Machine Learning Model for Injector Emittance

Raffaele Campanile

September 24, 2018

Abstract

In this report is summarized the work I did at the LCLS facility at SLAC (Stanford Linear Accelerator Center) during the INFN/SLAC Summer Exchange Program. I worked in the group of Daniel Ratner in collaboration with Auralee Edelen, our goal was to create a Machine Learning (ML) model able to predict the Laser injector emittance.

The project is composed of three main sections. In the first part I got familiar with the data archive; the LCLS database it's large and stores a lot of information, during this time I learnt how to deal with big and complicated data structures. Once able to interact easily with the archive, I started selecting the injector parameter to take into account into the ML model, the selection was made basing on physical consideration, enforced by correlation analysis, and discussed with the LCLS Team.

In the second part I focused my the attention on the transformation of raw data into a consistent *dataset*. I was aware of the crucial role the quality of the dataset would have played for the success of the learning process, so I spent the time due assembling it. Besides grouping the data together by time and storing them into a matrix form, I also carefully dropped from the collection misleading data.

Once built the dataset, I trained the ML model on it. I opted for a feed forward Neural Network made of fully connected dense layers and I trained it using the stochastic gradient descent method and backpropagation algorithm. After a reconsideration of our inputs variables and an hyperparameters optimization, we managed to create a model able to roughly predict the injector emittance. I tried to go beyond obtaining the best result achievable using the limited available data, indicating the way to go in order to improve the quality of the prediction and pushing the LCLS community to gather data in a more frequent and systematic way.

I cannot estimate what I learnt during my time at SLAC, it had an invaluable impact on my education as future scientist. I had the occasion to share ideas with an outstanding Team of very competent and kind people, their advices were so helpful that changed my way to face problems. At the same time, I had the occasion to take part to the set up of a really complicated experiment, understanding how does it looks like to work in a huge collaboration.

Contents

1 Injector background concepts

The most important part of the whole *LCLS* facility for our studies, is the injector. Since describing how it works goes beyond the scope of this report we refer the reader to the *LCLS* manual.
By the way we want to stress that to understand the following you don't need to have a deep knowledge on how the injector is made or how it works.

2 Available data

The data are available in the archive of the *LCLS*. To accede them you need to be logged in the Physics server, from the Laser control room or from your own laptop. To access any information about a variable (i.e the magnetic field of a solenoid inside the injector) the corresponding PV is required. The last has to be inserted as first field in the Matlab history function together with the time interval of interest. A cell Matlab array is got as output: the first column contains the times, included in the interval specified above, in which the variable was measured while in the second column is stored its value.
The *LCLS* injector is formed by an enormous quantity of devices each one contributing in different ways to the features of the output Laser beam. Taking into account every single component in the ML model is not an option. To go through this, we discussed in a group meeting the devices and their property that we supposed to play the most important role basing on physical consideration.
The result of the discussion was the following table, in which the above mentioned devices of interested and their properties are listed with their PVs and a brief description of the role the play. Aside the quantity listed above, there is another important source of information available that is the *Virtual Cathode Chamber* (VCC) image.

