, the CMS trigger system utilizes two levels [3]. The first one (L1) is implemented in custom hardware (FPGAs and ASICs). Within $4\mu s$ of a collision, it makes selection of candidates objects based on raw data from calorimeters and muon detectors and it restricts the output rate to 100 kHz. Events are then passed to the High-Level Trigger (HLT) that further refines the purity of the physics objects, and selects an average rate of 400 Hz for final offline storage.

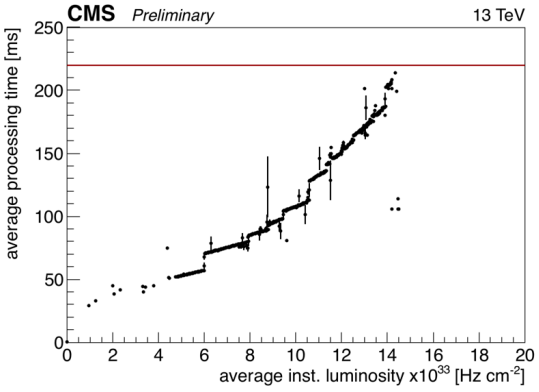

Figure 1. Average processing time per event with respect to the average instantaneous luminosity. [4]

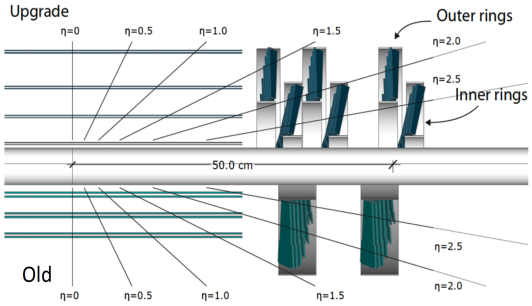

Figure 2. The Run 2 updated Silicon Pixel Tracker geometrical layout, optimized to offer full 4-hit tracking coverage up to a value of the pseudo-rapidity of 2.5 [5]

2. Track reconstruction at HLT

The HLT hardware consists of a processor farm, the Event Filter Farm (EVF) with about 26000 Intel Xeon cores [3].The HLT gathers information from all the CMS detectors and among them the Silicon Pixel Tracker has a crucial role in two of the first steps for the selection: the track reconstruction and the primary vertex finding. The track reconstruction software used at HLT is practically identical to that used for offline reconstruction with the difference that it has to fulfill stringent CPU timing constraints: the event selection has to be fulfilled with an average latency smaller than 260 ms. With the instantaneous luminosity increasing up to the ultimate goal of $5 \cdot 10^{34} cm^{-2} s^{-1}$ for the High Luminosity LHC (HL-LHC) upgrade, this time constraint is rapidly approaching (see Figure 1). Beside this, the number of simultaneous proton collisions per bunch crossing ("pileup") will continue to increase: the highest pileup reached averaged over a data run in 2011 and 2012 was ≈ 16 and ≈ 34, respectively, while in 2016 it exceeded 50 [6].

The track reconstruction procedure can be roughly split in four steps [3]. The first step is the construction of seeds (seeding), i.e. few (2-4) Pixel hits that act as the building blocks for track candidates and define the initial estimate for trajectory parameters. These candidates are then propagated by means of Kalman Filter techniques to find further compatible hits and build a track (building). The tracks are then fitted and marked with quality flags (fitting and selection).

In the current configuration, the very first step of seeding consists in collecting and mapping all the valid hits on the Silicon Pixel Detector layer. The Run 2 updated Silicon Pixel Tracker is arranged in four barrel layers (BPIX) and three forward disks (FPIX) in each end cap region (see Figure 2). Then, iterating on all the couples of consecutive *seeding layers*, the algorithm selects and collects all the compatible hit pairs (*doublets*), i.e. the hit pairs that could belong to the same particle and track. The compatibility between two hits is evaluated only on the basis of geometrical considerations, such as cuts in η, ϕ and r.

The doublet production is thus highly dominated by combinatorial background and constitutes a bottleneck for the subsequent steps. For example, for $t\bar{t}$ simulated events at energy of the center of mass of $\sqrt{s} = 13$ TeV, with average pileup $< PU >= 35$ and bunch time spacing of 25 ns, about 10^6 doublets are produced per each event. Therefore the reduction of the track reconstruction time hinges on the rejection of seeding combinatorial background. In the next sections a method for filtering doublet seeds based on CNNs is proposed.

