information

MDPI

Article

A Machine Learning Approach for the Tune Estimation in the LHC

Leander Grech [1,2,*], Gianluca Valentino [1] and Diogo Alves [2]

[1] Department of Communications and Computer Engineering, University of Malta, MSD 2080 Msida, Malta; gianluca.valentino@um.edu.mt
[2] Accelerator Systems Department, CERN, 1211 Geneva, Switzerland; diogo.alves@cern.ch
* Correspondence: leander.grech@cern.ch

Abstract: The betatron tune in the Large Hadron Collider (LHC) is measured using a Base-Band Tune (BBQ) system. The processing of these BBQ signals is often perturbed by 50 Hz noise harmonics present in the beam. This causes the tune measurement algorithm, currently based on peak detection, to provide incorrect tune estimates during the acceleration cycle with values that oscillate between neighbouring harmonics. The LHC tune feedback (QFB) cannot be used to its full extent in these conditions as it relies on stable and reliable tune estimates. In this work, we propose new tune estimation algorithms, designed to mitigate this problem through different techniques. As ground-truth of the real tune measurement does not exist, we developed a surrogate model, which allowed us to perform a comparative analysis of a simple weighted moving average, Gaussian Processes and different deep learning techniques. The simulated dataset used to train the deep models was also improved using a variant of Generative Adversarial Networks (GANs) called SimGAN. In addition, we demonstrate how these methods perform with respect to the present tune estimation algorithm.

Keywords: LHC; betatron tune; deep learning; SimGANs

Citation: Grech, L.; Valentino, G.; Alves, D. A Machine Learning Approach for the Tune Estimation in the LHC. *Information* **2021**, *12*, 197. https://doi.org/10.3390/info12050197

Academic Editor: Giorgio Kaniadakis

Received: 23 March 2021
Accepted: 27 April 2021
Published: 29 April 2021

1. Introduction

The tune (Q) of a circular accelerator is defined as the number of betatron oscillations per turn [1]. This is a critical parameter in the Large Hadron Collider (LHC), which has to be monitored and corrected in order to ensure stable operations [2] and adequate beam lifetime. The Base-Band Q (BBQ) system [3] in the LHC is used to measure the tune and essentially consists of an electromagnetic pickup followed by a diode-based detection and acquisition system. The diode detectors pick-up a small modulation caused by betatron motion on the large beam intensity pulses and converts it to baseband, which for the LHC is in the audible frequency range.

The BBQ system in the LHC is sensitive enough to not require that the beam be externally excited in order to measure the tune. This normally results in a frequency spectrum, such as the one shown in Figure 1, where the value of the betatron tune frequency should, in principle, be the frequency position of the dominant peak [3,4]. The frequency spectrum in the aforementioned figure was obtained in 2018 and is representative of the spectra obtained during real operation. FLATTOP is a beam mode which occurs after the LHC energy ramp and before collision optics are set [5].

Since the start of the LHC, spectral components at harmonics of the 50 Hz mains frequency have been observed with several different diagnostic systems. Studies have shown that these modulations are on the beam itself with the source of the error found to be the main dipoles. Therefore, the harmonic perturbations are not a result of instrumentation but are real beam excitations [6]. These harmonics are clearly visible in the BBQ system, resulting in a frequency spectrum polluted with periodic spikes every 50 Hz. Since these harmonics are also present around the betatron tune, they are a potential source of error for the tune estimation algorithm in use until LHC Run 2 (herein referenced as the BQ

algorithm). The current tune estimation algorithm applies a series of filters and averaging techniques which have been developed in order to mitigate the impact of the 50 Hz harmonics on the final estimated value. However, it is not uncommon to have the estimated tunes oscillate between neighbouring 50 Hz harmonics. The fact that the tune estimate locks onto a particular 50 Hz harmonic is clearly not desirable. In addition, having the tune jump from one harmonic to another affects the tune feedback (QFB) system, causing it to switch off as a protective measure against unstable behaviour.

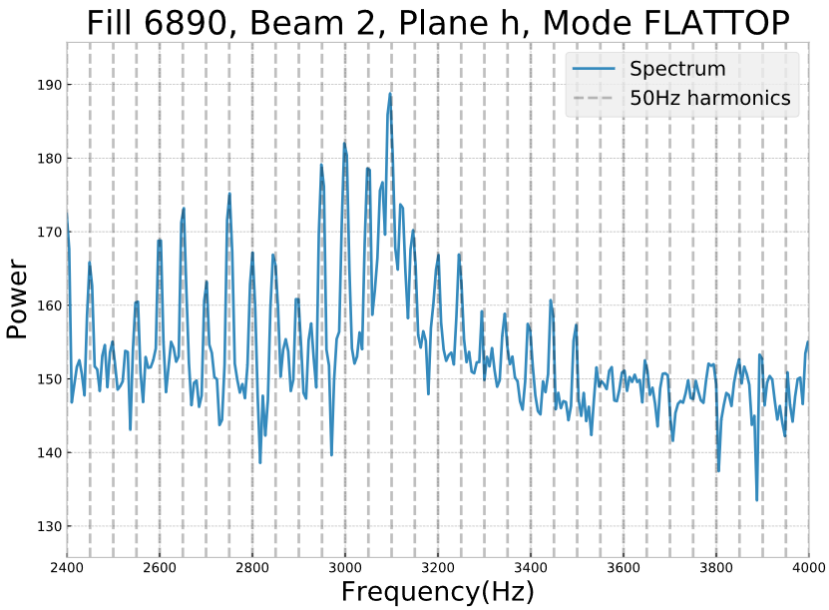

Figure 1. Example of 50 Hz harmonics present in the BBQ spectrum, obtained from LHC Fill 6890, Beam 2, horizontal plane in the FLATTOP beam mode.

The work presented in this paper builds upon the progress made in designing alternative algorithms which are able to reliably estimate the tune from spectra polluted with 50 Hz harmonics [7]. Namely two alternative algorithms will be considered, one which uses a Weighted Moving Average (WMA) and the other uses Gaussian Processes in order to reconstruct the BBQ spectra without 50 Hz harmonics. The difference between the alternative algorithms and this work that now we take a Machine Learning (ML) approach to solve the tune estimation problem. ML offers a set of useful tools that can be used to design a mathematical model that attempts to solve a problem from experience, and automatically improves its performance over time. The aim of this work is to use ML to train a predictor function which can estimate the position of the tune directly from BBQ spectra.

We start by introducing the BQ algorithm, which obtained tune estimates from the BBQ spectra until Run 2 of the LHC and the purpose of which in this work is purely historical. We also briefly explain the alternative algorithms [7] that aim to improve the estimates obtained from a BBQ spectrum. We then introduce a novel, albeit a simple approach using Artificial Neural Networks (ANNs), and highlight its deficiencies and limitations along with its potential room for improvement. Next we aim to improve upon the simple approach by considering the inadequacy of simulated spectra to train an ANN and consider a variant of Generative Adversarial Networks (GANs), called SimGAN as a potential solution to this problem. Finally, we collate and discuss the results obtained from the current and alternative algorithms, and all the ML approaches.

2. Tune Estimation Algorithms

The BBQ system is normally configured to provide a spectrum of 1024 frequency bins at a rate of 6.25 Hz. Since the original signal is sampled at the LHC revolution frequency of approximately 11.25 kHz, the spectral resolution is approximately 5.5 Hz. This defines

the frequency range and resolution available for any system that estimates the tune from BBQ spectra.

We start by considering the BQ algorithm, where the set of sequential processing blocks that form the present tune estimation algorithm is depicted in Figure 2. First, each calculated spectrum update is passed through a bank of independent exponential moving average filters. Each filter is applied to a single frequency bin with the aim of reducing spectral noise. Median and average filters are subsequently applied to the latest spectrum to increase its smoothness and mitigate the effect of the 50 Hz harmonics. At this stage the frequency corresponding to the maximum value of the processed spectrum is taken. This frequency is subsequently refined by going back to the output of the bank of exponential moving averages and performing a Gaussian fit of the spectral region in its immediate vicinity. This last step attempts to obtain a better estimate of the tune, beyond the frequency resolution of the spectrum.

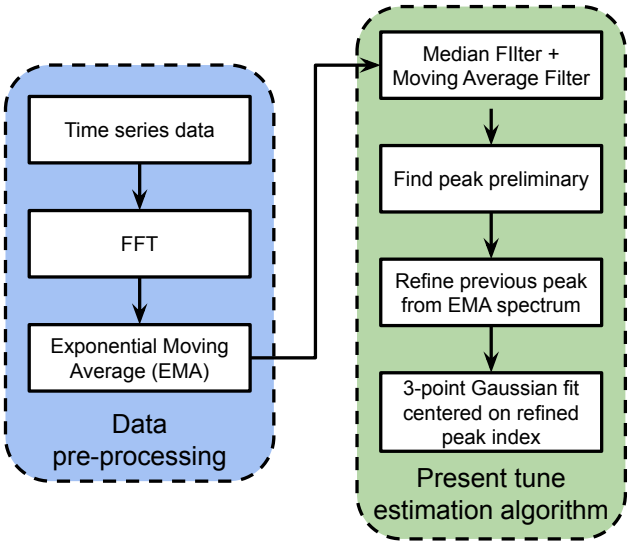

Figure 2. Present BQ tune estimation algorithm [7].

