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, Msida, Malta
[2] Accelerator Systems Department, CERN, Geneva, Switzerland
* 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

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 is 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].

*+

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. 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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0605.v1

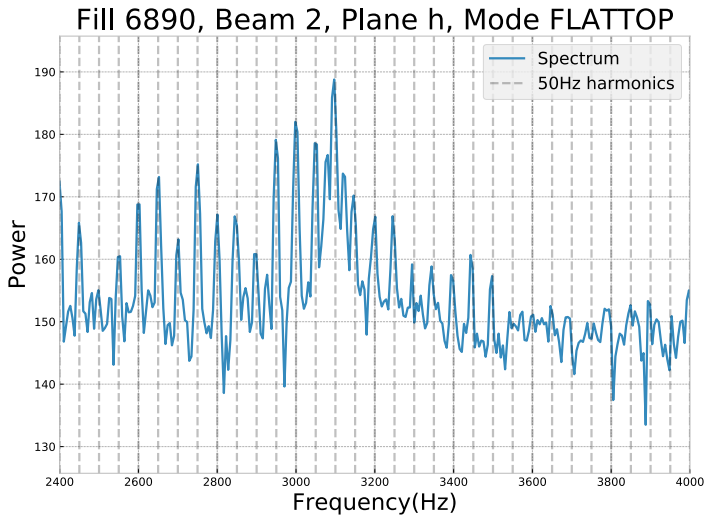

Figure 1. Example of 50 Hz harmonics present in the BBQ spectrum

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.

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 [5]. 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 whose purpose in this work is purely historical. We also briefly explain the alternative algorithms [5] 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.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0605.v1

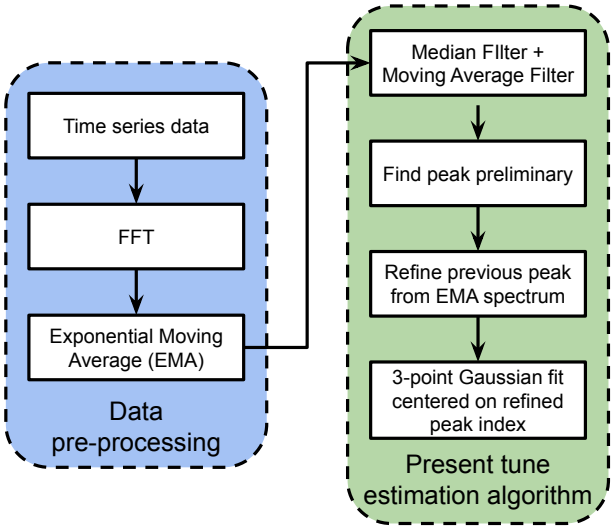

Figure 2. Present BQ tune estimation algorithm [5].

We start by considering the BQ algorithm, where the set of sequential processing blocks which 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.

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 [5].

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 [5].

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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0605.v1

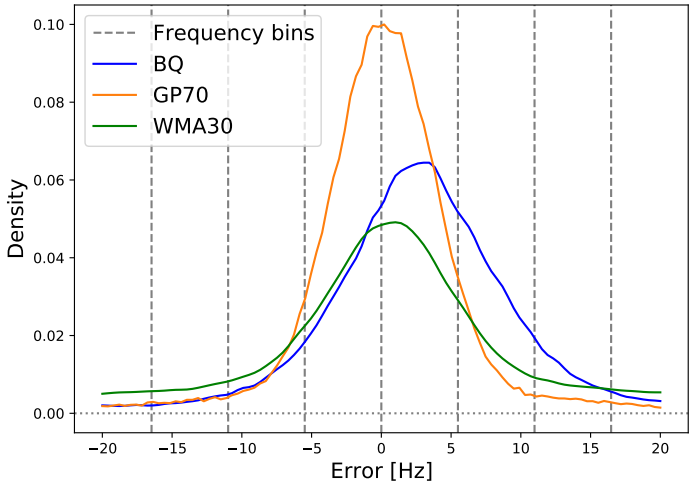

Figure 3. Density plot of the errors of the tune estimates of simulated data from non-ML algorithms

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 [5].

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 which 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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0605.v1

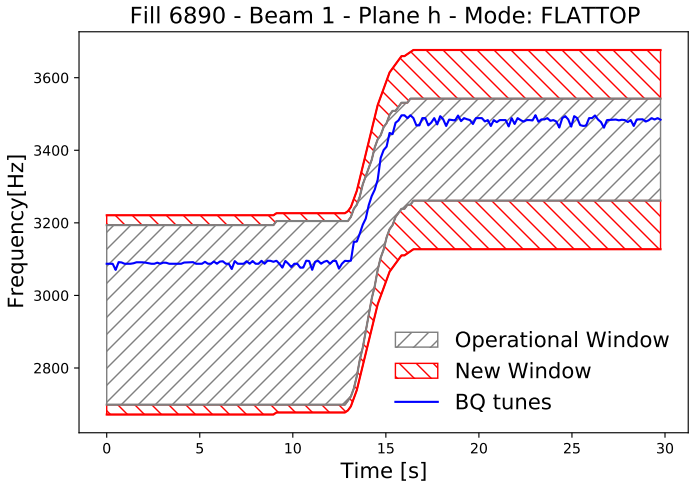

Figure 4. Comparing the operational frequency windowing used in the BQ algorithm to the new window length used in this work.

used in operation and as it can be observed, the tune estimates around the 15s 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.

