CS230

Accelerating computation of two-dimensional coherent synchrotron radiation using a neural network

River R. Robles
Department of Applied Physics
Stanford University
riverr@stanford.edu

Abstract

Particle accelerator physics is "big science". Building a new accelerator costs anywhere from tens of millions to one billion dollars. As such, it is critical to have accurate and efficient modeling tools in order to mitigate the risk associated with building a new accelerator. Coherent synchrotron radiation (CSR), the interaction of a beam with itself as it moves through a bending magnet, is by far one of the most difficult physical phenomena to model, and even the simplest simulation approaches increase total accelerator simulation times by as much as an order of magnitude. We have designed, trained, and tested a neural network capable of simulating the CSR effect in two dimensions. By comparing against full CSR simulations of the final bend magnet of the FACET-II accelerator, we demonstrate that our model not only accurately predicts CSR forces at one instant in time, but that compounding errors from many iterations of the model are still small. We show that across a broad parameter range, our model can predict critical accelerator metrics such as beam emittance with a few percent accuracy in negligible simulation time.

1 Introduction

Particle accelerators have enabled much of the innovation in modern physics, chemistry, and biology [1, 2]. Particle accelerator physics on the scale relevant to scientific discovery machines is "big science", meaning that building a new accelerator is extremely costly. Price tags easily range from tens of millions [3], for smaller machines, to one billion dollars for the largest machines, while taking up as much as several kilometers of ground space. As such, accurate numerical modeling in the design stage is essential to mitigate the risks associated with a potential project. Coherent synchrotron radiation (CSR), the interaction of a particle beam with itself as it traverses a curved trajectory, is one of the most difficult accelerator phenomena to model, and is usually only simulated with a one-dimensional (1D) model that neglects the three-dimensional (3D) nature of particle beams [4, 5, 6]. This choice is made because even 1D regime CSR simulations are computationally expensive, easily increasing the duration of a simulation by an order of magnitude. Recent machine designs have begun to reach regimes where the 1D models no longer apply, demanding even more computationally demanding two or three-dimensional calculations that are completely impractical to incorporate into start-to-end accelerator modeling [7, 8].

CSR effects are primarily of concern when beams travel through bending magnets called dipoles. In simulation, CSR is dealt with in a kick-step-kick model. First, the beam is propagated through a short section of the magnet. After this short propagation, the forces due to CSR are calculated given the state of the beam at that moment and applied to the beam at the same moment. The beam then propagates through another short section of the magnet before this process repeats until the magnet

CS230: Deep Learning, Fall 2022, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

end is reached. In the 1D regime CSR produces a force $W_z(z)$ in the direction of the beam's motion, which is a function of the beam density profile (z) and its distance into the magnet s. z here is a coordinate along the length of the beam. In the 2D regime the problem is more complicated, as there are now two dimensions to consider: the original z and a transverse dimension x. Not only are there now two forces $W_x(x;z)$ and $W_z(x;z)$, but they are functions of both coordinates, and they must be calculated from the two-dimensional beam density $(x;z)$ and again the distance into the magnet s. The 3D regime similarly adds the final spatial dimension for maximal complexity.

Recently, SLAC researchers trained a neural network to compute the longitudinal CSR force $W_z(z)$ given a density profile (z) and distance s in the one-dimensional regime, showing that they could faithfully reproduce the quantitative physics with negligible computation time [9]. This is progress, however as we noted before many machines are moving into regimes where minimally 2D effects must be considered to model the correct physics. Extending on their work, we have trained a neural network to calculate the CSR forces $W_x(x;z)$ and $W_z(x;z)$ in the 2D regime. Inherenty, the 2D CSR problem is one of taking an image – the beam density $(x;y)$ on a grid – and a scalar value – the distance into the magnet s – and predicting two images – the forces $W_x(x;z)$ and $W_z(x;z)$ on the same grid as the change density. We applied our neural network to numerical studies of the final dipole magnet in the FACET-II accelerator at SLAC National Accelerator Laboratory [10], and found excellent agreement in critical metrics between full 2D CSR simulations and our neural network. Furthermore, the efficient neural network implementation made it such that the CSR calculation had a negligible impact on the simulation durations, where for the standard code it was by far the most expensive part of the simulations.

2 Related work

Traditionally, CSR is calculated using numerical implementations of analytical formulas determining the forces. In practice these formulas are represented as multi-dimensional integrals. Early approaches were especially time consuming, such as the code CSRTrack [11], which takes many hours to run on multiple CPUs for most problems. Recent theoretical advancements in the past several years have yielded greatly simplified expressions for the forces which are more efficient to implement numerically, which has reduced standard simulation times from many hours to roughly 1 hour [12, 13, 14]. Even further success has been had by taking advantage of modern computing languages and hardware, such as the fast Julia code and GPUs. One of the fastest 2D CSR codes to date is JuliaCSR2D [15], which can calculate the CSR forces for given charge density and s with high accuracy in about one minute. Thus a full simulation takes roughly half an hour on a GPU.