The development of an improved algorithm was prompted after the observation that the tune estimates from the BQ algorithm sporadically jump to adjacent 50 Hz harmonic peaks, thus providing incorrect tune estimates. This erratic estimate is used by the QFB, which drives the currents in the quadrupole magnets in order to maintain a reference tune. Unsurprisingly, the performance of the QFB is affected by the lack of stability and accuracy in the tune estimates. The alternative algorithms try to improve the performance of the tune estimation algorithm by taking into consideration the 50 Hz harmonics into the estimation process [7].

As before, the tune value is assumed to be located at the maximum peak of the spectrum obtained from the BBQ system, however this time, the frequency bins in the immediate vicinity of the 50 Hz harmonics are removed from the spectrum. In this work, a frequency range of 12 Hz with a harmonic frequency as the center was removed. This results in a spectrum with gaps, where only approximately $\frac{2}{3}$ of the frequency bins can be used. The alternative algorithms can estimate the position of the maximum peak in the presence of gaps in the spectrum. Analysis of the results show that the performance is somewhat improved.

Figure 3 shows the distribution of the tune estimation errors obtained from simulated spectra with artificially injected 50 Hz harmonics. This figure also shows the results of the alternative algorithms when being used with specific parameters. Namely the Gaussian Process (GP) fit was used with a Radial Basis Function (RBF) kernel having a length scale of 70 while the Weighted Moving Average (WMA) used a window size of 30 [7].

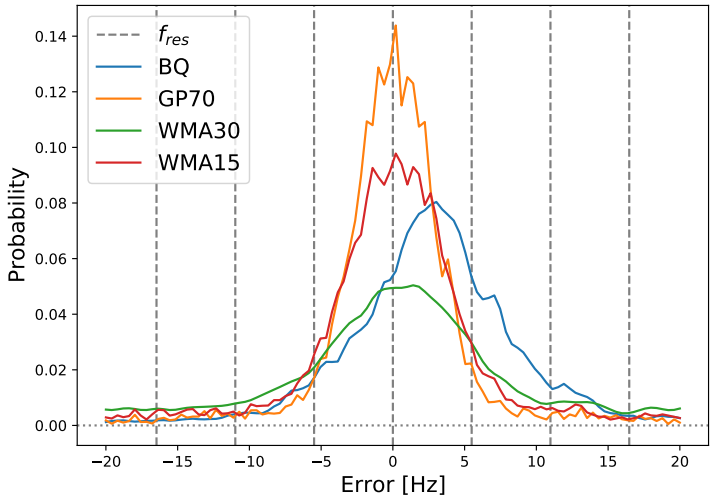

Figure 3. Probability density plot of the errors obtained by non-ML algorithms. The errors are the difference of the respective tune estimates and the ground-truth resonance of he simulated data. BQ is the algorithm used until the LHC Run 2; GP70 uses Gaussian Processes with a Radial Basis Function kernel having a length scale of 70; WMA30 and WMA15 use a weighted moving average with a half-window length of 30 and 15, respectively. GP and WMA were introduced in [7].

3. Simulations

Due to the nature of the tune estimation problem, any BBQ data that has been collected thus far contains the spectra from the BBQ system and tune estimates from the BQ algorithm. Thus when using logged data, it can not be assumed that the logged tune values are correct. Due to this limitation another source of pairs of spectra and tune values had to be obtained, which could be used to train and test the ML approaches. Previous work approached this problem by considering that since the motion of a particle in a circular accelerator can be described by a Hill's type equation, we can approximate the shape of a frequency spectrum obtained by the BBQ system by using a second order system simulation [7].

The first part of this work continues to use simulations, however this time in order to generate a large dataset of spectra and tune value pairs which can be used to train neural network models. Specifically, the frequency spectrum of a second order system is given by the following formula:

$$G(\omega) = \frac{\omega_{res}^2}{\sqrt{(2\omega\omega_{res}\zeta)^2 + (\omega_{res}^2 - \omega^2)^2}} + \mathcal{N}(0, \sigma), \tag{1}$$

where we can obtain ω_{res} by using the following:

$$\omega_{res}^{true} = \sqrt{1 - 2\zeta^2}\omega_{res}. \tag{2}$$

In Equation (1), $\mathcal{N}(0, \sigma)$ denotes an additive Gaussian noise term with zero mean and σ standard deviation while ζ is the damping factor which controls the width of the resonance peak obtained. In addition, a finite value of ζ also shifts the true position of the resonance in $G(\omega)$ according to Equation (2).

All of the ML models considered in this work require a fixed length input, and can only provide a fixed length output. For the models that estimate the tune from a frequency spectrum, it would have been possible to feed the entire spectrum however this would require a model with a large input length, implying a large number of parameters to train. This approach is not necessary since in real operational conditions, the spectral region inside which to find the tune frequency is generally well known.

In this work, the frequency window was chosen to be 100 frequency bins long, while guaranteeing that the tune peak lies within this frequency window. This value was chosen to be slightly larger than the frequency windows chosen during real machine operation using the BQ algorithm. The average operational frequency window obtained from a sample of parameters used in the BBQ system for the beam during FLATTOP is around 80 frequency bins long. It was empirically observed however that sometimes the dominant peak lay close to the edges of the chosen window, which subsequently limits the performance of the BQ algorithm. Figure 4 compares the new window length to that used in operation and as it can be observed, the tune estimates around the 15 s mark are close to edge of the operational window. This example also paints a clearer picture of the bounds on the inputs used throughout this work where we see that in all the cases the input size chosen will always be adequate and representative of real operation.

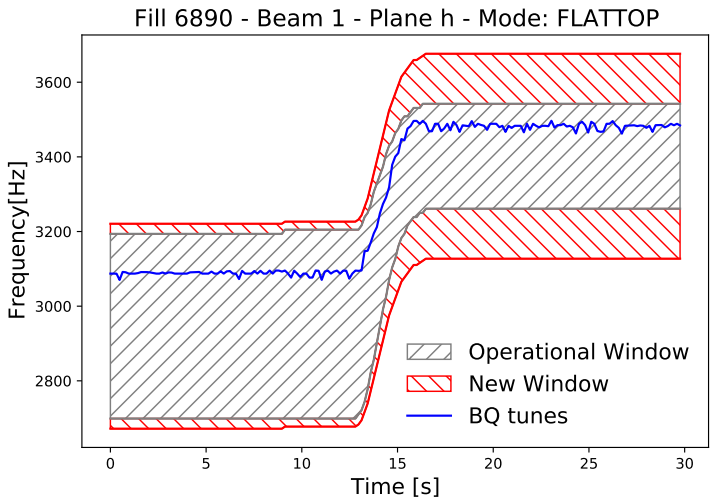

Figure 4. An example of the operational frequency windowing used by the BQ algorithm during the LHC Run 2 to the new window length used in this work. The tunes were obtained from LHC fill 6890, horizontal plane of beam 1 in the FLATTOP beam mode. During FLATTOP the optics are changed and thus the tune shifts to a new frequency.

It is important to note that the absolute magnitudes of the spectra are not needed. For example in Figure 1 the vertical axis is in the range of 160 dB and this is calculated by the BBQ system to be representative of the real power in each frequency bin. For training neural networks it is imperative that we normalise the input data to be either in the range $[0, 1]$ or $[-1, 1]$. This is due to the type of activation functions that are used in between layers in a neural network, which are designed to operate in normalised space. Due to this, the real power of the spectra need not be generated by the simulator. Another important detail is that the value of the tune, while equivalent to ω_{res}^{true}, had to be normalised with respect to the frequency window passed to the model. This is not detrimental to the operation of the model as the choice of the frequency window is chosen by the operators, and the real value of the tune can be easily transformed to Hertz.

By performing a Monte Carlo simulation of the ω_{res}^{true} and ζ required by the second order model as shown in Equation (1), we can explore a myriad of possible spectra, with an exact value of the resonant frequency for each spectrum. ω_{res}^{true} was sampled from the bounds of the frequency window in radians and ζ was sampled from $10^{\mathcal{U}(-2.5,-1.8)}$ where $\mathcal{U}$ is a uniform distribution. The normalised amplitude of the injected 50 Hz harmonics was drawn from $\mathcal{U}(0.5, 2)$ and after adding the harmonics to the second order spectrum, a simple linear digital filter of size 3 was passed forward and backward to the spectrum in order to give width to the harmonics as can be observed in Figure 1. The spectrum is then normalised again, and ω_{res}^{true} is found in terms of the normalised frequency range. Hence

by using this generated dataset, we can expect the ANN to generalise well and provide a robust tool which can reliably estimate the tune even from a BBQ spectrum directly.