Process Variable	Device Name	Description
LASR:LR20:1:UV_LASER_MODE	Active Laser	Which of the Lasers is active
PSDL:LR20:117:TACT	Stacker Delay (Actual)	For normal charge and normal ops, always ~2 ps relative to the delay resulting in zero overlap
WPLT:LR20:117:PSWP_ANGLE.RBV	Stacker Waveplate	Changes the intensity balance between the two polarizations in the laser stacker
SIOC:SYS0:ML00:AO468	UV laser pulse length (FWHM)	The most recently measured laser pulse length
OSC:LR20:20:POC	Coherent 1 phase offset	Should only be changed during a systematic phase scan
OSC:LR20:20:PDES	Coherent 1 phase desired	Generally this is -30 degrees during normal operation
OSC:LR20:10:POC	Coherent 2 phase offset	Should only be changed during a systematic phase scan
OSC:LR20:10:PDES	Coherent 2 phase desired	Generally this is -30 degrees during normal operation
ROOM:LR20:1:AIRTEMP1	Laser room temperature	One of several laser room temp monitors
CATH:IN20:111:QE	Cathode quantum efficiency	Quantum efficiency
TORO:IN20:215:TMIT1H	Charge from gun	Absolute measure of charge emitted from gun
PMTR:LR20:121:PWR	Laser power on iris	Figure of merit when it comes to steering the laser before the iris.
LASR:IN20:196:PWR1H	Laser power on cathode	The lase power measured on the cathode
MIRR:LR20:125:M13_MOTR_H.RBV	IRIS steering mirror (horiz. and vert.)	Iris steering mirror, both the horizontal and the vertical one
SIOC:SYS0:ML00:AO328 (:AO329)	VCC X and y offset	The position on VCC if alignment has been performed, feedback o.
WPLT:IN20:181:VCC_ANGLE.RBV	VCC waveplate	Angle of the VCC waveplate
LASR:IN20:475:PWR1H	Heater_power	Heater power
SIOC:SYS0:ML01:AO972	Bunch_Length_XTCAV	Bunche Lenght
CAMR:IN20:186:XRMS (and :YRMS)	Virtual cathode camera	RMS size from fit
CAMR:IN20:186:BACT	Gun solenoid	Actual magnetic field of Gun solenoid
SOLN:IN20:121:BACT	Gun corrector quad	Actual magnetic field of correctior quadrupole
QUAD:IN20:121:BACT	Gun skew quad	Actual magnetic field of skew quadrupole
QUAD:IN20:122:BACT	Quadrupole magnet	Actual magnetic field used for macthing the solution
QUAD:IN20:361:BACT	Quadrupole magnet	Actual magnetic field used for macthing the solution
QUAD:IN20:371:BACT	Quadrupole magnet	Actual magnetic field used for macthing the solution
QUAD:IN20:425:BACT	Quadrupole magnet	Actual magnetic field used for macthing the solution
QUAD:IN20:441:BACT	Quadrupole magnet	Actual magnetic field used for macthing the solution
QUAD:IN20:511:BACT	Quadrupole magnet	Actual magnetic field used for macthing the solution
QUAD:IN20:525:BACT	Quadrupole magnet	Actual magnetic field used for macthing the solution

Figure 1: *This table lists the variable (PVs) we suppose to impact the emittance value.*

3 *Virtual Cathode Chamber (VCC) Images*

The VCC images contain a lot of important physical information that potentially could be used to make predictions on the emittance but unfortunately, due to their memory requirements (480x640 pixels), they are only saved twice a day. The last circumstance implies that it's really rare to have a VCC image captured when an emittance measurement is going on, posing at serious risk their introduction into the dataset. People working at *LCLS* overcame this hurdle by finding the VCC image decomposition in Zernike functions, a sequence of polynomials that are orthogonal on the unit disk. Named after optical physicist Frits Zernike, winner of the 1953 Nobel Prize in Physics and the inventor of phase-contrast microscopy, they play an important role in beam optics. In this way, since the the Zernike Polynomials are well known, it's sufficient to save the coefficients and then run the reconstruction to obtain an image that approximate the original VCC image. In general an infinite number of coefficients is required to reproduce the input image exactly but previous studies highlighted that, for the image considered here, a set of 45 coefficients it's enough to obtain a very good agreement with the VCC image, moreover increasing the number of coefficients there were no significant improvements. From May 2018 the Zernike Coefficients started to be recorded very often during a day making them a good candidate to take part in the input data.
In this section we ascertained if the Zernike coefficients could safely play the role of VCC images in our dataset. To make this happens we looked at two aspects:

- The agreement among VCC images and the ones reconstructed by Zernike

Polynomials.

- The autocorrelation time of Zernike coefficients to be negligible compared to the time required for emittance measurements.

To check the first point we calculated the *pearson's correlation coefficient* (ρ) among VCC images and the reconstructed ones for a large set of entries. The histogram in figure 2 shows that there's a really good agreement, in particular the value of the mean value (98%) clearly states that from this point of view we can safely replace VCC images Zernike Coefficients.

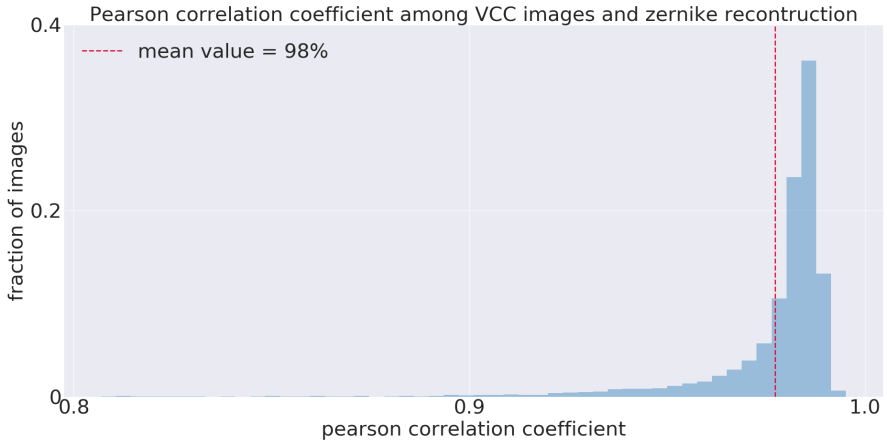

Figure 2: *Histogram of pearson's correlation coefficient among VCC images and reconstruction using Zernike coefficients.*