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.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0605.v1

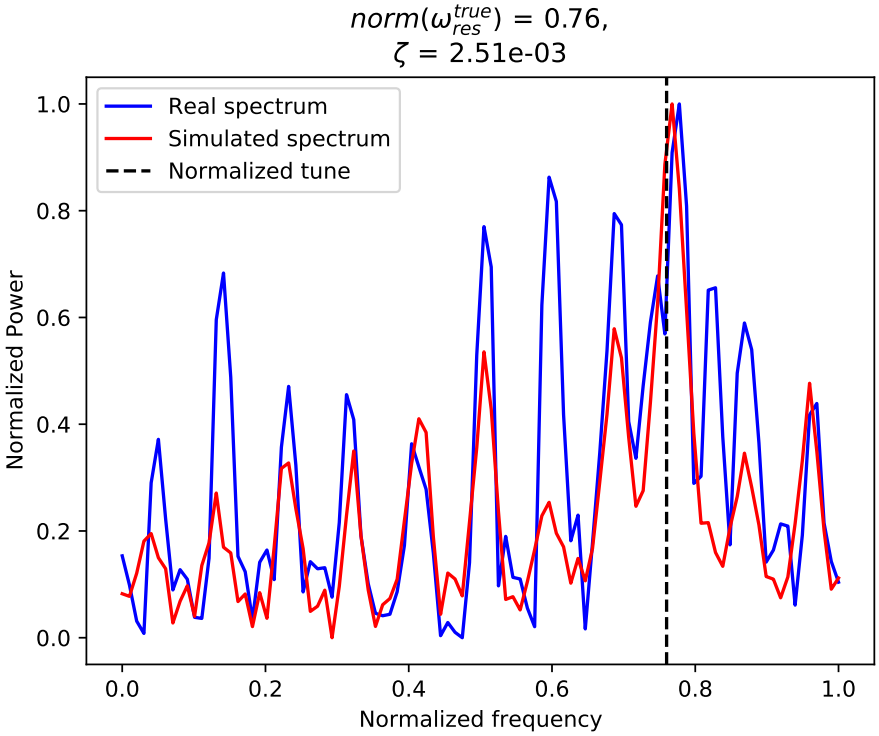

Figure 5. Second order simulation of a real spectrum obtained from the BBQ system.

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 values found from the most reliable algorithm from those discussed in the previous section, are used as the labels. In this approach however, the ANN can never be better than the algorithm that produced the tune label. That is why it is preferable to generate spectra using a second order system simulation, since from simulation we would precisely know the resonant frequency (i.e. position of dominant peak in the spectrum) used to generate the spectrum.

4.1. Fully-connected layers

The first network architecture that was considered was a 3-layer, fully-connected network (also called a dense network) as shown in Figure 6. Here the spectrum is first normalised to the range $[0, 1]$, and then is passed to the network. Each node in the network is connected to all other nodes in the previous layer, as well as to all the nodes in the next layer. The output of every node in all the layers but the last use a ReLU (Rectified Linear Unit) activation function, which clamps negative values to zero. Finally the last layer is then connected to one node to produce one scalar which is the tune estimate.

Three model architectures as seen in Table 1 were attempted. Figure 7 shows the density plots of the errors of the tune estimates provided by each respective model in Table 1. Here it can be seen that Model #1 and Model #2 have the best performance, with the highest accuracy and precision. It can also be noted that even though Model #2 has much more parameters than Model #1, the latter still performed as good. From Figure 7 it can also be seen that unlike the BQ algorithm and the alternative algorithms proposed in [5] (GP70, WMA30), the density plot of the three models drops to zero after