Figure 5 illustrates a normalised real spectrum clipped to the relevant frequency window, along with a second order simulation. The procedure described above was iteratively performed to locate suitable parameters for the simulated spectrum of $norm(\omega_{res}^{true}) = 0.76$, $\zeta = 10^{-2.6}$ and $\sigma = 0.04$. As can be observed, the shape of the simulated tune peak matches with that observed in real BBQ spectra.

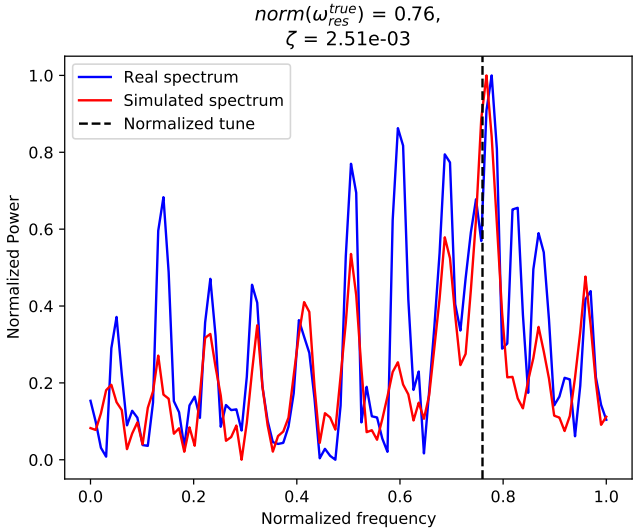

Figure 5. Second order simulation of a real spectrum obtained from the BBQ system.

4. Simple Approach

The simplest way to use ML tools to solve the tune estimation problem is by using an Artificial Neural Network (ANN) that is trained to estimate the tune value from any BBQ spectrum. The idea would be to first choose a sensible network architecture and then train it using the real BBQ spectra as the input. The tune estimates obtained by the most reliable algorithm mentioned in Section 2 would be used as the labels. However, when using this approach the ANN can never predict better tune estimates than the algorithm that produced the tune label. It is preferable to train the network with simulated second-order system spectra, since the resonant frequency used to create the spectrum is known.

In the simple approach, the pairs of simulated spectra and normalised resonant frequencies were generated according to the range of parameters described in Section 3. The training dataset consisted of 20,000 batches of 32 samples each. Another 1000 samples were generated and used as validation data. For all models described in this section, the loss function used was the Mean Squared Error (MSE) and the gradient descent algorithm was Stochastic Gradient Descent (SGD) with a learning rate of 0.01.

4.1. Fully-Connected Layers

The first network architecture that was considered was a 3-layer, fully-connected network (also called a dense network). In all the models attempted, the input layer had a size of 100 and the output layer had a size of 1. The activation function of all hidden layers was the Rectified Linear Unit (ReLU) and the output layer was linear. The Glorot normal initialiser was used for all layers.

Table 1 shows the parameters of the three model architectures attempted. Figure 6 shows the density plots of the errors of the tune estimates provided by each respective model. Here it can be seen that ML#1 and ML#2 have the best performance, with the highest accuracy and precision. It can also be noted that even though ML#2 has almost triple the number of parameters of ML#1, the latter still performed as good. From Figure 6 it can also be seen that unlike the BQ algorithm and the alternative algorithms proposed in [7]

(GP70, WMA30), the density plot of the three models drops to zero after approximately 2 frequency bins. This contrasts with the current and the alternative algorithms which show a tail extending beyond the bounds of Figure 3.

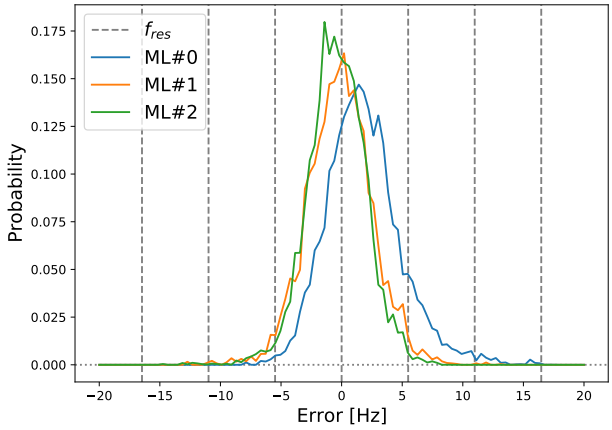

Figure 6. Probability density of the errors obtained by fully-connected networks. The errors are the differences between the predicted tunes and the resonances used to simulate the spectra.

Table 1. Model architectures presented for fully-connected networks.

	Layer 1	Layer 2	Layer 3	# [1]
ML#0	150	50	10	23,221
ML#1	300	100	20	62,441
ML#2	500	250	50	188,351

[1] Number of trainable parameters.

4.2. Convolutional Layers

Convolutional layers were invented in order to more efficiently solve tasks whose data is known to have a grid-like topology where spatial locality exists. They work on the same principle as the visual cortex of mammals, which is to collect subsets of the input (such as raw pixels in an image) and processing each subset independently of each other. Convolutional layers can be used on the output of previous layers in order to capture more complex features [8,9].

Convolutional layers use kernels (also called filters) to perform the convolution operation, where a kernel is parametrised by a set of weights. Each kernel is convolved with a small subset of the input to produce a feature which is then placed in a feature map, all the while maintaining the spatial order of the features with respect to the original data. When this operation is done, the kernel is then shifted to the next subset of the input to produce a new feature. Note that the length of each shift of the kernel is also called stride length. An important advantage of using convolutional layers is the significantly reduced number of parameters needed to achieve the same performance as an equivalent in function, fully-connected network [8].

Figure 7 illustrates a network architecture utilising three convolutional layers and one dense layer to produce a scalar output. This architecture was trained under various configurations of parameters to try and solve the tune estimation problem. The activation function of all hidden layers was ReLU and the output layer was linear. Table 2 lists the model architectures using convolutional layers. Similar to Figure 6, Figure 8 shows the probability density of the errors obtained by the various convolutional network architectures attempted. From this plot, it can be seen that the best convolutional model is ML#5.

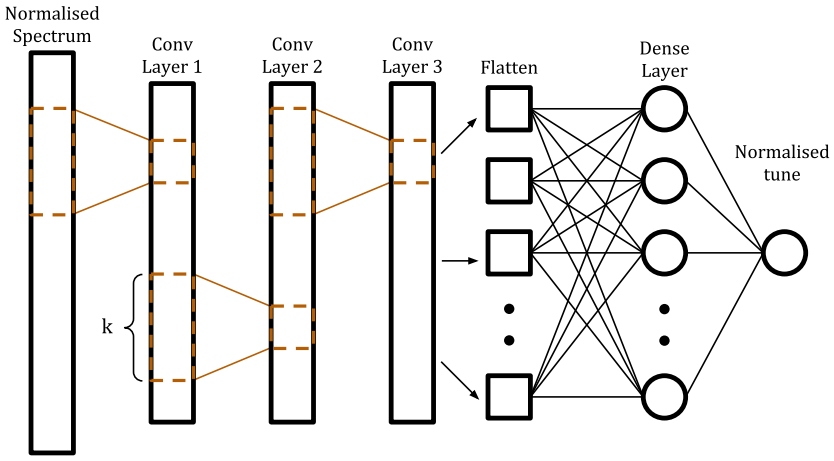

Figure 7. Network using convolutional layers.

Table 2. Model architectures presented for CNNs.

	Layer 1			Layer 2			Layer 3			Dense	# [4]
	f [1]	k [2]	s [3]	f	k	s	f	k	s		
ML#3	32	3	3	16	3	3	8	3	3	20	2753
ML#4	32	3	1	16	3	1	8	3	1	20	18,113
ML#5	64	3	3	32	3	1	16	3	1	20	18,905
ML#6	128	3	3	64	3	3	16	3	3	20	29,561
ML#7	64	3	1	32	3	1	16	3	1	20	40,025

[1] Number of filters. [2] Kernel size of convolution. [3] Stride length, shift size of kernel. [4] Number of trainable parameters.

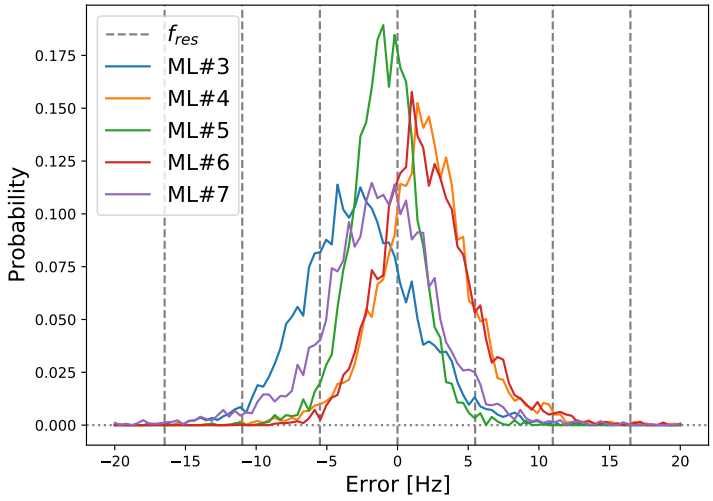

Figure 8. Probability density of the errors obtained by convolutional networks. The errors are the differences between the predicted tunes and the resonances used to simulate the spectra.