We estimated the autocorrelation time among Zernike Coefficients calculating the mean square distance among them in function of time. In particular we fixed an initial time (t_0), with the corresponding Zernike Coefficients $\{c_i\}_0$, and calculated the mean square distance among the coefficients $\{c_i\}_t$ and $\{c_i\}_0$ in function of time. As can be noticed looking at figure 3, the coefficients, as expectable, slowly becomes uncorrelated, becoming completely independent after a characteristic time of ~ 30 hours, plenty larger ($\sim$2 order of magnitude) than the few minutes required for an emittance measurement. As last proof, we just show some examples of comparison among VCC images and ones obtained by running the Zernike reconstruction algorithm.

4 Turning data into a dataset

The data available in the LCLS database are not ready to be analyzed. In the language of Computing Science they do not constitute a *dataset*, an ordered collection of data presented in tabular form, each column representing a particular

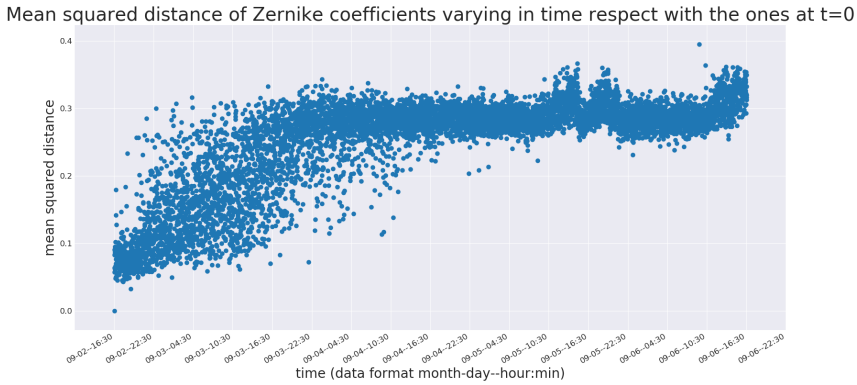

Figure 3: *This picture shows the autocorrelation among Zernike coefficients during time. To measure how two set of Zernike coefficients are correlated we used the mean square distance among them. We are are aware that to obtain a reliable estimated of the autocorrelation time we should have averaged on different starting time, but we just wanted to get a rough estimate of it. Since it turned out to be so bigger compared to the time needed for an emittance measurement (few minutes) we didn't push further our analysis.*

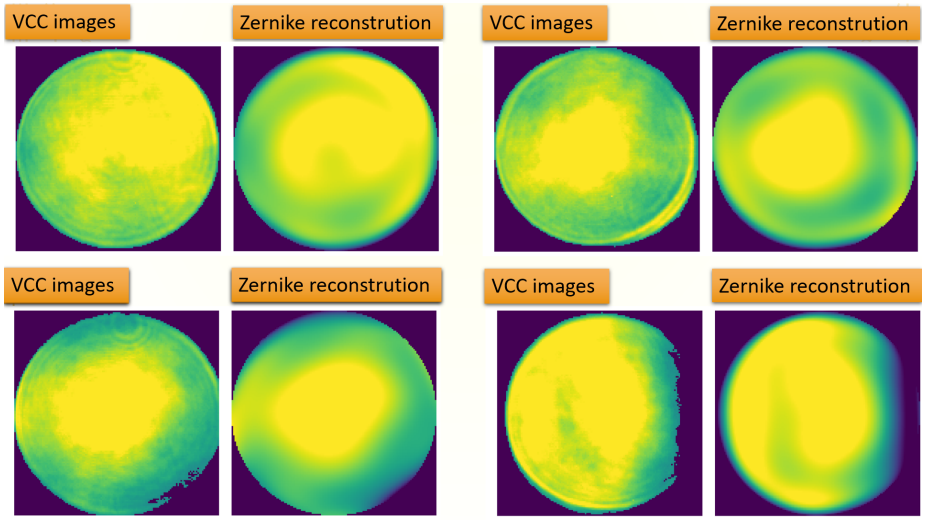

Figure 4: *Some examples of VCC image reconstructed by Zernike polynomials. As can be noticed the approximate matrix is really consistent with the VCC one and recapitulates its most important features in each case, including the more difficult.*