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0605.v1

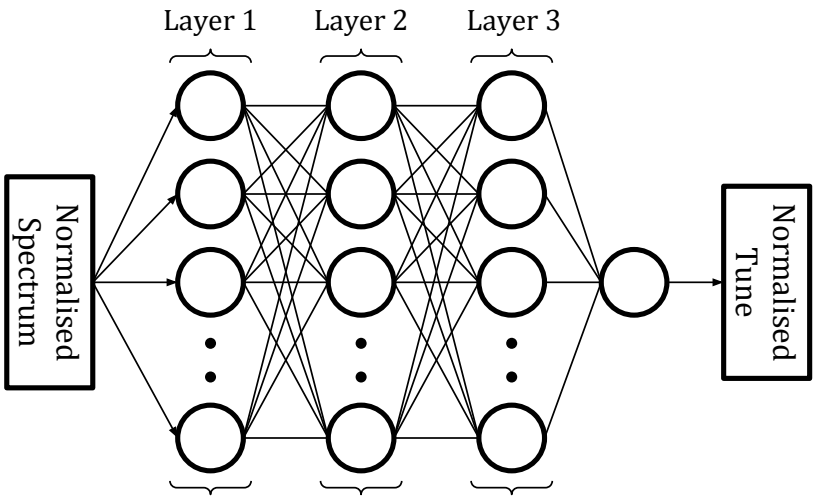

Figure 6. Fully connected network

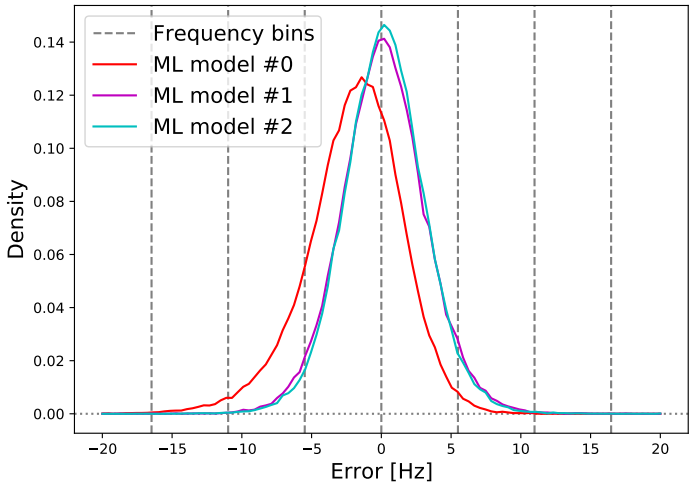

Figure 7. Density plot of the errors of the tune estimates of simulated data using fully-connected networks

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0605.v1

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

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

[1] Number of trainable parameters.

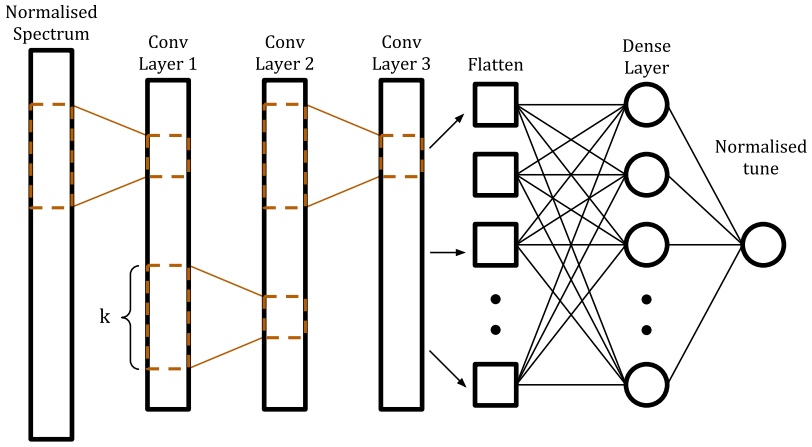

Figure 8. Network using convolutional layers

approximately 2 frequency bins. This contrasts with the current and the alternative algorithms which show a tail extending beyond the bounds of Figure 3.

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. Note that convolutional layers can be used on the output of previous layers in order to capture more complex features [6,7].

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.

Figure 8 illustrates a network architecture utilising 3 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 difference from the network architecture in Figure 6 is that now the input spectrum is condensed to feature maps, and only in the last layer are the features interconnected to produce the tune estimate. Table 2 lists the model architectures presented in this work which utilise convolutional layers [6].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0605.v1

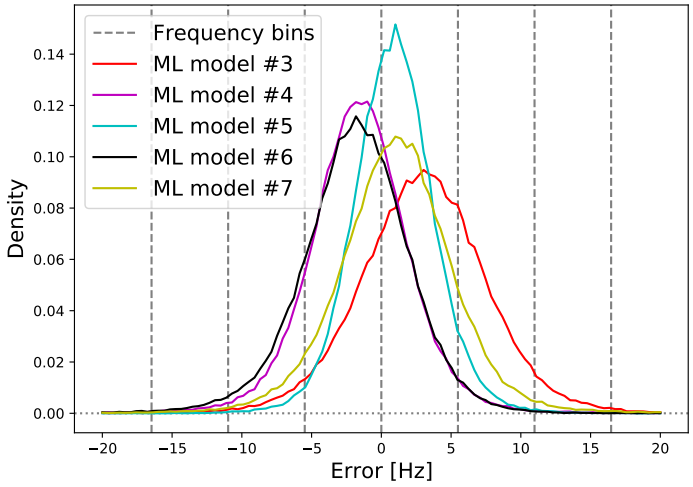

Figure 9. Density plot of the errors of the tune estimates of simulated data using convolutional networks

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		
Model #3	32	3	3	16	3	3	8	3	3	20	2,753
Model #4	32	3	1	16	3	1	8	3	1	20	18,113
Model #5	64	3	3	32	3	1	16	3	1	20	18,905
Model #6	128	3	3	64	3	3	16	3	3	20	29,561
Model #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.

4.3. 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.