4.3. Evaluation

A set of 5000 simulated spectra were used to evaluate and compare the performance of ML#1 and ML#5 to that of BQ, WMA and GP. The tune estimates obtained from each respective model and algorithm were subtracted from the ground-truth resonances used to generate the spectra to obtain a set of errors per tune estimation system. Figure 9 shows the probability density of these errors in log scale. The vertical dashed lines are centered around 0 and are separated by the spectral resolution, f_{res}. The vertical lines give an

indication of the scale of the errors with respect to the available resolution of the spectra. Similar to the density plots shown in Figures 3, 6, 8 and 9 compares the accuracy and precision of the best DNN model (ML#1), the best CNN model (ML#5), the WMA using two different half-window lengths of 15 and 30, GP with an RBF kernel having a length scale of 70 and the BQ algorithm.

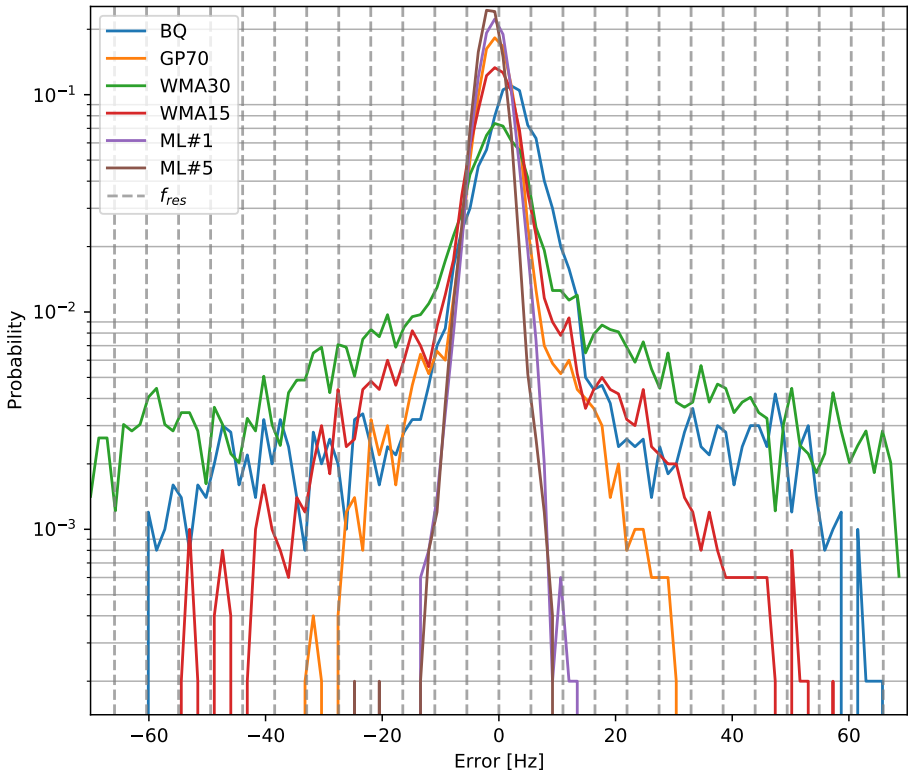

Figure 9. Error distribution of tune estimates of trained ML models and algorithmic approaches to the ground-truth resonance. ML#1 and ML#5 are the best DNN and CNN respectively; BQ is the algorithm used in the LHC until Run 2; GP70 uses Gaussian Processes with a RBF kernel having a length scale of 70; WMA15 and WMA30 use a Weighted Moving Average with half window lengths of 15 and 30 respectively.

From Figure 9 it can be observed that when using simulated spectra, the probability of both ML models to obtain an absolute error larger than $3f_{res}$ is practically zero. In contrast, all the other tune estimation algorithms presented exhibit a non-zero probability of having an absolute error larger than $3f_{res}$. In particular, the probability of the ML models obtaining an error of approximately $-2.5f_{res}$, is one order of magnitude lower than all the other tune estimation algorithms.

4.4. Limitations

Training a neural network to solve a supervised task requires a large set of correctly labelled data. The problem of inaccurate tune estimates from real spectra was avoided by simulating second-order system spectra, artificially injecting 50 Hz noise harmonics and Gaussian noise in order to mimic what is observed in a BBQ spectrum. However it was observed that regardless of the model architecture used, the performance of the model trained on simulated spectra is sub-optimal in estimating the tune from a real spectrum (Section 6 illustrates the sub-optimal tune estimates obtained by ML#1 and ML#5 when using BBQ spectra instead of simulated spectra).

Figure 10 shows the training and validation losses of ML#1. Validation losses are obtained by comparing the tune estimates in the validation dataset to the predicted tune

estimate of the network. The figure shows that the training loss and the validation loss are very similar when using unseen simulated spectra for validation. This indicates that the network is successfully learning to predict the tune values from a simulated spectrum.

The problem with the simple approach is exposed when using a validation dataset composed of real spectra as inputs and tune estimates obtained from BQ, GP70 and WMA30, respectively as the labels. A gap of approximately an order of magnitude between the training and validation loss from BQ can be observed. Normally when training a neural network, such a gap between the two losses is attributed to either over-fitting or under-fitting of the model over the training data. However in this case, this discrepancy in the losses was expected since it is known that the algorithms used to obtain the validation labels are not reliable sources of tune estimates.

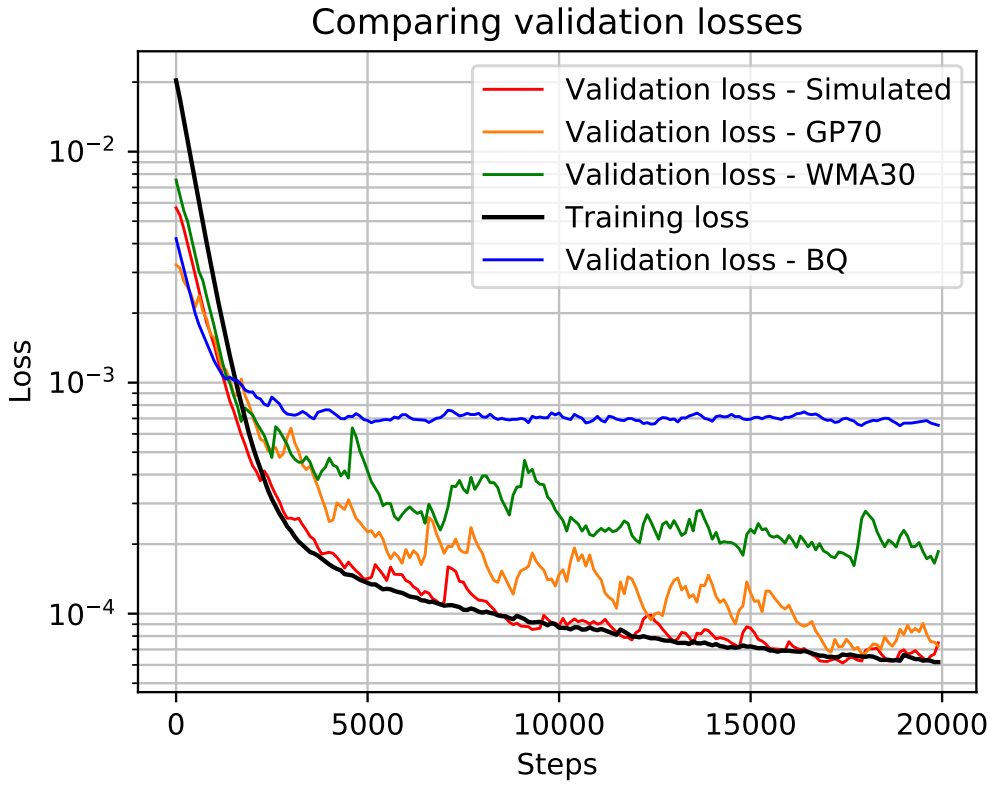

Figure 10. Comparison of validation losses when using simulated and real spectra respectively. Note that the dataset containing the real spectra was used to obtain tune estimates from BQ, GP70 and WMA30, thus creating 3 separate validation datasets.

From Figure 3, the average error of the BQ algorithm can be estimated to be half a frequency bin which implies that on a normalised frequency window, the average inherent loss from the BQ tunes would be approximately $\frac{0.5\,\text{bin}}{100\,\text{bins}} \approx 5 \times 10^{-3}$. This would explain the size of the gap between the training and the BQ validation loss in Figure 10, however it might not be the only contribution to producing said gap. Another possible contribution could stem from the fact that the models trained so far are over-fit to some features only present in simulated spectra, which would explain the sub-optimal performance on real data. Following this hypothesis, another source of realistic training data was needed to train a better ML model.