variable and each row corresponds to a given member of the dataset in question. The transformation of raw information into an ordered table or matrix is

the most important and difficult task when trying to apply Machine Learning methods to any problem. A Machine Learning model could perform fruitfully on a well-built dataset completely failing with a bad one. This process it's so important that requires the most of time spent on the project our case not being an exception.
Since our goal is to make prediction on the emittance, we started targeting periods in which there were measurements of this quantity. We decided to focus on the last two years considering the emittance measurements ranging from the 1 January 2017 to 10 September 2018, we could not go behind in time because some PVs, that according to us could play a role in the emittance prediction, weren't saved before the starting date we chose, this circumstance due to updates to the Laser instrumentation. There are actually two ways to measure the emittance of the *LCLS* injector: on the OTRS camera and on the wire scanner. We decided to start using only the data on the OTRS (**OTRS:IN20:571:EMITN_X** and **OTRS:IN20:571:EMITN_Y**), mainly because they were a greater number. In the next section is reported how we managed to introduce the wire emittance measurements too. Figure 5 shows the values of the OTRS emittance along the x-axis during period of time under consideration (the one on the y-axis being similar is not illustrated for the sake of simplicity).

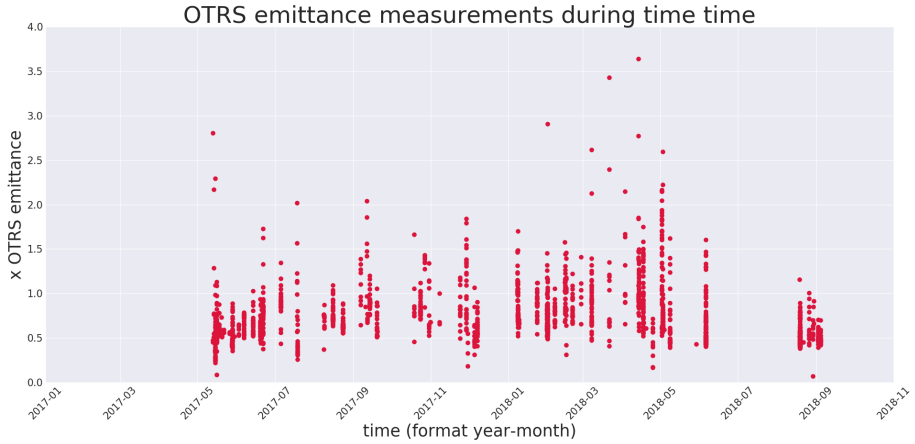

Figure 5: *Emittance measurements in the time range considered. As can be noticed the values are often captured in groups, often corresponding to injector tuning session. The absence of data in the early 2017 and in the summer of 2018 corresponds to shutdown periods of the LCLS.*

We could load only 2075 emittance measurements, this value is now the upper limit in our dataset size, meaning that we can't aspire to have a bigger one, rather we should check if any of these data point is a good candidate to enter in our dataset. For each one to be accepted, we need to have the values of all

the inputs parameters we want to use to make predictions on the emittance. In order to get these ones, we looked for measurement of the input quantities (table 1) in times as close as possible to the emittance ones. Unfortunately the LCLS data acquisition system is not saving the values of each parameter at the same time, however positive it could be, this is not an option because it would be too space-intense. So every different parameter is measured at a different frequency depending on its importance and on its requirements in term of resources. The sampling is not regular because when the Laser is used for particular tasks (i.e injector tuning) each value can be saved by people using it.
To overcome the hurdle of non-contemporaneous measurements, we acted in this way: considering that a the time to make a measure of the emittance is of the order of few minutes we looked for values of input variables staying in that range and considered the most close in time to the emittance measurement. This is an approximation whose impact couldn't be evaluated *a-priori*, if the model manage to learn how to make predictions it means we have safely employed it, on the other hand, in case of failure, this could be one of the possible reasons. We didn't include a data points in our dataset even if just one of the input variables was not measured in a range of few minutes from the emittance capture, this reducing the dataset size.
In this way we obtained a first draft of the dataset on which run the ML model. Since we want to predict the emittance when the laser is running in the standard configuration we looked for evidence of atypical data points in our collection. At this purpose we used the variables listed in figure 6.