Figure 10 shows the training and validation losses of a model having an architecture similar to that illustrated in Figure 6. Validation losses are obtained by comparing the labels in the validation dataset to the prediction of the network over the unseen data. The plot shows that the loss of the network during training and the loss obtained during validation, i.e. the loss from unseen data, are very similar when using simulated spectra for validation. This indicates that the network is successfully learning to predict the tune values from a simulated spectrum.

The problems with the simple approach are exposed when validating the network using real spectra whilst still training with simulated spectra, and having the BQ, GP70 and WMA30 estimated tunes used as the labels, we can observe a gap of approximately an order of magnitude between the training and validation loss from BQ. 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. This discrepancy in the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0605.v1

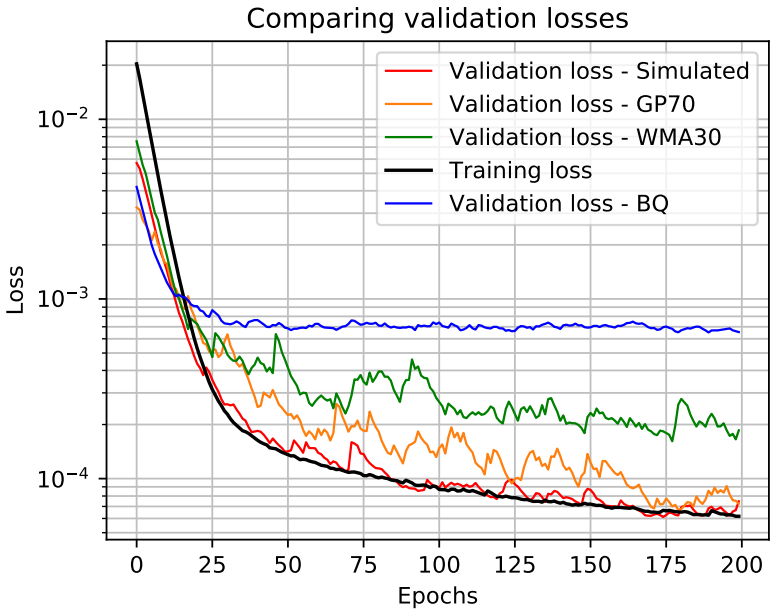

Figure 10. Comparison of validation losses when using simulated and real spectra respectively. Note that the dataset containing the real spectra was used to obtain tune estimates from BQ, GP70 and WMA30, thus creating 3 separate validation datasets.

losses is to be expected when knowing that the algorithms used to obtain the validation labels are erroneous.

From Figure 3 it can be seen that the average error of the BQ algorithm is almost at 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, we can try to improve our training data distributions by making them look more real.

5. Improving the dataset

A technique introduced in [8] called SimGAN was considered as a potential solution to the above problem. SimGANs build upon the work done in [9] which introduced the Generative Adversarial Network (GAN). The goal of the GAN architecture as shown in Figure 11 is to train a *Generator Network* (generator) to transform a random input, into an image which looks similar to the images in the dataset of *Real Images*. 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.

5.1. SimGAN

SimGAN is a modified version of the GAN, where the goal is to refine a synthetic, or simulated image to look more realistic [8]. Figure 12 illustrates he architecture of

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0605.v1

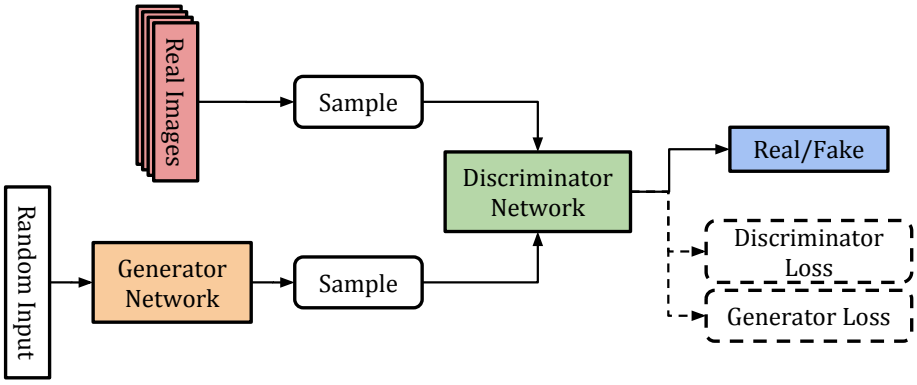

Figure 11. Overview of the Generative Adversarial Network architecture.

SimGANs, which is similar to that of GANs, however with two notable differences: The input of the generator is now coming from a simulator; The addition of a regularisation loss to the generator loss, which restricts the output of the network to remain similar to the input. In turn, the discriminator is continuously trained to distinguish between refined and real images in order to ultimately help the generator create more realistic images in an adversarial manner. Since the role of the generator is now to refine an input, in SimGAN nomenclature it is referred to as a *Refiner Network* (R).