5. Improving the Dataset

A technique introduced in [10], called SimGAN, was considered a potential solution to the above problem. SimGANs build upon the work done in [11], which introduced the Generative Adversarial Network (GAN). The goal of the GAN architecture (Figure 11) is to train a *Generator Network* (generator) to transform a random input, into an image which

looks similar to the *Real Images* dataset. Therefore the generator needs to capture the data distribution of the Real Images dataset whilst the *Discriminator Network* (discriminator) evaluates the probability that the image came from the Real Images dataset and not from the generator (fake). During training, the discriminator loss is used to update the discriminator via supervised learning where the input is either a real or a fake image and each label is 1 or 0 respectively. Both the generator loss and the discriminator loss are used to update the parameters of the generator in the direction which maximises the probability of the discriminator making a mistake in classifying its input.

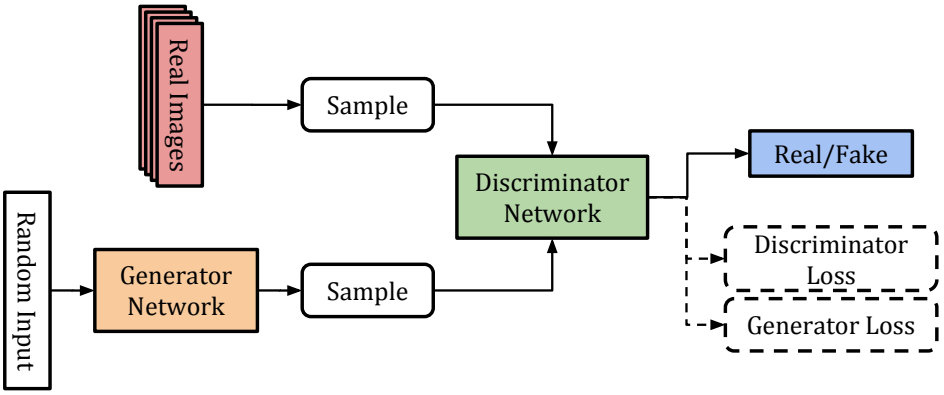

Figure 11. Overview of the Generative Adversarial Network architecture.

5.1. SimGAN

[H] SimGAN is a modified version of the GAN, where the goal is to refine a synthetic or simulated image to look more realistic [10]. Since the role of the GAN generator is now changed to refine an input synthetic image, SimGAN nomenclature refers to the generator network as a *Refiner Network* (R). Figure 12 illustrates the architecture of the SimGAN used in this work. Two notable differences which distinguish SimGANs from GANs are—the input of the refiner is now coming from a simulator; the addition of a regularisation loss to the refiner loss, which restricts the output of the network to remain similar to the input. The discriminator is continuously trained to distinguish between refined and real spectra. This ultimately helps the refiner to create more realistic spectra in an adversarial manner.

SimGANs were originally presented to work with images via 2D convolutional networks. Since the aim is to refine simulated spectra to look more like spectra obtained from the BBQ system, all the networks in Figure 12 use 1D convolutional networks. The theory presented below remains valid since the same losses are minimised, regardless of the input shape and the network type.

The goal is to use a set of unlabelled real data, $y_i \in \mathcal{Y}$, to learn a refiner network, $R_\theta(x)$ that refines simulated data x, with θ as the function parameters. Therefore we can define refined data, $\tilde{x}$ as:

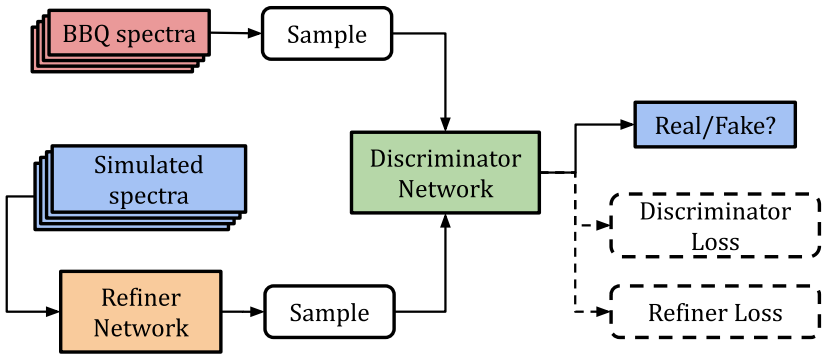

Figure 12. Overview of the SimGAN architecture.

$$\tilde{x} := R_\theta(x)$$

The key requirement is that $\tilde{x}$ should look similar in appearance to the real data in the set $\mathcal{Y}$, while still preserving the annotation information from the simulator. The training procedure of θ involves the minimisation of a combination of two losses:

$$\mathcal{L}_R(\theta) \quad = \quad \sum_i \ell_{real}(\theta; x_i, \mathcal{Y}) \quad + \quad \lambda \ell_{reg}(\theta; x_i), \tag{3}$$

where x_i is the i^{th} simulated spectrum. ℓ_{real} adds realism and ℓ_{reg} regulates the preservation of the annotation information of the input. ℓ_{reg} is scaled by a scalar $\lambda > 0$, which balances the effect of regularisation over realism. Since a refiner makes it difficult to classify spectra as real or refined, an adversarial discriminator D_ϕ is trained to classify spectra as real or refined. Thus the output of the discriminator can be considered as the probability that the input spectrum is real on a scale $[0, 1]$. The discriminator updates ϕ by minimising the following cross-entropy loss:

$$\mathcal{L}_D(\phi) \; = \; -\sum_i \log(D_\phi(\tilde{x}_i)) \quad - \quad \sum_j \log(1 - D_\phi(y_j)). \tag{4}$$

Note that ℓ_{real} can be formed by using D_ϕ to update θ:

$$\ell_{real}(\theta; x_i, \mathcal{Y}) \quad = \quad -\log\big(1 - D_\phi(R_\theta(x_i))\big). \tag{5}$$

By minimising Equation (5), θ is updated in the direction that makes D_ϕ classify refined spectra as real, thus improving the performance of the refiner by using the discriminator as a metric. Apart from this, ℓ_{reg} is introduced as a self-regularisation measure to preserve the annotation information of the input:

$$\ell_{reg} \quad = \quad ||\psi(\tilde{x} - x)||_1,$$

where $\psi(\cdot)$ modifies how the regularisation loss is obtained. For example ψ could be an identity matrix which maps the input to itself. In this work, ψ was configured to satisfy an empirical observation that the 50 Hz noise harmonics in real spectra are always additive artefacts and never manifest as dips. Due to this observation $\psi(\cdot)$ was designed as shown in Equation (6) so that any artefacts introduced below the baseline have a higher cost on the model for $\eta > 1$.

$$\begin{aligned} e &\triangleq \tilde{x} - x \\ \ell_{reg} &= \sum_i \max(0,\, e_i) \; + \; \eta \sum_i \max(0,\, -e_i). \end{aligned} \tag{6}$$

5.2. Training SimGAN on Simulated and BBQ Spectra

The second-order spectrum simulator that was used to create the spectra to train the models in the Simple Approach was modified to not inject the 50 Hz noise harmonics. Instead the simulator would now only create a spectrum from Equation (1) and add Gaussian noise. Note that the effect of noise on the input is twofold: It helps the refiner to generalise the refinement process, as in any update process utilising backward propagation [12]; It is the only source of randomness which the refiner can use to generate somewhat different artefacts in its output.

The real dataset, $\mathcal{Y}$, used in Equation (4) was obtained from the BBQ data logged during Fill 6890, Beam 1, horizontal plane, during FLATTOP. Since the input size of all networks considered in this work was 100, each real spectrum was truncated by a frequency window having a randomised position, while guaranteeing that the dominant peak of the spectrum lies within said window. This was done to ensure that the discriminator does not over-fit to the dominant peak always occurring at the same relative location with respect

to the frequency window. This ensures stabler and more generalised training of all the constituent components of SimGAN.

The network architectures used for both the refiner and discriminator are a 1D version of the networks used in [10]. All convolutional layers used were 1D and unless otherwise specified, all stride lengths were 1. The refiner was a Residual Network (ResNet) with an input size of 100 which was fed to a first convolutional layer having 64 filters and a kernel size of 3. The output of this layer was fed to 4 ResNet blocks connected in sequence. Each block consisted of two convolutional layers having 64 filters each with a kernel size of 3 and a connection merging the input of the block with the output of the block's second convolutional layer. The output of the last ResNet block was then fed to a final convolutional layer with 1 filter having a kernel size of 1, producing an output of size 100. All hidden layers of the refiner used the ReLU activation function and the output was linear. The discriminator was a 1D convolutional neural network with an input layer size of 100 which was fed to a convolutional layer having 96 filters with a kernel size of 3 and a stride length of 2, followed by another convolutional layer having 64 filters with a kernel size of 3 and a stride length of 2. Following the second hidden layer, Max pooling was applied with a pool size of 3. The output of the pooling was fed to a convolutional layer having 32 filters, kernel size of 3 and a stride length of 3 followed by two convolutional layers having 16 and 2 filters respectively and a kernel size of 1. All hidden layers of the discriminator used the ReLU activation function and the last layer had an output size of 2 with the softmax activation function. The two outputs of the discriminator correspond to the probabilities that the input spectrum is real or refined respectively. All layer weights were initialised using the Glorot normal initialiser.