Process Variable	Device Name	Description
SHTR:LR20:100:UV_STS	Coherent 1 Laser shutter	Which Laser shutter is active, when both or none are it's a strange config.
SHTR:LR20:90:UV_STS	Coherent 2 Laser shutter	Same as above
SHTR:LR20:117:SARM_STS	Stacker S arm shutter	At normal charge and for normal ops, both arms are open. If only one is open, typically means operating in low-charge mode or performing some MD activity.
SHTR:LR20:117:PARM_STS	Stacker P arm shutter	Same as above
BPMS:IN20:525:TMIT1H	BPM	Important to be sure that the beam is on at screen/wire
SIOC:SYS0:ML01:CALCOUT008	Charge from gun	Values too far from 250 pC are not reliable
PS:IN20:1:SHUTTERSTATE	MPS shutter	No electron beam when closed

Figure 6: *This tab contains the process variables we employed to drop off the dataset measurements captured in non standard configurations. For instance the MPS Shutter property state if the electron beam is present, if it's not we should drop that data point since without the e-beam the Laser can't work properly.*

Another important step into the dataset preparation is the data normalization. Every process variable is saved in the archive with it's own units (i.e. seconds or meters and so on) while NN requires a-dimensional data moreover in the [0,1] range (or [-1,1] with the type neurons considered). At this purpose we employed standard methods of normalization for each single input variable paying great attention to distribution outliers. Considering the distribution of input or output variables there could be the presence of very high (or very low) values often captured by mistakes in the process of measurement. The presence of this outliers lower the ML model possibilities of success because of the important influence they have on the normalization process. For this reason we decided

to drop from our dataset the points over the 95 percentile of the distribution of each input and output variable, figure 7 showing the case of the output variable emittance on the y-axis.

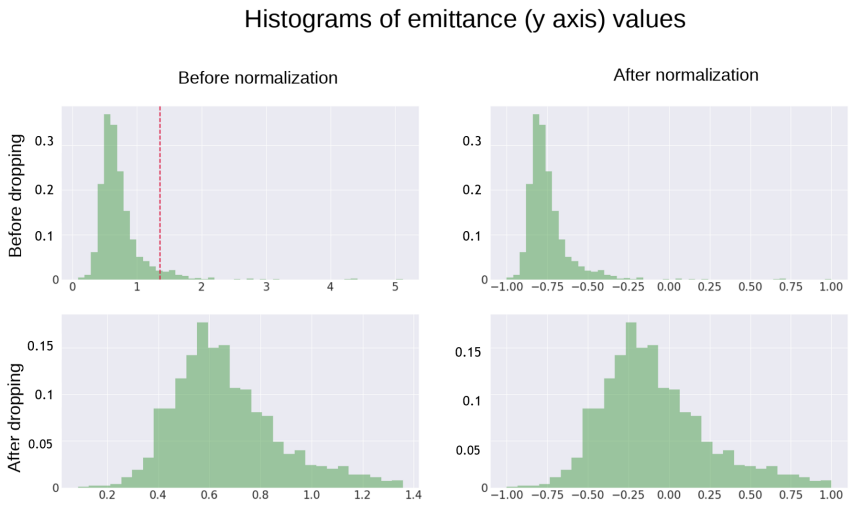

Figure 7: *In the upper pictures we can see the histogram of the emittance measurements before the normalization (left) and after it (right). The presence of few values one order of magnitude greater than the mean of the distribution adversely affects the normalization process. Running the ML model on a dataset with this points often has a negative impact on the learning process, moreover considering that those values are probably caused by mistakes in their estimation, we decided to cut them discarding the 95 percentile of the distribution. In the lower figures the histogram after the drop are shown.*

5 The Machine Learning Model

The ML model we decided to employ was a Neural Network (NN), a beautiful biologically-inspired programming paradigm which enables a computer to learn from observational data.
A NN is based on a collection of connected units or nodes called artificial neurons which loosely model the neurons in a biological brain. Each connection, like the synapses, can transmit a signal from one artificial neuron to another (figure 8).
At the beginning we were planning to employ a Convolutional Neaural Network to deal with VCC images, since we couldn't introduce them into the dataset, we switched to a simpler feed-forward NN with fully connected dense layers that we tried to train using the stochastic gradient descent method and backpropagation algorithm. To implement ideas, we used Keras, a high-level Neural Networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. In particular we employed TensoFlow as backend making large use