SimGANs were originally presented to work with images via 2D convolutional networks, however since we aim to refine simulated spectra, to look more like spectra obtained from the BBQ system we need to convert SimGAN to work with either 1D convolutional networks or dense networks. The theory presented below remains valid however as the same losses have to be minimised regardless of the input shape and 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 synthetic data x, with θ as the function parameters. Therefore we can define refined data, $\tilde{x}$ as:

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

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. To achieve this, we learn θ by minimising a combination of two losses:

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

where x_i is the i^{th} synthetic image. In Equation (3) ℓ_{real} adds realism and ℓ_{reg} preserves the annotation information. Since a refiner makes it difficult to classify images as real or refined, we need an adversarial discriminator D_ϕ, that is trained to classify

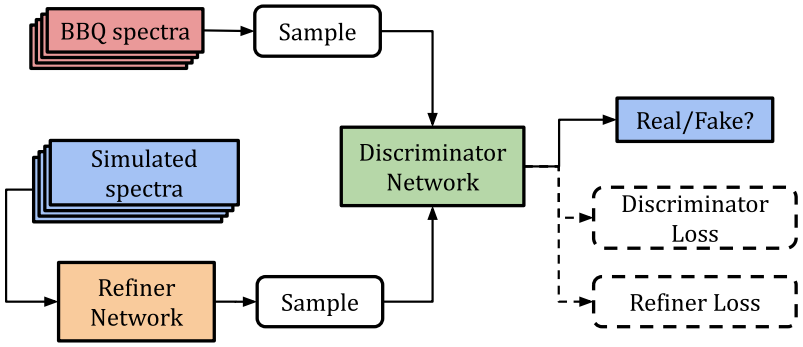

Figure 12. Overview of the SimGAN architecture.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0605.v1

images as real vs. 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:

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

Also note that now we can form ℓ_{real} by using D_ϕ to update θ:

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

By minimising Equation (5) we update θ 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:

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

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 50Hz noise harmonics. Instead the simulator would now only create a spectrum from Equation (1) and add Gaussian noise. Note that the noise is necessary in order for the refiner to create realistic images, otherwise it would be harder to generalise the refinement process, as in any update process utilising backward propagation [10].

The real dataset, $\mathcal{Y}$, used in Equation (4) to train the discriminator was obtained from BBQ data logged during FLATTOP of one fill from the LHC physics runs in July 2018. A smoothed version of these spectra can also be seen in Figure 14. Since the input size of all networks considered in this work was 100 frequency bins, the real spectra were clipped to fit in this window, and having its position being randomly chosen while guaranteeing that the dominant peak of the spectrum is within the window. This was done to ensure that the discriminator does not over-fit to the dominant peak occurring at always the same relative location with respect to the frequency window, thus ensuring better training of the whole SimGAN.

The network architectures used for both the refiner and discriminator are a 1D version of the networks used in [8]. The value of η in Equation (6) was set to 5 and initially λ in Equation (3) was set to $1 \times 10^{-3.8}$, a value which was found empirically to achieve the best balance between realism and preservation of annotation information. 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 as to be able to use them to reliably train a tune estimation model. It was found that 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 Equation (3) in order to obtain refiners which behave