The initial value of η in Equation (6) was set to 5 and the initial value of λ in Equation (3) was set to $1 \times 10^{-3.8}$. These values were found empirically to achieve the best balance between realism and preservation of the annotation information from the simulated spectra to the refined spectra. During the training of one SimGAN it was observed that it is possible for a refiner to learn to add valid artefacts which are able to trick its respective discriminator into classifying the refined spectra as real. However the ultimate goal is to generate spectra with enough variety in their shapes to reliably train a tune estimation model. Equation (3) only constrains the refiner network to make simulated spectra look similar in appearance to spectra obtained from the BBQ system. An individual refiner is not incentivised to learn different types of artefacts and in fact using one refiner to generate realistic spectra caused the tune estimation model to over-fit to the type of artefacts that one refiner produced. To overcome this problem, 500 SimGANs were trained while using values of λ sampled from $10^{\mathcal{U}(-4,-3.5)}$ in order to obtain refiners which behave somewhat differently from each other. In turn this allowed the tune estimation model to become more generalised and perform better over the real spectra. Results from randomly selected refiners can be seen in Figure 13. It can be seen that from a baseline simulated spectrum (red), a trained refiner is able to add artefacts to create a refined spectrum (blue). The refined spectra look similar to real BBQ spectra, such as the one shown in Figure 1. This set of trained refiners was used to create a refined training dataset which was used to train a new tune estimation model (ML-Refined) having the same architecture as ML#1.

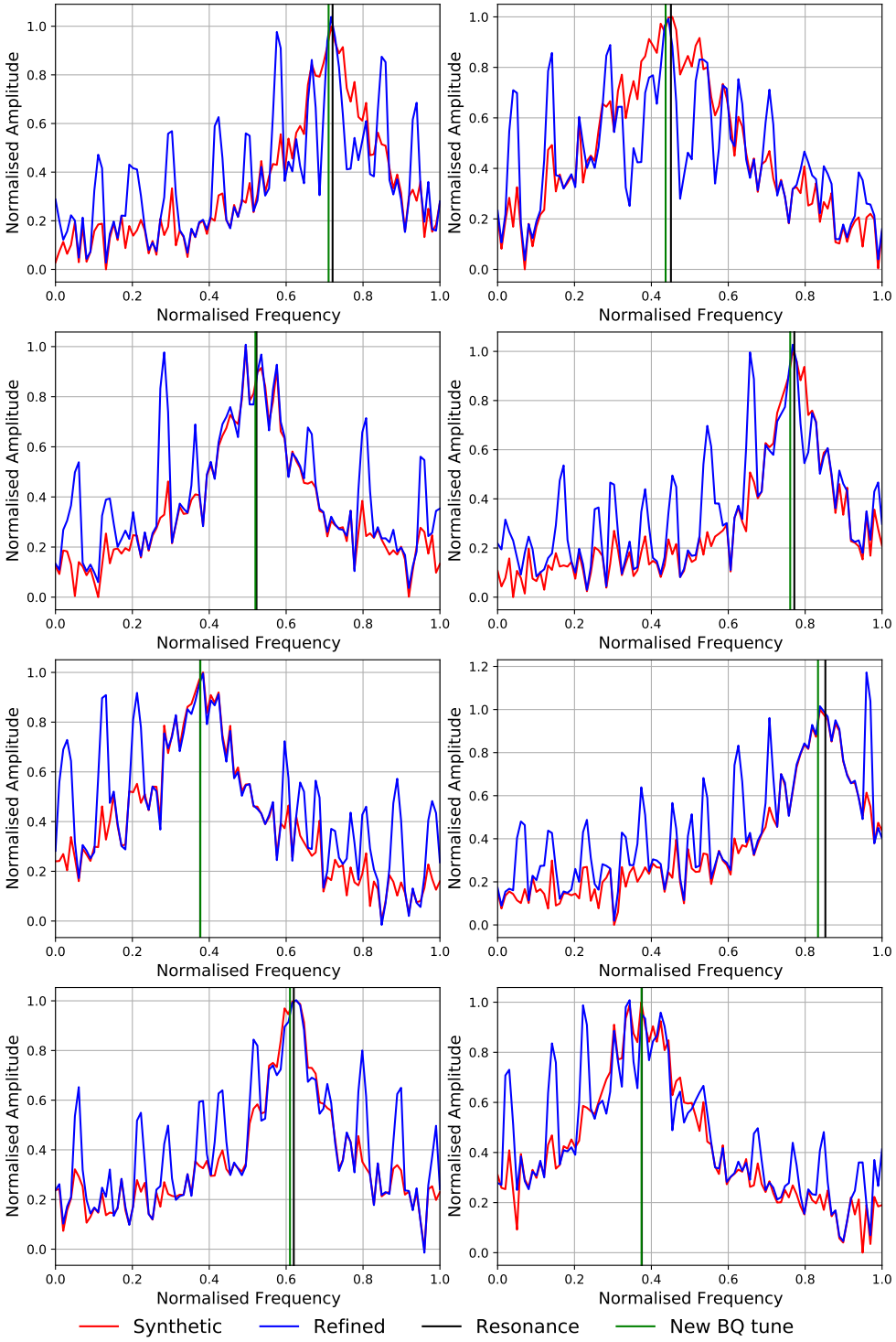

Figure 13. Results from a trained SimGAN. The green line represents the tune estimate from BQ algorithm on the refined spectrum.

6. Discussion

From real BBQ spectra it is difficult to show any objective difference between the performance of the new ML tune estimation models and the various algorithmic approaches (BQ, WMA and GP). The main reason is the absence of the tune ground-truth value in real spectra. Therefore, the accuracy and precision of any tune estimation system can only be assessed visually by superimposing the tune estimates over the real spectra.

Two sets of spectra obtained in the LHC Run 2 during the transition from PRERAMP to RAMP and from RAMP to FLATTOP were used to evaluate the performance of the tune estimation systems. It was made certain that the spectra used during evaluation were not used during the SimGAN training in order to remove any bias. Specifically, BBQ spectra from Fill 6768, Beam 1, horizontal plane were used. A tune evolution estimate was calculated from each spectrum by BQ, GP70, WMA15, ML#1, ML#5 and ML-Refined. The spectra were converted to heat maps by filtering out the 50 Hz harmonics and smoothing in time and frequency axes. Moving averages (μ) and moving standard deviations (σ) of the tune estimates were performed using a centered window. Subsequently the corresponding μ and 3σ were superimposed on the heat maps to obtain Figures 14 and 15.