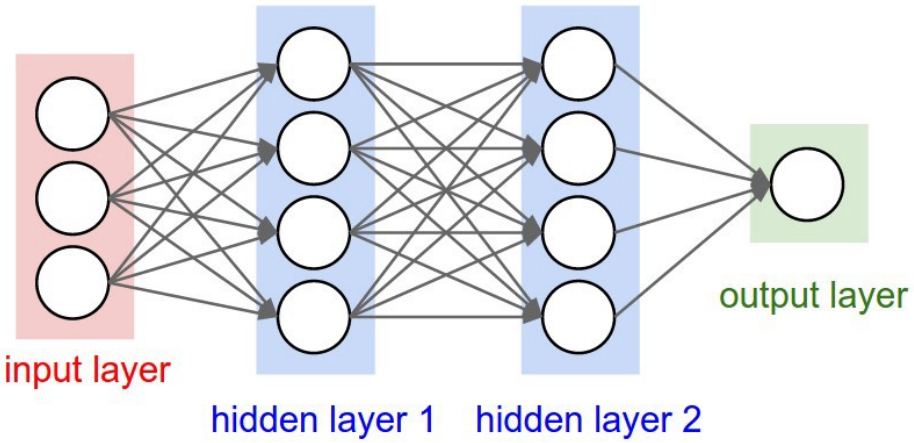

Figure 8: *Schematic representation of a Neural Network (NN), the artificial neurons are the circles, while the synapses are the line going to one circle to another. The way the information is propagated, from the input layer through the hidden ones to the output, is by matrix multiplication of weights (w) and the value neurons assume. The value of neurons on the input layers it's equal to the value of the input parameters of each instance of the dataset, while the values on the last layer have to be consistent with the corresponding with the output ones. At the beginning of the training process the weights are randomly assigned, the NN learns the best ones training on the database.*

of the TensorBoard to understand how was the learning process going.
The hardest part, when working with NN, is the hyperparameters setting. The hyperparameter we had to choose were the activation function of the neurons, the optimizer for the stochastic gradient descent, the number of layers, the number of neurons for each layer and the regularization. There are no general rules for finding the best ones and they varies depending on the dataset and there's great possibility to choose the more performing. We set the activation function of the neurons to be the hyperbolic tangent ($tanh$) that is the most spread out for this class of problems while we employed the Adam optimizer for stochastic gradient descent without optimizing any of its parameter since it wasn't necessary. Another important aspect of the learning process is the loss function that is the comparison meter among the prediction of the NN and the real data, we chose the mean squared error since it's the most commonly employed being simple and fast in the computation. About the regularization (that is important to avoid the overfitting), we chose the L2 one, setting its parameters to very common values. For the remaining hyperparameters, we made a limited grid search changing the values around the ones that in our experience worked better for these kinds of problems and that were found in literature in other studies too. A good kind of approach I would suggest for future studies, is to employ genetic algorithms or more generally adaptive techniques to. Unfortunately we didn't have enough time to push this further and limited to small and coarse

grained grid searches. We found the following best set of hyperparameters: 3 hidden layers, with respectively 100,50,25 neurons. At this point we examined the results on the training set and on the testing one. As we can see from figure 9, they were not so encouraging: the network could not perform properly neither on the training set nor on the testing set. In particular, it failed to make useful prediction and moreover it didn't managed to fit the part of the dataset on which it was trained on.

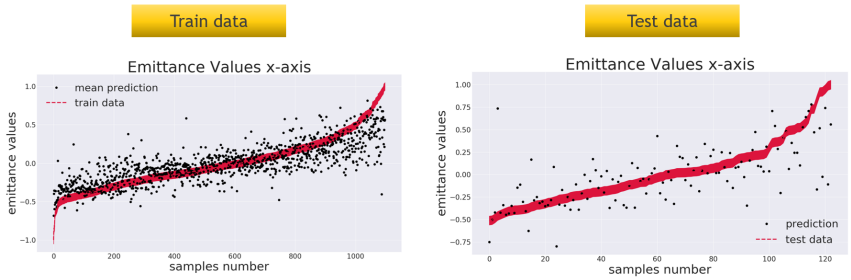

Figure 9: *In this picture we aim to measure the performance of the NN, on the left picture on the training set and on the right on the testing set. On the y axis there is the value of the emittance (specially scaled between [-1,1] as reported in the previous section) while on the x axis there's the samples number. The values are in ascending order of magnitude. The red dotted line represents the experimental data while the black dots correspond to the predictions of the NN. We consider a prediction consistent if it's inside the red shaded area that is thin because it considers the error on the emittance measurements. We could have employed error bars in the plot but we think that this is the best way to look at this particular results, the meaning is the same. As we can see the results are not promising: the distribution is really noisy and doesn't even follow the trend on the testing set.*

Seen the poor results, especially on the training set, we started doubting on the consistency of our dataset (and on the ability of the NN to solve the problem) and, before trying to improve the performance on the testing set, we assured that a simple ML model could at least fit the data on which it was trained on. At this purpose we used a really small NN (3 layers, with respectively 10, 8, 5 neurons) and trained it without using any regularization for a large number of epochs until the loss function on the training set was not improving significantly. In figure 10 you can see the results: despite its simplicity the NN was able to fit data well and this encouraged us on its possibilities to make predictions.