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0605.v1

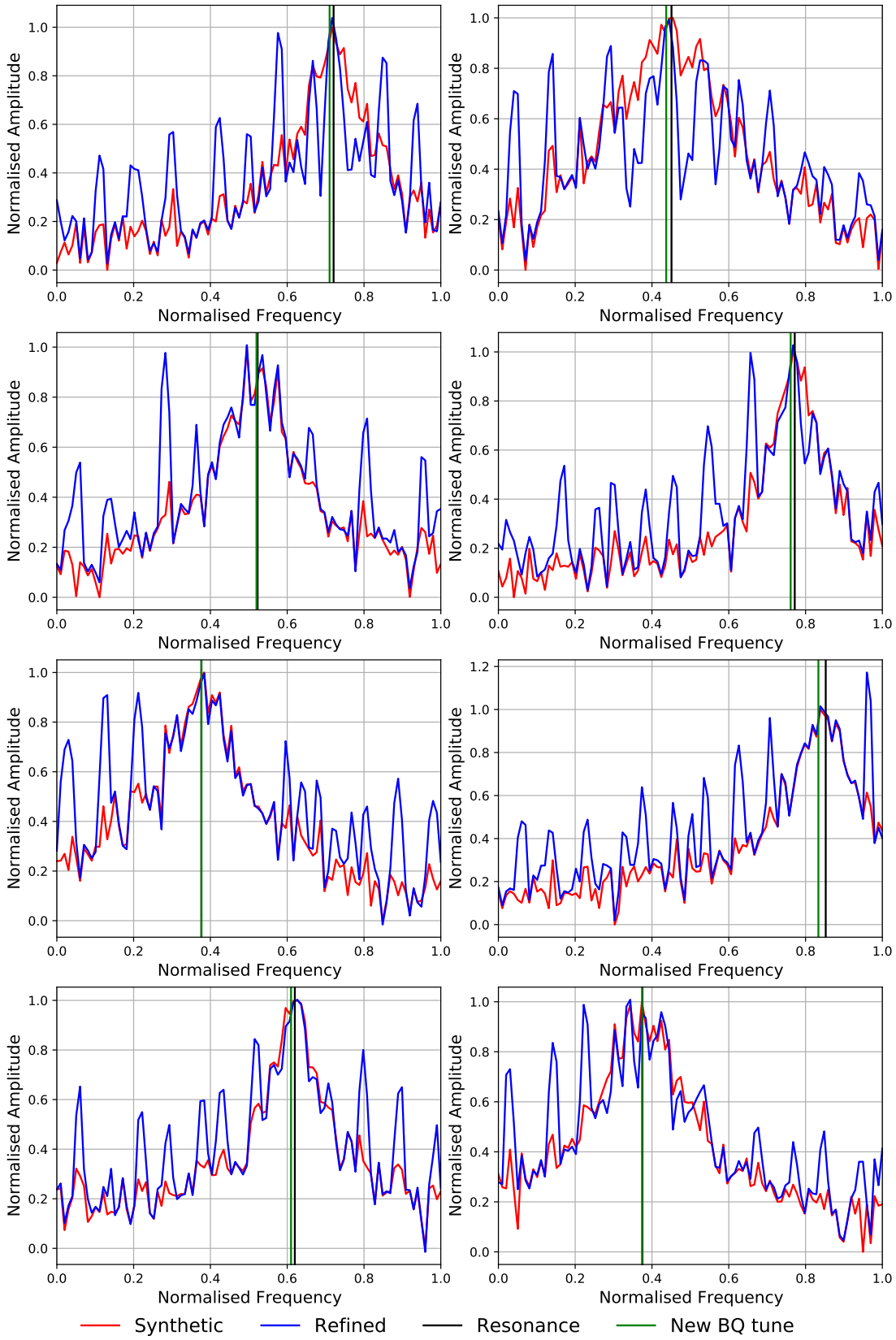

Figure 13. Results from a trained SimGAN. The green line represents the tune estimate from BQ algorithm on the refined spectrum.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0605.v1

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. Here we can observe that from a baseline obtained from Equation (1) with added Gaussian noise (red), a trained refiner is able to add artefacts which make the spectra look similar to a real spectrum like the one shown in Figure 1.

6. Discussion

It is difficult to show any objective difference in the performance of the new ML tune estimation models over each respective algorithm presented in this work (BQ, GP, WMA) while using real spectra to estimate the tune. The main reason being that we do not have access to the ground truth of the real tune value per spectrum. The performance in terms of accuracy and precision, of each method used in this work can only be assessed qualitatively by visualising the tune estimate traces over the real spectra.

Figure 14 shows the mean and 3σ of the tune traces obtained from BQ, GP, WMA, ML#1, ML#5 and ML-Refined, where the latter is the tune estimation model trained using generated spectra collated from a set of 500 refiners. The tune traces are superimposed on a heat map which is visualising the tune peak in the BBQ spectra. The heat maps were obtained after filtering the original spectra from the 50 Hz harmonics, as well as smoothing them to obtain a clear picture of where the tune estimate should be. Also note that each μ and σ of each tune trace was obtained after a moving average and moving standard deviation of the original tune traces using a centered window.

From Figure 14 we can observe that even if ML#5 performed well on simulated data, its performance is worse than the BQ algorithm in terms of its accuracy. The same can be said for ML#1 where we can see that around the 30s and 60s mark, there are instances where the tune estimate appears to be unstable. ML-Refined appears to be the most robust algorithm, where it can also be observed that its tune trace is the most stable of the three ML models shown.

As mentioned in the Introduction section, the tune estimates are used by the QFB to apply trims to the quadrupoles and correct the tune of the LHC to be at a reference value. In order to ensure stable operation the QFB performs a stability measure on the tune estimates, by keeping two exponential moving averages of the difference between subsequent tune estimates. Therefore:

$$Stability \;=\; q_{t+1} * (1-\alpha) + q_t * \alpha$$

where q_t represents the change in the tune value at time t and alpha being a time constant. These two averages use a 2 and 10 second equivalent time constant respectively, and in order to assert a stable tune estimate, both of these values have to be below a certain threshold. By default this value is 0.005. This measurement was recreated and the stability of the tune estimates coming from each respective model and algorithm in this work can be seen in Figure 15. This figure shows us that when using the stability as measured by the QFB as a performance metric, we are able to observe an increase in performance of the proposed ML solution using refined simulated spectra (ML-Refined) to train the dense tune estimation model.

7. Conclusion

Several ML tools, namely DNNs, CNNs and SimGANs were used to train a tune estimation model. Since for experimental data the associated tune values cannot be known with precise accuracy, the spectrum associated with a second order system was used to realistically model the beam response and to create a simulated dataset covering a wide range of tune values and damping ratios. Different model architectures were trained using this simulated data and the best model architectures for the tune estimation were chosen. Figure 7 and Figure 9 were used to choose the best DNN and CNN models.

 doi:10.20944/preprints202103.0605.v1

It was found that models trained on simulated data did not perform well on real data and therefore it was decided to use SimGANs 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. This refined dataset was then used to train a DNN tune estimation model.