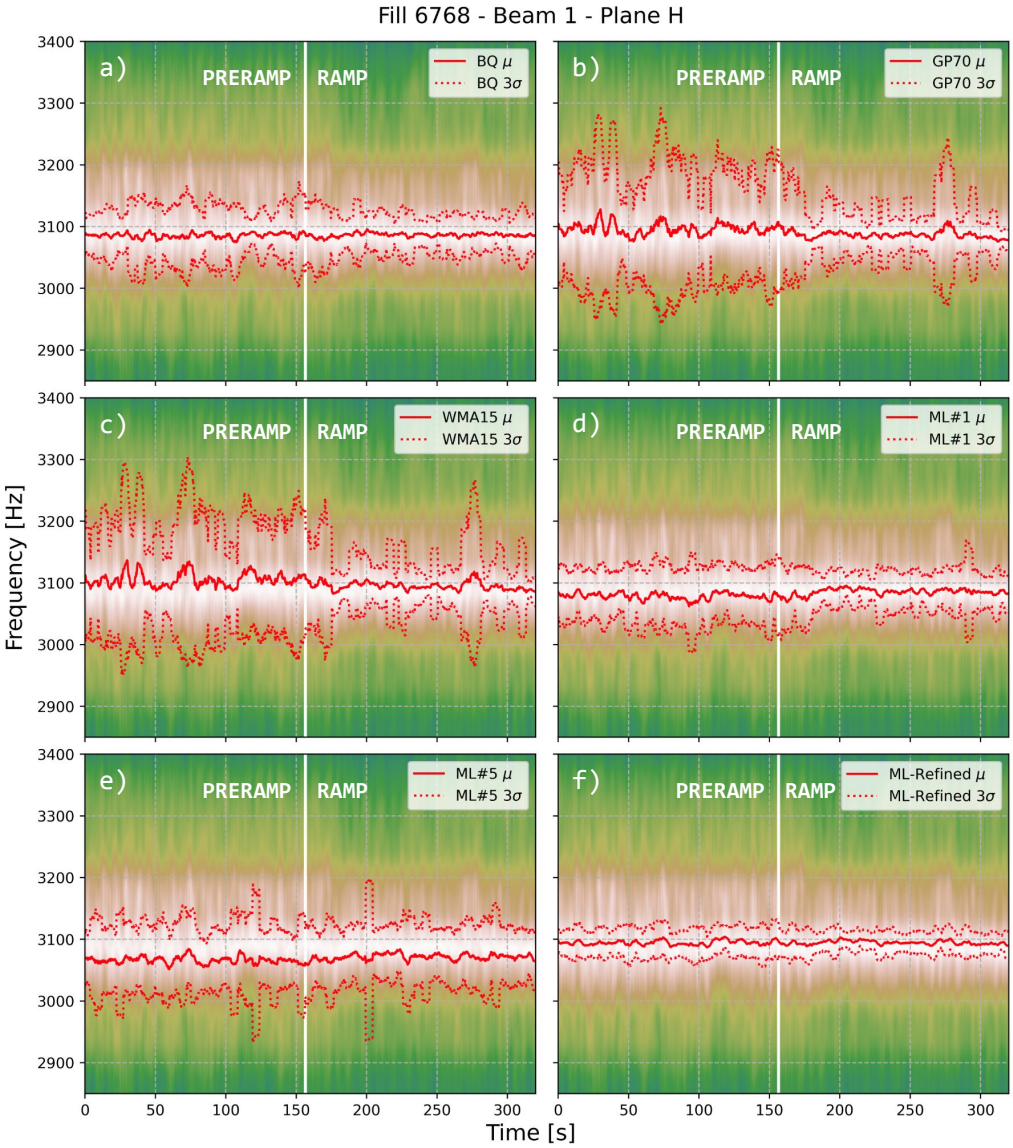

Figure 14. (Background) Heat map obtained by post-processing BBQ spectra from LHC Fill 6768, Beam 1, Horizontal plane, in the transition from PRERAMP to RAMP. (Foreground) Superimposed is the mean, μ, and scaled standard deviation, 3σ, of the tune evolution as estimated by different tune estimation algorithms and ML models. (**a**) The original algorithm used in the LHC until Run 2 (BQ) (**b**) Tune estimation using Gaussian Processes with an RBF kernel having a length scale of 70 (GP70) [7] (**c**) Tune estimation using a Weighted Moving Average with a window half length of 15 (WMA15) [7] (**d**) The best DNN model trained using simulated spectra (ML#1) (**e**) The best 1D CNN model trained using simulated spectra (ML#5) (**f**) DNN model with the same architecture as ML#1 and trained using spectra refined by SimGAN.

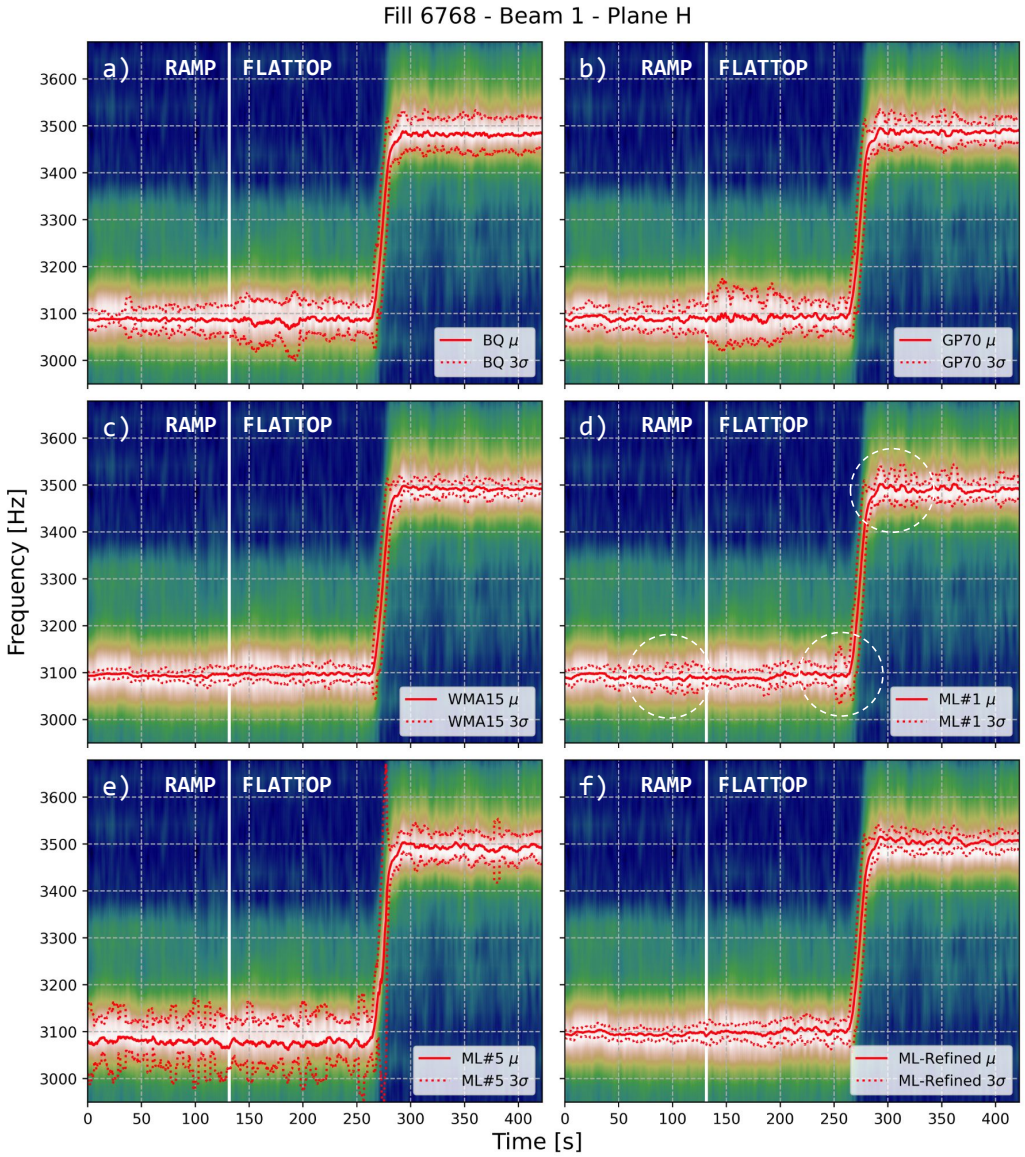

Figure 15. (Background) Heat map obtained by post-processing BBQ spectra from LHC Fill 6768, Beam 1, Horizontal plane, in the transition from RAMP to FLATTOP. (Foreground) Superimposed is the mean, μ, and scaled standard deviation, 3σ, of the tune evolution as estimated by different tune estimation algorithms and ML models. Three regions of interest are also marked (dashed white circles) (**a**) The original algorithm used in the LHC until Run 2 (BQ) (**b**) Tune estimation using Gaussian Processes with an RBF kernel having a length scale of 70 (GP70) [7] (**c**) Tune estimation using a Weighted Moving Average with a window half length of 15 (WMA15) [7] (**d**) The best DNN model trained using simulated spectra (ML#1) (**e**) The best 1D CNN model trained using simulated spectra (ML#5) (**f**) DNN model with the same architecture as ML#1 and trained using spectra refined by SimGAN.

Figure 8 shows that ML#5 performed well over simulated data however, Figures 14e and 15e show that the performance of ML#5 is worse than that of the BQ algorithm on real spectra. Figure 14a,d show that ML#1 performs just as well as the BQ algorithm. From Figure 14f it can be seen that ML-Refined produces the most stable tune estimates during the transition from PRERAMP to RAMP. Three regions of interest have been marked in Figure 15d (white dashed circles). In comparison with the same regions of Figure 15f, the improvement of ML-Refined over ML#1 becomes evident.

As mentioned in the Introduction, the tune estimates are used in the QFB to control the quadrupoles and correct the tune of the LHC. In order to ensure a stable operation, the QFB measures the stability of the tune estimates with the following formulas:

$$\begin{aligned} \Delta q_t &= q_t - q_{t-1} \\ S_t &= S_{t-1} * (1 - \alpha) + \Delta q_t * \alpha, \end{aligned} \tag{7}$$

where q_t is the tune estimate at time t, α is the time constant of the exponential moving average and S_t is the stability measure at time t. The evaluation of both the long and short term tune estimate stability is done via two independent stability measures with a different α. Each α corresponds to a fast (2 s) and slow (10 s) time constant. To ensure a stable tune estimate, both stabilities are required be below a certain threshold, which is by default 0.005.