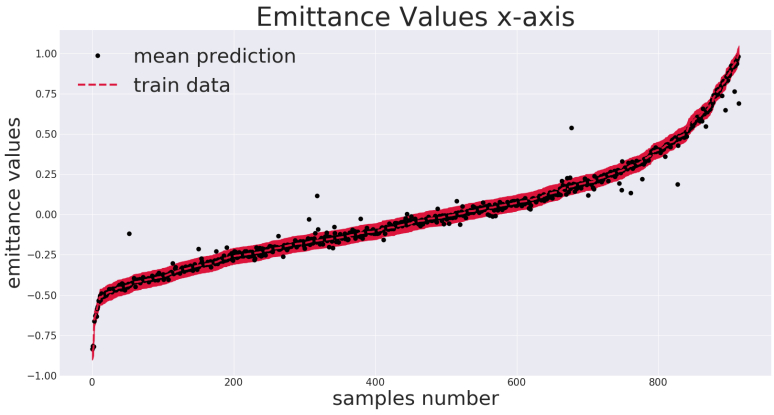

Figure 10: *The picture aims to show the same concepts of the previous one but focusing on the training set alone. As can be noticed, almost every output of the NN is consistent with the data.*

To improve the ability of the NN to make prediction, we worked on the points listed below:

- Instead of training just a single Neural Network, we created an ensamble of Neural Networks and averaged their predictions.
- We reviewed our PV list with the LCLS Team and excluded the most noisy ones (the laser heater and the bunch length on XCTAV).
- Since our feeling was to be data limited, we added to our collection the data related to the Wire Emittance (**WIRE:IN20:561:EMITN_X** and **WIRE:IN20:561:EMITN_Y**) almost increasing of 50% the size of the dataset.
- Instead of using L2 regularization we started including dropout layers.

The ensamble of NNs idea turned out to be very useful and improved a lot the performance. We introduced it because we were aware our PV list could be not enough to predict the emittance values. We could have missed some variable playing a significant role for the emittance measurement, in the best case the value of this variable is saved and stored in the *LCLS* database and we could include it making a greater effort in collecting the PV list. In the worst case random fluctuation in the surroundings or into the Laser itself could cause that the same values of input parameters return a slightly different value for the emittance. When working with a so complicated machine this possibility can't be excluded *a-priori*, obviously, in order to apply our studies, we should expect them to be a process of 'second order', the main part of the emittance being associated to the input values of beamline devices. In this scenario, averaging

over the prediction of a set of NNs should lower the noisy part of the output and highlight the one induced by the input parameters.
The removal of noisy PVs from the list also played a role; having a limited dataset its quality becomes even more important of the general case since each datapoint or input variable has a greater weight.
The introduction of wire emittance data is, together with the ensamble Networks idea, the most responsible of the significant boost on testing set predictions we obtained (see below). We decided OTRSto introduce the wire data since they are a very similar kind of measurement to the OTRS one, the first being considered more accurate and less noisier than the last. The reason why we didn't start working with the wire emittance is that for this variable the amount of data we can gather is ever lower (approximately half) of the OTRS. The introduction of wire emittance into the dataset was welcomed by correlation analysis: it emerged that wire emittance measurements were highly correlated with the OTRS ones (as expected) and that they shared a positive high correlation with some input variables too.
The impact of dropout was significant too, we didn't managed to construe the main reason, but we think that the way the L2 regularization works is not good for this kind of problem. In fact, it tries to avoid overfitting limiting the magnitude of the weights while in our case the weights linked to the most important variables needs to raise in order to predict emittance values.
After these important improvements, we did another coarse grained grid search finding the same optimal values for the number of layers and the number of neurons. This time the NN performance were boosted due to our changes, and as we can see in figure 11 we get decent curve following points either on the training set or on testing one.