3. Convolutional Neural Networks for Doublet Seeds Filtering

A traditional deep neural network is a machine learning model whose goal is to approximate a function $\hat{f}$ by composing a sequence of simpler functions. It consists of an input layer, one or more hidden layers and an output layer [7]. To a first approximation, each layer acts on the input as a composition of a matrix multiplication, tuned with a set of biases (b) and weights (w), and an activation function. Convolutional Neural Networks [8] are a specialized kind of neural network for processing data that has a grid-like structure, such as 2D images. The building block of a CNNs is a layer that uses *discrete convolution* in place of general matrix multiplication [7]. A convolutional layer takes in input a $n \times m \times r$ two dimensional image where r is the number of channels or depth, e.g. an RGB image has $r = 3$. The layer is characterized by k filters of size $k \times l \times q$ where k and l are smaller than the dimensions (n, m) of the image and typically $q = r$ (the filter has full depth receptive field). Each filter is devoted to detect some specific type of feature or shape at some spatial position in the input. The output of one or more stacked convolutional layers is usually passed to pooling layers that perform dimensionality reduction. A $p \times p'$ *pooling layer* replaces a $p \times p'$ region of the image with a single value, e.g. the maximum

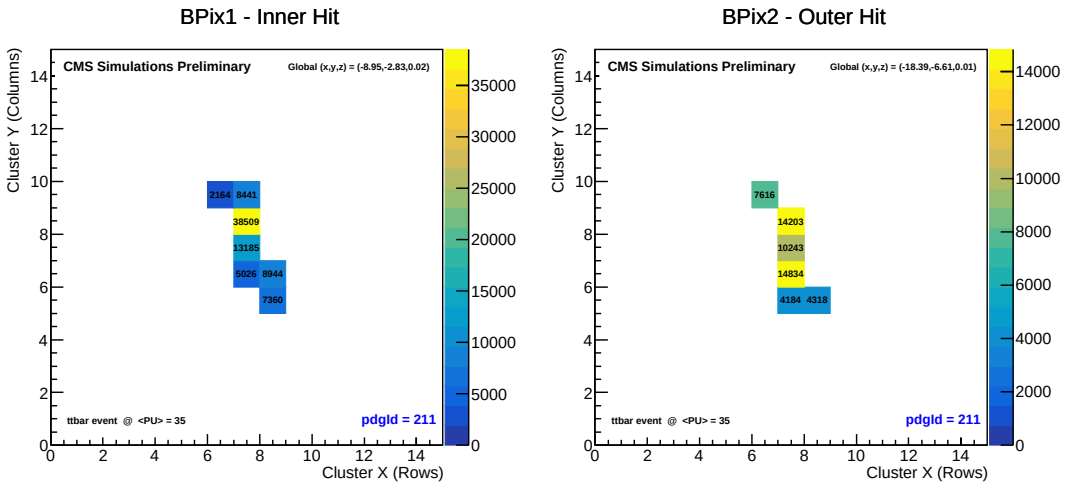

Figure 3. Example of a BPix1-BPix2 true doublet generated from $t\bar{t}$ simulated events corresponding to a π^+ track.

value in that region for a max pooling layer.

3.1. Doublet Hits Cluster Shape

As described above, while building seeds, the compatibility between two hits is evaluated only on the basis of geometrical considerations. A possible way of reducing the doublet fake ratio is taking into account that each hit is actually a cluster of pixels with its own shape. Each pixel is characterized by its 16-bit A.D.C. level ($\max_{\mathrm{ADC}} = 2^{16} - 1$) and its local position (x, y) on the layer, where x is the azimuthal direction in the barrel detector and radial direction in the forward detectors. Then a single hit can be considered as a collection of three vectors: the x, the y and the ADC levels of each of its pixels. For each hit a 15×15 squared matrix M is built with the pixel local x on the rows and the local y on the columns. The matrix center is matched with the hit center of charge and each element m_{ij} is set to the A.D.C. level of the corresponding (x_i, y_j) pixel. Those pixels that stride over the hit cluster boundaries are set to zero (zero-padding technique).With this procedure each doublet can be considered as a collection of two 15×15 matrices, an example is shown in Figure 3, or as a two channel 2D image ($n \times m \times r = 15 \times 15 \times 2$). Thus the rejection of fake doublets is reduced to an image/pattern recognition task, perfectly suitable for being dealt with CNNs.