Very little work has been done in implementing CSR with neural networks. Only one group has attempted this, and only for 1D CSR [9]. They had a great deal of success in this effort. To my knowledge no one has attempted a neural network approach to higher dimensional CSR.

3 Dataset and Features

Our dataset is made up of simulations of the FACET-II final dipole magnet. We used one of the fastest available 2D CSR codes, JuliaCSR2D, which runs on a GPU and can generate a single set of training data – force maps $W_x(x;z)$ and $W_z(x;z)$ for a given charge density map $(x;z)$ and distance s – in about 60 seconds. To ensure the dataset is sampled from the kinds of distributions we'd like to apply it to, the initial beam we consider was generated by a full simulation of the beamline leading up to the dipole. We simulated the dipole with CSR in 20 steps such that each simulation generated 20 datasets. To generate further data, we scaled the input beam dimensions in x and z at the start of the dipole by factors s_x and s_z ranging from 0.1 to as much as 10. We note that this is only at the entrance, the beam distribution at the subsequent 19 simulation points will vary dramatically due to the beam's dynamics in the bend affected by 2D CSR. In total we generated 2000 such datasets, which to be explicit were each made up of a three 201x101 single channel images for the charge density and force maps W_z and W_x as well as the scalar distance into the magnet s. We note that although this dataset size is quite small by the standards of more common machine learning applications, there is precedent in accelerator physics applications for not needing much data for the networks to learn complex features [16, 17, 18]. Due to the small amount of data we only split the data into two groups of size 1990 for training and 10 for validation and testing. We show a sample of images from 5 datasets in Figure 1.

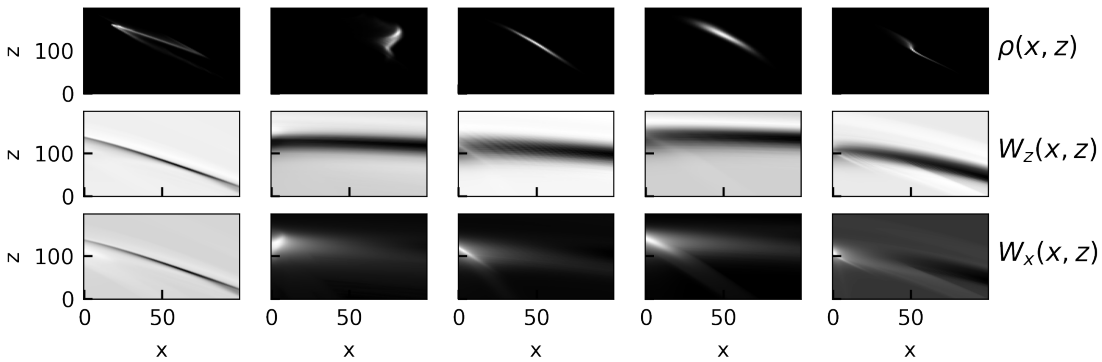

Figure 1: A set of characteristic example images from the dataset.

We then went through several steps of data preprocessing. First, in convolution networks it is much more convenient to work with images of dimensions which are powers of 2, so we resampled the images to be $256 \quad 128$ using built-in tensorflow functions. Furthermore, although all images had the same dimensions their pixels did not represent the same physical spacing. JuliaCSR2D does this so that the beam distributions are always well-resolved. To account for this we added the pixel spacing in each dimension to our scalar dataset: for x in microns and for z in nanometers, as these are the characteristic lengthscales for each dimension. We refer to the pixel widths as dx and dz. In general we did our best to avoid giving the NN very large or very small values. As such we also normalized the charge density images to their maximum value, then further provided these scale values normalized to the maximum of them as an additional scalar input to the model, since the total charge in the beam is also an important value in determining the forces. We refer to these normalized maximal charge densities by $\quad_{max}$. Finally we normalized the force images to the maximum value in the training dataset. This global scale factor was recorded so that the output fields could be properly rescaled after a prediction was made.

4 Methods

Fundamentally our problem is one of image prediction given both image and scalar inputs. Furthermore, the calculation of the 2D CSR wake is inherently a 2D convolution. As such, we decided to base our network design on the concept of autoencoders [19]. There are three basic parts as illustrated in a basic sense in Figure 2. First, we send the input charge density image through what we call the encoder, which consists of a convolutional neural network made up of sequences of (convolution layer, batch normalization, max pooling) which gradually reduces the image size until finally it goes through a single dense layer generating a low-dimensional representation of the original image. This low-dimensional vector representation x is subsequently concatenated with the rest of the scalar inputs s, dx, dz, and $\quad_{max}$. This new vector is then fed into a fully connected network made up of a sequence of dense layers. The final dense layer has $2^m \quad 2^{m+1}$ nodes such that its output can be reshaped into a $2^m \quad 2^{m+1}$ image, where m is an integer hyperparameter. This new low-dimensional image is fed into a final convolutional neural network which now uses sequences of (transpose convolution layer, batch normalization, upsampling) to gradually increase the dimensions back up to 256x128. The final layer of the network is a two-channel convolution layer with a linear activation, with output representing the predicted images $\hat{W}_z$ and $\hat{W}_x$. For a loss function we experimented with squared error $L(y; \hat{y}) = |y|^2 \quad |\hat{y}|^2|$ and absolute error $L(y; \hat{y}) = |y \quad \hat{y}|$. We found that absolute error generally yielded better results.