Finally a sample of real data was used to compare the performance of the new tune estimation model with the models trained on simulated data, as well as the BQ, GP70 and WMA30 algorithms. The accuracy and precision of the tune estimation models was illustrated by Figure 14 where it could be observed that the models were successfully producing a tune estimate similar to BQ, GP70 and WMA30, with similar variation. It was also shown through Figure 15 that ML-Refined offers improved stability in the presence of 50 Hz harmonics which implies a more reliable data 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.

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.

References

1. Baird, S. *Accelerators for pedestrians*; CERN, 2007.
2. Steinhagen, R.J. LHC Beam Stability and Feedback Control-Orbit and Energy. PhD thesis, RWTH Aachen U., 2007.
3. Gasior, M.; Jones, R. High sensitivity tune measurement by direct diode detection. Proceedings of 7th European Workshop on Beam Diagnostics and Instrumentation for Particle Accelerators (DIPAC 2005), 2005, p. 4.
4. Gasior, M. Tuning the LHC. *BE newsletter* **2012**, pp. 5–6.
5. 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. Proceedings of the 9th International Beam Instrumentation Conference (IBIC 2020), 2020.
6. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press, 2016. http://www.deeplearningbook.org.
7. Lindsay, G.W. Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future. *J. Cogn. Neurosci.* **2020**, pp. 1–15.
8. Shrivastava, A.; Pfister, T.; Tuzel, O.; Susskind, J.; Wang, W.; Webb, R. Learning from simulated and unsupervised images through adversarial training. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2107–2116.
9. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*; Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N.D.; Weinberger, K.Q., Eds.; Curran Associates, Inc., 2014; pp. 2672–2680.
10. Matsuoka, K. Noise injection into inputs in back-propagation learning. *IEEE Trans. Syst. Man Cybern.* **1992**, *22*, 436–440.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0605.v1