3.2. Doublet dataset: generation and features

To test the feasibility of this kind of approach, the generation (via PYTHIA 8 [9]) and the reconstruction of $t\bar{t}$ events at energy of the center of mass of $\sqrt{s} = 13$ TeV, with average pileup $< PU >= 35$ and bunch time spacing of 25 ns has been simulated within the CMS software framework (CMSSW [10]). For each event, both all the doublets produced and all the MC matched reconstructed tracks, i.e. associated with a tracking particle, are collected. A doublet is then labeled as *true* only if it is formed by pixel hits belonging to the same MC matched track. About 10^6 doublets are produced per each event and the ratio between *true* and *fake* doublets is between 300-400. For each doublet 537 parameters are stored:

- 225 + 225 *pixels* for the inner and the outer hit;
- 63 *doublet features* defined for each doublet and that include detector information and further hit and cluster characteristics;

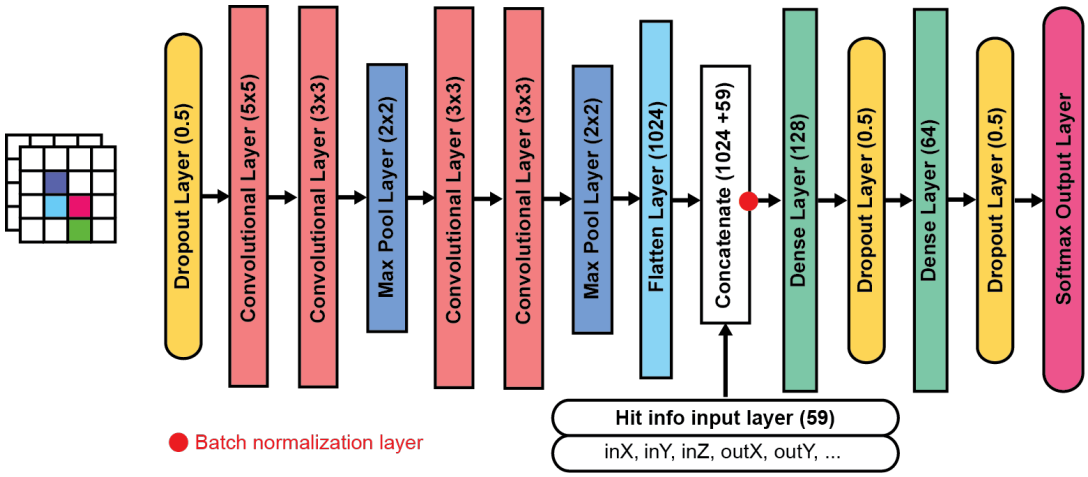

Figure 4. Layer map model architecture

- 24 *track labels* defined only for MC matched doublets, e.g. the corresponding track vertex coordinates, pt and eta;

On the whole, 1000 events are simulated, 800 for the training dataset, 150 for testing and 50 for validation. The training and the validation set are balanced so that the ratio between fake and true doublet is one. The whole balanced training dataset is composed of approximately 2.5 millions doublets.

4. CNN Classifier: the layer map model

The architecture for the doublet filtering classifier (*layer map model*) is shown in Figure 4 and it concatenates:

(i) *CNN block*: conventional stack of convolutional layers (4) with 5×5 or 3×3 filters and max pooling layers (2) whose output is reduced to a one dimensional structure through a flatten layer that returns a 1024 elements vector.

(ii) *Dense block*: stack of two fully connected layers that are fed with the one dimension reduced images from the previous block and 59 further doublets info (such as hits' detectors and coordinates).

The architecture is completed by a batch normalization layer inserted between the two blocks, *linear rectifier* activations for the convolutional and dense layers, three dropout layers and a *softmax* output layer for binary classification [7]. The two output neurons return the complementary probabilities that a doublet is fake or true ($p_{true} + p_{fake} = 1.0$). The 15×15 doublets pads, before being fed to the first block, are pre-processed as follows.

- Each pixel content is standardized with the mean and the standard deviation pre-computed on the whole 2.5 millions doublet dataset: $\mu = 13382.00$, $\sigma = 10525.13$.
- The input is split in 4 *angular channels*, 2 for the inner hit and 2 for the outer hit. From each hit pixel matrix $M_{i,o}$ two matrices are extracted as extra channels: one ($M^{cos}_{i,o}$) resulting from the multiplication of the original matrix by the cosine of the hit incident angle and one by the sine of the angle ($M^{sin}_{i,o}$) (θ_i for the inner and θ_o for the outer hit, see Figure 5).
- The input is split in 20 additional *detector channels*, 10 for the inner hit and 10 for the outer hit. Every channel corresponds to a single layer of the detector (4 barrel and 6 endcap) and is initialized to a 15×15 zero matrix. Then only the two channels corresponding to the inner and the outer layer are set, respectively, to the inner hit and the outer hit cluster matrix. This approach allows us to separate the hit shape clusters based on the specific layer and to apply a different transformation.