5 Experiments/Results/Discussion

While training the model we went through a series of hyperparameter and model architecture tuning steps before finding one that worked well. During training we settled on the Adam optimizer with an initial learning rate of 0.01 and a minibatch size of 50. Using keras callback functions we set the learning rate to decrease by a factor of 2 on plateaus with a patience of 50 epochs. We trained the

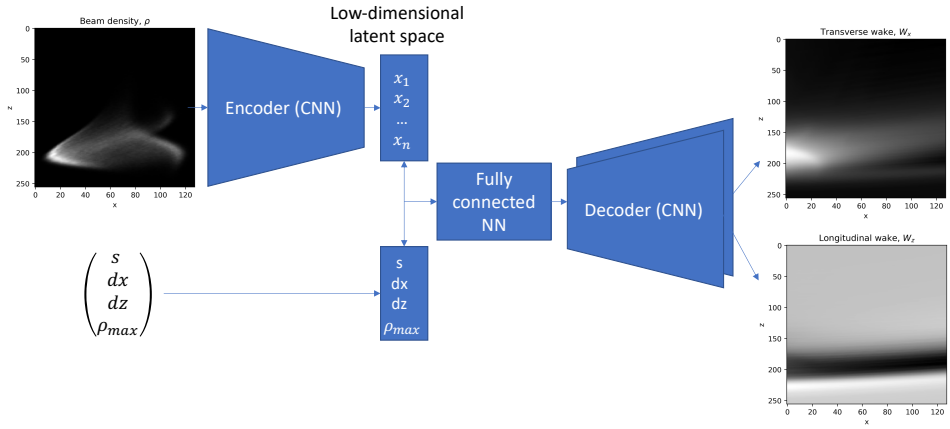

Figure 2: Basic summary of the different elements of the chosen network architecture.

model for a total of 1000 epochs. Furthermore, we set all activation functions to rectified linear units except for the final layer which had a linear activation function to properly sample possible outputs. Early on we decided to fix the convolution and transpose convolution layers to have strides of 1 with 'same' padding, prefering to control the image size evolution in the encoder and decoder layers using MaxPooling and UpSampling layers. Further, we fixed the MaxPooling and UpSampling layers to 2 2, such that each sequence (convolution, batch normalization, max pooling) reduced the image size by a factor of two in each dimension in the encoder and conversely increased them by a factor of two in the decoder. The key remaining hyperparameters to consider were the length of the encoding vector x, the number of channels and kernel sizes in each convolution and transpose convolution layer, the number of dense layers and the number of nodes in each in the intermediate fully connected layer, and finally the starting decoder image dimension parameter m. Although further improvements may be possible, we found very good performance with the set of hyperparameters summarized in Figure 3. Note that in addition to the simple summary in the Figure 2, we ended up extending the convolution and transpose convolution layers to two different kernel sizes, one 3 3 to account for fine features and one 7 7 to account for broader features.

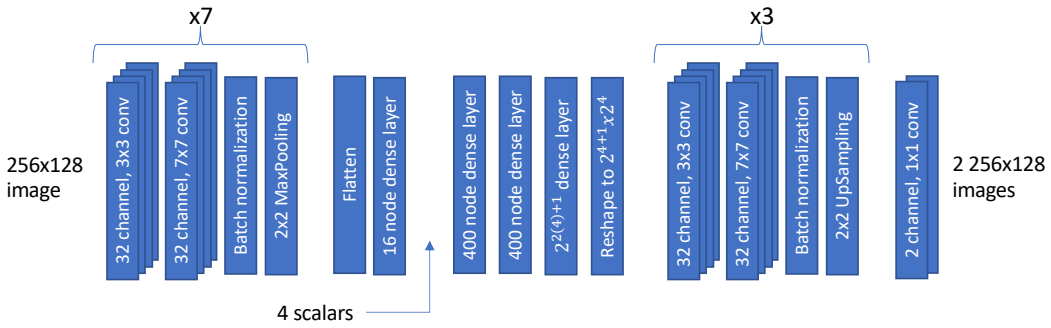

Figure 3: Final network configuration with associated hyperparameters.

The most important metric for the success of the model is not necessarily our cost function, but rather its ability to predict the output of full simulations. This is because it is important to evaluate the effect of compounding errors: as the beam gets further into the dipole magnet, any error in the CSR prediction up to that point will only increase further errors downstream. Thus to evaluate the utility and accuracy of our model we incorporated it into a full simulation package of a beam propagating in a dipole magnet, essentially simply replacing the CSR calculation step of JuliaCSR2D with a call to our network. The time to evaluate our network in these simulations was negligible compared to the 60+ seconds required to calculate the CSR forces in the original code. We compared our simulation outputs to the same simulation configurations used in our dataset. Notice however that this does not mean we simply re-evaluated the model at the same images. Although the first image of, say, the