The stability metric shown in Equation (7) was used to quantitatively assess the performance of the trained models with regard to the requirements of the QFB. Figure 16 shows the stability probability distribution of each model and algorithm used in this work. Tune estimates were obtained from the unseen BBQ spectra of Fill 6768, beam 1, horizontal plane, from PRERAMP to FLATTOP and amounting to a total of approximately 10,000 real BBQ spectra. The blue and orange histograms correspond to the fast and slow stability measurements, respectively. ML#5 is shown to provide the most unstable tune estimates. GP70 and WMA15 exhibit a wide distribution of both fast and slow stabilities indicating that in real operation the risk of the QFB switching off is high. ML#1 is shown to be stabler, with only the fast stability exceeding the threshold. Surprisingly the BQ algorithm is shown to be the second most stable algorithm. Upon examination of its tune estimates it can be seen that the effect of the 50 Hz harmonics may have an impact due to the tune estimates being stuck on the position of noise harmonics close to the tune peak. Finally ML-Refined is shown to provide the most stable tune estimate with a zero probability of the QFB switching off.

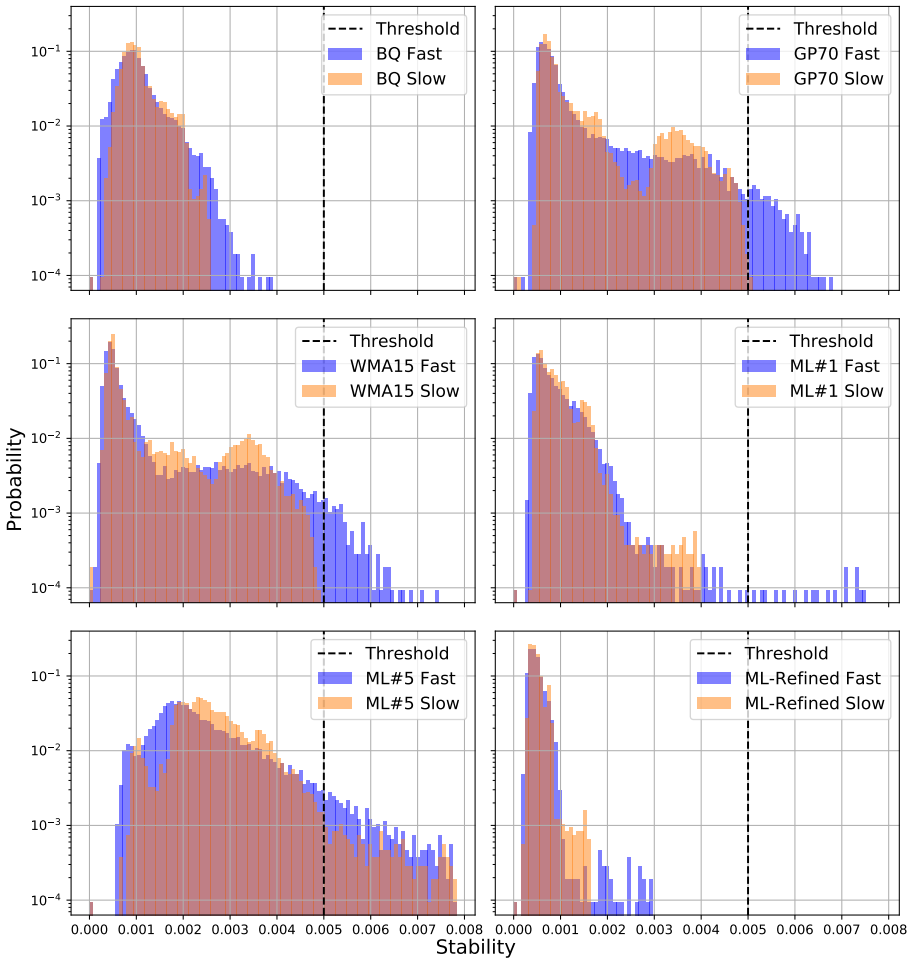

Figure 16. Probability distribution of the tune estimation stability. Obtained from Fill 6768, beam 1, horizontal plane using spectra from PRERAMP to FLATTOP. Slow and Fast correspond to stability measures having time constants of 10 s and 2 s respectively (Equation (7)). The threshold was chosen in the early design of the beam-based feedback systems and used in operation during the LHC Run 2. BQ was the original tune estimation algorithm used until the LHC Run 2; GP70 uses Gaussian Processes with a RBF kernel of length scale 70; WMA15 uses a Weighted Moving Average with a half-window length 15; ML#1 and ML#5 were the best performing DNN and CNN networks respectively trained with simulated spectra; ML-Refined had the same architecture as ML#1 and trained with refined spectra.

7. Conclusions

Several ML tools, namely DNNs, CNNs and SimGANs, were used to train tune estimation models. Since the exact tune values for the BBQ spectra were not available, simulated spectra from a second order system were used to realistically model the beam response and to create a simulated dataset covering a wide range of tune peak positions and shapes. Different model architectures were trained using this simulated data and the best model architectures for tune estimation were chosen. Figures 6 and 8 were used to choose the best DNN and CNN models.

It was found that when models were trained by a dataset comprised only of simulated spectra, there was a discrepancy in their performance between simulated and real BBQ spectra. As a solution to this problem, SimGANs were used to refine a simulated dataset, by using real unlabelled data to help with the refinement training. A set of 500 SimGANs were trained with random initial conditions which were then used to generate a refined dataset of spectra with correctly labelled tune values. The refined dataset was then used

to train a DNN tune estimation model, having the same network architecture as the best performing model trained by simulated spectra.

Finally, a sample of unseen real data was used to compare the performance of the best performing tune estimation models and the classical algorithmic approaches. The accuracy, precision and stability of all the tune estimation systems considered in this work were analysed and compared by using Figures 14–16. It was shown that the model trained by the refined dataset of simulated spectra obtained the best performance, which implies a more reliable tune estimate source for the QFB.

Author Contributions: Conceptualisation, L.G., G.V.; methodology, L.G., G.V. and D.A.; software, L.G.; validation, L.G., G.V. and D.A.; formal analysis, L.G.; investigation, L.G.; resources, L.G. and G.V.; data curation, L.G.; writing—original draft preparation, L.G.; writing—review and editing, L.G., G.V. and D.A.; visualisation, L.G.; supervision, G.V. and D.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was carried out in association with CERN.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: We would like to thank Thibaut Lefevre, Manuel Gonzalez Berges, Stephen Jackson, Marek Gasior and Tom Levens for their contributions in acquiring resources and reviewing the results obtained in this work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Baird, S. *Accelerators for Pedestrians*; Technical Report AB-Note-2007-014. CERN-AB-Note-2007-014. PS-OP-Note-95-17-Rev-2. CERN-PS-OP-Note-95-17-Rev-2; CERN: Meyrin, Switzerland, 2007.
2. Steinhagen, R.J. LHC Beam Stability and Feedback Control-Orbit and Energy. Ph.D. Thesis, RWTH Aachen University. 2007. Available online: https://cds.cern.ch/record/1054826 (accessed on 29 April 2021).
3. Gasior, M.; Jones, R. High sensitivity tune measurement by direct diode detection. In Proceedings of the 7th European Workshop on Beam Diagnostics and Instrumentation for Particle Accelerators (DIPAC 2005), Lyons, France, 6–8 June 2005; p. 4.
4. Gasior, M. Tuning the LHC. *BE Newsletter* **2012**, *4*, 5–6.
5. Alemany, R.; Lamont, M.; Page, S. *Functional specification—LHC Modes*; Technical Report LHC-OP-ES-0005; CERN: Meyrin, Switzerland, 2007.
6. Kostoglou, S.; Arduini, G.; Papaphilippou, Y.; Sterbini, G.; Intelisano, L. Origin of the 50 Hz harmonics in the transverse beam spectrum of the Large Hadron Collider. *Phys. Rev. Accel. Beams* **2021**, *24*, 034001. [CrossRef]
7. Grech, L.; Valentino, G.; Alves, D.; Gasior, M.; Jackson, S.; Jones, R.; Levens, T.; Wenninger, J. An Alternative Processing Algorithm for the Tune Measurement System in the LHC. In Proceedings of the 9th International Beam Instrumentation Conference (IBIC 2020), Santos, Brazil, 14–18 September 2020.
8. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016. Available online: http://www.deeplearningbook.org (accessed on 29 April 2021).
9. Lindsay, G.W. Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future. *J. Cogn. Neurosci.* **2020**, 1–15. [CrossRef] [PubMed]
10. Shrivastava, A.; Pfister, T.; Tuzel, O.; Susskind, J.; Wang, W.; Webb, R. Learning from simulated and unsupervised images through adversarial training. In Proceedings of the IEEE conference on computer vision and pattern recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2107–2116.
11. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. In Proceedings of the Advances in Neural Information Processing Systems 27, Montreal, QC, Canada, 8–13 December 2014; pp. 2672–2680.
12. Matsuoka, K. Noise injection into inputs in back-propagation learning. *IEEE Trans. Syst. Man Cybern.* **1992**, *22*, 436–440. [CrossRef]