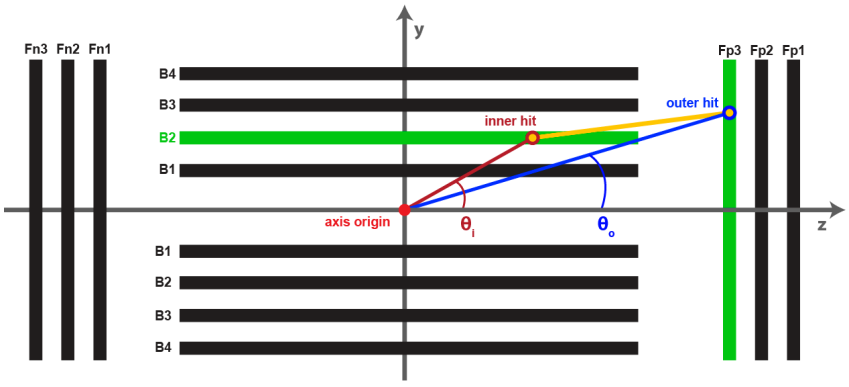

Figure 5. Example of an hit doublet with the inner hit on the third barrel layer (B3) and the outer hit on the first forwared layer (Fp1).

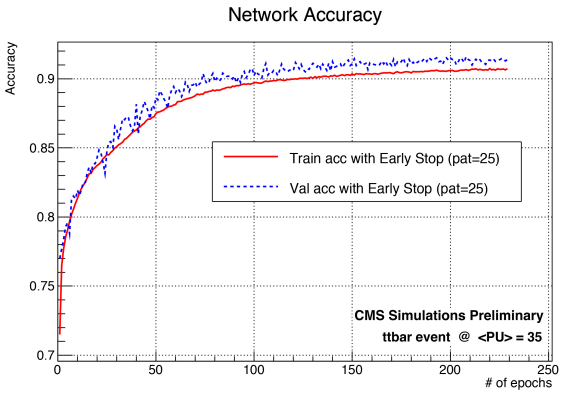

Figure 6. Accuracy for train (red ——) and validation datasets (blue - - - -) with early stopping patiance=25.

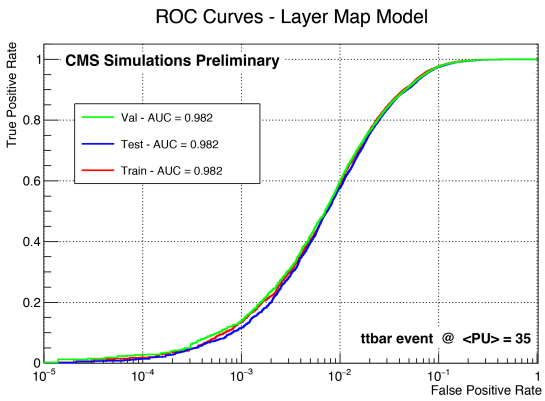

Figure 7. ROC curves for the train,test and validation dataset.

To sum up, the input dataset for the *layer map* model is composed of 24 channels 15×15 images standardized with their mean and their standard deviation. The model has been developed and trained with the `Keras` package [11], a high-level neural networks API, running on the top of `TensorFlow` library [12]. The network has been trained on NVIDIA Tesla K20 nodes at Bari Physics Department Tier 2 (ReCaS) [13] and on NVIDIA GTX1080 Ti GPUs at CERN.

5. Model testing and results

Once tested and tuned on a smaller sample, the model has been trained on the whole dataset (2.5M doublets) split in ten batches of 250k doublets in order to fit the available memory. The model has been trained with a *categorical cross entropy* loss function [7], using *Adam* [14] optimizer and accuracy as evaluation metric. At each batch training iteration the weights and the parameters of the trained network are passed to the next step. This procedure has been carried out for 20 global epochs on the whole dataset and each iteration is run with an *early stopping* callback, that is a form of regularization used to avoid over-fitting. It stops the network training when the selected metric (validation accuracy in our case) does not improve for a given number of consecutive epochs, denoted as *patience* (p=25 for the *layer map* setup). The training took about 5 days on a NVIDIA Tesla K20 node.