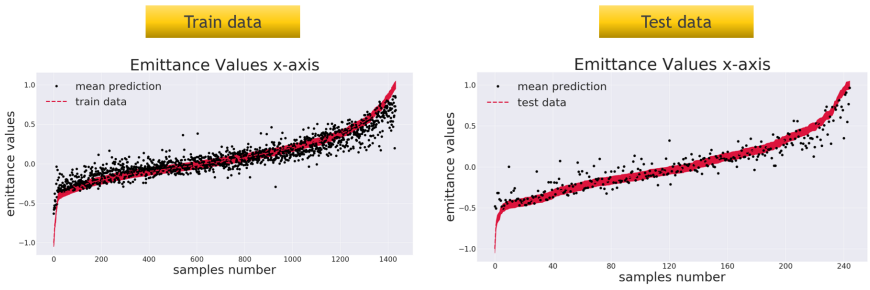

Figure 11: *The results shown here corresponds to the best NN trained after the improvements listed above. The performance on the training set got a huge boost but the great breakthrough we want to highlight is that we can fairly good reproduce the trend present in the testing set. The NN has never 'seen' the data in the testing set during the training process, so the picture on the left is stating the our ML model is capable to roughly predict the emittance following its trend.*

To fully appreciate the results we obtained, let's compare them with the ones in figure 9. On the training set we got some important improvements: the distribution of the points is less noisy and now is following quite well the trend of the data, being able to reproduce the change of concavity too. However this results are already very nice, we want to highlight that the most important breakthrough was made on the test set. The first NN trained (figure 9) was completely unable to give any indications on the emittance value without knowing it and being trained on it. The most important change is that we moved from a NN with zero chances of making predictions to one that is quite sure at least of the order of magnitude of the emittance. This is really important for two reasons. On one hand, being the first at facing this problem, we clearly demonstrated that it's possible to roughly predict the emittance value of the *LCLS* Laser Injector (that was not given for granted by anyone) but on the other hand, we created a NN that is already employable to make emittance predictions: for some purposes, when using the *LCLS* Laser there's no need to know the very precise value of the emittance but just to know its value roughly and our NN does a really good job for this task.
By the way, we are convinced that there is the chance to make the predictions even better. For this reason we report here a table with all the information related to the best NN and its hyperparameter, hoping in this way to help the next that will try address this challenging problem.

Hyperparameter	Our choice	Description
Loss function	Mean squared error	This is just right, should not need any change
Activation function	tanh	This is just right, should not need any change
Optimizer	Adam	Here you could do a lot, we didn't try to optimize the Adam parameters. So just looking for better values would be good. We didn't tried any other kind of optimizer, so you could even replace the Adam with anything else.
Regularization	Dropuout	Here you could try to improve the dropout positioning (and magnitude) in the NN. As second step looking deeper at L1 and L2 would be useful too.
Number of Layers	3	This is maybe the most important thing to push further. Our grid search was really coarse grained, so there are huge possibilities to improve. You could stick to the grid search into the phase space of parameters or try something really cool like genetic algoritmh or Simulated Annealing optimizations and so on... In any case, be sure to define a good metric to compare the results of two different NNs.
Number of neurons	[100,50,25]	

Figure 12: *Here we summarize in a table the properties and hyperparameters of the best NN we identified. The first column contain the name of the property while the second is filled with the particular one we chose. The third one express how much work we did into the optimization of the corresponding property and, in our opinion, if it's worth to try to replace our best guess with a better one trying to give some advices on how to do it too. We really hope this could help the next person who wants to face this challenging problem.*

Conclusions

We created a Machine Leaning model able to roughly predict the emittance of the Laser Injector of the *LCLS* facility at SLAC. Considering the the hardness of the task and the limited resources in terms of data, the results we obtained are really satisfying. Moreover, the way in which we managed to boost the NN performance is cheering, the great of the contribution coming from the addition of more data. We were aware that we were handling with a huge and complicated dataset, having available a small set of noisy data and so we weren't expecting to be able to make really accurate predictions. By the way we can safely state that the results of our study went beyond our most optimistic expectations.
I am grateful to have had the chance to face a so challenging ML problem; one of the most important things I learnt working at SLAC, it's how does it looks like handling with a very complicated experimental machine and working in a big Team. I am really happy of this because it's one of the hardest thing to learn studying on books and can be acquired only with on-field experience, so I am grateful to the people who gave me the possibility to make this happen and to the *LCLS* Team who guided me through this challenging task. To the last goes an additional big thanks for everything they taught me: without their continuous, active and experienced feedback I wouldn't have been able to complete this work.