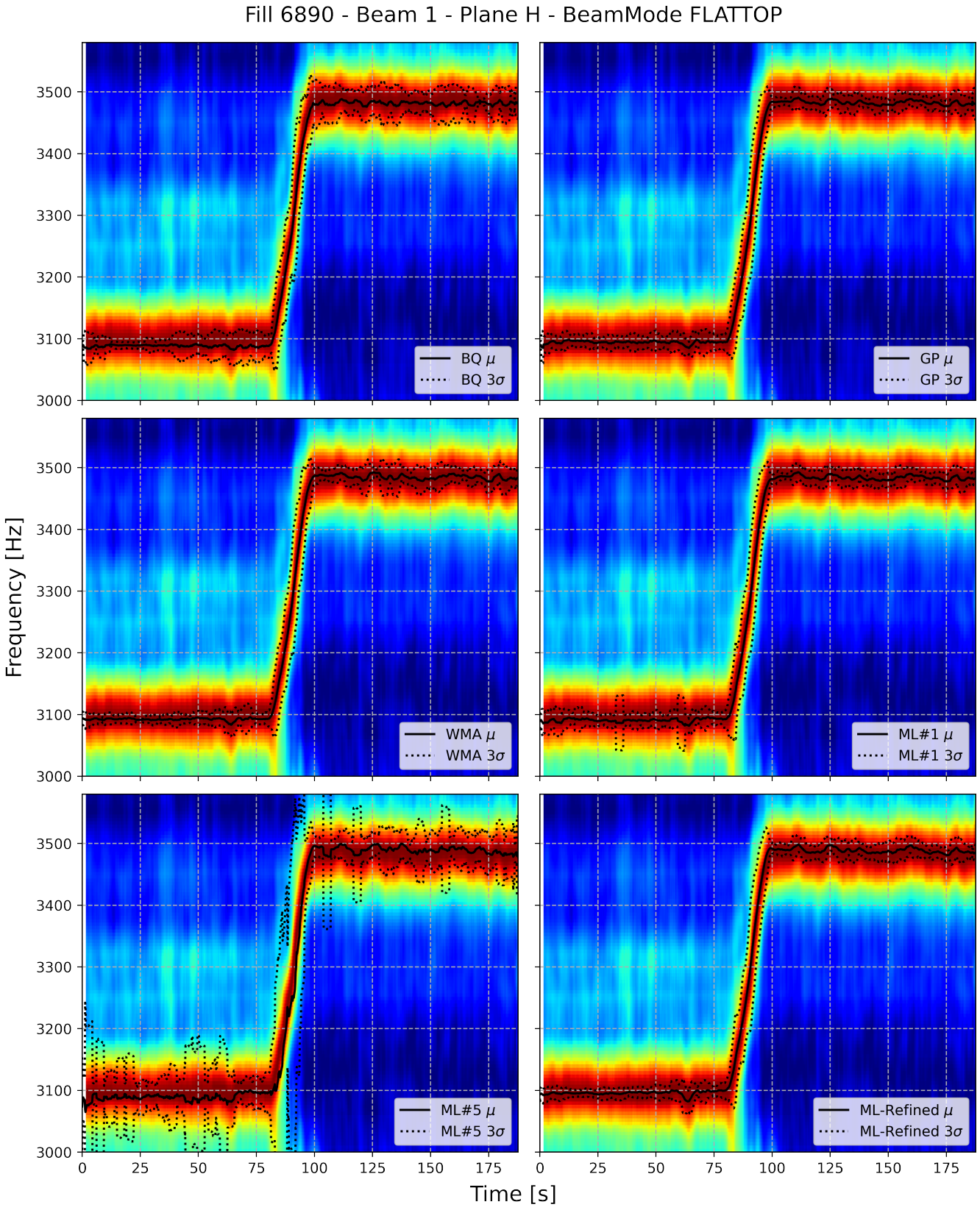

Figure 14. Tune trace plots of the algorithms and models presented in this paper super-imposed on the same heat map of the spectra.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0605.v1

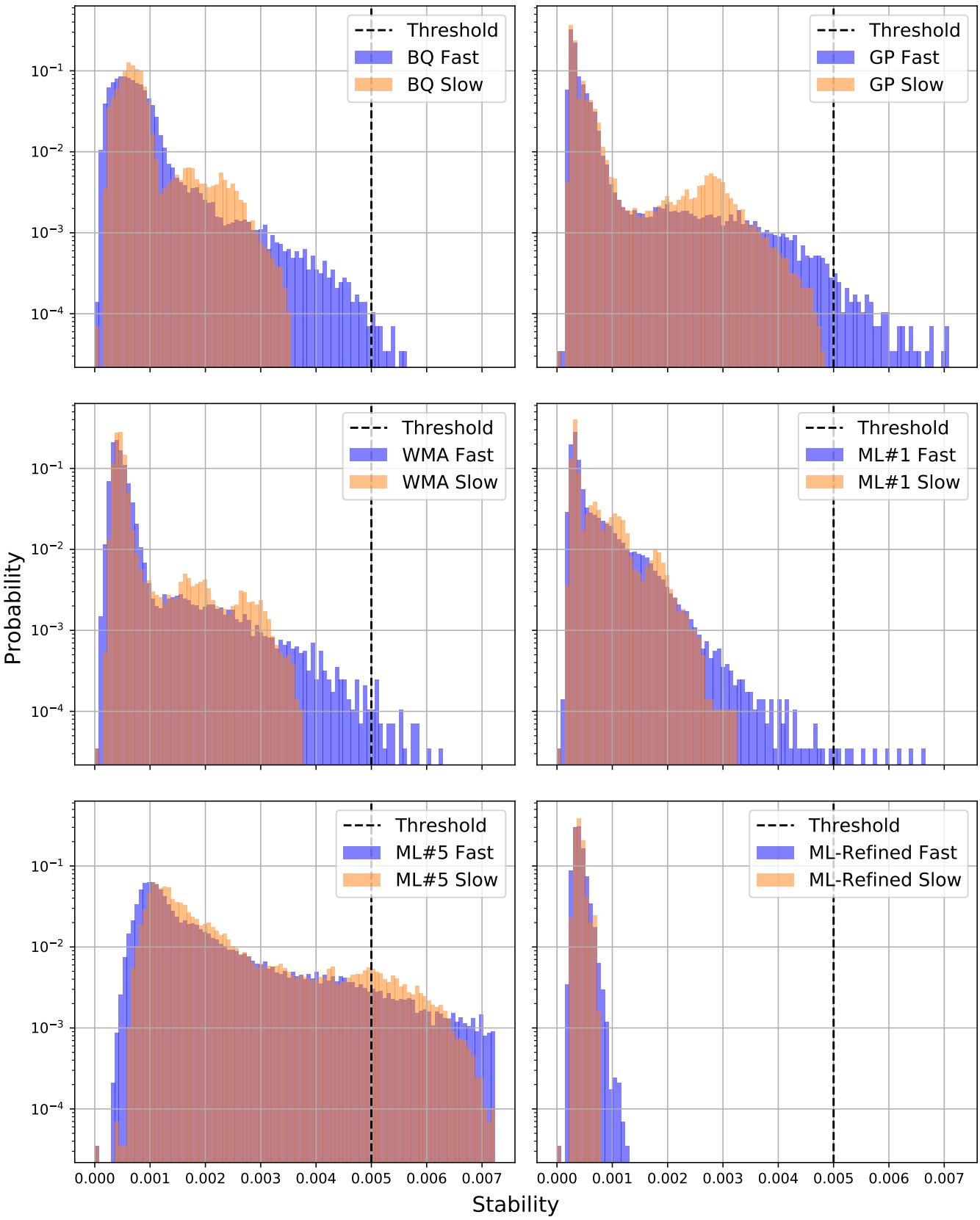

Figure 15. Probability distribution of the tune estimation stability. Obtained from Fill 6890, beam 1, horizontal plane. Slow corresponds to EMA with time constant of 10s and Fast to EMA with time constant of 2s. The threshold was chosen in the early design of the beam-based feedback systems and used in operation during Run 2. [2]