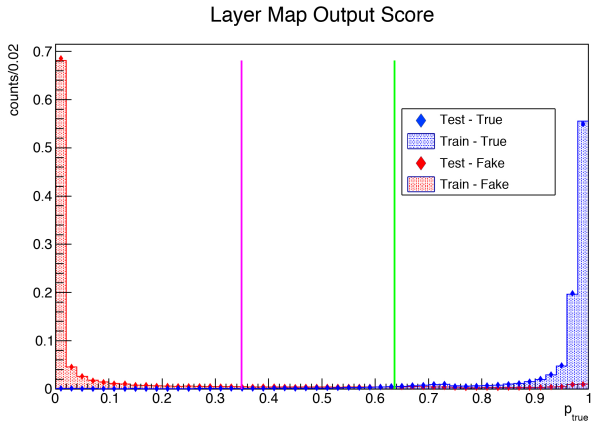

Figure 8. Network score for true (blue) and fake (doublets) for train (filled histogram) and test dataset (diamond markers). In purple the 0.94 accuracy threshold. In green the 0.99 efficiency threshold.

Table 1. Layer map network scores for train,validation and test dataset.

			TN/N at fixed TP/P		**TP/P at fixed TN/N**	
	AUC	**Acc**	**TP/P = 0.99**	**TP/P= 0.999**	**TN/N=0.99**	**TN/N=0.5**
Train	0.982	0.940	0.85426	0.6706	0.5896	0.9997
Test	0.982	0.941	0.8542	0.6709	0.5899	0.9996
Val	0.982	0.939	0.8525	0.6707	0.5948	0.9996

Figure 6 shows the model accuracy for train and validation dataset versus the number of training epoch for the first 250k batch. Validation curves follow the same trend as the training ones and this indicates that the network is not over-fitting on the training data. Note that the validation accuracy is always greater than the training one. This behavior depends on the fact that dropout layers are turned off when the network process the validation data. Therefore the network has more connections and neurons active thus is more complete and accurate.

The ROC curves for validation, test and training dataset, shown in Figure 7, completely overlay each other, and the area under the curves (AUCs) is more than 0.98. While assuring a 0.99 efficiency (true positive rate), network's sensitivity (true negative rate) reaches 0.85. The highest accuracy reached is about 0.94 for all the three datasets. See Table 1 for further network performance results. The normalized output score, namely the network estimated probability that a doublet is true (p_{true}) shows optimal separation between fake and true doublets sample. Both train and test p_{true} distributions are plotted in Figure 8 and the cut for an efficiency of 0.99. In order to compare them a two sided Kolmogorov-Smirnov test has been performed. This tests whether two samples are drawn from the same distribution [15]. For both true and fake histograms, the resulting score is $KS \approx 0.070$ corresponding to a p-value of $p_{val} \approx 0.961$, that assures us that the two histogram come from the same distribution with a very high level of confidence.

6. Conclusions and acknowledgments

In conclusion, the results described show that CNN techniques for mitigating combinatorial explosion look very promising and need to be further explored. Ongoing work includes the verification of the effect on the downstream track reconstruction, the exploration of different hardware architectures for fast inference and the final integration in the CMS reconstruction framework.

References

[1] CMS collaboration, *The CMS experiment at the CERN LHC*, 2008 JINST **3** S08004.
[2] L. Evans and P. Bryant, *LHC machine*, 2008 JINST **3** S08001.
[3] CMS collaboration, The CMS trigger system, JINST **12** (2017) P01020.
[4] https://twiki.cern.ch/twiki/pub/CMSPublic/HLTplotsCHEP2016/ .
[5] CMS Collaboration, *CMS Technical Design Report for the Pixel Detector Upgrade*, CERN-LHCC-2012-016, CMS-TDR-011 (2012).
[6] https://twiki.cern.ch/twiki/bin/view/CMSPublic/LumiPublicResults .
[7] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, The MIT Press (2016).
[8] Y. LeCun, et al. (1989), *IEEE Communications Magazine*, **27**(11), 4146.
[9] T. Sjstrand, S. Mrenna and P. Skands, *JHEP05* (2006) **026**, *Comput. Phys. Comm.* 178 (2008) **852**.
[10] https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookCMSSWFramework .
[11] Chollet, François and others, Keras, 2015, GitHub, https://github.com/fchollet/keras .
[12] TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, http://tensorflow.org .
[13] ReCas is a project financed by the italian MIUR (*PONa3_00052, Avviso 254/Ric.*); its web page is `http://www.recas-bari.it/index.php/en/`.

[14] D. P. Kingma, J. Ba, *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980.
[15] W.T. Eadie et al. *Statistical Methods in Experimental Physics*, North-Holland, Amsterdam (1971).